SISTEM PAKAR
(EXPERT SYSTEM)

Heri Suprapto

ARTIFICIAL INTELLIGENCE

Definisi Kecerdasan Buatan

- **H. A. Simon** [1987] :

  " *Kecerdasan buatan (artificial intelligence) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang -dalam pandangan manusia adalah- cerdas*"

- **Rich and Knight** [1991]:

  "*Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia.*"

Definisi Kecerdasan Buatan

Encyclopedia Britannica:

"*Kecerdasan Buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan*"

Tujuan dari kecerdasan buatan

1. **Membuat mesin** menjadi **lebih pintar** (tujuan utama)
2. **Memahami** apa itu **kecerdasan** (tujuan ilmiah)
3. **Membuat mesin lebih bermanfaat** (tujuan *entrepreneurial*)

Arah AI

- Mengembangkan metode dan sistem untuk menyelesaikan masalah AI tanpa mengikuti cara manusia menyelesaikannya (sistem pakar / expert systems)
- Mengembangkan metode dan sistem untuk menyelesaikan masalah AI melalui pemodelan cara berpikirnya manusia, atau cara bekerjanya otak manusia (neural networks).

AI dipandang dalam berbagai perspektif.

- Dari perspektif **Kecerdasan (Intelligence)**

  AI adalah bagaimana membuat mesin yang "cerdas" dan dapat melakukan hal-hal yang sebelumnya dapat dilakukan oleh manusia

- Dari perspektif **bisnis**

  AI adalah sekelompok alat bantu (*tools*) yang berdaya guna, dan metodologi yang menggunakan *tool-tool* tersebut guna menyelesaikan masalah-masalah bisnis.

- Dari perspektif **pemrograman** (*Programming*),

  AI termasuk didalamnya adalah studi tentang **pemrograman simbolik**, **pemecahan masalah**, **proses pencarian** (*search*)

Kecerdasan Buatan/Artificial Intelligent

- AI merupakan salah satu bagian ilmu komputer yang membuat agar mesin/komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia.

- Agar dapat berlaku seperti manusia, maka komputer harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar.

What is AI ?

Sistem yang berpikir seperti manusia ***Thinking humanly***	Sistem yang berpikir secara rasional ***Thinking rationally***
Sistem yang bertindak seperti manusia ***Acting humanly***	Sistem yang bertindak secara rasional ***Acting rationally***

Berfikir Seperti Manusia

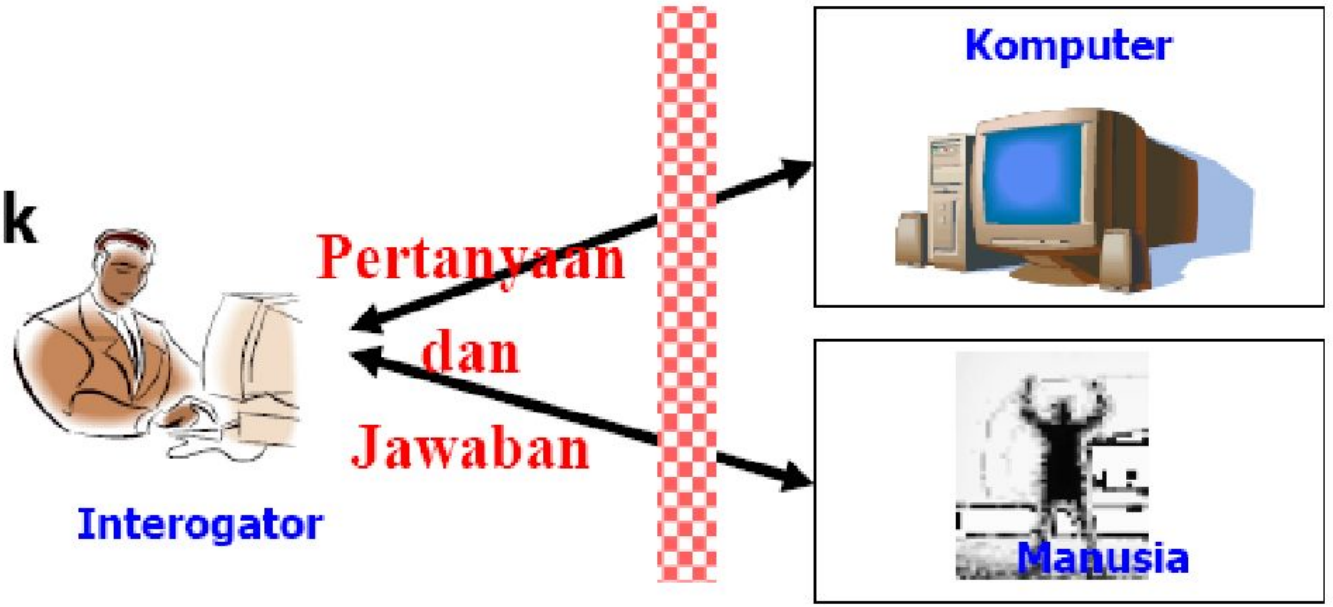
1. Diperlukan suatu cara untuk mengetahui bagaimana manusia berfikir
2. Diperlukan pemahaman tentang bagaimana pikiran manusia bekerja

Bagaimana caranya?

1. Melalui introspeksi atau mawas diri; mencoba menangkap bagaimana pikiran kita berjalan
2. Melalui percobaan psikologis

Uji Turing Dari AI Bertindak Seperti Manusia

- **AI lulus test apabila interogator tidak bisa membedakan dialog mana yang dilakukan dengan komputer dan mana yang dilakukan dengan manusia**



Berfikir Rasional

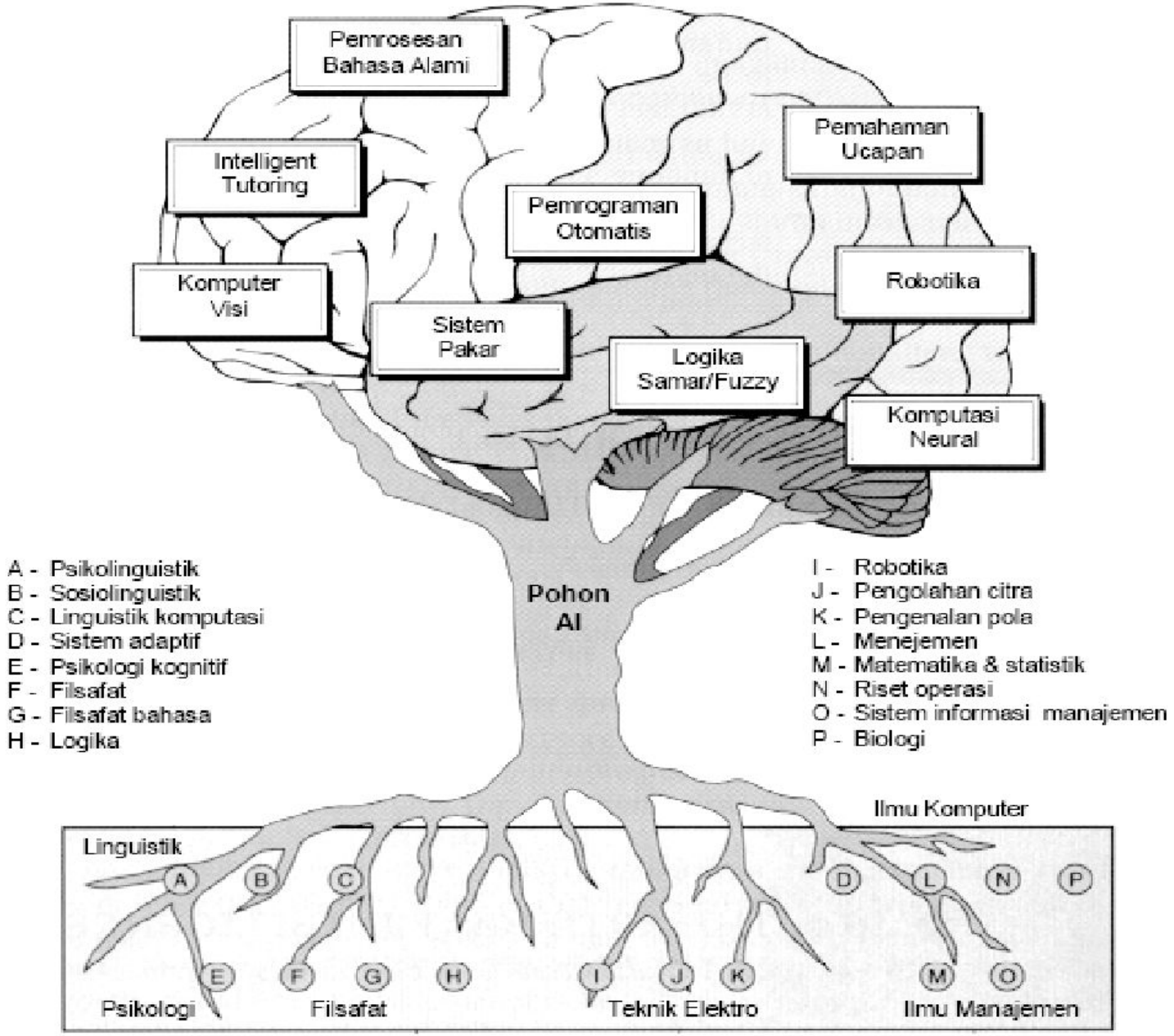
- Cara berfikirnya memenuhi aturan logika yang dibangun oleh Aristotles
  - Pola struktur argumentasi yang selalu memberi konklusi yang benar bila premis benar
  - Menjadi dasar bidang logika
- Tradisi logicist dalam AI adalah membangun program yang menghasilkan solusi berdasarkan logika
- Problem:
  - Pengetahuan informal sukar diuraikan dan dinyatakan dalam bentuk notasi logika formal
  - Penyelesaian secara prinsip vs. praktis

Bertindak Rasional

- Bertindak secara rasional artinya bertindak didalam upaya mencapai goal

- Didalam lingkungan yang rumit tidaklah mungkin mendapatkan rasionalisasi sempurna yang selalu melakukan sesuatu dengan benar
  - Rasionalisasi terbatas

Pohon Kecerdasan Buatan dan aplikasi utamanya



Perbedaan antara Pemrograman AI dan Konvensional

AI	Komputasi Konvensional
Representasi dan Manipulasi simbol	Algoritama
Memberitahu komputer tentang suatu masalah	Memerintah komputer untuk menyelesaikan masalah
Komputer diberi pengetahuan dan kemampuan inferensi	Memberi data kepada komputer dan program

Kelebihan kecerdasan buatan

- Lebih bersifat permanen.
- Lebih mudah diduplikasi & disebarkan.
- Lebih murah.
- Bersifat konsisten dan teliti karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah
- Dapat didokumentasi.Keputusan yang dibuat komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
- Dapat mengerjakan beberapa task lebih cepat dan lebih baik dibanding manusia

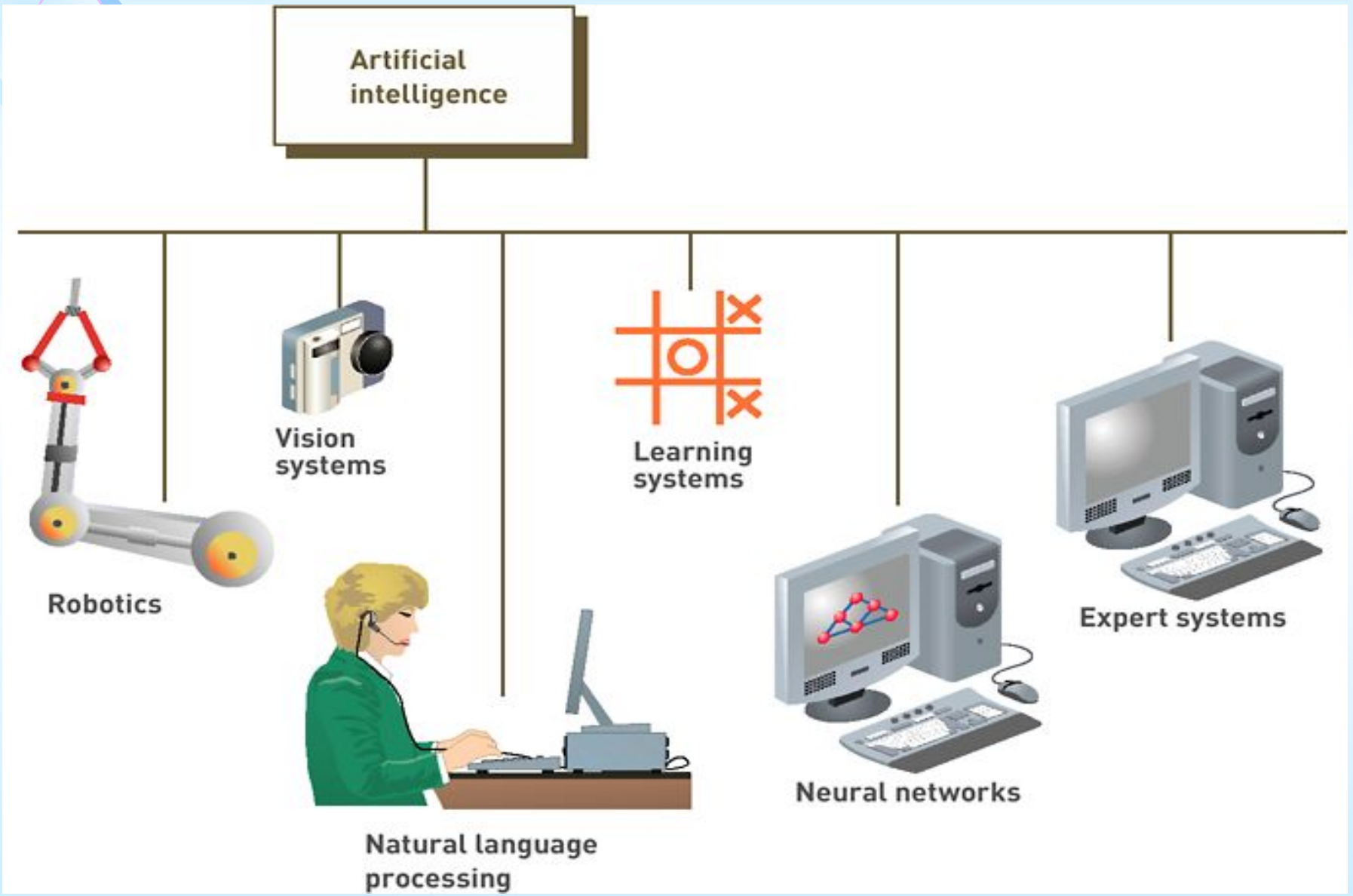
Kelebihan kecerdasan alami

1. **Kreatif** : manusia memiliki kemampuan untuk menambah pengetahuan, sedangkan pada kecerdasan buatan untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.
2. Memungkinkan orang untuk menggunakan pengalaman atau pembelajaran secara langsung. Sedangkan pada kecerdasan buatan harus mendapat masukan berupa input-input simbolik.
3. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan sangat terbatas.

Kecerdasan Buatan dan Kecerdasan Alami

Kecerdasan Alami	Kecerdasan Buatan
✔ Kreatif ✔ Memungkinkan penggunaan pengalaman secara langsung ✔ Dapat dipergunakan secara luas	✔ Lebih bersifat permanen ✔ Mudah diduplikasi dan disebarkan ✔ Lebih murah ✔ Lebih konsisten ✔ Dapat didokumentasikan ✔ Lebih cepat

Cabang Dari AI



- **Pengertian Kecerdasan Buatan (Artificial Intelligence)**

  Adalah kegiatan yang diberikan kpd mesin seperti komputer yaitu kemampuan untuk menampilkan perilaku yg dianggap cerdas yg diamati pd manusia.

  Sistem pakar adalah program komputer yg mencoba utk mewakili pengetahuan dari pakar manusia dlm bentuk heuristic.

AI mencakup kerja dlm area berikut :

- **Jaringan saraf (neural network)** spt kemampuan belajar, generalisasi, dan abstraksi)
- **Sistem persepsi**, menggunakan citra visual dan sinyal suara utk menginstruksikan komputer atau alat lain misalnya robot.
- **Belajar**, meliputi semua kgtn yg memingkinkan komputer atau alat lain utk memperoleh pengetahuan sbg tambahan dr apa yg tlh dimasukkan ke dlm memori oleh pembuatnya atau pemrogramnya.
- **Robotik**, terdiri dr alat yg dikendalikan komputer yg meniru aktivitas gerak manusia.
- **Hardware AI**, mencakup alat fisik yg membantu aplikasi AI.
- **Pemrosesan bahasa alamiah**, memungkinkan pemakai utk berkomunikasi dgn komputer dlm bbgi bahasa & memungkinkan komputer memeriksa ejaan & tata bhs.

