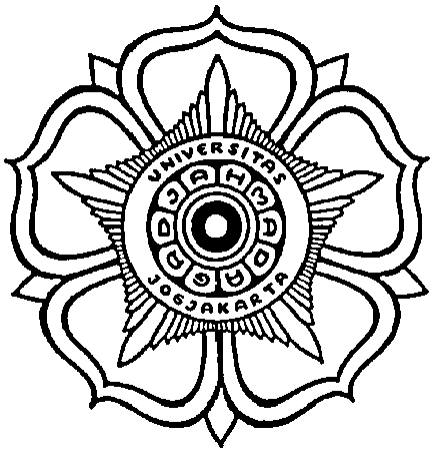
DIKTAT MATA KULIAH

KECERDASAN BUATAN



Oleh:

Ir. Balza Achmad, M.Sc.E.

Jurusan Teknik Fisika
Fakultas Teknik
Universitas Gadjah Mada

Yogyakarta

BAB I. KECERDASAN BUATAN

A. Pendahuluan

Di masa kini, Kecerdasan Buatan (*Artificial Intelligence*, AI) telah menjadi wacana umum yang sangat penting dan jamak dijumpai. Namun masih banyak menyisakan pertanyaan skeptis tentang 'mesin berfikir': "*Betulkah sebuah mesin dapat benar-benar berfikir dengan dirinya sendiri?*", atau "*Jika benar-benar dapat berfikir sendiri, apakah proses berfikirnya sama dengan kita?*", dan "*Seberapa handal?*".

1. Sejarah Kecerdasan Buatan

Di awal abad 20, seorang penemu Spanyol, Torres y Quevedo, membuat sebuah mesin yang dapat men'skak-mat' raja lawannya dengan sebuah ratu dan raja. Perkembangan secara sistematis kemudian dimulai segera setelah diketemukannya komputer digital. Artikel ilmiah pertama tentang Kecerdasan Buatan ditulis oleh Alan Turing pada tahun 1950, dan kelompok riset pertama dibentuk tahun 1954 di Carnegie Mellon University oleh Allen Newell and Herbert Simon. Namun bidang Kecerdasan Buatan baru dianggap sebagai bidang tersendiri di konferensi Dartmouth tahun 1956, di mana 10 peneliti muda memimpikan mempergunakan komputer untuk memodelkan bagaimana cara berfikir manusia. Hipotesis mereka adalah: "*Mekanisme berfikir manusia dapat secara tepat dimodelkan dan disimulasikan pada komputer digital*", dan ini yang menjadi landasan dasar Kecerdasan Buatan.

2. Definisi Kecerdasan Buatan

Tidak ada kesepakatan mengenai definisi Kecerdasan Buatan, di antaranya adalah:

a. Sebuah studi tentang bagaimana membuat komputer mengerjakan sesuatu yang dapat dikerjakan manusia (Rich, 1991)
b. Cabang ilmu komputer yang mempelajari otomatisasi tingkah laku cerdas (Setiawan, 1993)
c. Suatu perilaku sebuah mesin yang jika dikerjakan oleh manusia akan disebut cerdas (Turing, et. al, 1996)

Kebanyakan ahli setuju bahwa Kecerdasan Buatan berhubungan dengan 2 ide dasar. Pertama, menyangkut studi proses berfikir manusia, dan kedua, berhubungan dengan merepresentasikan proses tersebut melalui mesin (komputer, robot, dll)

Kemampuan untuk *problem solving* adalah salah satu cara untuk mengukur kecerdasan dalam berbagai konteks. Terlihat di sini bahwa mesin cerdas akan diragukan untuk dapat melayani keperluan khusus jika tidak mampu menangani permasalahan remeh/kecil yang biasa dikerjakan orang secara rutin. Terdapat beberapa alasan untuk memodelkan performa manusia dalam hal ini:

a. Untuk menguji teori psikologis dari performa manusia
b. Untuk membuat komputer dapat memahami penalaran (*reasoning*) manusia
c. Untuk membuat manusia dapat memahami penalaran komputer
d. Untuk mengeksploitasi pengetahuan apa yang dapat diambil dari manusia

Menurut Winston dan Prendergast (1984), tujuan dari Kecerdasan Buatan adalah:

a. Membuat mesin menjadi lebih pintar.
b. Memahami apakah kecerdasan (*intelligence*) itu.
c. Membuat mesin menjadi lebih berguna.

3. Kecerdasan

Dari kamus, arti kecerdasan adalah: kemampuan untuk mengerti/memahami (*The faculty of understanding*). Perilaku cerdas dapat ditandai dengan:

a. Belajar atau mengerti dari pengalaman
b. Memecahkan hal yang bersifat mendua atau kontradiktif
c. Merespon situasi baru dengan cepat (fleksibel)
d. Menggunakan alasan untuk memecahkan problem secara efektif
e. Berurusan dengan situasi yang membingungkan
f. Memahami dengan cara biasa/rasional
g. Menerapkan pengetahuan untuk memanipulasi lingkungan
h. Mengenali elemen penting pada suatu situasi

Sebuah ujian yang dapat dilakukan untuk menentukan apakah sebuah komputer/ mesin menunjukkan perilaku cerdas didesain oleh Alan Turing. Tes Turing menyatakan sebuah mesin dikatakan pintar hanya apabila seorang pewawancara (manusia) yang berbicara dengan orang lain dan mesin yang dua-duanya tidak terlihat olehnya, tidak mampu menentukan mana yang manusia dan mana yang mesin, meskipun dia telah berulang-ulang melontarkan pertanyaan yang sama.

4. Perbandingan AI dengan program komputer konvensional

Program komputer konvensional prosesnya berbasis algoritma, yakni formula matematis atau prosedur sekuensial yang mengarah kepada suatu solusi. Algoritma tersebut dikonversi ke program komputer yang memberitahu komputer secara pasti instruksi apa yang harus dikerjakan. Algoritma yang dipakai kemudian menggunakan data seperti angka, huruf, atau kata untuk menyelesaikan masalah.

