PSICOLOGIA GENÉTICA E INTELIGÊNCIA ARTIFICIAL:
UMA PROPOSTA DE ESTUDO INTERDISCIPLINAR

Maria Victoria Gusmão Cavalcanti de A. Cunha

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS–GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Carlos Alberto da Silva Franco, D.Sc.
(Presidente)

Prof. Nelson Maculan Filho, Ph.D.

Prof. Antonio de Almeida Pinho, D.Sc.

Prof. Franco Lo Presti Seminério, Doutor e Livre Docente em Psicologia

Prof. Ued Martins Manjud Maluf, Doutor e Livre Docente em Psicologia

RIO DE JANEIRO, RJ – BRASIL
março de 1990

CUNHA, MARIA VICTORIA G. C. A.

Epistemologia e Psicologia Genética e Inteligência Artificial: Uma Proposta de Estudo Interdisciplinar [Rio de Janeiro] 1990.

X, 131 p. p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1990)

Tese – Universidade Federal do Rio de Janeiro. COPPE

1. Psicologia e Inteligência Artificial I. COPPE/UFRJ II. Título (Série).

A Daniela, Cristiana e Lucas

AGRADECIMENTOS

Ao Prof. Carlos Alberto da Silva Franco, que acreditou numa proposta interdisciplinar, admitindo no Curso de Mestrado em Engenharia de Sistemas e Computação, candidata de outra área, e, com seu conhecimento, orientou as reflexões e reformulações desenvolvidas nesse trabalho.

Ao Prof. Franco Lo Presti Seminério, diretor, mestre e amigo durante todo o exercício profissional, por seus ensinamentos e apoio a essa e demais buscas que procurei conduzir.

À Profa. Monique Augras, que como chefe concordou, apoiou e incentivou uma procura de conhecimentos em outra disciplina, em substituição ao Curso de Doutorado em Psicologia.

Ao Prof. Ued Maluf, que como chefe deu condições sem as quais não teria sido possível a execução desse trabalho, acreditou, participou, e demonstrou interêsse e amizade em tantos momentos significativos.

À Profa. Ligia Barros, que com seus ensinamentos abriu–me uma nova perspectiva de conhecimentos.

À Elida Singelman, Heidi Gonsalves dos Santos, Yara Silveira Faria e a todos os colegas do ISOP, que acompanharam e estiveram presentes nessa tarefa.

À Therezinha Ebbert e ex–colegas do ISOP, que com sua amizade contritribuiram para o desenvolvimento desse estudo, mesmo após a mudança de nossos rumos profissionais.

À Ivana e Daisy, que digitaram esse trabalho.

A Luiz, que cedeu seu microcomputador para implementação e experimentações do programa.

À minha mãe, a meus irmãos, a Chico, amigo muito caro, presente em todos os momentos.

À Dani, Cris e Lucas, filhos e grandes amigos, que não só aceitaram como

participaram dessa busca, a despeito dela tornar–me menos disponível nesse período.

RESUMO DA TESE APRESENTADA À COPPE/UFRJ COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.)

PSICOLOGIA GENÉTICA E INTELIGÊNCIA ARTIFICAL: UMA PROPOSTA DE ESTUDO INTERDISCIPLINAR

Maria Victoria Gusmão Cavalcanti de A. Cunha

Março de 1990

ORIENTADOR: Carlos Alberto da Silva Franco
PROGRAMA: Engenharia de Sistemas e Computação

Esse trabalho parte do pressuposto que a Psicologia Genética e a Inteligência Artificial podem obter um enriquecimento, com o desenvolvimento de pesquisas interdisciplinares.

Um arcabouço teórico, que focaliza a construção e não apenas a aplicação do conhecimento, parece constituir fundamentação particularmente adequada a investigações da Inteligência Artificial, sobre mecanização de processos de descoberta.

Modelos mais ricos que os buscados na lógica, e contexto experimental propício à observação da aquisição do conhecimento e à inferência de mecanismos cognitvos podem ser encontrados pela Psicologia Genética.

Com base nessa suposição apresenta–se proposta em que conhecimentos desenvolvidos nas duas áreas fornecem suporte. Define–se um processo interativo, em que homem e máquina abstraem regras a partir de sua própria ação, e da ação e descobertas do outro. Aquisições do sujeito cognoscente da Psicologia e do sujeito cognoscente de Inteligência Artificial são focalizadas enquanto resultantes de processo interdependente.

A fim de explicitar e desenvolver essa proposta, apresenta–se: um modelo genérico, em que meta–regras e operadores de alto nível, aplicados a regras, determinam o processo construtivo da máquina; uma aplicação, em que regras e operadores são específicos ao domínio em que a investigação se processa.

Limitado a uma primeira investigação, experimentações e reformulações, de

uma idéia inicial, são ainda necessárias. Para que reconstruções de aspectos conceituais do problema possam conduzir a definições mais precisas, e a modelos mais adequados.

ABSTRACT OF THESIS PRESENTED CO COPPE/UFRJ AS PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (M.Sc.)

GENETIC PSYCHOLOGY AND ARTIFICIAL INTELLIGENCE: A INTERDISCIPLINARY STUDY PROPOSITION

Maria Victoria Gusmão Cavalcanti de A. Cunha
March, 1990

THESIS SUPERVISOR: Carlos Alberto da Silva Franco
DEPARTMENT: Systems and Computation Engineering

This work starts from the supposition that the Genetic Psychology and the Artificial Intelligence can gain an enrichment from the devellopment of interdisciplinary research.

A theoretical framework that foccuses on the construction and not only on the application of knowledge, seems to constitute a particularly adequate basis to investigations on Artificial Intelligence, about mecanization of discovery processes.

Riches modelels than thouse searched in logics, and experimental context propicious to the observation of knowledge acquisition and to the inference of cognitives mechanisms can be found by Genetic Psychology.

Based on this supposition we present a proposition in which developped knowledges in both areas give support. We define an interactive process as that in which man and machine abastract rules from their own action and from action and discoveries from each other.

Psychology subject and Artificial Intelligence subject acquisitions are foccused while resulting from interdependent process.

With the objective of expliciting and devellopping this proposition it is presented: a general model, in which meta–rules and high level operators aplied to rules determine the constructive process of the machine; an application, in which rules and operators are specific to the domain in which process the investigation.

Limit to a first inestigation, experimentations and reformulations of a initial idea are still necessaries, in order that reconstructions of conceptual aspects of the problem may conduct to more precise definitions and to more adequate models.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

Problemas focalizados pela Inteligência Artificial tem levado seus pesquisadores à conclusão da necessidade de um conhecimento prévio, sobre processos mentais humanos. O fato dessa disciplina ser relativamente nova, e da psicologia cognitiva ter desenvolvido um significativo corpo de conhecimento, que lhe é necessário e complementar, vem cada vez mais aproximando essas duas ciências. Embora as metas de cada uma sejam distintas, ambas se preocupam com a representação do conhecimento (o que é conhecido, e como esse conhecimento é organizado) e com processos de aquisição e/ou aplicação do conhecimento (como representações são desenvolvidas, transformadas e aplicadas por meio de processos cognitivos).

De fato, encontra–se em SMITH (1985) um levantamento de teorias de psicologia cognitiva que vem sendo incorporadas em programas computacionais.

No entanto, acredita–se que a riqueza e fecundidade de idéias de Epistemologia e Psicologia Genética, tornem–na particularmente propícia a estudos interdisciplinares. BOVET e MONTANGERO (1983) consideram que essa teoria pode inspirar diversos tipos de pesquisa. Não só as que retomam o estudo, sob um ângulo novo, de problemas colocados anteriormente, como as que estendem o campo da psicologia genética até criar ligações com outras disciplinas.

O caráter interdisciplinar dos estudos desenvolvidos em Genêve levou alguns cientistas cognitivistas a interessarem–se por essa troca.

JEROME BRUNER (1983), já há algumas décadas, vira pontos em comum entre seu pensamento e o de Inhelder, e interessou–se pelo desenvolvimento de estudos em paralelo, por época das pesquisas longitudinais da discípula de Piaget.

SEYMOUR PAPERT (1973, 1980) considera entre as raízes da linguagem LOGO a Epistemologia Genética de Piaget (além de modelos da Inteligência Artificial e dos estudos de Bru ner). Coloca–se como adepto das idéias piagetianas, relativas à teoria mais geral do conhecimento. Princípios dessa teoria embasam o que denomina "ambiente LOGO" e "uma filosofia LOGO". Focaliza a criança como construtura ativa de suas próprias estruturas intelectuais, a partir de sua interação com o ambiente; enfatiza a importância de sua participação ativa,

num processo de elaboração livre, em que é dada vazão à sua curiosidade e raciocínio.

Reversamente, psicólogos construtivistas interessam–se pela perspectiva de intercâmbio com correntes cognitivistas da Inteligência Artificial. Nos termos de ACKERMANN–VALLADÃO (1983) "o ambiente informático LOGO promete constituir, ao prolongar as nossas situações experimentais atuais, um novo universo propício, oferecendo indefinidamente ao sujeito a possibilidade de ser noviço, e ao experimentador a vantagem de explicitação", (pp. 68), dos passos do sujeito em cada instante de sua pesquisa, permitindo–lhe seguir e decodificar as diferentes etapas dessa atividade, e as representações que as guiam.

Contudo, essa possibilidade de troca nem sempre foi vista de forma irrestrita. Ao contrário, divergências tornaram–se acentuadas, na década de 70, quando surgiram dúvidas acerca de conceitos integrantes do bojo explicativo da epistemologia genética. MOORE e NEWELL (1974) questionaram, então, o caráter científico do construtivismo, argumentando que os procedimentos descritos não constituiam procedimentos efetivos (o que pode ser compreendido como não sendo programáveis). Críticas a Piaget "dos estágios" foram insistentemente colocadas, por diversos pesquisadores, entre os quais PAPERT (1980).

CELLERIER (1979a, 1979b) analisa o enfoque dessas duas linhas de estudo, procurando definir a extensão dessa divergência. Como ciência cognitiva, considera o agrupamento formado por: inteligência artificial; simulação de processos cognitivos; e processamento de informação. Supõe que as duas abordagens divergem quanto a seus objetivos teóricos e suas epistemologias. Coloca na questão da relação entre conhecimento e ação, o aspecto central dessa divergência. O objeto de estudo da psicologia genética é designado por Cellerier de "transformação epistêmica": a construção do conhecimento a partir da ação, isto é, a reconstituição de etapas intermediárias de construção, e a inferência de mecanismos de transformação da ação em conhecimento. Por outro lado, Cellerier considera que o cognitivismo é orientado em direção ao estudo do efetuável e da ação, mas numa direção recíproca daquela da psicologia genética: designa o objeto do cognitivismo de "transformação pragmática" do conhecimento em ação. Coloca o esforço do construtivismo na explicação do estado atual de uma noção, com base em seus estados precedentes, na seqüência psicogenética. Reversamente, o esforço da ciência cognitiva, segundo esse autor, é colocado na composição de programas, que simulam a aplicação do conhecimento em ação, a partir da preocupação com: de que forma ações são controladas por uma representação interna do universo,

possuída pelo organismo (isto é, por suas estruturas de conhecimento).

Essas duas abordagens, aparentemente contraditórias, são, no entanto, focalizadas por CELLERIER (1979a) como complementares. Coloca que, se a epistemologia e a psicologia genética preocupam–se essencialmente com a aquisição dos conhecimentos, enquanto que o cognitivismo preocupa–se com sua aplicação, os dois "sujeitos" dessas teorias, o sujeito epistêmico e seu correspondente pragmático, se fundem no sujeito psicológico individual. Essa fusão ocorre uma vez que toda atividade cognitiva comporta uma aplicação ("assimiladora") de conhecimentos anteriores que vêm organizar a própria ação; e toda aplicação de conhecimentos precedentes, a uma situação particular, comporta uma reconstituição dessas particularidades, em termos do conhecido (aquisição "acomodadora"). A aplicação do conhecido é uma atividade que se desenvolve no tempo, e numa seqüência de aproximações que convergem para o "equilíbrio" entre assimilação e acomodação. Essa seqüência ocorre num processo cíclico de interação reguladora entre conhecimento e ação. Em síntese a aquisição do conhecimento é concomitante à sua aplicação.

A colocação de CELLERIER (1979a, 1979b) revela uma evolução ocorrida dentro da psicologia genética, a partir de um impasse inicial. De fato essa disciplina passou a incorporar modelos da ciência cognitiva, a fim de estudar tanto a construção de conhecimentos quanto sua atualização em ações. A abordagem atual da Escola de Genêve é o estudo das estratégias e das heurísticas. Ao aspecto estrutural é acrescentado conhecimento mais funcionalista, das estratégias de resolução de problemas.

A ciência cognitiva, por sua vez, parece também ter buscado um enriquecimento, a partir do que parecia uma divergência irreconciliável. Supõe–se que estudos da Inteligência Artificial, desenvolvidos na década de 80, aproximam–se dos objetivos de estudo da psicologia genética. A pesquisa nessa disciplina começa a buscar a substituição, em situações específicas, de sistemas de computação baseados em derivação formal, por sistemas capazes de atingir resultados imprevistos, obtidos através de caminhos não pré–determinados. Cada vez mais, a Inteligência Artificial passa a focalizar processos de descoberta e invenção, a se preocupar em compreender como o conhecimento é construído e adquirido, e não simplesmente aplicado. Nessa linha podem ser citados trabalhos como os de KANT e NEWELL (1984), STEINER e KANT (1985), KANT (1985), ADELSON e SOLOWAY (1985) e DOUGLAS LENAT (1982, 1983a, 1983b,

1984).

LENAT (1982, 1983a, 1983b, 1984) preocupa–se com o processo de construção de regras heurísticas, que guiam a aquisição de novos conhecimentos. Apresenta um sistema que simula a aquisição de conhecimentos por sujeito "noviço". Embora esse autor não faça nenhuma menção explícita nesse sentido, suas colocações teóricas parecem fortemente inspiradas na epistemologia e psicologia genética.

O trabalho apresentado parte do pressuposto que a aproximação entre inteligência artificial e psicologia genética pode resultar em um desenvolvimento nas duas disciplinas. Uma teoria cujo objeto de estudo genuíno é a construção e não apenas a aplicação do conhecimento, parece fornecer fundamentação, particularmente adequada, a pesquisas sobre mecanização de processos de descoberta. Reversamente, um modêlo mais rico que a lógica, e um contexto experimental propício à observação da aquisição do conhecimento podem ser encontrados pela psicologia genética.

Essa troca interdisciplinar resultante de interesses complementares pode ser mais ou menos sequencial, através de passos subseqüentes: uma ciência incorporando conhecimentos adquiridos pela outra, e utilizando–os segundo propósitos específicos.

Mas pode também ocorrer num passo único, se a construção de conhecimento do sujeito cognoscente da inteligência artificial – máquina – e do sujeito cognoscente da psicologia – homem – forem focalizadas enquanto resultantes de um processo interdependente. Especificamente, o que se propõe é a construção de sistema que adquire conhecimento, enquanto, progressivamente, capacita o sujeito com quem interage a propiciar–lhe novas aquisições. Isto é, enquanto, por sua vez, favorece o desenvolvimento do indivíduo.

A fim de desenvolver as suposições teóricas, de torná–las mais explícitas e de exemplificar concretamente, buscou–se nesse trabalho, uma primeira aplicação: um protótipo de sistema, que faz descobertas sobre um procedimento qualquer, definido em Linguagem LOGO, à medida que o sujeito constrói seu conhecimento.

A situação de interação proposta visa propiciar a aprendizagem gradativa (do sujeito e da máquina), que resulta da ação, da própria interação, e da posterior reflexão; favorecer construções e reconstruções progressivas que incluam e superem

aquisições anteriores, preparem e viabilizem posteriores.

Procura–se oferecer ao sujeito a possibilidade de refletir sobre resultados da própria ação e de, ao mesmo tempo, verificar um trabalho complementar ao seu: diferentes descobertas que podem ser feitas, a partir de suas próprias idéias.

A máquina é direcionada a gradativamente capacitar o sujeito a construir novos conhecimentos. Para então torná–lo apto a propiciar sua própria aprendizagem: seja pela extração de regras que guiem a ação do sujeito, com maior ou menor grau de conscientização, seja pela reconstrução de seu conhecimento a partir de resistências e contra–provas impostas externamente.

Conceitos integrantes do bojo explicativo de teoria piagetiana fornecem suporte ao modelo de processo de aprendizagem desenvolvido.

CAPÍTULO II

REVISÃO DA LITERATURA

O levantamento bibliográfico, que é apresentado a seguir, foi direcionado com o objetivo de buscar–se conhecimento sobre: em que medida pesquisas desenvolvidas pela psicologia genética, e pela inteligência artificial, vêm aproximando os enfoques dessas duas disciplinas; até que ponto os conhecimentos adquiridos numa área vem sendo incorporados pela outra.

De uma forma geral, verifica–se que a mudança de rumos na Escola de Genève conduziu à adoção de um quadro referencial epistemológico duplo, em que:

– de um lado, são considerados aspectos propostos pela ciência cognitiva em particular relativos a modêlos de inteligência artificial;

– de outro são focalizados conceitos explicativos da teoria piagetiana relativos a mecanismos de formação de conhecimentos a partir da ação, como abstração, equilibração, etc.

No primeiro ítem desse capítulo faz–se uma rápida revisão de trabalhos que mostram essa evolução.

A nova preocupação da inteligência artificial, com aprendizagem da máquina em sentido amplo, conduz seus pesquisadores a focalizar a construção do conhecimento, e não apenas sua aplicação. De uma forma geral, verifica–se uma convergência quanto a consideração da necessidade de maior conhecimento, sobre processos mentais humanos. Formulações teóricas são apresentadas, tanto relativas à cognização humana, quanto numa perspectiva em que o sujeito cognoscente é a máquina.

O segundo ítem desse capítulo apresenta alguns resultados, propostas e conclusões de pesquisadores que buscam a mecanização de processos de descoberta.

A incorporação de conhecimentos adquiridos pela epistemologia e psicologia genética, em programas computacionais, não parece ser frequente. De fato, em levantamento apresentado por SMITH (1985), sobre teorias de psicologia que vem fornecendo suporte à inteligência artificial, a teoria piagetiana não é sequer mencionada.

No trabalho de LENAT (1982, 1983a), no entanto, verificam–se colocações que parecem baseadas em conceitos integrantes do bojo explicativo da epistemologia e psicologia genética. Afirmações do autor, relativas a processos de aprendizagem, humano ou de máquina, que conduzem a essa suposição, são vistas em vários trechos, como: "a aprendizagem pode assumir muitas formas, dependendo de quem é o "professor" de quão ativo é o papel do aprendiz, o que ele deve fazer para adquirir, assimilar, acomodar o novo conhecimento, etc... Se o professor é a natureza, o aprendiz deve ter um papel mais ativo..." (LENNAT, 1983a, pp. 32).

Ou ainda, quando descreve o processo de aprendizagem, em que um corpo de meta–heurísticas faz com que o sistema: funcione em estado de equilíbrio enquanto as heurísticas correntes são adequadas; descubra quando elas se tornam menos efetivas; pesquise novas heurísticas, a fim de readquirir seu estado de equilíbrio. "Gradualmente o nível dos objetos evolui: novos conceitos são encontrados e focalizados (...) e a medida que isso ocorre, as antigas representações do conhecimento e o antigo conjunto de guia heurístico tornam–se menos ideais, menos efetivos. Isso, por sua vez, é detectado por alguma das heurísticas de heurísticas (...). Elas fazem com que o sistema tente recuperar seu equilíbrio, para mais uma vez tratar efetivamente com objetos e operações no nível do objeto". (LENAT, 1982, pp. 205). Assim, de tempos em tempos, o sistema volta–se "para o problema de encontrar novas heurísticas (talvez pela abstração de recentes experiências no domínio da tarefa). O descobridor, mais tarde, retoma sua tarefa original, armado com heurísticas novas e mais poderosas" (LENAT, 1982, pp. 205).

Mas nenhuma menção explícita é feita pelo autor[*1], sobre o referencial teórico, que embasa suas colocações.

Uma crítica à metodologia da inteligência artificial, e especificamente ao trabalho de LENAT (1976, 1977, 1979, In: RITCHIE e HANNA, 1984) é feita por RITHIE e HANNA (1984). Esses autores abordam, entre outros aspectos, a fraca correspondência entre conhecimento e base teórica, e descrição de performance de máquina.

[*1] A única menção feita por Lenat (LENAT e BROWN, 1984), a Piaget, refere–se à compilação por seu sistema AM (LENAT, 1983b) de conceitos pré–numéricos, descritos por Piaget na aquisição da noção de número.

A revisão de SMITH (1984) e a reflexão de RITCHIE e HANNA (1984), sobre conhecimento teórico e implementação de sistemas, são apresentados no terceiro ítem.

