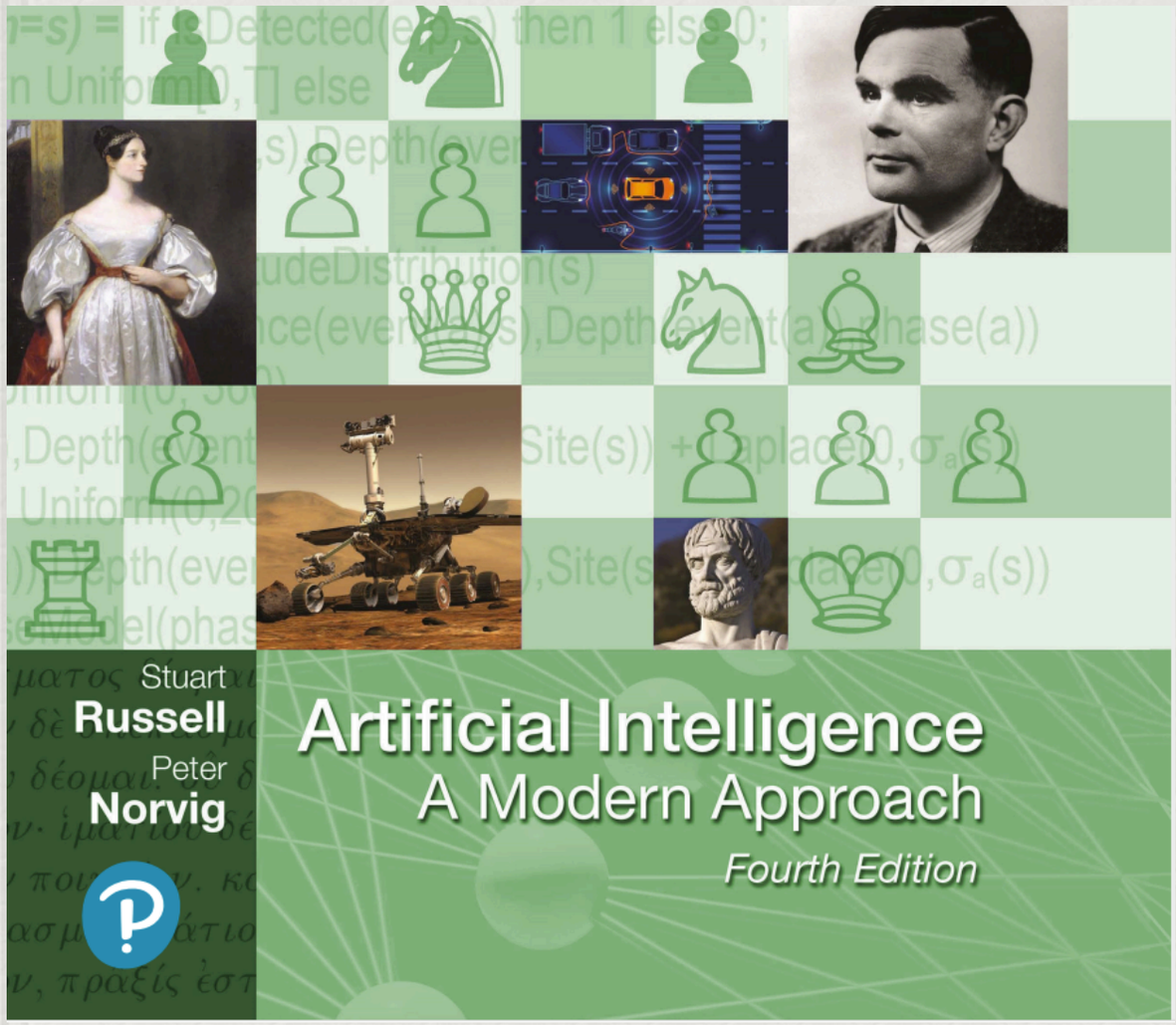
Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial
Aula 9 - Agentes Racionais

Prof. José Reinaldo Silva

reinaldo@usp.br



CHAPTER 7

LOGICAL AGENTS

In which we design agents that can form representations of a complex world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do.

Knowledge-based agents
Reasoning
Representation

Humans, it seems, know things; and what they know helps them do things. In AI, **knowledge-based agents** use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take.

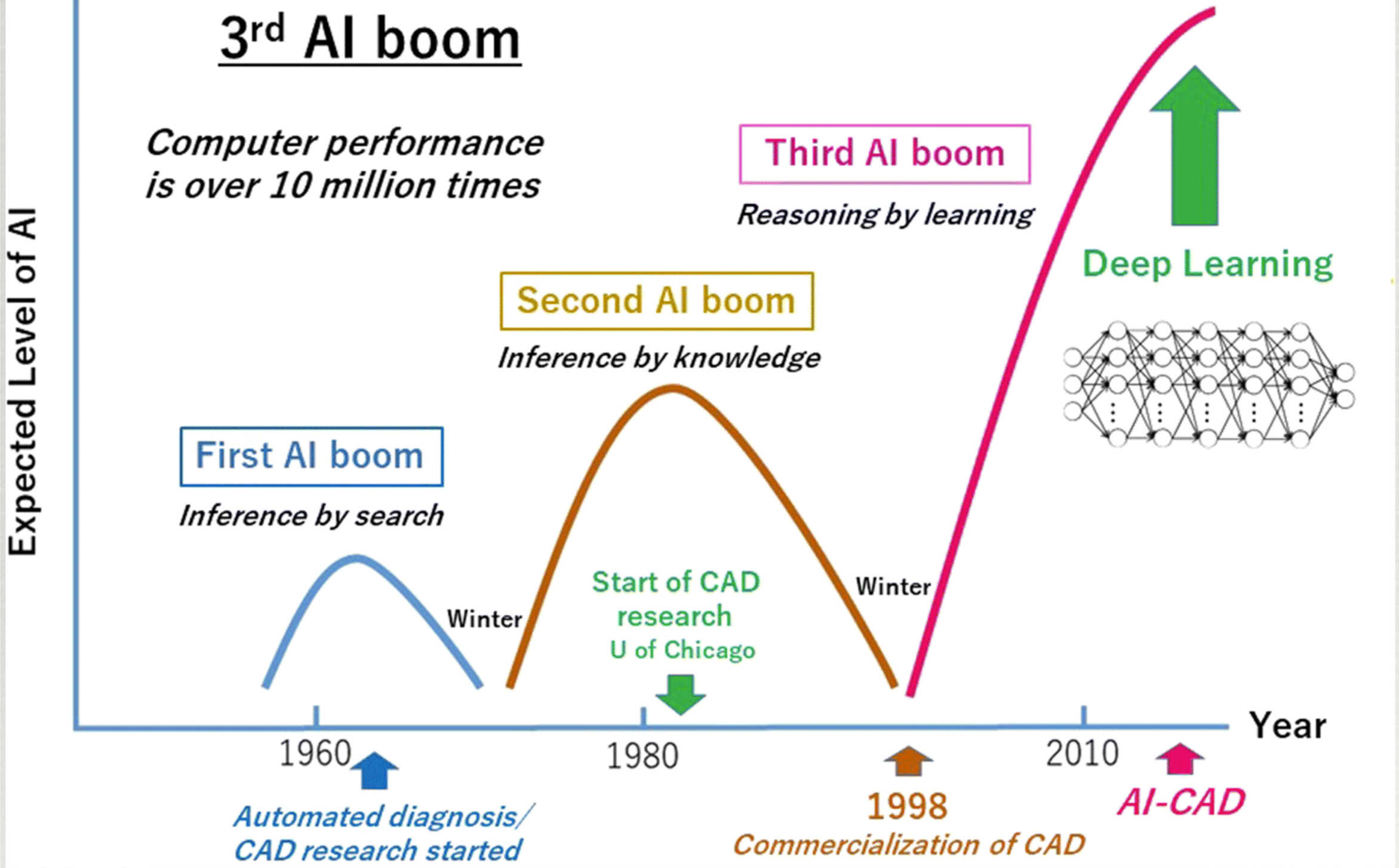
The problem-solving agents of Chapters 3 and 4 know things, but only in a very limited, inflexible sense. They know what actions are available and what the result of performing a specific action from a specific state will be, but they don't know general facts. A route-finding agent doesn't know that it is impossible for a road to be a negative number of kilometers long. An 8-puzzle agent doesn't know that two tiles cannot occupy the same space. The knowledge they have is very useful for finding a path from the start to a goal, but not for anything else.

