Aula 3 – Sistemas de Produção

Parte 1 – Inferência Lógica
22705/1001336 - Inteligência Artificial
2019/1 - Turma A
Prof. Dr. Murilo Naldi

naldi@dc.ufscar.br

Agradecimentos

- Agradecimentos pela base do material utilizado nesta aula foi cedido ou adapatado do material dos professores Heloísa Camargo e Ricardo Cerri e Andréia Bonfante.

Na aula anterior

- Vimos como conhecimento pode ser representado
- Dois exemplos:
  - Através de lógica
    - Estabelecer um modelo
  - Uso de categorias
    - Facilite organização e generalização

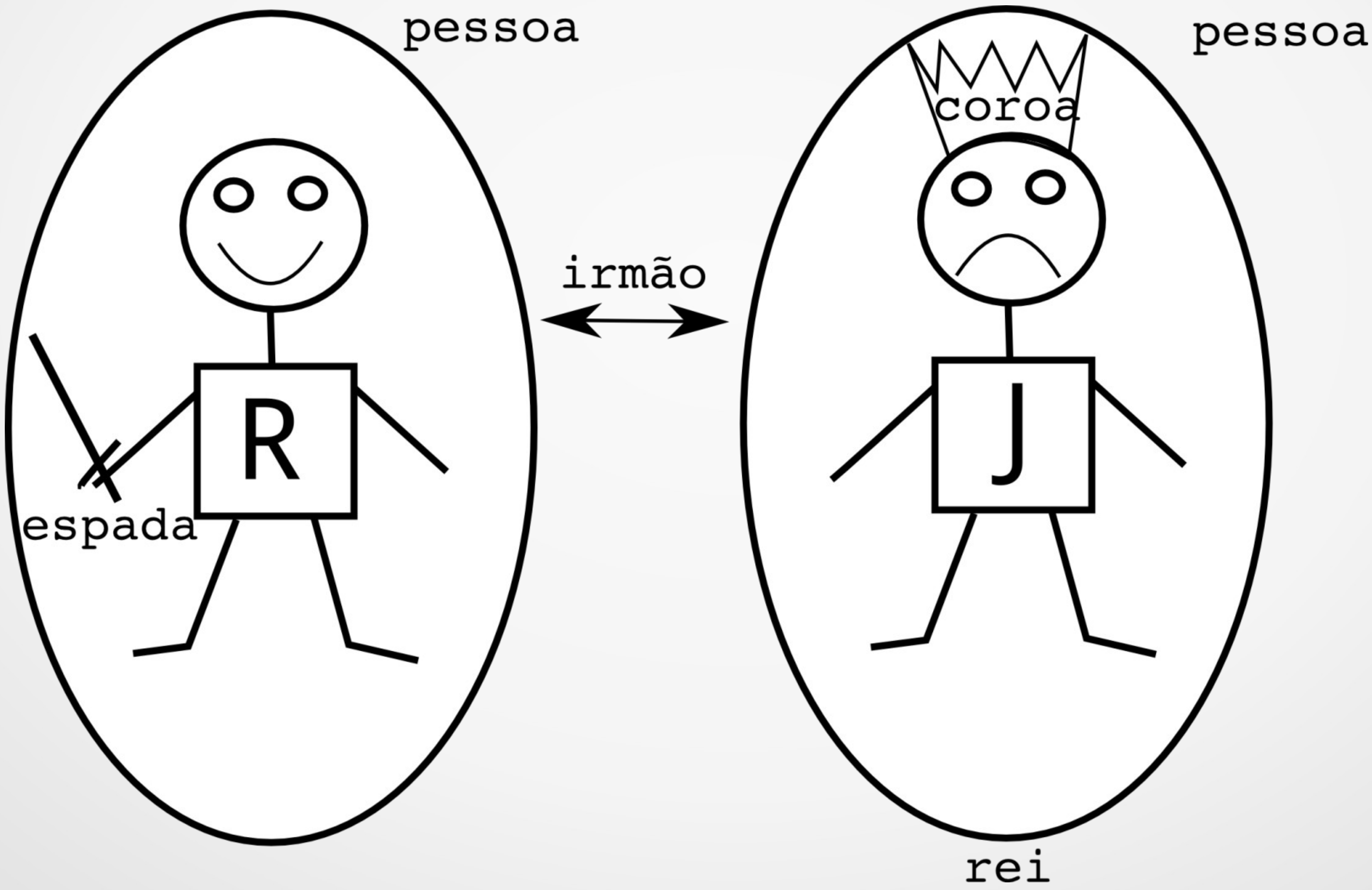
Lógica de Primeira Ordem

- Vimos na aula anterior que lógica de primeira ordem é uma das formas mais naturais de representar conhecimento
  - Também categorias, representadas por predicados e quantificadores $\forall$ e $\exists$
  - Vimos também algumas regras de inferência importantes, como *Modus Ponens*:

