Raciocínio e Conhecimento

Prof. Hendrik Macedo

Direção do raciocínio

$P \Rightarrow Q$

1. rac dirigido por objetivo (Q -> P) vs. dirigido por dados (P -> Q)
2. encadeamento para frente (P -> Q) vs. para trás (Q -> P)

PROLOG

(Production systems)

Sistemas de Produção

Raciocínio: Encadeamento para frente sobre as regras

Produções

par condição-ação

padrão

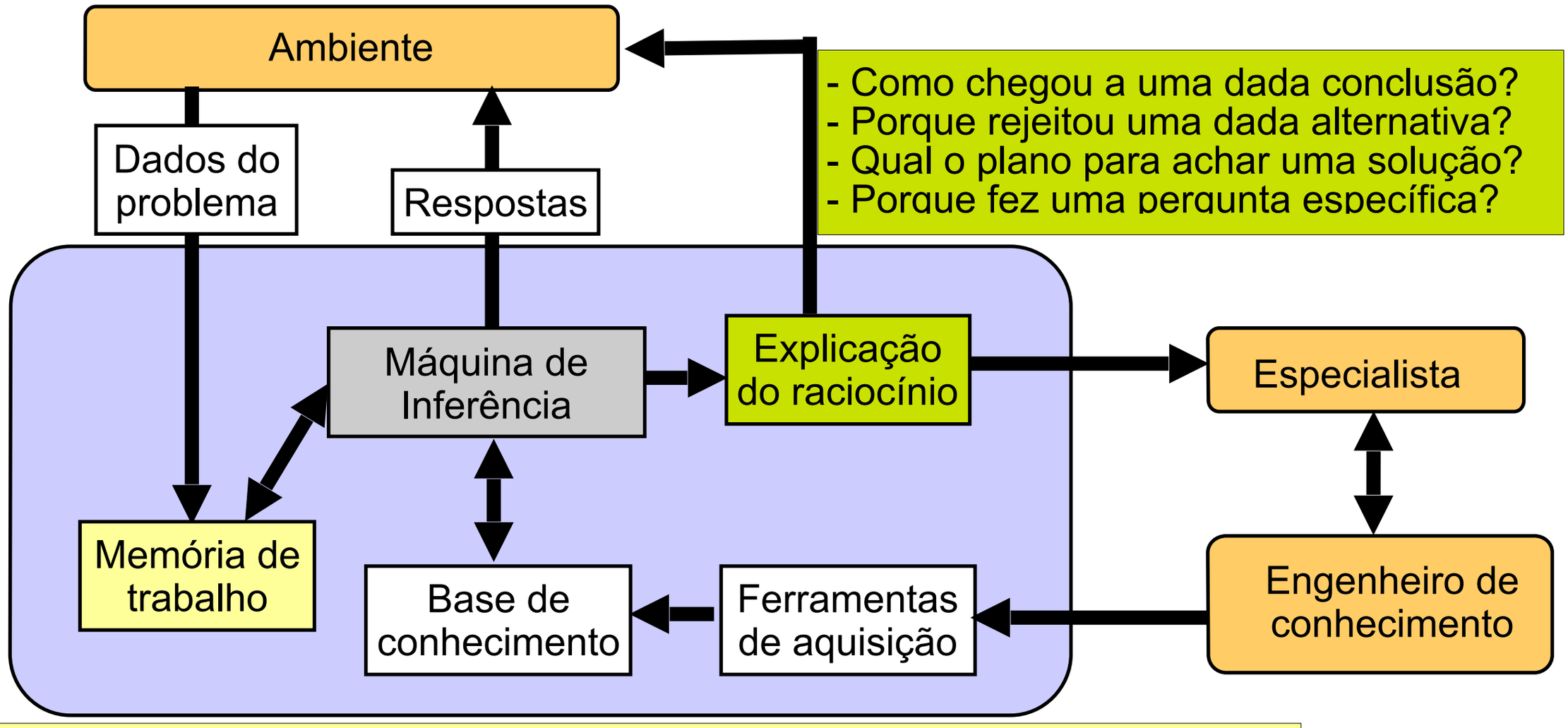
(Rule-based Systems)

