EA 072 Inteligência Artificial
em Aplicações Industriais

6 - Sistemas Baseados em Regras

Introdução

- Sistemas de produção:
  - origem: teoria formal de computação (Post, 1943)
  - modelo geral de computação
  - relevância: arquitetura para SBC
  - mecanismo de busca em grafos (*pattern–driven search*)

- Exemplos em IA
  - GPS (Newell e Simon, 1961)
    - modelo de solução de problemas por humanos
    - modularidade regras
    - separação conhecimento/controle
    - memória trabalho/conhecimento solução problemas
  - sistemas especialistas
  - linguagens (OPS5, CLIPS, JESS)

Sistemas de Produção

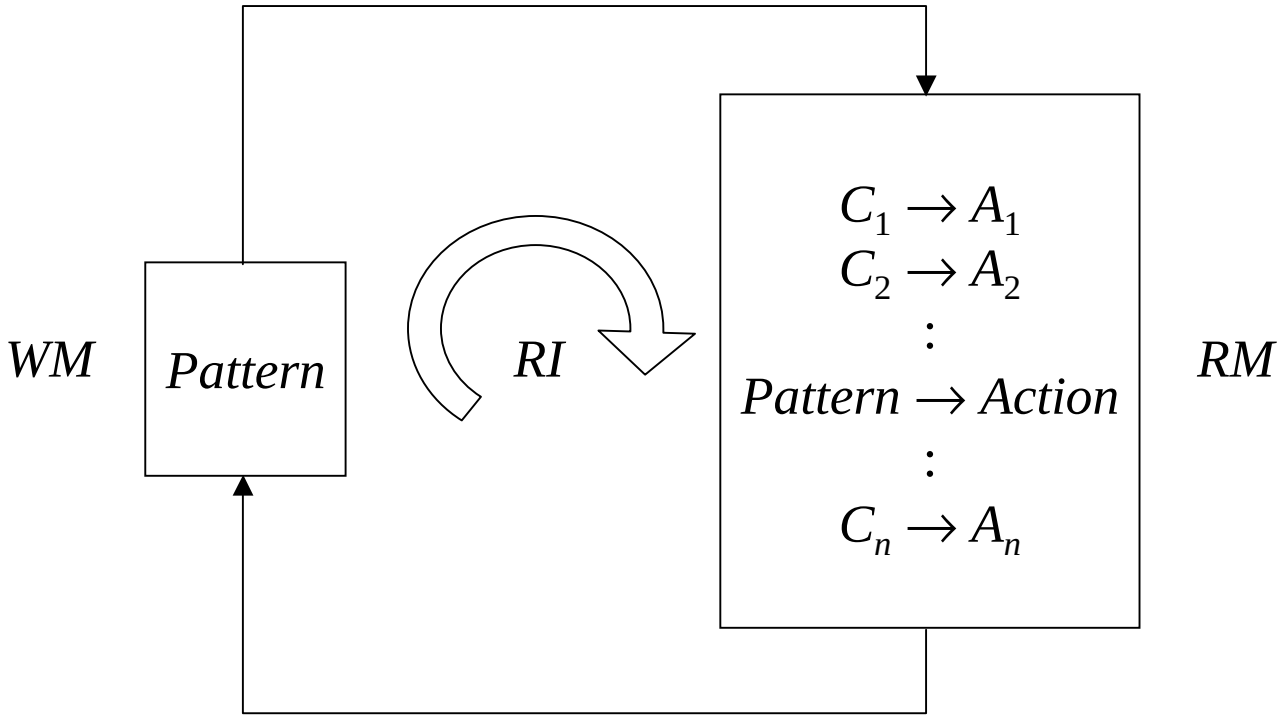
1–Memória de trabalho (*WM*): conjunto de dados que representa estado corrente do *mundo*

2–Regras de produção (*RM*): um conjunto de regras na forma $C \rightarrow A$

if < *condition in WM*> then <*action*>

3–Interpretador de regras (*RI*): módulo de controle que executa uma operação de comparação (*matching*) para determinar qual regra a ser ativada em um ciclo de processamento.

Sistema de Produção



function PRODUCTION_SYSTEM (*estado_inicial*)

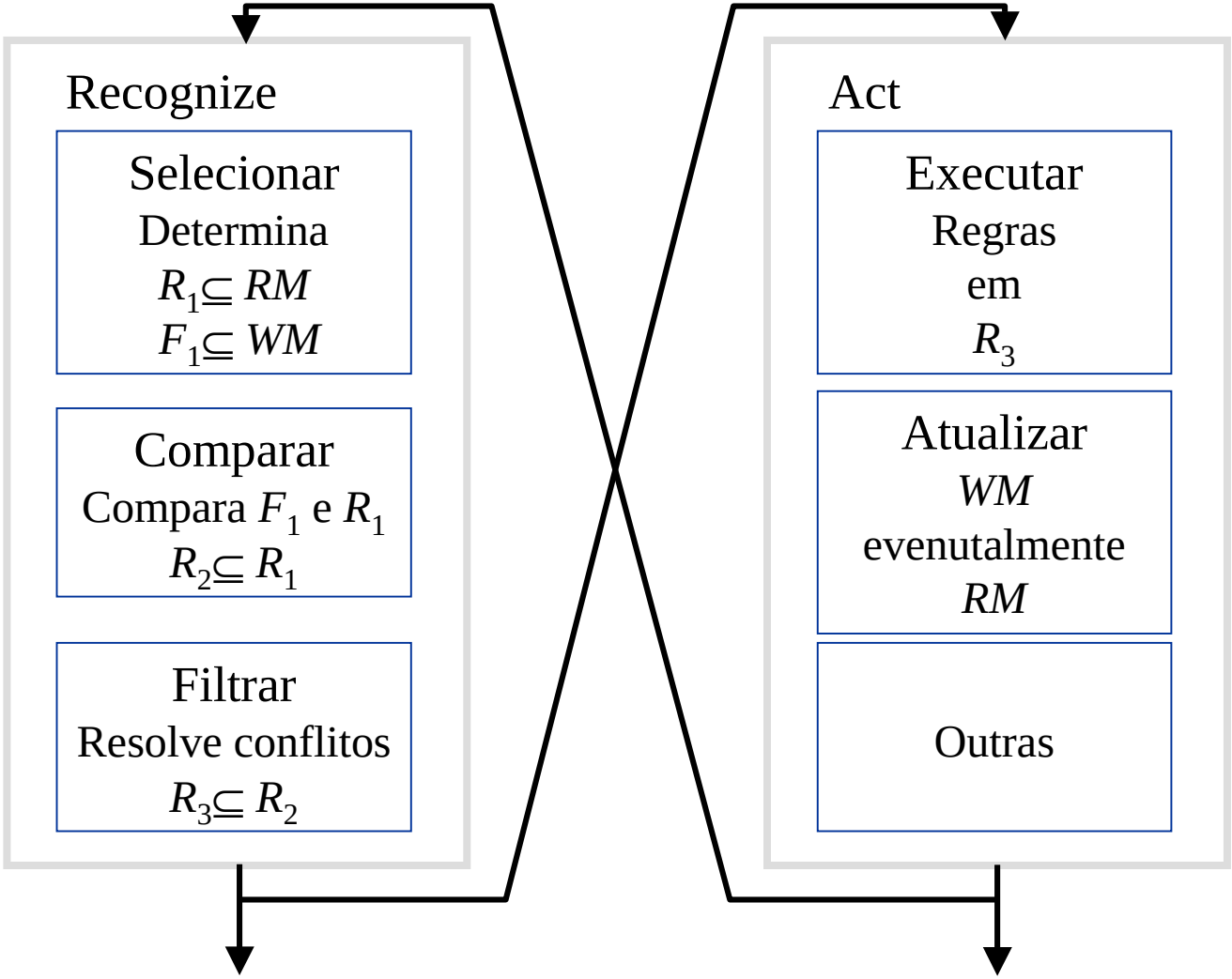
WM ← *estado_inicial*

until *WM* satisfaz condição parada **do**

begin *% RI recognize_and_act*
recognize selecionar regras da *RM*
ativadas pelos dados *WM* *% recognize*
WM ← resultado execução *R* à *WM* *% act*
end