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

Exemplo de modelagem

- Podemos modelar um cenário usando LPO



Exemplo de modelagem

- No exemplo:
  - Constantes: ricardo, joao, coroa, espada
  - Predicados: pessoa, rei, irmao, pertence
- Relações:
  - pessoa(ricardo), pessoa(joao)
  - irmao(ricardo,joao), irmao(joao,ricardo)
  - pertence(espada,ricardo), pertence(coroa,joao)
  - rei(joao)

Modelo do cenário

- Como modelar o cenário usando LPO?
- “João é irmão de Ricardo”
  - ???
- “Se João tem coroa, então ele é rei”
  - ???
- “Todo rei possui sua coroa”
  - ???
- “O pai de uma pessoa é pai do irmão dessa pessoa”
  - ???

Modelo do cenário

- Como modelar o cenário usando LPO?
- “João é irmão de Ricardo”
  - *irmao*(*joao*,*ricardo*)
- “Se João tem coroa, então ele é rei”
  - ???
- “Todo rei possui sua coroa”
  - ???
- “O pai de uma pessoa é pai do irmão dessa pessoa”
  - ???

Modelo do cenário

- Como modelar o cenário usando LPO?
- "João é irmão de Ricardo"
  - *irmao*(*joao*,*ricardo*)
- "Se João tem coroa, então ele é rei"
  - *pertence*(*coroa*,*joao*) → rei(*joao*)
- "Todo rei possui sua coroa"
  - ???
- "O pai de uma pessoa é pai do irmão dessa pessoa"
  - ???

Modelo do cenário

- Como modelar o cenário usando LPO?
- “João é irmão de Ricardo”
  - *irmao*(*joao*,*ricardo*)
- “Se João tem coroa, então ele é rei”
  - *pertence*(*coroa*,*joao*) → rei(*joao*)
- “Todo rei possui sua coroa”
  - $\forall$ *X rei*(*X*) → *pertence*(*coroa*(*X*), *X*)
- “O pai de uma pessoa é pai do irmão dessa pessoa”
  - ???

Modelo do cenário

- Como modelar o cenário usando LPO?
- “João é irmão de Ricardo”
  - *irmao*(*joao*,*ricardo*)
- “Se João tem coroa, então ele é rei”
  - *pertence*(*coroa*,*joao*) → rei(*joao*)
- “Todo rei possui sua coroa”
  - $\forall$ *X rei*(*X*) → *pertence*(*coroa*(*X*), *X*)
- “O pai de uma pessoa é pai do irmão dessa pessoa”
  - $\forall$ *X, Y, Z pai*(*X*,*Y*) $\wedge$ *irmao*(*Y*,*Z*) → *pai*(*X*,*Z*)

Inferência em LPO

- A inferência em LPO pode ser obtida reduzindo a base de conhecimento em lógica proposicional e utilizando inferência proposicional.
- Para isso é necessário aplicar algumas regras para os quantificadores universal e existencial.

Instanciação Universal

- Consiste em gerar todas as instancias possíveis para uma determinada variável, substituindo-a:
- Exemplo:
  - $\forall\ X\ rei(X) \wedge ganancioso(X) \rightarrow mal(X)$
- A instanciação universal deve ver obtida com a substituição {*X*/*joão*}, {*X*/*ricardo*}, {*X*/pai(*joão*)}...