Sistemas baseados em regras

Base para os Sistemas Especialistas

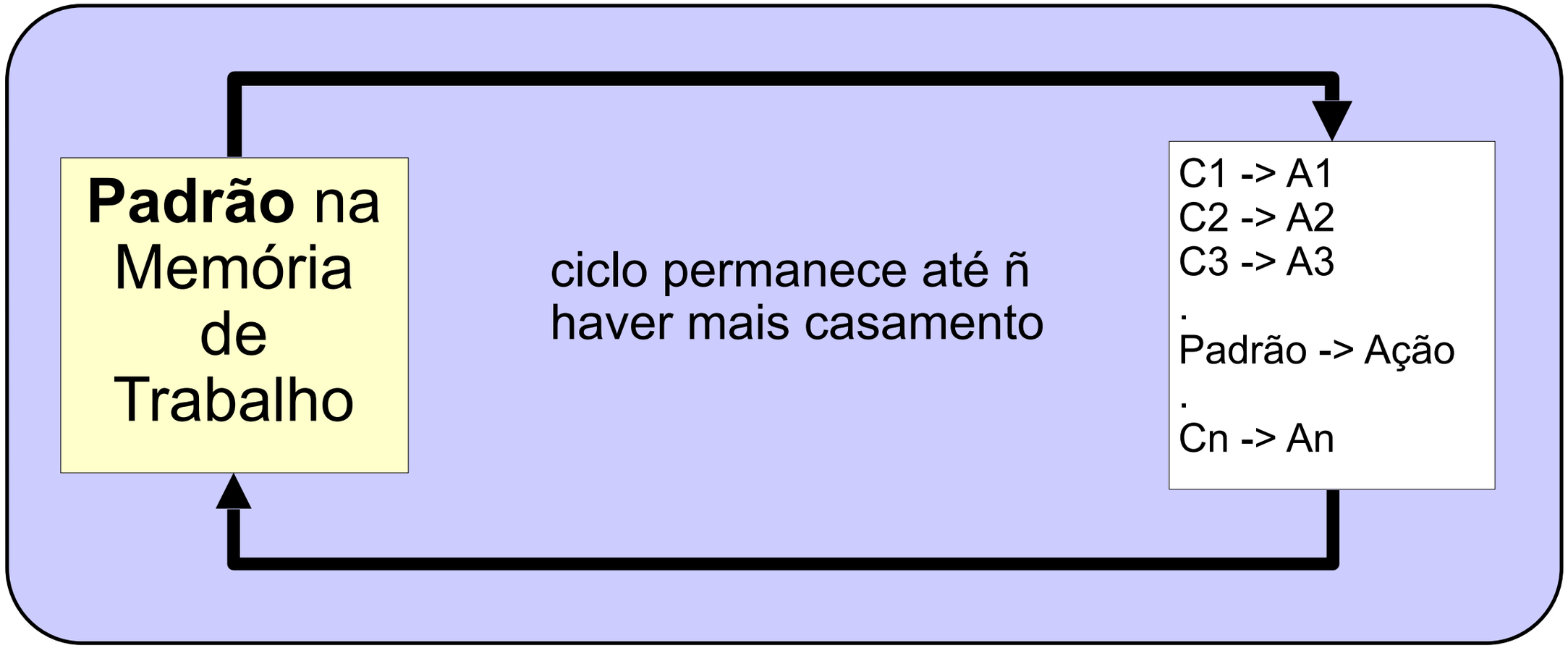
Agente Especialista

- Interpretador (unificação, casamento e execução)
- **Resolvedor de Conflito** (ordena segundo estratégias)



- **Volátil** e registra:
  - ações potenciais esperando por execução
  - hipóteses e alternativas que o sistema já tenha produzido