- Interpretador de regras: executa ciclos *recognize-act*
  - seleciona de regras (*select*)
    - determina regras e dados da *WM* a serem considerados no ciclo
  - compara (*matching*)
    - compara dados na *WM* com antecedentes das regras da *RM*
  - determina as regras ativas e formar o conjunto conflito
  - resolve conflitos (*scheduling*, *filter*)
    - escolhe regra conjunto conflito a ser disparada
  - ação (*execution*, *action*)
    - executa (disparar) regra escolhida
    - insere resultado da ação na *WM*

Interpretador de regras ≡ máquina de inferência



Exemplo

Regras Produção

1. ba $\rightarrow$ ab
2. ca $\rightarrow$ ac
3. cb $\rightarrow$ bc

RM

Iteração	*WM*	Conflito	Regra
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	$\varnothing$	pára

Exemplo

Start state:

2	8	3
1	6	4
7		5

Goal state:

1	2	3
8		4
7	6	5

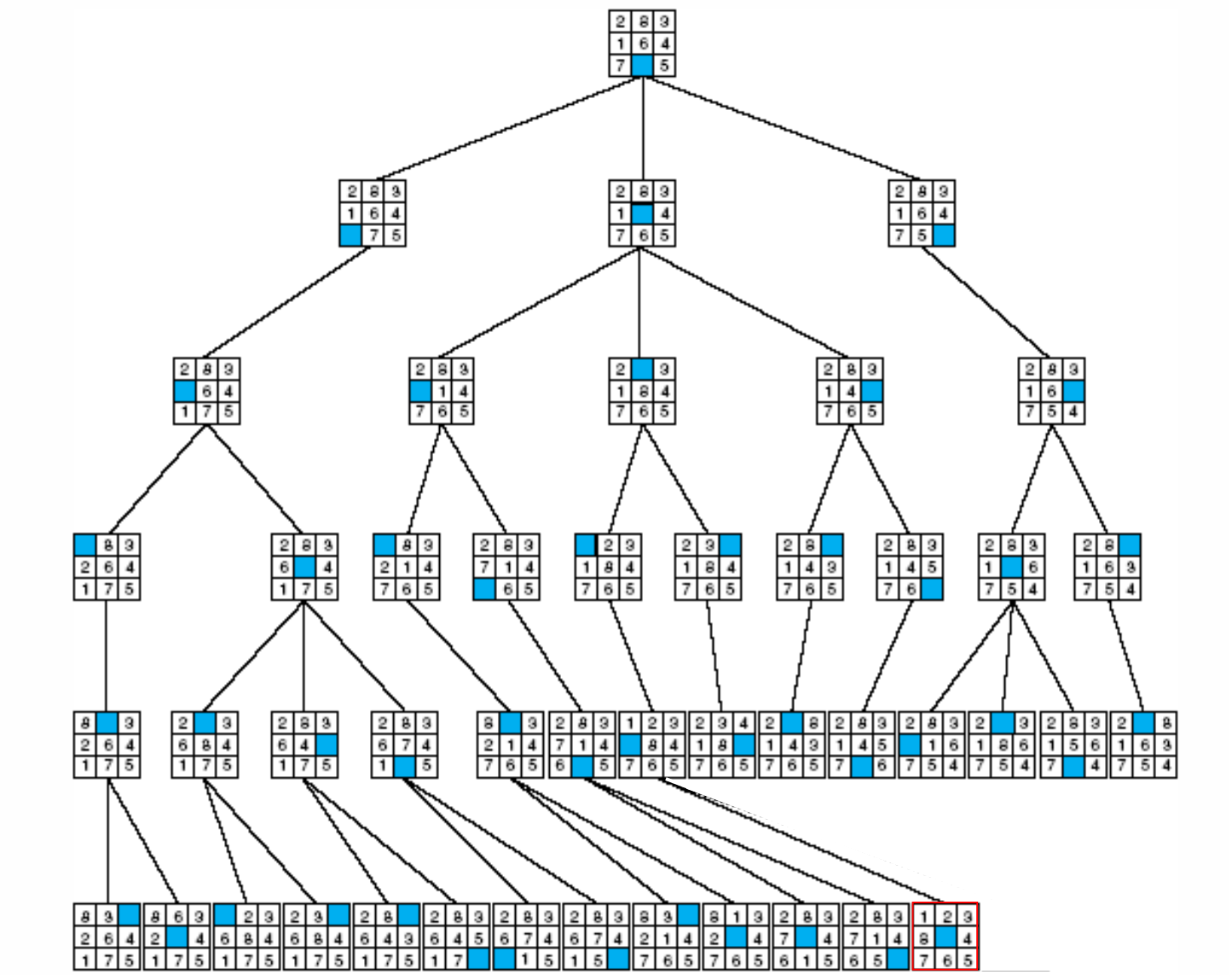
Production set:

Condition		Action
goal state in working memory	→	halt
blank is not on the left edge	→	move the blank left
blank is not on the top edge	→	move the blank up
blank is not on the right edge	→	move the blank right
blank is not on the bottomedge	→	move the blank down

Working memory is the present board state and goal state.