Perangkat lunak AI berbasis representasi serta manipulasi simbolik. Di sini simbol tersebut berupa huruf, kata, atau angka yang merepresentasikan obyek, proses dan hubungan keduanya. Sebuah obyek bisa jadi seorang manusia, benda, pikiran, konsep, kejadian, atau pernyataan suatu fakta. Menggunakan simbol, kita dapat menciptakan basis pengetahuan yang berisi fakta, konsep, dan hubungan di antara keduanya. Kemudian beberapa proses dapat digunakan untuk memanipulasi simbol tersebut untuk menghasilkan nasehat atau rekomendasi untuk penyelesaian suatu masalah. Perbedaan dasar antara AI dengan program komputer konvensional diberikan dalam Tabel I-1.

Tabel I-1. Perbandingan antara AI dan Program Konvensional

Aspek	AI	Program konvensional
Pemrosesan	Sebagian besar simbolik	Algoritmik
Input	Tidak harus lengkap	Harus lengkap
Pendekatan pencarian	Sebagian besar heuristik	Algoritma
Penjelasan/eksplanasi	Tersedia	Biasanya tidak tersedia
Fokus	Pengetahuan	Data
Pemeliharaan & peningkatan	Relatif mudah	Biasanya sulit
Kemampuan berpikir secara logis	Ada	Tidak ada

B. Cabang Kecerdasan Buatan

Pencarian. Program AI seringkali harus mengevaluasi kemungkinan yang jumlahnya banyak sekali, misalnya kemungkinan langkah dalam permainan catur atau penyimpulan dari program untuk membuktikan suatu teori.

Pengenalan Pola.

Representasi, yakni bagaimana merepresentasikan/menuliskan fakta-fakta yang ada ke dalam simbul-simbul atau bahasa logika matematis.

Inferensi.

Pengetahuan dan penalaran yang masuk akal (*common sense knowledge and reasoning*).

Belajar dari pengalaman.

Perencanaan. Program perencanaan bermula dari fakta-fakta umum (terutama fakta mengenai efek dari suatu aksi), fakta tentang situasi yang khusus, dan suatu pernyataan tentang tujuan. Dari sini kemudian dibuat sebuah strategi untuk mencapai tujuan tersebut. Secara umum, biasanya strategi tersebut berupa urut-urutan aksi.

Epistemologi, yakni studi tentang sumber, sifat, dan keterbatasan pengetahuan yang digunakan untuk pemecahan masalah.

Ontologi, ilmu tentang keberadaan dan realitas.

Heuristik, yaitu suatu cara atau teknik untuk mencoba menemukan suatu benda/ide.

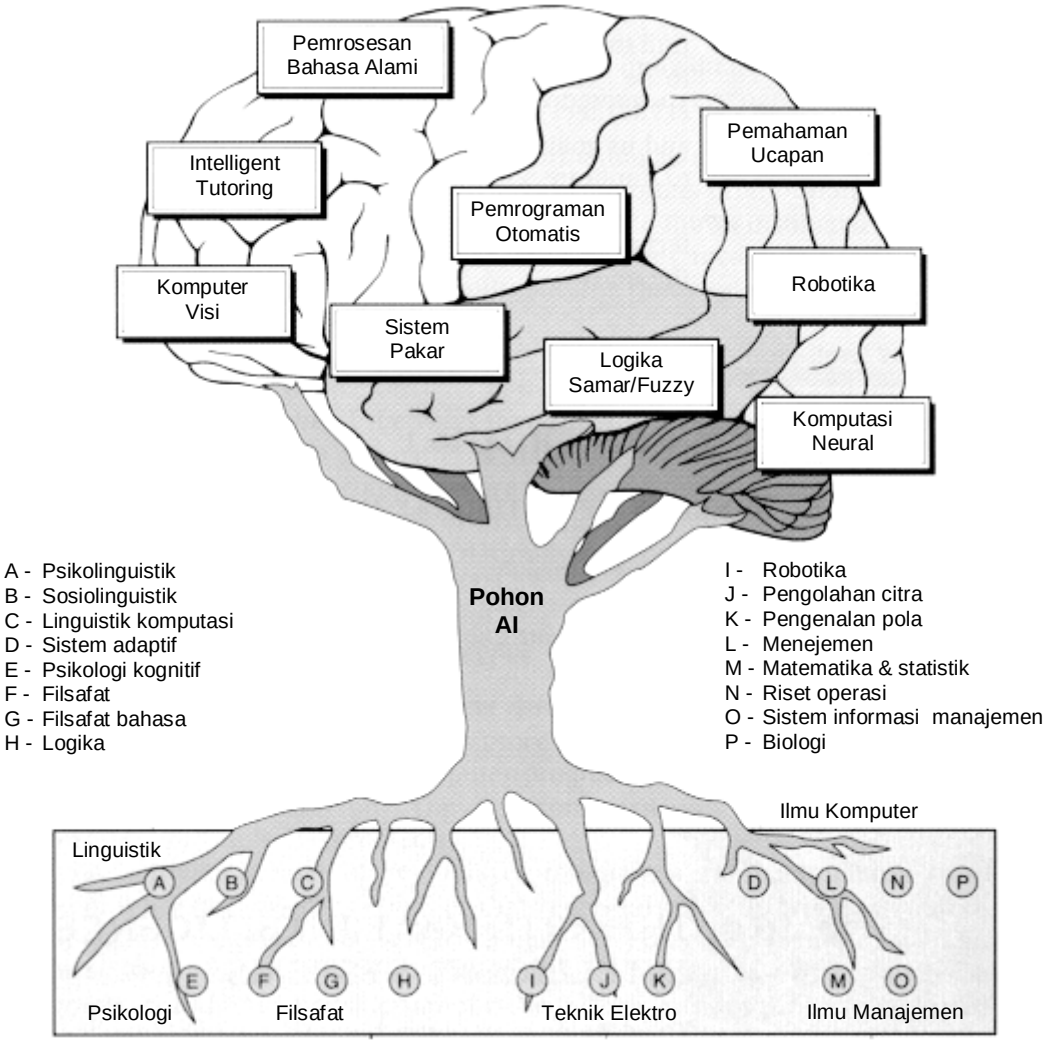
C. Bidang Aplikasi Kecerdasan Buatan

Penerapan Kecerdasan Buatan meliputi berbagai bidang seperti ditunjukkan pada bagian akar pohon AI dalam Gambar I-1, antara lain: Bahasa/linguistik, Psikologi, Filsafat, Teknik Elektro, Ilmu Komputer, dan Ilmu Manajemen. Sedangkan sistem cerdas yang banyak dikembangkan saat ini adalah:

Sistem Pakar (*Expert Systemi)*, yaitu program konsultasi (*advisory*) yang mencoba menirukan proses penalaran seorang pakar/ahli dalam memecahkan masalah yang rumit. Sistem Pakar merupakan aplikasi AI yang paling banyak. Lebih detil tentang Sistem Pakar akan diberikan dalam bab berikutnya.

