

UNIVERSIDADE ESTADUAL VALE DO ACARAÚ - UVA  
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET  
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO I  
PROFESSOR: HUDSON COSTA  
CURSO: CIÊNCIAS DA COMPUTAÇÃO



# Programação Lógica

25 de maio

2011

**Equipe:**

Yuri Nascimento

Joyce Monte de Oliveira

Carlos Ederson

Paulo Mauricio

Sobral, maio de 2011.

## Súmarío

1	Introdução .....	1
2	O que é Programação Lógica? .....	<b>Erro! Indicador não definido.</b> ,5
3	Paradigmas do PROLOG .....	5,6
4	Vantagens e Desvantagens .....	6
5	Diferença entre programas convencionas e programas em lógica .....	6,7
6	Aplicações da linguagem lógica .....	8,9, <b>Erro! Indicador não definido.</b>
7	Prolog.....	<b>Erro! Indicador não definido.</b> 0,11
8	Conclusão .....	12
8	Bibliografia .....	13

## Introdução

Iremos iniciar o nosso trabalho falando inicialmente o que é a Programação Lógica logo após falarmos sobre os paradigmas e sobre a linguagem Prolog que é servida por técnicas declarativas de representação + procura tipo primeiro em profundidade + raciocínio para trás. O mecanismo de resolução de problemas de programação em lógica, como o PROLOG é um demonstrador de teoremas que manipula estruturas simbólicas, conduzido por regras de inferência. Esta linguagem alia a lógica e o controle no algoritmo, ou seja não é puramente lógica, mas contém teorias de procura que se enquadram no contexto das evoluções em Inteligência Artificial, o aperfeiçoamento dos agentes inteligentes, apesar de não ter sido a primeira é muito utilizada nesta linguagem. Através deste trabalho poderemos desfrutar do conhecimento em programação lógica.

## O que é Programação Lógica?

Programação lógica é um paradigma de programação que faz uso da lógica matemática. John McCarthy [1958] foi o primeiro a publicar uma proposta de uso da lógica matemática para programação.

A primeira linguagem de programação lógica foi a Planner, a qual permitia a invocação orientada a padrões de planos procedimentais de asserções e de objetivos. Com a necessidade de adaptação aos sistemas de memória muito limitada, que eram disponíveis quando ela foi desenvolvida. A linguagem Planner usava estruturas de controle de *backtracking*, de tal forma que apenas um único caminho computacional tinha que ser armazenado por vez. Em seguida, o Prolog foi desenvolvido como uma simplificação do Planner que permitia a invocação orientada a padrões apenas a partir de objetivos (também baseado em *backtracking*).

A partir do Planner, foram desenvolvidas as linguagens de programação QA-4, Popler, Conniver, e QLISP. As linguagens de programação Mercury, Visual Prolog, Oz e Frill, foram desenvolvidas a partir do Prolog. Atualmente existem linguagens de programação lógica concorrente (não baseadas em *backtracking*) derivadas do Planner (por exemplo, a Ether) e derivadas do Prolog (ver Shapiro 1989 para um apanhado geral).

Uma brevia sobre a história da programação lógica é uma idéia que tem sido investigada no contexto da inteligência artificial pelo menos desde o momento em que John McCarthy propôs: "programas para manipular com sentenças instrumentais comuns apropriadas à linguagem formal (muito provavelmente uma parte do cálculo de predicado)". O programa básico formará conclusões imediatas a partir de uma lista de premissas. Essas conclusões serão tanto sentenças declarativas quanto imperativas. Quando uma sentença imperativa é deduzida, o programa toma uma ação correspondente.

A lógica de programação é utilizada quando se pretende realizar alguma função ou um esquema lógico por meio de parâmetros e metas.

Essa linguagem possui base na Lógica Matemática e qual o sentido de pegar esta base para a programação lógica é trazer o estilo da lógica matemática à programação de computadores. Matemáticos e filósofos encontram na lógica uma ferramenta eficaz para desenvolvimento de teorias. Vários problemas são naturalmente expressos como teorias. Dizer que um problema precisa de solução frequentemente equivale a perguntar se uma nova hipótese é consistente com

uma teoria existente ou se é conseqüência dela. A lógica proporciona uma maneira de demonstrar se uma questão é verdadeira ou falsa.

