Aplicações de Programação em Lógica

Ademilson Santos Lima

Departamento de Ciências da Computação

Universidade Federal da Bahia (UFBA) – Salvador, BA – Brasil

adesanlim21@gmail.com

Resumo. *Este artigo descreve a Programação em Lógica desde as suas raízes até aplicações em que se pode encontrá-la nos dias atuais. É feito um estudo sobre alguns resultados relevantes que proporcionaram a união da lógica matemática com a computação. Também é realizado um estudo sobre umas das mais utilizadas linguagens deste paradigma: Prolog. Por fim, são exibidas cinco aplicações reais que fazem uso da Programação em Lógica.*

1. Introdução

O emprego das linguagens para Programação em Lógica ganhou significativo impulso com o projeto Japonês de Sistemas Computacionais de Quinta Geração (1982-1992), o qual investigou alternativas de hardware e software para atender o desenvolvimento de aplicações que contemplavam metas ambiciosas, tais como reconhecimento de imagens, processamento da linguagem natural, processamento de conhecimento etc.

As linguagens para Programação em Lógica, a exemplo do Prolog, anteriormente empregadas principalmente na prototipação, já podem ser utilizadas para resolver, com bom desempenho, complexos problemas reais de IA. Isto se tornou possível pela disponibilidade de processadores poderosos a custos reduzidos, bem como pela disseminação do uso de arquiteturas paralelas.

No decorrer deste texto são ressaltados os porquês de se estudar a Programação em Lógica, exibindo como motivação seu histórico, os principais resultados que renomados matemáticos do século passado concluíram, um breve comentário sobre a linguagem Prolog e finalizando com aplicações reais baseadas neste paradigma.

2. Histórico

O uso da lógica na representação dos processos de raciocínio remonta aos estudos de Boole (1815-1864) e de De Morgan (1806-1871), sobre o que veio a ser mais tarde chamado "Álgebra de Boole". Como o próprio nome indica, esses trabalhos estavam mais próximos de outras teorias matemáticas do que propriamente da lógica. Deve-se ao matemático alemão Friedrich Ludwig Göttlob Frege, no seu "Begriffsschrift" (1879 - *Ideografia* [*Ideography*] é uma tradução sugerida em carta pelo próprio autor, outra opção seria *Notação Conceptual*), a primeira versão do que hoje denominamos cálculo de predicados, proposto por ele como uma ferramenta para formalizar princípios lógicos. Esse sistema oferecia uma notação rica e consistente que Frege pretendia adequada para a representação

de todos os conceitos matemáticos e para a formalização exata do raciocínio dedutivo sobre tais conceitos, o que, afinal, acabou acontecendo.

No final do século XX a matemática havia atingido um estágio de desenvolvimento mais do que propício à exploração do novo instrumento proposto por Frege. Os matemáticos estavam abertos a novas áreas de pesquisa que demandavam profundo entendimento lógico assim como procedimentos sistemáticos de prova de teoremas mais poderosos e eficientes do que os até então empregados. Seguem nos próximos parágrafos algumas das contribuições mais relevantes.

A **reconstrução axiomática da geometria abstrata** por **David Hilbert**. Transformação da geometria euclidiana em axiomas, com uma visão mais formal que Euclides, para torná-la consistente, publicada no seu *Grundlagen der Geometrie* (*Bases de geometria*).

A **aritmética proposta** por **Giuseppe Peano**. Foi o fundador da moderna lógica matemática e teoria dos conjuntos, para cujos conceitos e notações contribuíram de forma decisiva. Na obra "Arithmetices Principia Nova Methodo Exposita" de 1889, Peano desenvolveu os famosos axiomas de Peano, considerados até hoje como a axiomatização padrão dos números naturais. Em resumo, a construção dos naturais proposta é de que existe um número 0 (zero) que não é sucessor de qualquer número. Existe uma função *sucessor* (suc) que pode ser invocada recursivamente para descrever qualquer outro número da maneira a seguir:

suc(0) = 1, suc(1) = suc(suc(0)) = 2, suc(suc(suc(0))) = 3 e assim sucessivamente.

