Epistemologia genética e inteligência artificial: linguagens Lisp e Logo*

MARIA VICTORIA G. CAVALCANTI A. CUNHA**

1. Introdução; 2. A linguagem Lisp; 3. Evolução do Lisp ao Logo; 4. Instrumento de auxílio ao pensamento.

Aborda a utilização do computador, sob o enfoque da psicologia cognitiva. Procura-se relacionar características de classes de linguagens computacionais com conceitos da epistemologia genética de Jean Piaget, supondo-se que a busca de soluções de problemas, com auxílio de computador e com o uso de linguagens como Lisp e Logo, possa favorecer o desenvolvimento de estruturas intelectuais. A reflexão sobre processos cognitivos, com ajuda de recursos da inteligência artificial, constitui um tema de interesse interdisciplinar, aberto a investigações.

1. Introdução

A questão da utilização do computador como um instrumento propício ao desenvolvimento cognitivo, assim como à observação desse processo, envolve a indagação de que linguagens computacionais seriam adequadas a esse objetivo. Dentre as alternativas existentes, Logo destaca-se como uma linguagem apropriada.

Seymour Pappert considera, como uma das raízes do Logo, a epistemologia genética de Jean Piaget. Coloca-se explicitamente como um adepto das idéias piagetianas, relativas à teoria mais geral do conhecimento.[1] Princípios da Teoria de Desenvolvimento Cognitivo embasam o que denomina "ambiente Logo" e "uma filosofia Logo". Focaliza a criança como construtora ativa de

* Artigo apresentado à Redação em 17.1.86.

** Psicóloga; mestre em psicologia teórico-experimental pela PUC/RJ; pesquisadora do Centro Brasileiro de Pesquisas Psicossociais do ISOP/FGV. Chefe do Centro: Monique R. A. Augras. (Endereço da autora: Rua Com. Francisco Leal, 192 — Itanhangá — Barra — Rio de Janeiro, RJ.)

[1] Embora questione o caráter de necessidade da ordem dos estágios, colocado por Piaget, Pappert considera que a natureza dos estímulos, presente numa cultura, determina a ordem em que as estruturas cognitivas são construídas. Admite uma alteração na ordem de sucessão, assim como a construção de estruturas mais avançadas, a partir da interação da criança com estímulos qualitativamente distintos, como o computador.

suas próprias estruturas intelectuais, a partir de sua interação com o ambiente (Pappert, 1980); enfatiza a importância de sua participação ativa, num processo de elaboração livre, em que é dada vazão à sua curiosidade e raciocínio (Pappert, 1973).

No entanto, caso objetive-se avaliar aspectos relacionados ao desenvolvimento cognitivo, decorrentes da interação da criança com o computador, faz-se necessária uma análise que aborde aspectos específicos da linguagem em si, e não apenas do ambiente criado para sua utilização.

Na análise apresentada a seguir, procura-se focalizar, simultaneamente, duas das raízes do Logo apontadas por Pappert — a epistemologia genética e a inteligência artificial (linguagem Lisp e modelos utilizados) — assim como aspectos acrescentados por Pappert à linguagem de origem do Logo, visando uma aplicação voltada para o desenvolvimento da criança.

2. A linguagem Lisp

Lisp (desenvolvida no MIT, por McCarthy, no final da década de 50 e início da de 60) é uma linguagem funcional, utilizada sobretudo em computação simbólica e inteligência artificial. Contrasta fortemente com linguagens imperativas.

Segundo Carlo Chezzi (1985) grande número das linguagens existentes forma uma progressão ao longo de um mesmo conjunto de idéias. Podem ser consideradas abstrações construídas sobre a arquitetura da máquina de Von Neumann. A base do projeto, de linguagens imperativas, decorre da preocupação com a eficiência de execução, medida pelo desempenho em um computador convencional.

Lisp, ao contrário, foi a primeira linguagem projetada sobre uma firme base matemática, e, portanto, a utilizar um modelo independente de arquitetura de máquina (embora não deixe de mostrar alguns efeitos dessa arquitetura em seu projeto).

Do contraste entre esses dois enfoques, para o projeto de linguagens de programação, ou entre os modelos subjacentes às linguagens, resulta uma importante conseqüência. A programação feita em linguagens imperativas ou funcionais apresenta estilos bastante distintos.