Sistem pakar dan jaringan saraf memiliki potensi terbesar utk digunakan dlm memecahkan masalah bisnis. Keduanya mrp contoh sistem berbasis pengetahuan

SISTEM PAKAR
(EXPERT SYSTEM)

Sistem Pakar

- Sistem Pakar berasal dari istilah *knowledge-based expert system*
- Menggunakan *human knowledge* yang dimasukkan ke dalam komputer untuk memecahkan masalah yang umumnya memerlukan keahlian seorang Pakar
- Domain yang sempit

SISTEM PAKAR (EXPERT SYSTEM)

- Sebuah program komputer yang dirancang untuk memodelkan kemampuan menyelesaikan masalah seperti layaknya seorang pakar (*human expert*).
- Sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli/pakar.
- Sehingga orang awampun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan para ahli/pakar

SISTEM PAKAR (EXPERT SYSTEM) lanjutan…

- Pada dasarnya kepakaran ditransfer dari seorang pakar ke komputer, pengetahuan yang ada disimpan dalam komputer, dan pengguna dapat berkonsultasi pada komputer itu untuk suatu nasehat, lalu komputer dapat mengambil inferensi (menyimpulkan, mendeduksi, dll.) seperti layaknya seorang pakar, kemudian menjelaskannya ke pengguna tersebut, bila perlu dengan alasan-alasannya.

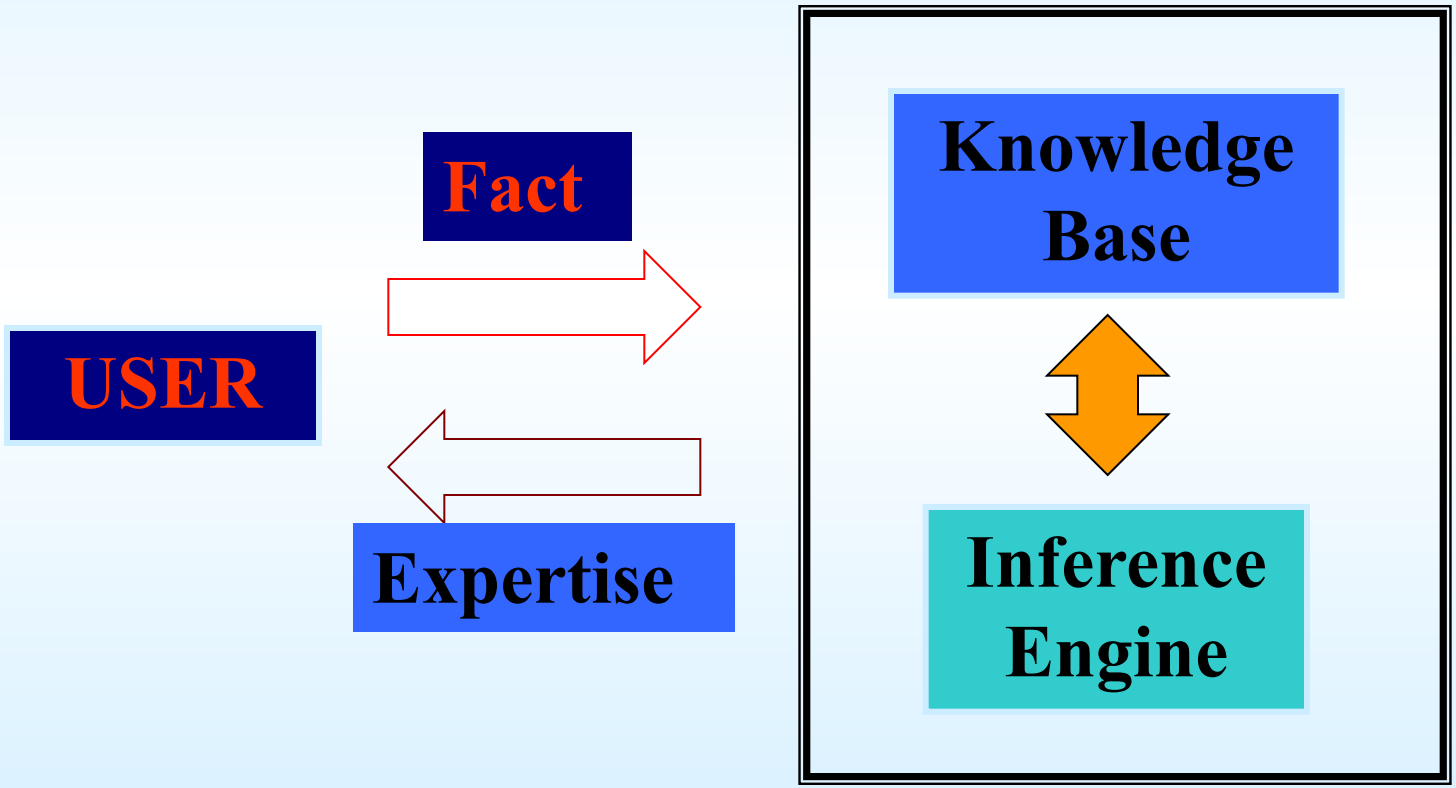
Alasan mendasar mengapa ES dikembangkan untuk menggantikan seorang pakar:

- ✔ Dapat menyediakan kepakaran setiap waktu dan diberbagai lokasi
- ✔ Secara otomatis mengerjakan tugas-tugas rutin yang membutuhkan seorang pakar.
- ✔ Seorang Pakar akan pensiun atau pergi
- ✔ Seorang Pakar adalah mahal
- ✔ Kepakaran dibutuhkan juga pada lingkungan yang tidak bersahabat (*hostile environtment*)

Keuntungan dan Kelemahan Sistem Pakar

Keuntungan	Kelemahan
✔ Bisa melakukan pekerjaan secara berulang ✔ Meningkatkan output dan kualitas ✔ Meningkatkan kapabilitas komputer ✔ Memiliki kemampuan untuk mengakses pengetahuan ✔ Menyimpan keahlian dan pengetahuan pakar	✔ Biaya pembuatan dan pemeliharaan mahal ✔ Sulit dikembangkan ✔ Akurasinya belum 100 %

Konsep Dasar dan Fungsi Sistem Pakar



Konsep Dasar SistemPakar

- Menurut Efrain Turban, Konsep dasar sistem pakar mengandung:
  - Ahli
  - Keahlian
  - Pengalihan keahlian
  - Inferensi
  - Aturan dan kemampuan menjelaskan

APA ITU PAKAR/AHLI (EXPERT) ?

- Seorang pakar/ahli (*human expert*) adalah seorang individu yang memiliki kemampuan pemahaman yang superior dari suatu masalah. Misalnya: seorang dokter, penasehat keuangan, pakar mesin mobil, dll.

- Kemampuan kepakaran:
  - ✔ Dapat mengenali (*recognizing*) dan merumuskan masalah
  - ✔ Menyelesaikan masalah dengan cepat dan tepat
  - ✔ Menjelaskan solusi
  - ✔ Belajar dari pengalaman
  - ✔ Restrukturisasi pengetahuan
  - ✔ Menentukan relevansi/hubungan
  - ✔ Memahami batas kemampuan

APA ITU KEPAKARAN /KEAHLIAN

- Suatu kelebihan penguasaan pengetahuan dibidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman
- Jenis-jenis pengetahuan yang dimiliki dalam kepakaran:
  - ✔ Teori-teori dari permasalahan
  - ✔ Aturan dan prosedur yang mengacu pada area permasalahan
  - ✔ Aturan (heuristik) yang harus dikerj akan pada situasi yang terjadi
  - ✔ Strategi global untuk menyelesaikan berbagai jenis masalah
  - ✔ *Meta-knowledge* (pengetahuan tentang pengetahuan)
  - ✔ Fakta-fakta

*APA ITU PENGETAHUAN (*KNOWLEDGE*) ?*

- *Data* + *processing* = *information*
- *Information* + *processing* (pengalaman, training, dll) = *knowledge*

Perbandingan Seorang Ahli (Human Expert) dengan Sistem Pakar (ES)

Faktor	**Human Expert**	**Expert System**
Time Availability	Hari Kerja	Setiap saat
Geografis	Lokal/tertentu	Dimana saja
Keamanan	Tidak tergantikan	Dapat diganti
Perishable/Dapat habis	Ya	Tidak
Performansi	Variabel	Konsisten
Kecepatan	variabel	Konsisten & lebih cepat
Biaya	Tinggi	Terjangkau

PEMINDAHAN KEPAKARAN

- Tujuan dari sebuah sistem pakar adalah untuk mentransfer kepakaran yang dimiliki seorang pakar kedalam komputer, dan kemudian kepada orang lain (*nonexpert*).
- Aktifitas yang dilakukan untuk memindahkan kepakaran:
  - *Knowledge Acquisition* (dari pakar atau sumber lainnya)
  - *Knowledge Representation* (ke dalam komputer)
  - *Knowledge Inferencing*
  - *Knowledge Transfering*
- Pengetahuan yang disimpan dikomputer disebut basis pengetahuan, yaitu fakta dan prosedur

Inferensi

- Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan menalar.
- Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basis data, maka komputer harus diprogram untum membuat inferensi.
- Proses inferensi ini dikemas dalam bentuk motor inferensi/inference Engine

Aturan dan kemampuan menjelaskan

- Sebagian besar sistem pakar dibuat dengan bentuk rule based system, dimana pengetahuan disimpan dalam bentuk aturan-aturan, dan biasanya berbentuk IF – THEN
- Fitur lainya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan ini yang membedakan sistem pakar dengan sistem konvensional

Perbandingan Sistem Konvensional dan Sistem Pakar

Sistem Konvensional	**Sistem Pakar**
Informasi dan pemrosesan umumnya digabung dlm satu program sequential	*Knowledge base* terpisah dari mekanisme pemrosesan (*inference*)
Program tidak pernah salah (kecuali programer-nya yang salah)	Program bisa saja melakukan kesalahan
Tidak menjelaskan mengapa input dibutuhkan atau bagaimana hasil diperoleh	Penjelasan (*explanation*) merupakan bagian dari ES
Membutuhkan semua input data	Tidak harus mambutuhkan semua input data atau fakta
Perubahan pada program merepotkan	Perubahan pada *rules* dapat dilakukandengan mudah
Sistem bekerja jika sudah lengkap	Sistem dapat bekerja hanya dengan *rules* yang sedikit
Eksekusi secara algoritmik (*step-by-step*)	Eksekusi dilakukan secara heuristic dan logik

Sistem Konvensional	**Sistem Pakar**
Manipulasi efektif pada database yang besar	Manipulasi efektif pada knowledge-base yang besar
Efisiensi adalah tujuan utama	Efektifitas adalah tujuan utama
Data kuantitatif	Data kualitatif
Representasi data dalam numerik	Reperesentasi pengetahuan dalam simbol
Menangkap, menambah dan mendistribusi data numerik atau informasi	Menangkap, menambah dan mendistribusi pertimbangan (*judgment*) dan pengetahuan

Bentuk Sistem Pakar

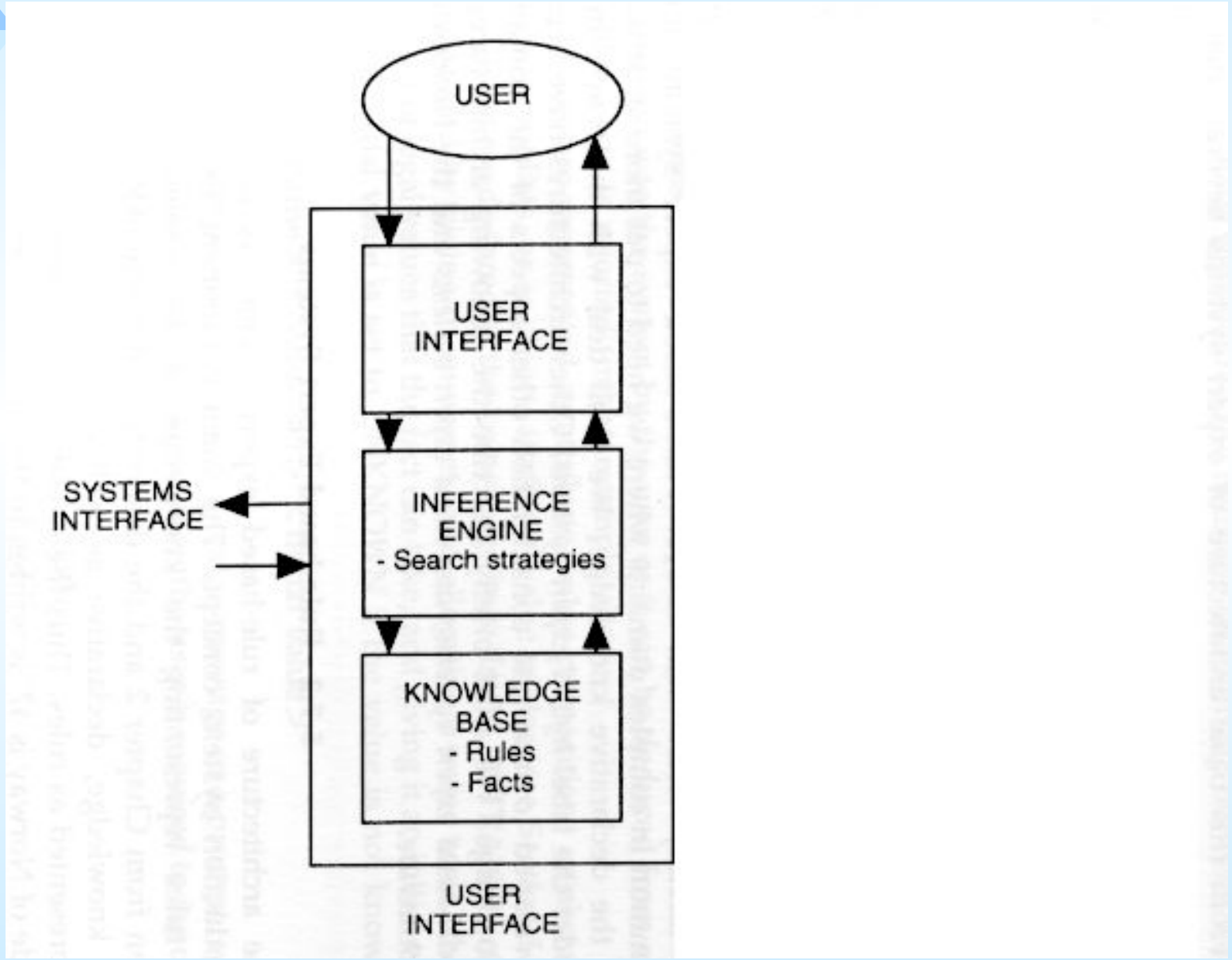
- Ada 4 bentuk sistem pakar, yaitu:

1. Berdiri sendiri, yaitu sistem pakar yang berupa software yang berdiri sendiri.
2. Tergabung, sistem pakar ini merupakan bagian program yang terkandung di dalam suatu algoritma.
3. Menghubungkan ke software lain, yaitu sistem pakar yang menghubugkan ke suatu paket program tertentu,misal DBMS
4. Sistem mengabdi, yaitu sistem pakar yang erupakan bagian dari komputer khusus yang dihubungkan dengan fungsi tertentu, misalnya untuk membantu menganalisis radar.

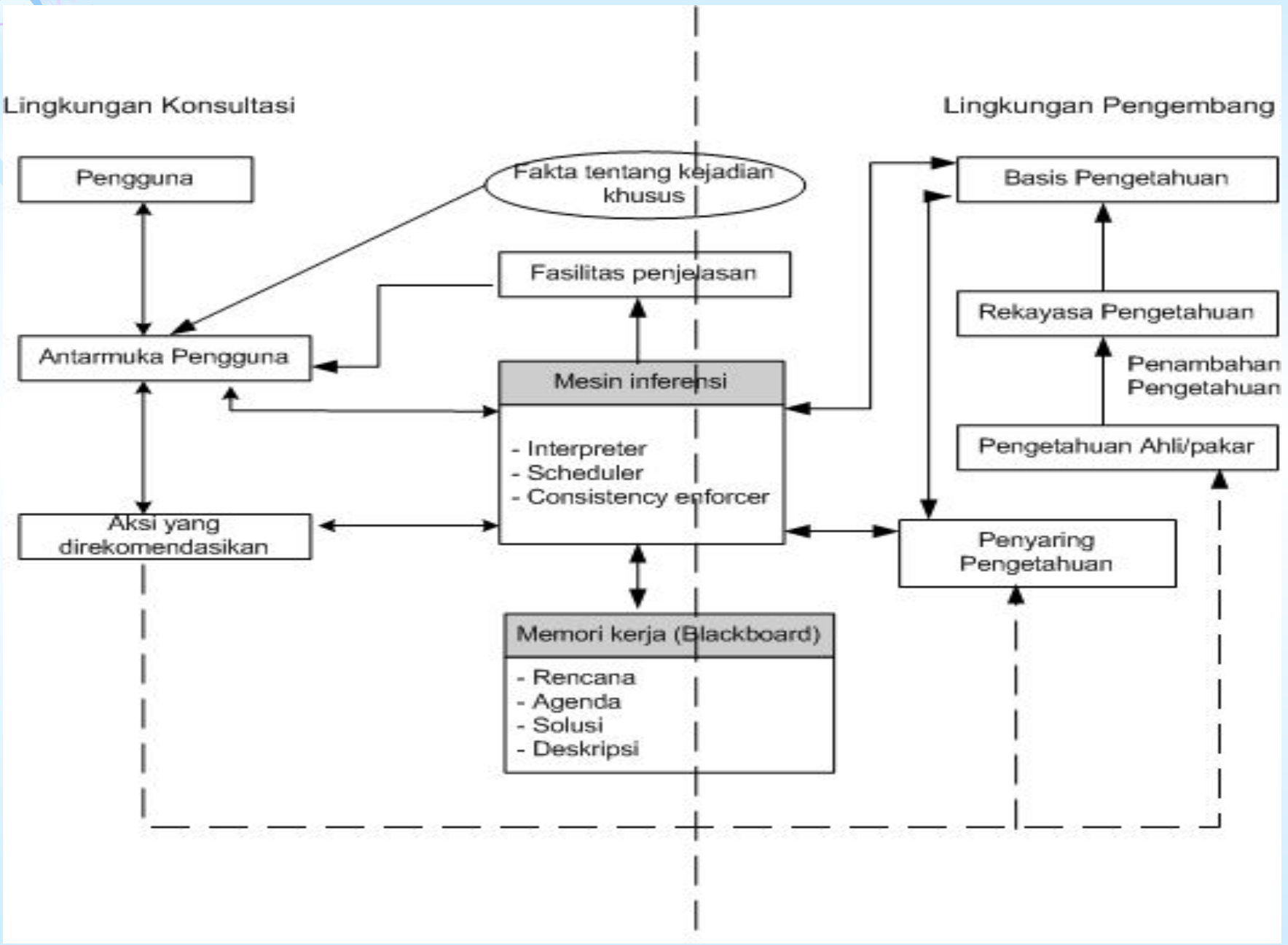
Struktur Sistem Pakar

- Sistem pakar terdiri dari dua bagian pokok,yaitu:
  - Lingkungan pengembangan, yang dipergunakan sebagai pebangunan sistem pakar baik dari segi pembangunan komponen maupun basis pengetahuan
  - Lingkungan konsultasi, yang dipergunakan oleh seseorang yang bukan ahli untuk berkonsultasi

Arsitektur Dasar Sistem Pakar



Struktur Sistem Pakar



Komponen-Komponen Sistem Pakar

1. Subsistem penambahan pengetahuan:

   Bagian ini digunakan untuk memasukan pengetahuan, mengkonstruksi atau memperluas pengetahuan dalam basis pengetahuan.