Instanciação Universal

- Substituindo:
  - $rei(joão) \wedge ganancioso(joão) \rightarrow mal(joão)$
  - $rei(ricardo) \wedge ganancioso(ricardo) \rightarrow mal(ricardo)$
  - $rei(\text{pai}(joão)) \wedge ganancioso(\text{pai}(joão)) \rightarrow mal(\text{pai}(joão))$
  - ...
  - …

Instanciação Existencial

- Consiste em substituir a variável da sentença por um símbolo constante qualquer que não apareça em nenhum outro lugar da base de conhecimento.
- Exemplo:
  - $\exists\, X\ coroa(X) \wedge nacabeça(X, joão)$
- Substituindo ficaria:
  - $coroa(c_1) \wedge nacabeça(c_1, joão)$
- de forma que c_1 não existia na base de conhecimento.

Forma Normal Conjuntiva

- Uma sentença está na Forma Normal Conjuntiva (FNC) se, e somente se está na forma:
  - $a_1 \wedge a_2 \wedge a_3 \wedge ... \wedge a_n$ com $n \geq 1$
- onde cada a_i está na forma:
  - $b_1 \vee b_2 \vee b_3 \vee ... \vee b_m$ com $m \geq 1$
- e cada b_j é uma variável proposicional, ou a negação de uma.

Forma Normal Conjuntiva

- Para toda sentença existe uma FNC equivalente, que pode ser obtida assim:
  - Elimine $\leftrightarrow$ e $\rightarrow$ substituindo:
  - $a \rightarrow b$ por $\neg a \lor b$
  - $a \leftrightarrow b$ por $(\neg a \lor b) \land (a \lor \neg b)$
- Reduza o escopo de $\neg$ substituindo até não poder mais:
  - $\neg\neg\, a$ por a
  - $\neg(a \lor b)$ por $(\neg a \land \neg b)$
  - $\neg(a \land b)$ por $(\neg a \lor \neg b)$

Forma Normal Conjuntiva

- Em seguida, coloque toda disjunção para as posições internas das fórmulas por meio de distribuição:
  - $a \vee (b \wedge c)$ por $(a \vee b) \wedge (a \vee c)$
- Desta forma, os termos são formados por disjunções e ligados por conjunções

Forma Normal Conjuntiva

- Diferentes operadores ∨ possuem precedência igual e o resultado não muda com a ordem de aplicação
  - $(a \vee b) \vee (c \vee d) = a \vee b \vee c \vee d$
- O mesmo pode ser feito com o operador ∧
  - $(a \wedge b) \wedge (c \wedge d) = a \wedge b \wedge c \wedge d$
- Contudo, ∨ e ∧ possuem precedência distintas ou seja:
  - $(a \wedge b) \vee (c \wedge d) \neq a \wedge b \vee c \wedge d$

Exercícios

- Converta para FNC as seguintes fórmulas:
  - $(A \rightarrow B) \rightarrow C$
  - $A \rightarrow (B \rightarrow C)$
  - $(A \rightarrow B) \vee (B \rightarrow A)$
  - $A \rightarrow B) \rightarrow (\neg C \rightarrow (D \wedge E))$

Inferência LPO

- Para inferência é essencial que se possa aplicar a regra de *Modus Ponens*
- Como generalizar essa regra para que ela possa ser utilizada na LPO?

Modus Ponens Generalizado

- Para sentenças atômicas p_i, p_i' *e* q, *onde há uma substituição* θ *tal que* $subst(\theta, p_i') = subst(\theta, p_i)$ para todo i:
    - $p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q)$
    - $subst(\theta, q)$
- existem $n+1$ premissas a essa regra: as n sentenças atômicas p_i' e a implicação de q.
- É importante observar que a premissa deve estar na **FNC**

Modus Ponens Generalizado

- Exemplo:
  - $\forall\ X\ rei(X) \wedge ganancioso(X) \rightarrow mal(X)$
  - $p_1 = rei(X)$
  - $p_2 = ganancioso(X)$
  - $q = mal(X)$
- aplicando a substituição:
  - $\theta = \{X/joão\}$
  - $p_1' = rei(joão)$
  - $p_2' = ganancioso(joão)$
- Sabendo que $p_1' \wedge p_2' \rightarrow q$, logo podemos utilizar a regra
  - $subst(\theta, q) = mal(joão)$

Unificação

- Algumas substituições fazem com que diferentes expressões lógicas se mostrem idênticas!
- Esse processo é chamado de unificação e é o componente chave para todos os algoritmos de inferência de primeira ordem.