Pemrosesan Bahasa Alami (*Natural Language Processing*), yang memberi kemampuan pengguna komputer untuk berkomunikasi dengan komputer dalam bahasa mereka sendiri (bahasa manusia). Sehingga komunikasi dapat dilakukan dengan cara percakapan alih-alih menggunakan perintah yang biasa digunakan dalam bahasa komputer biasa. Bidang ini dibagi 2 lagi:

a. Pemahaman bahasa alami, yang mempelajari metode yang memungkinkan komputer mengerti perintah yang diberikan dalam bahasa manusia biasa. Dengan kata lain, komputer dapat memahami manusia.
b. Pembangkitan bahasa alami, sering disebut juga sintesa suara, yang membuat komputer dapat membangkitkan bahasa manusia biasa sehingga manusia dapat memahami komputer secara mudah.



Gambar I-1. Pohon Kecerdasan Buatan dan aplikasi utamanya

Pemahaman Ucapan/Suara (*Speech/Voice Understanding*), adalah teknik agar komputer dapat mengenali dan memahami bahasa ucapan. Proses ini mengijinkan seseorang berkomunikasi dengan komputer dengan cara berbicara kepadanya. Istilah "pengenalan suara" mengandung arti bahwa tujuan utamanya adalah mengenai kata yang diucapkan tanpa harus tahu artinya, di mana bagian itu merupakan tugas "pemahaman suara". Secara umum prosesnya adalah usaha untuk menerjemahkan apa yang diucapkan seorang manusia menjadi kata-kata atau kalimat yang dapat dimengerti oleh komputer.

Sistem Sensor dan Robotika. Sistem sensor, seperti sistem visi dan pencitraan, serta sistem pengolahan sinyal, merupakan bagian dari robotika. Sebuah robot, yaitu perangkat elektromekanik yang diprogram untuk melakukan tugas manual, tidak semuanya merupakan bagian dari AI. Robot yang hanya melakukan aksi yang telah diprogramkan dikatakan sebagai robot bodoh yang tidak lebih pintar dari lift. Robot yang cerdas biasanya

mempunyai perangkat sensor, seperti kamera, yang mengumpulkan informasi mengenai operasi dan lingkungannya. Kemudian bagian AI robot tersebut menerjemahkan informasi tadi dan merespon serta beradaptasi jika terjadi perubahan lingkungan.

Komputer Visi, merupakan kombinasi dari pencitraan, pengolahan citra, pengenalan pola serta proses pengambilan keputusan. Tujuan utama dari komputer visi adalah untuk menerjemahkan suatu pemandangan. Komputer visi banyak dipakai dalam kendali kualitas produk industri.

Intelligent Tutoring/Intelligent Computer-Aided Instruction, adalah komputer yang mengajari manusia. Belajar melalui komputer sudah lama digunakab, namun dengan menambahkan aspek kecerdasan di dalamnya, dapat tercipta komputer “guru” yang dapat mengatur teknik pengajarannya untuk menyesuaikan dengan kebutuhan “murid” secara individiual. Sistem ini juga mendukung pembelajaran bagi orang yang mempunyai kekurangan fisik atau kelemahan belajar.

Mesin Belajar (*Machine Learning*), yang berhubungan dengan sekumpulan metode untuk mencoba mengajari/melatih komputer untuk memecahkan masalah atau mendukung usaha pemecahan masalah dengan menganalisa kasus-kasus yang telah terjadi. Dua metode mesin belajar yang paling populer adalah **Komputasi Neural** dan **Logika Samar** yang akan dipelajari lebih dalam di bab-bab berikutnya.

Aplikasi lain dari AI misalnya untuk merangkum berita, pemrograman komputer secara otomatis, atau menerjemahkan dari suatu bahasa ke bahasa yang lain, serta aplikasi dalam permainan (Ingat pertandingan catur antara Grand Master Anatoly Karpov dengan komputer Deep Thought dari IBM).

BAB II. SISTEM PAKAR

A. Pendahuluan

Ketika hendak membuat suatu keputusan yang komplek atau memecahkan masalah, seringkali kita meminta nasehat atau berkonsultasi dengan seorang pakar atau ahli. Seorang **pakar** adalah seseorang yang mempunyai pengetahuan dan pengalaman spesifik dalam suatu bidang; misalnya pakar komputer, pakar uji tak merusak, pakar politik dan lain-lain. Semakin tidak terstruktur situasinya, semakin mengkhusus (dan mahal) konsultasi yang dibutuhkan.

Sistem Pakar (*Expert System*) adalah usaha untuk menirukan seorang pakar. Biasanya Sistem Pakar berupa perangkat lunak pengambil keputusan yang mampu mencapai tingkat performa yang sebanding seorang pakar dalam bidang problem yang khusus dan sempit. Ide dasarnya adalah: kepakaran ditransfer dari seorang pakar (atau sumber kepakaran yang lain) 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. Sistem Pakar malahan terkadang lebih baik unjuk kerjanya daripada seorang pakar manusia!

Kepakaran (*expertise*) adalah pengetahuan yang ekstensif (meluas) dan spesifik yang diperoleh melalui rangkaian pelatihan, membaca, dan pengalaman. Pengetahuan membuat pakar dapat mengambil keputusan secara lebih baik dan lebih cepat daripada non-pakar dalam memecahkan problem yang kompleks. Kepakaran mempunyai sifat berjenjang, pakar top memiliki pengetahuan lebih banyak daripada pakar yunior.

Tujuan Sistem Pakar adalah untuk mentransfer kepakaran dari seorang pakar ke komputer, kemudian ke orang lain (yang bukan pakar). Proses ini tercakup dalam rekayasa pengetahuan (*knowledge engineering*) yang akan dibahas kemudian.