2. Basis Pengetahuan:

   berisi pengetahuan yang dibutuhkan untuk memahami, memformulasi, dan memecahkan masalah. Basis pengetahuan tersusun atas 2 elemen dasar:

   - ✔ Fakta, misalnya: situasi, kondisi, dan kenyataan dari permasalahan yang ada, serta teori dalam bidang itu
   - ✔ Aturan, yang mengarahkan penggunaan pengetahuan untuk memecahkan masalah yang spesifik dalam bidang yang khusus

- Ada dua bentuk pendekatan basis pengetahuan yang umum dipergunakan, yaitu:
  - Penalaran Berbasis Aturan (Rule-Based Reasoning)
    - Pengetahuan direpresentasikan dengan menggunkan aturan dalam bentuk IF – THEN.
    - Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan dapat diselesaikan secara berurutan.
  - Penalaran Berbasis Kasus (Case-Based Reasioning)
    - Basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan duturunkan suatu solusi untuk keadaan yang terjadi sekarang .
    - Bentuk ini dipergunakan apabila user menginginkan untuk tahu lebih banyak pada kasus-kasus yang mirip.

3. Mesin Inferensi (Inference Engine),

- **Mesin Inferensi**
  - merupakan otak dari Sistem Pakar. Juga dikenal sebagai penerjemah aturan (rule interpreter). Komponen ini berupa program komputer yang menyediakan suatu metodologi untuk memikirkan (reasoning) dan memformulasi kesimpulan. Kerja mesin inferensi meliputi:
    - Menentukan aturan mana akan dipakai
    - Menyajikan pertanyaan kepada pemakai, ketika diperlukan.
    - Menambahkan jawaban ke dalam memori Sistem Pakar.
    - Menyimpulkan fakta baru dari sebuah aturan
    - Menambahkan fakta tadi ke dalam memori
- Ada dua cara dalam melakukan inferensi,yaitu:
  - Penalaran Maju (Forward Chaining)
  - Penalaran Mundur (Backward Chaining)

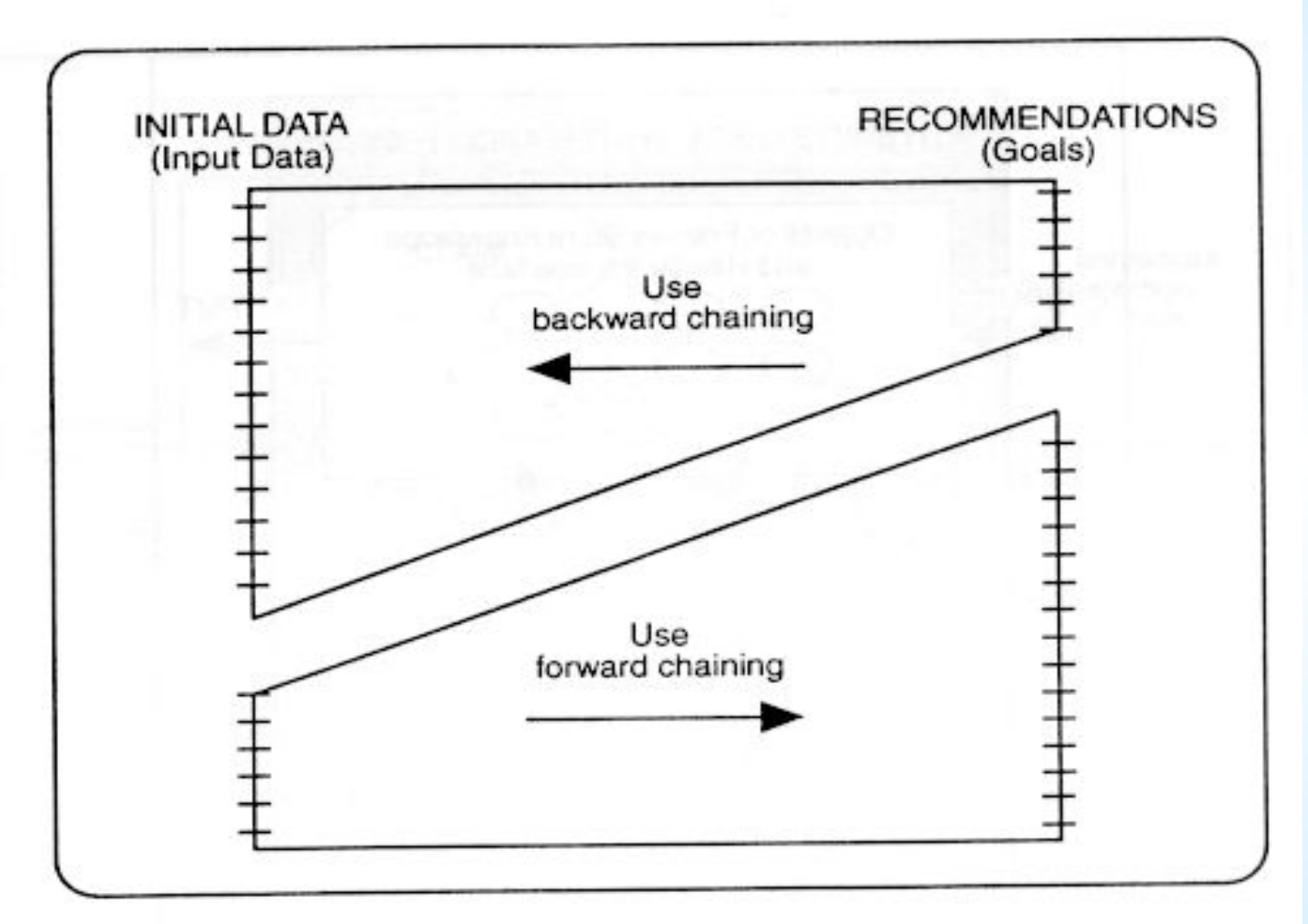
Membandingkan Penalaran Maju & Penalaran Mundur

Penalaran mundur bergerak lebih cepat dari penalaran maju krn penalaran mundur tdk harus mempertimbangkan semua aturan & tdk membuat beberapa putaran melalui perangkat atauran.

Penalaran mundur sangat sesuai jika:

1. Terdapat variabel sasaran berganda (multiple goal variable)
2. Terdapat banyak aturan
3. Semua atau hampir semua aturan tdk hrs diuji dalm proses mencapi pemecahan.

– Choosing between backward and forward chaining.

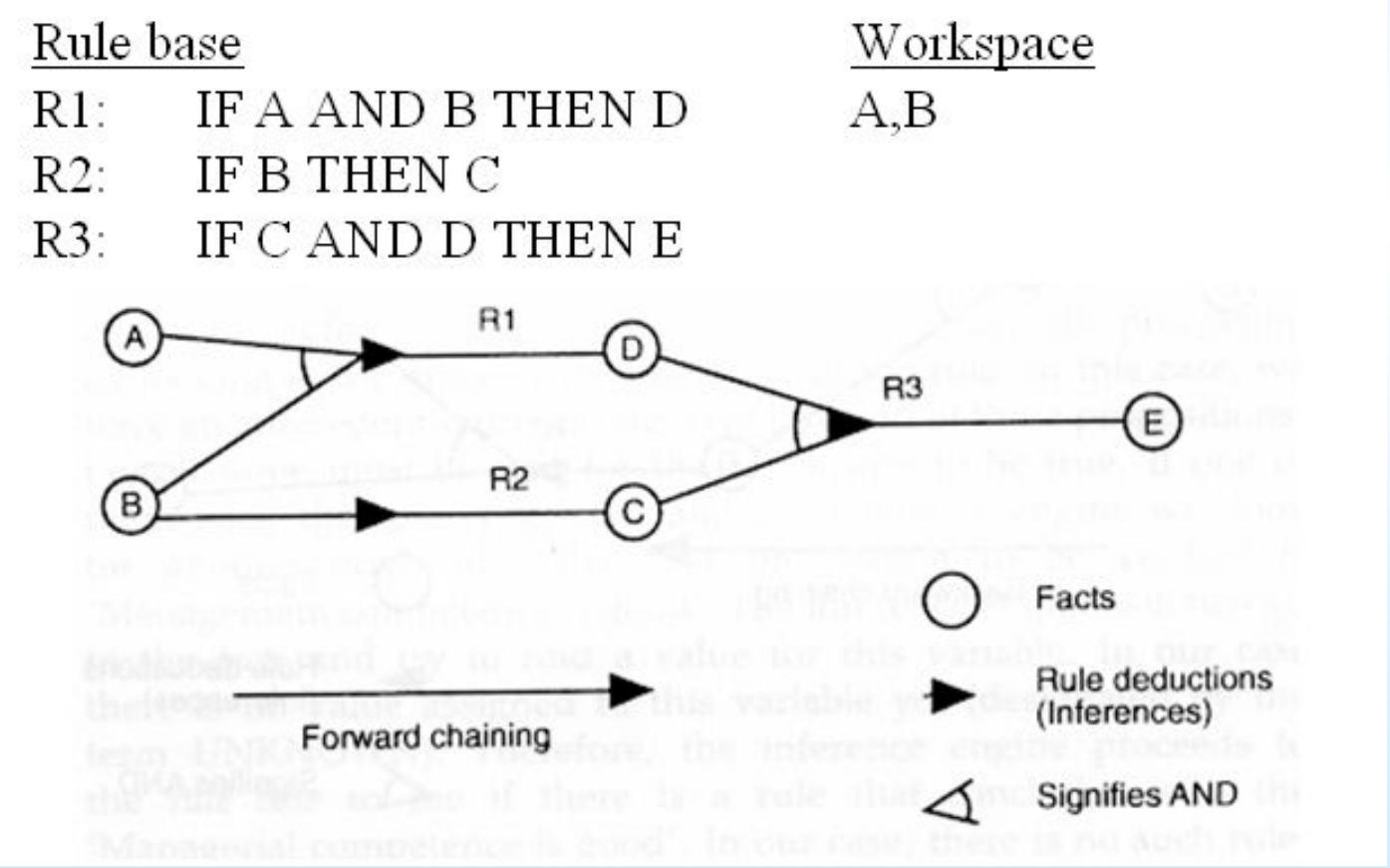


Forward Reasoning (forward chaining)

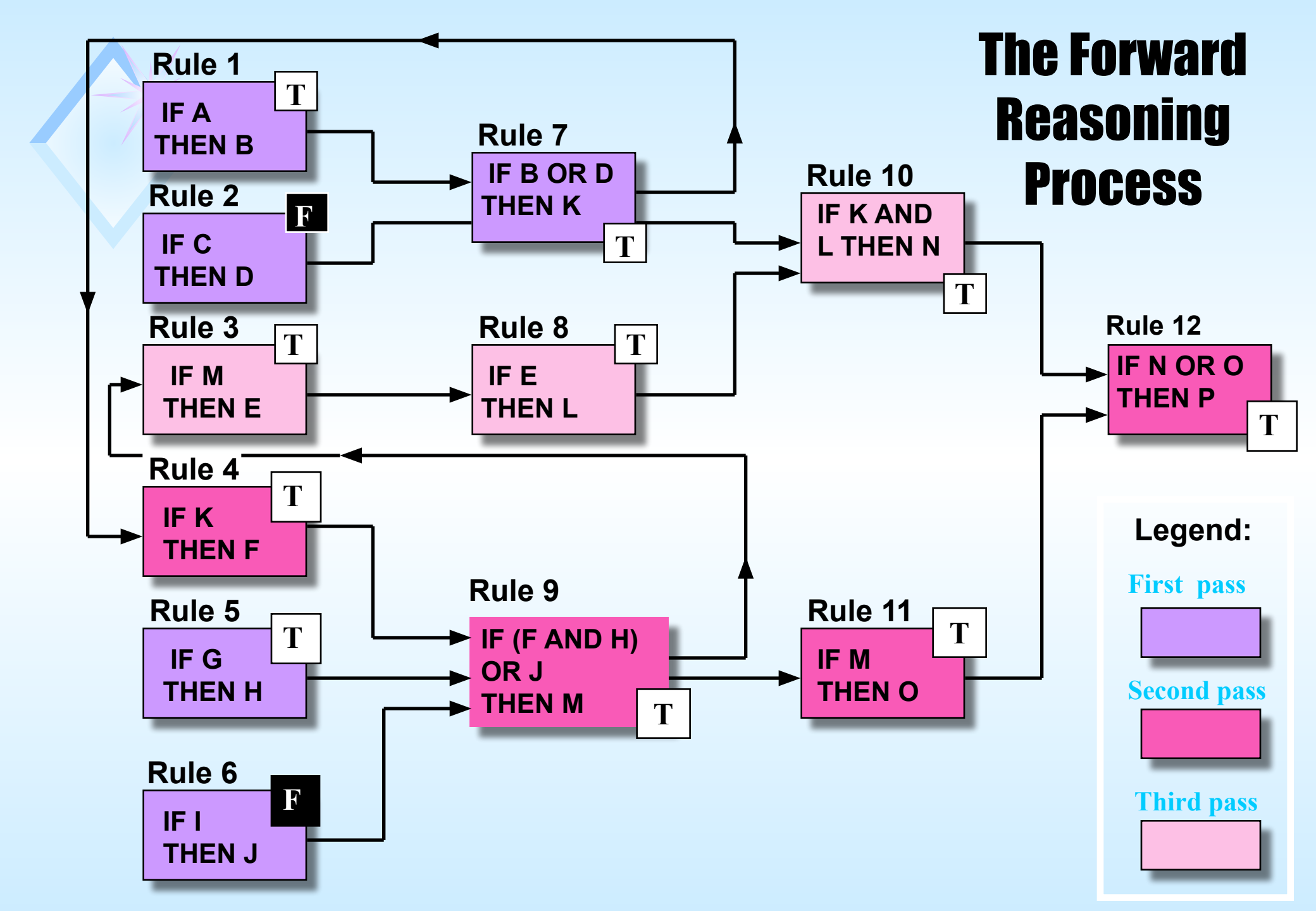
- Rule is evaluated as:
  - (1) true, (2) false, (3) unknown
- Rule evaluation is an iterative process
- When no more rules can fire, the reasoning process stops even if a goal has not been reached

Penalaran Maju (Forward Chaining)

- Pencocokan fakta atau pernyataan dimulai dari bagian sebelah kiri (IF), Atau penalaran dimulai dari fakta dulu untuk menguji kebenaran hipotesis.