O processo de construir uma demonstração é bem conhecido, portanto a lógica é um meio confiável de responder perguntas. Sistemas de programação lógica automatizam este processo. A inteligência artificial teve uma influência importante no desenvolvimento da programação lógica.

## **Paradigmas da programação**

As linguagens de programação tem um papel fundamental no funcionamento dos computadores, existem diferentes tipos de linguagens cada uma seguindo seus padrões de sintática e semântica com suas funcionalidades e características próprias. Assim como os seres humanos os computadores também aprendem lições, basta alguém ensina-los, e ai é preciso o uso da linguagem que vai ser interpretada pela maquina, para depois ser armazenada como um conjunto de instruções.

Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa. Eles são diferenciados pela proibição ou pela permissão de técnicas.

## **Paradigmas lógicos**

Tem sua base logica num subconjunto da logica, que são chamadas de clausulas de Horn, basicamente as linguagens do paradigma logico tem seguinte princípio básico. As expressões ou regras, pois que para que algo retorne verdadeiro, e necessário que todas as regras sejam verdadeiras, Como o paradigma funcional, a execução é principalmente recursiva.

O Paradigma Lógico enfatiza a descrição declarativa de um problema, ao invés da decomposição do problema em uma implementação algorítmica. Tais programas são mais próximos de uma especificação do que a tradicional forma de programar.

A linguagem do paradigma que foi mais bem sucedida, desde sua criação, foi o Prolog, que é largamente utilizado em IA e data-minning.

Uma das principais ideias da programação em lógica é de que um algoritmo é constituído por dois elementos disjuntos: a lógica e o controle. O componente lógico corresponde à definição do que deve ser solucionado, enquanto que o componente de controle estabelece como a solução pode ser obtida.

O programador precisa somente descrever o componente lógico de um algoritmo, deixando o controle da execução para ser exercido pelo sistema de programação em lógica utilizado. Em outras palavras, a tarefa do programador passa a ser simplesmente a *especificação* do problema que deve ser solucionado, razão pela qual as linguagens lógicas podem ser vistas simultaneamente como linguagens para especificação formal e linguagens para a programação de computadores. Um *programa em lógica* é então a representação de determinado problema ou

situação expressa através de um conjunto finito de um tipo especial de sentenças lógicas denominadas *cláusulas*. Ao contrário de programas em Pascal ou C, um programa em lógica *não é* a descrição de um procedimento para se obter a solução de um problema. Na realidade o sistema utilizado no processamento de programas em lógica é inteiramente responsável pelo procedimento a ser adotado na sua execução. Um programa em lógica pode também ser visto alternativamente como uma base de dados, exceto que as bases de dados convencionais descrevem apenas  *fatos*  tais como "Oscar é um avestruz", enquanto que as sentenças de um programa em lógica possuem um alcance mais genérico, permitindo a representação de  *regras*  como em "Todo avestruz é um pássaro", o que não possui correspondência em bases de dados convencionais. Na figura abaixo se procura explicitar as principais diferenças entre programação convencional e programação em lógica.

Vantagens:

- Existem mecanismos de inferência eficientes que deduzem o valor verdade de uma fórmula em tempo (e espaço) linear em função do número de constantes da fórmula
- Existem também mecanismos de inferência eficientes para abdução (diagnóstico) e indução (aprendizagem).

Desvantagens:

Em domínios complexos, explosão combinatória da base de conhecimento não permite:

- Representar relações genéricas entre conceitos representados em intenção
- Estruturação modular do conhecimento
- Representação de conhecimento em ambientes dinâmicos ou não deterministas ou contínuos.

### **Diferenças entre programas convencionais e programas em lógica**

Uma das principais idéias da programação em lógica é de que um algoritmo é constituído por dois elementos disjuntos: a lógica e o controle. O componente lógico corresponde à definição do que deve ser solucionado, enquanto que o componente de controle estabelece como a solução pode ser obtida. O programador precisa somente descrever o componente lógico de um algoritmo, deixando o controle da execução para ser exercido pelo sistema de programação em lógica utilizado. Em outras palavras, a tarefa do programador passa a ser simplesmente a  *especificação*  do problema que deve ser solucionado, razão pela qual as linguagens lógicas podem ser vistas simultaneamente como linguagens para especificação formal e linguagens para a programação de computadores.