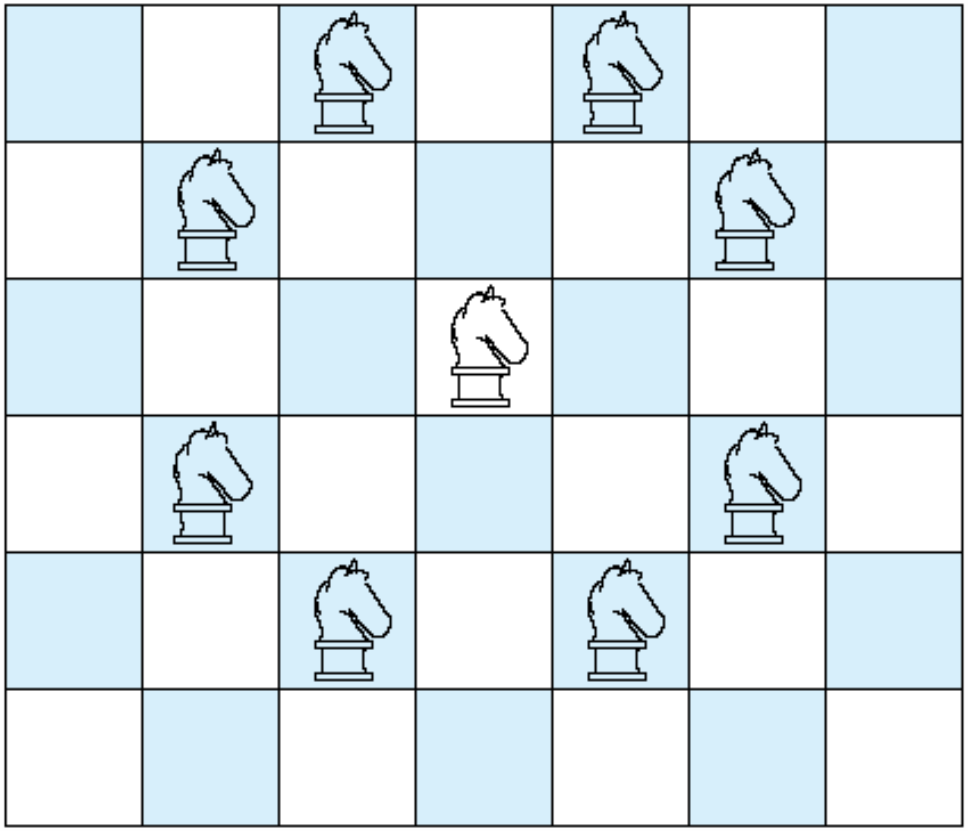
Control regime:

1. Try each production in order.
2. Do not allow loops.
3. Stop when goal is found.



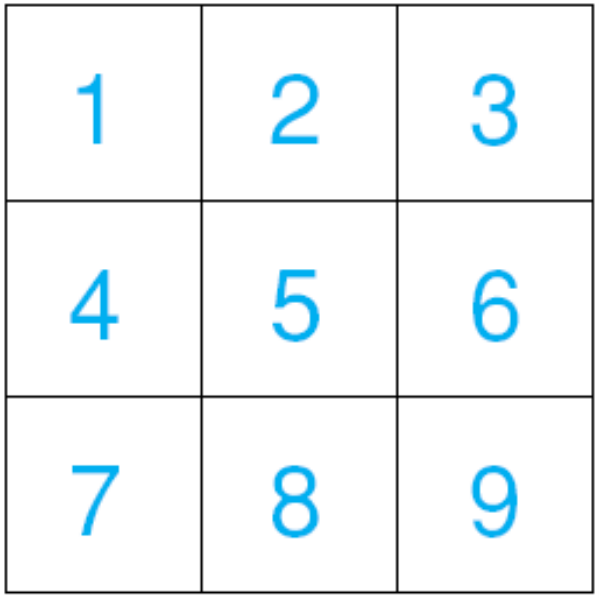
sem ciclos e limite profundidade 5

Exemplo



- Caso simplificado: tabuleiro 3×3

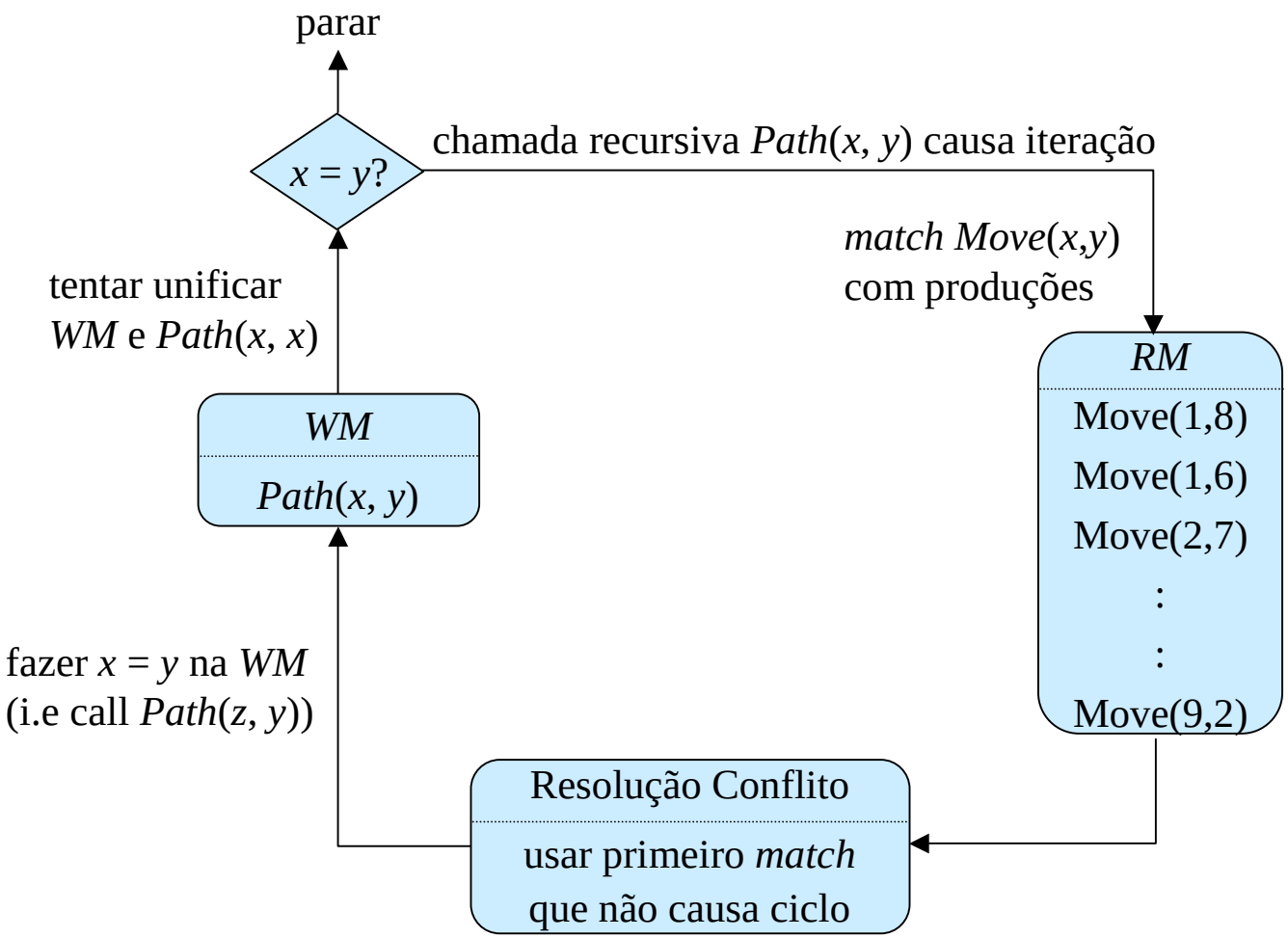
move(1,8)	move(6,1)
move(1,6)	move(6,7)
move(2,9)	move(7,2)
move(2,7)	move(7,6)
move(3,4)	move(8,3)
move(3,8)	move(8,1)
move(4,9)	move(9,2)
move(4,3)	move(9,4)