Unificação

- Tomemos um procedimento unifica que pega duas sentenças e retorna um unificador para elas, se este existir.

$$unifica(P,Q) = \theta, \quad onde\ subst(\theta, P) = subst(\theta, Q)$$

- Esse unificador é a substituição que torna as duas sentanças idênticas!

Unificação

- A *unifica*(*P*,*Q*) é bem sucedida se:
  - *P* e *Q* forem constantes iguais
  - *P* e/ou *Q* forem variáveis e assumirem um único valor por variável
  - *P* e *Q* forem funções ou predicados que:
    - possuam o mesmo identificador (funtor);
    - possuam a mesma aridade
    - cada elemento também se unifique

Unificação

- Exemplos:
  - *unifica*(*A* , *A*) = ???
  - *unifica*(*unifica*(*A*, *B*), *unifica*(*B*, *abc*)) = ???
  - *unifica*(*unifica*(*xyz*, *C*), *unifica*(*C*, *D*)) = ???

Unificação

- Exemplos:
  - *unifica*(*A* , *A*) = {*A*/*A*} (tautologia)
  - *unifica*(*unifica*(*A*, *B*), *unifica*(*B*, *abc*)) = ???
  - *unifica*(*unifica*(*xyz*, *C*), *unifica*(*C*, *D*)) = ???

Unificação

- Exemplos:
  - $unifica(A\ ,\ A) = \{A/A\}$ (tautologia)
  - $unifica(unifica(A, B), unifica(B, abc)) = \{A/abc, B/abc\}$
  - $unifica(unifica(xyz, C), unifica(C, D)) =$???

Unificação

- Exemplos:
  - *unifica*(*A* , *A*) = {*A*/*A*} (tautologia)
  - *unifica*(*unifica*(*A*, *B*), *unifica*(*B*, *abc*)) = {*A*/*abc*, *B*/*abc*}
  - *unifica*(*unifica*(*xyz*, *C*), *unifica*(*C*, *D*)) = {C/*xyz*, *D*/*xyz*} (a unificação é simétrica)

Unificação

- Exemplos:
  - *unifica*(*abc*, *abc*) = ???
  - *unifica*(*abc*, *xyz*) = ???
  - *unifica*(*f*(*A*), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*abc*, *abc*) = A unificação é bem sucedida
  - *unifica*(*abc*, *xyz*) = ???
  - *unifica*(*f*(*A*), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*abc*, *abc*) = A unificação é bem sucedida
  - *unifica*(*abc*, *xyz*) = Falha em unificar porque são constantes diferentes
  - *unifica*(*f*(*A*), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*abc*, *abc*) = A unificação é bem sucedida
  - *unifica*(*abc*, *xyz*) = Falha em unificar porque são constantes diferentes
  - *unifica*(*f*(*A*), *f*(*B*)) = {*A*/*B*}

Unificação

- Exemplos:
  - *unifica*(*f*(*A*), *g*(*B*)) = ???
  - *unifica*(*f*(*A*), *f*(*B*, *C*)) = ???
  - *unifica*(*f*(*g*(*A*)), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*f*(*A*), *g*(*B*)) = Falha porque o identificador dos termos são diferentes
  - *unifica*(*f*(*A*), *f*(*B*, *C*)) = ???
  - *unifica*(*f*(*g*(*A*)), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*f*(*A*), *g*(*B*)) = Falha porque o identificador dos termos são diferentes
  - *unifica*(*f*(*A*), *f*(*B*, *C*)) = A unificação falha porque os termos têm aridades diferentes
  - *unifica*(*f*(*g*(*A*)), *f*(*B*)) = ???

Unificação

- Exemplos:
  - *unifica*(*f*(*A*), *g*(*B*)) = Falha porque o identificador dos termos são diferentes
  - *unifica*(*f*(*A*), *f*(*B*, *C*)) = A unificação falha porque os termos têm aridades diferentes
  - *unifica*(*f*(*g*(*A*)), *f*(*B*)) = {*B*/*g*(*A*)}

Unificação

- Exemplos:
  - *unifica(f(g(A), A), f(B, xyz))* = *???*
  - *unifica(A , f(A))* = *???*

Unificação

- Exemplos:
  - $unifica(f(g(A), A), f(B, xyz)) = \{A/xyz, B/g(xyz)\}$
  - $unifica(A, f(A)) = ???$

Unificação

- Exemplos:
  - *unifica*(*f*(*g*(*A*), *A*), *f*(*B*, *xyz*)) = {*A*/*xyz*, *B*/*g*(*xyz*)}
  - *unifica*(*A* , *f*(*A*)) = Unificação infinita, A é unificado com *f*(*f*(*f*(*f*(...)))). Na Lógica de Primeira Ordem propriamente dita e em vários dialetos modernos de Prolog, isto é proibido (OCCUR CHECK).