A programação em linguagens imperativas reflete a preocupação com alocação de memória, atribuição de valores a variáveis, que são guardados na memória, repetição de grupos de comandos. Em oposição, com a utilização de linguagens funcionais a atenção não é desviada para tais detalhes de processamento (valores não são atribuídos a variáveis, mas produzidos por aplicação de funções, e o controle de seqüência é obtido através de recursão); ao contrário, pensa-se em termos de estruturas totais e completas (estruturas hierárquicas de listas, em Lisp). Comandos que processam estruturas inteiras de dados e técnicas de programação recursiva, em lugar de comandos de controle de iterações e de desvios baseados em condicionais, induzem a uma programação que flui naturalmente (*top-down*) ao invés de apresentar-se como um *ziguezague*.

Em síntese, a riqueza da notação funcional pode, por um lado, facilitar a reflexão sobre o problema a ser resolvido e, por outro, refletir muito mais claramente a lógica subjacente à sua solução.

Um importante aspecto a ser considerado resulta do modelo comum utilizado na construção do Lisp e nos estudos de Piaget (lógica-matemática): a possibilidade de serem relacionados recursos oferecidos por essa linguagem para o desenvolvimento de programas aos disponíveis na estrutura cognitiva de quem a utiliza. Alguns dos aspectos que caracterizam Lisp como linguagem funcional permitem essa análise:

— um conjunto (pequeno repertório) de funções primitivas;

— um mecanismo para amarrar um nome a uma nova função sendo definida;

— um conjunto de dados com estrutura simples e regular;

— um conjunto de formas funcionais (métodos de combinar funções). Uma forma funcional é uma função que recebe outras como parâmetros e devolve uma nova função como resultado.

Esses componentes de Lisp apresentam interesse, do ponto de vista de processos intelectuais. A partir de um fácil aprendizado de pequeno número de funções primitivas, todas as demais podem ser definidas pelo usuário. Conceitos podem ser construídos e generalizados progressivamente, num processo em que sucessivamente cada definição mais abrangente pode incluir uma anterior.

Por outro lado, Lisp induz a uma programação modular; problemas tendem a ser focalizados no seu todo e simultaneamente nas partes que o compõem. Pequenas funções podem ser criadas para os subproblemas e inter-relacionadas enquanto componentes de um todo.

Acrescentando-se o fato de ser uma linguagem altamente interativa, Lisp possibilita que o efeito de um único comando seja isoladamente testado e compreendido e, de forma igualmente simples, que um conjunto de comandos seja imediatamente avaliado. A testagem de um programa modular, isto é, em seguida à definição de cada função, e finalmente do programa principal (geralmente uma única expressão), facilita a descoberta e compreensão da causa de erros (relativos a uma função particular ou resultante de inter-relacionamentos).

A avaliação de processos cognitivos envolvidos na programação através da construção de funções, e de sua combinação por meio de formas funcionais,[2] pode ser apoiada em estudos piagetianos. Piaget distingue o pensamento característico do período operacional concreto do pensamento operacional formal com base no sistema cujas transformações formam (com a operação de combinação) o grupo INCR (idêntica, inversa, recíproca e correlata).[3]

Outras características de Lisp poderiam ainda ser destacadas, como, por exemplo, o fato de estruturas de lista permitirem programação em linguagem natural.

3. Evolução do Lisp ao Logo

Diversos dialetos Lisp foram desenvolvidos. As modificações introduzidas no Lisp original objetivaram, sobretudo, torná-lo mais eficiente em termos de má-

[2] No Lisp original há uma única forma funcional: composição funcional. A função resultante equivale à aplicação de uma função, ao resultado de aplicar outra função.

[3] A composição funcional de duas ou mais das quatro transformações *I, N, R, C* é equivalente à aplicação particular de uma delas ($NR = C$; $RC = N$; $NC = R$; $NRC = I$ etc.).

quina (como através de inclusão de comandos não-aplicativos, tais como os que permitem atribuição de valores a variáveis, ou o controle de iterações). Além disso, buscaram facilitar o trabalho do programador com a expansão da biblioteca de funções, de forma a tornar disponíveis as mais freqüentemente utilizadas. A linguagem Logo (que em essência pode ser considerada como um dialeto Lisp) apresenta, no entanto, características que a distinguem dos demais dialetos.