RULE #	CONDITION		ACTION
1	knight on square 1	→	move knight to square 8
2	knight on square 1	→	move knight to square 6
3	knight on square 2	→	move knight to square 9
4	knight on square 2	→	move knight to square 7
5	knight on square 3	→	move knight to square 4
6	knight on square 3	→	move knight to square 8
7	knight on square 4	→	move knight to square 9
8	knight on square 4	→	move knight to square 3
9	knight on square 6	→	move knight to square 1
10	knight on square 6	→	move knight to square 7
11	knight on square 7	→	move knight to square 2
12	knight on square 7	→	move knight to square 6
13	knight on square 8	→	move knight to square 3
14	knight on square 8	→	move knight to square 1
15	knight on square 9	→	move knight to square 2
16	knight on square 9	→	move knight to square 4

Iteration #	Working memory		Conflict set (rule #'s)	Fire rule
	Current square	Goal square		
0	1	2	1, 2	1
1	8	2	13, 14	13
2	3	2	5, 6	5
3	4	2	7, 8	7
4	9	2	15, 16	15
5	2	2		Halt

- Algoritmo recursivo como sistema de produção



$\forall x \quad Path(x, x)$

$\forall x\, y\, Path(x, y) \Leftarrow \exists z\, (Move(x, z) \wedge Path(z, x))$

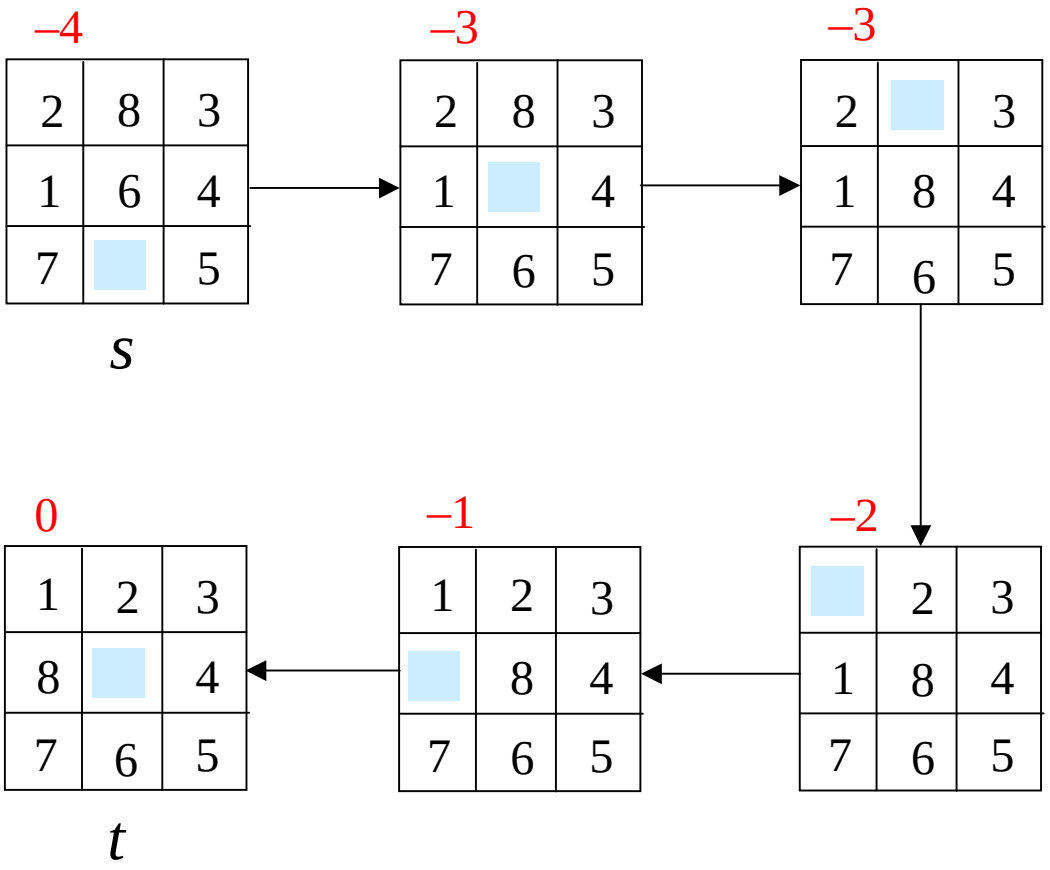
- Estratégias de seleção de regras (estratégias de controle)

  – irrevogável: regra escolhida é aplicada, sem reconsiderações posteriores
  Exemplo: *hill-climbing*

  – tentativa: regra escolhida é aplicada, mas é possível voltar atrás posteriormente para aplicar uma outra regra
  Exemplo: busca em grafos, *backtracking*

O funcionamento de um sistema de produção pode ser caracterizado como um processo de busca pois na maioria das aplicações em IA, a estratégia de seleção de regras não é suficiente para determinar a regra mais apropriada a ser disparada em cada ciclo (iteração, passo).