II.1 – PSICOLOGIA GENÉTICA: NOVO DIRECIONAMENTO

Por muitos anos a principal tarefa da Escola de Genève foi elucidar o desenvolvimento ontogenético de categorias adaptativas (aquisição de conceitos de número, tempo, etc.). A partir da década de 70, as pesquisas se distinguem das desenvolvidas inicialmente numa perspectiva estrutural. Entretanto, INHELDER (1979) considera que as pesquisas estruturais constituíram uma etapa fundamental e necessária, e que a evolução da pesquisa em psicologia genética permanece conciliável com o quadro teórico piagetiano. Com o desenvolvimento de pesquisas longitudinais, o trabalho dessa autora é orientado para a abordagem direta de problemas de transição, em ação no sujeito em adaptação. Coloca em evidência uma sucessão de etapas na passagem de uma estrutura a outra, discute o papel dos conflitos, e mostra a importância das características da situação enquanto desencadeadora deles. Mecanismos de desenvolvimento como contradição, equilibração, abstração reflexiva, tornam–se o alvo de estudos.

Essa ampliação de perspectiva já se manifestara na década de 50, com o deslocamento da tônica, das descrições das etapas, em direção aos caminhos que asseguraram a transição entre elas. A preocupação passou então a ser com processos evolutivos, isto é, com processos efetivos das evoluções individuais. Tratava–se de estudar a modificação dos conhecimentos, mas não somente na sua formação no curso de gênese, como também na sua atualização nos contextos particulares.

BROWN e WEISS (1987) analisam o novo direcionamento ocorrido na Escola de Geneve. A mudança de interesse e as críticas desencadeadas pelos cientistas cognitivistas geraram, nos termos desses autores, dúvidas sobre a existência de estruturas gerais e sobre a utilidade de análises estruturais. A nova preocupação tornou–se primeiramente evidente quando BÄRBEL INHELDER e colaboradores (1976), desviou–se do programa usual da epistemologia genética na direção de procedimentos de invenção e descoberta (BROWN e WEISS, 1987). "Enquanto estudos estruturais focalizam o mecanismo de compreensão da **realidade** e conseqüentemente, a **construção** de estruturas gerais, chegou a hora de haver maior interesse nos processos de **invenção** ou **descoberta** do sujeito, como ele

pesquisa por soluções para problemas particulares, bem diferenciados. Invenção, como uma função de problemas particulares, sempre consiste de um conjunto de procedecimentos individuais, que pode variar de um para outro sujeito e de uma situação para outra..." (INHELDER e colaboradores, 1976, pp. 58–59).

Em 1979, a distinção entre procedimentos e estrutura tornou–se um tema de pesquisa em Genève. Da dualidade dialética entre estruturas e procedimentos, emerge a importância de aspectos funcionais do pensamento, e dos contextos nos quais ele se elabora. INHELDER e PIAGET (1979) concluem que ..."toda estrutura é de fato, o resultado de construções procedurais, como cada procedimento utiliza algum aspecto ou outro de estruturas" (p. 173, 174), embora possuam diferenças fundamentais. Procedimentos empregam tansformações a fim de atingir metas particulares e variáveis, enquanto estruturas usam–nas apenas para compreender situações.

A mudança de enfoque implica na passagem de uma psicologia dos processos universais, a uma psicologia de processos individuais. Do estudo do sujeito epistêmico – o que há de "comum" às estruturas do conhecimento dos sujeitos do mesmo nível" – para o estudo do sujeito psicológico – o que há de comum aos sujeitos individuais.

Ao aspecto estrutural é acrescentado conhecimento mais funcionalista, das estratégias de resolução de problemas, definidos por INHELDER (1979) como "sistemas e seqüências de procedimentos finalizados, repetitíveis e transferíveis, constituindo os meios para atingir o fim visado pelo sujeito... O objetivo de nossos estudos atuais é a pesquisa dos mecanismos funcionais em jogo, nas estratégias próprias a cada sujeito individual, nos diversos níveis de seu desenvolvimento" (pp. 102).

CELLERIER (1979b) conclui que a resolução de problemas é particularmente adequada ao estudo das relações entre conhecimento e ação, e procura caracterizá–la sob esse enfoque teórico. Descreve a compreensão do enunciado de um problema como sua assimilação às estruturas cognitivas do sujeito; a realização da tarefa como sua assimilação e seus esquemas de ação. Acomodações, por sua vez, são requeridas: da estrutura cognitiva às particularidades da situação, com a construção de um quadro cognitivo (que guiará a atribuição de uma determinada interpretação dos componentes do problema); dos esquemas de ação, com a construção de um procedimento de ação especializado ao cumprimento da tarefa.

A abordagem atual da Escola de Genêve é o estudo das estratégias e das heurísticas. Qualquer... "construção matemática comporta estruturas, que devem ser usadas ou inventadas, mas, ao mesmo tempo, tal utilização ou invenção pressupõe procedimentos que POLYA (1945) estudou sob o nome de "heurísticas". No outro extremo qualquer estratégia adotada pela criança, a fim de resolver um problema de inteligência prática, envolve procedimentos, mas esses procedimentos necessariamente fazem apelo a conhecimentos estruturais já adquiridos ou que devem ser descobertos no percurso" (INHELDER e PIAGET, 1979, pp. 165).

ACKERMANN–VALLADÃO e outros (1983), colaboradores de Inhelder, destacam alguns aspectos analisados nos estudos conduzidos por esse grupo de pesquisadores, sobre o tema das estratégias de resolução de problemas pela criança. A interrelação entre os estudos desenvolvidos em Geneve e pelos cognitivistas, é ainda focalizada.

Essas pesquisas visam colocar em evidência vários aspectos dos modelos, isto é, das representações específicas construídas pelo sujeito, e pertinentes a uma situação. Esses aspectos incluem:

– sua formação progressiva e sua utilização no curso da resolução;

– a cada etapa dessa formação e utilização, sua natureza estrutural (conteúdo e organização interna) e os mecanismos funcionais em jogo.

Segundo o grau de formação (estruturação) do modelo e o grau de sua adequação (aplicabilidade) à tarefa, o sujeito pode, em situação experimental, apresentar condutas e se encontrar em estados psicológicos diferentes.

Esse grupo de pesquisadores propõe–se a analisar o conjunto dessas condutas e estados, considerando os quatro seguintes casos possíveis:

QUADRO 1 – Representações e condutas analisadas por VALLADÃO e outros (1983)

GRAU DE ESTRUTURAÇÃO: Grau de Aplicabilidade	Modelo Formado	Modelo Transitório ou em Formação
Modelo adequado	1	3
Modelo não–adequado	2	4

Caso 1– Corresponde ao que os cognitivistas chamam de "especialistas". Na teoria piagetiana a idéia de especialista corresponde de um lado àquele do sujeito epistêmico que atingiu o nível cognitivo caracterizado pelo fechamento da estrutura (pertinente à tarefa), e de outra parte, em termos mais funcionais, à etapa γ de equilibração.

O conjunto de suas condutas pode ser interpretado como: as representações do sujeito apresentam um boa diferenciação e coordenação dos meios e dos fins, o que lhe permite planificar sua atividade de forma ótima. O modêlo exerce um controle do tipo descendente sobre as condutas de resolução. Não há real problema para o sujeito.

Caso 2– O sujeito dispõe de modelo elaborado em situações anteriores (para as quais era pertinente), mas que não leva à solução do problema na situação presente. A forte organização do modelo (há diferenciação e coordenação entre meios e fins, mas não são funcionais na situação) resulta numa predominância de controle do tipo descendente (o sujeito se convence da pertinência de suas representações), levando do modelo à conduta (apesar dos "feed–back" negativos recebidos a partir da ação sobre os objetos). Mas ao compreender o impasse, adota uma das possibilidades:

- forma pouco a pouco (sobre a base de seu modelo atual, ou depois de abandoná–lo) um novo modêlo que seja adequado à tarefa;
- ou abandona seu modelo, substituindo por outro modelo bem formado (adequado ou não);
- ou modifica o modelo presente de foma a torná–lo adequado à sua tarefa (se consegue, passa a ser caracterísico do caso 1).

Diferentes heurísticas e mecanismos, tornam possível a modificação do modelo inicial.

Algumas heurísticas consistem em pesquisar, mais ou menos explicitamente, as causas de discrepância entre efeitos esperados e obtidos, por aplicação cíclica do modelo sobre diferentes elementos da situação. Uma heurística tem o objetivo de utilizar o objeto como suporte externo para destacar as propriedades, relações, configurações, que o sujeito é capaz de reconhecer. Outra heurística tem o objetivo de avaliar os próprios procedimentos de realização ou seu encadeamento no modelo.

Ao invés dessas heurísticas intencionais, pode intervir um processo em que, um novo quadro de representação é invocado, quando após um resultado não esperado da ação sobre o objeto, uma nova significação é atribuída à ação ou ao objeto.

Estas heurísticas constituem um conjunto de condutas que implicam num controle do tipo descendente, conduzindo das representações do sujeito em direção à sua atividade. Mas esse controle tem por fim produzir informações ascendentes, permitindo modificações no nível representativo.

Na teoria piagetiana, caracteriza–se os comportamentos descritos em 2 como de "sujeitos em estágios intermediários", ou como correspondendo a etapas α de equilibração (quando o sujeito, numa 1ª fase, resiste a modificar seu modelo), ou β (quando a modificação está em curso).

Segundo ACKERMANN VALLADÃO e colaboradores (1983) os cognitivistas "não estudam geralmente esses comportamentos, característicos dos sujeitos ditos noviços, na medida em que seus comportamentos são difíceis de simular ou de representar por um modelo formal" (pg. 65).

Caso 3– Modelo em formação – significa em nível de organização que o sujeito possui numerosas representações específicas à situação, mas que elas não são articuladas entre sí, (são potencialmente funcionais, mas não ainda verdadeiramente significantes para o sujeito). Ou que o sujeito dispõe de uma ou várias representações gerais que não são nem diferenciadas, nem especificadas em função da tarefa (mas guiam o trabalho numa direção adequada).

O sujeito persegue a construção, sem abandonar nem modificar radicalmente o modelo. Esse trabalho de elaboração se traduz por um conjunto de condutas exploratórias que vão permitir integrar os elementos (objetos, ações, procedimentos), necessários ao sucesso da tarefa.

Nas condutas exploratórias predomina o controle de tipo ascendente, cujo fim é perseguir a formação do modelo, que exerce, de volta, um controle descendente sobre as ações do sujeito.

Corresponde na teoria piagetiana às condutas de nível intermediário, e à etapa β de equilibração.

Nas teorias cognitivas, trata–se de sujeitos noviços.

Caso 4– Em Psicologia Genética trata–se de sujeitos de estágios iniciais, que não possuem estrutura necessária para assimilar a situação, isto é, nenhum meio adequado para resolver o problema. Eles o transformam em um outro, que lhes é accessível, em função dos meios cognitivos de que dispõem. Em termos funcionais trata–se da conduta α de equilibração, isto é, conduta de rejeição sobre o problema em seu conjunto.

O sujeito pode:

– formular a tarefa em termos ou ações significativas para ele;

– testar sistematicamente certas propriedades dos objetos e das ações;

– verificar, efeitos e limites dos pontos de vista (exercitar), a fim de lhes dar o "status" de meio com relação ao fim;

– transgredir (voluntariamente) a tarefa para construir meios pertinentes, por eliminação dos que não o são.

A autora destaca ainda, que esse caso corresponde, na terminologia de Newell e Simon, aos sujeitos que não possuem um nível cognitivo suficiente para a dificuldade do problema, e é de saída eliminado de estudos de cognitivistas.

As quatro correspondências entre o grau de formação das representações do

sujeito e sua aplicabilidade à situação, servem não para classificação, mas para caracterizar as diferenças e estados psicológicos que podem se apresentar, num mesmo sujeito, no curso da microgenese na solução de problemas. Essa microgênese supõe um caminhamento dinâmico através do problema, representável por uma livre passagem entre os casos. As representações específicas construídas pelo sujeito desempenham as funções psicológicas: organizar a informação disponível (servindo de quadro assimilador e permitindo atribuir significações aos elementos da situação e às ações do sujeito); guiar e controlar a atividade cognitiva.

II.2 – INTELIGÊNCIA ARTIFICIAL: MECANIZAÇÃO DE PROCESSOS DE DESCOBERTA

O interesse pela mecanização de processos criativos vem direcionando o desenvolvimento de pesquisas na inteligência artificial. Algumas propostas, idéias e conclusões de autores, que se preocupam com essa linha de estudo são apresentadas nesse sub–capítulo.

II.2.1 – Proposta de Kant, Newell e Steiner

Alguns trabalhos desses autores (KANT, 1985; KANT e NEWELL, 1984; STEINER e KANT, 1985) referem–se a sistemas especialistas capazes de construir algoritmos ou assistir usuários nessa tarefa. Partem da suposição que alguns aspectos da construção de "software" são pouco compreendidos, que a observação de sujeitos, engajados nessa tarefa, constitui um passo prévio útil ao desenvolvimento de sistemas desse tipo.

Idéias teóricas sobre processos mentais envolvidos na construção de algoritmos são apresentadas, com base na observação de dois sujeitos submetidos à essa tarefa para resolução de problema específico. Considerando a limitação do estudo, e a necessidade de observações adicionais, concluem que esse conhecimento pode ser incorporado a sistemas, tais como DESIGNER implementado pelos autores numa primeira versão. Uma síntese das colocações feitas é apresentada a seguir.

KANT e NEWELL (1984) interessam–se pela investigação de estratégias utilizadas na resolução de problemas e justificam esse interesse pelas razões descritas a seguir:

(i) o estudo de como as pessoas desenvolvem algoritmos complexos pode ser uma abordagem adequada, para obter–se conhecimento sobre como mecanizar etapas mais difíceis de construção de projeto de "software";

(ii) a construção de sistemas com organizações similares a do ser humano permite a utilização de técnicas humanas como fonte, para o sistema ser iniciado e examinado a medida em que evolui;

(iii) uma vez que a organização humana permite contínua adaptação e desenvolvimento, essa abordagem pode conduzir à construção de sistemas que aprendem alguns princípios por si próprios;

(iv) esse conhecimento pode ser necessário à construção de programas capazes de seguir sugestões para executar pequenas tarefas, dar conselhos a programadores com alguma experiência ou ensinar a novatos.

KANT e NEWELL (1984) consideram que a partir do estudo de técnicas de resolução de problemas empregadas na construção de algoritmos pode ser obtido conhecimento necessário à mecanização dessa tarefa. Relatam ter verificado que um pequeno número de operadores, de propósito geral, constrói e modifica as representações; estes operadores são adaptados ao estado do problema corrente por análise meio–fim. A execução simbólica mostrou–se um método particularmente versátil sendo usada para refinar ou explicar componentes do algoritmo. Além do espaço principal do problema, os sujeitos também trabalharam no espaço geométrico de domínio da tarefa. Os autores consideram que o intercâmbio entre soluções de problemas nos dois espaços torna possível o processo de descoberta.

Esses autores relatam ter iniciado a implementação de um sistema que recriará os algoritmos projetados por dois sujeitos. Continuam estudando protocolos de indivíduos e supõem ser necessário, ao desenvolvimento do sistema, a adição de novos algoritmos e de conhecimento de controle de pesquisa baseado nesses estudos.

KANT (1985) levanta a importância da reflexão a partir da própria ação ("... você deve observar seu próprio comportamento na resolução de um problema...", pg. 1361), e procura resposta a indagações investigando a ação humana.

Define os seguintes passos componentes do processo de construção de algoritmo

(i) compreender o problema;

(ii) planejar uma solução em torno de uma idéia central, e refiná–la ou elaborá–la;

(iii) notificar e formular dificuldades ou oportunidades;

(iv) verificar se a estrutura é uma solução (atende à especificação);

(v) avaliar a solução.

A descrição desses passos apresentada pela autora é sintetizada a seguir.

(i) Compreensão do problema envolveu, nesse estudo, a construção de figuras que auxiliavam na observação e no raciocínio sobre propriedades de objetos.

(ii) A escolha de um plano de solução, em torno de uma iéia central, segue–se à especificação. Os passos básicos dessa idéia são seguidos, a menos que ela se mostre completamente inadequada. Um conhecimento anterior pode ser utilizado (o projetista sabe o que faz e conhece as implicações), ou pesquisa pode ser requerida.

(iii) Duas estratégias distintas foram observadas pela autora, para o registro de dificuldades e oportunidades. Uma delas baseada em dados concretos (execução de teste de caso); outra baseada em exemplares simbólicos. Registros de dificuldades (passos ausentes, inconsistências entre partes, etc.) ocorreram tanto na descrição abstrata do algoritmo, quanto a partir de exemplos de figuras construídas para auxiliar no raciocínio, acerca do domínio da tarefa. KANT (1985) relata que todas as "descobertas" ocorreram quando o projetista, baseado numa figura exemplo, reconheceu uma propriedade geométrica ou a possibilidade de utilizar um passo chave de outro algoritmo. A observação de algo não visto anteriormente, e sua incorporação ao algoritmo, resolvia uma meta importante. Essa meta não era a que preocupava o projetista, mas tinha uma relação com ela. A autora considera, que tanto observações

chave no domínio do problema, quanto o conhecimento de técnicas de construção de algoritmo são importantes.

(iv) Verificação de correção foi observada através de testagem com exemplos específicos. Um algoritmo para caso geral pode ser modificado, nesse passo, para incluir exceções.

(v) Avaliação de solução pode ser baseada em conhecimento específico sobre princípios de projeto de algoritmo, ou em análise de custo.

KANT (1985) define o comportamento, na resolução de problema, como uma pesquisa no espaço do problema, com conhecimento utilizado para limitá–la. Nos termos da autora, espaço do problema é um conjunto de estados para descrever problemas e soluções parciais. Operadores podem ser aplicados a um estado, para produzir novo estado. Em problemas difíceis, o solucionador pode não ter conhecimento suficiente para encaminhar–se diretamente ao estado de solução. Pode tentar vários passos, alguns dos quais conduzem a fins mortos, e retornar a estados iniciais. Frequentemente o solucionador não possui conhecimento de controle de pesquisa, que guie a seleção de operadores a serem aplicados. Em síntese, o solucionador gradualmente explora o espaço, adquire conhecimento sobre ele e eventualmente dá um passo apropriado.

A partir dessa visão KANT (1985) procura especificar onde se localizam fontes de poder para resolução de problemas, no projeto de algoritmo. As idéias da autora sobre essas fontes são sintetizadas a seguir.

(i) O poder da pesquisa – pesquisa supre conhecimento ausente e não pode nunca ser completamente eliminada, mas pode ser desejável limitá–la.

(ii) O poder de múltiplos espaços do problema – a autora observou os projetistas trabalharem em quatro espaços. Os dois principais são espaços de projeto de algoritmo (conhecimento do que é acessível em sistemas de computação e de princípios de projeto de algoritmo), e espaço de domínio de aplicação. Além desses, espaços de execução de algoritmo (uma extensão do primeiro, que tem novos objetos e operadores), e espaço de geração de exemplo (extensão do segundo, no qual figuras são usadas como exemplos padrão, contra–exemplos, etc. e que tem novos operadores).

(iii) O poder do reconhecimento – considera a maior fonte de poder em resolução de problema, a capacidade de reconhecer objetos e a aplicabilidade de operadores. O processo de pesquisa não é dirigido por um algoritmo, que seleciona elementos do contexto em uma ordem fixa, mas, ao contrário, pela observação de quando algum elemento do contexto deve mudar. As condições para recognição podem ser simbólicas, no espaço de projeto de algoritmo, ou imagens visuais do espaço do domínio. Considera um exemplo de recognição o registro de propriedades de exemplos gerados.

(iv) O poder do conhecimento de execução – execução por teste, em projeto de algoritmo, serve ao propósito de controlar a ordem do processo de refinamento, de limitar as inferências feitas e de tirar erros ("debugar").

(v) O poder do conhecimento de eficiência – conhecimento de eficiência está geralmente contido no espaço do projeto de algoritmo, mas pode também ser específico ao domínio de aplicação particular. Além disso, algumas informações sobre custo de operação e algumas heurísticas sobre melhores princípios de projeto, ou escolhas de implementação, são construídas sobre um modelo de custo para a arquitetura visada.

A autora relata ter implementado uma primeira versão do sistema DESIGNER, a fim de testar idéias descritas nesse artigo (1985).

STEINER e KANT (1985) procuram descrever como pessoas constrõem projetos de software. Consideram alguns recursos, que são utiliados para guiar o desenvolvimento de projetos, parcialmente completos:

(i) execução simbólica – produção de expressões simbólicas, para cada saída de um programa, em termos de entradas, por interpretação dinâmica;

(ii) avaliação simbólica – produção de expressões simbólicas para saída de programas, por análise estática;

(iii) meta–avaliação – análise estática de especificações de alto nível, para resolver ambigüidades;

(iv) análise partilhada – procedimento para executar a especificação e a implementação de um módulo, e comparar os resultados;

(v) simulação de modelos mentais – processos para detectar interações inesperadas entre novos elementos em um plano ou projeto.