Dentre as diferenças entre Logo e Lisp, algumas serão consideradas a seguir.

- *Logo gráfico*

Constitui o aspecto que mais distingue a linguagem Logo de Lisp.[4] No Logo gráfico, a expressão de movimentos relativos é mais simples do que em uma abordagem baseada em coordenadas cartesianas: as referências são a posição e a direção da tartaruga. Apoiando-se em contribuições da inteligência artificial, Pappert conduz a criança a "ensinar" a tartaruga a executar um processo. Do ponto de vista cognitivo, ensinar a tartaruga exige da criança uma reflexão sobre a tarefa; visualizar os passos que constituem esse processo é concretizar uma abstração (Pappert, 1980).

- *Simplificação da notação*

Duas dificuldades notacionais de Lisp foram minoradas em Logo. Reduzindo a quantidade de parênteses necessários, Logo mantém as vantagens da simplicidade e regularidade da sintaxe de Lisp (tais como a facilidade de aprendizagem de regras básicas), eliminando a desvantagem resultante (ou seja, uma das causas de erros de programação nessa linguagem). Além disso, a notação funcional, reversa da polonesa, foi adicionada à notação infixada em funções numéricas, padrão para operadores aritméticos.

- *Comandos não-aplicativos*

Se, por um lado, a introdução de comandos não-aplicativos pode levar a maior eficiência em termos de máquina, por outro pode ser necessária, se um dos objetivos da linguagem é ser acessível a crianças. Com base em resultados obtidos no Centro de Epistemologia Genética, podemos supor a impossibilidade de utilização, em períodos de desenvolvimento anteriores ao operacional formal, de técnicas recursivas ou de composição funcional; isto é, de programar sem o recurso de comandos de atribuição e de controle de iteração.[5]

- *Biblioteca de funções*

Embora funções tenham sido acrescidas ao Lisp original nos diversos dialetos, na linguagem Logo elas se apresentam mais sistematizadas e, conseqüentemente, mais favoráveis ao desenvolvimento do raciocínio.

4. Instrumento de auxílio ao pensamento

De uma forma geral, a atividade de programar, em qualquer linguagem computacional, pode ser considerada propícia ao desenvolvimento cognitivo. Meca-

[4] Introduzido, a partir de Logo, no dialeto Mulisp.
[5] Piaget considera a forma geral do grupo INCR como modelo de cognição do adolescente.

nismos de *assimilação* e *acomodação* entram em atuação, na tarefa de elaborar e explicitar um procedimento de solução. Na interação do sujeito com o computador, "situações de conflito" são desencadeadas quando suposições iniciais são estabelecidas e em seguida confrontadas com os resultados obtidos, transformadas com base nesses resultados e em função de um objetivo final que vai, gradativamente, sendo alcançado.

O computador, utilizado como uma ferramenta para observação desse processo, pode tornar mais claros e explícitos os passos dessa dinâmica.

"Num certo sentido, todas as linguagens de programação são 'instrumentos para ajudar a pensar'. Se você pode resolver um problema com uma linguagem, você pode também fazê-lo com outra. O que diferencia as linguagens é o fato de que certos tipos de problemas são mais facilmente resolvidos com uma do que com outra" (Harvey, 1984).

Acrescenta-se a essa citação que certos tipos de solução são mais facilmente representados e testados com o uso de umas do que com o de outras linguagens. Conseqüentemente, a solução de um mesmo problema, com a utilização de diferentes linguagens, pode levar a construções qualitativamente distintas.

Supõe-se que linguagens imperativas auxiliam, mais do que linguagens funcionais, em problemas que admitem soluções em certo nível intuitivas (devido à existência de comandos de desvio baseados em testes de estruturas do tipo *while*). Esse tipo de facilitação induz a soluções parcialmente intuitivas. Linguagens funcionais, ao contrário, favorecem a "abstração reflexiva" e uma compreensão das regras que governam o problema, em um nível distinto.

Em processos de solução de problemas — desde ações iniciais basicamente exploratórias, passando por descobertas das primeiras propriedades e relações entre os diversos elementos, e pela discriminação, entre as possíveis, daquelas que são relevantes, até a constituição de hipóteses sobre regras que governam o domínio do problema — o uso de diferentes linguagens pode levar a estratégias distintas.