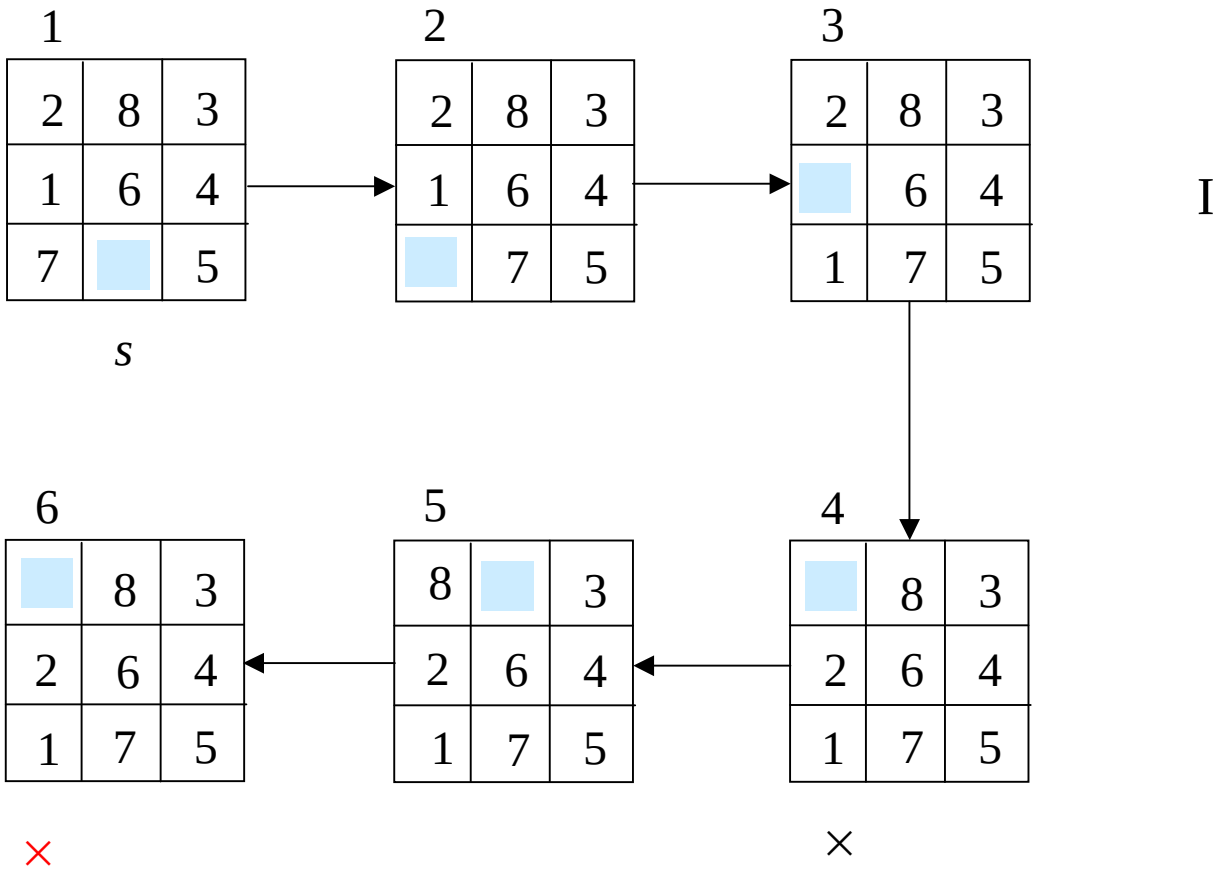
Exemplo: *Hill-climbing*

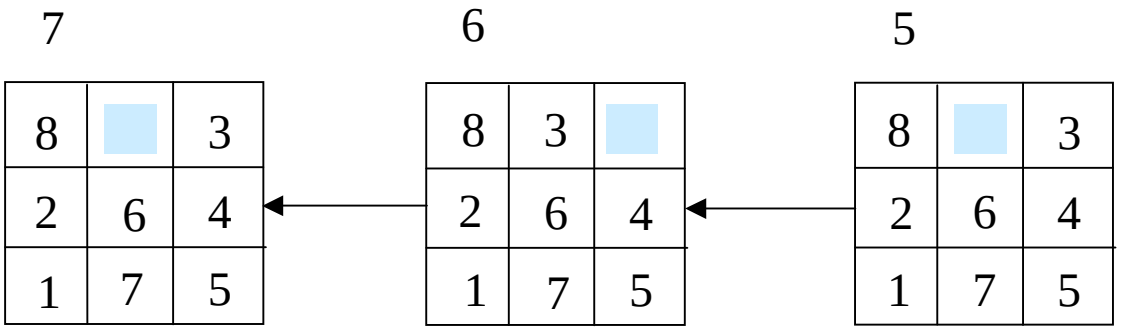


$f(x)$ = número de peças fora do lugar correto

x : estado s : estado inicial t : meta

Exemplo: *Bactraking*



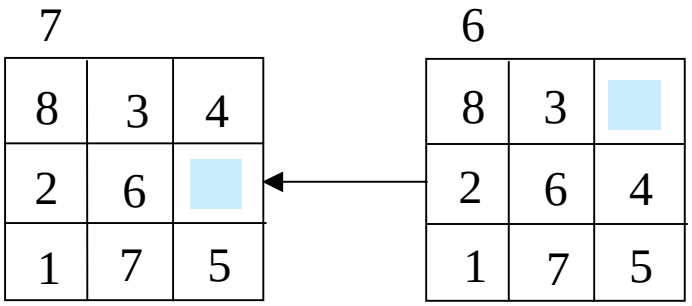


II

×

este estado já ocorreu antes: voltar para 6

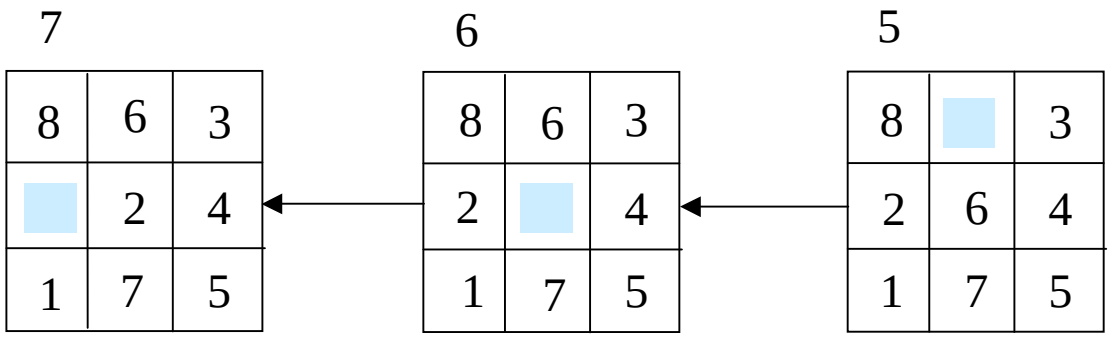
×



III

×

aplicamos todas regras a 6 e não atingimos meta: voltar para 5



IV

×

aplicamos todas regras e a meta não foi atingida, etc….

- Estratégias de seleção de regras

  – *data-driven*: em cada ciclo, antecedentes das regras na *RM* são examinados para determinar quais regras são ativadas pela *WM*
  Exemplo: *forward chaining* (encadeamento direto)

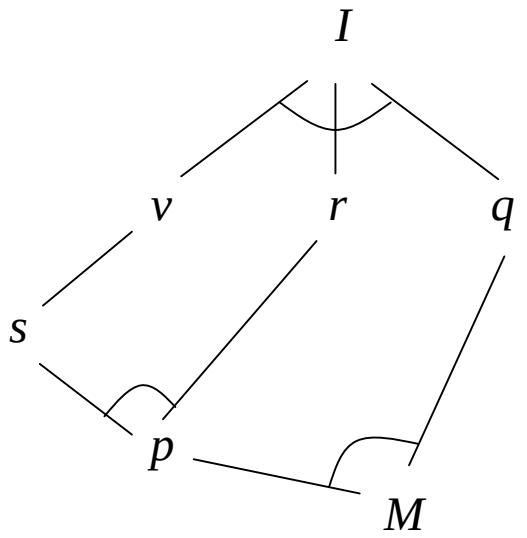
  – *goal-driven*:em cada ciclo, consequentes das regras na *RM* são examinados para verificar se algum deles contém a meta. A seguir o sistema determina os fatos necessários para ativar antecedente
  Exemplo: *backward chaining* (encadeamento reverso)

Exemplo: Encadeamento direto

Ciclo	*WM*	Regras Ativas	Regra Disparada
0	*I*	6	6
1	*I*, *v*, *r*, *q*	6, 5	5
2	*I*, *v*, *r*, *q*, *s*	6, 5, 2	2
3	*I*, *v*, *r*, *q*, *s*, *p*	6, 5, 2, 1	1
4	*I*, *v*, *r*, *q*, *s*, *p*, *M*	6, 5, 2, 1	parar

RM

1. $p \wedge q \rightarrow M$
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow q$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. $I \rightarrow v \wedge r \wedge q$