The atomic representations used by problem-solving agents are also very limiting. In a partially observable environment, for example, a problem-solving agent's only choice for representing what it knows about the current state is to list all possible concrete states. I could give a human the goal of driving to a U.S. town with population less than 10,000, but to say that to a problem-solving agent, I could formally describe the goal only as an explicit set of the 16,000 or so towns that satisfy the description.

Chapter 5 introduced our first factored representation, whereby states are represented as assignments of values to variables; this is a step in the right direction, enabling some parts of the agent to work in a domain-independent way and allowing for more efficient algorithms. In this chapter, we take this step to its logical conclusion, so to speak—we develop **logic** as a general class of representations to support knowledge-based agents. These agents can combine and recombine information to suit myriad purposes. This can be far removed from the needs of the moment—as when a mathematician proves a theorem or an astronomer calculates the Earth's life expectancy. Knowledge-based agents can accept new tasks in the form of explicitly described goals; they can achieve competence quickly by being told or learning

3rd AI boom

Computer performance is over 10 million times

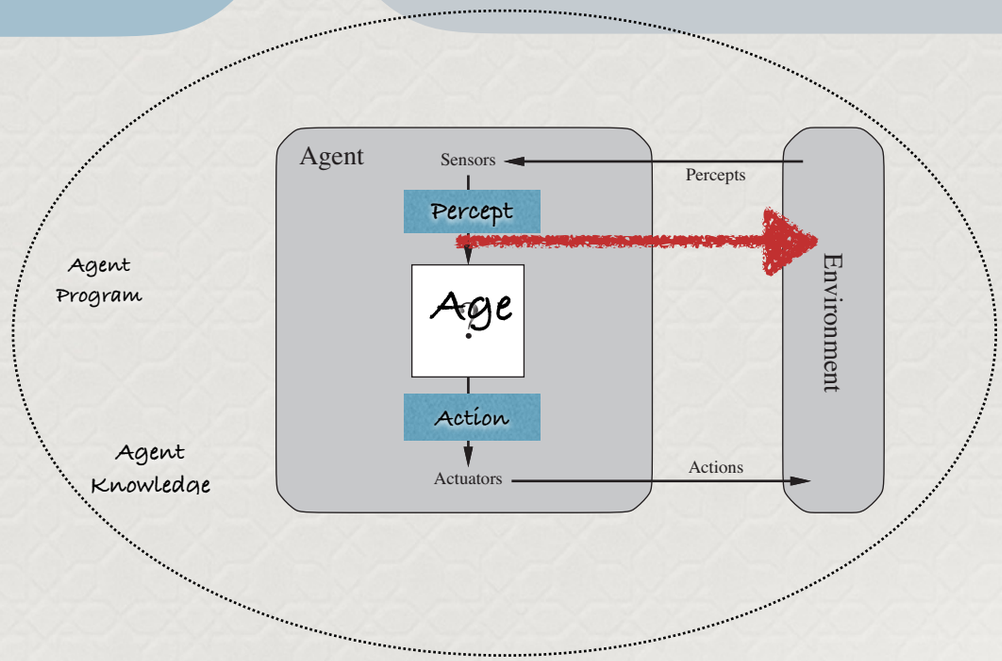


Aplicações Tradicionais da IA Clássica:

Jogos simples;
Sistemas de diagnóstico;
Sistemas educacionais;
Sistemas de segurança;
Sistemas logísticos...

Aplicações Recentes:

"Serious games";
Sistemas de diagnóstico com interação NLP;
Sistemas assistivos (healthcare);
RPA...



Vários agentes inteligentes podem ser usados para compor um sistema RPA, mas certamente os mais interessantes seriam formados por agentes goal-based, utility-based ou learning agents.

Agentes baseados em objetivos (Goal-based agents)

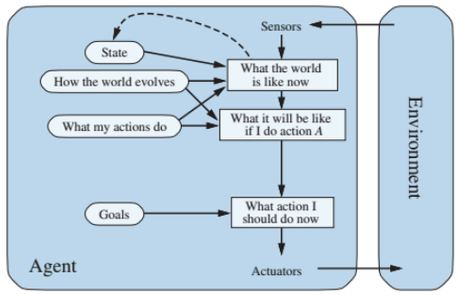


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Agentes baseados em serviço (Utility-based agents)

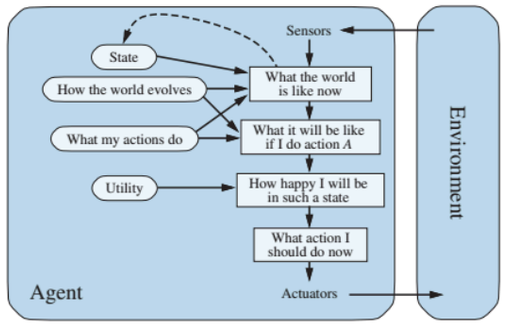


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Agentes que aprendem (Learning agents)