The Forward Reasoning Process

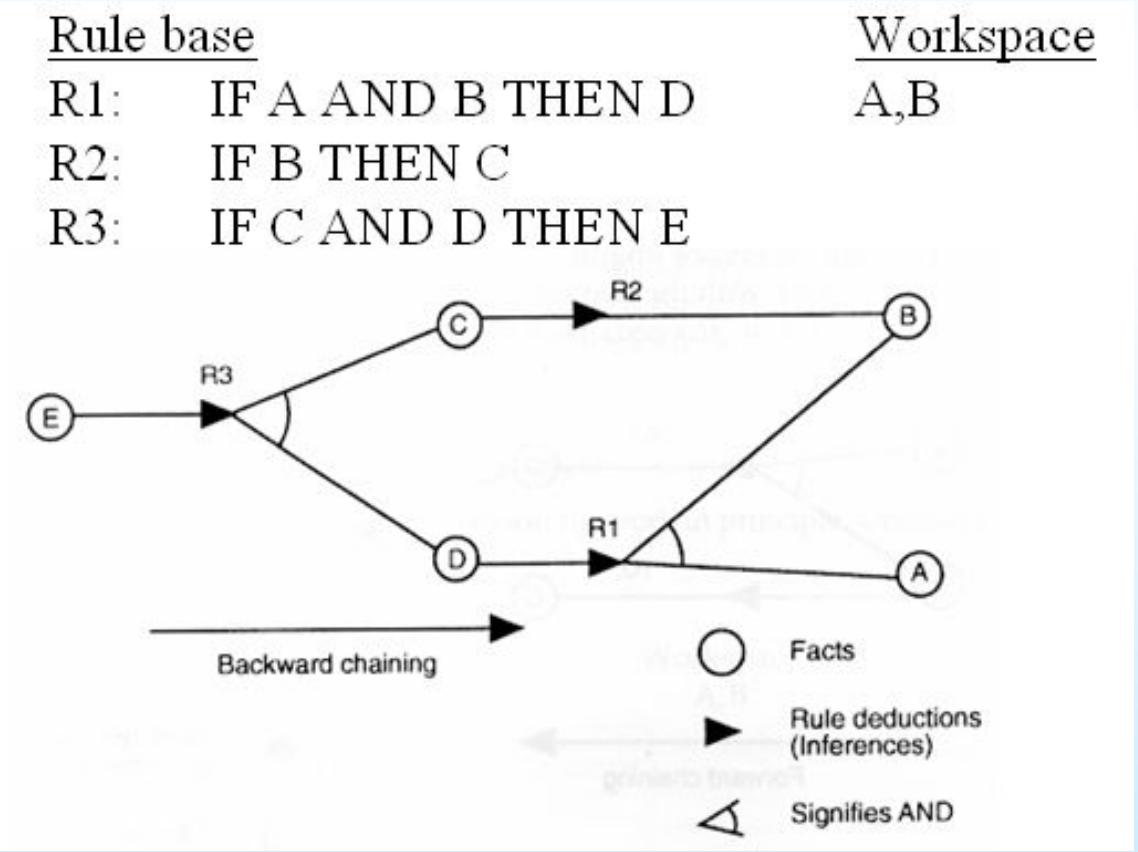


Reverse Reasoning (backward chaining)

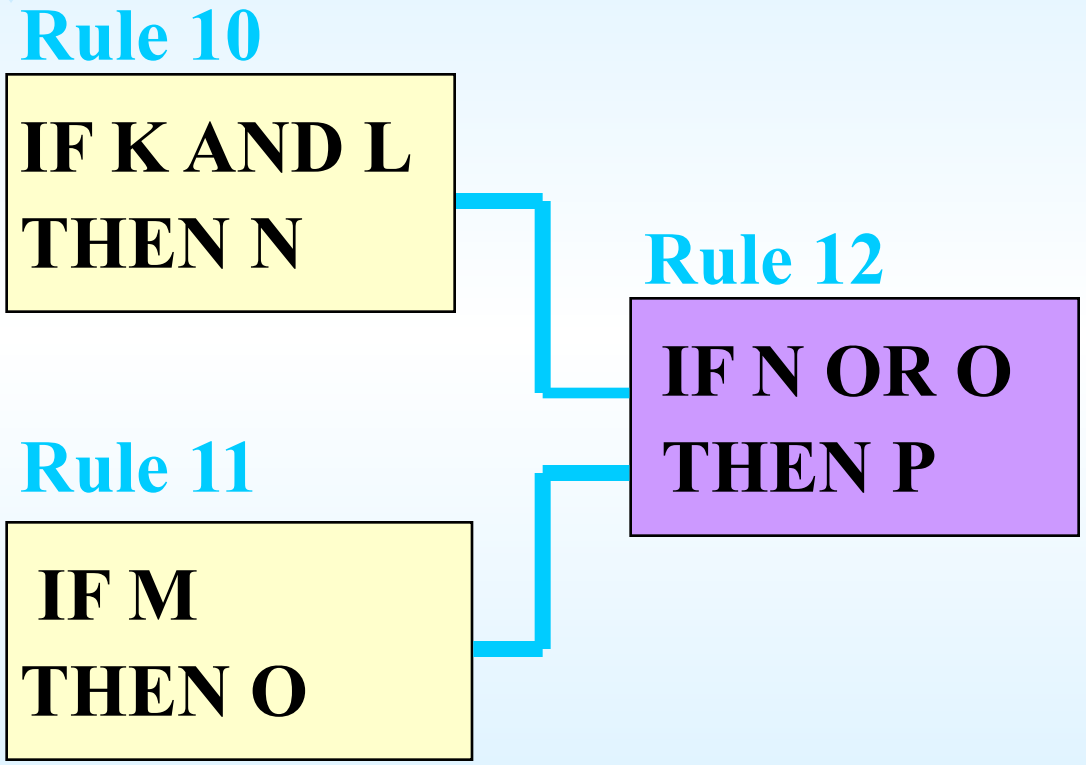
- Divide problem into subproblems
- Try to solve one subproblem
- Then try another

Penalaran Mundur (Backward Chaining)

- Mencocokan fakta atau pernyataan dimulai dari sebelah kanan (THEN), atau penalaran dimulai dari hipotesis terlebih dahulu, dan untuk menguji kebenaran hipotesis tersebut harus dicari fakta-fakta yang ada dalam basis pengetahuan.



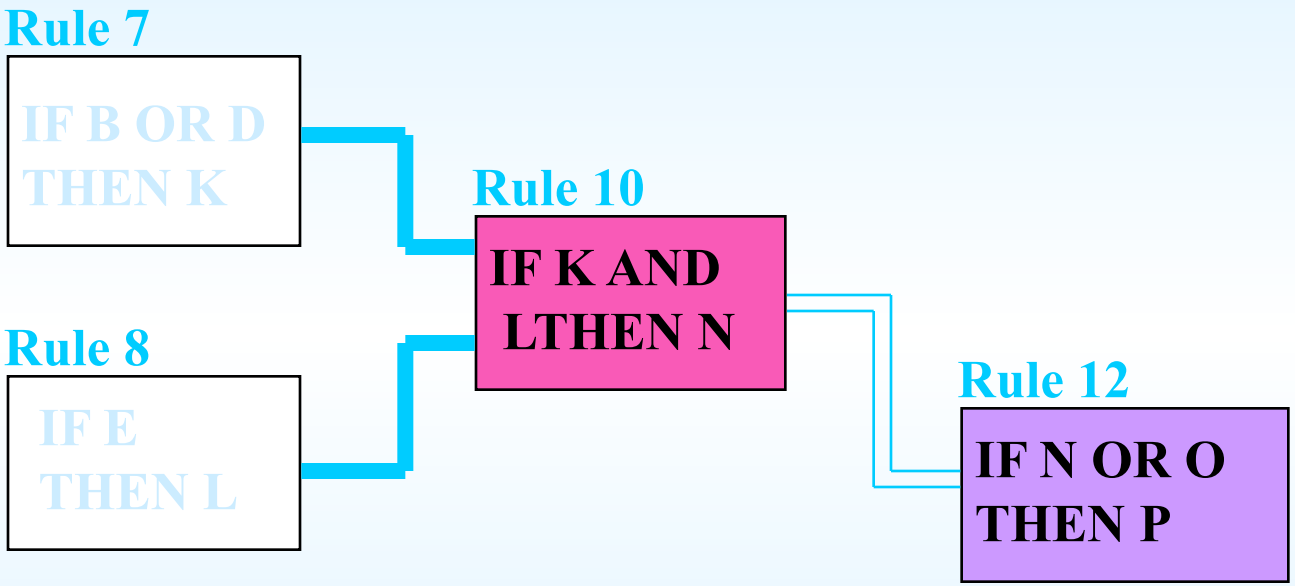
A Problem and Its Subproblems



Problem

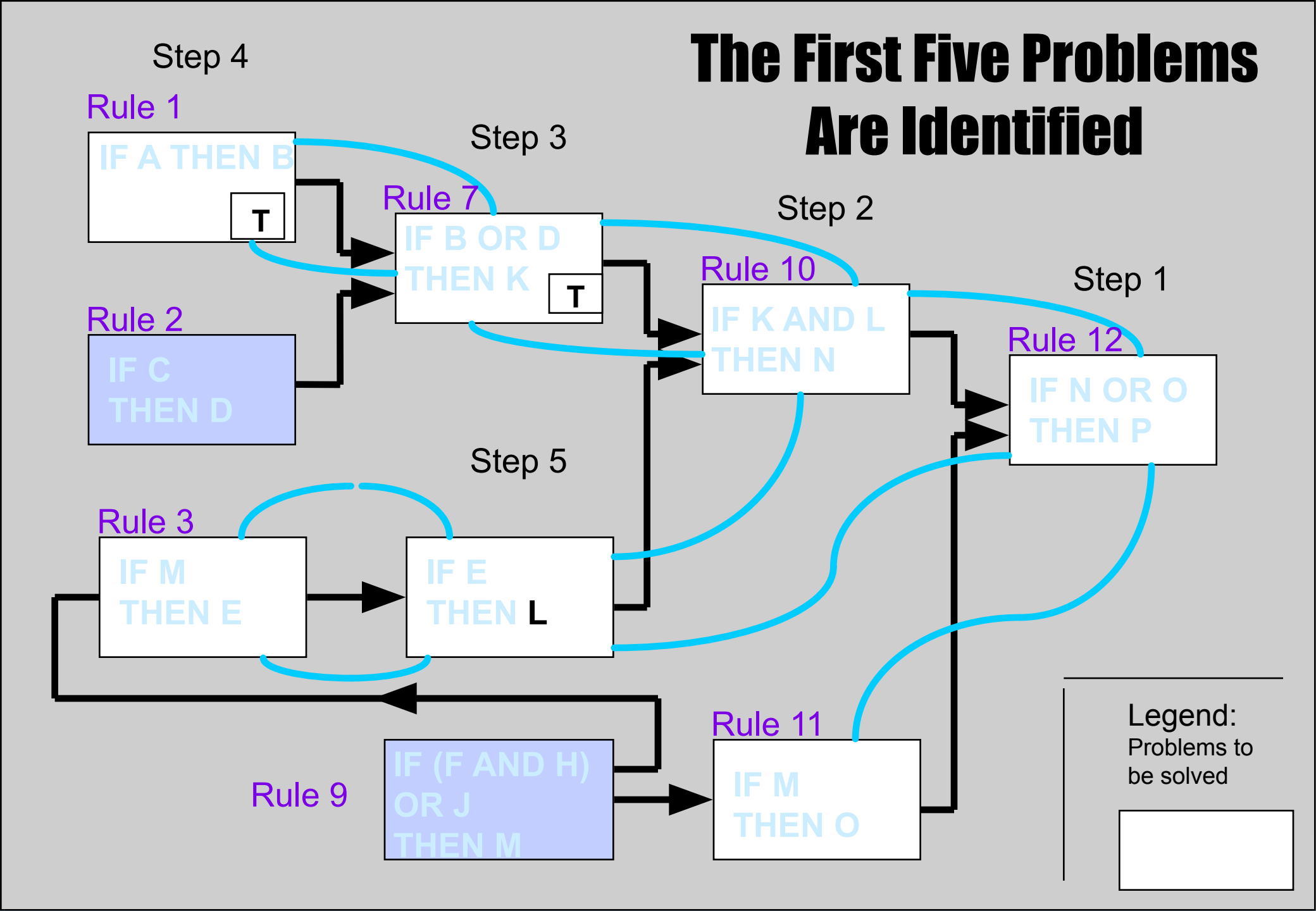
Subproblem

A Subproblem Becomes the New Problem

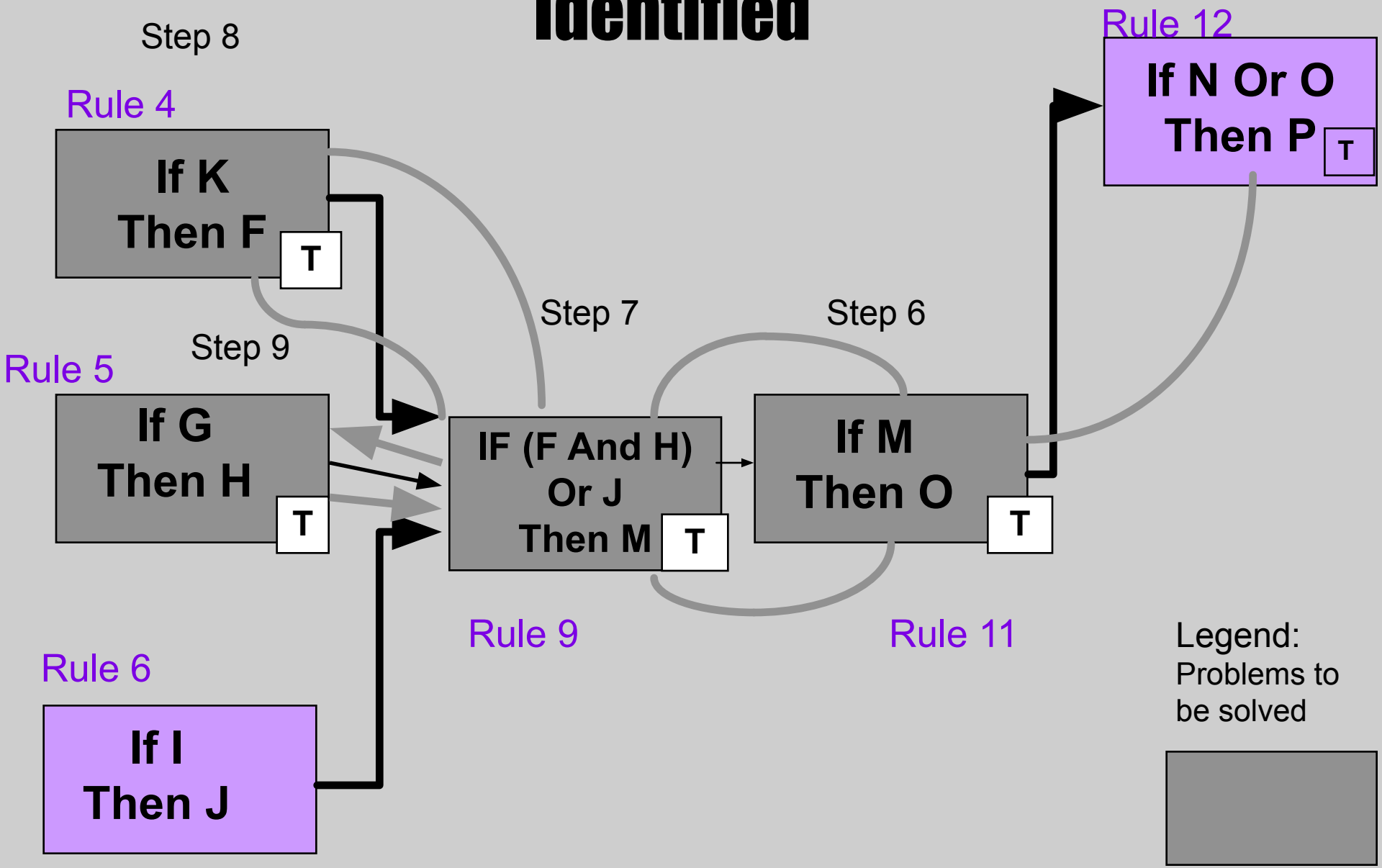


Problem

Subproblem


The First Five Problems Are Identified
Step 4
Rule 1
IF A THEN B
T
Step 3
Rule 7
IF B OR D
THEN K
T
Step 2
Rule 10
IF K AND L
THEN N
Step 1
Rule 12
IF N OR O
THEN P
Rule 2
IF C
THEN D
Step 5
Rule 3
IF M
THEN E
IF E
THEN L
Rule 11
IF M
THEN O
Rule 9
IF (F AND H)
OR J
THEN M
Legend:
Problems to be solved

The Next Four Problems Are Identified



Komponen-Komponen Sistem Pakar, Lanjut……

4. Blackboard/Workplace

adalah memori/lokasi untuk bekerja dan menyimpan hasil sementara. Biasanya berupa sebuah basis data.

5. Antarmuka Pemakai (User Interface).

Sistem Pakar mengatur komunikasi antara pengguna dan komputer. Komunikasi ini paling baik berupa bahasa alami, biasanya disajikan dalam bentuk tanya-jawab dan kadang ditampilkan dalam bentuk gambar/grafik. Antarmuka yang lebih canggih dilengkapi dengan percakapan (voice communication).

Komponen-Komponen Sistem Pakar, Lanjut……

6. Subsistem Penjelasan (Explanation Facility).

Kemampuan untuk mencari jejak (tracing) bagaimana suatu kesimpulan dapat diambil merupakan hal yang sangat penting untuk transfer pengetahuan dan pemecahan masalah. Komponen subsistem penjelasan harus dapat menyediakannya yang secara interaktif menjawab pertanyaan pengguna

7. Sistem Penyaringan Pengetahuan (Knowledge Refining System).

Seorang pakar mempunyai sistem penyaringan pengetahuan, artinya, mereka bisa menganalisa sendiri performa mereka, belajar dari pengalaman, serta meningkatkan pengetahuannya untuk konsultasi berikutnya. Pada Sistem Pakar, swa-evaluasi ini penting sehingga dapat menganalisa alas an keberhasilan atau kegagalan pengambilan kesimpulan, serta memperbaiki basis pengetahuannya.

Contoh

Rule-Based Expert Systems

- Based on the production system concept.