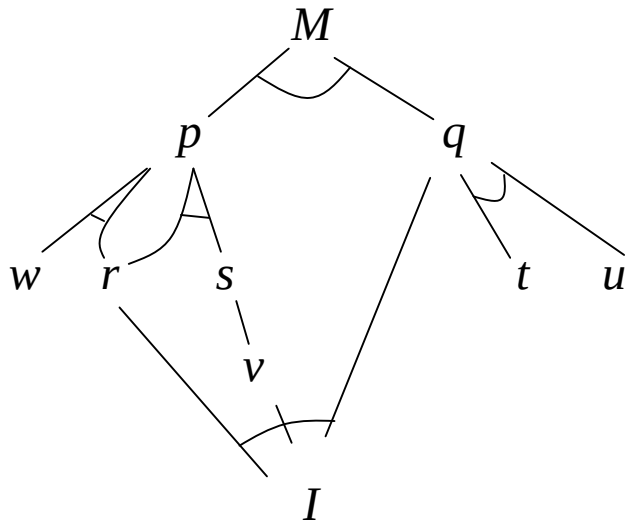


Exemplo: Encadeamento reverso

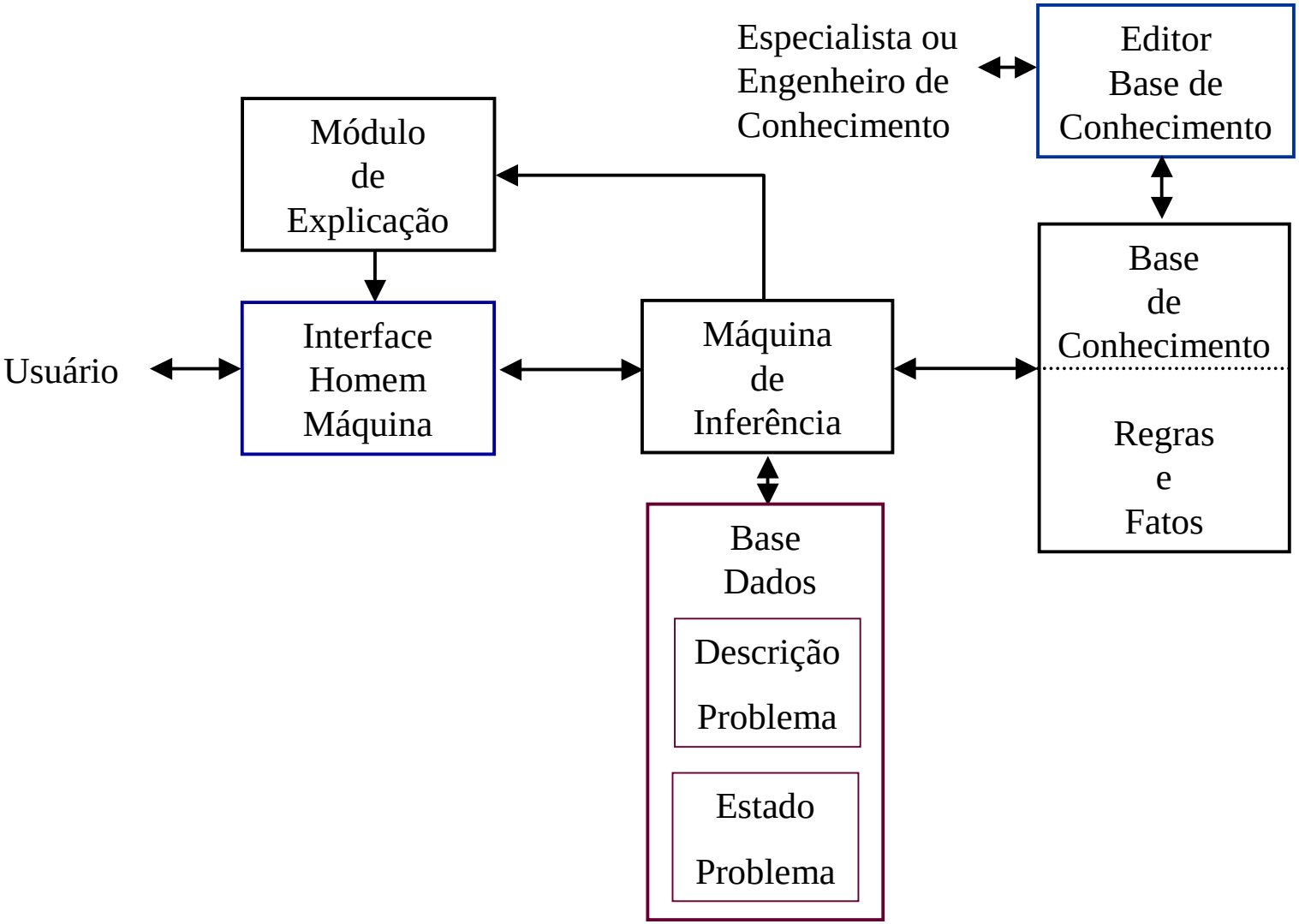
Ciclo	*WM*	Regras Ativas	Regra Disparada
0	M	1	1
1	M, p, q	1, 2, 3, 4	2
2	M, p, q, r, s	1, 2, 3, 4, 5	3
3	M, p, q, r, s, w	1, 2, 3, 4, 5	4
4	M, p, q, r, s, w, t, u	1, 2, 3, 4, 5	5
5	$M, p, q, r, s, w, t, u, v$	1, 2, 3, 4, 5, 6	6
6	$M, p, q, r, s, w, t, u, v, I$	1, 2, 3, 4, 5, 6	parar

RM

1. $p \wedge q \rightarrow M$
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow p$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. $I \rightarrow v \wedge r \wedge q$