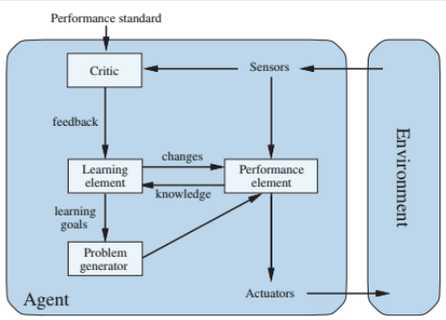
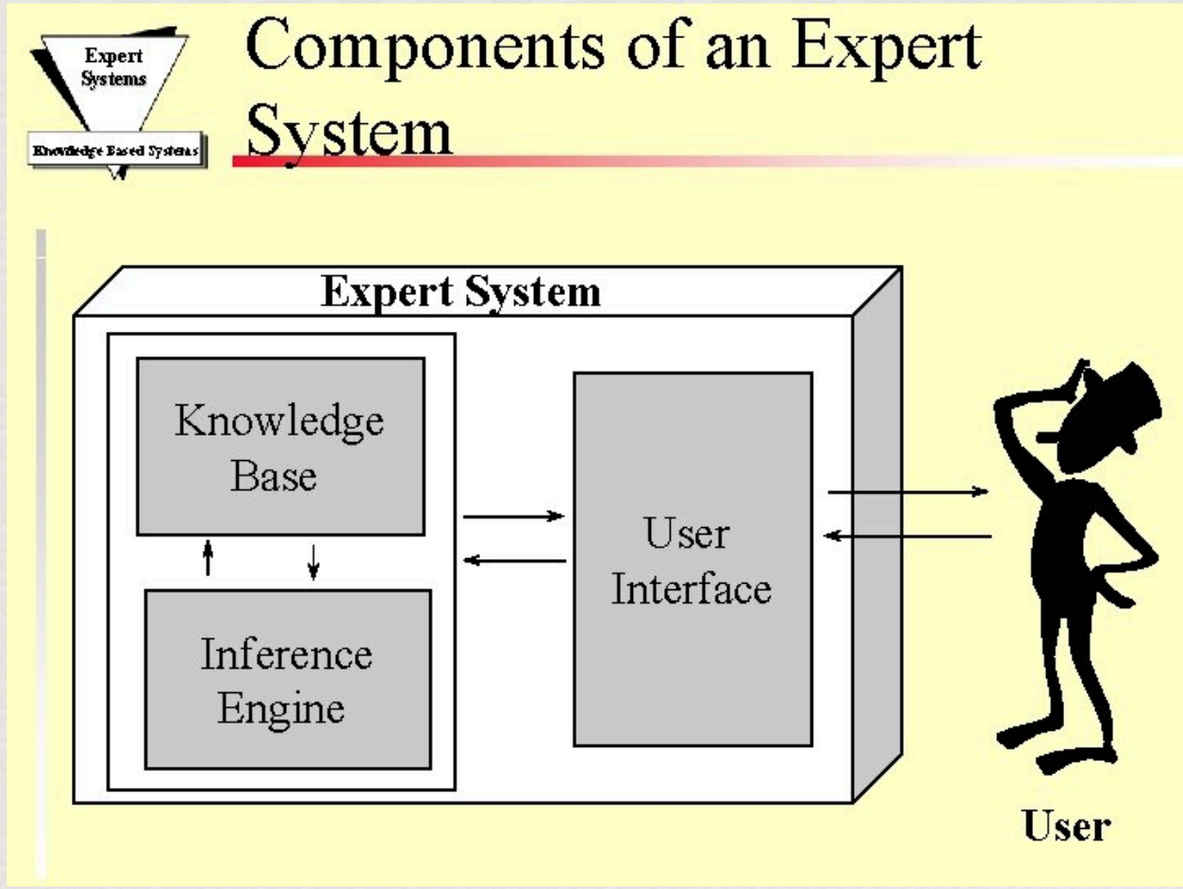
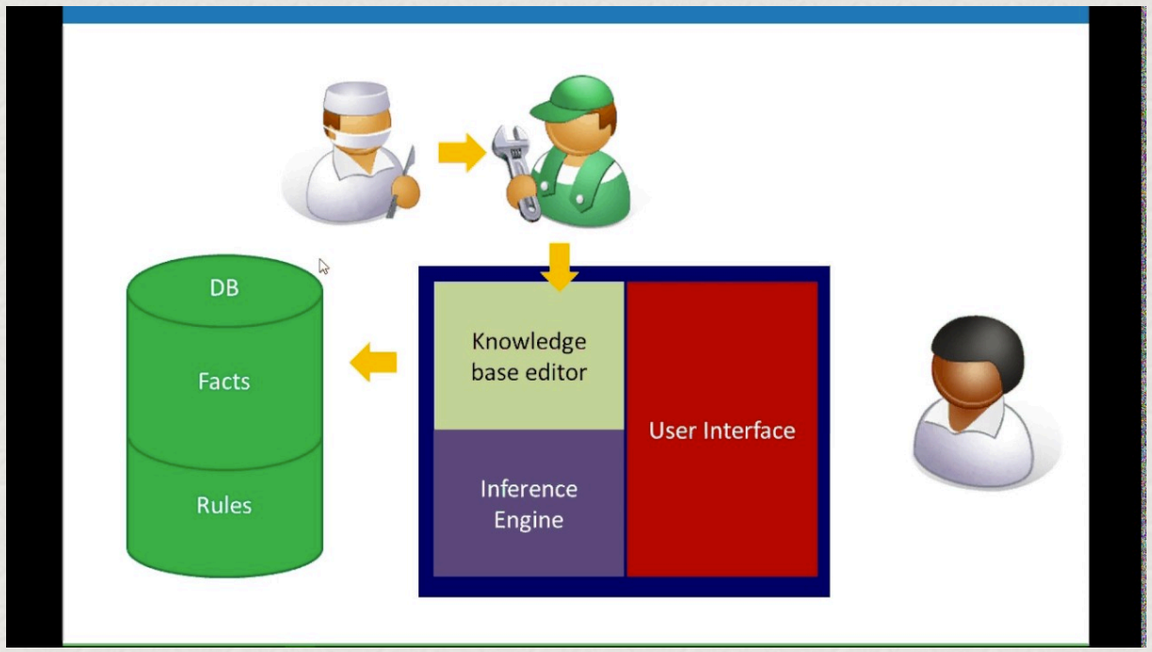
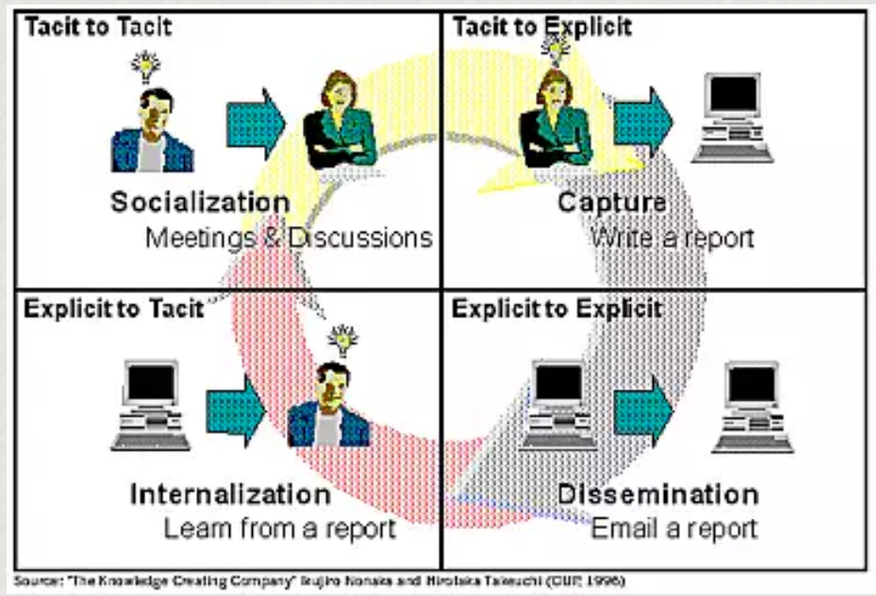


Figure 2.15 A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

Um KB-system é composto de fatos e regras associados a algum "environment" (domínio). Se esta base envolve um mix de conhecimento tácito e explícito pode também ser associada a um "sistema especialista".







Dado o número e a variedade de aplicações vale a pena entender mais profundamente como funciona um sistema baseado em conhecimento (KB system), o que significa entender o funcionamento da base de conhecimento e da máquina de inferência.

CHAPTER 7

LOGICAL AGENTS

In which we design agents that can form representations of a complex world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do.