STEINER e KANT (1985) supõem que se são conhecidos operadores que produzem apenas configurações úteis e bem formadas, eles devem, evidentemente, ser utilizados. Mas que, se não é óbvio que transformações se aplicam melhor a um projeto parcial, então um sistema baseado em técnicas de derivação formal não fará muito progresso. Relatam que os sujeitos observados frequentemente tinham idéias novas, fazendo a coisa correta sem uma compreensão completa da razão. Afirmam que não desejam impedir que seu sistema DESIGNER faça descobertas similares, que chamam de acidentais.

II.2.2 – Proposta de Adelson e Soloway

Numa mesma linha de estudo, ADELSON e SOLOWAY (1985) preocupam–se em descrever o comportamento de indivíduos engajados na tarefa de construção de "software".

Discutem a utilidade funcional, isto é, a contribuição para a execução da tarefa, dos seguintes comportamentos descritos abaixo:

(i) formação de modelos mentais – permite que o projetista simule seu projeto em progresso;

(ii) simulação – possibilita a integração de material familiar, de forma modificada, e a checagem do comportamento de um mecanismo parcialmente especificado, em relação ao comportamento do estado a ser atingido;

(iii) expansão sistemática – permite que a simulação funcione regularmente, garantindo que a entrada de todos os elementos seja no mesmo nível de detalhe;

(iv) representação de restrições – o projetista necessita explicitar restrições, quando contrói um modelo mental que é não familiar;

(v) recuperação de nomes para planos – nomes usados para indicar que existe um plano para uma parte do problema, resolvida previamente, que pode se usada no projeto

(vi) notas – permite ao projetista manter sua expansão automática, sem esquecer–se de aspectos importantes que não estão ainda no corrente nível de detalhe.

Os autores comparam o desempenho de especialistas ao de novatos e relatam que esses comportamentos podem ser afetados pela experiência do projetista.

II.2.3 – Proposta de Douglas Lenat

DOUGLAS LENAT (1982) buscou a construção de programas cuja especialização é a própria extração, modificação, formulação e descoberta de heurísticas.

LENAT (1983a) analisa motivos pelos quais a Inteligência Artificial deve preocupar–se com programas que aprendem por descoberta.

1. Interesse da Inteligência Artificial em mecanizar qualquer atividade cognitiva humana.

2. A abordagem padrão para construção de sistemas especialistas envolve a extração explícita de regras de avaliação de conhecimento, de especialistas humanos, e:

   2.1 – suas regras conscientes podem não ser suficientes para dar conta do comportamento científico criativo;

   2.2 – o campo pode ser jovem, sem especialistas humanos, sem regras heurísticas para guiar sua exploração.

Ambos os casos, 2.1 e 2.2, constituem problemas difíceis na construção de sistemas especialistas, embora o primeiro deles possa ser contornado de alguma forma (o engenheiro de conhecimento pode extrair regras de conhecimento adicionais usadas pelo especialista, através de técnicas, como confrontar decisões inconsistentes com o conjunto de regras afirmadas).

O autor distingue o processo de descoberta – ascendente, de um conceito mais simples a um mais complexo – do processo presente no trabalho de axiomatizadores e pedagogos – descendente, do conceito mais complexo ao mais simples, analítico. Considera que, uma vez feita uma descoberta, é muito mais fácil racionalizá–la, achar alguns passos descendentes, dela a conceitos conhecidos, do que foi fazer a descoberta inicial.

Supõe que o mundo é muito complexo para ser modelado profundamente, de qualquer maneira formal; mas que um corpo de heurísticas crescendo dinamicamente pode ser suficiente. Indução, usando um modelo profundo, é precisamente o que o autor entende por formação de teoria. Heurísticas consistem de guias necessários para lidar com o mundo, elas podem crescer e melhorar gradualmente. Mesmo esse tipo de modelo é difícil de construir, na medida em que heurísticas não são facilmente extraídas de especialistas (nos campos em que eles existem).

LENAT (1982) lembra que Polya advocou o estudo de heurísticas como uma disciplina científica separada, há 40 anos. Nos termos do autor, há uma década atrás Pospelow e Pushkin tentaram definir o campo como ciência que estuda as leis que governam o planejamento de novas ações em novas situações.

A definição inicial de LENAT (1982) para heurísticas é: a peça do conhecimento capaz de sugerir ações, plausíeis a seguir ou implausíveis a evitar. Posteriormente acrescenta que cada heurística deve especificar uma situação ou contexto no qual suas ações são especialmente apropriadas (parte IF e parte THEN). E, além disso, estende a definição, operacionalisando–a da forma: uma heurística consiste de grupo de atributos (e valores correspondentes) que inclui muitos tipos de condições (parte IF), muitos tipos de ações (parte THEN), e ainda, vários atributos não executáveis tais como utilidade, origem e tempo médio de execução.

O procedimento experimental proposto por LENAT (1982) para responder questões sobre heurísticas (tais como sua fonte de poder, suas origens, seu impacto quantitativo em uma pesquisa, a interação entre várias heurísticas trabalhando em direção a um mesmo fim, seus atributos úteis, etc.) consiste do "uso do paradigma padrão de inquérito empírico, que domina a Inteligência Artificial, isto é: teste de hipóteses sobre heurísticas, pela construção e estudo de programas de computador, tais como A. M. e EURISKO, que usam heurísticas e tentam encontrar novas"

(pp. 193).

As primeiras tentativas de LENAT (1982) com programas que raciocinavam indutivamente, foram pequenos sistemas que tentavam induzir programas LISP, de coleção de pares de entrada/saída. Essas experiências conduziram–no a concluir que pode–se aprender mais sobre indução, se a tarefa do programa for mais aberta, próxima à formação de teoria, ao invés de solução de problema. Isso conduziu–o ao planejamento e construção do A. M., que explorou conceitos matemáticos elementares. A. M. foi guiado por um corpo de regras heurísticas informais, que ajudaram–no a definir novos objetos e operações úteis, coletar dados, notificar padrões, formar conjecturas, e avaliar as suas descobertas como quão interessantes. Trabalho mais recente, com EURISKO, trata com vários domínios diferentes.

A questão do projeto A. M. foi mostrar que um conjunto, relativamente pequeno, de heurísticas gerais, pode guiar um processo de descoberta não trivial, por caminhos não préconcebidos. Cada atividade, cada tarefa do A. M. (na agenda), era proposta por alguma regra heurística, que era usada de tempos em tempos. EURISKO demonstra que heurísticas previamente fornecidas, em um domínio, podem com sucesso guiar a exploração de domínios novos, diferentes.

A observação da performance de A. M. (na medida em que os conceitos que ele definia afastavam–se dos primitivos, com os quais iniciara, perdia velocidade) levou LENAT (1983a) a concluir que o que as pessoas aprendem é mais do que termos, objetos, operações, resultados, etc. São heurísticas específicas de domínio necessárias para operar eficientemente, com aqueles conceitos. A deficiência chave de A. M. era a falta dessa capacidade. "Em essencia A. M. era um sistema de programação automática cujas ações primitivas eram modificações de peças de códigos LISP, códigos que representavam as funções características de conceitos matemátcos" (LENAT, 1983b, pp. 61).

Para remediar essa situação Lenat concebeu EURISKO. Sua suposição fundamental foi que a tarefa de descobrir e modificar heurísticas novas e úteis é qualitativamente similar à tarefa de A. M. Além disso, o autor supõe que a síntese heurística pode ser executada por um programa como A. M., mas, em adição a "frames" para objetos (como conjuntos de valores de verdade), os "frames" iniciais incluem regras heurísticas. A. M. tinha operadores primitivos como Componha, Ache a Interseção, aos quais EURISKO adicionou novos operadores que trabalhavam com heurísticas: gere, avalie, e modifique–as. Estas meta–heurísticas

servem para propor sínteses e modificações plausíveis e executar, experimentos a tentar, etc. No planejamento de EURISKO, meta–heurísticas não são distinguidas de heurísticas.

LENAT (1982), com base em resultados empíricos de seus sistemas, supõe que novas heurísticas surgem de três fontes: generalização, especialização e analogia.

(i) Especialização de heurísticas mais gerais existentes, tem a forma de adaptar, ligar, emparelhar um modelo a dados observáveis, produzindo um resultado mais especializado, mais eficiente (análogos da ciência, da computação, a esse processo, são compilação e programação estruturada). Uma segunda forma ocorre quando é notada exceção a uma heurística geral (análoga a tal acomodação: "debugging").

(ii) Generalização de heurísticas mais especializadas existentes, pode consistir da abstração a partir de dados observados. Nesse caso a heurística é uma predição sobe APROPRIADO (Ação, Situação) para um total domínio de situações e ações, com base em um ou mais elementos daquele domínio. Outra forma pode ser aplicar fora de um domínio D, um teorema (ou técnica), novo e poderoso, provado no domínio D.

(iii) Analogia pode referir–se a heurísticas existentes, ou a atos passados bem sucedidos de criar novas heurísticas. Mesmo para dois domínios diferentes, heurísticas análogas podem ser igualmente poderosas. Mesmo que heurísticas de dois domínios pareçam discrepantes, os caminhos que foram seguidos, para obter as heurísticas podem ser similares (ex. examine tentativas bem sucedidas e mal sucedidas, de achar provas, e incorpore em novas heurísticas qualquer característica que as discrimine).

O modelo de formação de teoria proposto por LENAT (1983a) que embasou seus programas A. M. e, após revisão (acréscimo de passos 5 e 6), seu sistema EURISKO é apresentado a seguir.

Passo 1: Dados alguns objetos, definições, operações, regras, etc. novos (não totalmente explorados), imediatamente colete dados empíricos sobre eles: ache exemplos deles, tente aplicá–los, etc.

Passo 2: A medida que o passo 1 progride, tente notificar regularidades, modelos e exceções a modelos, nos dados.

Passo 3: A partir dessas observações, forme novas hipóteses, e modifique antigas. Num mundo sobre o qual você tem algum controle, planeje e faça experimentos para testar estas hipóteses.

Passo 4: A medida que um corpo de conjecturas desenvolve–se, economize fazendo novas definições que encurtem a afirmação das conjecturas mais úteis. O inteiro processo cíclico agora começa novamente no passo 1, dando estas novas definições como alimento.

Passo 5: A medida que o loop acima (1–4) procede, é necessário, de tempos em tempos, abstrair algumas heurísticas novas compilando o que foi compreendido pelo aprendiz.

Passo 6: Em algumas ocasiões mais raras, é necessário aumentar ou trocar a representação na qual o conhecimento de domínio é codificado.

Passo 7: Para todos os passos nesse modelo, mesmo passos 5, 6 e 7, é suficiente coletar e usar um corpo de heurísticas, regras julgamentais informais, que guiam o explorador em direção às alternativas mais plausíveis e afastam as mais implausíveis.

LENAT (1983a) considera que dentre as várias suposições desse modelo, a maioria é facilmente satisfeita para aprendizes humanos, mas não tão trivialmente para máquinas.

Passo 1 supõe a habilidade para coletar dados por si mesmo. A maioria dos campos que A. M. e EURISKO explora é internamente formalizável, ou algum subcampo cuidadosamente selecionado de algumas disciplinas, subcampo que admite uma adequada formalização de máquina.

Passo 2 supõe que haja um grande registro de modelos conhecidos, para reconhecer (por observação), ou que o trabalho seja em um mundo, onde um conjunto adequado possa ser aprendido muito rapidamente.

Passo 3 consiste de generalização (de regularidades notificadas), seguido de

especialização (em novas questões específicas e casos que experimentos possam testar). A maioria dos fenômenos, então, deve ser explicável por um pequeno conjunto de leis simples ou regularidades, o conhecimento é obtido por inquérito racional, a causalidade é inviolável, coincidências possuem significado, etc.

Passo 5 supõe que heurísticas possam ser sintetizadas, avaliadas, modificadas, etc. da mesma forma que operações ou objetos de domínio (isso não foi parte do modelo de A. M. mas apenas de EURISKO). Qualquer programa que explore novo território, deve possuir um método concreto para adquirir as novas heurísticas necessárias.

Passo 6 faz suposição análoga para representação do conhecimento: que o programa possa raciocinar sobre, produzir e modificar novas peças de sua própria linguagem de representação (ex. EURISKO define um novo tipo de aspectos para sua linguagem semelhante a "frame"). A síntese e modificação de tanto heurísticas como representações de conhecimento é potencialmente explosiva, e deve ser uma atividade rara.

Passo 7 supõe que um amplo corpo de heurísticas é disponível, pode ser acessado, etc. Que um corpo de heurísticas pode guiar: a descoberta, a avaliação e a modificação de heurísticas; a evolução da representação empregada; a aplicação de heurísticas em cada situação.

LENAT (1983a) adiciona ao modelo, sua convicção de que cada passo envolve raciocínio indutivo, que cada passo pode ser adequadamente modelado como uma pesquisa.

Essa pesquisa toma lugar num imenso espaço de pesquisa (ex.: o espaço de todas as regularidades que podem ser procuradas, de todas novas definições que podem ser feitas, etc.) e as heurísticas servem para restringir a geração e a exploração desses espaços.

O autor supõe que esse modelo seja adequado. Nem os exemplos, nem os programas descritos nas próximas seções, têm o objetivo de "testar" esse modelo, mas apenas de operacionalizar, ilustrar e empregar o modelo. Considera que apenas o sucesso ou falha desse programa de pesquisa tem algo a dizer sobre a adequação desse modelo.

Resultados apresentados por LENAT (1982, 1983a) obtidos por EURISKO

em alguns dominios, são sintetizados a seguir.

(i) Resultados no domínio de jogos

EURISKO explorou o modelo de uma frota naval de acordo com um corpo (várias centenas) de regras. EURISKO planejou uma frota de navios e venceu, tornando–se o campeão dos Estados Unidos (torneio Nacional de Jogos de Guerra Origens). Essa vitória foi mais significativa pelo fato de Lenat nunca ter jogado este jogo, ou similares, segundo o autor.

Foram dadas a EURISKO as regras de jogo e as restrições, e ele gastou uma grande quantidade de tempo manejando um processo de evolução guiado heuristicamente. A batalha simulada foi analisada por algumas heurísticas de EURISKO para determinar: (1) que planejamentos eram vencedores, e (2) ocasionalmente, que circunstâncias fortuitas poderiam ser abstraídas em novas heurísticas de planejamento. O autor (a cada 12 horas de execução) eliminava heurísticas que ele considerava inválidas ou indesejáveis e premiava as que compreendia e gostava. Segundo o autor, nem ele nem EURISKO teriam vencido sozinhos (60% Lenat e 40% EURISKO). Enquanto a maioria das batalhas do torneio levou de 2 a 4 horas, a maioria das partidas com EURISKO era vencida em poucos minutos. As regras para o torneio do ano seguinte foram mudadas para reduzir dramaticamente as singularidades de planejamento que EURISKO e Lenat descobriram.

(ii) Resultados em programação e representação

Umas poucas centenas das funções INTERLISP mais comuns foram representadas como unidades em EURISKO. Isto capacitou–o a monitorar e modificar seu próprio comportamento, tanto quanto a sintetizar e modificar novas funções LISP. EURISKO coleta dados sobre LISP, da mesma forma que ele faz sobre matemática elementar ou jogos. Por exemplo, foram dadas originalmente a EURISKO unidades para EQ e EQUAL, sem nenhuma conexão explícita entre elas. Eventualmente ele procurou exemplos e contra–exemplos para elas e conjecturou que EQ era uma restrição (um predicado mais especializado) de EQUAL. Uma heurística sugeriu testar EQ na frente de EQUAL, pois isto poderia acelerar EQUAL. Surpreendentemente, para o autor, isso ocorreu. Uma vez tendo a conjectura de que EQ era um tipo especial de EQUAL, ele era capaz de procurar seus códigos e especializar "bits" substituindo EQUAL por EQ, ou generalizá–los, substituindo na ordem reversa.

iii) Resultados em heurísticas

EURISKO sintetizou muitas novas heurísticas e algumas vezes isto ocorreu como um sub–produto de outras atividades, durante o trabalho em algum domínio de tarefa particular.

Algumas das novas heurísticas descobertas pelo programa são:

– específica a jogos (ex.: "Planejando uma força militar vá próximo mas não absolutamente a extremos");

– específicas a matemáticas (ex.: "Se uma função inversa vai ser usada ainda mais uma vez, então é provavelmente valioso pesquisar por um algoritmo mais rápido para computá–la");

– específicas a programação (ex.: "Se você pode usar EQ ao invés de EQUAL, faça–o para poupar tempo", ou "Algumas vezes "e" significa "faça em sequência" e algumas vezes significa "faça simultaneamente", e é importante distinguir esses dois casos, antes de considerar a generalização ou especificação de uma peça do código");

– umas poucas específicas a heurísticas em si mesmas (ex.: "Se você está generalizando uma heurística, então evite mudar o principal conectivo das premissas da heurística de "e" para "ou", isto é uma generalização mas conduz a terríveis resultados tais como "loops" infinitos e erros em LISP).

Muitas heurísticas adicionais foram criadas sinergeticamente, com crédito para EURISKO e para uma ou mais pessoas trabalhando com o programa.

LENAT (1983b) sumariza conclusões obtidas sobre descoberta automatizada, e usa–as para explicar: porque A. M. trabalhou inicialmente tão bem, porque A. M. posteriormente falhou, porque custou tanto fazer EURISKO, e porque EURISKO agora trabalha.

(1) O domínio deve ser tão pouco explorado quanto possível. Campos que são já muito compreendidos não são candidatos promissores, nos quais pesquisar novas dscobertas, nem a nivel de domínio, nem a nível de heurísticas. Campos jovens, como teoria dos grafos, são mais promissores. Campos bem

estabelecidos tornaram–se já algoritmizados. Se o programa inicia com adequada representação (já formada, em campos bem desenvolvidos) a descoberta de fatos e de heurísticas será grandemente facilitada, conduzindo a informação errônea sobre o processo de descoberta.

(2) Deve haver uma forma para simular ou diretamente efetuar experimentos.

(3) O "espaço de pesquisa" deve ser excessivamente grande para outros métodos de trabalho. Geralmente esse critério significa "excessivamente grande para exploração sistemática". LENAT (1983b) refere–se a "excessivamente grande para pesquisa exaustiva manual". Isto é, possuir o "tamanho" certo para ser estudado por métodos de Inteligência Artificial. Esse critério possui uma relação conversa: o espaço de pesquisa não deve ser excessivamente grande para métodos heurísticos.

(4) Lá devem estar muitos objetos, operadores, tipos de objetos, e tipos de operadores. Eles devem ser relacionados hierarquicamente, e de outras formas. Isso torna útil uma representação baseada em unidade ("frame"), orientada por classes. Cada forma na qual duas entidades podem ser relacionadas é um tipo diferente de aspecto.

(5) O domínio da tarefa deve ser rico em estrutura heurística. A complexidade do domínio dá origem à utilidade de raciocínio inexato, plausível, tanto mais quanto a inferência precisa (exata) torna–se impossível. Deve haver muitas heurísticas boas que podem ser aplicadas, e nenhum algoritmo bom.

(6) Deve haver formas para gerar, podar e avaliar. Devem ser disponiveis muitas heurísticas que geram (sugerem movimentos plausíveis), avaliam (julgam a utilidade e problemas específicos com as descobertas), e que podam (eliminam caminhos implausíveis, antes que sejam explorados muito profundamente).

(7) A "linguagem" usada para representar conceitos deve ser natural, dado o conjunto de objetos e operadores.

Essa é a conclusão mais crucial, a que LENAT (1983b) chegou apenas recentemente. Num nível muito abstrato, pode–se ver a tarefa de domínio como sendo uma tarefa na qual operadores de domínio (ex.: procedimento de laboratório, funções matemáticas) são aplicados a objetos de domínio (ex.: culturas, conjuntos).

Além disso EURISKO tem uma coleção de operadores de nível mais alto que combinam entidades de domínio em novos (ex.: Componha) e que executam cirurgias em entidades de domínio individual, para produzir outras modificadas (ex.: Generalize). A tarefa de um descobridor como A. M. é aplicar esses operadores de nível mais alto, de uma maneira eficiente, com alta taxa de acerto. A tarefa de um descobridor do tipo de EURISKO é encontrar heurísticas que guiem a aplicação de operadores de nível mais alto, de forma que os resultados sejam frequentemente frutíferos. Se os operadores de alto nível são bem emparelhados à forma pela qual as entidades de domínio são representadas, a maioria das aplicações conduzirão à novos conceitos com significado.

Em outras palavras, mesmo se a descoberta de novas heurísticas é importante, a presença (e manutenção) de uma representação adequada para conhecimento é ainda mais necessária. Uma vez havendo tal emparelhamento, como em A. M., o principal problema parece ser então, a descoberta de novas heurísticas.

(8) Critérios que tornam um domínio adequado, para uma exploração semelhante a do A. M., são os mesmos critérios que tornam um domínio adequado à exploração semelhante a do EURISKO.

O domínio, para A. M., deve estar em aberto, não estar projetado, ser internamente formalizável, e possuir uma rica estrutura de heurísticas. Em casos extremos um domínio é tão inexplorado, que não existem heurísticas ou especialistas humanos no campo. Nessa situação, a abordagem de EURISKO será frutiferamente aplicável: cada heurística útil produzida será uma nova descoberta.