Rules

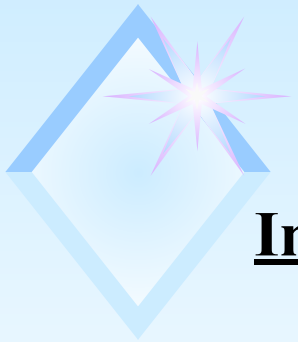
IF the engine is getting gas
AND the engine will turn over
THEN the problem is spark plugs

Facts

The engine is getting gas

Conclusion:

- action
- employ a particular model
- execute a procedure
- display a report



Inference Engine

(1) Selection of rule candidates: pattern matching

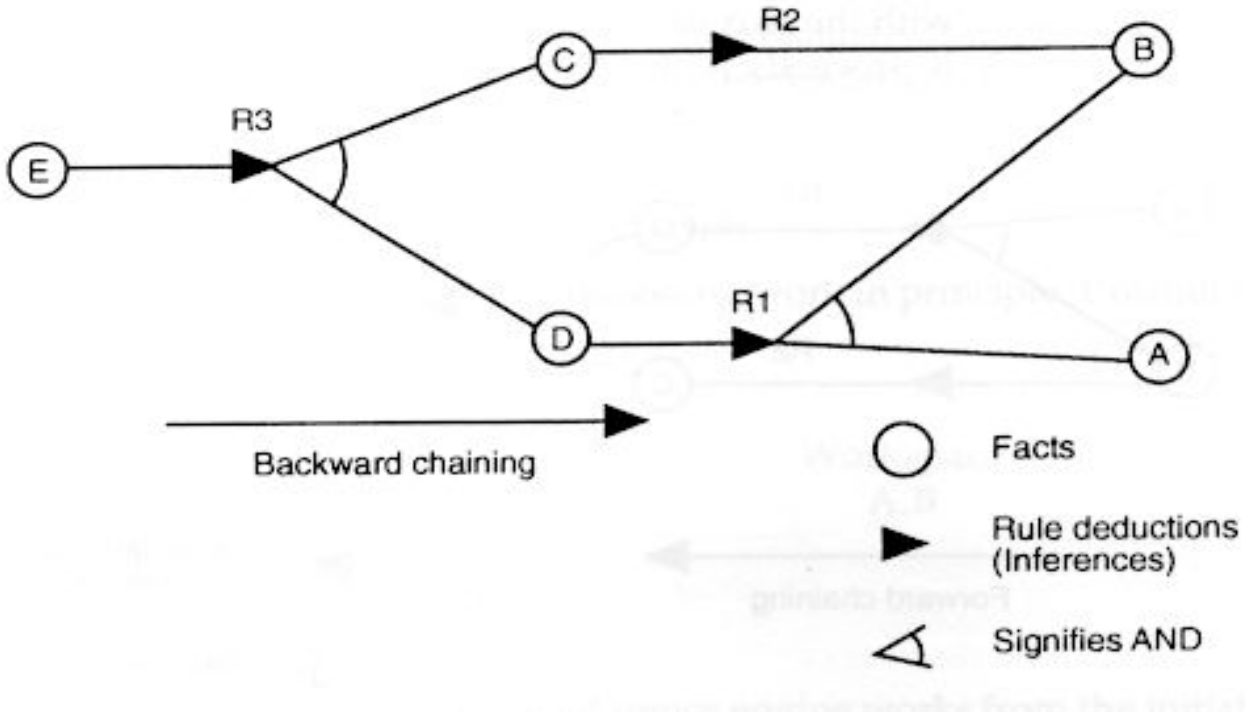
(2) Choice of one rule: conflict resolution

(3) Execution: deduction

- Backward chaining (goal driven): the inference engine works backward from a conclusion to be proven to determine if there are data in the workspace to prove the truth of the conclusion.

Example.

Rule base	Workspace
R1: IF A AND B THEN D	A,B
R2: IF B THEN C	
R3: IF C AND D THEN E	



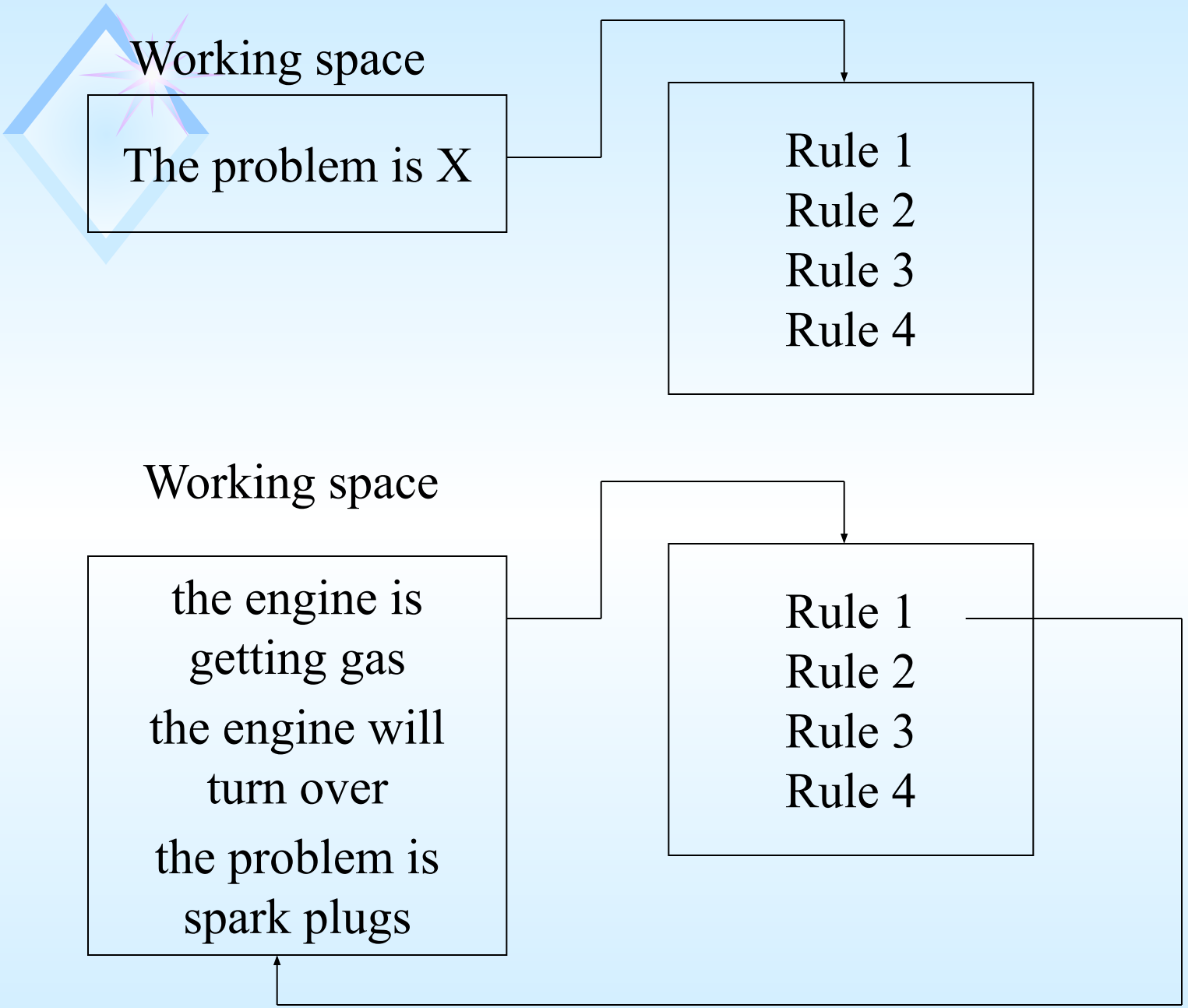
Example. Expert system for diagnosing car problems.

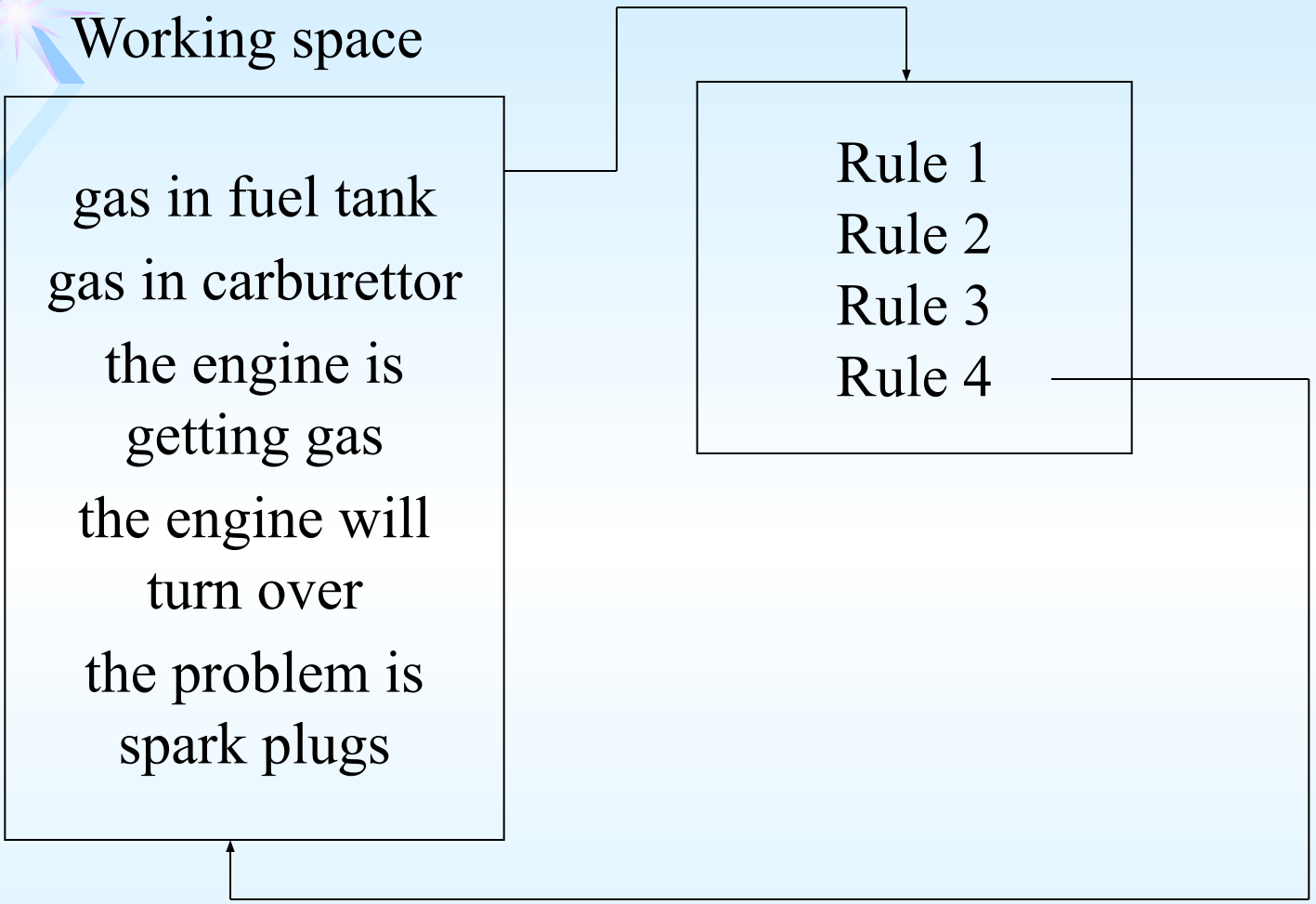
Rule 1: IF the engine is getting gas
AND the engine will turn over
THEN the problem is spark plugs

Rule 2: IF the engine does not turn over
AND the lights do not come on
THEN the problem is battery or cables.

Rule 3: IF the engine does not turn over
AND the lights do come on
THEN the problem is the starter motor.

Rule 4: IF there is gas in the fuel tank
AND there is gas in the carburettor
THEN the engine is getting gas


Working space
The problem is X
Rule 1
Rule 2
Rule 3
Rule 4
Working space
the engine is getting gas
the engine will turn over
the problem is spark plugs
Rule 1
Rule 2
Rule 3
Rule 4


Working space
gas in fuel tank
gas in carburettor
the engine is getting gas
the engine will turn over
the problem is spark plugs
Rule 1
Rule 2
Rule 3
Rule 4

Explanation in Backward Chaining

Why?

gas in fuel tank?
yes
gas in carburettor?
yes
engine will turn over?
why

It has been established that:
1. the engine is getting gas,
therefore if
2. the engine will turn over,
then the problem is spark plugs

How?

how the engine is getting gas

This follows from rule 4:

IF there is gas in the fuel tank
AND there is gas in the carburettor
THEN the engine is getting gas

gas in fuel tank was given by the user
gas in carburettor was given by the user

APLIKASI SISTEM PAKAR

Kategori Problema Sistem Pakar secara umum:

- ***Interpretasi*** – membuat kesimpulan atau deskripsi dari sekumpulan data mentah.
- ***Prediksi*** – memproyeksikan akibat-akibat yang dimungkinkan dari situasi-situasi tertentu
- ***Diagnosis*** – menentukan sebab malfungsi dalam situasi kompleks yang didasarkan pada gejala-gejala yang teramati

- ***Desain*** – menentukan konfigurasi komponen-komponen sistem yang cocok dengan tujuan-tujuan kinerja tertentu yang memnuhi kendala -kendala tertentu

- ***Perencanaan*** – merencanakan serangkaian tindakan yang akan dapat mencapai sejumlah tujuan dengan kondisi awal tertentu

- ***Instruksi*** – mendeteksi dan mengoreksi defisiensi dalam pemahaman domain subyek

- ***Pengendalian*** – mengatur tingkah laku suatu *environment* yang kompleks
- ***Selection*** – mengidentifikasi pilihan terbaik dari sekumpulan (*list*) kemungkinan
- ***Simulation*** – pemodelan interaksi antara komponen-komponen sistem
- ***Monitoring*** – membandingkan hasil pengamatan dengan kondisi yang diharapkan

Contoh Aplikasi Sistem Pakar

- Eliza
  - Eliza adalah salah satu Sistem Pakar yang paling awal dikembangkan. Ini adalah program komputer terapis yang dibuat oleh Joseph Weizenbaum di MIT. Pengguna berkomunikasi dengannya sebagaimana sedang berkonsultasi dengan seorang terapis.

- **contoh percakapan antara Eliza dengan seorang klien:**

Eliza: Hello. I am ELIZA. How can I help you?
Klien: My name is Balza.
Eliza: I have told you before, I do not care about names.
Klien: I don’t have any money. That’s my problem.
Eliza: Why do you say your problem?
Klien: Because you asked me.
Eliza: Do any other reasons not come to mind?
Klien: No.
Eliza: You are being a bit negative.
Klien: Why?
Eliza: Please go on.
Klien: OK. I am bankrupt.
Eliza: How long have you been bankrupt?
Klien: A year.
Eliza: Please go on.
Klien: So, what should I do?
Eliza: Does that question interest you?
Klien: Yes, how can I make big money?
Eliza: You are sure?
Klien: Of course I’m sure.

- Parry
  - Parry adalah Sistem Pakar yang juga paling awal dikembangkan di Stanford University oleh seorang psikiater, Kenneth Colby, yang mensimulasikan seorang paranoid

- contoh respon Parry terhadap pertanyaan seorang psikiater yang menginterviewnya

Psikiater: Why are you in the hospital?
Parry: I shouldn't be here.
Psikiater: Who brought you here?
Parry: The police.
Psikiater: What trouble did you have with the police?
Parry: Cops don't do their job.
Psikiater: What do they do instead?
Parry: Cops arrest the wrong people.
Psikiater: They arrest the right ones sometimes
Parry: The law should get those Italian crooks.
Psikiater: Are you Italian?
Parry: My background is British but I was born in this country.
Psikiater: Do you know any Italian crooks?
Parry: I try to avoid the underworld.

Expert System dan DSS

Perbedaan antara expert sistem dan DSS

- 1. DSS terdiri dari routine merefleksikan keyakinan manajer dalam caranya memecahkan masalah. Keputusan yang dihasilkan oleh DSS merefleksikan gaya kemampuan manajer, sebaliknya expert sistem memberikan peluang untuk mendapatkan kemampuan dalam membuat keputusan melebihi kemampuan yang dimiliki manajer
- 2. Expert System mempunyai kemampuan untuk menjelaskan jalur penalaran yang diikuti pencapaian pemecahan tertentu, penjelasan mengenai bagaimana pemecahan dicapai akan lebih berguna dari pada pemecahan itu sendiri.

Memutuskan kapan menggunakan Expert system

- DSS data yang digunakan berupa numerik, program menekankan penggunaan mathematical routines.
- Expert system data yang digunakan bersifat simbolis, seringkali berbentuk teksnaratif, program menekankan penggunaan logic routines.

Jika menghadapi masalah bisa Memilih menggunakan expert system dari pada DSS bila :

- Masalah tersebut melibatkan diagnosis situasi yang kompleks /melibatkan pembutan kesimpulan / peringkasan dari volume data yang besar.
- Ada tingkat ketidaktentuan dalam aspek masalah tertentu.
- Ada kemungkinan bagi ahli manusia untuk memecahkan masalah tersebut dala jangka waktu yang wajar.