Um *programa em lógica* é então a representação de determinado problema ou situação expressa através de um conjunto finito de um tipo especial de sentenças lógicas denominadas *cláusulas*. Ao contrário de programas em Pascal ou C, um programa em lógica *não é* a descrição de um procedimento para se obter a solução de um problema. Na realidade o sistema utilizado no processamento de programas em lógica é inteiramente responsável pelo procedimento a ser adotado na sua execução. Um programa em lógica pode também ser visto alternativamente como uma base de dados, exceto que as bases de dados convencionais descrevem apenas  *fatos*  tais como "Oscar é um avestruz", enquanto que as sentenças de um programa em lógica possuem um alcance mais genérico, permitindo a representação de  *regras*  como em "Todo avestruz é um pássaro", o que não possui correspondência em bases de dados convencionais. Na figura abaixo se procura explicitar as principais diferenças entre programação convencional e programação em lógica.

### Programas Convencionais x Programas em Lógica

<b>PROGRAMAS CONVENCIONAIS</b>	<b>PROGRAMAS EM LÓGICA</b>
Processamento Numérico	Processamento Simbólico
Soluções Algorítmicas	Soluções Heurísticas
Estruturas de Controle e Conhecimento Integradas	Estruturas de Controle e Conhecimento Separadas
Difícil Modificação	Fácil Modificação
Somente Respostas Totalmente Corretas	Incluem Respostas Parcialmente Corretas
Somente a Melhor Solução Possível	Incluem Todas as Soluções Possíveis

## Aplicações

Um dos primeiros usos da programação em lógica foi a representação e análise de subconjuntos da linguagem natural. Esta foi inclusive a aplicação que motivou Alain Colmerauer a desenvolver a primeira implementação da linguagem Prolog. Logo em seguida, outros pesquisadores da área da inteligência artificial propuseram diversas novas aplicações para o novo instrumento. Alguns dos primeiros trabalhos com Prolog envolviam a formulação de planos e a escrita de compiladores, por Pereira e Warren (1977) prova de teoremas em geometria por R. Welhan (1976) e a solução de problemas de mecânica, por Bundy *et al.* (1979). As aplicações relatadas desde então, multiplicaram-se velozmente. Concentraremos aqui a atenção em um conjunto das principais áreas investigadas com o concurso da programação em lógica.

- **□ Sistemas Baseados em Conhecimento (SBCs)**

Ou *knowledge-based systems*, são sistemas que aplicam mecanismos automatizados de raciocínio para a representação e inferência de conhecimento.

Tais sistemas costumam ser identificados como simplesmente "de inteligência artificial aplicada" e representam uma abrangente classe de aplicações da qual todas as demais seriam aproximadamente subclasses. A tecnologia dos SBCs foi identificada na Inglaterra pelo Relatório Alvey (1982) como uma das quatro tecnologias necessárias à completa exploração dos computadores de quinta geração. As outras seriam: interface homem-máquina (MMI), integração

de circuitos em ultra-grande escala (ULSI) e engenharia de software (SE). O relacionamento entre SBCs e a nova geração de computadores é, na verdade, altamente simbiótica, cada uma dessas áreas é necessária para a realização do completo potencial da outra.

- **Sistemas de Bases de Dados (BDs)**

Uma particularmente bem definida aplicação dos SBCs são bases de dados. BDs convencionais tradicionalmente manipulam dados como coleções de relações armazenadas de modo extensional sob a forma de tabelas. O modelo relacional serviu de base à implementação de diversos sistemas fundamentados na álgebra relacional, que oferece operadores tais como junção e projeção. O processador de consultas de uma BD convencional deriva, a partir de uma consulta fornecida como entrada, alguma conjunção específica de tais operações algébricas que um programa gerenciador então aplica às tabelas visando a recuperação de conjuntos de dados (n-tuplas) apropriados, se existirem. O potencial da programação em lógica para a representação e consulta à BDs foi simultaneamente investigado, em 1978, por Van Emden, Kowalski e Tärnlund. As três pesquisas estabeleceram que a recuperação de dados um problema básico em BDs convencionais - é intrínseco ao mecanismo de inferência dos interpretadores lógicos. Desde então diversos sistemas tem sido propostos para a representação de BDs por meio de programas em lógica.