Knowledge-based agents
Reasoning
Representation

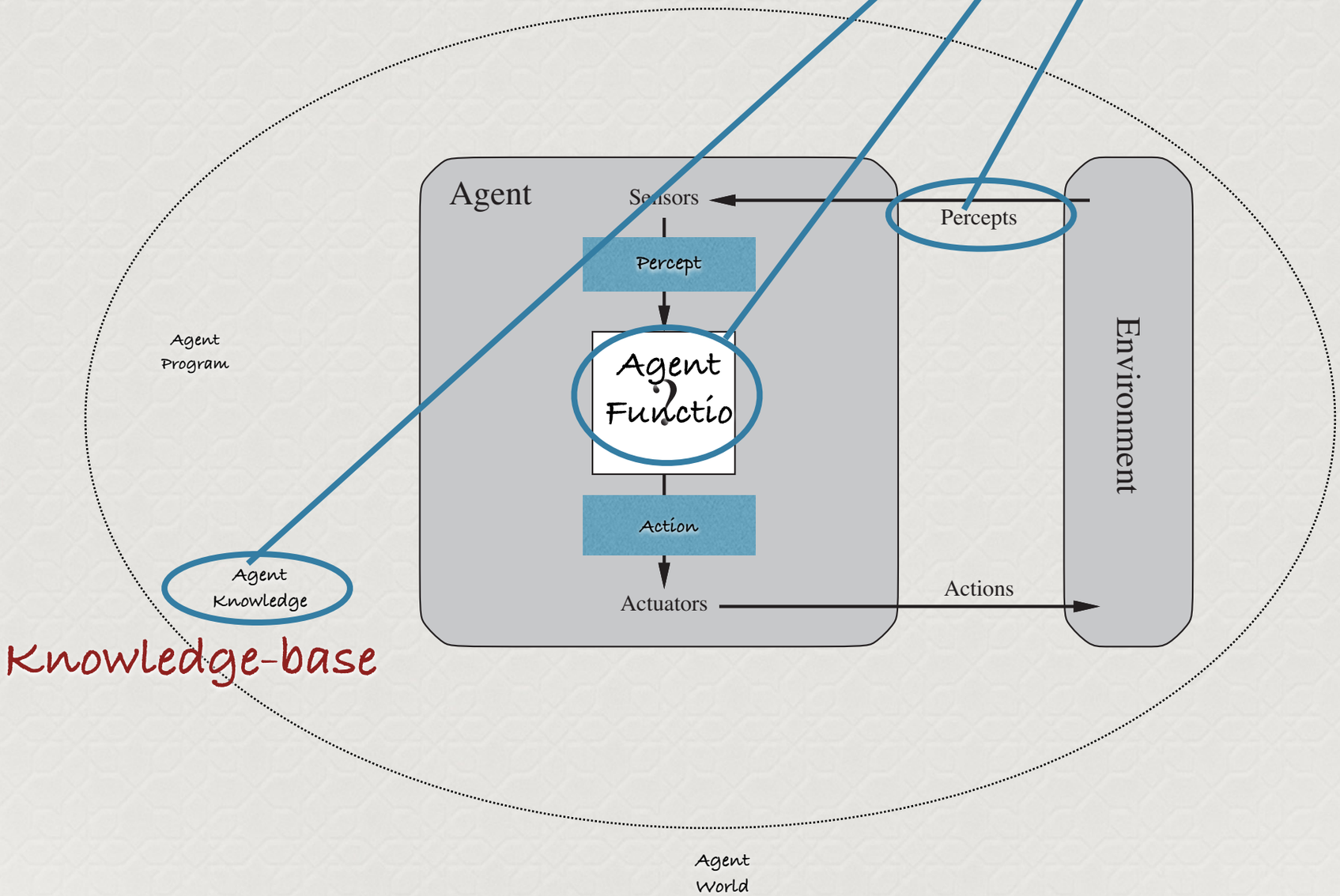
Humans, it seems, know things; and what they know helps them do things. In AI, **knowledge-based agents** use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take.

The problem-solving agents of Chapters 3 and 4 know things, but only in a very limited, inflexible sense. They know what actions are available and what the result of performing a specific action from a specific state will be, but they don't know general facts. A route-finding agent doesn't know that it is impossible for a road to be a negative number of kilometers long. An 8-puzzle agent doesn't know that two tiles cannot occupy the same space. The knowledge they have is very useful for finding a path from the start to a goal, but not for anything else.

The atomic representations used by problem-solving agents are also very limiting. In a partially observable environment, for example, a problem-solving agent's only choice for representing what it knows about the current state is to list all possible concrete states. I could give a human the goal of driving to a U.S. town with population less than 10,000, but to say that to a problem-solving agent, I could formally describe the goal only as an explicit set of the 16,000 or so towns that satisfy the description.

Chapter 5 introduced our first factored representation, whereby states are represented as assignments of values to variables; this is a step in the right direction, enabling some parts of the agent to work in a domain-independent way and allowing for more efficient algorithms. In this chapter, we take this step to its logical conclusion, so to speak—we develop **logic** as a general class of representations to support knowledge-based agents. These agents can combine and recombine information to suit myriad purposes. This can be far removed from the needs of the moment—as when a mathematician proves a theorem or an astronomer calculates the Earth's life expectancy. Knowledge-based agents can accept new tasks in the form of explicitly described goals; they can achieve competence quickly by being told or learning

Representação de conhecimento

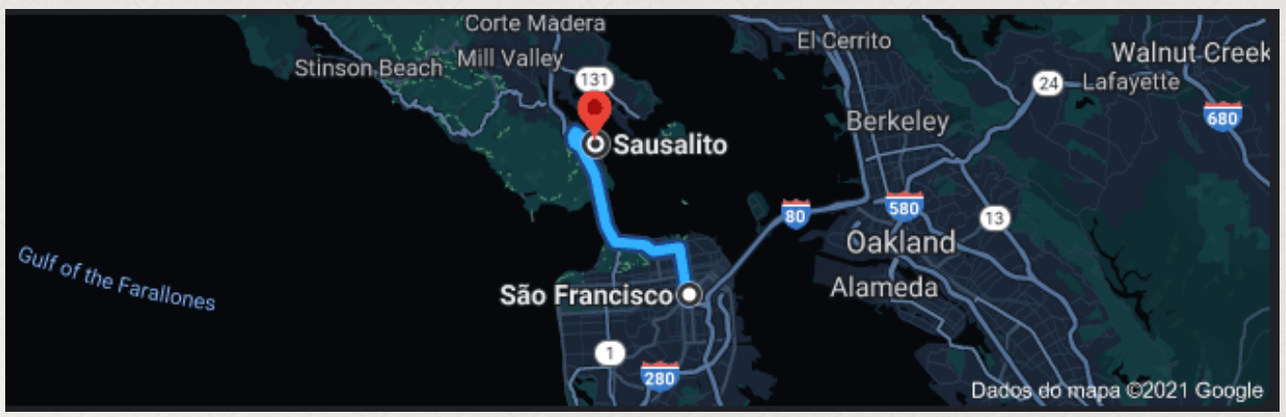


Portanto, "resolver problemas" significava:

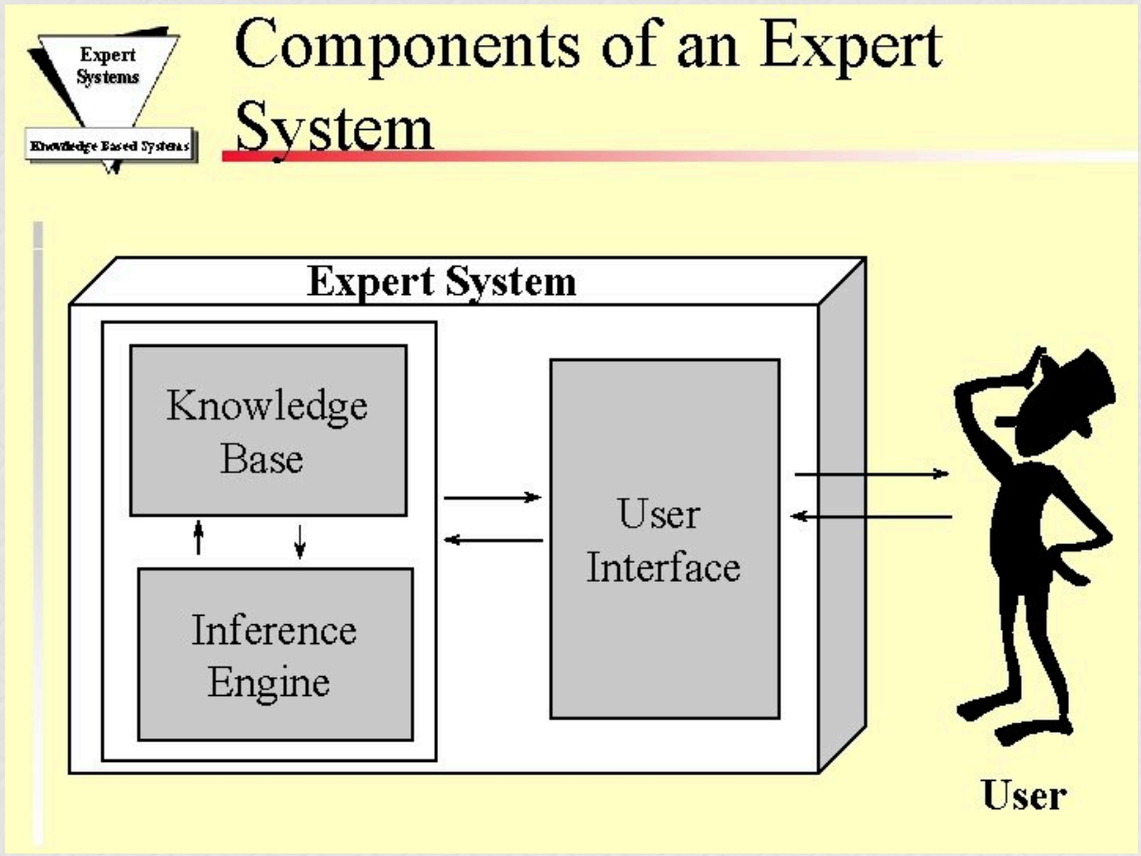
- Adiquir os "percepts" na forma de fatos e constantes;
- Gerar o espaço de estados (previamente ou durante a busca);
- Escolher a ação (ou sequência de ações) que levem a um estado objetivo, ou o tipo de resposta (por acionamento).

KB-agents podem transformar os percepts em "conhecimento" ou objetivos e com eles:

- Usar sua base de conhecimento para verificar se existe uma ou mais ações que satisfaçam este objetivo;
- Senão, inferir novos conhecimentos que ajudem a "resolver o problema";
- Inferir qual ou quais ações devem ser executadas.



O nosso primeiro desafio é saber como se constrói uma base de conhecimento, incluindo fatos, constantes, regras.

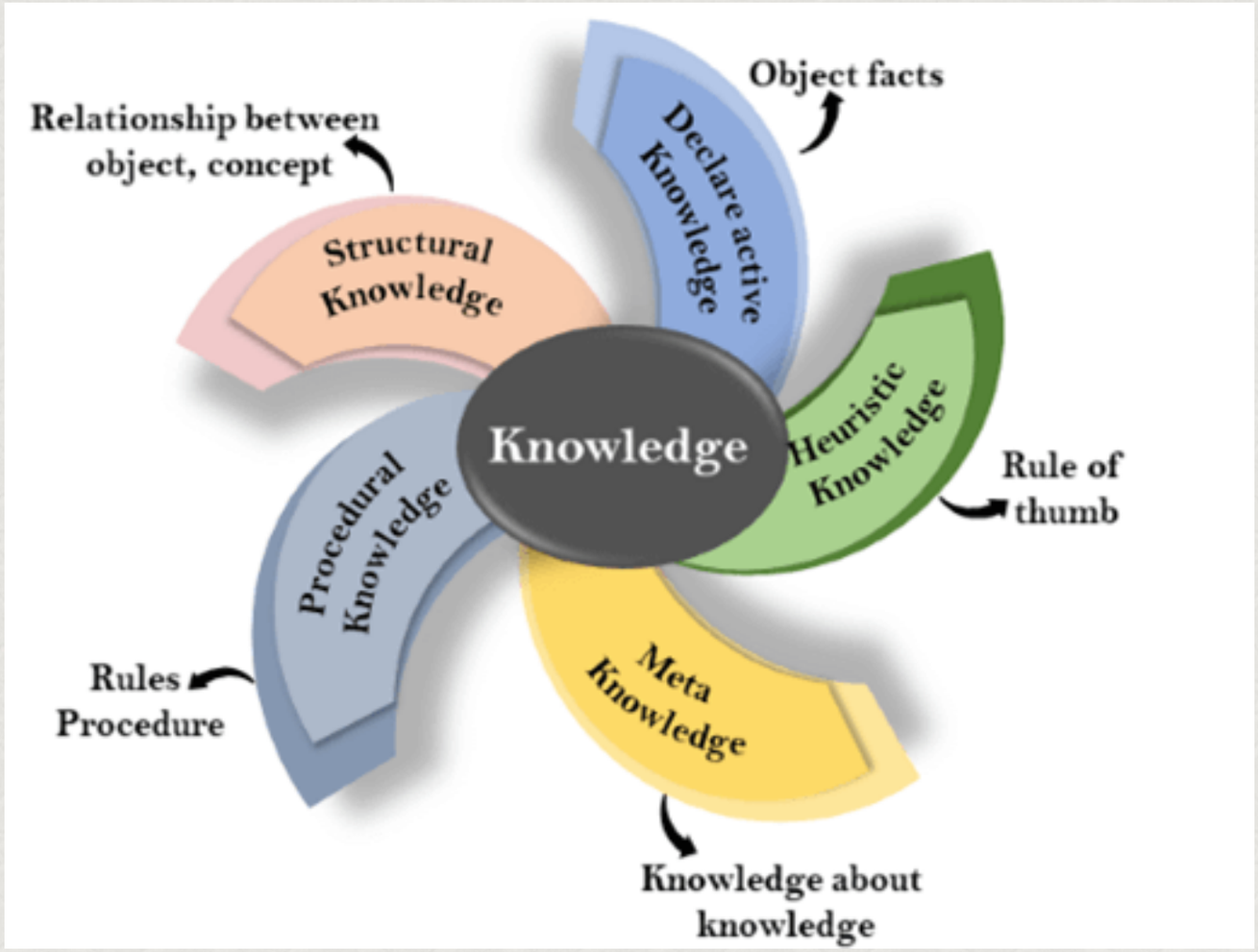


Knowledge Representation and Reasoning (KRR)