Postula-se que soluções corretas possam ser atingidas através de procedimentos em que os fatores componentes são considerados, mas nem todos de forma totalmente clara e consciente. O sujeito que descobre uma solução, mas ainda nesse nível, é capaz de executar concretamente um processo que o conduz a uma resposta correta, mas não de explicitá-lo integralmente. Na tarefa de programar, ao procurar definir os passos que constituem a solução, não são consideradas todas as regras seguidas, mas apenas aquelas conscientemente utilizadas. A descrição do processo de solução, conseqüentemente, não equivale à estratégia semi-intuitiva descoberta, e o resultado obtido no computador poderá ser diferente (como eventualmente poderá ser o mesmo, dependendo de efeitos aleatórios de fatores não-incluídos). A situação de conflito cognitivo, gerada pela discrepância entre o resultado esperado e o verificado, propicia a descoberta e conscientização dos aspectos que distinguem os dois procedimentos. No entanto, um passo cognitivo adicional é necessário para a formulação de uma lei mais genérica e abrangente, que inclua simultaneamente todos os fatores em jogo. Supõe-se que linguagens como Lisp facilitem o desenvolvimento da compreensão do problema, para que esse passo seja atingido; e que linguagens convencionais não só tornem obscuras inter-relações existentes, como facilitem soluções em que alguns aspectos permanecem não

totalmente compreendidos, mas apenas controlados como exceções à regra, e resolvidos por critérios adicionais.[6]

Finalmente, linguagens como Logo, simples o suficiente para serem programadas com recursos cognitivos adquiridos anteriormente ao nível operacional formal, e poderosas no auxílio a um pensamento de nível formal, podem ser propícias a uma evolução gradativa entre um e outro tipo de solução. Isto é, podem favorecer a aquisição de estruturas cognitivas na evolução do período operacional concreto ao formal (decalagem vertical), assim como o desenvolvimento dentro desse último período (decalagem horizontal).

A utilização da linguagem Logo por crianças vem cada vez mais despertando o interesse de especialistas em educação, visando aplicações específicas nessa área.

As questões focalizadas nesse texto, diretamente relacionadas com aspectos cognitivos, vêm, no entanto, sendo pouco investigadas. Salienta-se a importância da análise da utilização do computador e de linguagens computacionais não só por sujeitos de nível operacional concreto, como também de nível formal. E não apenas objetivando a aprendizagem de conteúdos específicos, mas a aquisição de estruturas de pensamento. E sobretudo, viabilizando o esclarecimento de aspectos teóricos sobre processos cognitivos envolvidos nessa dinâmica.

Esse tema, de interesse interdisciplinar, constitui uma recente área de estudos a serem desenvolvidos pela psicologia cognitiva e pela inteligência artificial.

Abstract

Discusses the utilization of the computer, from the point of view of cognitive psychology. An effort is made to relate the characteristics of some classes of computational languages, to the concepts of Jean Piaget's genetical epistemology. It is supposed that the search of problem solutions with the help of the computer, using languages lake Lisp and Log, can enhance the development of intelectual structures. To reflect about cognitive processes, with the help of artificial intelligence resources, is a theme of interdiciplinary interest, open to investigations.

Referências bibliográficas

Abelson, H. *Logo for the Apple II.* Massachusetts, McGraw-Hill, 1982.

Chezzi, C. & Jazayeri, M. *Conceitos de linguagens de programação.* Rio de Janeiro, Campus, 1985.

Harvey, B. In: Barros, Lygia. Apostilas de curso de Logo. Rio de Janeiro, 1984.

Pappert, S. *Uses of technology to enhance education.* Massachusetts, MIT, Division for Study and Research in Education/Cambridge, Artificial Intelligence Laboratory, 1973.

———. *Mindstorms: children, computers and powerfull ideas.* New York, Basic Books, 1980.

Pratt, T. W. *Programming languages: design and implementation.* New York, Prentice-Hall, 1985.

Winston, P. H. & Horn, B. C. P. *Lisp.* California, Addison-Wesley, 1981.

[6] Estudos relativos às postulações sobre processos de solução de problemas, aqui apresentados, vêm sendo desenvolvidos pela autora.