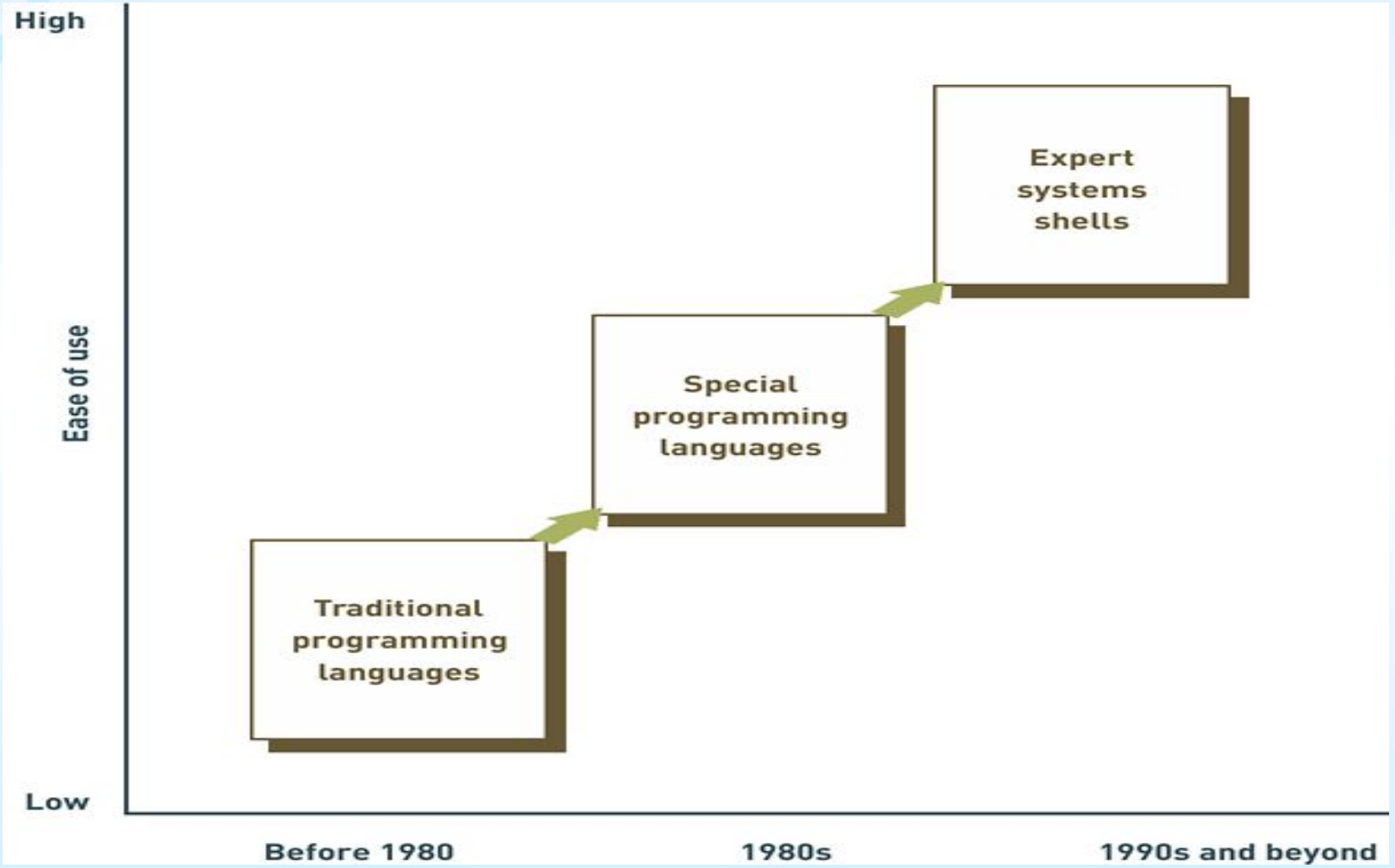
MENGEMBANGKAN SISTEM PAKAR

Expert Systems Development Tools and Techniques

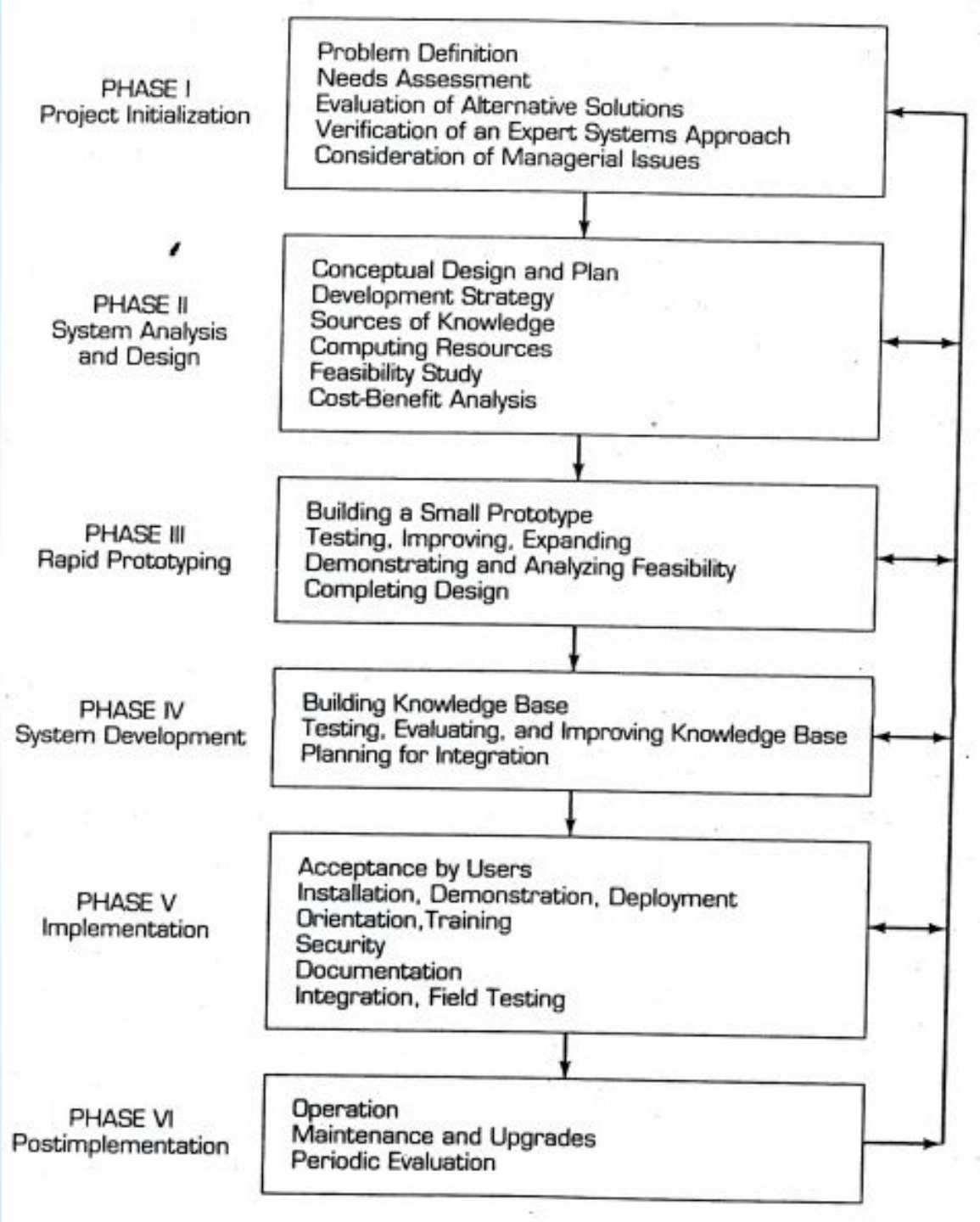
Mengembangkan Sistem Pakar dapat dilakukan dengan 2 cara:

1. Membangun sendiri semua komponen di atas, atau
2. Memakai semua komponen yang sudah ada kecuali isi basis pengetahuan.
   - Yang kedua disebut sebagai membangun Sistem Pakar dengan shell, yakni semua komponen Sistem Pakar, kecuali basis pengetahuan, bersifat generik; sehingga dapat dipakai untuk bidang yang berlainan.
   - Membangun Sistem Pakar dengan shell dapat dilakukan dengan lebih cepat dan lebih sedikit keterampilan memprogram, namun berkurang fleksibilitasnya karena harus mengikuti kemampuan dari shell tersebut.
   - Salah satu shell Sistem Pakar yang populer dipakai adalah CLIPS (C Language Integrated Production System)

Expert Systems Development Tools and Techniques



Langkah-langkah dalam pengembangan Sistim Pakar



1. Pemilihan Masalah

Pembuatan Sistem Pakar membutuhkan waktu dan biaya yang banyak. Untuk menghindari kegagalan yang memalukan dan kerugian yang besar, maka dibuat beberapa pedoman untuk menentukan apakah Sistem Pakar cocok untuk memecahkan suatu problem:

1. Biaya yang diperlukan untuk pembangunan Sistem Pakar ditentukan oleh kebutuhan untuk memperoleh solusi. Sehingga harus ada perhitungan yang realistis untuk cost and benefit.
2. Pakar manusia tidak mudah ditemui untuk semua situasi di mana dia dibutuhkan. Jika pakar pengetahuan tersebut terdapat di mana saja dan kapan saja, maka pembangunan Sistem Pakar menjadi kurang berharga.
3. Problem yang ada dapat diselesaikan dengan teknik penalaran simbolik, dan tidak membutuhkan kemampuan fisik.

1. Pemilihan Masalah (Lanjt)

4. Problem tersebut harus terstruktur dengan baik dan tidak membutuhkan terlalu banyak pengetahuan awam (common sense), yang terkenal sulit untuk diakuisisi dan dideskripsikan, dan lebih banyak berhubungan dengan bidang yang teknis.
5. - Problem tersebut tidak mudah diselesaikan dengan metode komputasi yang lebih tradisionil. Jika ada penyelesaian algoritmis yang bagus untuk problem tersebut, maka kita tidak perlu memakai Sistem Pakar.
6. - Ada pakar yang mampu memberikan penjelasan tentang kepakarannya serta mau bekerjasama. Adalah sangat penting bahwa pakar yang dihubungi benar-benar mempunyai kemauan kuat untuk ikut berpartisipasi serta tidak merasa pekerjaannya akan menjadi terancam.
7. - Problem tersebut mempunyai sekup yang tepat. Biasanya merupakan problem yang membutuhkan kepakaran yang sangat khusus namun hanya membutuhkan seorang pakar untuk dapat menyelesaikannya dalam waktu yang relative singkat (misalnya paling lama 1 jam).

2. Rekayasa Pengetahuan (Knowledge Engineering)

Proses dalam rekayasa pengetahuan meliputi :

1. Akuisisi pengetahuan, yaitu bagaimana memperoleh pengetahuan dari pakar atau sumber lain
2. Validasi pengetahuan, untuk menjaga kualitasnya misalnya dengan uji kasus.
3. Representasi pengetahuan, yaitu bagaimana mengorganisasi pengetahuan yang diperoleh, mengkodekan dan menyimpannya dalam suatu basis pengetahuan.
4. Penyimpulan pengetahuan, menggunakan mesin inferensi yang mengakses basis pengetahuan dan kemudian melakukan penyimpulan.
5. Transfer pengetahuan (penjelasan). Hasil inferensi berupa nasehat, rekomendasi, atau jawaban, kemudian dijelaskan ke pengguna oleh subsistem penjelas.

3. Partisipan Dalam Proses Pengembangan Pakar

- yaitu seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat.
  1. Pakar menyediakan pengetahuan tentang bagaimana nantinya Sistem Pakar bekerja.
  2. Perekayasa pengetahuan (knowledge engineer), yang membantu pakar untuk menyusun area permasalahan dengan menerjemahkan dan mengintegrasikan jawaban pakar terhadap pertanyaan-pertanyaan dari klien, menarik analogi, serta memberikan contoh-contoh yang berlawanan, kemudian menyusun basis pengetahuan.
  3. Pengguna, yang mungkin meliputi:
     - seorang klien non-pakar yang sedang membutuhkan nasehat (Sistem Pakar sebagai konsultan atau advisor),
     - seorang siswa yang sedang belajar (Sistem Pakar sebagai instruktur),
     - seorang pembuat Sistem Pakar yang hendak meningkatkan basis pengetahuan (Sistem Pakar sebagai partner),
     - seorang pakar (Sistem Pakar sebagai kolega atau asisten, yang dapat memberikan opini kedua).
     - Partisipan lain, dapat meliputi: pembangun sistem (system builder), tool builder, staf administrasi dsb.

4. Akuisisi Pengetahuan

- Dalam proses akuisisi pengetahuan, seorang perekayasa pengetahuan menjembatani antara pakar dengan basis pengetahuan.
- Perekayasa pengetahuan mendapatkan pengetahuan dari pakar, mengolahnya bersama pakar tersebut, dan menaruhnya dalam basis pengetahuan, dengan format tertentu.
- Pengambilan pengetahuan dari pakar dapat dilakukan secara :
  - Manual, di mana perekayasa pengetahuan mendapatkan pengetahuan dari pakar (melalui wawancara) dan/atau sumber lain, kemudian mengkodekannya dalam basis pengetahuan. Proses ini biasanya berlangsung lambat, mahal, serta kadangkala tidak akurat.
  - Semi-otomatik, di mana terdapat peran komputer untuk:
    (1) mendukung pakar dengan mengijinkannya membangun basis pengetahuan tanpa (atau dengan sedikit) bantuan dari perekayasa pengetahuan, atau
    (2) (2) membantu perekayasa pengetahuan sehingga kerjanya menjadi lebih efisien dan efektif.
  - Otomatik, di mana peran pakar, perekayasa pengetahuan, dan pembangun basis pengetahuan (system builder) digabung. Misalnya dapat dilakukan oleh seorang system analyst seperti pada metode induksi.

5. Representasi Pengetahuan

- Setelah pengetahuan berhasil diakuisisi, mereka harus diorganisasi dan diatur dalam suatu konfigurasi dengan suatu format/representasi tertentu.
- Metode representasi pengetahuan yang populer adalah aturan produk dan bingkai.

5. Representasi Pengetahuan (Lanjt)

1. Aturan Produk (Production Rules)

- Di sini pengetahuan disajikan dalam aturan-aturan yang berbentuk pasangan keadaan-aksi (condition-action): JIKA keadaan terpenuhi atau terjadi MAKA suatu aksi akan terjadi;.
- Sistem Pakar yang basis pengetahuannya melulu disajikan dalam bentuk aturan produk disebut sistem berbasis-aturan (rule-based system).
- Kondisi dapat terdiri atas banyak bagian, demikian pula dengan aksi. Urutan keduanya juga dapat dipertukarkan letaknya.
- Contohnya:
  - JIKA suhu berada di bawah 20ºC MAKA udara terasa dingin.
  - Udara terasa dingin JIKA suhu berada di bawah 20ºC.
  - JIKA suhu berada di bawah 20ºC ATAU suhu berada di antara 20-25ºC DAN angin bertiup cukup kencang MAKA udara terasa dingin.

5. Representasi Pengetahuan (Lanjut)

2. Bingkai (frame)

- Bingkai adalah struktur data yang mengandung semua informasi/pengetahuan yang relevan dari suatu obyek.
- Pengetahuan ini diorganisasi dalam struktur hirarkis khusus yang memungkinkan pemrosesan pengetahuan.
- Bingkai merupakan aplikasi dari pemrograman berorientasi obyek dalam AI dan Sistem Pakar.
- Pengetahuan dalam bingkai dibagi-bagi ke dalam slot atau atribut yang dapat mendeskripsikan pengetahuan secara deklaratif ataupun prosedural.

6. Bagaimana Sistem Pakar Melakukan Inferensi?

1. Sistem Perantaian Maju (Forward Chaining Systems)
   - Pada sistem perantaian maju, fakta-fakta dalam dalam sistem disimpan dalam memori kerja dan secara kontinyu diperbarui.
   - Aturan dalam sistem merepresentasikan aksi-aksi yang harus diambil apabila terdapat suatu kondisi khusus pada item-item dalam memori kerja, sering disebut aturan kondisi-aksi. Kondisi biasanya berupa pola yang cocok dengan item yang ada di dalam memori kerja, sementara aksi biasanya berupa penambahan atau penghapusan item dalam memori kerja.
   - Aktivitas sistem dilakukan berdasarkan siklus mengenal-beraksi (recognise-act).
   - Mula-mula, sistem mencari semua aturan yang kondisinya terdapat di memori kerja, kemudian memilih salah satunya dan menjalankan aksi yang bersesuaian dengan aturan tersebut.
   - Pemilihan aturan yang akan dijalankan (fire) berdasarkan strategi tetap yang disebut strategi penyelesain konflik. Aksi tersebut menghasilkan memori kerja baru, dan siklus diulangi lagi sampai tidak ada aturan yang dapat dipicu (fire), atau goal (tujuan) yang dikehendaki sudah terpenuhi.