LENAT (1983b) analisa o comportamento de A. M. e de EURISKO, aplicando os critérios prévios, para explicar falhas, sucessos e dificuldades encontrados. Considera esse argumento admitidamente circular, uma vez que aqueles critérios foram abstraídos justamente de tais experiências. Coloca que apenas pesquisa futura com EURISKO, em novos domínios de tarefa, e nestes domínios mas de forma mais profunda, será capaz de testar estas afirmações.

A medida que A. M. trabalhou, ele construiu definições cada vez mais amplas para seus conceitos derivados. Os antigos combinadores e mutadores não mais eram capazes de manter uma alta "taxa de acerto". A. M. necessitava ao menos de:

(i) novos operadores de alto nível;

(ii) novas heurísticas para guiar o processo;

(iii) um novo e diferente conjunto de aspectos (ou equivalentemente um conjunto de primitivos de programação diferente), de forma que definições de conceitos fossem novamente expressões curtas, que operadores de alto nível pudessem trabalhar frutiferamente;

(iv) uma boa interface com especialistas humanos a fim de haver interação homem–máquina.

Segundo LENAT (1983b) os dois itens finais são os mais poderosos e plausíveis. A quarta não é útil nos campos de EURISKO, uma vez que ainda não há especialistas humanos neles.

O terceiro ítem não foi considerado até 1979. Até então, cada heurística representada superficialmente como uma unidade tinha dois aspectos executáveis que basicamente definiam–na: SE e ENTÃO. Com a passagem do tempo, mais e mais aspectos, que uma heurística pode ter, foram definidos. Esse conjunto de aspectos deve ser expandível dinamicamente. Para cada sete heurísticas, em média, um novo tipo de aspecto foi definido (substituindo SE e ENTÃO).

II.3 – BASE TEÓRICA E IMPLEMENTAÇÃO DE SISTEMAS COMPUTACIONAIS

II.3.1 – Teorias de Psicologia Cognitiva Incorporadas em Programas Computacionais: Revisão de Smith

SMITH (1985) apresenta um levantamento de teorias cognitivistas que vêm sendo incorporadas em programas computacionais. Uma síntese desse trabalho é apresentada a seguir.

Estudos Sobre o Raciocínio Dedutivo

O tema dominante em teorias sobre o pensamento dedutivo, nos termos de SMITH (1985), é que o raciocínio dedutivo envolve a aplicação de regras de inferência mental, às premissas e conclusões de argumentos. Uma teoria psicológica de raciocínio dedutivo requer uma especificação de todas as regras de inferência

mental, e das condições sob as quais elas são aplicáveis. Teorias desse tipo tem sido desenvolvidas por OSHERSON (1974), RIPS (1983) e JOHNSON–LAIRD (1983, In: SMITH, 1985).

O modelo de RIPS (in: SMITH, 1985) tem a propriedade adicional de permitir que o sujeito que raciocina, trabalhe de volta das conclusões, tanto quanto a partir das premissas. De acordo com esse modelo, o sujeito com uma estrutura avalia assertivas, incluindo as premissas e proposições que podem ser derivadas das premissas, por aplicação de regras de inferência. Com outra estrutura avalia sub–metas, incluindo a conclusão e cada proposição que a garante. O sujeito julga um argumento válido, se um emparelhamento pode ser encontrado entre adequadas sub–metas e assertivas.

Recentemente, JOHNSON–LAIRD (In SMITH, 1985) contestou idéias aceitas sobre o raciocínio de inferência. Seu argumento mais forte é a existência de "efeitos de conteúdo", isto é, o fato de um sujeito falhar em um problema, quando ele é expresso em termos de um domínio de conteúdo (particularmente o de símbolos abstratos), mas resolvê–lo quando ele é expresso em termos de outro domínio (por exemplo, alguma situação concreta). Argumenta que se a dedução fosse uma questão de aplicar regras abstratas, a performance não dependeria do conteúdo em que o problema é estruturado. A resposta desse autor é que efeitos de conteúdo surgem porque o raciocínio é feito no próprio conteúdo.

SMITH (1985) considera atrativa a abordagem dedutiva desses modelos mentais. Supõe que algumas das intuições que a dirigem encaixam–se com recentes desenvolvimentos sobre conceitos e sobre indução. Assim, de acordo com modelos mentais, as pessoas raciocinam com conteúdos específicos e não com regras lógicas abstratas. Considera que é improvável que sejamos capazes de explicar o raciocínio indutivo sem algum recurso de regras lógicas, e que, portanto, é de se esperar alguma harmonia entre o raciocínio diário e o raciocínio formal, em futuros modelos nessa área.

Estudos Sobre Raciocínio Indutivo

A partir de 1979, estudos foram desenvolvidos por KAHNEMAN e TVERSKY (in: SMITH, 1985). Contradizendo posicionamento, até então aceito, esses autores supõem que as pessoas julgam "tipicamente" e não "probabilisticamente": baseiam suas estimativas na similaridade de uma descrição com seu protótipo, e ignoram informações que poderiam ter acerca de frequências.

Os autores referem–se à "representatividade" heurística, que parece envolver processos equivalentes aos que serão discutidos mais adiante, sobre teorias de protótipos.

Esses autores estudaram julgamentos heurísticos, um dos quais é chamado "disponibilidade": as pessoas julgam a frequência de alguma classe ou evento, através da facilidade com que suas instâncias se tornam disponíveis.

Outra heurística é aquela da "análise causal", por meio da qual as pessoas julgam a probabilidade de algum evento, pela facilidade com que podem gerar uma avaliação causal sobre ela.

SMITH (1985) considera que os trabalhos sobre pensamento indutivo, em oposição aos sobre pensamento dedutivo, são pouco articuláveis em um nível computacional, devido ao fato de apresentarem um nível de descrição quase abstrato.

Estudos Sobre Conceitos e Representação do Conhecimento

Um significativo trabalho nessa linha, foi desenvolvido por BRUNER (1950). Esse autor preocupou–se com a natureza de conceitos (ou categorias), como eles eram aprendidos, enquanto eles eram úteis para solução de problemas. O estudo de Bruner, de formação de hipóteses e estratégias para evitar erros em aprendizagem, é considerado particularmente bem desenvolvido. Bruner focalizou a resolução de problemas sob o aspecto de estímulo e resposta, onde o estímulo é o significante, e a resposta é a ação que a categorização dita.

Segundo SMITH (1985), a partir da década de 70, o trabalho de Bruner foi de certa forma considerado superado pelo de ELEONOR ROSH (in: SMITH, 1985), sobre a representação do conhecimento.

A visão de conceitos adotada por essa autora é conhecida como teoria de "protótipos". Postula que um conceito é mentalmente representado por um "protótipo", isto é, por propriedades que são verdadeiras para algumas instâncias desse conceito. Um objeto é categorizado como uma instância de um conceito na extensão em que ele é similar ao protótipo do conceito. Algumas instâncias são mais "típicos" membros de um conceito do que outras. Pesquisas empíricas tem demonstrado que a tipicidade de uma instância afeta: a rapidez com que ela pode

ser categorizada e com que ela será aprendida; o quão fácil é relembrá–la; a quantidade de suporte que ela pode fornecer a uma inferência indutiva, etc. As idéias básicas são similares a conceitos da Inteligência Artificial, segundo SMITH (1985).

Uma linha mais recente de desenvolvimento em Teoria de Protótipos refere–se à regra ou à função pela qual similaridade ao protótipo é determinada. Outro recente desenvolvimento de teorias de protótipos envolve combinação conceitual. Se conceitos são mentalmente representados como protótipos, deve haver um procedimento mental para combiná–los; para que as pessoas sejam capazes de gerar um número indefinido de conceitos complexos novos, a partir dos simples. Nessa linha, o trabalho de Smith e Osherson, segundo SMITH (1985), é compatível com recentes trabalhos em Inteligência Artificial.

CLANCEY (In: SMITH, 1985) apresenta uma distinção, em resolução de problemas, entre a classificação e a construção. Na classificação heurística o espaço de solução é conhecido do sujeito como um conjunto de alternativas explícitas, e a resolução do problema toma a forma de "provar" que uma delas é melhor ou mais aceitável. Em outras situações, possíveis estados finais não podem ser praticamente enumerados, exaustivamente aprendidos (da experiência ou por ensino direto), ou por alguma razão, uma solução previamente usada não é aceitável. Soluções devem então ser construídas ao invés de selecionadas, embora classificações possam tomar algum lugar.

II.3.2 – Metodologia de Pesquisa na Inteligência Artificial: Reflexão de Ritchie e Hanna

RITCHIE e HANNA (1984) consideram que "grande parte da pesquisa em Inteligência Artificial é baseada na construção de amplos programas, impressionantes no aspecto, cujos conteúdos teóricos nem sempre são claramnente afirmados. Isto não é produtivo do ponto de vista de construção de uma base estável para pesquisas posteriores" (pg. 249). Ilustram esse problema com o programa A. M. de LENAT (1976, 1977, 1979, In: RITCHIE e HANNA, 1984).

Os autores focalizam a Inteligência Artificial como uma ciência relativamente nova, na qual existem várias influências de diferentes disciplinas (psicologia, filosofia, ciência da computação, etc.). Supõem que um sintoma dessa situação é a falta de qualquer forma claramente definida de procedimento de pesquisa nesse campo. A deficiência do estilo predominante de pesquisa é sua

esterilidade teórica – nele são descritos o que complexos programas fazem, e não são desenvolvidos princípios, nem definidos os reais problemas de pesquisas. Recentemente essa deficiência vem sendo considerada: complexos programas com comportamento impressivo são ainda construídos, mas é acrescentada uma afirmação de como eles atingem essa performance. Infortunadamente, em alguns casos, existem confusões metodológicas, resultantes de discrepâncias entre a "teoria" escrita e o que o programa faz. Em consequência a tentativa de replicar ou desenvolver a pesquisa pode levar à necessidade de um retorno ao estágio inicial, uma vez que a "teoria" de seu predecessor (estado da arte) fornece poucas diretrizes, especificando uma execução a ser atingida mas não como atingí–la.

RITCHIE e HANNA (1984) ilustram esse ponto de vista, exemplificando com o A. M. Supõem que esse trabalho foi altamente considerado na comunidade de Inteligência Artificial. Aceitam que o programa A. M. seja capaz de produzir os "outputs" publicados, mas supõem que os registros escritos dão uma visão confusa de como o programa trabalhou, a ponto de obscurecer a real contribuição do trabalho e de confundir outros autores do campo.

Consideram A. M. uma peça substancial do trabalho em Inteligência Artificial, além de criativo e original, mas acreditam que é um exemplo de um estilo de relatório científico em Inteligência Artificial que não é propício a continuidade e ao gradual desenvolvimento de uma construção teórica. O ponto central de suas críticas é a discrepância entre as reivindicações publicadas e os detalhes de implementação.

Resumem suas dúvidas:

1) A estrutura de controle é simplesmente o mecanismo uniforme e mínimo apresentado, com a tarefa corrente definindo exatamente que regras executar ?

2) As estruturas de dados são uniformemente representadas como Conceitos, como Aspectos e Sub–Aspectos, como definido ?

3) Não existe nenhum mecanismo de processamento extra, que possuam efeitos significativos ?

4) Quanto do trabalho de descoberta real é oculto em procedimentos não explicados ?

5) Se as descobertas matemáticas são realmente um aspecto importante do trabalho, é possível mostrar que elas resultaram diretamente do procedimento de pesquisa como descrito ?

Acreditam que o estágio de desenvolvimento da Inteligência Artificial é pré–científico, assim como sua metodologia, e que muito poucas pesquisas em Inteligência Artificial correspondem ao "paradigma experimental" tradicional, no qual hipóteses bem definidas são refutadas por investigações empíricas.

Consideram que existem várias formas nas quais o projeto típico de Inteligência Artificial (ex.: tese de doutorado) pode contribuir para o conhecimento coletivo:

1) Pode introduzir, em linhas gerais, uma idéia ou conjunto de idéias novas (ou particularmente novo). Geralmente este tipo de trabalho não é preciso o suficiente para admitir refutação empírica.

2) Pode elaborar os detalhes, de alguma abordagem. Isto é, partindo de uma idéia (do tipo anterior) a pesquisa pode criticá–la ou preenchê–la com mais detalhes, a fim de transformar uma metáfora em uma teoria mais completa. Esta atividade é comparável à construção de teoria numa ciência tradicional, mas não é amarrada a algum meio de testagem padronizado.

3) Pode aplicar alguma teoria articulada, a algum assunto efetivo e relatar as consequências. Este é o ponto de partida para o experimento tradicional mas difere em alguns aspectos importantes. A investigação prática tipicamente procede através da escrita e execução de um programa de computador, e a avaliação dos resultados dessa atividade é difícil, desde que as idéias de Inteligência Artificial tendem a não ser formuladas de tal forma a permitir julgamentos específicos de sucesso/fracasso.

Um experimento empírico de I. A., como A. M., pode ser escrutinado de várias formas:

3.1– Projeto de experimento (estático): a estrutura do programa reflete a teoria que se propõe a testar ?

3.2– Projeto do experimento (dinâmico): o comportamento de processamento interno do programa corresponde a aspectos dinâmicos da teoria ?

3.3– Consistência: o programa roda com êxito (repetidamente) ?

3.4– Resultados: quais são as consequências (em termos de comportamento interno e de saída) da execução do programa ?

3.5– Interpretação: o que significam esses resultados em termos de constructos teóricos ?

3.6– Conclusões: quais são as consequências para a teoria dos resultados interpretados ?

RICHIE e HANNA (1984) acreditam que a maioria do trabalho em Inteligência Artificial contribuiu nas três categorias, com a maior parte do esforço sendo dispendida em 2 e 3; e que A. M. contribuiu com relação à 1ª. A substância de 2 é menos clara, desde que não há afirmação da teoria, separada da descrição do programa. Entretanto, o principal problema surge com a questão da testagem empírica (as respostas a 3.1 e 3.2 parecem ser negativas). Estranham que nenhum trabalho corroborativo tenha surgido nos últimos 5 anos, embora considerem esse fato uma das peculiaridades de Inteligência Artificial.

Consideram como contribuição positiva do A. M. a proposta de que a descoberta matemática pode ser uma forma sistemática de inferência, baseada num tipo particular de estrutura de conhecimento ("frames" organizados hierarquicamente), e um arranjo particular de regras (indexados a conceitos relevantes). O que não foi atingido foi um teste detalhado e programado destas idéias, desde que A. M. parte desta proposta original. Como contribuição negativa levantam a pouca clareza da força e fraqueza de tal esquema, e a dificuldade para qualquer um seguir sistematicamente esta pesquisa. Acreditam ser extremamente difícil futuras pesquisas na área basearem–se no A. M., desde que a disparidade, entre os relatos escritos e o programa, significa que não há de fato uma base teórica testada a partir da qual trabalhar. Muito da informação do complexo programa A. M. permanece conhecida apenas por seu criador. Concluem que seria desejável: maior esforço para suprir afirmações teóricas (distintas de descrições de implementações); mais clara correspondência entre teoria e programa; relato mais detalhado do trabalho, objetivando o seguimento por pesquisadores do campo.

CAPÍTULO III

MÉTODO

Duas suposições são colocadas nesse trabalho:

– a aprendizagem do sujeito cognoscente da Inteligência Artificial e a aprendizagem do sujeito cognoscente da Psicologia podem ser focalizadas num processo interdependente;

– pesquisas sobre sistemas capazes de descoberta podem encontrar um referencial teórico particularmente adequado nos constructos explicativos da epistemologia e psicologia genética.

A fim de investigar essas suposições, buscou–se definir:

– um modelo de processo de aprendizagem desenvolvido interativamente entre sujeito e máquina, com base na teoria de Piaget;

– uma situação específica em que esse processo é implementado em protótipo, como um exemplo de aplicação.

O método empregado nesse estudo, portanto, corresponde ao definido por LENAT (1982) como paradigma padrão de inquérito empírico, que domina a Inteligência Artificial. Nos termos desse autor, esse método consiste de teste de hipóteses a partir de construção de sistema computacional.

Nos próximos itens é feita uma análise de aspectos teóricos que nortearam a definição desses itens.

III.1 – MODELO DO PROCESSO DE APRENDIZAGEM – PRESSUPOSTOS TEÓRICOS

Definir um modelo de processo de aprendizagem em que sujeito e sistema, ambos noviços, se enriquecem, ao mesmo tempo em que propiciam recíprocamente novas aquisições, requer pensar–se num sistema capaz de:

– construir novos objetos não imaginados pelo sujeito, realimentando o processo construtivo do indivíduo;

- descobrir regras, a partir de sua própria ação e da ação do sujeito, e em função de seus conhecimentos precedentes, que orientem a construção de objetos;

- reconstruir essas regras, inexatas e parciais, regulando sua própria ação;

- satisfazer os itens anteriores de forma contínua e gradativa, isto é, desenvolver uma busca limitada por meta–regras (e não sistemática e exaustiva), retomando–a munido de novos recursos, em cada próximo LOOP.

Imaginar um sistema que é capaz de descobrir novos objetos, requer pensar–se num sistema capaz de aprendizagem ao menos em sentido estrito.

Mas, se essas regras devem, elas próprias, ser descobertas, então esse sistema deve ser capaz de uma aprendizagem em sentido amplo, isto é, de construir seus recursos cognitivos.

Algumas conclusões de Piaget e seus seguidores, que mais de perto nortearam a definição do processo de aprendizagem, são apresentadas a seguir.

A Escola de Geneve considera fundamental a diferença entre o registro de uma necessidade empírica, e a compreensão de uma necessidade lógica. Nos termos de grêco (1974) "... não parece duvidoso, que a aprendizagem só seja eficaz na medida em que conduz a uma estruturação, e que essa estruturação não poderia ser produzida por uma simples acumulação passiva de constatações empíricas" (pg. 181).

GRÊCO (1974) supõe a existência de dois métodos distintos de aprendizagem:

- constatações empíricas, que levam a conhecimentos precários e lábeis, limitados a dados observados;

- estruturações, que conduzem a conhecimentos mais estáveis, duráveis e generalizáveis.

A primeira forma de aprendizagem, "forma discriminativa", corresponde à

distinção de casos em que se aplica uma lei. Conduz à formação de saberes empíricos, fatos admitidos mas não compreendidos, sem uma razão lógica.

Na segunda, "forma estruturante" trata–se de compreender porque uma lei se aplica.

Essas estruturações podem permanecer incompletas, inacabadas, e os esquemas podem não se aplicar mais a uma situação diferente. Nesse caso, trata–se de estruturações parciais, de esquemas "semi–lógicos", de "quase–noções".

Em presença de uma situação acessível, o sujeito procura compreendê–la em função de seus conhecimentos precedentes. Nesse sentido, segundo esse autor não é nem surpreendente que permaneça muito tempo rebelde à verdade constatada nem paradoxal que as constatações repetidas levem finalmente à conhecimentos novos.

Esses conhecimentos podem ser:

– convicções aceitas, em que um fato anormal, repetindo–se com certa constância, é aceito como normal – nesse caso há justaposição, uma exceção é acoplada à regra;

– compreensão, em que o fato anormal é admitido como normal – nesse caso há diferenciação do esquema anterior, através do processo de equilibração.

Nos termos de GOUSTARD (1974), a aprendizagem não depende somente dos dados fornecidos pela experiência externa, mas igualmente de um "recurso das condições de organização interna" (pg. 163). Condutas aprendidas R_n, R_n+1, etc., são relativas aos modos de coordenação C_n, C_n+1, etc. que exprimem o poder de organização lógica ou pré–lógica dos sujeitos.

PIAGET (1974) conclui que essas coordenações "constituem o domínio específico da equilibração porque seu desenvolvimento consiste em uma organização progressiva, orientada no sentido da reversibilidade operatória, isto é, de compensações sempre mais poderosas. É precisamente como mecanismos de compensação que as coordenações não são aprendidas no sentido estrito mas resultam de uma atividade do sujeito em reação as perturbações exteriores" (pg. 184).

O aspecto interativo, salientado no presente trabalho, está presente em toda psicologia genética, e é focalizado com ênfaze nos estudos de Inhelder.

BOVET e MONTANGERO (1983) analisam a constância com que esse tema é tratado pela discípula de Piaget (INHELDER, 1936, 1945 INHELDER e outros, 1974; In: BOVET e MONTANGERO, 1983).

Esses autores lembram que, já em 1936, a formação de hipóteses, originada na discordância, entre resultados previstos e obsevados experimentalmente, é estudada por Inhelder.

O aspecto interativo é retomado, nos anos 50, quando a pesquisa do adolescente é vista como um diálogo com a experiência.

Na década de 70, nos estudos de aprendizagem, é salientada a importância de mecanismos de interação entre os esquemas assimilativos do sujeito e procedimentos que lhe são propostos.

Mais recentemente, nas pesquisas sobre procedimentos de resolução de problemas, é focalizada a inter–relação entre esquemas do indivíduo e dados da situação experimental.