A exploração intuitiva da **teoria geral dos conjuntos**, por **Georg Cantor**. Conhecido por ter elaborado a moderna teoria dos conjuntos. Foi a partir desta teoria que chegou ao conceito de número transfinito (é a forma rigorosa usada pela matemática para contar o número de elementos de conjuntos infinitos), incluindo as classes numéricas dos cardinais e ordinais, estabelecendo a diferença entre estes dois conceitos, que colocam novos problemas quando se referem a conjuntos infinitos.

O **relacionamento entre lógica e matemática** foi profundamente investigado por Alfred North Whitehead e Bertrand Russel, que em "Principia Mathematica" (1910) demonstraram ser a lógica um instrumento adequado para a representação formal de grande parte da matemática.

Um passo muito importante foi dado em 1930, em estudos simultâneos, porém independentes, realizados pelo alemão Kurt Gödel e o francês Jacques Herbrand. Ambos, em suas dissertações de doutorado, demonstraram que o mecanismo de prova do cálculo de predicados poderia oferecer uma prova formal de toda proposição logicamente verdadeira. O resultado de maior impacto foi, entretanto, produzido por Gödel, em 1931, com a descoberta do "**teorema da incompletude dos sistemas de formalização da aritmética**". O segundo este teorema Gödel mostra que a aritmética básica não pode ser usada para provar sua própria consistência, portanto a proposta de Hilbert de reduzir a Matemática a um conjunto finito de axiomas completo e consistente não podia ser possível. A prova deste teorema se baseava nos denominados paradoxos de auto-referência (declarações do tipo: "Esta sentença é falsa", que não podem ser provada nem verdadeira nem falsa).

Em 1934, Alfred Tarski produziu a primeira **teoria semântica** rigorosamente formal do **cálculo de predicados**, introduzindo conceitos precisos para "satisfatibilidade", "verdade" (em uma dada interpretação), "conseqüência lógica" e outras noções relacionadas. Ainda na década de 30, diversos outros estudos - entre os quais os de Alan Turing, Alonzo Church e outros – aproximaram muito o cálculo de predicados da forma com que é hoje conhecido e estudado.

No início da Segunda Guerra Mundial, em 1939, toda a fundamentação teórica básica da lógica computacional estava pronta. Faltava apenas um meio prático para realizar o imenso volume de computações necessárias aos procedimentos de prova. Apenas exemplos muito simples podiam ser resolvidos manualmente. O estado de guerra deslocou a maior parte dos recursos destinados à pesquisa teórica, nos EUA, Europa e Japão para as técnicas de assassinato em massa. Foi somente a partir da metade dos anos 50 que o desenvolvimento da então novíssima tecnologia dos computadores conseguiu oferecer aos pesquisadores o potencial computacional necessário para a realização de experiências mais significativas com o cálculo de predicados.

A programação em lógica em sistemas computacionais, entretanto, somente se tornou realmente possível a partir da pesquisa sobre prova automática de teoremas, particularmente no desenvolvimento do *Princípio da Resolução* por J. A. Robinson (1965). O princípio da resolução é uma regra de inferência que dá origem a uma técnica de demonstração por refutação para sentenças e inferências da lógica proposicional e da lógica de primeira ordem.

Um dos primeiros trabalhos relacionando este princípio com a programação de computadores deve-se a Cordell C. Green (1969) que mostrou como o mecanismo para a extração de respostas em sistemas de resolução poderia ser empregado para sintetizar programas convencionais.

A expressão "programação em lógica" (logic programming, originalmente em inglês) é devido a Robert Kowalski (1974) e designa o uso da lógica como linguagem de programação de computadores. Kowalski identificou, em um particular procedimento de prova de teoremas, um procedimento computacional, permitindo uma interpretação procedimental da lógica e estabelecendo as condições que nos permitem entendê-la como uma linguagem de programação de uso geral. Este foi um avanço essencial, necessário para adaptar os conceitos relacionados com a prova de teoremas às técnicas computacionais já dominadas pelos programadores. Aperfeiçoamentos realizados nas técnicas de implementação também foram de grande importância para o emprego da lógica como linguagem de programação. O primeiro interpretador experimental foi desenvolvido por um grupo de pesquisadores liderados por Alain Colmerauer (cientista da computação Francês) na Universidade de Aix-Marseille (1972) com o nome de Prolog, um acrônimo para *Programmation en Logique*.