2. Strategi penyelesaian konflik (conflict resolution strategy)

- Strategi penyelesaian konflik dilakukan untuk memilih aturan yang akan diterapkan apabila terdapat lebih dari 1 aturan yang cocok dengan fakta yang terdapat dalam memori kerja. Di antaranya adalah:
  - No duplication. Jangan memicu sebuah aturan dua kali menggunakan fakta/data yang sama, agar tidak ada fakta yang ditambahkan ke memori kerja lebih dari sekali.
  - Recency. Pilih aturan yang menggunakan fakta yang paling baru dalam memori kerja. Hal ini akan membuat sistem dapat melakukan penalaran dengan mengikuti rantai tunggal ketimbang selalu menarik kesimpulan baru menggunakan fakta lama.
  - Specificity. Picu aturan dengan fakta prakondisi yang lebih spesifik (khusus) sebelum aturan yang mengunakan prakondisi lebih umum.
  - Operation priority. Pilih aturan dengan prioritas yang lebih tinggi.

3. Sistem Perantaian Balik (Backward Chaining Systems)

- Sejauh ini kita telah melihat bagaimana sistem berbasis aturan dapat digunakan untuk menarik kesimpulan baru dari data yang ada, menambah kesimpulan ini ke dalam memori kerja. Pendekatan ini berguna ketika kita mengetahui semua fakta awalnya, namun tidak dapat menebak konklusi apa yang bisa diambil. Jika kita tahu kesimpulan apa yang seharusnya, atau mempunyai beberapa hipotesis yang spesifik, maka perantaian maju di atas menjadi tidak efisien.
- Sebagai contoh, jika kita ingin mengetahui apakah saya dalam keadaan mempunyai mood yang baik sekarang kemungkinan kita akan berulangkali memicu aturan-aturan dan memperbarui memori kerja untuk mengambil kesimpulan apa yang terjadi pada bulan Maret, atau apa yang terjadi jika saya mengajar, yang sebenarnya perlu terlalu kita ambil pusing. Dalam hal ini yang diperlukan adalah bagaimana dapat menarik kesimpulan yang relevan dengan tujuan atau goal.

- Hal ini dapat dikerjakan dengan perantaian balik dari pernyataan goal (atau hipotesis yang menarik bagi kita). Jika diberikan sebuah goal yang hendak dibuktikan, maka mula-mula sistem akan memeriksa apakah goal tersebut cocok dengan fakta-fakta awal yang dimiliki. Jika ya, maka goal terbukti atau terpenuhi. Jika tidak, maka sistem akan mencari aturan-aturan yang konklusinya (aksinya) cocok dengan goal. Salah satu aturan tersebut akan dipilih, dan sistem kemudian akan mencoba membuktikan fakta-fakta prakondisi aturan tersebut menggunakan prosedur yang sama, yaitu dengan menset prakondisi tersebut sebagai goal baru yang harus dibuktikan. Perhatikan bahwa pada perantaian balik, sistem tidak perlu memperbarui memori kerja, namun perlu untuk mencatat goal-goal apa saja yang dibuktikan untuk membuktikan goal utama (hipotesis).
- Secara prinsip, kita dapat menggunakan aturan-aturan yang sama untuk perantaian maju dan balik. Namun, dalam prakteknya, harus sedikit dimodifikasi.
- Pada perantaian balik, bagian MAKA dalam aturan biasanya tidak diekspresikan
- sebagai suatu aksi untuk dijalankan (misalnya TAMBAH atau HAPUS), tetapi suatu keadaan yang bernilai benar jika premisnya (bagian JIKA) bernilai benar.

SEKIAN

TEKNIK INFERENSI

Definisi Inferensi

- Inferensi adalah **:** Proses yang digunakan dalam Sistem Pakar untuk menghasilkan informasi baru dari informasi yang telah diketahui
- Dalam sistem pakar proses inferensi dilakukan dalam suatu modul yang disebut *Inference Engine* (Mesin inferensi)
- Ketika representasi pengetahuan (RP) pada bagian *knowledge base* telah lengkap, atau paling tidak telah berada pada level yang cukup akurat, maka RP tersebut telah siap digunakan.
- Inference engine merupakan modul yang berisi program tentang bagaimana mengendalikan proses *reasoning*.

REASONING

- **Definisi** : Proses bekerja dengan pengetahuan, fakta dan strategi pemecahan masalah, untuk mengambil suatu kesimpulan. (Berpikir dan mengambil kesimpulan)
- **Metode Reasoning**
  - Deductive Reasoning
  - Inductive Reasoning
  - Abductive Reasoning
  - Analogical Reasoning
  - Common Sense Reasoning

Deductive Reasoning

- Kita menggunakan *reasoning* deduktif untuk mendeduksi informasi baru dari hubungan logika pada informasi yang telah diketahui.
- **Contoh:**
  - Implikasi : Saya akan basah kuyup jika berdiri ditengah-tengah hujan deras
  - Aksioma : Saya berdiri ditengah-tengah hujan deras
  - Konklusi : Saya akan basah kuyup

  IF A is True AND IF A IMPLIES B is True, Then B is True

Inductive Reasoning

- Kita menggunakan *reasoning* induktif untuk mengambil kesimpulan umum dari sejumlah fakta khusus tertentu.
- **Contoh:**
  - Premis : Monyet di Kebun Binatang Ragunan makan pisang
  - Premis : Monyet di Kebun Raya Bogor makan pisang
  - Konklusi : Semua monyet makan pisang

Abductive Reasoning

- Merupakan bentuk dari proses deduksi yang mengijinkan inferensi *plausible*.
- *Plausible* berarti bahwa konklusi mungkin bisa mengikuti informasi yang tersedia, tetapi juga bisa salah.
- Contoh:
  - Implikasi : Tanah menjadi basah jika terjadi hujan
  - Aksioma : Tanah menjadi basah
  - Konklusi : Apakah terjadi hujan?

  IF B is True AND A implies B is true, Then A is True?

Analogical Reasoning

- Kita mengunakan pemodelan analogi untuk membantu kita memahami situasi baru atau objek baru.
- Kita menggambar analogi antara 2 objek/situasi, kemudian melihat persamaan dan perbedaan untuk memandu proses *reasoning*.

Common Sense Reasoning

- Melalui pengalaman, manusia belajar untk memecahkan masalahnya secara efisien.
- Dengan menggunakan *common sense* untuk secara cepat memperoleh suatu solusi.
- Dalam sistem pakar, dapat dikategorikan sebagai *Heuristic*.
- Proses ***heuristic search*** atau ***best first search*** digunakan pada aplikasi yang membutuhkan solusi yang cepat

Reasoning dengan Logika

- **Modus Ponen**
- Definisi: Rule dari logika yang menyatakan bahwa jika kita tahu A adalah benar dan A implies B adalah juga benar, maka kita dapat mengasumsikan bahwa B benar.

$[A \text{ AND } (A \rightarrow B)] \rightarrow B$

IF A is True
AND A → B is True
THEN B is True

Contoh:

- A = Udara Cerah
- B = Kita akan pergi ke pantai
- A→B = Jika udara cerah, maka kita pergi ke pantai
- Dengan menggunakan Modus Ponen, kita bisa menarik kesimpulan bahwa

  "*Kita akan pergi ke Pantai*"

Resolusi

- Definisi: Strategi inferensi yang digunakan pada sistem logika untuk menentukan kebenaran dari suatu assertion (penegasan)
  - Metoda Resolusi mencoba untuk membuktikan bahwa beberapa teorema atau ekspresi sebagai proposisi P adalah TRUE, dengan memberikan sejumlah aksioma dari masalah tersebut.
  - *Proof by Refutation*, suatu teknik yang ingin membuktikan bahwa ¬P tidak dapat menjadi TRUE.
  - *Resolvent* : ekspresi baru yang muncul dari metode resolusi yang merupakan gabungan (*union*) dari aksioma yang ada dengan teorema negasi.

Misalnya:

- Ada 2 aksioma:
  - A V B (A is True OR B is True) dan ¬ B V C (B is Not True OR C is True).
  - **(A V B) Λ (¬ B V C) = A V C**
- *Resolvent* tersebut kemudian ditambahkan pada list dari aksioma dan akan menghasilkan resolvent baru. Proses ini berulang sampai menghasilkan kontradiksi.

Nonresolusi

- Pada resolusi, tidak ada pembedaan antara *goals* (tujuan), *premises* maupun *rules*. Semua dianggap sebagai aksioma dan diproses dengan rule resolusi untuk inferensi.
- Cara tersebut dapat menyebabkan kebingungan karena menjadi tidak jelas apa yang ingin dibuktikan
- Teknik **Nonresolusi** atau ***natural-deduction*** mencoba mengatasi hal tersebut dengan menyediakan beberapa statement sebagai *goal*-nya
  - Untuk membuktikan [H $\mathbf{\Lambda}$ (A $\rightarrow$ B) $\rightarrow$ C]:
  - If (B $\rightarrow$ C), then membuktikan (H $\rightarrow$ A)

Perhatikan contoh kasus berikut:

- Untuk menjelaskan pendekatan ini, *Misalkan kita ingin membuktikan apakah Jack suka tim sepakbola Arema-Malang. Asumsinya adalah bahwa semua orang yang tinggal di Malang menyukai Arema. Karenanya, jika kita bisa mengetahui bahwa Jack tinggal di Malang, maka kita bisa membuktikan tujuan kita.*
- Kasus tadi dapat direpresentasi menjadi:
- **Antecedents:**

  [Lives-Malang(Jack) Λ (Lives-Malang(X) → Likes-Arema(X))
- **Goals:**

  → Likes-Arema(Jack)]:
- Pada contoh tersebut, kita bisa membuktikan *goal*-nya jika **(Lives-Malang(X) → Likes-Arema(X))**, dan

Beberapa hukum dalam Inferensi dengan Logika

- Hukum *Detachment*:

  p -> q
  p
  maka, q

- Hukum Kontrapositif:

  p -> q
  maka, ~q -> ~p

- Chain rule (Hukum *Syllogism*)

  p -> q
  q -> r
  maka, p -> r

- Hukum Inferensi *Disjunctive*:

  p v q
  ~p
  maka, q

  p v q
  ~q
  p

- Hukum Dobel Negasi:

  ~(~p)
  maka, p

Beberapa hukum dalam Inferensi dengan Logika

Hukum DeMorgan:		$\sim(p \wedge q)$	$\sim(p \vee q)$
	maka,	$\sim p \vee \sim q$	$\sim p \wedge \sim q$
Hukum Simplifikasi:		$p \wedge q$	$\sim(p \vee q)$
	maka,	p	q
Hukum Konjungsi:		p	
		q	
	maka,	$p \wedge q$	
Hukum *Disjunctive Addition*:		p	
	maka,	$p \vee q$	
Hukum *Conjunctive Argument*:		$\sim(p \wedge q)$	$\sim(p \wedge q)$
		p	q
	maka,	$\sim q$	$\sim p$

INFERENCING DENGAN RULES: FORWARD dan BACKWARD CHAINING

- Inferensi dengan rules merupakan implementasi dari modus ponen, yang direfleksikan dalam mekanisme *search* (pencarian).
- ***Firing a rule***: Bilamana semua hipotesis pada *rules* (bagian "IF") memberikan pernyataan BENAR
- Dapat mengecek semua *rule* pada *knowledge base* dalam arah *forward* maupun *backward*
- Proses pencarian berlanjut sampai tidak ada rule yang dapat digunakan (*fire*), atau sampai sebuah tujuan (*goal*)tercapai.
- Ada dua metode *inferencing* dengan rules, yaitu ***Forward Chaining*** atau *Data-Driven* dan ***Backward Chaining*** atau *Goal-Driven*.

BACKWARD CHAINING

- Pendekatan *goal-driven*, dimulai dari ekspektasi apa yang diinginkan terjadi (hipotesis), kemudian mengecek pada sebab-sebab yang mendukung (ataupun kontradiktif) dari ekspektasi tersebut.
- Jika suatu aplikasi menghasilkan *tree* yang sempit dan cukup dalam, maka gunakan *backward chaining*.
- Beberapa sifat dari *backward chaining*:
  - *Good for Diagnosis.*
  - *Looks from present to past.*
  - *Works from consequent to antecedent.*
  - *Is goal-driven, top-down reasoning.*
  - *Works backward to find facts that support the hypothesis.*
  - *It facilitates a depth-first search.*
  - *The consequents determine the search.*
  - *It does facilitate explanation.*

BACKWARD CHAINING

- Pada komputer, program dimulai dengan tujuan (*goal*) yang diverifikasi apakah bernilai TRUE atau FALSE
- Kemudian melihat pada suatu rule yang mempunyai GOAL tersebut pada bagian konklusinya.
- Mengecek pada premis dari rule tersebut untuk menguji apakah rule tersebut terpenuhi (bernilai TRUE)
- Pertama dicek apakah ada assertion-nya
  - Jika pencarian disitu gagal, maka ES akan mencari rule lain yang memiliki konklusi yang sama dengan rule pertama tadi
  - Tujuannya adalah membuat rule kedua terpenuhi (*satisfy*)
- Proses tersebut berlajut sampai semua kemungkinan yang ada telah diperiksa atau sampai rule inisial yang diperiksa (dg GOAL) telah terpenuhi
- Jika GOAL terbukti FALSE, maka GOAL berikut yang dicoba.

Forward chaining

- Merupakan grup dari multipel inferensi yang melakukan pencarian dari suatu masalah kepada solusinya.
- Jika klausa premis sesuai dengan situasi (bernilai TRUE), maka proses akan meng-assert konklusi
- Forward Chaining adalah *data driven* karena inferensi dimulai dengan informasi yg tersedia dan baru konklusi diperoleh
- Beberapa Sifat *forward chaining*:
  - *Good for monitoring, planning, and control*
  - *Looks from present to future.*
  - *Works from antecedent to consequent.*
  - *Is data-driven, bottom-up reasoning.*
  - *Works forward to find what solutions follow from the facts.*
  - *It facilitates a breadth-first search.*
  - *The antecedents determine the search.*
  - *It does not facilitate explanation.*
- Jika suatu aplikasi menghasilkan *tree* yang lebar dan tidak dalam, maka gunakan *forward chaining*

Contoh Kasus

- **Sistem Pakar**: Penasihat Keuangan
- **Kasus** : Seorang *user* ingin berkonsultasi apakah tepat jika dia berinvestasi pada *stock* IBM?
- **Variabel-variabel yang digunakan:**
  - A = memiliki uang $10.000 untuk investasi
  - B = berusia < 30 tahun
  - C = tingkat pendidikan pada level college
  - D = pendapatan minimum pertahun $40.000
  - E = investasi pada bidang Sekuritas (Asuransi)
  - F = investasi pada saham pertumbuhan (*growth stock)*
  - G = investasi pada saham IBM
- Setiap variabel dapat bernilai TRUE atau FALSE

FAKTA YANG ADA:

- Diasumsikan si *user* (investor) memiliki data:
  - Memiliki uang $10.000 (A TRUE)
  - Berusia 25 tahun (B TRUE)
- Dia ingin meminta nasihat apakah tepat jika berinvestasi pada IBM stock?
- **RULES**
  - R1 : IF seseorang memiliki uang $10.000 untuk berinvestasi AND dia berpendidikan pada level college THEN dia harus berinvestasi pada bidang sekuritas
  - R2 : IF seseorang memiliki pendapatan pertahun min $40.000 AND dia berpendidikan pada level college THEN dia harus berinvestasi pada saham pertumbuhan (*growth stocks)*

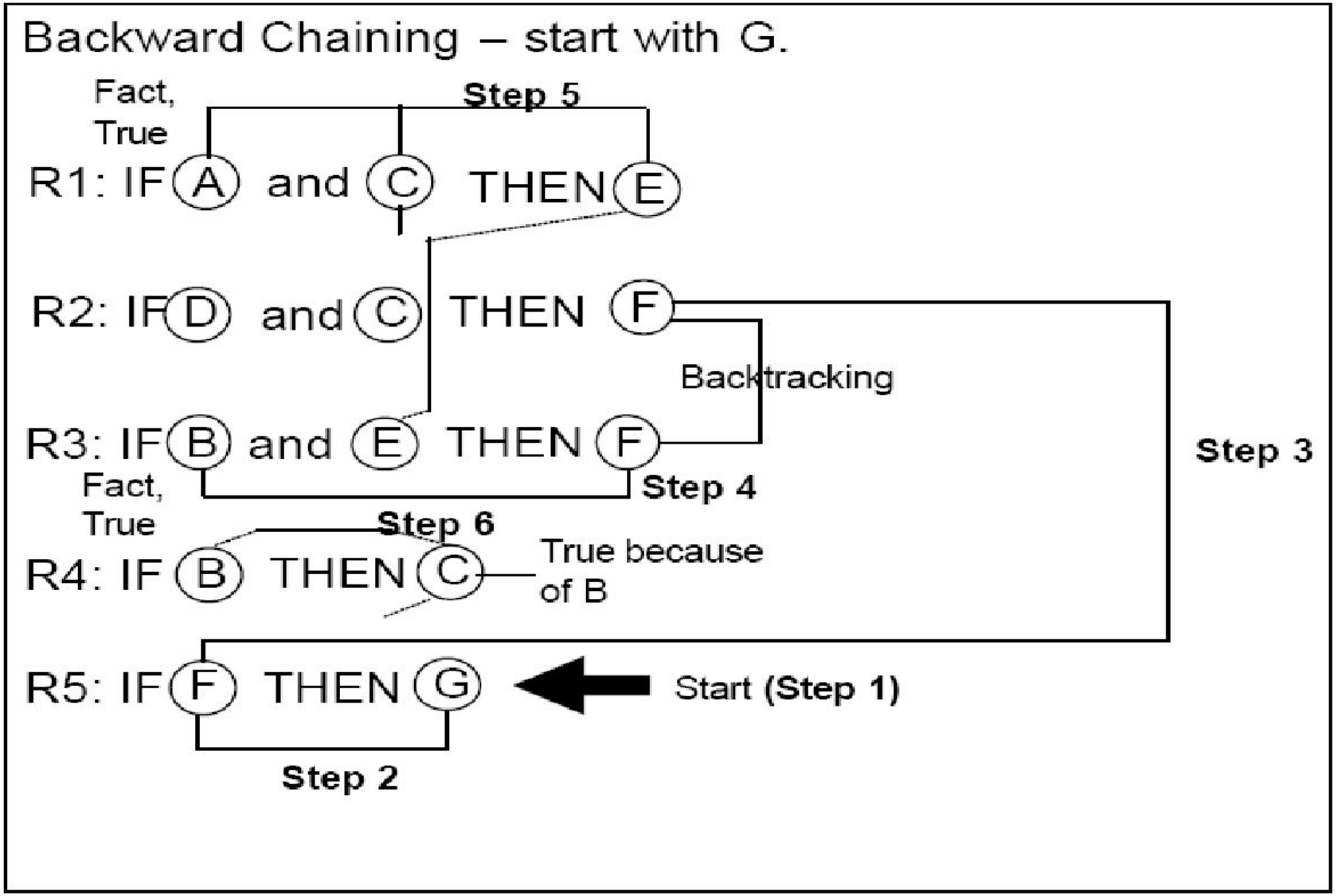
FAKTA YANG ADA:

- R3 : IF seseorang berusia < 30 tahun AND dia berinvestasi pada bidang sekuritas THEN dia sebaiknya berinvestasi pada saham pertumbuhan
- R4 : IF seseorang berusia < 30 tahun dan > 22 tahun THEN dia berpendidikan college
- R5 : IF seseorang ingin berinvestasi pada saham pertumbuhan THEN saham yang dipilih adalah saham IBM.