Unificação

- Exemplos:
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*joão*,*jane*)) =

???

  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*bill*)) =

???

  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*mãe*(*Y*))) =

???

  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) =

???

Unificação

- Exemplos:
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*joão*,*jane*)) =

  {*X*/*jane*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*bill*)) =

  ???
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*mãe*(*Y*))) =

  ???
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) =

  ???

Unificação

- Exemplos:
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*joão*,*jane*)) =

  {*X*/*jane*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*bill*)) =

  {*X*/*bill*, *Y*/*joão*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*mãe*(*Y*))) =

  ???
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) =

  ???

Unificação

- Exemplos:
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*joão*,*jane*)) =

  {*X*/*jane*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*bill*)) =

  {*X*/*bill*, *Y*/*joão*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*mãe*(*Y*))) =

  {*Y*/*joão*, *X*/*mãe*(*joão*)}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) =

  ???

Unificação

- Exemplos:
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*joão*,*jane*)) =

    {*X*/*jane*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*bill*)) =

    {*X*/*bill*, *Y*/*joão*}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*mãe*(*Y*))) =

    {*Y*/*joão*, *X*/*mãe*(*joão*)}
  - *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) =

    falha.

Unificação

- *unifica*(*gosta*(*joão*,*X*),*gosta*(*X*,*elizabete*)) = *falha.*
  - Pode ser resolvida com padronização
    - Que consiste em renomear *X* em uma das sentenças, **se *X* não fizer referência ao mesmo objeto em ambas as sentenças**
  - Somente neste caso, a unificação fica:
    - *unifica*(*gosta*(*joão*,*X*),*gosta*(*Z*,*elizabete*)) = {*Z*/*joão*,*X*/*elizabete*}

Unificação

- *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*Z*))= *???*

Unificação

- *unifica*(*gosta*(*joão*,*X*),*gosta*(*Y*,*Z*))=
  - {*Y/joão, X/joão, Z/joão*} *// possível solução*
  - {*Y/joão, X/Z*} *// unificação mais geral*
- Ver algoritmo detalhado na página 269 do livro de IA do Russel e Norvig