B. Manfaat dan Keterbatasan Sistem Pakar

1. Manfaat Sistem Pakar

Mengapa Sistem Pakar menjadi sangat populer? Hal ini disebabkan oleh sangat banyaknya kemampuan dan manfaat yang diberikan oleh Sistem Pakar, di antaranya:

a. Meningkatkan output dan produktivitas, karena Sistem Pakar dapat bekerja lebih cepat dari manusia.
b. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
c. Mampu menangkap kepakaran yang sangat terbatas.
d. Dapat beroperasi di lingkungan yang berbahaya.
e. Memudahkan akses ke pengetahuan.
f. Handal. Sistem Pakar tidak pernah menjadi bosan dan kelelahan atau sakit. Sistem Pakar juga secara konsisten melihat semua detil dan tidak akan melewatkan informasi yang relevan dan solusi yang potensial.
g. Meningkatkan kapabilitas sistem terkomputerisasi yang lain. Integrasi Sistem Pakar dengan sistem komputer lain membuat lebih efektif, dan mencakup lebih banyak aplikasi .
h. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, Sistem Pakar dapat bekerja dengan inofrmasi yang tidak lengkap. Pengguna dapat merespon dengan: “tidak tahu” atau “tidak yakin” pada satu atau lebih pertanyaan selama konsultasi, dan Sistem Pakar tetap akan memberikan jawabannya.
i. Mampu menyediakan pelatihan. Pengguna pemula yang bekerja dengan Sistem Pakar akan menjadi lebih berpengalaman. Fasilitas penjelas dapat berfungsi sebagai guru.
j. Meningkatkan kemampuan problem solving, karena mengambil sumber pengetahuan dari banyak pakar.
k. Meniadakan kebutuhan perangkat yang mahal.
l. Fleksibel.

2. Keterbatasan Sistem Pakar

Metodologi Sistem Pakar yang ada tidak selalu mudah, sederhana dan efektif. Berikut adalah keterbatasan yang menghambat perkembangan Sistem Pakar:

a. Pengetahuan yang hendak diambil tidak selalu tersedia.
b. Kepakaran sangat sulit diekstrak dari manusia.
c. Pendekatan oleh setiap pakar untuk suatu situasi atau problem bisa berbeda-beda, meskipun sama-sama benar.

d. Adalah sangat sulit bagi seorang pakar untuk mengabstraksi atau menjelaskan langkah mereka dalam menangani masalah
e. Pengguna Sistem Pakar mempunyai batas kognitif alami, sehingga mungkin tidak bisa memanfaatkan sistem secara maksimal.
f. Sistem Pakar bekerja baik untuk suatu bidang yang sempit.
g. Banyak pakar yang tidak mempunyai jalan untuk mencek apakah kesimpulan mereka benar dan masuk akal.
h. Istilah dan jargon yang dipakai oleh pakar dalam mengekspresikan fakta seringkali terbatas dan tidak mudah dimengerti oleh orang lain.
i. Pengembangan Sistem Pakar seringkali membutuhkan perekayasa pengetahuan (*knowledge engineer*) yang langka dan mahal.
j. Kurangnya rasa percaya pengguna menghalangi pemakaian Sistem Pakar.
k. Transfer pengetahuan dapat bersifat subyektif dan bias.

C. Komponen Sistem Pakar

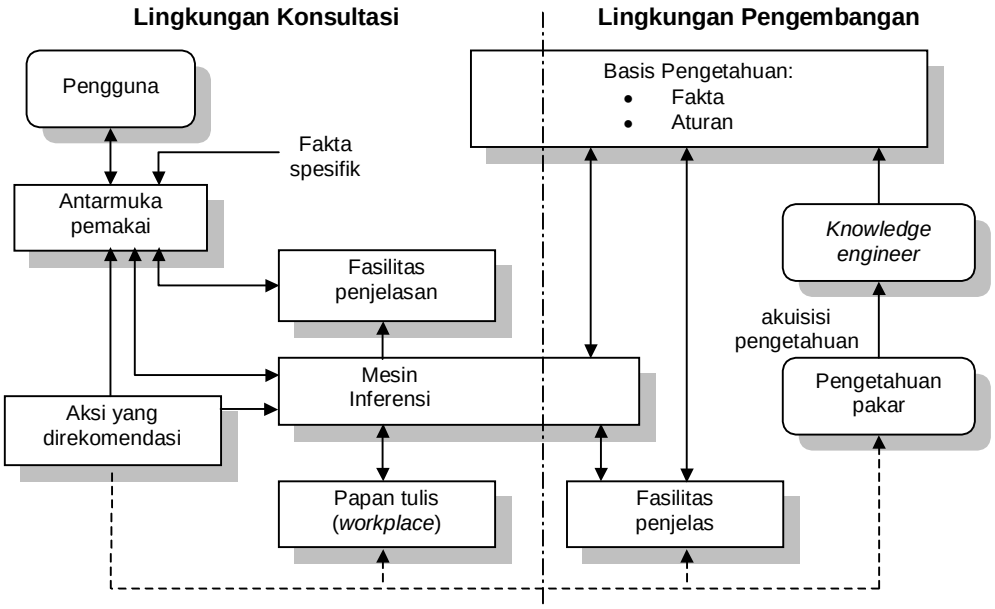
Secara umum, Sistem Pakar biasanya terdiri atas beberapa komponen yang masing-masing berhubungan seperti terlihat pada Gambar II-1.

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

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

Mesin Inferensi (*Inference Engine*), 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:

1. Menentukan aturan mana akan dipakai
2. Menyajikan pertanyaan kepada pemakai, ketika diperlukan.
3. Menambahkan jawaban ke dalam memori Sistem Pakar.
4. Menyimpulkan fakta baru dari sebuah aturan
5. Menambahkan fakta tadi ke dalam memori.



Gambar II-1. Struktur skematis sebuah Sistem Pakar

Papan Tulis (*Blackboard/Workplace*), adalah memori/lokasi untuk bekerja dan menyimpan hasil sementara. Biasanya berupa sebuah basis data.

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*).

Subsistem Penjelasan (*Explanation Facility*). Kemampuan untuk menjejak (*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, misalnya:

1. “Mengapa pertanyaan tersebut anda tanyakan?”
2. “Seberapa yakin kesimpulan tersebut diambil?”
3. “Mengapa alternatif tersebut ditolak?”
4. “Apa yang akan dilakukan untuk mengambil suatu kesimpulan?”
5. “Fakta apalagi yang diperlukan untuk mengambil kesimpulan akhir?”

Sistem Penghalusan Pengetahuan (*Knowledge Refining System*). Seorang pakar mempunyai sistem penghalusan 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 alasan keberhasilan atau kegagalan pengambilan kesimpulan, serta memperbaiki basis pengetahuannya.