III.2 – DEFINIÇÃO DE UMA SITUAÇÃO DE APRENDIZAGEM DE HOMEM E MÁQUINA – UM EXEMPLO

Uma primeira questão que se coloca é em que situações pode ser necessária a utilização de um sistema como o proposto nesse trabalho.

A resposta à essa indagação pode ser: quando o sujeito se desenvolve num campo em aberto, em que possibilidades são amplas, limites são pouco conhecidos, conceitos são definidos de forma instável, isto é, quando no domínio em que se processa a investigação não existe conhecimento acabado e estável. Em função do grau de acabamento possuido, as investigações podem partir de um nível inicial – exploratório, em que meios devem ser descobertos, e até mesmo os próprios fins podem ser redefinidos (sujeitos noviços) – ou de uma fase final – de acabamento, em que se trata de adquirir uma definição explícita de meios que são adequadamente utilizados, mas de forma parcialmente intuitiva ou não totalmente consciente (sujeitos já considerados especialistas em alguns domínios).

Na aplicação implementada sob forma de protótipo, pressupõe–se a existência de um procedimento computacional, definido para o problema investigado. A busca é desencadeada pela verificação do desempenho desse procedimento, pela análise de dados de saída, em função de diferentes dados de entrada, pela formulação de hipóteses sobre relação entre variáveis.

Suposições acerca do processo de descoberta envolvido na ação de construir e testar um procedimento computacional são feitas em trabalhos anteriores (CUNHA, 1986a, 1986b). Nesses estudos foram abordados aspectos relativos a: atuação de mecanismos de assimilação e acomodação; situações de conflito cognitivo que são desencadeados quando suposições iniciais são estabelecidas, e em seguida confrontadas com resultados inesperados obtidos; construções e reconstruções que são requeridas a partir de então.

Além disso, focalizou–se a ocorrência de processos mentais distintos, mas também intimamente relacionados, na medida em que, cada passo reune elementos e construções parciais, de um passo anterior, e prepara um posterior. Na construção de conhecimentos, no domínio da tarefa, observou–se soluções "corretas", mas qualitativamente distintas, para o problema investigado (caminho hamiltoniano). A passagem entre diferentes tipos de solução foi relacionada à equilibração progressiva (CUNHA, 1986a).

Na aplicação proposta nesse estudo, o procedimento investigado é definido pelo sujeito em linguagem LOGO–gráfica. A escolha dessa linguagem computacional, que pode ser considerada um dialeto LISP, decorre de algumas de suas características.

Vários aspectos de LOGO, alguns comuns a LISP, outros introduzidos por Papert, tornam a utilização dessa linguagem extremamente facilitada num patamar inicial.

Entre os primeiros destaca–se:

– a partir do aprendizado de pequeno número de funções primitivas, todas as demais podem ser desenvolvidas pelo sujeito;

– conceitos podem ser progressivamente definidos e generalizados;

– o efeito de um único comando pode ser isoladamente testado;

– e de forma igualmente simples, um conjunto de comandos pode ser imediatamente avaliado.

Dentre os últimos, cita–se a possibilidade de:

– utilizar comandos não aplicativos;

– visualizar concretamente na tela efeitos de comandos (LOGO gráfico);

– lidar com uma notação simplificada;

– dispor de funções inseridas em sua biblioteca.

Por outro lado, alguns conceitos de programação, característicos de LISP e mantidos em LOGO, não são dominados sem uma construção mais poderosa, dependendo do grau de estruturação e adequabilidade dos modelos do indivíduo:

– modularidade requer que o problema seja simultaneamente focalizado no seu todo e nas partes que o compõem, e que sejam consideradas interrelações entre as partes;

– a não atribuição direta de valores a variáveis, requer a compreensão de que eles podem resultar da transformação de outro valor, por aplicação de função;

– composição funcional requer que seja estabelecida a equivalência entre uma função e a aplicação de uma segunda ao resultado de uma terceira;*

– controle de sequência através de recursão requer a compreensão de que uma função pode ser composta com ela mesma, etc.

Supõe–se que uma consequência da utilização de uma linguagem como LOGO é que a construção de conhecimento pode tomar duas direções distintas:

---

* Piaget utiliza, como modelo de cognição do pensamento formal, o sistema cujas transformações formam (com a operação de combinação) o grupo INRC (idêntica, inversa, recíproca e correlata). A composição funcional de duas ou mais transformações I, N, R, C é equivalente à aplicação particular de uma delas (NR = C; RC = N; NC = R; NRC = I, etc.).

– aspectos pouco compreendidos, do problema a ser resolvido, podem ser controlados como exceções à regras, soluções parcialmente intuitivas podem ser admitidas e mantidas como tal;

– aspectos pouco compreendidos podem originar novas construções, soluções parcialmente intuitivas podem ser reconstruídas em outro nível (explicando também exceções).

Considera–se que isso possa ocorrer, na medida em que:

– se a compreensão e utilização de conceitos básicos é suficiente para soluções de alguma forma satisfatórias e o sujeito não é instigado a uma busca em outro nível, então sua ação permanece mais ou menos restrita à aplicação desses conhecimentos (assimilação a seus esquemas de ação);

– se o sujeito, ao contrário, é confrontado com a necessidade de busca, então novas descobertas e construções podem ser feitas, e munido de novos recursos pode tentar nova aplicação, em nível mais complexo.

Esse processo – LOOP de aplicação de conhecimento, novas construções – pode ser continuamente realimentado por um sistema tal como proposto nesse trabalho. O conhecimento do sujeito pode então ser contínua e gradativamente construído, cada passo incluindo e integrando resultados de passos anteriores, à medida em que a máquina faz descobertas, numa busca complementar à sua.

CAPÍTULO IV

RESULTADOS

Nesse capítulo são definidos:

- um modelo de processo de aprendizagem desenvolvido interativamente entre homem e máquina;

- uma aplicação específica desse modelo, implementada em forma de protótipo.

A Figura IV.1 apresentada a seguir sintetiza a ação da máquina, que constrói seu conhecimento em processo interativo com o sujeito.

O modelo construído define o processo de aprendizagem de uma forma genérica. Um conjunto de meta–regras direciona a ação do sistema, independentemente do conteúdo a ser aprendido ou do domínio de investigação.

Para cada conteúdo no entanto, alguns aspectos devem ser definidos de forma particular, tais como: operadores que devem ser utilizados na construção de regras; invariâncias que devem ser abstraídas pelo sistema, nessa construção.

A definição de uma situação de aprendizagem permitiu uma experimentação concreta, e, a descrição apresentada nesse capítulo refere–se a essa situação específica. Abstraídos aspectos relativos ao conteúdo, sobre o qual o conhecimento é construído, a descrição corresponde à do modelo genérico.

No exemplo ou aplicação apresentado, o problema a ser investigado por sujeito e sistema consiste de: descoberta de novos objetos que podem ser construidos (numa tela de computador) por procedimento computacional definido em linguagem LOGO.

Um novo objeto é obtido, a cada combinação diferente de valores, que são atribuídos a variáveis do procedimento.

A aprendizagem da máquina em sentido amplo consiste da construção de regras, que progressivamente capacitam–na a produzir objetos novos e adequados.

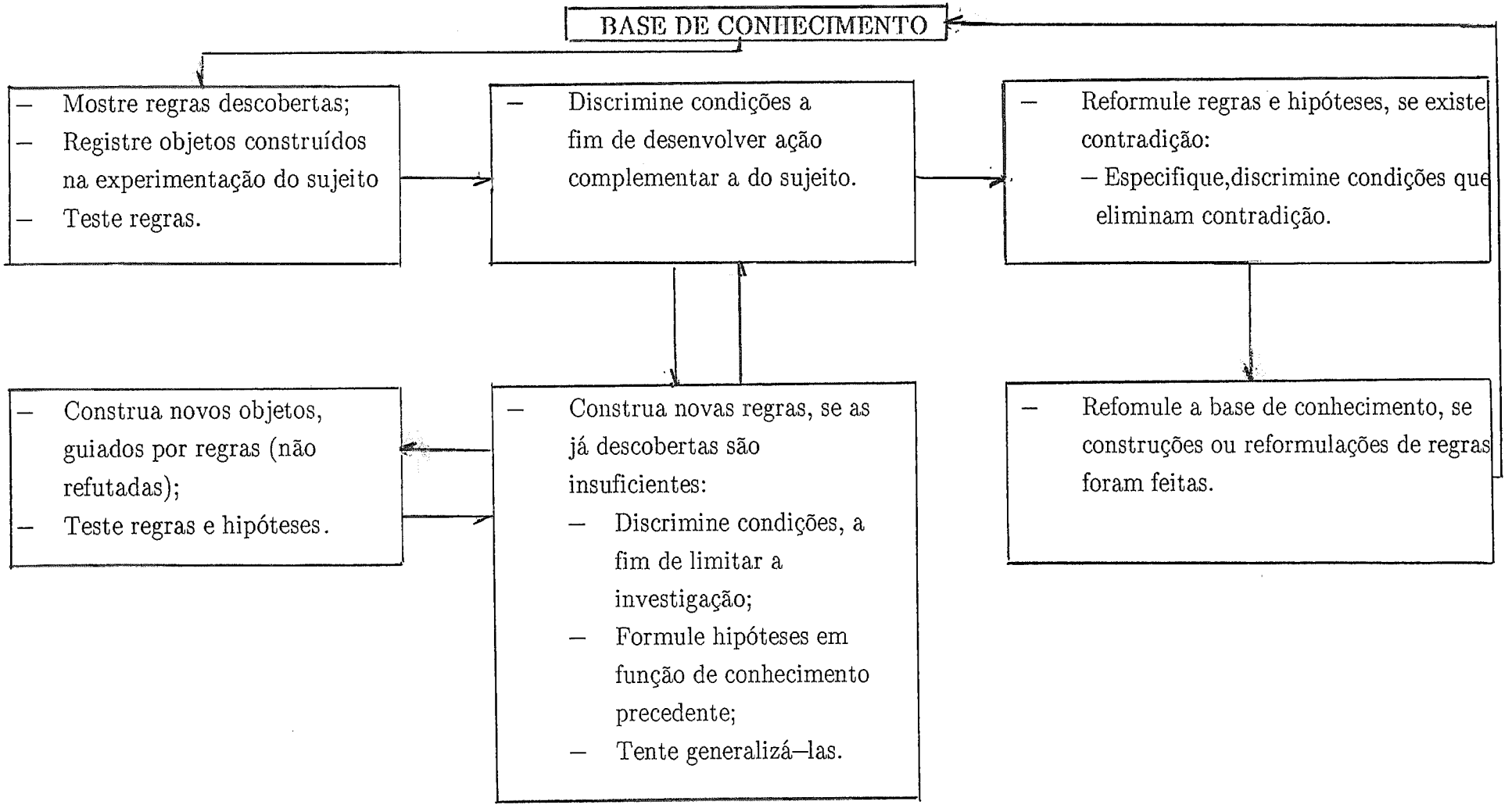

Figura IV.1 – Descoberta de máquina em processo interativo com sujeito

IV.1 – BASE DE CONHECIMENTO

(i) conjunto definido pelo usuário e incorporado pelo sistema à Base de Conhecimento:

– um procedimento qualquer, definido em LOGO;

– um conjunto de variáveis existentes nesse programa[*1] ;

– um conjunto de instâncias, que podem ser associadas a essas variáveis.

(ii) conjunto definido previamente, que faz parte integrante do sistema:

– regras que norteiam a construção e descoberta de novos objetos;
– meta–regras que norteiam a construção, reconstrução e avaliação de regras, e a reformulação da Base de Conhecimento;
– operadores aplicados à objetos;
– operadores de mais alto nível aplicados a regras;

(iii) conjunto gradativamente adquirido:

– regras heurísticas construídas pelo sistema que guiam a construção de objetos;[*2]

– regras inacabadas, parcialmente construídas, que podem guiar ou não a construção de objetos;

– novas instâncias assimiladas pelo sistema.

O conjunto definido pelo usuário, pode variar ou não, em distintas ocasiões. Sujeito e sistema podem adquirir conhecimento sobre um conjunto definido em (i); e retomar ou substituir esse conjunto numa próxima interação.

[*1] Constituem dados de entrada do procedimento definido pelo usuário, sobre os quais é feita a construção do conhecimento.

[*2] Define–se objeto, por uma combinação específica de instâncias (cada objeto, portanto, corresponde a um resultado que pode ser obtido com o procedimento).

O segundo conjunto inclui meta–regras e operadores de mais alto nível (aplicados a regras) que, de forma geral, independem do domínio no qual o conhecimento é construído, e regras e operadores aplicados à objetos, que podem variar em função da natureza dos objetos.

O terceiro conjunto, é inicialmente vazio. Gradativamente novas instâncias são assimiladas e novas regras são construídas e reconstruídas pelo sistema, para um determinado conjunto definido por usuário. Esse conhecimento progride, durante um processo interativo, e pode ser retomado numa próxima interação com um mesmo indivíduo.

Os dois últimos conjuntos são definidos nos próximos itens*1 . O primeiro conjunto é descrito mais detalhadamente no item 5.

IV.2 – REGRAS CONSTRUÍDAS PELA MÁQUINA

Uma regra descoberta pelo sistema consiste da construção de expressão, a partir da qual é abstraída uma invariância que implica em inadequação de objetos.

A fim de simplificar a notação, uma regra é designada da forma geral:

$$(I(F_k) = r) \Longrightarrow F_p$$

onde:

– a parte THEN da regra (F_p) é resultado inadequado de objetos, previsto pela máquina;

– a parte IF da regra é formada por:

– F_k é uma função que recebe dois argumentos, das formas:

$F_k(x, y)$;
$F_k(y, F_j)$;

---

*1 A fim de destacar aspectos que mais caracterizam essa proposta, regras e operadores aplicados à objetos são definidos apenas quando necessário à compreensão desses aspectos.

$F_k(F_i, F_j)$;

onde:

– x e y são variáveis do procedimento definido pelo usuário;

– F_j e F_i são funções do mesmo tipo, que da mesma forma que F_k recebem como argumentos variáveis ou funções;

– F_k, no protótipo implementado, é uma função aritmética (+, –, x, ÷);

– I é uma função que recebe F_k como argumento e retorna um resultado com algum significado no domínio do problema. Nessa aplicação, foram considerados: 0; 1; –1; MÚLTIPLO DE 360/NÃO MÚLTIPLO DE 360; DIVISOR DE 360/NÃO DIVISOR DE 360; INTEIRO/NÃO INTEIRO; POSITIVO/NÃO POSITIVO;

– r é um desses resultados, ou uma disjunção desses resultados.

Uma regra inacabada, descoberta pelo sistema, distingue–se de uma regra, tal como descrito, por ser restrita a condição ou circunstância específica, da forma:

$((I(F_k) = r) \wedge z_i) \Rightarrow F_p$

onde: z_i é uma instância de alguma variável estudada.

IV.3 – META–REGRAS QUE NORTEIAM O PROCESSO CONSTRUTIVO DO SISTEMA

IV.3.1 – Meta–Regras Baseadas no Conceito de Equilíbrio

O primeiro conjunto de regras baseia–se no conceito de equilíbrio.

Define–se equilíbrio da Base de Conhecimento ou de uma regra específica, com base no confronto entre a predição feita pelo sistema, do resultado de um objeto, e resultado efetivamente verificado.

Define–se equilíbrio do conjunto de regras do sistema utilizadas para guiar a construção de objetos, por:

$$(C_1 \text{ v } C_2 \text{ v } \ldots \text{ v } C_n) \equiv F_v$$

onde:

– $C_1 \ldots C_n$ são partes IF das regras que predizem inadequação de objetos;

– F_v é a inadequação efetivamente verificada.[*1]

Define–se equilíbrio de uma regra específica da Base de Conhecimento, ou de uma hipótese de regra, por:

$$C_i \Longrightarrow F_v$$

Em distintas situações o sistema baseia–se no conceito de equilíbrio/não equilíbrio, para acionar conjuntos de regras.

Essas regras serão descritas nos próximos itens, na medida em que forem acionadas ou acionarem outras regras.

De uma forma geral, a não satisfação da definição de equilíbrio aciona a formulação de hipóteses e a reformulação de regras ou hipóteses.

A modificação da Base de Conhecimento é acionada por:

– constatação de não equilíbrio de regras construídas;

– restauração de equilíbrio, após construção ou reformulações de regras;

– constatação de equilíbrio, dada regra inacabada da Base de Conhecimento, que novamente é submetida e resiste a provas de refutação – nesse caso, uma regra apesar de inacabada é admitida como critério de poda.

IV.3.2– Meta–Regras que Norteiam a Construção e Reconstrução de Regras, e a Modificação da Representação

O segundo conjunto de meta–regras diz respeito à:

[*1] A inadequação/adequação de um objeto é definida por avaliação do sujeito.

– formulação de hipóteses de regras;

– teste de regras e/ou de hipóteses;

– reformulação de regras e/ou de hipóteses;

– reformulação de Base de Conhecimento.

IV.3.2.1 – Formulação de Hipóteses de Regras

O grupo de meta–regras que direciona a formulação de hipóteses é acionado por uma constatação de não equilíbrio: cada objeto, de um conjunto, é admitido por todas as regras (não satisfaz critérios de poda) e não é o caso que todos os objetos sejam adequados.[*1]

Esse conjunto de objetos satisfaz uma condição discriminada pelo sistema. Nesse modelo a condição prevista consiste de uma instância x_i, comum a todos os objetos[*1].

A primeira regra consiste de critério para discriminar condições de não equilíbrio que norteiam a investigação, e se a condição não é satisfeita, a busca é encerrada.

Dado o percentual de adequação de subconjuntos de objetos, que tem em comum cada uma das instância (y_1, y_2, ..., y_n; ... z_1, z_2, ..., z_n) experimentadas simultaneamente a x_i.

1) Se existe ao menos uma instância (y_i, ..., z_i) com percentual de adequação maior do que nenhum e menor do que ao menos um, então

1.1) Para cada variável (y) com alguma instância (y_i) que satisfaça a condição anterior, determine sub–conjuntos de maior percentual de inadequação (y_1,..., y_n) e deflagre regras 2 a 5.

[*1] A constatação de não equilíbrio (que deflagra a aplicação desse conjunto de meta–regras) e a discriminação de condição (x_i) são definidas no item IV.3.3. São tratadas, nesse momento, como fatos dados.

1.2) Se não, encerre a formulação de hipóteses.

A segunda regra limita a geração de expressões ($F_i(F_j(x, y), y)$) para formulação de hipóteses ($I(F_i(F_j(x, y), y) = r) \Rightarrow F_P$).

2) Dada a variável y determinada em 1.1, gere função (F_k) que opera variáveis e/ou outras funções (F_i, F_j), tal que:

- x e y são argumentos de alguma função da expressão, e todas as funções da expressão operam x ou y *1

- se a função gerada F_k tem como argumento outra função, então F_k deve ser uma composição funcional de função(ões)* construída(s) anteriormente, e incluída(s) na parte IF de alguma regra, ou de regra inacabada.

A terceira e a quarta indicam caminhos a seguir na formulação de hipóteses.

3) Se existem argumentos (a_1, a_2) que podem ser operados por função F_k satisfazendo condição anterior, então selecione F_k de forma tal que:

3.1) Se F_k (a_1, a_2) já foi investigado em loop anterior, e gerou regra acabada incluída na Base de Conhecimento, então F_k deve ter a mais alta prioridade na formulação de hipótese.

3.2) Se F_k (a_1, a_2) já foi investigado em loop anterior, e não gerou regra acabada*2 , então F_k (a_1, a_2) deve ter a mais baixa prioridade na formulação de hipóteses.

4) Se existe função F_k gerada e selecionada, então formule hipótese da forma $((I(F_k) = r) \wedge x_i) \Rightarrow F_P$, com base em resultados observados no conjunto investigado.

---

*1 Exemplos:
$F_k(x, y)$
$F_k(x, F_i(y, z)^*)$
$F_k(F_i(x,w)^*, y)$
$F_k(F_j(x,y)^*, F_i(y,z)^*)$

*2 Essa informação é registrada na **Reformulação de Regras e/ou Hipóteses** (item IV.3..2.3).

A quinta e a sexta determinam se uma hipótese formulada é de regra inacabada ou acabada.

5) Se existe uma hipótese formulada, então:

- generalize–a, eliminando a condição associada (x_i), da forma ($I(F_k) = r \Rightarrow F_p$;
- colete exemplos do conjunto investigado;
- submeta à prova de refutação, acionando **Teste de Hipóteses e de Regras** (item IV..3.2.2), dado o conjunto de objetos adequados construídos em toda interação.

6) Se e somente se existe (m) contra–exemplo(s) para hipótese de regra acabada tal(ais) que o resultado (r) da hipótese não pode ser especificado de forma a eliminar a contradição, considere a hipótese de regra inacabada.*1

A sétima generaliza o resultado (r) de uma hipótese de regra acabada.