Sistemas Especialistas: Arquitetura

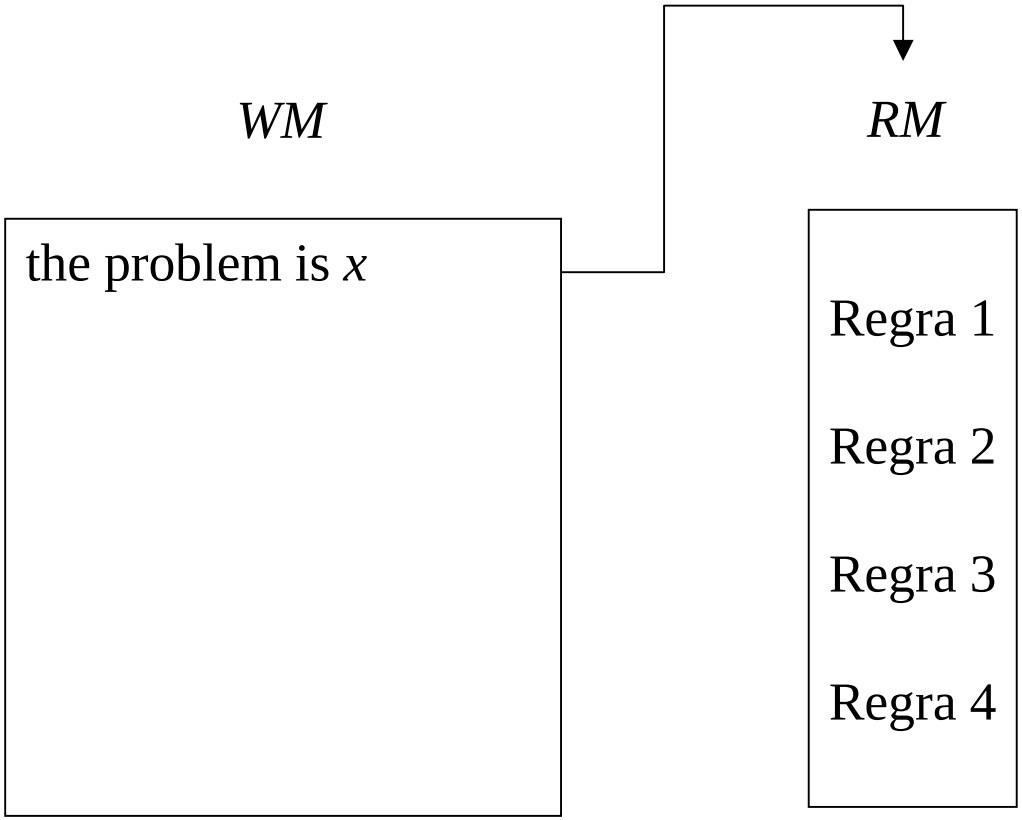


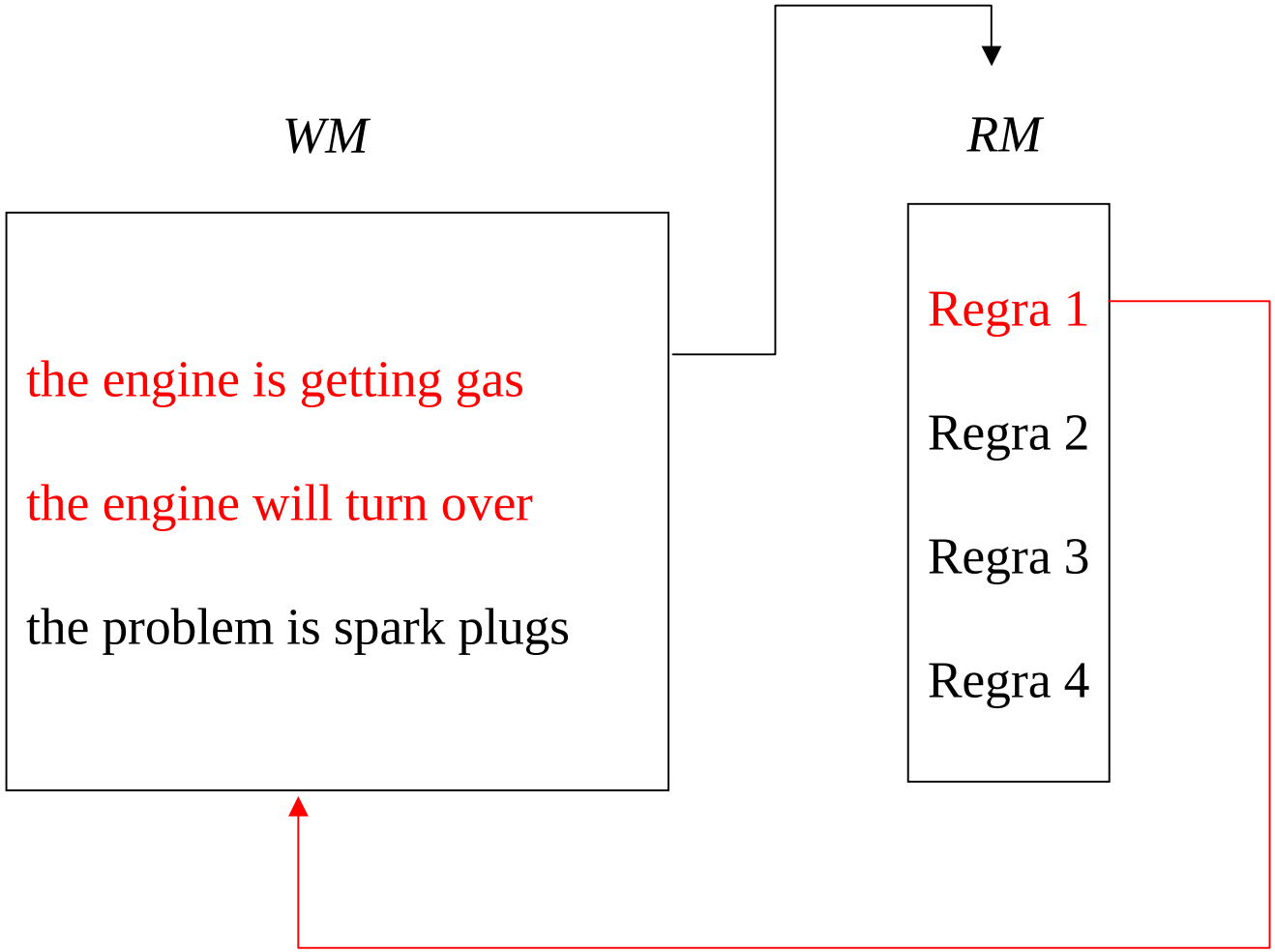
Exemplo: Diagnóstico

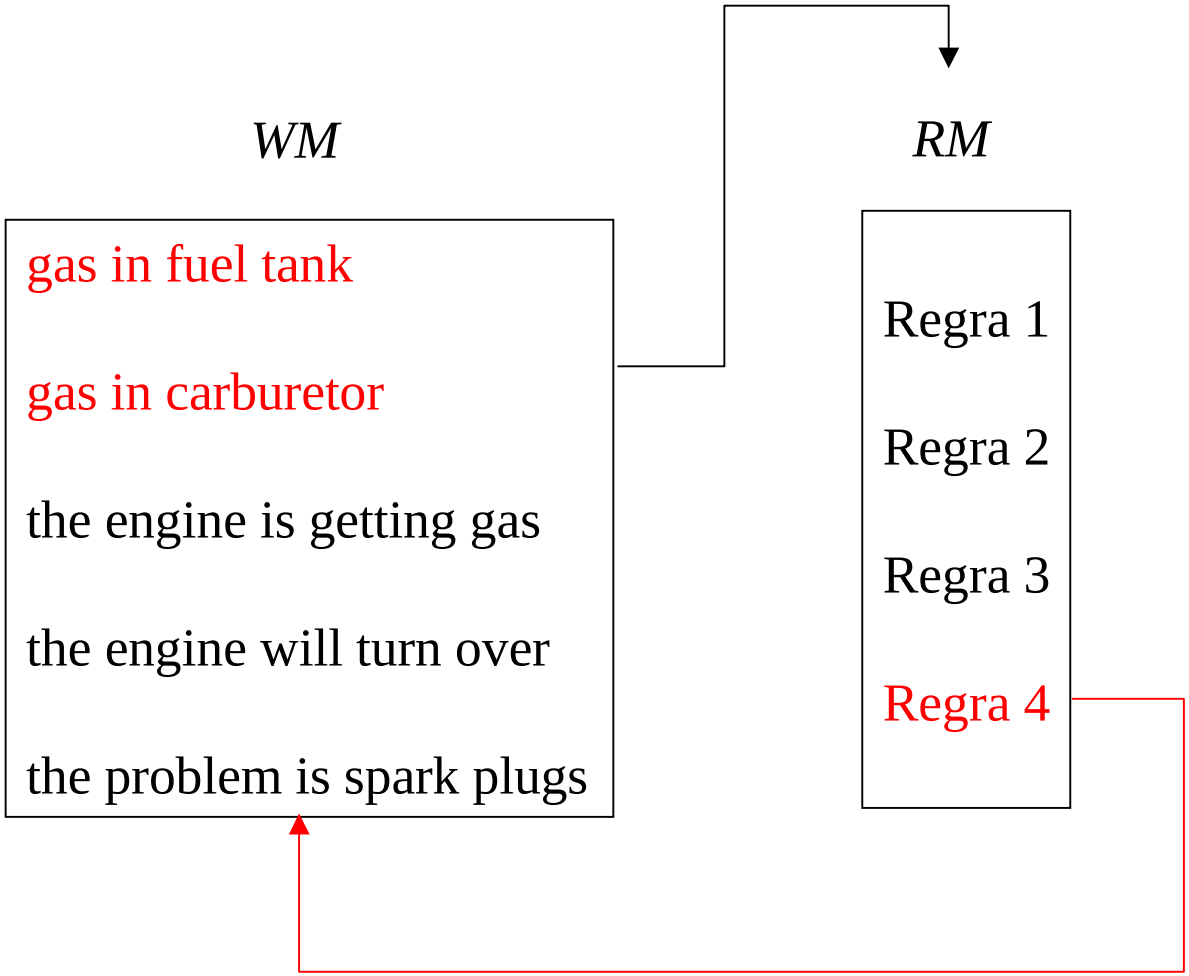
- Base de regras (*KB*)