D. Pembangunan Sebuah Sistem Pakar

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*) yang dapat di*download* dari internet.

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:

a. Biaya yang diperlukan untuk pembangunan Sistem Pakar ditentukan oleh kebutuhan untuk memperoleh solusi. Sehingga harus ada perhitungan yang realistis untuk *cost and benefit*.
b. 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.
c. Problem yang ada dapat diselesaikan dengan teknik penalaran simbolik, dan tidak membutuhkan kemampuan fisik.
d. 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.
e. 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.

f. 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.

g. 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 relatif singkat (misalnya paling lama 1 jam).

2. Rekayasa Pengetahuan (*Knowledge Engineering*)

Proses dalam rekayasa pengetahuan meliputi (Gambar II-2):

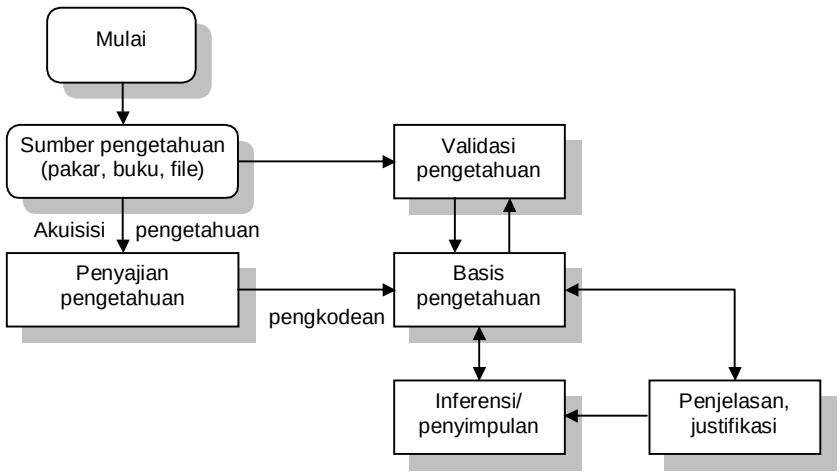
a. Akuisisi pengetahuan, yaitu bagaimana memperoleh pengetahuan dari pakar atau sumber lain (sumber terdokumentasi, buku, sensor, file komputer, dll.).

b. Validasi pengetahuan, untuk menjaga kualitasnya misalnya dengan uji kasus.

c. Representasi pengetahuan, yaitu bagaimana mengorganisasi pengetahuan yang diperoleh, mengkodekan dan menyimpannya dalam suatu basis pengetahuan.

d. Penyimpulan pengetahuan, menggunakan mesin inferensi yang mengakses basis pengetahuan dan kemudian melakukan penyimpulan.

e. Transfer pengetahuan (penjelasan). Hasil inferensi berupa nasehat, rekomendasi, atau jawaban, kemudian dijelaskan ke pengguna oleh subsistem penjelas.



Gambar II-2. Proses dalam rekayasa pengetahuan

3. Partisipan Dalam Proses Pengembangan

Pakar, yaitu seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat. Pakar menyediakan pengetahuan tentang bagaimana nantinya Sistem Pakar bekerja.

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.

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.

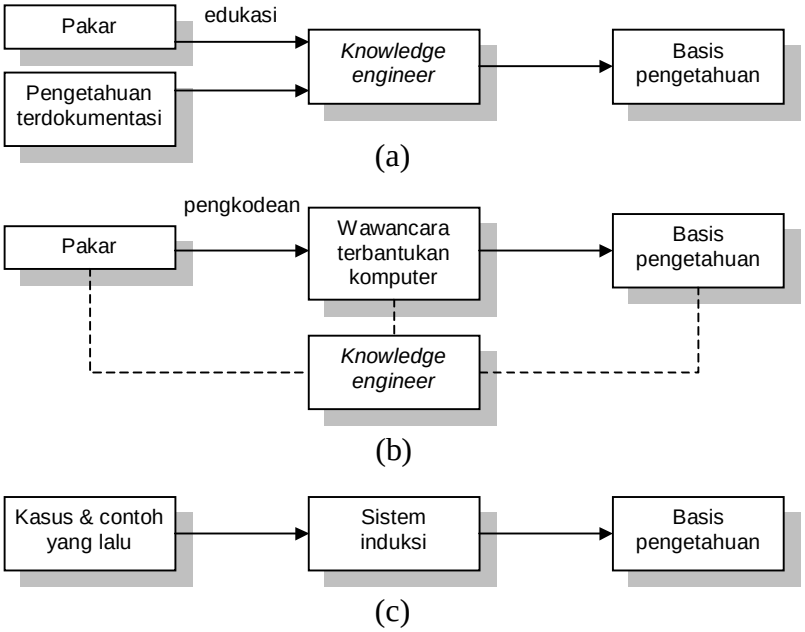
4. Akuisisi Pengetahuan

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 (Gambar II-3):

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) 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.



Gambar II-3. Metode akuisisi pengetahuan (a) manual (b) akuisisi terkendali-pakar (c) induksi

E. 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.

1. Aturan Produk (*Production Rules*)

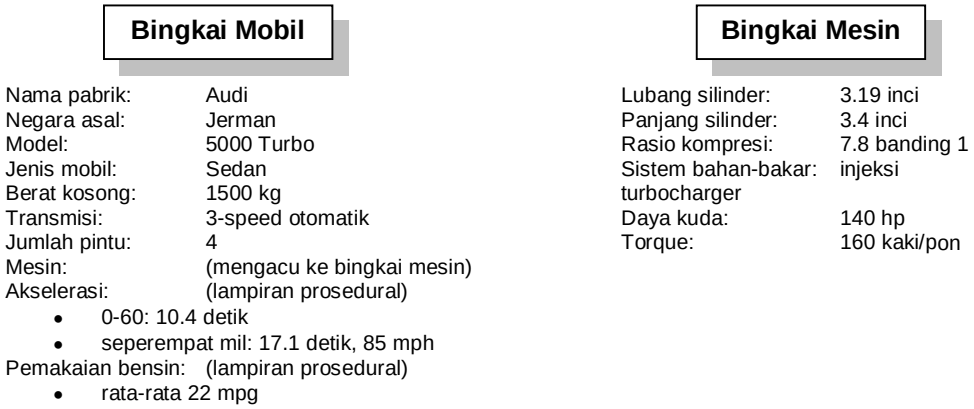
Di sini pengetahuan disajikan dalam aturan-aturan yang berbentuk pasangan keadaan-aksi (*condition-action*): "JIKA keadaan terpenuhi atau terjadi MAKA suatu aksi akan terjadi". Sistem Pakar yang basis pengetahuannya melulu disajikan dalam bentuk aturan produk disebut sistem berbasis-aturan (*rule-based system*). Kondisi dapat terdiri atas banyak bagian, demikian pula dengan aksi. Urutan keduanya juga dapat dipertukarkan letaknya. Contohnya:

a. JIKA suhu berada di bawah 20ºC MAKA udara terasa dingin.
b. Udara terasa dingin JIKA suhu berada di bawah 20ºC.
c. JIKA suhu berada di bawah 20ºC ATAU suhu berada di antara 20-25ºC DAN angin bertiup cukup kencang MAKA udara terasa dingin.
d. Contoh dari MYCIN, Sistem Pakar untuk mendiagnosis dan merekomendasikan perlakuan yang tepat untuk infeksi darah tertentu:

IF the infection is pimary-bacteremia
AND the site of the culture is one of the sterile sites
AND the suspected portal of entry is the gastrointestinal tract
THEN there is suggestive evidence (0.7) that infection is bacteroid.

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. Contoh frame untuk menggambarkan sebuah mobil diberikan dalam Gambar II-4 berikut ini.



Gambar II-4. Bingkai pengetahuan untuk sebuah mobil

Contoh lain adalah bingkai untuk merepresentasikan pengetahuan mengenai gajah:

```
Mamalia
  subkelas:          Binatang
  berdarah_panas:    ya

Gajah
  subkelas:          Mamalia
  * warna:           abu-abu
  * ukuran:          besar

Clyde
  instance:          Gajah
  warna:             merah_muda
  pemilik:           Fred

Nellie:
  instance:          Gajah
  ukuran:            kecil
```

F. 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.

Sebagai contoh, lihat pada sekumpulan aturan sederhana berikut (Di sini kita memakai kata yang diawali huruf kapital untuk menyatakan suatu variabel. Pada sistem lain, mungkin dipakai cara lain, misalnya menggunakan awalan ? atau ^):

```
1. JIKA (mengajar X)
   DAN (mengoreksi_tugas X)
   MAKA TAMBAH (terlalu_banyak_bekerja X)
2. JIKA (bulan maret)
   MAKA TAMBAH (mengajar balza)
3. JIKA (bulan maret)
   MAKA TAMBAH (mengoreksi_tugas balza)
4. JIKA (terlalu_banyak_bekerja X)
   ATAU (kurang_tidur X)
   MAKA TAMBAH (mood_kurang_baik X)
5. JIKA (mood_kurang_baik X)
   MAKA HAPUS (bahagia X)
6. JIKA (mengajar X)
   MAKA HAPUS (meneliti X)
```

Kita asumsikan, pada awalnya kita mempunyai memori kerja yang berisi fakta berikut:

```
(bulan maret)
(bahagia balza)
(meneliti balza)
```

Sistem Pakar mula-mula akan memeriksa semua aturan yang ada untuk mengenali aturan manakah yang dapat memicu aksi, dalam hal ini aturan 2 dan 3. Sistem kemudian memilih

salah satu di antara kedua aturan tersebut dengan strategi penyelesaian konflik. Katakanlah aturan 2 yang terpilih, maka fakta `(mengajar balza)` akan ditambahkan ke dalam memori kerja. Keadaan memori kerja sekarang menjadi:

```
(mengajar balza)
(bulan maret)
(bahagia balza)
(meneliti balza)
```

Sekarang siklus dimulai lagi, dan kali ini aturan 3 dan 6 yang kondisinya terpenuhi. Katakanlah aturan 3 yang terpilih dan terpicu, maka fakta `(mengoreksi_tugas balza)` akan ditambahkan ke dalam memori kerja. Lantas pada siklus ketiga, aturan 1 terpicu, sehingga variabel X akan berisi (*bound to*) balza, dan fakta `(terlalu_banyak_bekerja balza)` ditambahkan, sehingga isi memori kerja menjadi:

```
(terlalu_banyak_bekerja balza)
(mengoreksi_tugas balza)
(mengajar balza)
(bulan maret)
(bahagia balza)
(meneliti balza)
```

Aturan 4 dan 6 dapat diterapkan. Misalkan aturan 4 yang terpicu, sehingga fakta `(mood_kurang_baik balza)` ditambahkan. Pada siklus berikutnya, aturan 5 terpilih dan dipicu, sehingga fakta `(bahagia balza)` dihapus dari memori kerja. Kemudian aturan 6 akan terpicu dan fakta `(meneliti balza)` dihapus pula dari memori kerja menjadi:

```
(mood_kurang_baik balza)
(terlalu_banyak_bekerja balza)
(mengoreksi_tugas balza)
(mengajar balza)
(bulan maret)
```

Urutan aturan yang dipicu bisa jadi sangat vital, terutama di mana aturan-aturan yang ada dapat mengakibatkan terhapusnya item dari memori kerja. Tinjau kasus berikut: andaikan terdapat tambahan aturan pada kumpulan aturan di atas, yaitu:

```
7. JIKA (bahagia X)
   MAKA TAMBAH (memberi_nilai_bagus X)
```

Jika aturan 7 ini terpicu sebelum `(bahagia balza)` dihapus dari memori, maka Sistem Pakar akan berkesimpulan bahwa saya akan memberi nilai bagus[1](1). Namun jika aturan 5 terpicu dahulu, maka aturan 7 tidak akan dijalankan (artinya saya tidak akan memberi nilai bagus).

[1] Perhatian. Contoh ini tidak merefleksikan perilaku saya yang sebenarnya dalam mengajar, karena bagi saya nilai tergantung dari bagus tidaknya pekerjaan dan tidak tergantung dari mood saya.