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. Esta linguagem usava estruturas de controle de *backtracking*, de tal forma que apenas um único caminho computacional tinha que ser armazenado por vez. Em seguida, Prolog foi

desenvolvido como uma simplificação da Planner que permitia a invocação orientada a padrões apenas a partir de objetivos (também baseado em *backtracking*).

3. O algoritmo da programação 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 "Willy é uma baleia", enquanto que as sentenças de um programa em lógica possuem um alcance mais genérico, permitindo a representação de regras como em "Toda baleia é um mamífero", o que não possui correspondência em bases de dados convencionais.

3.1. Cláusulas de Horn

Em lógica, uma cláusula de Horn é uma cláusula (disjunção de literais) com no máximo um literal positivo. Uma cláusula de Horn com exatamente um literal positivo é dita uma cláusula definida (ou regra), já uma cláusula de Horn sem literais positivos é às vezes dita cláusula objetivo (ou fato), especialmente no contexto da programação lógica. Já uma fórmula de Horn é uma fórmula na forma normal conjuntiva cujas cláusulas são todas de Horn; em outras palavras, é uma conjunção de cláusulas de Horn.

Ex: ~p | ~q | ... | ~t | u

Em programação lógica, lê-se como: Para mostrar u, mostre p e mostre q e ... .e mostre t.

É fácil ver que esta fórmula pode ser reescrita como (p & q & ... & t) → u, apenas usando de equivalências entre a implicação e a disjunção. É mostrado na lógica matemática que A→ B é equivalente a ~A | B.

A cláusula de Horn mostrada é literalmente escrita em Prolog como u :- p, q, ..., t. Onde:

- “u” é o que queremos mostrar.
- ”p”, “q”, ..., “t” é o que temos que mostrar para mostrar “u”.

Na prática, “u” é verdadeiro se todos “p”, “q”, ... “t” são verdadeiros.

4. Programação em Lógica

Este novo paradigma de programação só ganhou o devido reconhecimento a partir dos anos 80, quando foi escolhida como base do projeto japonês para o desenvolvimento dos denominados computadores de quinta geração. O ponto focal da programação em lógica consiste em identificar a noção de *computação* com a noção de *dedução*. Mais precisamente, os sistemas de programação em lógica reduzem a execução de programas à pesquisa da refutação das sentenças do programa em conjunto com a negação da sentença que expressa a consulta, seguindo a regra: *"uma refutação é a dedução de uma contradição"*.

Pode-se então expressar conhecimento (programas e/ou dados) em Prolog por meio de cláusulas de dois tipos: fatos e regras. Um fato denota uma verdade incondicional, enquanto que as regras definem as condições que devem ser satisfeitas para que certa declaração seja considerada verdadeira. Como fatos e regras podem ser utilizados conjuntamente, nenhum componente dedutivo adicional precisa ser utilizado. Além disso, como regras recursivas e não-determinismo são permitidos, os programadores podem obter descrições muito claras, concisas e não-redundantes da informação que desejam representar. Não há distinção entre argumentos de entrada e de saída, de modo que qualquer combinação de argumentos pode ser empregada.

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.

- **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 quer que sejam necessárias, 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.

5. Aplicações Reais

Serão exibidas cinco aplicações onde se podem encontrar os conceitos da programação em lógica na prática. A primeira delas será o processamento de linguagem natural (PLN), com a demonstração da validação de sentenças em uma determinada linguagem. Em seguida um Sistema Especialista na Área Médica, SEAMED, que auxilia doutores na prescrição de um fármaco adequado, baseado em informações sobre paciente. A terceira trata-se do miniEliza, um programa psicoterapêutico que mantém uma espécie de diálogo com o usuário de forma que pareça uma outra pessoa conversando com ele. Na seqüência segue uma passagem de um livro infantil “Alice no País das Maravilhas”, no qual a protagonista consegue descobrir qual o dia da semana aplicando princípios básicos de raciocínio lógico. Será exibido também o código fonte deste problema modelado na linguagem Prolog. Por fim, será mostrada como última aplicação o ToonTalk, um ambiente de programação voltado especialmente para o público infantil onde vários conceitos da programação em lógica são aplicados com interessantes abstrações computacionais.