Sistemas de Produção



Resolvedor de conflitos

escolhe uma das regras
segundo **alguma estratégia**

Ex. Conjunto de produção:
(1) ba -> ab
(2) ca -> ac
(3) cb -> bc

Iteração	Memória Trab.	Conj. Conflito	Regra Disparada
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2,3	2
3	acbac	1,3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	-	Parar

Resolvedor de conflitos estratégias

1) Escolha aleatória

2) Ordem: considerar a primeira regra aplicável por ordem de apresentação

3) Especificidade: selecionar a regra aplicável com condições mais específicas

Ave(x) => Voa(x)

Ave(x) $\wedge$ Maior(Peso(x),100) => NaoVoa(x)

Ave(x) $\wedge$ Igual(x, Pinguim) => NaoVoa(x)

4) Proximidade: selecione uma regra baseado no quão recentemente ela foi utilizada

- mais recentemente utilizada
- menos recentemente utilizada (garante alguma chance para todas as regras)

5) Refratariedade: não usar uma regra que acabou de ser aplicada com os mesmos valores de variáveis. Isto evita loops.

Resolvedor de conflitos combinações

1) Descartar regras que já tenham sido utilizadas
2) Ordenar regras restantes em termos da Proximidade: as que cara com a 1a condição, depois a 2a condição, etc…
3) Ordenar regras pelo numero de condições
4) Selecionar aleatoriamente entre as sobreviventes

RETE

Sistemas antigos gastavam muito tempo no casamento

1. Memória de Trabalho muda pouco em cada ciclo
2. Muitas regras compartilham condições

=>

Passar Mem Trab incrementalmente através de uma **rede de testes**

Objetos que conseguem passar pela rede formam o conflito

Novo conjunto de conflito é gerado a partir do antigo

RETE

IF (Person father *y* age {< 14} name *x*)
(Person name *y* occupation doctor)
THEN ...

Sistema MYCIN

Desenvolvido em Stanford para ajudar médicos no tratamento de infecções bacterianas

500 regras para reconhecer 100 causas

IF

the type of *x* is primary bacteremia

the suspected entry point of *x* is the gastrointestinal tract

the site of the culture of *x* is one of the sterile sites

THEN

there is evidence that *x* is bacteroides

+

other more static data structures (not in WM)

- lists of organisms
- clinical parameters

Assistente SIRI

Desenvolvido pela Apple

Sistemas OOR

Linguagens híbridas (componentes de IA): regras + objetos

- **Ex: CLIPS, JESS, NeOpus, JEOPS, etc.**

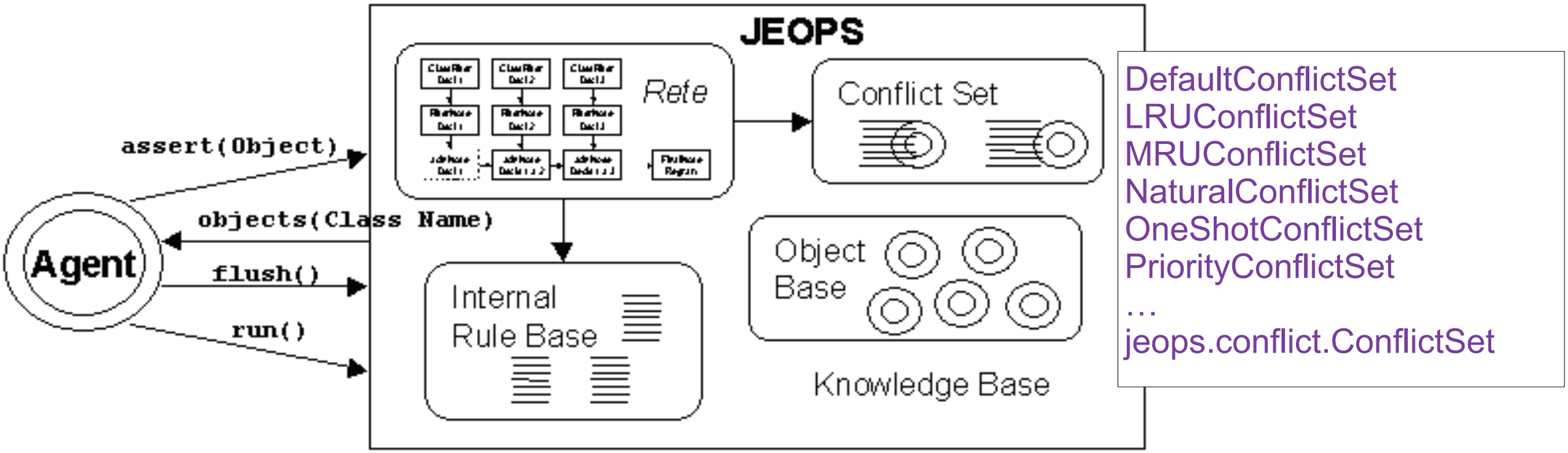
Regras são compiladas e fazem uso de **objetos Java**

