Factos: afirmar que uma relação entre objetos é verdadeira.

- Exemplo: father(abraham, isaac)
  - Relação ou predicado: father.
  - Objetos ou indivíduos: abraham, isaac.

Convenções sintáticas:

- Nomes de predicados e objetos começam com minúscula (átomos).
- Frases terminam com ponto final.

Um conjunto finito de factos é um programa em lógica.

Variável Lógica: representa um indivíduo não especificado.

- Não é uma posição de memória onde se coloca um valor!
- Convenção: começa por maiúscula.

Termos: constantes, variáveis, estruturas (termos complexos).

Estruturas: functor. Exemplos: s(0), hot(milk), name(john, doe)...

- nome(um átomo)
- aridade(número de argumentos): s/1, hot/1, name/2

Termos:

- sem variáveis: ground (totalmente instanciado)
- com variáveis: nonground

Conjunção de objetivos: a vírgula corresponde ao "e"lógico ($\wedge$).

Procedimento: coleção de regras com o mesmo predicado na cabeça.

Um programa em lógica é um conjunto finito de regras. O significado de um programa em lógica P, M(P), é o conjunto de objetivos totalmente instanciados dedutíveis de P (se o programa apenas tem factos, o seu significado é o próprio programa).

Uma base de dados em lógica contém:

- Factos: permitem definir relações, como em bases de dados relacionais.
- Regras: permitem definir perguntas relacionais complexas (vistas).

Programa recursivo linear: o corpo da regra recursiva tem apenas um objetivo recursivo.

```
ancestor(Ancestor, Descendant) :-
    parent(Ancestor, Descendant).
ancestor(Ancestor, Descendant) :-
    parent(Ancestor, Person), ancestor(Person, Descendant).
```

Uma lista é uma estrutura de dados binária: .(X, Y)

- 1º argumento (cabeça): elemento; 2º argumento (cauda): resto da lista.
- Símbolo constante para fim da recursão: lista vazia (nil ou []).
- Sintaxe alternativa: $.(X, Y) \equiv [X|Y]$
- Os seus elementos podem ser quaisquer termos.
- Predicados importantes:
  - Membro de uma lista: member(X, L)
    - Verifica se um elemento X é membro da lista L.
    - Obtém um elemento da lista L.
    - Obtém uma lista que contém um elemento X.
  - Prefixo: prefix(L1, L)
  - Sufixo: suffix(L1, L)
  - Sublista: sublist(L1, L)
  - Concatenação de listas: append(L1, L2, L)
  - Inversão de uma lista: reverse(L1, L)
  - Comprimento de uma lista: length(L, Length)
  - Eliminar elementos: delete(L, X, NL)
  - Selecionar elemento: select(X, L1, L)
  - Permutações: permutation(L1, L)

Frase consistente: tem uma instância verdadeira.

Frase inválida: tem uma instância falsa.

Um unificador de dois termos é uma substituição que os torna idênticos, portanto, um unificador encontra uma instância comum.

Condições de unificação:

- Variável com Variável: unificam sempre.
- Atómico com Atómico: unificam se os valores forem iguais.
- Atómico ou Estrutura com Variável: unificam sempre.
- Estrutura com Estrutura: unificam se os functores são iguais, o número de argumentos é igual e os argumentos unificam dois a dois.

Regras de Dedução:

- $1^{\underline{a}}$ - Identidade: de P deduzir P?
  - Uma pergunta é uma consequência lógica de um facto idêntico.
- $2^{\underline{a}}$ - Generalização: de Pθ deduzir P?
  - Uma pergunta existencial é uma consequência lógica de uma instância sua.
- $3^{\underline{a}}$ - Instanciação: de P deduzir Pθ
  - De um facto quantificado universalmente, deduz-se uma instância sua.
- $4^{\underline{a}}$ - Modus Ponens Universal: de $A \leftarrow B_1, ..., B_n$ e de $B_1\theta, ..., B_n\theta$ deduzir $A\theta$
  - Da instanciação do corpo de uma regra podemos deduzir a instanciação da cabeça.