7) Se existe uma hipótese de regra acabada com resultado (r_i), tal que pode ser generalizada se o resultado é transformado numa disjunção de termos complementares (r_i/r_i'), então acione **Teste de Hipóteses e de Regras** (item IV.3.2.2), dado r_i' e o conjunto de objetos adequados construídos na interação*2.

A oitava determina se a investigação, com a variável y ou outras variáveis

---

*1 Exemplos:
(i) $((I(F_k) = \text{NEGATIVO v INTEIRO}) \Rightarrow F_p)$
$\wedge ((I(F_k) = \text{NEGATIVO} \wedge \text{INTEIRO}) \wedge T_v)$
$\Rightarrow$ hipótese de regra inacabada
(ii) $((I(F_k) = \text{NEGATIVO v INTEIRO}) \Rightarrow F_p)$
$\wedge \sim ((I(F_k) = \text{NEGATIVO} \wedge \text{INTEIRO}) \wedge T_v$
$\Rightarrow$ hipótese de regra acabada.

*2 Exemplos:
(i) $((I(F_k) = \text{NEGATIVO} \wedge \neq -1) \Rightarrow F_p)$
teste $((I(F_k) = -1) \Rightarrow F_p)$
(ii) $((I(F_k) = \text{DIVISOR DE 360} \wedge \neq -1 \wedge \neq 1) \Rightarrow F_p)$
teste $((I(F_k) = 1 \text{ v} -1) \Rightarrow F_p)$

determinadas na regra 1.1, deve ou não prosseguir.

8) Se existe hipótese formulada e testada, então:

8.1) Se a hipótese é de regra inacabada, então continue a investigação, reacionando a regra 2.

8.2) Se não (regra acabada), então marque objetos do(s) sub–conjunto(s) determinado(s) em 1.1 que satisfazem a condição da regra, e

8.2.1) Se existe algum objeto não marcado, então reacione a regra 2 e prossiga na investigação, considerando apenas objetos que não satisfazem a condição da regra descoberta.

8.2.2) Se não, (todos os objetos considerados satisfazem a condição) então:

8.2.2.1) Se as regras 2 a 5 já foram acionadas para todas as variáveis determinadas em 1.1, então elimine objetos inadequados determinados em 1.1, e reacione o conjunto de regras de **Formulação de Hipóteses.**

8.2.2.2) Se não, reacione as regras 2 a 5, dada a próxima variável determinada em 1.1.

Com base na segunda meta–heurística, uma hipótese é formulada integrando o que é pouco conhecido e pouco compreendido, ao que já faz algum sentido. Uma nova construção inclui e integra construções anteriores, através de composição funcional.

Com base na regra IV.8.2, uma descoberta é sempre limitada ao mais aparente, isto é, a primeira explicação encontrada mesmo havendo a possibilidade de construções mais acabadas serem feitas.

Além disso, generalização, especificação, e analogia são feitas na formulação de hipóteses.

(i) generalização:

– resultados verificados num pequeno conjunto de objetos, são generalizados

para quaisquer objetos que satisfaçam a mesma condição*1 ;

– uma hipótese de regra restrita a uma condição dá origem à formulação de uma hipótese de regra sem essa restrição*2 ;

– um resultado verificado, num conjunto de objetos é generalizado para resultados mais abrangentes, pela função I*3 ;

– uma hipótese de regra acabada pode dar origem a uma hipótese com resultado complementar (e ambas à hipótese com resultado mais genérico)*4 .

(ii) Especificação:

– uma hipótese refutada pode ainda ser admitida, se existe contra–exemplo para algum mas não todos os termos do resultado*5 .

(iii) Analogia:

– uma função (F_k) que já gerou uma condição ($I(F_k) = r_i$) de regra acabada é preferencialmente experimentada, na formulação de nova hipótese ($I(F_k) = r_j \Rightarrow F_p$*6 ;

---

*1 Exemplo:
$(I(F_k) = r \Rightarrow F_v$
$(I(F_k) = r) \Rightarrow F_p$

*2 Exemplos:
$(I(F_k) = r \wedge x_i) \Rightarrow F_v$
$(I(F_k) = r) \Rightarrow F_p$

*3 Exemplo:
$(F_k) = -7$
$I(F_k)$ = NEGATIVO v INTEIRO

*4 Exemplos:
$(I(F_k)$ = NEGATIVO $\wedge \neq -1) \Rightarrow F_p$
$(I(F_k) = -1) \Rightarrow F_p$

*5 Exemplos:
$(I(F_k)$ = NEGATIVO v INTEIRO$) \Rightarrow F_p$
$\wedge$ ($I(F_k)$ = NEGATIVO $\wedge$ NÃO INTEIRO) $\wedge T_v$

*6 Exemplo:

– reversamente, uma função que já foi experimentada e não gerou uma condição de regra acabada é preterida na formulação de nova hipótese.

IV.3.2.2 – Teste de Regras e de Hipóteses

As duas primeiras regras detectam desequilíbrio, dada uma regra ou hipótese de regra ($C_i \Longrightarrow F_p$) e um contra–exemplo ($C_i \wedge T_v$).

1) Se existe objeto tal que é contra–exemplo para regra da Base de Conhecimento, então:

   – acione **Reformulação da Base de Conhecimento** (item IV.3.2.4) a fim de eliminar a contradição*1;

   – inclua regra e resultado refutado no conjunto de regras sendo investigadas*1 ;

2) Se existe objeto tal que é contra–exemplo para regra sendo investigada ou para hipótese, então registre–o como tal.

As duas últimas regras detectam equilíbrio dada uma regra ou hipótese ($(C_i \Longrightarrow F_p) \wedge (C_i \wedge F_v)$).

$(I(F_k) = \text{NEGATIVO}) \Longrightarrow F_p$
$(I(F_k) = \text{INTEIRO}) \Longrightarrow F_p$

*1 Exemplos:

(i) $(((I(F_k) = \sim \text{INTEIRO v} \sim \text{POSITIVO}) \Longrightarrow F_p) \wedge$
$\wedge((I(F_k) = \sim \text{INTEIRO} \wedge \text{POSITIVO}) \wedge T_v)$
Reformule a Base de Conhecimento para:
$((I(F_k) = \sim \text{POSITIVO}) \Longrightarrow F_p)$
Considere regra investigada:
$((I(F_k) = \sim \text{INTEIRO}) \Longrightarrow F_p)$

(ii) $(((I(F_k) = \sim \text{INTEIRO v} \sim \text{POSITIVO}) \Longrightarrow F_p) \wedge$
$\wedge((I(F_k) = \sim \text{INTEIRO} \wedge \sim \text{POSITIVO} \Longrightarrow T_v)$
Elimine a regra da Base de Conhecimento, considere regra investigada:
$((I(F_k) = \sim \text{INTEIRO v} \sim \text{POSITIVO}) \Longrightarrow F_p)$

(iii) $((((I(F_k) = \text{NEGATIVO v INTEIRO}) \wedge x_i) \Longrightarrow F_p) \wedge$
$\wedge(((I(F_k) = \text{NEGATIVO} \wedge \sim \text{INTEIRO}) \wedge x_i) \wedge T_v)$
Reformule a Base de Conhecimento para:
$((I(F_k) = \text{INTEIRO} \wedge x_i) \Longrightarrow F_p)$
Considere regra investigada:
$((I(F_k) = \text{NEGATIVO}) \Longrightarrow F_p)$

3) Se existe objeto tal que é exemplo de regra da Base de Conhecimento, então considere–a como hipótese (sem alterar a Base de Conhecimento).

4) Se existe objeto tal que é exemplo de hipótese, então registre–o como tal.

IV.3.2.3 – Reformulação de Hipóteses e de Regras Refutadas

Esse conjunto de regras é acionado ao término do ciclo,[*1] se desequilíbrio de regras ou de hipóteses foi detectado durante um ciclo interativo. Têm o objetivo de restaurar o equilíbrio de uma das formas abaixo:

(i) especificando o resultado (r) da parte IF;
(ii) determinando uma condição (instância) que se associada à regra elimina a contradição.

A primeira reformulação (i) ajusta uma formulação de hipóteses que é tão genérica quanto o conjunto de dados investigado permite.

A segunda reformulação (ii) objetiva preservar regras refutadas, discriminando condições limite de sua aplicação. É acionada apenas para regras que permanecem explicando resultados inadequados não explicados pelas demais regras descobertas nesse LOOP ou nos anteriores.

O êxito nessa tentativa provocará a inclusão de regra (descoberta ou reformulada) na Base de Conhecimento (vide item IV.3.2.4).

Dada uma hipótese ou uma regra sendo investigada, para a qual existe contra–exemplo.

1) Se desequilíbrio foi detectado em uma hipótese e o resultado (r) da parte IF $(I(F_k) = r) \Rightarrow F_p)$ é uma disjunção tal que, excluído um ou mais dos termos a contradição é eliminada, então reformule a hipótese com o novo r*[2]

---

*1 Vide item IV.3.3.

*2 $((I(F_k) = \text{INTEIRO v POSITIVO}) \Rightarrow F_p)$
$\wedge ((I(F_k) = \text{INTEIRO} \wedge \text{NEGATIVO}) \wedge T_v)$
Reformule hipótese para:
$((I(F_k) = \text{POSITIVO}) \Rightarrow F_p)$

1.1) Se a função (F_k) da hipótese está incluída em conjunto de mal sucedidas, então retire–a de lá.

2) Se a regra 1 não foi deflagrada, então:

2.1) Se não existe ao menos um exemplo então colete aleatoriamente exemplos, dentre objetos inadequados construídos em LOOPs anteriores

2.2) Para cada termo do resultado (r) da regra ou hipótese refutada, acione as demais regras.

3) Se a regra refutada é inacabada, então gere nova hipótese eliminando a condição associada (hipótese de regra acabada) e acione a regra 4.

4) Se existe hipótese ou regra acabada refutada, então:

4.1) Dado o conjunto de exemplos da regra ou hipótese refutada, elimine todos os que são exemplos de alguma regra utilizada como critério de poda (incluída na Base de Conhecimento) ou de alguma nova regra acabada não refutada construída nesse loop, e

4.1.1) Se após a eliminação ainda existe mais de um exemplo então tente transformá–la numa regra inacabada, acionando regras 5.

4.2) Se não, então acione regra 6.

5) Se existe hipótese ou regra acabada refutada, a ser reformulada para inacabada então:

5.1) Se existe condição tal que associada à parte IF a contradição é eliminada, dados exemplos e contra–exemplos coletados, então, formule nova hipótese de regra inacabada, e

5.1.1) Se não existe contra–exemplo dados objetos construídos em LOOPs anteriores então registre nova hipótese de regra inacabada.[*1]

[*1] Exemplo:
$((I(F_k) = r) \Rightarrow F_p)$

5.2) Se não existe tal condição, abandone a hipótese ou regra.

6) Se a função (F_k) da hipótese ou regra refutada não faz parte de nenhuma regra acabada da Base de Conhecimento, então inclua F_k em conjunto de mal sucedidas[*1]

IV.3.2.4 – Reformulação da Base de Conhecimento

A primeira regra desse conjunto é acionada quando é detectado desequilíbrio de alguma regra da Base de Conhecimento (item IV.3.2.2).

1) Se existe contradição dada uma regra da Base de Conhecimento e um contra–exemplo, então reformule a Base de Conhecimento, excluindo termo(s) do resultado, ou eliminando a regra, a fim de eliminar a contradição.

As demais regras são acionadas ao término de um LOOP, se equilíbrio é detectado em ao menos uma das três circunstâncias:

– quando o equilíbrio é restaurado, após reformulações;

– quando uma nova regra é descoberta;

– quando uma regra inacabada da Base de Conhecimento é submetida e resiste a provas de refutação (passando, nesse caso a ser admitida como critério de poda, apesar de inacabada).

Dada uma hipótese ou regra para a qual não existe contra–exemplo.

2) Se a regra está incluída na Base de Conhecimento, então:

2.1) Se a regra é inacabada e não é utilizada como critério de poda, então (reconsidere e) passe a utilizá–la como tal.

$\wedge\ ((I(F_k) = r \wedge T_v)$
$\wedge\ (((I(F_k) = r) \wedge x_i) \Rightarrow F_p)$

[*1] Uma função investigada que não gerou regra acabada com ao menos um r terá baixa prioridade na formulação de hipóteses (item IV.3.2.1).

2.2) Se não, então assinale para o sujeito (ele deve utilizá–la como poda ou então refutá–la).

3) Se não (não está incluída na Base de Conhecimento), então

3.1) Se a função (F_k) da parte IF da hipótese, é igual à função de alguma regra da Base de Conhecimento, e se o resultado (r) é a única diferença da parte IF, então reformule r, tornando–o uma disjunção, ou incluindo mais um termo na disjunção.

3.2) Se não, inclua a hipótese na Base de Conhecimento, como uma regra (construída ou reconstruída), e

3.2.1) Se e somente se a regra é acabada considere–a como critério de poda.

IV.3.3 – Meta–Regras que Norteiam o Processo Interativo Homem–Máquina

Esse conjunto de regras determina o fluxo de ação do sistema, em cada loop interativo.

Embora esse conjunto de regras seja mais diretamente relacionado à interação homem x máquina, do que meta–heurísticas descritas nos itens anteriores, todas, de uma forma geral, tem importância nesse processo.

Cada meta–heurística que favorece a mais eficiente centração e descentração na ação da máquina, para nova centração na ação do sujeito, intervem no processo construtivo, uma vez que a investigação da máquina deve regular–se de forma a ser:

– breve, para que a atenção do sujeito seja mantida e sua ação não seja interrompida por muito tempo;

– eficaz, para que descobertas ainda que parciais sejam feitas a cada passo.

As duas primeiras heurísticas dirigem a ação do sistema de forma a instigar o sujeito a novas experimentações, e, eventualmente, a construir contra–exemplos para regras descobertas pela máquina.

1) Se o sujeito acha necessário, então relembre regras já construídas e utilizadas como critério de poda.

2) Solicite a participação do sujeito, e

2.1) Se o sujeito produziu um objeto, então:

- registre–o e a avaliação (adequado/não adequado) feita pelo sujeito;
- acione **Teste de Regras e Hipóteses** (item IV.3.2.2), dado o conjunto de regras refutadas nesse LOOP e o conjunto de regras não refutadas;
- torne a acionar regra 1.

2.2) Se há recusa, então:

2.2.1) Se existe(m) objeto(s) produzido(s) pelo sujeito (registrado(s) em 2.1), então acione regra 3.

2.2.2) Se não, então:

2.2.2.1) Se o sujeito desejar e se existe objeto que pode ser construído, e admitido por regras de poda e ainda não produzido na interação, então:

– exiba–o, registre–o e avaliação do sujeito;

– reaplique regra 1.

2.2.2.2) Se não, então atualize arquivo do sujeito com nova Base de Conhecimento e FIM da interação.

A regra 3 consiste de critério heurístico para discriminar condições que podem estar associadas à investigação e/ou novas descobertas do sujeito, e tornar a ação do sistema complementar à do indivíduo.

3) Se o sujeito concluiu sua experimentação, então elimine objetos não admitidos por regras de poda (não refutadas), e:

3.1) Se alguma instância não incluída na base de conhecimento foi experimentada, com alguma adequação, pelo sujeito, então, inclua–a na base de conhecimento, e investigue–a deflagrando as regras 4, 5, 6, 7 e 8.

3.2) Se a regra 3.1 não chegou a ser acionada, então determine a variável investigada pelo sujeito com maior número de instâncias, e investigue cada instância dessa ariável deflagrando as regras 4, 5, 6, 7, 8.

As regras 4, 5, 6, 7 e 8 direcionam a ação do sistema de forma que:

– a formulação de hipóteses parte da experimentação do sujeito, possibilitando uma eventual extração de regras que guiaram a ação do indivíduo (regra 4);

– a experimentação do sistema possibilita a reformulação de hipóteses, e/ou de regras refutadas, e/ou a formulação de novas hipóteses (regras 5, 6 e 8);

– a experimentação do sistema seja guiada por critérios de poda (regra 5) e regulada pela avaliação do sujeito (regras 7 e 8).

4) Dado conjunto de objetos construídos pelo sujeito, que tem em comum a instância x_i (selecionada em 3):

– se não é o caso que todos os objetos são adequados, então acione **Formulação de Hipóteses** (item IV.3.2.1).

5) Se a construção de objetos, tais que,

i) na variável x só é admitida a instância x_i;

ii) para as demais variáveis é selecionada amostra aleatória de instâncias;

iii) regras de poda guiam essa construção;

resulta em conjunto vazio, então relaxe regras de poda, admitindo aleatoriamente objetos (que satisfazem (i) e (ii)).

6) Enquanto o sujeito concordar e houver objeto a ser mostrado (construído

em 5), prossiga com a experimentação de x_i:

- exibida objeto, registre–o e a avaliação (adequado/inadequado) fornecida pelo indivíduo;

- acione **Teste de Regras e de Hipóteses** (item IV.3.2.2) dados conjuntos de hipóteses formuladas, de regras refutadas, e de regras não refutadas e não utilizadas como critério de poda;

- se regras de poda foram relaxadas em 5, então acione **Teste de Regras e de Hipóteses** (item IV.3.2.2) dadas regras utilizadas como poda.

7) Durante a experimentação de x_i, registre resultados (adequado/inadequado) obtidos em cada instância, e só exiba objetos formados por instâncias tais que nenhuma apresente adequação média inferior a x_i.

8) Se a experimentação com x_i terminou, então:

8.1) Se poda foi relaxada em 5 e se algum objeto construído pelo sistema não é admitido por regras de poda (não refutadas) então elimine–o.

8.2) Se não é o caso que todos os objetos (não eliminados) são adequados, então acione **Formulação de Hipóteses** (item IV.3.2.1).

A regra 9.1 aciona **Reformulação de Regras ou Hipóteses** (item IV.3.2.3), se não equilíbrio foi detectado em **Teste de Regras ou Hipóteses** (item IV.3.2.2).

A regra 9.2 aciona **Reformulação da Base de Conhecimento** (item IV.3.2.4) se houve restauração de equilíbrio após reformulações (em IV..3..2.3), e/ou se uma nova regra foi descoberta, e/ou se uma regra inacabada da Base de Conhecimento foi submetida e resistiu a provas de refutação.

9) Se a regra 3 não pode ser reativada, então:

9.1) Se existe não equilíbrio definido por $((C_i \Rightarrow F_p) \wedge (C_i \wedge T_v))$ (onde C_i é a parte IF de alguma regra ou de alguma hipótese), então acione **Reformulação de Regras e/ou Hipóteses** (item IV.3.2.3).

9.2) Se existe equilíbrio definido por $((C_i \Rightarrow F_p) \wedge \sim (C_i \wedge T_v))$ (onde C_i é a

parte IF de alguma regra reformulada ou de alguma hipótese), então acione **Reformulação da Base de Conhecimento** (item IV.3.2.4).

10) Reinicie o LOOP interativo acionando regra 1.

IV.4 – PROCESSO INTERATIVO EM MAIS DE UMA ETAPA E COM MAIS DE UM SUJEITO

Alguns aspcetos adicionais, não definidos até o momento, referem–se a uma interrupção do processo de aprendizagem, e uma retomada posterior.

No momento da interrupção o sistema atualiza arquivo definido para sujeito e procedimento estudado, modificando os conjuntos anteriores, de regras construídas e instâncias aprendidas (que inicialmente é vazio).

Numa nova interação o processo de aprendizagem é retomado no ponto de sua interrupção.

Um dado ainda não definido, e que é afetado por essa interrupção, refere–se a objetos construídos no processo interativo que, não incluídos na Base de Conhecimento, se perdem.

Essa questão é tratada no protótipo implementado, da seguinte forma: antes de ser iniciada cada interação, e num espaço exterior ao ciclo apresentado na Figura IV.1, o sistema gera uma amostra aleatória de objetos, construídos a partir de Base de Conhecimento, exibe–os e registra juntamente com a avaliação do sujeito. Novos objetos construídos, a cada reiniciada do ciclo, por sujeito e sistema, são acrescidos a essa amostra. Esses objetos são utilizados na Formulação de Hipóteses (item IV.3.2.1) no Teste de Regras e Hipóteses (item IV.3.2.2) e na Reformulação de Regras e Hipóteses (item IV.3.2.3).

Outros aspectos referem–se a sujeitos que definem procedimentos a serem estudados ou que experimentam um procedimento já definido.

O sistema se desenvolve junto com cada sujeito, partindo do pressuposto que na primeira interação o sujeito é noviço.

Um sujeito pode experimentar um procedimento definido por ele, ou por outro. Iniciar a interação com seu arquivo, ou com o de outro.

O objetivo dessa abordagem é simultâneamente permitir que:

– o nível de descobertas do sistema não seja incompatível com a possibilidade de compreensão, refutação e novas construções do sujeito;

– descobertas de elementos de um grupo sejam experimentados por outros, sejam essas descobertas relativas a novos procedimentos, ou a resultados de um mesmo procedimento.