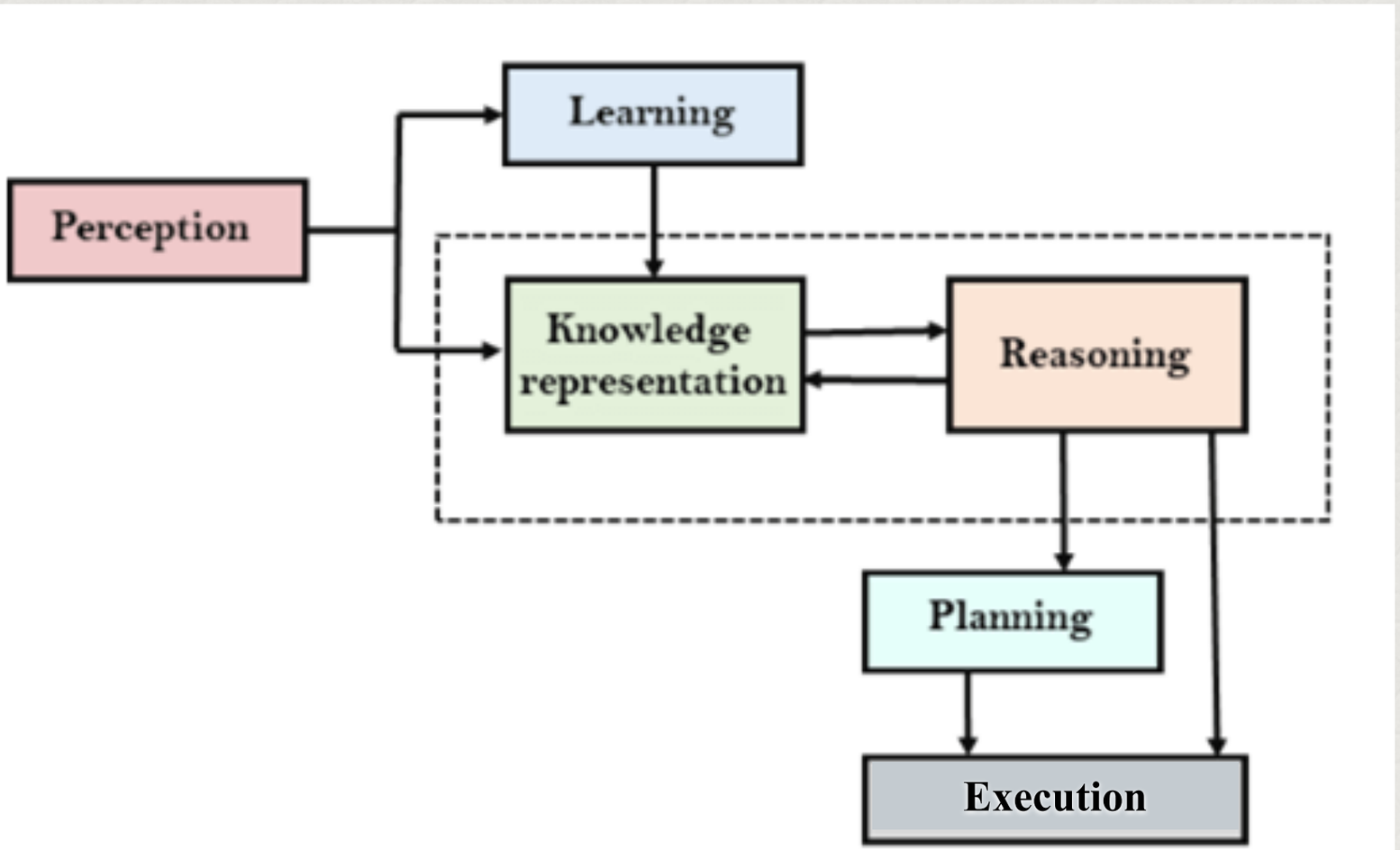
What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

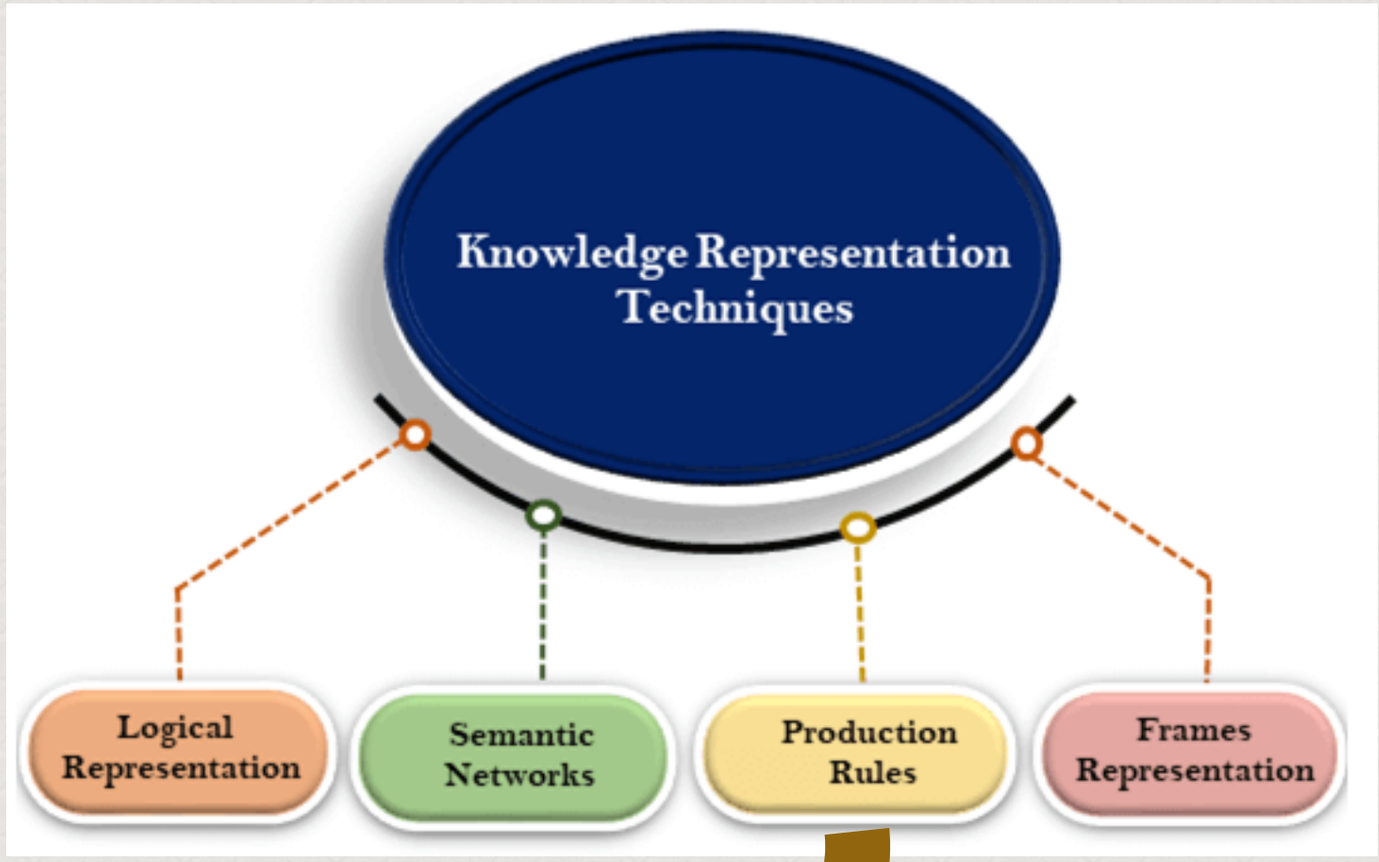
- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).


Object facts
Relationship between object, concept
Declare active Knowledge
Structural Knowledge
Knowledge
Heuristic Knowledge
Rule of thumb
Procedural Knowledge
Rules Procedure
Meta Knowledge
Knowledge about knowledge

Sensing



Acting



- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

Intelligent Fire Alarm: forward chaining with multiple actions:

1. If smoky is detected go to alarm_mode_1
2. If in alarm_mode_1 then {check(temp1), wait(10), check(temp2) and if temp2 > temp1 go to alarm_mode_2}
3. If in alarm_mode_2 then if visual_detect(flames) go to alarm_mode_3
4. If in alarm_mode_3 then {turn_on(sprinklers), close(elevators), close(cutting_fire_doors), sound_alerts, call(fire_force)}.

Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Podemos também representar "regras de produção" com cláusulas Horn.

Seja uma cláusula definida,

$$\neg p_1 \vee \neg p_2 \vee \ldots \neg p_n \vee q$$

composta por um conjunto de sentenças negadas p_i e somente uma sentença positiva q. Esta cláusula é equivalente a,

$$\overline{(p_1 \wedge p_2 \wedge \ldots p_n)} \vee q$$

$$p_1 \wedge p_2 \wedge \ldots p_n \rightarrow q$$

$$p_1 \wedge p_2 \wedge \ldots p_n \rightarrow q$$

$$q \leftarrow p_1 \wedge p_2 \wedge \ldots p_n$$

$$q :- p_1, p_2, \ldots p_n$$

Podemos partir de um conjunto de sentenças básicas, isto é, que não são derivadas de outras sentenças, chamadas de "atômicas". Seja uma interpretação I que valida este conjunto de sentenças.

$$\{s_i\} \xrightarrow{I} V$$

Sentenças não-atômicas podem ser derivadas de um conjunto básico (ground) por algum mecanismo de dedução. Por exemplo, se uma sentença β é derivada de um conjunto de sentenças $\{s_i\}$, dizemos que

$$\{s_i\} \vDash \beta$$

A pergunta é: como se pode deduzir novas sentenças à partir de um conjunto inicial?