Sistemas de produção

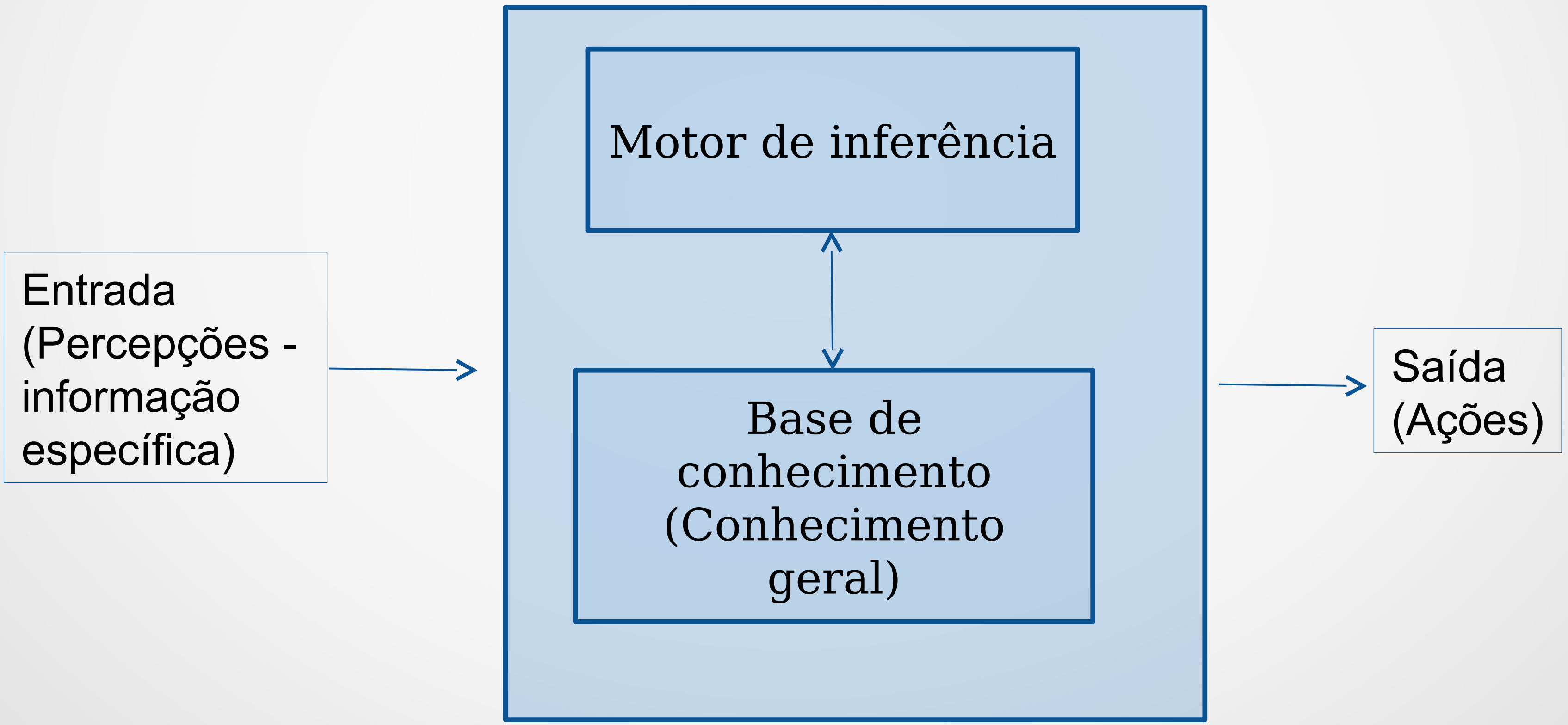
- Modelo de Computação usado em IA para busca e resolução de problemas
- Os Sistemas de Produção são úteis para o projeto de Sistemas Especialistas baseados em regras, por fornecerem um modelo para as tarefas que são centrais para esses sistemas, como:
  - Codificar a perícia humana na forma de regras
  - Projetar algoritmos de busca guiada por padrão

Sistemas de Produção

- Pode ser aplicado em diversos problemas e cenários distintos
  - Em especial, no apoio a decisão
- Um sistema de produção é definido por:
  - Conjunto de regras de produção
  - Memória de trabalho
  - Ciclo reconhecimento - ação

Componentes do SBC

- Sistema de produção pode ser usado como componente de um SBC, produzindo das ações



Composição

- Conjunto de regras de produção
- Uma **regra** (ou produção) é um par condição-ação :

  SE <condição> ENTÃO <ação>

  que define uma parcela de conhecimento
- <condição>: padrão que determina quando a regra pode ser aplicada
- <ação>: define um passo na resolução do problema