Ao ser definido um procedimento, heurísticas básicas específicas devem ser incluídas. O sujeito deve ser estimulado a formular explicitamente, em forma de comandos, regras genéricas inerentes à lógica de seu procedimento e conscientemente consideradas, sobretudo aquelas que dizem respeito ao domínio de variáveis (tais como argumentos que se experimentados conduzem à LOOP, ou geram desenhos excessivamente demorados).

Dessa forma a construção de conhecimento pelo sistema parte de níveis ainda não superados pelo sujeito.

IV.5 – EXEMPLOS DE APLICAÇÃO DO MODELO DE APRENDIZAGEM

Três aspectos distintos, do processo de aprendizagem definido, devem ser considerados:

– modelo genérico descrito na Figura IV.1;

– conjunto de meta–heurísticas que direcionam a ação do sistema;

– operadores utilizados e invariâncias consideradas, na construção de regras.

O processo de aprendizagem descrito na Figura IV.1 e as meta–heurísticas, foram formulados de uma forma genérica, e independente do domínio em que o conhecimento é construído.

As condições que o sistema discrimina (para tornar sua ação complementar à do sujeito e para construir regras), são aspectos do problema considerados na investigação: os operadores e as invariâncias devem ser determinados em função deles. Esses aspectos constituem dados de entrada (definidos local ou globalmente)

do procedimento definido pelo usuário.

A fim de ilustrar possibilidades de aplicação, são apresentados alguns exemplos.

IV.5.1 – Exemplos Compatíveis com Regras Definidas no Item IV.2

1) Exemplos de Procedimentos Definidos por Sujeito(s)

```
TO POLIS :GRAU :PASSO :TOTAL :RECURSÃO
IF :TOTAL = 0 [STOP]
IF :GRAU = 0 [PRINT [SE :GRAU = 0 ENTRO EM LOOP VOU
INTERROMPER] STOP]
RITHT :GRAU
FORWORD :PASSO
IF :RECURSÃO > 0 [POLIS (RUN :TRANSF. GRAU) ➝
(RUN :TRANSF. PASSO) (RUN :TRANSF. TOTAL) (:RECURSÃO – 1)]
POLIS :GRAU :PASSO (:TOTAL – :GRAU) : RECURSÃO.
END
TO ARVORE :PASSO :GRAU :RECURSÃO :TOTAL
IF :TOTAL = 0 [STOP]
FD :PASSO
RG :GRAU
IF :RECURSÃO > 0 [ARVORE (RUN :TRANSF. :PASSO) (RUN :TRANSF. ➝
GRAU) (:RECURSÃO – 1) (RUN :TRANSF. TOTAL]
RG :GRAU x – 1
FD :PASSO x – 1
FD :PASSO/2
RG :GRAU
ARVORE [:PASSO :GRAU :RECURSÃO (:TOTAL – :GRAU)
RG :GRAU x – 1
FD :PASSO/2 x – 1
ARVORE (:PASSO x 3/4) (:GRAU x – 1) :RECURSÃO: ((TOTAL – :GRAU) ➝
x–1)
END
```

2) **Exemplos de Variáveis (Ou Condições ou Aspectos) Investigados**

:GRAU; :PASSO; :TOTAL; :RECURSÃO;
:TRANSF. GRAU; :TRANSF. PASSO; :TRANSF. TOTAL

3) **Exemplos de Valores Associáveis às Variáveis**

:GRAU 36; 45; –120; 135; –144; etc.
:PASSO 30; 35; etc.
:TOTAL 360; 1080; –720; –360; 180; 270 etc.
:RECURSÃO 2; 3
:TRANSF. GRAU 60; –45; (:GRAU x – 1);
(:GRAU – 180); etc.
:TRANSF. PASSO (:PASSO x – 3/5): (:PASSO x – 1); 35; etc.
:TRANSF. TOTAL 360; –360; 120; 180; etc.

4) **Operadores Utilizados na Construção de Expressões (F_k)**

+, –, x, ÷

5) **Invariâncias Retornadas pela Função I (Que Recebe como Argumento uma Expressão Construída)**

MÚLTIPLO DE 360/NÃO MÚLTIPLO DE 360;
POSITIVO/NÃO POSITIVO;
INTEIRO/NÃO INTEIRO;
1;
–1;
0

ou disjunções desses resultados.

O procedimento POLIS foi utilizado para testagem da implementação do protótipo. A avaliação dos objetos (feita pelo sujeito) obedeceu sistematicamente a um mesmo critério: objetos foram avaliados como adequados se e somente se a tartaruga encontrava–se, ao término da figura, no mesmo ponto e na mesma posição em que a iniciara.

Alguns exemplos de regras construídas pela máquina, das formas ($I(F_k)$ =

r) $\Rightarrow$ F, são:

((:TOTAL ÷ : GRAU) = NÃO INTEIRO v NEGATIVO) $\Rightarrow$ F
((:TRANSF. TOTAL ÷ :TRANSF. GRAU)=NÃO INTEIRO v NEGATIVO)$\Rightarrow$ F
(((:TRANSF. TOTAL + :GRAU) = MÚLTIPLO DE 360) $\wedge$ (:RECURSÃO = 1))
$\Rightarrow$ F
((((:TRANSF. TOTAL + :GRAU) x (:TOTAL ÷ :GRAU)) = NÃO MÚLTIPLO
DE 360) $\wedge$ (:RECURSÃO = 1)) $\Rightarrow$ F

Essas construções feitas pela máquina foram verificadas numa fase de depuração do protótipo. Em consequência, são apresentadas, nesse momento, com o objetivo exclusivo de ilustrar tipos de regras construídas. Experimentações e reconstituições de passos são ainda necessárias, para uma ilustração de processo de aprendizagem.

IV.5.2 – Outros Exemplos de Aplicação

Nos exemplos de procedimento definido por usuário, do item anterior, os dados de entrada são valores numéricos, ou expressões que retornam um valor numérico. Os operadores e as invariâncias buscadas pelo sistema são adequados a esses dados.

Um outro exemplo seria um procedimento que recebe listas como dados de entrada. Nesse caso, operadores e invariâncias devem ser redefinidos em função desses dados e do domínio de investigação. Mas as meta–heurísticas e o processo de aprendizagem desenvolvido por sujeitos e máquina permanecem inalterados.

No processo definido, sujeito e máquina investigam, a cada processo interativo, resultados de um mesmo procedimento P, executado com diferentes dados de entrada. Cada dado é argumento de algum comando desse procedimento.

Essa definição é genérica, no sentido que admite desenvolvimento do conhecimento em vários domínios distintos.

Dentre procedimentos que recebem listas como dados de entrada, por exemplo, podem ser investigados resultados de um procedimento P em que os dados são comandos.

Nesse caso, comandos de P (como RUN que equivale a "execute o

comando") recebem comandos como argumentos e cada objeto (ou resultado de P) passa a constituir um novo procedimento. Isto é, a experimentação com diferentes dados de entrada resulta em construir procedimentos diferentes.

Os operadores, e as invariâncias que são buscadas devem ser definidos em função desse objetivo. Alguns exemplos de resultados (retornados por I, a partir de operadores de listas retornados pelas expressões) seriam: existência de comandos após chamada recursiva (recursão não caudal); existência de comandos que se anulam (RIGHT 90, RITHT – 90); etc.

Exemplo de objetos, que constituem novos procedimentos, e do procedimento inicial são apresentados a seguir:

– Procedimento definido por usuário:

```
TO P :GRAU :PASSO :NÍVEL :TOTAL
IF OR :TOTAL > 150 :TOTAL < – 150 (STOP)
RIGHT :GRAU FORWORD :PASSO
RUN :COMANDO.A
RUN :COMANDO.B
RUN :COMANDO.C
RUN :COMANDO.D
```

– Objeto 1: Procedimento semelhante ao primeiro procedimento descrito em IV.5.1.

```
COMANDO.A :"CHAMADA RECURSIVA"
COMANDO.B :"CHAMADA RECURSIVA"
COMANDO.C :[ ]
COMANDO.D :[ ]
```

– Objeto 2: Procedimento semelhante ao segundo procedimento descrito em IV.5.1.

```
COMANDO.A :"CHAMADA RECURSIVA"
COMANDO.B :"CHAMADA RECURSIVA"
COMANDO.C :(FORWORD – :PASSO RITHT – :GRAU)
COMANDO.D :"CHAMADA RECURSIVA"
```

A investigação nsse caso, poderia estar dirigida para testar e avaliar efeitos de substituir, trocar de ordem, eliminar, etc., comandos de um procedimento definido inicialmente.

Regras construídas pelo sistema poderiam ser do tipo: "se existência de recursão não caudal e inexistência de comandos que se anulam, então o objeto é inadequado".

Outro exemplo consiste de investigação em que a parte THEN de regras construídas admite resultados definidos em função de mais de uma variável (diferente de adequação/inadequação de objetos). Nesse caso, nova função I_2 deve ser definida, para abstrações de invariâncias da parte THEN, assim como operadores de resultados recebidos por I_2.

Em ambos os exemplos apresentados nesse item permanecem inalterados o processo descrito na Figura IV.1, assim como as meta–regras que norteiam esse processo e os operadores de mais alto nível aplicados a regras (tais como: "componha funções", "especifique condições", "exclua termos de uma disjunção", etc.).

Nehuma experimentação foi feita, no entanto, com construção de conhecimento para exemplos desses tipos. O objetivo, dessa apresentação é mostrar de forma mais concreta aplicações do modelo de aprendizagem proposto.

CAPÍTULO V

DISCUSSÃO E CONCLUSÕES

Alguns pressupostos gerais nortearam o desenvolvimento desse trabalho:

- a pesquisa na Inteligência Artificial voltada para a substituição, em situações específicas, de sistemas de computação baseados em derivação formal, por sistemas capazes de construir conhecimento, pode encontrar na epistemologia e psicologia genética fundamentação teórica particularmente adequada;

- a construção de conhecimento pelo sujeito cognoscente da Inteligência Artificial e pelo sujeito cognoscente da Psicologia podem ser focalizadas enquanto resultantes de um processo interdependente.

V.1– CONSTRUÇÃO DE CONHECIMENTO EM PROCESSO INTERATIVO HOMEM x MÁQUINA

O modelo de processo de aprendiagem definido visou criar uma situação de interação propícia à aprendizagem do sujeito e da máquina. Não cabe nem ao sujeito nem à máquina manter mais ou menos estagnado o seu conhecimento, enquanto favorece as descobertas do outro. A máquina simula papel de sujeito cognoscente que tanto se desenvolve a partir de resistências impostas externamente, quanto, a cada passo, favorece as aquisições do indivíduo com quem interage.

Embora, à primeira vista, essa questão possa parecer de interesse predominante da Psicologia, supõe–se que ela se relacione diretamente com pesquisas de Inteligência Artificial que focalizam processos de descoberta, uma vez que:

- sistemas capazes de descoberta são úteis, sob o enfoque da Inteligência Artificial, em áreas nas quais não existe conhecimento de equilíbrio estável, e, portanto, não existem especialistas;

- sujeitos não especialistas podem tornar–se progressivamente menos capazes de assimilar e compreender, ou de construir interpretação própria e mais rica para descobertas do sistema, se suas acomodações progressivas não tem

lugar;

— apenas enquanto é capaz de compreender ou de construir interpretação própria para aquisições do sistema, o sujeito está apto a fornecer resistência ou contraprovas;

— sistemas desse tipo enriquecem–se sobretudo com resistências e contraprovas impostas pelo sujeito.

A importância do aspecto interativo homem x máquina foi, de fato, analisada por Lenat e outros pesquisadores* (LENAT, 1983b; LENAT e BROWN, 1984). Mas, no caso do sistema AM, essa questão parece ter sido considerada apenas "a posteriori", e, até mesmo, de certa forma relegada: LENAT (1983b) considera esse aspecto prejudicado, pela falta de especialistas humanos, nas áreas em que o sistema é útil.

A questão que distingue as colocações desse trabalho e as de LENAT (1983b) e LENAT e BROWN (1984) pode ser resumida da seguinte forma: exatamente pelo fato de não existirem especialistas, nos domínios em que tais sistemas são de interesse, é tão importante, para a construção de conhecimento pela máquina, que o processo seja interativo.

Se o sujeito é especialista no domínio estudado, cada descoberta do sistema não constitui real problema para ele: poderá compreender o que o sistema faz, e construir interpretação própria, equivalente ou mais rica. Um exemplo seria a aquisição da noção de número pelo sistema, a partir de conceitos pré–numéricos (com base em Piaget) descrita por LENAT e BROWN (1984). Nessa testagem, o conjunto inicial fornecido ao sistema (operadores, objetos, heurísticas...) era de nível mais primitivo que o possuído pelo autor.

Mas se essa situação artificial não é a existente, se o sujeito não é especialista, supõe–se que, dependendo do grau de seus conhecimentos no domínio, da estruturação e adequabilidade dos modelos que possua, sua participação torne–se progressivamente menos propiciadora do desenvolvimento do sistema.

---

* Ao desenvolver o sistema EURISKO, afim de suprir deficiências observadas no AM, LENAT (1983b) considera, como um dos aspectos mais importantes, a boa interface, permitindo adequada interação. LENAT e BROWN (1984) relatam experiência de Knuth, focalizando efeitos da avaliação do sujeito, referente a descobertas do programa: esse fator é descrito com a maior fonte de sinergia do sistema.

Sem que ele próprio reconstrua seus conhecimentos a partir de sua própria ação, progressivamente pode tornar–se mais difícil a tarefa de compreender as descobertas do sistema e, consequentemente, atribuir uma interpretação própria adequada.

O modêlo construído prevê uma participação ativa do sujeito intercalada com a investigação da máquina.

Um conjunto de meta–heurísticas direciona a ação do sistema, de forma a possibilitar maior conscientização, do que esteja apenas parcialmente compreendido pelo sujeito.

As buscas são orientadas de forma a desenvolver o que é focalizado pelo indivíduo. Novas e diferentes possibilidades do que é experimentado pelo sujeito são apresentadas, e, eventualmente, são construídas contraprovas para hipóteses que dirijam sua ação. Regras mais ou menos acabadas lhe são exibidas, instigando–o a construir refutações.

Em consequência, ao sujeito é oferecida a possibilidade de agir, de refletir sobre sua própria ação, assim como sobre resultados da ação do sistema, eventualmente complementar à sua.

Do ponto de vista da aprendizagem da máquina, meta–heurísticas direcionam–na, num primeiro passo, a procurar extrair regras que porventura guiem a ação do sujeito, com maior ou menor grau de conscientização. A partir de resultados da ação do sujeito, são buscadas hipóteses de novas regras, ou de regras que incluam antigas numa construção mais abrangente.

Após essa investigação, e eventual extração desse conhecimento, a máquina faz novas buscas, a partir do resultado de sua própria ação. Constrói objetos, formula novas hipóteses, reformula as hipóteses ou regras anteriores – especifica ou inclui em construções mais acabadas.

Em síntese, a máquina é direcionada a capacitar o sujeito a construir novos conhecimentos, tornando–o apto a propiciar sua própria aprendizagem.

V.2– EPISTEMOLOGIA E PSICOLOGIA GENÉTICA COMO BASE TEÓRICA DE MODELO DE APRENDIZAGEM

A definição do modelo de processo de aprendizagem apresentada nesse trabalho baseia–se na Teoria Piagetiana. Em vários momentos, decisões sobre como a ação da máquina seria deflagrada e direcionada por meta–heurísticas, foram tomadas à luz desse conhecimento desenvolvido pela psicologia.

Além do processo de equilibração entre assimilação e acomodação a que se refere LENAT (1983a)* , conceitos integrantes do bojo explicativo dessa teoria forneceram suporte.

Uma síntese do processo de aprendizagem é apresentada a seguir.

Um conjunto de meta–regras faz com que:

– a ação da máquina se restrinja a assimilar novos dados e a aplicar o conhecimento adquirido, para construção de novos objetos, enquanto suas regras são adequadas e efetivas;

– a ação seja dirigida para busca de novas regras e/ou reformulações de antigas, a fim de restaurar o equilíbrio, em caso contrário.

Nesse segundo caso, o conhecimento é construído passo a passo, e em função do que já foi descoberto.

Em consequência, nos primeiros ciclos pode ocorrer que: pouco ou nenhum conhecimento seja adquirido, e a máquina permaneça recaindo em erros já cometidos; regras restritas a condições específicas (inacabadas) sejam, inadequadamente, admitidas de forma genérica.

Gradativamente a máquina constrói seu conhecimento. Reiniciando o processo munida de novos recursos, dispõe, cada vez mais, de conhecimentos que dão origem a construções mais completas, e serão por sua vez retomadas num passo posterior.

---

* A ligação entre essa teoria e o trabalho de LENAT (1983a) é feita pela autora; LENAT não faz nenhuma referência explícita a esse embasamento teórico, nos artigos consultados.

De uma forma geral, na busca de novos conhecimentos, e na reconstrução de antigos, pode conseguir:

- iniciar construções, ainda que restritas a condições específicas;
- discriminar condições em que uma regra refutada ainda se aplica;
- admitir uma exceção acoplada a uma regra, se uma construção mais acabada não é feita, e uma condição é constantemente associada a ela;
- identificar regras que são pouco efetivas, e construir uma nova que a abranja.

Num próximo passo, cada uma dessas construções poderá permanecer como tal, ou dar origem a novas construções.

V.3 – CONSIDERAÇÕES FINAIS

O processo de aprendizagem desenvolvido por homem e máquina, tal como descrito no item anterior, foi definido de uma forma genérica, e experimentado numa aplicação específica. Meta–regras que dirigem a ação do sistema e operadores de nível mais alto aplicados a regras independem do domínio em que o conhecimento é construído; operadores de nível mais baixo, aplicados à objetos, foram determinados em função do domínio de aplicação.

A proposta desse trabalho aproxima–se da linha de pesquisa, de Inteligência Artificial, que focaliza processos de descoberta desenvolvidos pela máquina. Mais especificamente da abordagem de LENAT (1982, 1983a, 1983b), enquanto se preocupa com a construção de regras (que direcionam a geração de objetos), ao invés de com a geração de resultados a partir de regras prédefinidas.

Problemas relativos a aspectos metodológicos que conduziram a críticas feitas por RITCHIE e HANNA (1984), ao programa AM, constituem uma das limitações desse trabalho.

Mas supõe–se que o fato de experimentos científicos serem conduzidos,

quando a aprendizagem da máquina não inclui novas descobertas, não implique em que o mesmo método possa ser utilizado na nova proposta da Inteligência Artificial.

Limitações que se verificam, nesse momento, podem caracterizar um passo inicial, e necessário, em estudos sobre processos de descoberta desenvolvidos pela máquina. Podem resultar de falta de conhecimentos já construídos pela Inteligência Artificial, e pela Psicologia.

Embora a construção do conhecimento constitua objeto de estudo genuíno da Teoria Piagetiana, a compreensão e a definição precisa de aspectos de interesse da Inteligência Artificial requerem investigações sob esse novo enfoque.

A escassez de trabalhos desenvolvidos por Piaget e seus seguidores, após o período definido como operacional concreto, fez com que, conceitos piagetianos, a não ser aqueles mais gerais, fornecessem pouco suporte, em estudo anteriormente citado (CUNHA, 1986a).

Vários aspectos do problema colocado e tratado na presente proposta, foram focalizados à luz da Teoria Piagetiana. Mas nem sempre pôde ser feita uma aplicação de conceitos precisamente definidos, à nova situação.

Dificuldades, que se relacionam ao problema, são apontadas por MOSCA e FAGUNDES (1986), no contexto específico de utilização da máquina pelo homem, e por SEMINÉRIO (1987), na interação do sujeito cognoscente com um meio que também é ativo.

MOSCA e FAGUNDES (1986) salientam a necessidade de estudos, que "permitirão o progresso da própria psicologia", sobre "a) certos processos mentais ainda pouco esclarecidos na psicologia cognitiva (como procedimentos cognitivos na resolução de problemas). b) possibilidade de "geração" na criança de formas de raciocínio não evidenciadas em sua interação com o mundo físico". (p. 58).

SEMINÉRIO (1987) discute um aspecto intimamente relacionado à proposta desse trabalho. No processo de aprendizagem definido, coloca–se na possibilidade de transformar o conhecimento e ação resultante, do outro, a condição que enriquece a própria ação e o conhecimento decorrente. Essa dinâmica, e a limitação de formulações teóricas relativas a ela são analisadas por SEMINÉRIO (1987):

"É todavia importante notar que para Piaget a ação – pese à sua concepção dialética – é sempre na realidade, centrada no sujeito, o qual ao transformar o meio por assimilação se transformaria, numa acomodação, isto é, aperfeiçoando os seus instrumentos assimiladores. Entendemos que a tal condição corresponda realmente uma parte do processo. Se a assimilação é realizada dentro da decodificação de informações correspondentes a uma linguagem já presente, é óbio que a sintaxe existente representaria a estrutura assimilativa. No entanto, os paradigmas dessa informação deveriam provir do meio circundante. Se este for inerte, como um meio físico ou químico, será o próprio sujeito que irá promover essa busca de modo autômo e seletivo. Mas à medida que o próprio meio passe a ser constituído por outros seres, dotados de ação, surgiria em nosso entender um sistema de trocas sempre mais centrado nessa interação entendida como ação recíproca que caracterizaria assim uma transformação gradativa do próprio processo e em sentido amplo do meio. (...) Haveria consequentemente alguma redução da autonomia do sujeito individual, cujas sintaxes informacionais estariam sendo preenchidas mediante paradigmas oferecidos por esse tipo de meio também ativo. Esse tipo de troca não nos parece ter sido suficientemente enfatizado por Piaget, que todavia o reconheceu ..." (pp. 27–28).