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:

a. *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.
b. *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.
c. *Specificity*. Picu aturan dengan fakta prakondisi yang lebih spesifik (khusus) sebelum aturan yang mengunakan prakondisi lebih umum. Contohnya: jika kita mempunyai aturan "JIKA (burung X) MAKA TAMBAH (dapat_terbang X)" dan "JIKA (burung X) DAN (pinguin X) MAKA TAMBAH (dapat_berenang X)" serta fakta bahwa tweety adalah seekor pinguin, maka lebih baik memicu aturan kedua dan menarik kesimpulan bahwa tweety dapat berenang.
d. *Operation priority*. Pilih aturan dengan prioritas yang lebih tinggi. Misalnya ada fakta (bertemu kambing), (ternak kambing), (bertemu macan), dan (binatang_buas macan), serta dua aturan: "JIKA (bertemu X) DAN (ternak X) MAKA TAMBAH (memberi_makan X)" dan "JIKA (bertemu X) DAN (binatang_buas X) MAKA TAMBAH (melarikan_diri)", maka kita akan memilih aturan kedua karena lebih tinggi prioritasnya.

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*[(2)].

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. Jadi aturan-aturan di atas diubah menjadi:

```
1. JIKA (mengajar X)
   DAN (mengoreksi_tugas X)
   MAKA (terlalu_banyak_bekerja X)
2. JIKA (bulan maret)
   MAKA (mengajar balza)
3. JIKA (bulan maret)
   MAKA (mengoreksi_tugas balza)
4. JIKA (terlalu_banyak_bekerja X)
   ATAU (kurang_tidur X)
   MAKA (mood_kurang_baik X)
5. JIKA (mood_kurang_baik X)
   MAKA TIDAK BENAR (bahagia X)
```

dengan fakta awal:

```
(bulan maret)
(meneliti balza)
```

Misalkan kita hendak membuktikan apakah mood saya sedang kurang baik. Mula-mula kita periksa apakah goal cocok dengan fakta awal. Ternyata tidak ada fakta awal yang

[2] Metode ini sering disebut juga sistem terarah-tujuan (*goal-driven*).

menyatakan demikian, sehingga langkah kedua yaitu mencari aturan mana yang mempunyai konklusi `(mood_kurang_baik balza)`. Dalam hal ini aturan yang cocok adalah aturan 4 dengan variabel X diisi dengan (*bound to*) `balza`. Dengan demikian kita harus membuktikan bahwa prakondisi aturan ini, `(terlalu_banyak_bekerja balza)` atau `(kurang_tidur balza)`, salah satunya adalah benar (karena memakai ATAU). Lalu diperiksa aturan mana yang dapat membuktikan bahwa adalah `(terlalu_banyak_bekerja balza)` benar, ternyata aturan 1, sehingga prakondisinya, `(mengajar X)` dan `(mengoreksi_tugas X)`, dua-duanya adalah benar (karena memakai DAN). Ternyata menurut aturan 2 dan 3, keduanya bernilai bernilai benar jika `(bulan maret)` adalah benar. Karena ini sesuai dengan fakta awal, maka keduanya bernilai benar. Karena semua goal sudah terpenuhi maka goal utama (hipotesis) bahwa mood saya sedang kurang baik adalah benar (terpenuhi).

Untuk mencatat goal-goal yang harus dipenuhi/dibuktikan, dapat digunakan *stack* (tumpukan). Setiap kali ada aturan yang konklusinya cocok dengan goal yang sedang dibuktikan, maka fakta-fakta prakondisi dari aturan tersebut ditaruh (*push*) ke dalam stack sebagai goal baru. Dan setiap kali goal pada tumpukan teratas terpenuhi atau dapat dibuktikan, maka goal tersebut diambil (*pop*) dari tumpukan. Demikian seterusnya sampai tidak ada goal lagi di dalam *stack*, atau dengan kata lain goal utama (yang terdapat pada tumpukan terbawah) sudah terpenuhi.

4. Pemilihan Sistem Inferensi

Secara umum kita dapat memakai panduan berikut untuk menentukan apakah kita hendak memilih perantaian maju atau balik untuk Sistem Pakar yang kita bangun. Panduan tersebut tercantum dalam Tabel II-1 berikut ini.

Tabel II-1. Panduan untuk memilih sistem inferensi

Perantaian Maju	Perantaian Balik
• Ada banyak hal yang hendak dibuktikan • Hanya sedikit fakta awal yang dipunyai • Ada banyak aturan berbeda yang dapat memberikan kesimpulan yang sama	• Hanya akan membuktikan fakta (hipotesis) tunggal • Terdapat banyak fakta awal • Jika terdapat banyak aturan yang memenuhi syarat untuk dipicu (*fire*) pada suatu siklus

5. Ketidakpastian dalam Aturan

Sejauh ini kita menggunakan nilai kebenaran tegas dalam fakta dan aturan yang dipakai, misalnya: jika terlalu banyak bekerja maka pasti mood kurang baik. Pada kenyataanya, seringkali kita tidak bisa membuat aturan yang absolut untuk mengambil kesimpulan secara pasti, misalnya: jika terlalu banyak bekerja maka kemungkinan besar mood kurang baik. Untuk itu, seringkali aturan yang dipakai memiliki nilai kepastian (*certainty value*). Contohnya: jika terlalu banyak bekerja maka pasti mood kurang baik (kepastian 0,75).

G. Contoh Aplikasi Sistem Pakar

1. Aplikasi Sederhana: Sistem Pakar Bengkel Mobil

Ini adalah contoh Sistem Pakar sederhana, yang bertujuan untuk mencari apa yang salah sehingga mesin mobil pelanggan yang tidak mau hidup, dengan memberikan gejala-gejala yang teramati. Anggap Sistem Pakar kita memiliki aturan-aturan berikut:

```
1. JIKA mesin_mendapatkan_bensin
   DAN starter_dapat_dihidupkan
   MAKA ada_masalah_dengan_pengapian
2. JIKA TIDAK BENAR starter_dapat_dihidupkan
   DAN TIDAK BENAR lampu_menyala
   MAKA ada_masalah_dengan_aki
3. JIKA TIDAK BENAR starter_dapat_dihidupkan
   DAN lampu_menyala
   MAKA ada_masalah_dengan_starter
4. JIKA ada_bensin_dalam_tangki_bahan_bakar
   MAKA mesin_mendapatkan_bensin
```

Terdapat 3 masalah yang mungkin, yaitu: `ada_masalah_dengan_pengapian`, `ada_masalah_dengan_aki` dan `ada_masalah_dengan_starter`. Dengan sistem terarah-tujuan (*goal-driven*), kita hendak membuktikan keberadaan setiap masalah tadi.