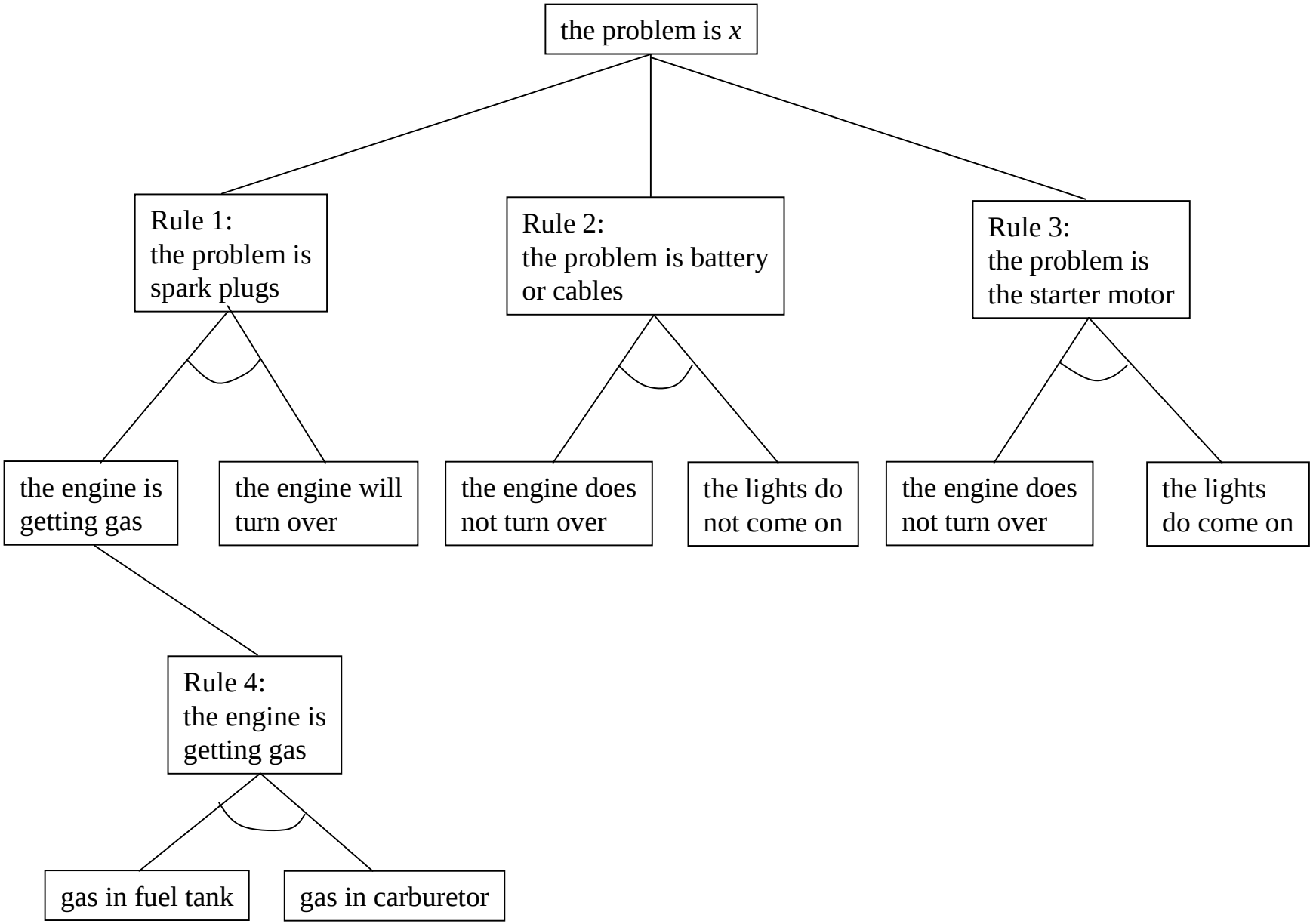
Regra 1:	if	engine is getting gas and the engine will turn over
	then	the problem is spark plugs
Regra 2:	if	the engine does not turn over and the lights do not come on
	then	the problem is battery or cables
Regra 3:	if	the engine does not turn over and the lights do come on
	then	the problem is the starter motor
Regra 4:	if	there is gas in the fuel tank and there is gas in the carburetor
	then	the engine is getting gas

Ask (*KB*, the problem is *x*)









Executando o sistema especialista

SE: gas in the fuel tank ?

U: Yes

SE: gas in carburetor?

U: Yes

SE: engine will turn over?

U: why

SE: it has been established that:
1-engine is getting gas
therefore if
2-engine will turn over
then the problem is spark plugs

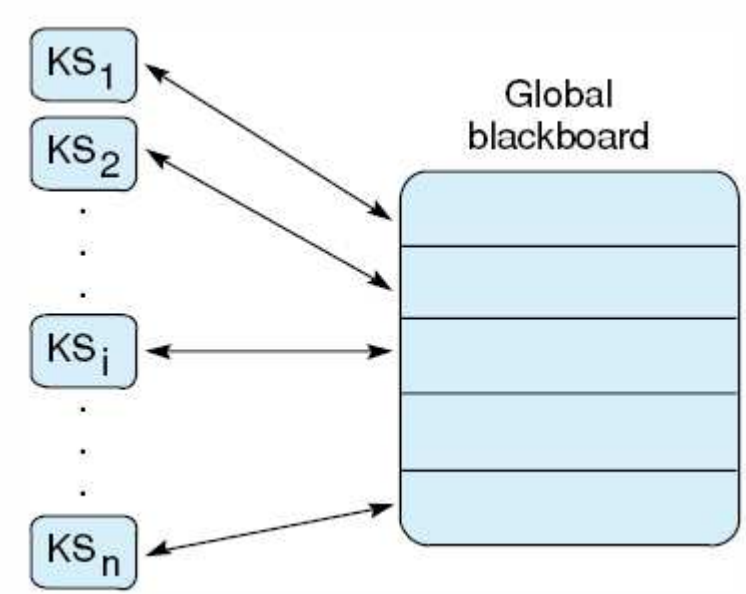
U: how engine is getting gas

SE: this follows from rule 4:

if gas in fuel tank, and
gas in carburetor
then engine is getting gas

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

Arquitetura Blackboard



Observação

Este material refere-se às notas de aula do curso EA 072 Inteligência Artificial em Aplicações Industriais da Faculdade de Engenharia Elétrica e de Computação da Unicamp. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.