Detecção de incêndio (real)

(S1) Existe fumaça;

(S2) Existem chamas;

(S3) A temperatura está em elevação;

(S4) se (Existe fumaça) e ((Existem chamas) ou (A temperatura está em elevação) então (Existe um incêndio).

$$(S_1 \wedge (S_2 \vee S_3)) \rightarrow S_4$$

Um processo de inferência pode ser programado em uma "máquina" ou algoritmo.

Um processo de inferência pode ser programado usando o "Modus Ponens":

$$(A \wedge (A \rightarrow B)) \rightarrow B$$

Se alguma interpretação valida A e $(A \rightarrow B)$, então também valida a sentença B.

$$(A \wedge (A \rightarrow B)) \rightarrow B$$

Se alguma interpretação valida A e $(A \rightarrow B)$, então também valida a sentença B.

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

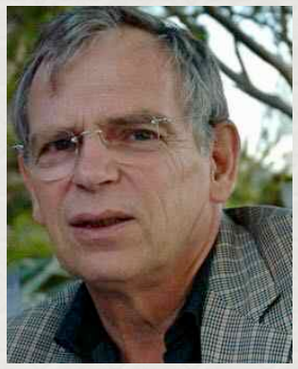
P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

P	Q	P ↔ Q
T	T	T
T	F	F
F	T	F
F	F	T

Aula 1

A linguagem Prolog (de Programação Lógica) foi criada para servir ao propósito de expressar de forma declarativa (e não procedural) ações cognitivas, dentro de um sub-conjunto da lógica de primeira ordem. Foi criada por Alain Colmerauer e Robert Kowalski e lançada em 1972.



Alain Colmerauer (1941-2017)

Robert Kowalski (1945-2017)

Detecção de incêndio (real)

$Tem\_fumaca(X)$.

$Tem\_chamas(X)$.

$$(Temp(X, T_1) < Temp(X, T_2)) \wedge (T_1 < T_2)$$

$$Incendio(X) \leftarrow Tem\_fumaca(X) \wedge (Tem\_chamas(X) \vee ((Temp(X, T_1) < Temp(X, T_2)) \wedge (T_1 < T_2)$$

$$Tem\_fumaca(X).$$

$$Tem\_chamas(X).$$

$$Incendio(X) \leftarrow Tem\_fumaca(X) \wedge (Tem\_chamas(X) \vee (Temp(X, T_1) < Temp(X, T_2)) \wedge (T_1 < T_2)$$

```
temp_elev(X) : − temp(X, T1, TP1), temp(X,T2, TP2),
                 (TP1 < TP2), (T1 < T2).

incendio(X) : −tem_fumaca(X), tem_chamas(X).
incendio(X) : −tem_fumaca(X), temp_elev(X).
```

Se colocarmos o programa abaixo no SWISH Prolog e perguntarmos incêndio(X) para saber se tem incêndio em algum cômodo o que acontece?

```
temp_elev(X) :- temp(X, T1, TP1), temp(X,T2, TP2),
                (TP1 < TP2), (T1 < T2).

incendio(X) :- tem_fumaca(X), tem_chamas(X).
incendio(X) :- tem_fumaca(X), temp_elev(X).
```

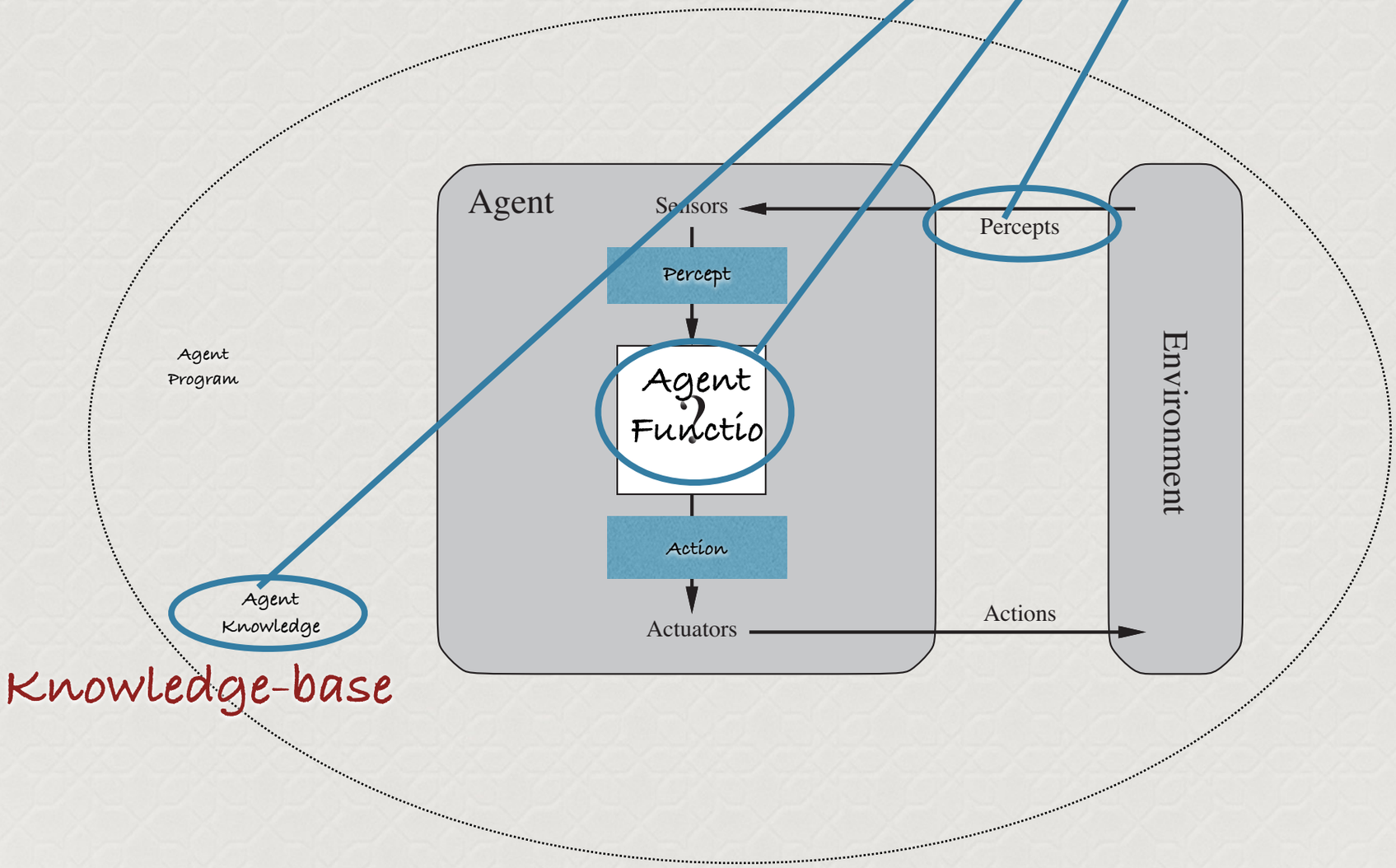
Os valores de TP1 e TP2 denotam a temperatura do local X em diferentes instantes de tempo, T1 e T2 e devem ser absorvidos por algum percept ou instanciada na base de conhecimento.

temp_elev(X) : − temp(X, T1, TP1), temp(X,T2, TP2),
(TP1 < TP2), (T1 < T2).

incendio(X) : −tem_fumaca(X), tem_chamas(X).
incendio(X) : −tem_fumaca(X), temp_elev(X).