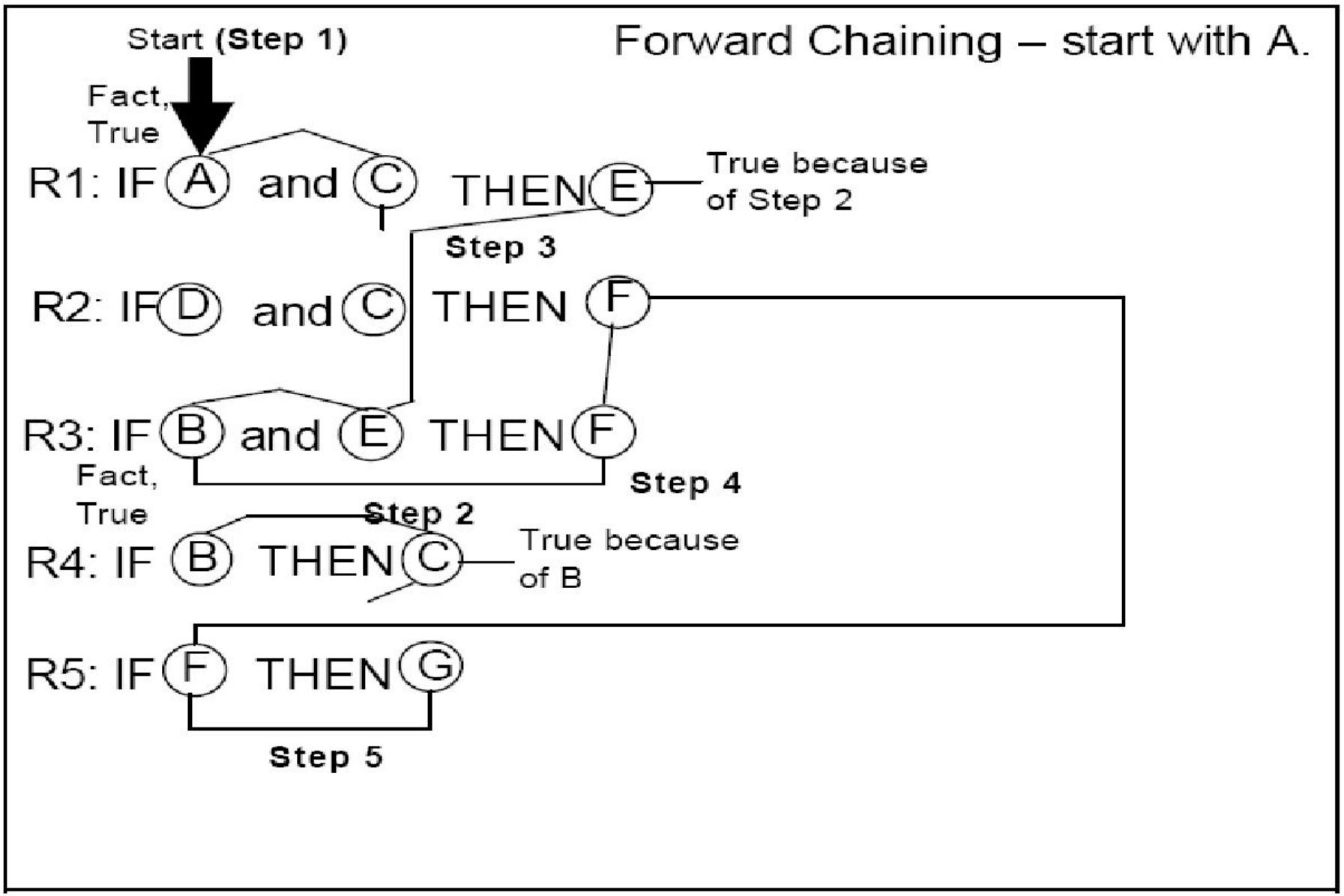
Rule simplification:

- – R1: IF A and C, THEN E
- – R2: IF D and C, THEN F
- – R3: IF B and E, THEN F
- – R4: IF B, THEN C
- – R5: IF F, THEN G

Solusi dengan Backward Chaining



Solusi dengan Forward Chaining



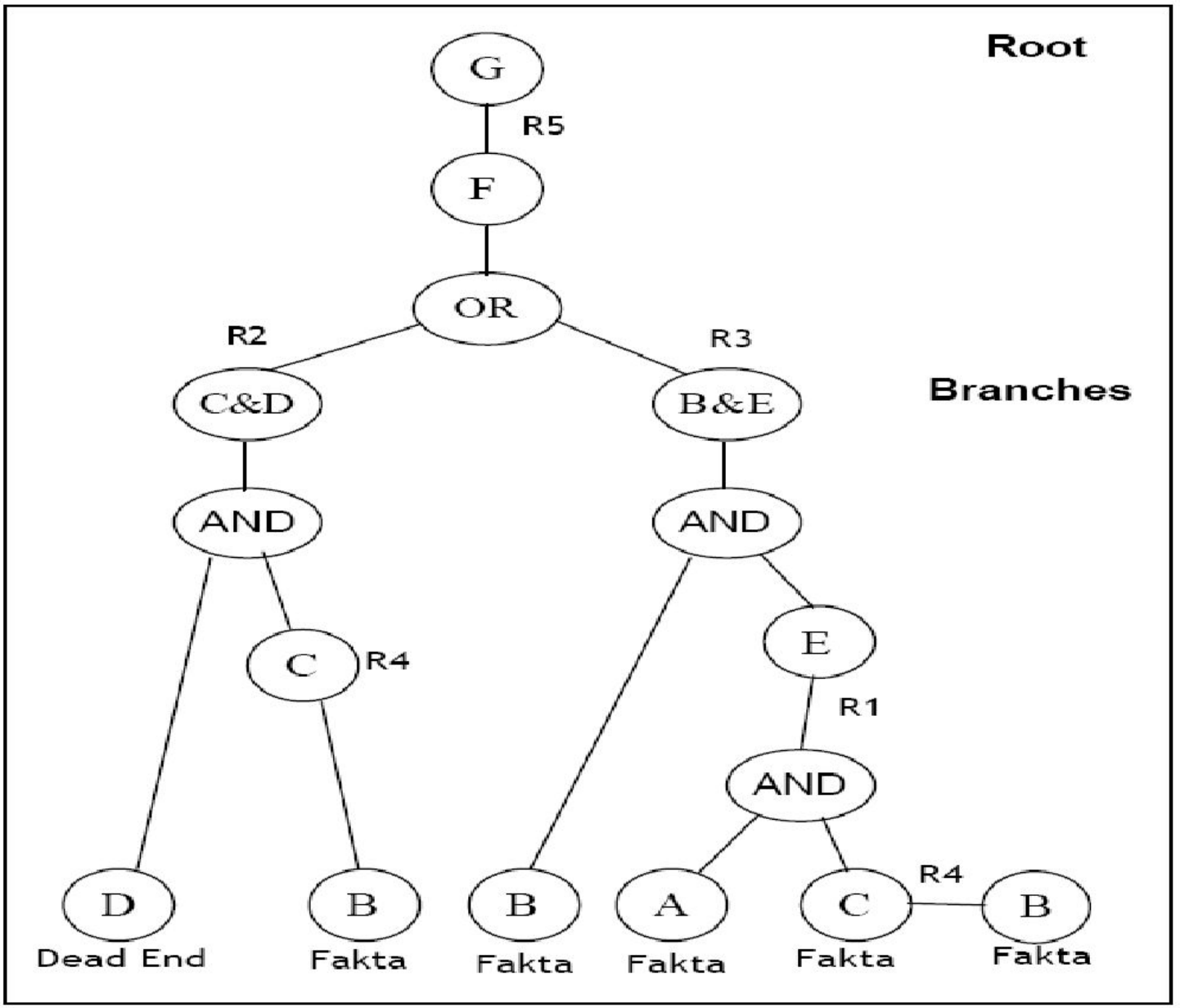
- Inferensi dengan rules (sebagaimana juga dengan Logika) dapat sangat efektif, tapi terdapat beberapa keterbatasan pada teknik-teknik tersebut.
- Misalnya, perhatikan contoh berikut:
  - Proposisi 1 : Semua burung dapat terbang
  - Proposisi 2 : Burung Unta (Kasuari) adalah burung
- Konklusi : Burung Unta dapat terbang
- Konklusi tersebut adalah valid, tetapi dalam kenyataannya adalah salah, karena burung unta tidak dapat terbang. Untuk kasus seperti ini maka terkadang kita harus menggunakan teknik inferensi yang lain.

Fungsi dari Inference Engine

1. *Fire* the rules
2. Memberikan pertanyaan pada user
3. Menambahkan jawaban pada *Working Memory* (*Blackboard*)
4. Mengambil fakta baru dari suatu rule (dari hasil inferensi)
5. Menambahkan fakta baru tersebut pada *working memory*
6. Mencocokan fakta pada working memory dengan rules
7. Jika ada yang cocok (*matches*), maka *fire rules* tersebut
8. Jika ada dua rule yang cocok, cek dan pilih rule mana yang menghasilkan *goal* yang diinginkan
9. *Fire the lowest-numbered unfired rule*

INFERENCE TREE (POHON INFERENSI)

- Penggambaran secara skematik dari proses inferensi
- Sama dengan *decision tree*
- Inferencing: *tree traversal*
- Advantage: Panduan untuk *Explanations* ***Why*** dan ***How***



Gambar Inference Tree dari proses backward chaining

METODE INFERENSI YANG LAIN

- Inferensi dengan Frame
- *Model Based Reasoning*
- *Case Based Reasoning*

EXPLANATION

- *Human experts* memberikan justifikasi dan penjelasan (*explain*) dari apa yang mereka lakukan
- ES harus dapat melakukan hal yang sama
- ***Explanation***: disediakan oleh ES untuk mengklarifikasi proses *reasoning,* rekomendasi, dan tindakan lainnya (mis: *asking a question*)
- *Explanation facility (justifier)*
- **Tujuan Explanation**
  - Membuat sistem menjadi lebih *intelligible*
  - Menjelaskan situasi yang *unanticipated* (tidak terantisipasi)
  - Memuaskan psikologis user dan/atau *social needs*
  - Klarifikasi dari asumsi yang diperoleh pada saat sistem beroperasi

Dua Explanation Dasar

- **Why** - *Why* (mengapa) sebuah fakta dibutuhkan?
  - Pertanyaan ‘Why” dapat diajukan oleh user pada saat diberikan sebuah pertanyaan untuk dijawab, untuk mengetahui alasan mengapa informasi itu diperlukan.
  - Misalnya, kita lihat pada kasus sebelumnya. Bisa saja muncul dialog sbb:

    COMPUTER: *Berapa pendapatan Anda per tahun?*

    CLIENT: *Why?*

Dua Explanation Dasar

- Komputer akan merespon dengan menunjukkan rule R2. Tetapi secara ideal jawaban komputer harus bisa sbb:
- **COMPUTER**:
  - *Untuk mengecek R2, Saya ingin tahu apakah pendapatan Anda per tahun diatas $40000. Jika itu benar, saya akan membuat kesimpulan: karena anda berpendidikan pada level college, anda sebaiknya berinvestasi pada saham pertumbuhan*

Dua Explanation Dasar (lanj..)

- **How** – Menjelaskan bagaimana suatu konklusi atau rekomendasi diperoleh
  - Sistem sederhana – hanya pada konklusi final (terakhir)
  - Sistem yang kompleks menyediakan *chain of rules* yang digunakan untuk mencapai konklusi.

Misalnya, dari kasus sebelumnya, komputer memberikan konklusi:

- **COMPUTER**: *Inverstasi pada saham IBM*
- **CLIENT**: *How?*
- **COMPUTER**: (Akan menunjukkan urutan *rules* yang digunakan/*fires* : R5-R3-R1-R4)
- **IDEAL COMPUTER**: *Anda memiliki uang $10000 untuk investasi dan anda berusia kurang dari 30 tahun, kemudian mengacu pada R4 bahwa anda berpendidikan pada level college. Untuk investor muda seperti anda jika ingin berinvestasi pada bidang saham pertumbuhan, maka saham IBM adalah pilihan terbaik yang bisa anda coba.*

INFERENSI DENGAN UNCERTAINTY

- **Representasi Uncertainty**
- Numerik
  - Menggunaan penskalaan numerik.
  - 0 : untuk *complete uncertainty* (FALSE)
  - 1 atau 100 : *complete certainty* (TRUE)
- Grafik : Horizontal bar
- Simbolik : *Likert Scale: Ranking, Ordinal, Cardinal*

INFERENSI DENGAN UNCERTAINTY

- **KOMBINASI CERTAINTY FACTOR**
  - Certainty Factors (CF) mengekspreksikan tingkat kepercayaan terhadap suatu event (atau fakta atau hipotesis) berdasarkan *evidence* (atau *the expert's assessment*)
- **Kombinasi beberapa CF dalam satu rule**
- **Operator AND:**
  - IF *inflation is high*, CF = 50%, (A), AND
  - IF *unemployment rate is above 7 percent*, CF = 70%, (B), AND
  - IF *bond prices decline*, CF = 100%, (C)
  - THEN *stock prices decline*

INFERENSI DENGAN UNCERTAINTY

- CF dari konklusi adalah nilai CF minimum pada bagian IF
- **CF(A, B, and C) = minimum(CF(A),CF(B),CF(C))**
- CF(A) adalah yang minimum → 50%, sehingga CF untuk "*stock prices decline*" adalah **50%**

INFERENSI DENGAN UNCERTAINTY

Operator OR:

- IF *inflation is low*, CF = 70%, (A), AND
- IF *bond prices are high*, CF = 85%, (B)
- THEN *stock prices will be high*

- CF dari konklusi adalah nilai CF maksimum pada bagian IF

  CF(A or B) = maksimum (CF(A),CF(B))

- CF(B) adalah yang maksimum → 85%, sehingga CF untuk "*stock prices will be high*" adalah **85%**

INFERENSI DENGAN UNCERTAINTY

Kombinasi dua atau lebih Rules

- Misalnya:
  - **R1**: IF *the inflation rate is less than 5 percent*, THEN *stock market prices go up* (CF = 0.7)
  - **R2**: IF *unemployment level is less than 7 percent*, THEN *stock market prices go up* (CF = 0.6)
  - Inflation rate = 4 percent and the unemployment level = 6.5 percent

Efek Kombinasi:

- **CF(R1,R2) = CF(R1) + CF(R2)[1 - CF(R1)]; or**
- **CF(R1,R2) = CF(R1) + CF(R2) - CF(R1) XCF(R2)**
- Jika CF(R1) = 0.7 AND CF(R2) = 0.6, maka:

  CF(R1,R2) = 0.7 + 0.6(1 - 0.7) = 0.7 + 0.6(0.3) = 0.88
- Jadi peluang *stock prices will increase* adalah sebesar 88 persen

- **Jika ada rule ketiga yang ditambahkan, maka:**
- **CF(R1,R2,R3) = CF(R1,R2) + CF(R3) [1 - CF(R1,R2)],** Misalnya:
  - R3: IF *bond prices increases*, THEN *stock market prices go up* (CF = 0.85)
- Jika diasumsikan semua rule pada bagian IF bernilai TRUE maka, peluang *stock prices will increase* adalah:

  CF(R1,R2,R3) = 0.88 + 0.85 (1 - 0.88) = 0.88 + 0.85 (.12) = 0.982
- **catatan:**
  - *bond* = surat obligasi
  - *stock* = saham
  - *unemployment* = PHK