- **Sistemas Especialistas (SEs):** Um sistema especialista é uma forma de SBC especialmente projetado para emular a especialização humana em algum domínio específico. Tipicamente um SE irá possuir uma *base de conhecimento* (BC) formada de fatos, regras e *heurísticas* sobre o domínio, juntamente com a capacidade de entabular comunicação interativa com seus usuários, de modo muito próximo ao que um especialista humano faria. Além disso, os SEs devem ser capazes de oferecer sugestões e conselhos aos usuários e, também, melhorar o próprio desempenho a partir da experiência, isto é, adquirir novos conhecimentos e heurísticas com essa interação. Diversos sistemas especialistas foram construídos com base na programação em lógica, como por exemplo, o sistema ORBI, para a análise de recursos ambientais desenvolvido por Pereira *et al.* na Universidade Nova de Lisboa.
- **Processamento da Linguagem Natural (PLN):** O PLN é da maior importância para o desenvolvimento de ferramentas para a comunicação homem-máquina em geral e para a construção de interfaces de SBCs em particular. A implementação de sistemas de PLN em computadores requer não somente a formalização sintática, como também - o grande problema - a formalização semântica, isto é, o correto *significado* das palavras, sentenças, frases, expressões, etc. que povoam a comunicação natural humana. O uso da lógica das cláusulas de Horn<sup>3</sup> para este propósito foi inicialmente investigado por Colmerauer, o próprio criador do Prolog (1973), e posteriormente por Kowalski (1974). Ambos mostraram que as cláusulas de Horn eram adequadas à representação de qualquer *gramática livre-de-contexto* (GLC), permitiam que questões sobre a estrutura de sentenças em linguagem natural fossem formuladas como objetivos ao sistema, e que diferentes procedimentos de prova aplicados a representações lógicas da linguagem natural correspondia a diferentes estratégias de análise.
- **Educação:** A programação em lógica poderá vir a oferecer no futuro uma contribuição bastante significativa ao uso educacional de computadores. Esta proposta foi testada em 1978 quando Kowalski introduziu a programação em lógica na Park House Middle School em Wimbledon, na Inglaterra, usando acesso on-line aos computadores do Imperial College. O sucesso do empreendimento conduziu a um projeto mais abrangente denominado "Lógica como Linguagem de Programação para Crianças", inaugurado em 1980 na Inglaterra com recursos do Conselho de Pesquisa Científica daquele país. Os resultados obtidos desde então tem mostrado que a programação em lógica não somente é assimilada mais facilmente do que as linguagens convencionais, como também pode ser introduzida até mesmo a crianças na faixa dos 10 a 12 anos, as quais ainda se beneficiam do desenvolvimento do pensamento lógico-formal que o uso de linguagens como o Prolog induz.
- **Arquiteturas Não-Convencionais:** Esta área vem se tornando cada vez mais um campo extremamente fértil para o uso da programação em lógica especialmente na especificação e implementação de máquinas abstratas de processamento paralelo. O paralelismo pode ser modelado pela programação em lógica em variados graus de atividade se implementado em conjunto com o mecanismo de unificação. Duas implementações iniciais

nesse sentido foram o *Parlog*, desenvolvido em 1984 por Clark e Gregory, e o *Concurrent Prolog* (CP), por Shapiro em 1983. O projeto da Quinta Geração, introduzido na próxima seção, foi fortemente orientado ao uso da programação em lógica em sistemas de processamento paralelo.

Muitas outras aplicações poderiam ainda ser citadas, principalmente na área da inteligência artificial, que tem no Prolog e no Lisp as suas duas linguagens mais importantes. Novas tecnologias de hardware e software tais como sistemas massivamente paralelos, redes de computadores, assistentes inteligentes, bases de dados semânticas, etc. Tornando assim o uso do Prolog (e de outras linguagens baseadas em lógica) cada vez mais atraentes.