Na nossa versão mais simples de KB-system teríamos que inserir manualmente na KB os locais onde tem fumaça, onde tem chamas, e a temperatura medida nos tempos T1 e T2.

Representação de conhecimento



Esta informação deve agora vir do percept.

É o percept que contém os dados de detectores de fumaça, de chamas, e medidores de temperatura (feita em tempos discretos). Estes dados devem ser transformados em fatos e regras e acrescentados ao nosso banco de conhecimento (KB) para que a inferência possa ser feita.

SWISH File Edit Examples Help

Prolog-introdução | Mundo de blocos - Rework 2.0 | Program

```
1  tem_fumaca(sala).
2  tem_fumaca(cozinha).
3
4  tem_chamas(quintal).
5
6  temp(cozinha, 2, 22).
7  temp(cozinha, 5, 25).
8
9  temp_elev(X):- temp(X, T1, TP1), temp(X,T2, TP2),(TP1 < TP2), (T1 < T2).
10
11 incendio(X):- tem_fumaca(X), tem_chamas(X).
12 incendio(X):- tem_fumaca(X), temp_elev(X).
13
14
```

SWISH File Edit Examples Help

1 users online

Search

Prolog-introdução | Mundo de blocos - Rework 2.0 | Program

```
1  tem_fumaca(sala).
2  tem_fumaca(cozinha).
3
4  tem_chamas(quintal).
5
6  temp(cozinha, 2, 22).
7  temp(cozinha, 5, 25).
8
9  temp_elev(X):- temp(X, T1, TP1), temp(X,T2, TP2),(TP1 < TP2), (T1 < T2).
10
11 incendio(X):- tem_fumaca(X), tem_chamas(X).
12 incendio(X):- tem_fumaca(X), temp_elev(X).
13
14
```

incendio(X).

X = cozinha

Next | 10 | 100 | 1,000 | Stop

?-

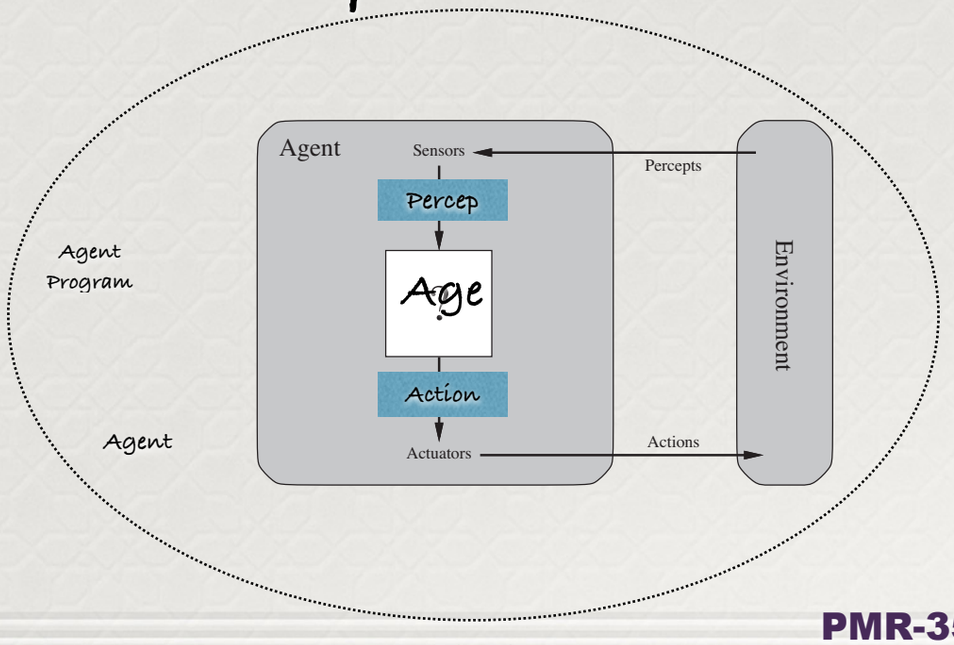
```
incendio(X).
```

Examples | History | Solutions | table results | Run!

Os fatos

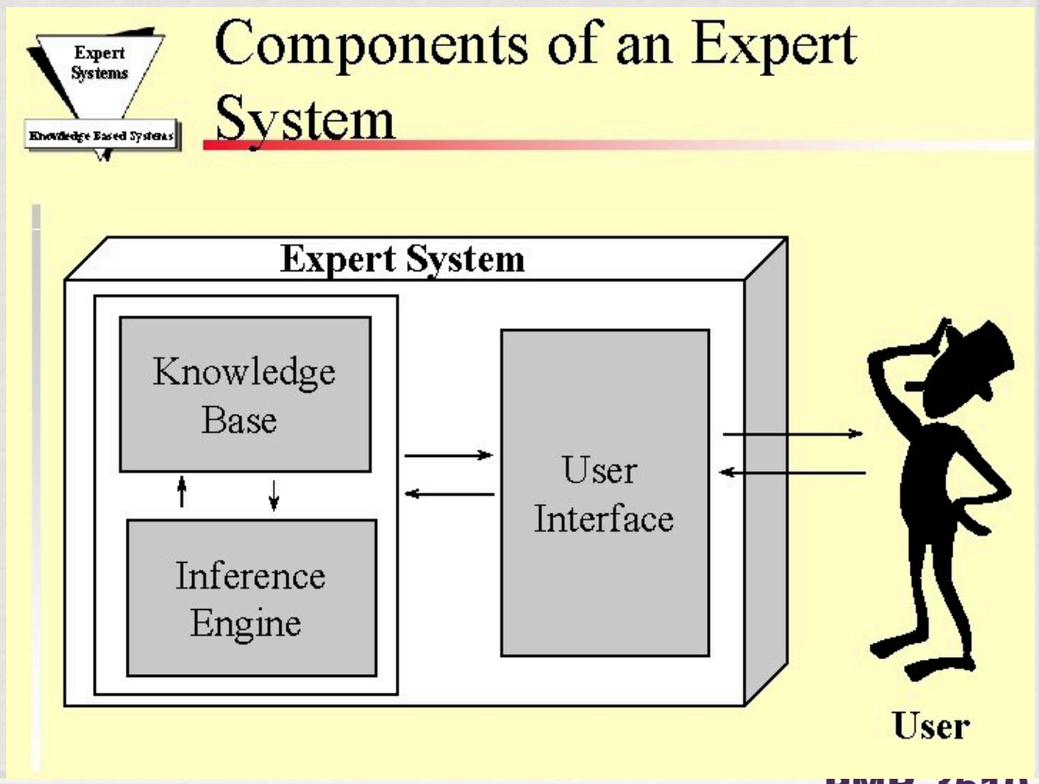
```
tem_fumaca(sala).
tem_fumaca(cozinha).

tem_chamas(quintal).

temp(cozinha, 2, 22).
temp(cozinha, 5, 25).
```

Devem ter sido gerados a partir de leitura de sensores, e transformados para o formato relacional.



Assim, temos um modelo aproximado e bastante simplificado de como funciona a programação lógica. Deve ter ficado mais claro o papel da base de conhecimento (KB) e como preparar estas bases para dar suporte a sistemas automatizados. Vamos usar isso para a resolução de problemas na abordagem cognitiva.

Na próxima aula vamos tratar de maneira um pouco mais formal o processo de construção de um sistema especialista.



Perguntas?