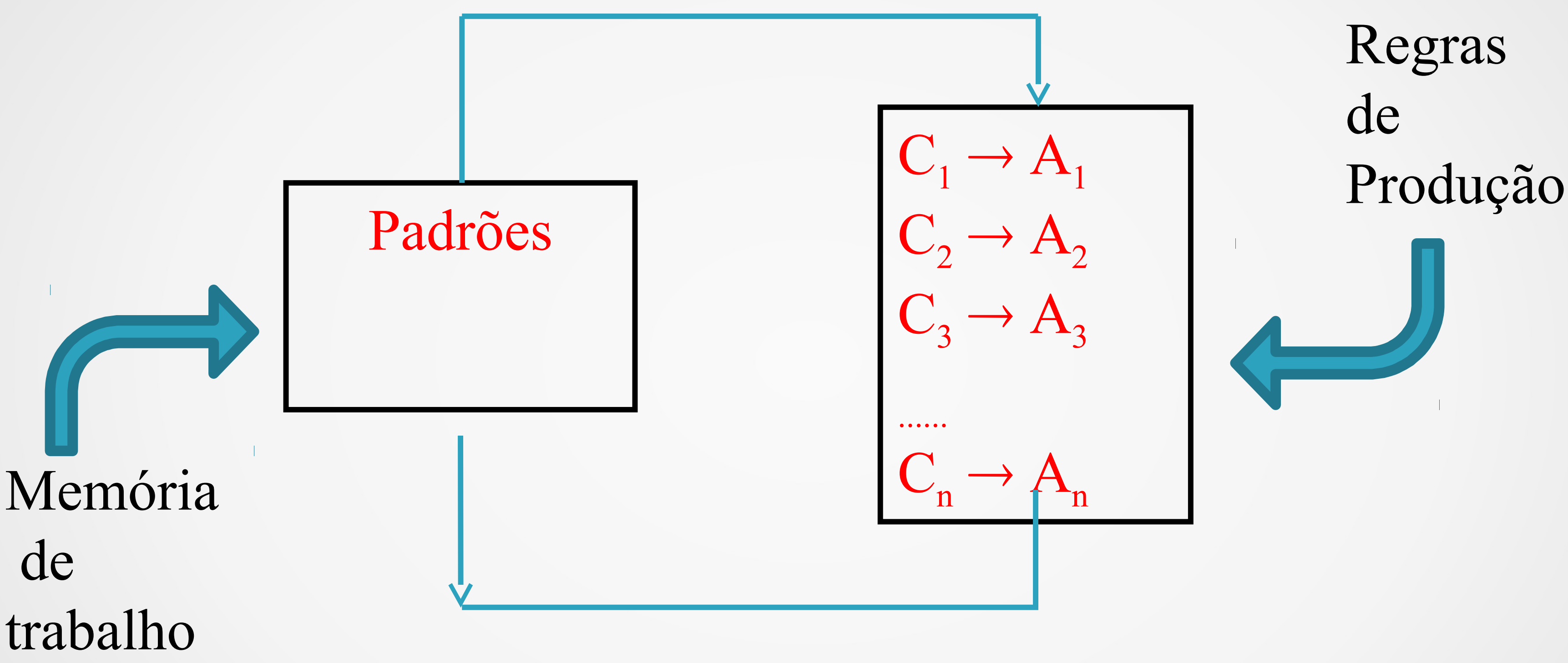
Composição

- **Memória de trabalho**
  - Conteúdo: descrição do estado corrente do problema num processo de raciocínio.
  - A descrição é formada por um ou mais padrões que são comparados com as condições das produções.
- **Finalidade**: quando a condição de uma regra casa com o conteúdo da memória de trabalho, a ação correspondente deve ser tomada.
- As ações são projetadas para alterar o conteúdo da memória de trabalho.

Composição

- O SP aplica um **ciclo reconhecimento-ação**
  - Reconhecimento da memória de trabalho
  - Ação sobre a memória de trabalho
- Memória de trabalho é inicializada com a descrição inicial do problema
- Os padrões da memória de trabalho são comparados com as condições das regras de produção
  - Que por sua vez produzem um subconjunto de regras habilitadas
  - Uma entre elas é escolhida (resolução de conflito)
- O ciclo é repetido usando a memória de trabalho modificada, até que nenhuma regra case com o conteúdo da memória de trabalho

Esquema básico de um SP



- O controle realiza ciclos até que o padrão contido na memória de trabalho não case mais com nenhuma condição das produções

Resolução de conflito

- Mais de uma regra pode ser disparada
  - Qual utilizar para produzir?
- A resolução de conflito é a seleção de uma regra para disparar, dentre as regras habilitadas
- Estratégias precisam ser definidas podendo ser:
  - Simples (primeira regra, regra menos usada, etc)
  - ou envolver heurísticas mais complexas, dependendo do problema sendo resolvido.