## Prolog

Os termos "programação em lógica" e "programação Prolog" tendem a ser empregados indistintamente. Deve-se, entretanto, destacar que a linguagem Prolog é apenas uma particular abordagem da programação em lógica. As características mais marcantes dos sistemas de programação em lógica em geral e da linguagem Prolog em particular - são as seguintes:

- **Especificações são Programas:** A linguagem de especificação é entendida pela máquina, e é por si só, uma linguagem de programação. Naturalmente, o refinamento de especificações é mais efetivo do que o refinamento de programas. Um número ilimitado de **cláusulas** diferentes pode ser usado e predicados (procedimentos) com qualquer número de argumentos são possíveis. Não há distinção entre o programa e os dados. As cláusulas podem ser usadas com grande vantagem sobre as construções convencionais para a representação de tipos abstratos de dados. A adequação da lógica para a representação simultânea de programas e suas especificações a torna um instrumento especialmente útil para o desenvolvimento de ambientes e protótipos.
- **Capacidade Dedutiva:** O conceito de computação confunde-se com o de (passo de) inferência. A execução de um programa é a prova do teorema representado pela consulta formulada, com base nos **axiomas** representados pelas cláusulas (fatos e regras) do programa.
- **Não-determinismo:** Os procedimentos podem apresentar múltiplas respostas, da mesma forma que podem solucionar múltiplas e aleatoriamente variáveis condições de entrada. Através de um mecanismo especial, denominado "backtracking", uma seqüência de resultados alternativos pode ser obtida.
- **Reversibilidade das Relações:** (Ou "computação bidirecional"). Os argumentos de um procedimento podem alternativamente, em diferentes chamadas representar ora parâmetros de entrada, ora de saída. Os procedimentos podem assim ser projetados para atender a múltiplos propósitos. A execução pode ocorrer em qualquer sentido, dependendo do contexto. Por exemplo, o mesmo procedimento para inserir um elemento no topo de uma pilha qualquer pode ser usado, em sentido contrário, para remover o elemento que se encontrar no topo desta pilha.

- **Tríplice Interpretação dos Programas em Lógica:** Um programa em lógica pode ser semanticamente interpretado de três modos distintos: (1) por meio da semântica declarativa, inerente à lógica, (2) por meio da semântica procedimental, onde as cláusulas dos programas são vistas como entrada para um método de prova e, (3) por meio da semântica operacional, onde as cláusulas são vistas como comandos para um procedimento particular de prova por refutação. Essas três interpretações são intercambiáveis segundo a particular abordagem que se mostrar mais vantajosa ao problema que se tenta solucionar.
- **Recursão:** A recursão, em Prolog, é a forma natural de ver e representar dados e programas. Entretanto, na sintaxe da linguagem não há laços do tipo "for" ou "while" (apesar de poderem ser facilmente programados), simplesmente porque eles são absolutamente desnecessários. Também são dispensados comandos de atribuição e, evidentemente, o "goto". Uma estrutura de dados contendo variáveis livres pode ser retornada como a saída de um procedimento. Essas variáveis livres podem ser posteriormente instanciadas por outros procedimentos produzindo o efeito de atribuições implícitas a estruturas de dados. Onde forem necessários, variáveis livres são automaticamente agrupadas por meio de referências transparentes ao programador. Assim, as variáveis lógicas um potencial de representação significativamente maior do que oferecido por operações de atribuição e referência nas linguagens convencionais.

## **Conclusão**

Atraves deste trabalho pudemos perceber o quanto é vasto o mundo da programação pois a programação lógica e suas extensões que tivemos conhecimento aparti deste trabalho nos fizeram perceber que a programação pode interligar varios conhecimentos como a lógica matemática que é base da programação em questão.Sendo de grande valia para a implementação dos nossos conhecimentos em programação pois quem sabe um dia seremos capazes de criarmos uma linguagem.

## Bibliografia

- <http://pt.wikipedia.org/wiki/Prolog>
- [http://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o\\_l%C3%B3gica](http://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica)
- [http://www.mpsoftwares.com/site1/index.php?option=com\\_kunena&Itemid=1&func=view&catid=4&id=5](http://www.mpsoftwares.com/site1/index.php?option=com_kunena&Itemid=1&func=view&catid=4&id=5)
- <http://www.cassao.eti.br/portal/programacao-logica>
- Livro - INTRODUÇÃO À PROGRAMAÇÃO PROLOG –  
**Autora:**Luiz A. M. Palazzo