5.1. Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em alguma língua natural (por exemplo, tradução e interpretação de textos, busca de informações em documentos e interface homem-máquina). 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 e expressões que povoam a comunicação natural humana. O uso da lógica das cláusulas de Horn 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 correspondiam a diferentes estratégias de análise.

Uma gramática é uma especificação formal da estrutura das sentenças permitidas numa linguagem. O modo mais simples de definir uma gramática é especificando um conjunto de símbolos terminais, denotando palavras da linguagem, um conjunto de símbolos não-terminais, denotando os componentes das sentenças, e um conjunto de regras de produção, que expandem símbolos não-terminais numa seqüência de símbolos terminais e não-terminais. Além disso, a gramática deve ter um símbolo não-terminal inicial.

Seja, por exemplo, a gramática a seguir define um fragmento da língua portuguesa:

frase → sujeito, predicado
sujeito → artigo, substantivo
predicado → verbo, artigo, substantivo
artigo → [o]
substantivo → gato | rato
verbo → caçou

Simulando o reconhecimento automático de frase passo a passo para:

?-frase([o,gato,caçou,o,rato],[]).

A Figura 1 mostra como a programação lógica pode ser usada no PLN.

Figura 1 – reconhecimento de *frase([o,gato,caçou,o,rato],[])*

5.2. SEAMED

5.2.1 Sistemas Especialistas

Um sistema especialista é a representação, em um programa computacional, de um ser humano especialista de um determinado domínio de aplicação, devendo agir como tal. Ele deve ser capaz de agir racionalmente, lidar com incertezas e informações incompletas na solução de problemas que precisam da decisão de um especialista. Podem ser considerados sistemas baseados em conhecimentos, porém com a capacidade de explicar seu comportamento e decisões para o usuário, permitindo melhor avaliação da incerteza. Os domínios de aplicação mais comuns hoje em dia, incluem sistemas para diagnósticos médicos, localização de falha de equipamentos ou interpretação de dados de medição.

5.2.2 Projeto Seamed

O projeto Seamed é responsável pelo desenvolvimento de sistemas especialistas (SE) para a área médica e que utiliza diferentes tecnologias de apoio a decisão. Os SE são a grande contribuição que o estudo da Inteligência Artificial vem trazendo para tornar a vida mais simples. Este projeto se torna único pela proposta inovadora de levar o conhecimento de um especialista humano, geralmente localizado em um grande centro de pesquisa para os médicos no interior do país, onde é grande a carência de informações atualizadas. As tecnologias utilizadas no projeto podem ser dividias em: (1) **Sistemas baseados em regras** Estes sistemas são programas computacionais, capazes de estabelecer conclusões, baseando-se nas regras utilizadas por especialistas humanos, para resolver problemas objetivos e de domínio específico. A especialidade médica utilizada para o desenvolvimento do SE foi o uso de fármacos em psiquiatria (transtornos afetivos) e em cardiologia (hipertensão no idoso); (2) S**istemas baseados em redes probabilísticas:** Estes sistemas utilizam representações gráficas de dependências probabilísticas com redes causais. Essa representação permite manipular a incerteza com base em princípios matemáticos fundamentados e modelar o conhecimento do especialista do domínio de uma forma intuitiva. A área médica abordada no desenvolvimento do SE foi a de apoio a escolha do diagnóstico em cardiopatias congênitas.