Representam fatos na KB

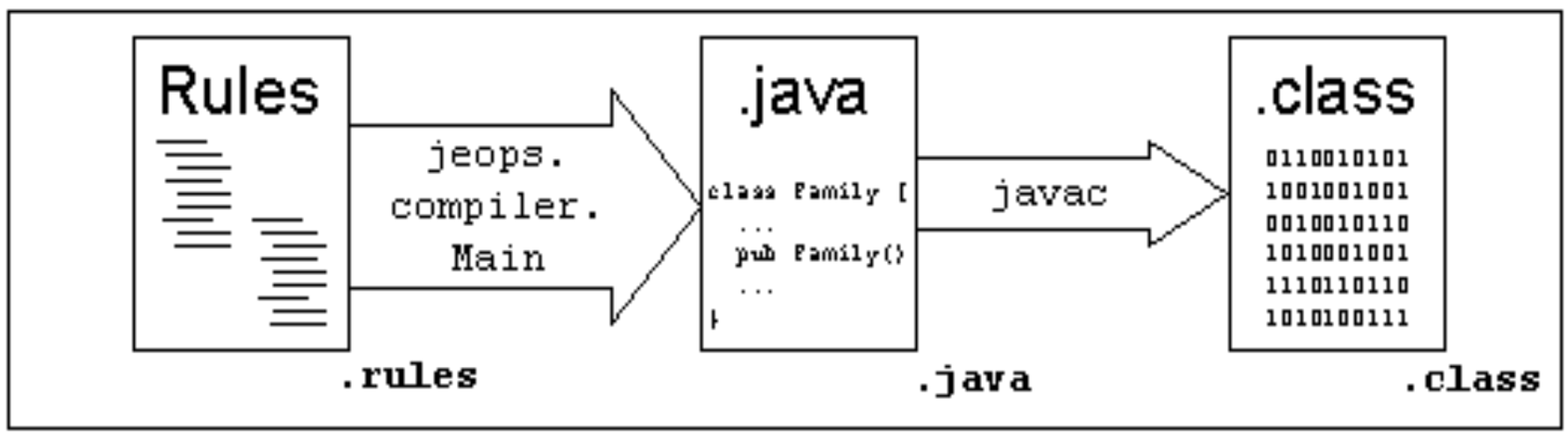
Hierarquia de Classes + atributos + métodos + relações
modelam Ontologia(s) do domínio!

JEOPS
Java Embedded Object Production Systems

Objetos Java + Regras de Produção




```
kb = new KB();
...
kb.insert(...);
....
kb.run();
```



"Se um comerciante vende um produto de que o cliente precisa, por um preço que o cliente possa pagar, então o negócio será fechado."

```
rule trade {
  declarations
    Salesman s;
    Customer c;
    Product p;
  conditions
    c.needs(p);
    s.owns(p);
    s.priceAskedFor(p) <= c.getMoney();
  actions
    s.sell(p);
    c.buy(p);
}
```

para qualquer instância de...