Pertama, Sistem Pakar berusaha untuk membuktikan kebenaran `ada_masalah_dengan_pengapian`. Di sini, aturan 1 dapat digunakan, sehingga Sistem Pakar akan menset goal baru untuk membuktikan apakah `mesin_mendapatkan_bensin` serta `starter_dapat_dihidupkan`. Untuk membuktikannya, aturan 4 dapat digunakan, dengan goal baru untuk membuktikan `mesin_mendapatkan_bensin`. Karena tidak ada aturan lain yang dapat digunakan menyimpulkannya, sedangkan sistem belum memperoleh solusinya, maka Sistem Pakar kemudian bertanya kepada pelanggan: "*Apakah ada bensin*

dalam tangki bahan bakar?". Sekarang, katakanlah jawaban klien adalah "*Ya*", jawaban ini kemudian dicatat, sehingga klien tidak akan ditanyai lagi dengan pertanyaan yang sama.

Nah, karena sistem sekarang sudah dapat membuktikan bahwa mesin mendapatkan bensin, maka sistem sekarang berusaha mengetahui apakah `starter_dapat_dihidupkan`. Karena sistem belum tahu mengenai hal ini, sementara tidak ada aturan lagi yang dapat menyimpulkannya, maka Sistem Pakar bertanya lagi ke klien: "*Apakah starter dapat dihidupkan?*". Misalkan jawabannya adalah "*Tidak*", maka tidak ada lagi aturan yang dapat membuktikan `ada_masalah_dengan_pengapian`, sehingga Sistem Pakar berkesimpulan bahwa hal ini bukanlah solusi dari problem yang ada, dan kemudian melihat hipotesis berikutnya: `ada_masalah_dengan_aki`. Sudah diketahui (dibuktikan) bahwa mesin tidak dapat distarter, sehingga yang harus dibuktikan adalah bahwa lampu tidak menyala. Sistem Pakar kemudian bertanya: "*Apakah lampu menyala?*". Misalkan jawabannya adalah "*Tidak*", maka sudah terbukti bahwa ada masalah dengan aki.

Sistem ini mungkin berhenti sampai di sini, tetapi biasanya ada kemungkinan terdapat lebih dari satu solusi (misalnya terdapat lebih dari satu kerusakan), atau ada kemungkinan terdapat solusi lain yng lebih tepat, sehingga biasanya semua hipotesis diperiksa kebenarannya. Sistem Pakar ini kemudian mencoba membuktikan bahwa `ada_masalah_dengan_starter`, namun dari fakta yang sudah diperoleh, yaitu lampu tidak menyala, maka pembuktiannya menjadi gagal. Dengan demikian solusi yang diberikan oleh Sistem Pakar adalah ada masalah dengan aki.

Secara lengkap, interaksi antara Sistem Pakar dengan klien mungkin seperti ini:

System Pakar: *Apakah ada bensin dalam tangki bahan bakar?*
Klien: Ya
System Pakar: *Apakah starter dapat dihidupkan?*
Klien: Tidak
System Pakar: *Apakah lampu menyala?*
Klien: Tidak
System Pakar: *Saya berkesimpulan bahwa ada masalah dengan aki*

Fasilitas Penjelasan

Kebanyakan Sistem Pakar mempunyai fasilitas penjelasan yang mengijinkan klien untuk bertanya mengapa dia diberi suatu pertanyaan atau bagaimana dapat mencapai suatu kesimpulan. Pertanyaan dari klien tersebut dijawab dengan mengacu pada goal, aturan yang dipakai, dan data-data yang ada. Contoh pemakaian fasilitas penjelas dalam Sistem Pakar di atas dapat dilihat pada percakapan berikut:

System Pakar: *Apakah ada bensin dalam tangki bahan bakar?*
Klien: Ya
System Pakar: *Apakah starter dapat dihidupkan?*
Klien: Tidak
System Pakar: *Apakah lampu menyala?*
Klien: Mengapa?
System Pakar: *Saya mencoba membuktikan bahwa ada masalah dengan aki. Sudah ditunjukkan adalah tidak benar starter dapat dihidupkan. Sehingga jika tidak benar bahwa lampu menyala, maka ada masalah dengan aki.*
Apakah lampu menyala?
Klien: Tidak
System Pakar: *Saya berkesimpulan bahwa ada masalah dengan aki*
Klien: Bagaimana bisa?
System Pakar: *Hal ini sesuai dengan aturan no 2:*
JIKA TIDAK BENAR starter_dapat_dihidupkan
DAN TIDAK BENAR lampu_menyala
MAKA ada_masalah_dengan_aki
bahwa tidak benar starter_dapat_dihidupkan sudah dijawab,
bahwa tidak benar lampu_menyala sudah dijawab,
sehingga dapat disimpulkan bahwa ada_masalah_dengan_aki.

Fasilitas penjelasan berhubungan dengan pencatatan aturan apa saja yang digunakan untuk mengambil kesimpulan, dan menggunakan catatan tersebut untuk menyusun penjelasan. Pembangkitan penjelasan sederhana seperti contoh di atas bukanlah persoalan yang sulit, namun pemahaman klien terhadap penjelasan tersebut tidaklah sama, sehingga seringkali klien tetap tidak mengerti maksudnya. Hal ini karena biasanya penjelasan hanya meliputi pengetahuan yang dangkal atau hanya permukaan saja sebagaimana tercantum dalam autran yang dipunyai, bukannya pengetahuan yang mendalam yang mendasari aturan tersebut. Dalam contoh di atas, Sistem Pakar hanya mengatakan bahwa hal ini sesuai dengan aturan no 2, tetapi tidak menjelaskan secara lebih detil mengenai aturan 2 tersebut. Mungkin klien perlu diberi pengertian bahwa lampu dan starter dua-duanya menggunakan aki untuk menyalakannya. Hal penting lain adalah ketika klien gagal mengerti atau menerima penjelasan yang diberikan, biasanya Sistem Pakar tidak dapat memberi penjelasan dengan cara lain (tidak begitu halnya dengan seorang pakar yang mampu menggunakan kalimat yang lain untuk mencoba menjelaskannya).

2. 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. Berikut ini adalah 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.
...

3. Parry

Parry adalah Sistem Pakar yang juga paling awal dikembangkan di Stanford University oleh seorang psikiater, Kenneth Colby, yang mensimulasikan seorang paranoid. Berikut ini contoh respon Parry terhadap pertanyaan seorang psikiater yang menginterviunya.

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.*
...