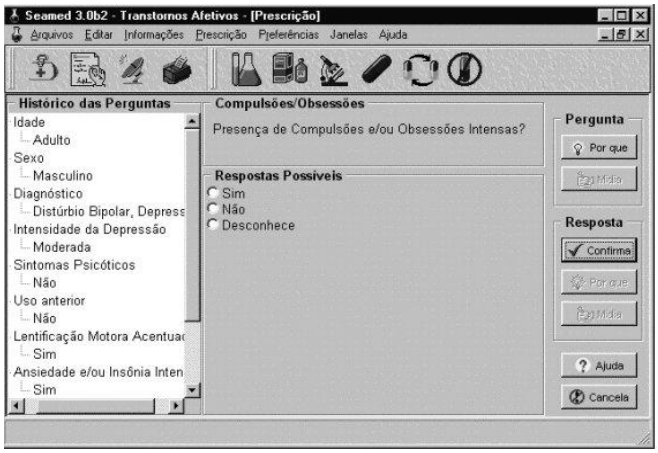

Figura 2 – Interface do software SAMED com o usuário.

A figura representa a interface com o usuário do SEAMED. Os dados de entrada serão adotados como fatos para o programa e através das regras internas será possível deduzir qual o fármaco mais indicado para aquele paciente. Observe que neste caso as regas internas são muito bem definidas uma vez que há um formulário fixo. Este software possui um banco de dados formado por mais de 5.000 medicamentos já cadastrados.

5.3. miniEliza

O programa *Prolog* miniEliza é uma miniatura do famoso "programa-terapeuta" Eliza (Joseph Weizenbaum, 1966), que conduz uma "conversa" com o usuário à maneira de um psicoterapeuta estereotípico, evitando perguntas diretas e geralmente dando respostas vagas, neutras, mas bem-escolhidas, que levem o "paciente" a definir seus problemas em suas próprias palavras. Era um programa bem simples, possuindo apenas 204 linhas de código fonte.

Eliza foi o primeiro software para simulação de diálogos, os chamados “robôs de conversação”. Foi declaradamente concebido para não ser inteligente, mas convencer o usuário do contrário. Dessa forma, pode ser considerado como uma crítica ao Teste de Turing, que atribui "inteligência" a um sistema computacional se um humano não for capaz de distinguir entre o comportamento humano e o comportamento do sistema em questão.

Eliza e, por conseguinte, miniEliza jamais tentam compreender a entrada do usuário, mas apenas a varrem em busca de umas poucas palavras-chave ou seqüências que permitam produzir uma resposta que dê continuidade à conversa. Esse processo pode ser entendido como a aplicação de regras simples que mapeiam padrões de entrada em respostas plausíveis. Vejamos exemplos e alguns cuidados que toma antes de apresentar alguma resposta

regra: *I am X ----> Why are you X ?*

aplicação: *I am very sad and lonely ----> Why are you very sad and lonely?*

you: *I am sure <u>you are</u> against <u>me</u>.*

eliza: *Why are you sure <u>I am</u> against <u>you</u>?*

A inversão de sentido do discurso, por sua vez, pode ser obtida com alto grau de acerto por meio de simples regras de substituição, tais como:

regra 1: *you are ----> i am*
regra 2: *me ----> you*

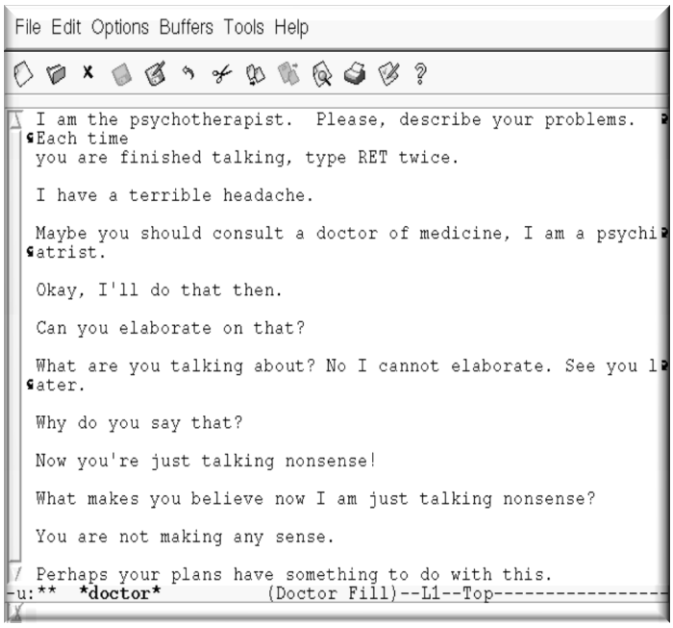

Figura 3 – Exemplo de um diálogo entre um usuário e o miniEliza.