Exemplo

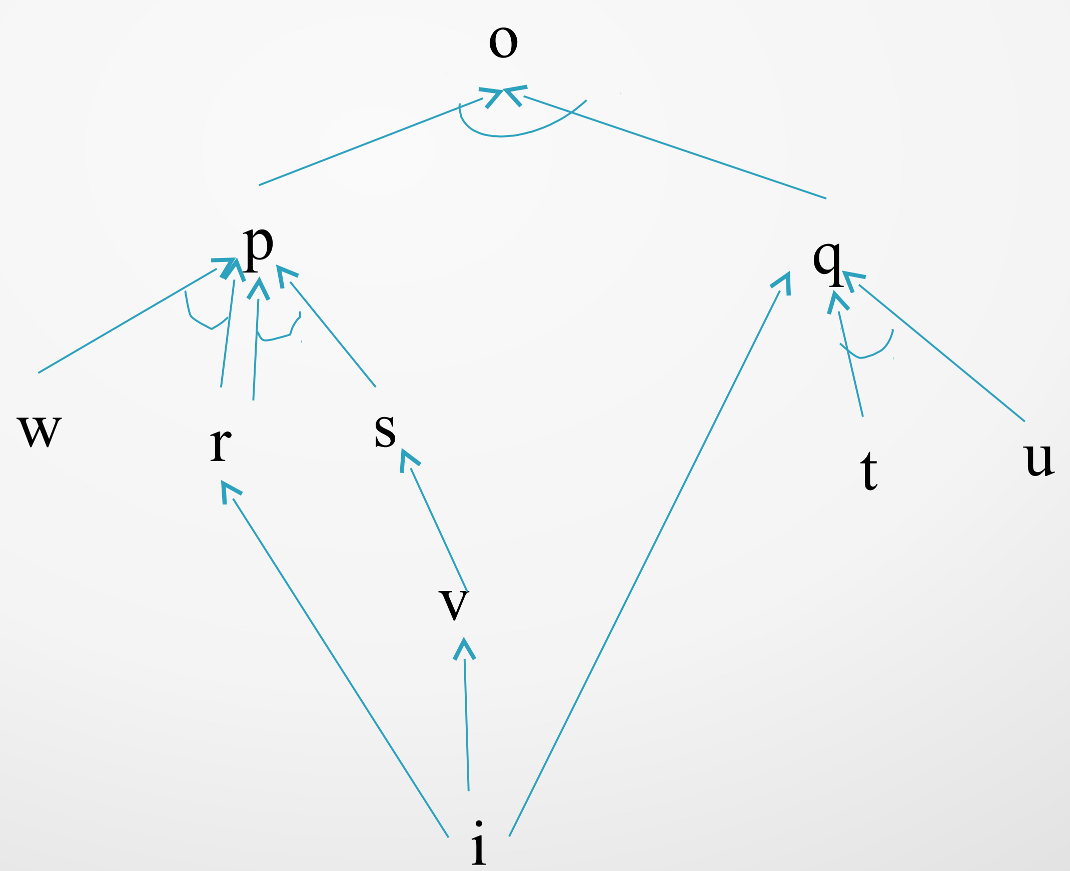
- Problema: Ordenar uma cadeia de caracteres composta de letras a, b, c
- Regras de Produção:
  1. ba $\Rightarrow$ ab
  2. ca $\Rightarrow$ ac
  3. cb $\Rightarrow$ bc
- Uma produção está habilitada se sua condição casar com uma subcadeia da cadeia que está na memória de trabalho.
- O disparo de uma regra causa a substituição da subcadeia que casou com a condição da regra pelo consequente da regra.

Exemplo

Iteração	Memória de Trabalho	Conjunto de Conflito	Regra Disparada
0	cbaca	1,2,3	1
1	cabca	2	2
2	acbca	3,2	2
3	acbac	1,3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	0	Pare

Representação do SP

- O conjunto de regras pode ser representado na forma de grafo and/or
- Essa representação é chamada de **grafo completo** do SP.



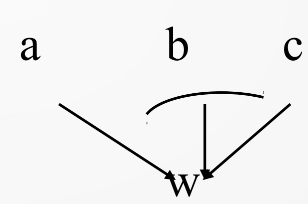
Grafo AND/OR

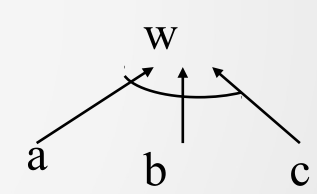
- Supor regras com apenas um consequente:

a,b,c $\rightarrow$ d,e equivale a a,b,c $\rightarrow$ d

a,b,c $\rightarrow$ e

- Dois tipos de nós:
  - AND / conjunção
  - OR / disjunção





Representação de Regras

- a,b,c $\rightarrow$ w
- a,b $\rightarrow$ x
- c,d,e $\rightarrow$ x