A Figura 3 representa a interface com o usuário para o miniEliza. Observe o modo como o programa interage como o usuário. No mínimo possui um diálogo mais diversificado do que o de um psicólogo de verdade, uma vez que não repete sempre o mesmo discurso frente a qualquer problema de seus pacientes: “E como você se sente sobre isso?”

5.4. Alice no país das maravilhas

Alice's Adventures in Wonderland (freqüentemente abreviado para "*Alice in Wonderland*"), é a obra mais conhecida de Lewis Carroll (Charles Lutwidge Dodgson, escritor e matemático britânico) e uma das mais célebres do gênero literário *nonsense*, sendo considerada obra clássica da literatura inglesa. O livro, que começou a ser escrito em 1862, conta a história de uma menina chamada Alice que cai em uma toca de coelho e vai parar num lugar fantástico povoado por criaturas peculiares e antropomórficas.

O livro faz brincadeiras e enigmas lógicos, o que contribuiu para sua popularidade. Carroll também faz alusões a poemas da Era Vitoriana (no Reino Unido foi o período do reinado da Rainha Vitória, em meados do Século XIX, a partir de Junho de 1837 a Janeiro de 1901) e a alguns de seus conhecidos, o que torna a obra mais difícil de ser compreendida por leitores contemporâneos. É uma das obras escritas da literatura inglesa que tiveram mais adaptações na história do cinema, TV e teatro.

Em uma das passagens do livro, Alice entra em uma floresta e perde a noção dos dias da semana. O Leão e o Unicórnio eram duas das criaturas que habitavam a floresta. Alice os encontra e deseja obter alguma informação sobre o dia da semana. O problema é que o Leão mente em determinados dias da semana e o mesmo acontece com o Unicórnio e Alice sabe que eles mentem e sabe em que dia cada um mente.

Naquela época o Leão mentia as segundas, terças e quartas e falava a verdade nos outros dias da semana. O Unicórnio mentia as quintas, sextas e sábados e falava a verdade nos outros dias da semana. Quando Alice os encontra o Leão ele diz: “Ontem foi um dos meus dias de mentir!”. O Unicórnio diz: “Ontem foi um dos meus dias de mentir!”. A partir dessas informações, Alice, garota muito esperta, descobriu o dia da semana. Segue a modelagem deste problema feita na linguagem Prolog.

```
mente(leao,segunda).
mente(leao,terca).
mente(leao,quarta).
mente(unicornio,quinta).
mente(unicornio,sexta).
mente(unicornio,sabado).

antes(segunda,domingo).
antes(terca,segunda).
antes(quarta,terca).
antes(quinta,quarta).
antes(sexta,quinta).
antes(sabado,sexta).
antes(domingo,sabado).

alice :- ((K = leao, J = unicornio) ; (K = unicornio, J = leao)),
        antes(X,A),
      ( (mente(J,X), not(mente(K,X)), not(mente(J,A)), mente(K,A)) ;

        (mente(J,X), mente(K,X), not(mente(J,A)), not(mente(K,A))) ;

        (not(mente(J,X)), not(mente(K,X)), mente(J,A), mente(K,A)) ),

        nl,nl,write('Alice deduziu que era '),write(X),nl,nl.
```

Ao executar este programa tem-se como resposta: “Alice deduziu que era Quinta”.

5.5. ToonTalk

É simultaneamente uma linguagem de programação e um ambiente de programação. O objetivo principal do ToonTalk é ser um sistema de programação de computadores ao alcance de crianças, incluindo crianças muito novas. O texto inicial, "toon", é uma abreviatura de "cartoon", ou desenho animado. É um sistema que utiliza as técnicas de programação por demonstração e que permite que o programador generalize as ações gravadas removendo, explicitamente, seus detalhes.

Em ToonTalk, o programa é executado como uma coleção de processos autônomos que se comunicam assincronicamente, e o comportamento do processo é especificado como um conjunto de cláusulas que são guardadas. Uma cláusula é construída pela execução de

operações em uma única estrutura de dados de exemplo. Para fazer com que a cláusula seja capaz de operar em outras estruturas de dados, o programador precisa, apenas, remover detalhes do que está escondido ou a parte condicional da cláusula.

O ToonTalk é construído com a idéia de programação animada. Programas animados não são construídos pela digitação de textos ou pela construção de diagramas ou seqüência de figuras. Em vez disso, o programador é colocado como um personagem dentro do mundo virtual animado onde as abstrações da programação são substituídas por comandos tangíveis. Segue uma tabela com algumas destas abstrações:

Abstração Computacional	**Concretização no ToonTalk**
Computação	Cidade
Métodos	Robôs
Pré-condições de método	Conteúdo da mente do robô
Ações de métodos	Ações ensinadas aos robôs
Capacidade de transmitir canais	Pássaros
Capacidade de receber canais	Ninhos
Armazenagem de programas	Agendas

Tabela 1 – Relação entre os elementos do ToonTalk e a computação.

Uma estrutura de dados, por exemplo, é uma caixa cujos buracos podem ser preenchidos com números, pedaços de texto, outras caixas, pássaros, ninhos e robôs. Pássaros e ninhos são analogias concretas da capacidade de enviar e receber dados em canais de comunicação. Um robô é uma estrutura treinada pelo programados para entrar em ação quando receber uma caixa. O pensamento bolha de um robô mostra as condições que precisam ser satisfeitas antes do robô executar uma ação. Para generalizar um robô, um programador precisa usar um aspirador de pó animado para remover os detalhes dentro da bolha de pensamentos do robô.

6. Conclusão

É inegável a relação existente entre a Matemática a e Computação. Sem a primeira talvez não fosse possível existir todos estes avanços tecnológicos com os quais estamos acostumados nos dias atuais. Os resultados de grandes matemáticos do passado influenciaram o desenvolvimento da área da computação e, conseqüentemente, para o surgimento da programação em lógica.

Pelo fato de muitas pessoas iniciarem seus estudos na área da computação usando o paradigma imperativo, há quem diga que este tipo de programação não seja a mais natural. Mas tudo não passa de uma maior familiaridade com este paradigma. Trata-se apenas de um novo modo de pensar, de um jeito mais lógico e racional, uma vez que todo programa

será escrito através de fatos e regras definidas de tal modo que novos fatos possam ser inferidos a partir de deduções lógicas.

Foi feita uma breve introdução sobre os conceitos da programação em lógica e uma abordagem sobre uma das mais conhecidas linguagens deste tipo de programação: Prolog. Além de explicar o seu funcionamento, foi exemplificado com um código fonte de um dos problemas encontrados no livro do escritor britânico e matemático Charles Dodgson: Alice no país das maravilhas. Por fim, foram exibidas cinco aplicações reais nas quais puderam ser encontrados os conceitos da programação em lógica.

7. Referências

[1] GRANDI, Roges Horácio. **SEAMED – Sistema Especialista para a Área Médica**. Disponível em: <http://www.inf.ufrgs.br/~dflores/seamed/default.htm>. 1999. Acesso em: 17 nov. 2009.

[2] Animated Programs – **ToonTalk: "Tornando a programação brincadeira de criança".** Disponível em: <http://toontalk.com/br/toontalk.html>. 2002. Acesso em: 18 nov. 2009.

[3] PELIZZONI, Jorge Marques. **Introdução à Inteligência Artificial – Projeto I – miniEliza.** Disponível em: <http://www.icmc.sc.usp.br/~sandra/8/miniEliza.htm>. Acesso em: 17 nov.2009

[4] PALAZZO, Luiz A. M. **Introdução a Programação PROLOG**. Ed. UCPEL. Pelotas, 1997.

[5] COURA, Debora P. **Produzindo animações através da programação por demonstração.** 2006. Dissertação (Mestrado em Ciências da Computação). Universidade Federal de Viçosa – Minas Gerais.

[6] COVINGTON, M. **NLP for Prolog Programmers**, Prentice-Hall, 1994.

[7] TOSCANO, Wagner. **Linguagem Prolog Introdução**. USP, ago. 2009.