De fato, embora Piaget e seus seguidores enfatizem a importância do aspecto interativo (como salientado no Capítulo III), nenhuma distinção é encontrada em termos de uma interação com um objeto físico (em que abstrações são feitas a partir desse objeto, ou da ação exercida sobre esse objeto), ou com um outro, co–responsável pelo desenvolvimento do processo (em que abstrações podem ser feitas a partir da ação do outro exercida sobre o objeto e, ainda, das abstrações feitas pelo outro).

A necessidade de formulações teóricas mais precisas, sobre mecanismos cognitivos relativos a essas duas questões – colocadas por MOSCA e FAGUNDES (1986), e por SEMINÉRIO (1987) torna–se ampliada, quando se trata de processo interativo desenvolvido por homem e por máquina.

Face a dificuldades como as descritas, esse estudo limita–se a buscar uma primeira concretização, de pressupostos ainda excessivamente genéricos. Hipóteses não foram precisamente definidas, operacionalizadas, e submetidas a provas de refutação. A partir dessa primeira tentativa, experimentações e reformulações, são necessárias a fim de definir–se de forma mais acabada um processo desenvolvido pela máquina, que simula papel de sujeito que, interativamente com um outro,

constrói seu conhecimento. Um dos aspectos que se mostraram significativos, durante formulações, experimentações e reformulações, que conduziram a essa primeira definição, diz respeito a quão gradativa deve ser a investigação desenvolvida pela máquina, de forma a balancear adequadamente sua ação e ação do sujeito. A consideração desse aspecto parece ser a maior fonte de riqueza, não só para construções da máquina, como para descobertas sobre o próprio processo de descoberta. A proposta de implementação de sistema que constrói seu conhecimento centrando–se na ação do sujeito, na sua própria ação, descentrando–se dela para reiniciar o LOOP (sem que sua investigação perdure por horas consecutivas, antes que alguma intervenção humana seja feita), requer a busca de meta–regras cada vez mais eficientes. Essa busca, por sua vez, pode conduzir a construções, explicitações e reconstruções teóricas sucessivas, sobre mecanismos cognitivos.

Finalmente, mais do que numa ferramenta útil à observação do sujeito cognoscente da psicologia, ou do que em conhecimentos decorrentes desse contexto experimental, que podem ser incorporados e aplicados pela inteligência artificial, é no próprio trabalho interdisciplinar que se coloca a fonte de novos conhecimentos. Constructos hipotéticos inferidos de observações podem ser melhor explicitados, ao direcionarem a formulação de modelos de aprendizagem de máquina, diretamente avaliados, reconstruídos e reformulados, ao ser experimentado o funcionamento do programa, dando origem a um próximo passo no desenvolvimento desse conhecimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ACKERMANN–VALLADÃO, E., et al. (1983) – Formation et actualisation des modeles du sujet en situation de resolution de probleme. **Archives de Psychologie**, Vol. 51, No. 196, pp. 61–70.

ADELSON, B. & SOLOWAY, E. (1985) – The role of human experience in software design. **IEEE Transactions on Software Engineering**, Vol. 11, No. 11, pp. 1351–1360.

BOVET, M. & MONTANGERO, M. (1983) – Au dela des conservations: quelques jalons dans l'oeuvre de Bärbel Inhelder. **Archives de Psychologie**, Vol. 51, No. 196, pp. 106–110.

BROWN, T. & WEISS, L. (1987) – Structures, procedures, heuristics and affectvity. **Archives de Psychologie**, Vol. 55, No. 212, pp. 59–94.

BRUNER, J. (1983) – Function and strategy in thinking: a revisit. **Archives de Psychologie**, Vol. 51, No. 196, pp. 177–181.

CELLERIER, G. (1979) – Structures cognitives et schemes d'action I. **Archives de Psychologie**, Vol. 47, No. 180, pp. 87–106.

CELLERIER, G. (1979) – Structures cognitives et schemes d'action II. **Archives de Psychologie**, Vol. 47, No. 181, pp. 165–176.

CUNHA, M. V. G. C. A. (1986) – Epistemologia genética e inteligência artificial: estratégias cognitivas na solução de problemas. **Arquivos Brasileiros de Psicologia**, Vol. 38, No. 3, pp. 36–57.

CUNHA, M. V. G. C. A. (1986) – Epistemologia genética e inteligência artificial: Linguagens LISP e LOGO. **Arquivo Brasileiros de Psicologia**, Vol. 38, No. 4, pp. 51–56.

INHELDER, B.; ACKERMANN–VALLADÃO, E.; BLANCHET, A.; KARMILOFF–SMITH, A.; KILCHER–HADEDORN, H.; MONTANGERO, J. & ROBERT, M. (1976) – Des structures cognitives aux procédures de découverte. **Archieves de Psychologie**, Vol. 44, pp. 57–72.

INHELDER, B. (1976) – De l'approche structurale à l'approche procédurale: introduction à l'étude des stratégies. Actes du XXI–éme Congrès International de Psychologie, Paris: PUF., pp. 99–118.

INHELDER, B. & PIAGET, J. (1979) – Procedures et structures. **Archives de Psychologie**, Vol. 47, No. 181, pp. 165–175.

GOUSTARD, M.; GRÊCO, P.; MATALON, B. et PIAGET, J. (1974) – La logique des aprentissages, Kraus reprint.

GRÊCO, P. & PIAGET, J. – Aprentissage et conaissance. (1974), Nendeln, Kraus Réprint.

KANT, E. (1985) – Understanding and automating algorithm design. **IEEE Transactions on Software Engineering**, Vol. 11, No. 11, pp. 1361–1374.

KANT, E. & NEWELL, A. (1984) – Problem solving techniques for the design of algorithms. **Information Processing & Manegement**, Vol. 20, No. 1–2, pp. 97–118.

LENAT, D. B. (1982) – The nature of heuristics. **Artificial Intelligence**, Vol. 19, pp. 189–249.

LENAT, D. B. (1983a) – Theory formation by heuristic research. The nature of heuristics II: background and examples. **Artificial Intelligence**, Vol. 21, No. 1/2, pp. 31–59.

LENAT, D. B. (1983b) – EURISKO: a program that learns new heuristics and domain concepts. The nature of heuristics III: program design and results. **Artificial Intelligence**, Vol. 21, No. 1/2, pp. 61–98.

LENAT, D. B. & BROWN, J. S. (1984) – Why AM and EURISKO appear to work. **Artificial Intelligence**, Vol. 23, pp. 269–294.

MOORE, J. & NEWELL, A. (1974) – How can Merlin understand ? In: **Knowledge and Cognition**, New York, L. W., Gregg, Cap. 9, pp. 201–252.

PAPERT, S. (1980) – **Mindstorms: children, computers and powerfull ideas**,

New York, Basic Books Inc. Publishers.

PIAGET, J. (1967) – **O raciocínio na criança**, São Paulo, Record.

SEMINÉRIO, F. L. P. et al. (1987) – Elaboração dirigida – um caminho para o desenvolvimento metaprocessual da cognição humana. **Cadernos do ISOP**, No. 10, pp. 1–41.

SMITH, E. E. (1985) – Cognitive psychology. **Artificial Intelligence**, Vol. 25, pp. 247–253.

STEINER, D. M. & KANT, E. (1985) – The roles of execution and analysis in algoritm design. **IEEE Transactions on Software Engineering**, Vol. 11, No. 11, pp. 1375–1385.

APÊNDICE A

EXEMPLO DE PROCESSO DE DESCOBERTA

DESENVOLVIDO POR MÁQUINA

Com o objetivo de mostrar concretamente construções desenvolvidas pela máquina, apresenta–se um exemplo de utilização do protótipo implementado.

Durante a fase de depuração do sistema – designado EXPERTENTE – utilizou–se o procedimento POLIS definido no Capítulo IV. Experimentações foram conduzidas, com o objetivo de descobrir que condições deveriam ser satisfeitas, para obter–se um resultado determinado: desenhos produzidos por POLIS deveriam iniciar e terminar no mesmo ponto. Objetos foram avaliados como adequados se e somente esse resultado era obtido.

Essas experimentações, embora interrompidas por erros no programa, conduziram a descobertas, humanas e de máquinas.

Essas descobertas são definidas a seguir, e no exemplo apresentado procura–se analisar as construções feitas pela máquina, com base nelas.

1. BASE DE CONHECIMENTO

1.1 – Procedimento Assimilado por EXPERTENTE

```
TO POLIS :GRAU :TOTAL :RECURSÃO
IF :TOTAL = 0 [OP [ ] ]
IF :GRAU = 0 [PRINT [SE GRAU = 0 ENTRO EM LOOP. VOU
INTERROMPER] OP "INADEQUADO]
RIGHT :GRAU
FORWORD (:GRAU/(2 *(:RECURSÃO + 1)))
IF :RECURSÃO > 0 [IF (POLIS (RUN :TRANSF.GRAU) (:RUN
:TRANSF.TOTAL) (:RECURSSÃO – 1)) = "INADEQUADO [OP
"INADEQUADO]]
IF OR :TOTAL > 6.000 :TOTAL < – 6.000 [PRINT [NÃO SEI SE ESTOU EM
LOOP → DEVO INTERROMPER ? RESPONSA S/N] IF READLIST = [S] [OP
"INADEQUADO]]
POLIS :GRAU (:TOTAL – :GRAU) :RECURSÃO
END
```

1.2 – Variáveis de POLIS Investigadas por EXPERTENTE

EXPERTENTE registra, a cada objeto produzido por POLIS, valores associados a: grau, total, recursão transf. grau, transf. total.

Isto é, considera em sua investigação, os valores associados aos três parâmetros de POLIS, na chamada mais externa (grau, total, recursão) e os valores associados a dois parâmetros na próxima chamada recursiva não caudal (transf. grau, transf. total).

Antes de iniciar o processo interativo, conhece uma instância associada a cada uma dessas variáveis:

grau = 90
total = 360
recursão = 1
transf. grau = –90
transf. total = – 360

Nehuma regra acha–se incluída na Base de Conhecimento, no início do processo.

2. CONDIÇÕES FORMULADAS PARA OBTENÇÃO DE UM RESULTADO DEFINIDO

Buscas desenvolvidas junto com EXPERTENTE, durante a fase de sua depuração, conduziram à formalução de uma condição necessária e insuficiente para obtenção de um resultado definido: desenhos produzidos por POLIS iniciassem e terminassem num mesmo ponto.

(i) $((((((\text{total}_0+\text{grau}_1) * \frac{\text{total}_1}{\text{grau}_1}) + \text{grau}_2) * \frac{\text{total}_2}{\text{grau}_2}) +\ldots+ \text{grau}_n) * \frac{\text{total}_n}{\text{grau}_n})=$

$= \mp 360\ y)^{*1}$

---

*1 Exemplos

(1) A condição é satisfeita se os dados de entrada de POLIS são:
:grau=60.; :total=180; :recursão=3; :transf.grau=–45; :transf.total=–180

$((((\text{–}180+\text{–}45) \times \frac{-180}{-45}) +\text{–}45) \times \frac{-180}{-45}) + 60) \times \frac{180}{60}) = 360 \times y$

(2) A condição não é satisfeita se os dados de entrada de POLIS são:
:grau=90; :total=180; :recursão=2; :transf.grau= :grau x – 1;
:transf.total + :total x – 1.

onde:

- grau é um dos parâmetros de POLIS, ao qual é associado o grau virado a cada passo;

- total, outro parâmetro, recebe o somatório de graus que deve ser virado dado um conjunto de chamadas recursivas caudais (a condição de parada é satisfeita quando esse valor é igual a zero);

- o índice, 0...n, é associado ao parâmetro recursão, e corresponde ao nível de chamada recursiva não caudal, do mais interno (0) para o mais externo (nenhuma chamada recursiva não caudal é feita, quando o valor recebido é zero);

- y é um número inteiro.

Contra–exemplos e regras construídas por EXPERTENTE conduziram à constatação de uma exceção, em que a condição (i) não é suficiente à obtenção do resultado desejado. Essa exceção ocorre quando um múltiplo de 360 é obtido exatamente antes da última multiplicação ser efetuada em (i).

(ii) $((((total_0+grau_1) * \frac{total_1}{grau_1} + grau_2) * \frac{total_2}{grau_2}) +...+ grau_n) = \overline{+}\ 360\ y$

As condições definidas em (i) e em (ii) conduziram à formulação de regras, na busca efetuada[*1]

$V \triangleright C_i$

$C_{ii} \triangleright F$

$V \triangleright C_i \wedge \sim C_{ii}$

$((((180 + -90) \times \frac{-180}{-90}) + 90) \times \frac{180}{90}) = 540$

*1 Essas regras podem ser ou não verdadiras, acabadas ou restritas a condições específicas. O leitor é convidado a construir contra–exemplos para elas.

onde

V = desenho produzido por POLIS em que início e fim correspondem a um mesmo ponto

F = desenho em que esse resultado não é obtido

C_i e C_{ii} = condições definidas, respectivamente em (i) e em (ii).

Uma vez que a investigação desenvolvida por EXPERTENTE desconsidera valores associados a parâmetros de POLIS, após níveis recursivos caudais menores do que n–1, partiu–se da suposição, na construção do exemplo apresentado a seguir, que nenhuma formulação genérica, a qualquer número de níveis recursivos, poderia ser construída pela máquina.

3. EXEMPLO DE PROCESSO DE DESCOBERTA DESENVOLVIDO POR MÁQUINA

A fim de mostrar concretamente o processo de descobertas desenvolvido por máquina, numa investigação desse tipo, é apresentado o exemplo a seguir.

Esse exemplo, no entanto, não pode ser visto como um processo em que homem e máquina desenvolvem–se interativamente, sem algumas restrições.

Nessa investigação, sujeito e máquina partem com níveis de equilíbrio cognitivo bastante distintos. Do ponto de vista humano, regras como as descritas já haviam sido previamente construídas. Do ponto de vista de máquina, nenhuma regra achava–se incluída na Base de Conhecimento, no início do processo.

Uma vez que EXPERTENTE é direcionado a desenvolver busca complementar à do sujeito, eventualmente extraíndo regras que norteiam a busca humana, uma situação como a descrita pode conduzir a uma simplificação do processo construtivo da máquina.

A fim de controlar, ao menos parcialmente, interferências que conduzem à essa simpliicação, desenvolveu–se uma interação em que a participação humana é mínima: a cada loop, a experimentação humana limita–se à construção de no máximo dois objetos, sendo que nenhum inadequado.

Em consequência, no direcionamento da investigação da máquina são determinantes novas instâncias utilizadas pelo sujeito; as situações de desequilíbrio na construção de objetos, desencadeadoras de formulações de novas hipóteses, ocorreram apenas a partir da experimentação da máquina.

Os dados apresentados a seguir, consistem dos objetos construídos na experimentação humana, e das regras construídas pela máquina, numa sequência de passos.

Tabela A.1 – Instâncias associadas às variáveis de POLIS, na experimentação humana, numa sequência de passos[*1]

VARIÁVEIS	PASSOS						
	0[*2]	.1	.2	.3	.4	.5	.6
Grau	90	**– 60**	– 60	**30**	30,30	**– 30**	– 60
Total	360	**–360**	–360	**240**	**120,120**	**–120**	–120
Recursão	1	1	1	1	1, 1	2	1
Transf.grau	– 90	– 90	**60**	60	60, 60	60	**240**
Transf.total	–360	–360	**360**	**60**	60,**240**	240	240

Como resultado do desenvolvimento de todo processo descrito no Capítulo IV, a Base de Conhecimento de EXPERTENTE altera–se contínua e gradativamente. A tabela apresentada a seguir mostra a frequência de hipóteses formuladas, não refutadas ou refutadas (e reformuladas ou eliminadas), numa sequência de passos (loops).

[*1] O negrito indica novas instâncias experimentadas, discriminadas por EXPERTENTE para direcionamento de sua investigação.

[*2] No passo zero apresenta–se instâncias já existentes na Base de Conhecimento, no início do processo.

Tabela A.2 – Número de hipóteses formuladas numa sequência de passos

PASSOS	HIPÓTESES		
	NÃO REFUTADAS	REFUTADAS	
		REFORMULADAS	ELIMINADAS
1	1	1	–
2	1	1	–
3	3	–	4
4	2	–	4
5	3	1	8

As hipóteses formuladas e não refutadas a cada passo, passam a ser utilizadas como critério de poda. A tabela apresentada a seguir descreve essas regras, e indica sua manutenção ou não (reformulação ou eliminação) em passos posteriores à construção.

Tabela A.3 – Construção, eliminação e reformulação de regras de poda, numa sequências de passos

	CONDIÇÕES DE REGRAS CONSTRUÍDAS DAS DA FORMA $C_i \triangleright F_P$	PASSOS CONS–TRUÇÃO	REFOR–MULAÇÃO	ELIMI–NAÇÃO
1)	I(grau * total) = negativo	1	–	–
2)	I(transf.grau * transf. total) = negativo	2	–	–
3)	I(transf.total + grau) = 0	3	–	–
4)	I(total/transf.total) = não inteiro	3	–	4
5)	I(total/grau) = não inteiro	3	–	–
6)	I(transf.total/grau) = não inteiro	4	–	–
7)	I(total/grau) * (transf. total + grau) = não múltiplo de 360	4	5	–
8)	I(total/grau)/recursão) = 1	5	–	–
9)	I((transf./total+grau) * recursão) = múltiplo de 360	5	–	–
10)	I((total/transf.grau)/re–cursão) = 1	5	–	–

Cada uma dessas condições consiste da parte IF de uma regra construída. A parte THEN da regra consiste de inadequação prevista de objeto ($C_i \triangleright F_P$). Uma regra construída é utilizada como critério de poda na geração de um novo objeto, enquanto nenhum contra–exemplo é gerado na experimentação de sujeito e/ou máquina. Se existe um contra–exemplo construído, a regra é eliminada ou reformulada como descrito no Capítulo IV. Se reformulada para uma regra inacabada, definida como restrita a condição específica ($(C_i \wedge x_j) \triangleright F_P$), deixa de

ser utilizada como critério de poda. Nesse caso, em algum loop posterior é reincorporada à poda, se é submetida e resiste a provas de refutação em condição específica discriminada (x_j).

Dentre as regras construídas pela máquina e apresentadas na Tabela A.3, encontram–se regras verdadeiras, falsas e verdadeiras em algumas condições.

As regras (1), (2) e (5) são verdadeiras. Se alguma dessas condições não é satisfeita, a condição de parada de POLIS não é encontrada nunca, POLIS entra em loop.

A regra (4) é falsa, e um contra–exemplo construído no passo 5 conduziu EXPERTENTE a eliminá–la.

A regra (6) é falsa, mas nenhum contra–exemplo chegou a ser construído por homem ou máquina, até o ponto em que a experimentação foi conduzida*1. Uma situação de desequilíbrio relativa a essa regra não chegou a ser instaurada ($C_6 \triangleright F_P \wedge \sim (C_6 \triangleright F_V)$), e, em consequência, o processo que conduziria à sua eliminação (ou reformulação) não foi acionado.

As demais regras construídas pela máquina não são compreendidas de forma tão imediata. Ou, ao menos, não são tão facilmente interpretadas, dependendo do conhecimento adquirido, no domínio do problema, por quem as avalia. De uma forma geral relacionam–se às condições (i) e (ii) formuladas anteriormente.

As regras (3) e (7) são parcialmente verdadeiras, com base na condição (i), que se não satisfeita o resultado desejado não é obtido.

$$\text{(i)} \quad ((((((\text{total}_0 + \text{grau}_1) * \frac{\text{total}_1}{\text{grau}_1}) + \text{grau}_2) \text{ x } \frac{\text{total}_2}{\text{grau}_2}) + \ldots + \text{grau}_n) \text{ x}$$

$$\text{x } \frac{\text{total}_n}{\text{grau}_n}) = (\overline{+}\ 360) * y$$

A regra (7) construída pela máquina restringe–se a considerar os dois níveis recursivos não caudais mais externos da condição (i). Substituindo tem–se:

---

*1 Contra–exemplo para a regra (6): grau = 135; total = 270; recursão = 1; transf.grau = 45; transf.total = 45.

Regra (7): $((\text{total}_{n-1} + \text{grau}_n) \times \frac{\text{total}_n}{\text{grau}_n} \neq (\overline{+}\ 360 * y)) \triangleright F_p$

A regra (7) foi refutada no loop 5 por objeto construído por sujeito (vide Tabela A.1). EXPERTENTE reformulou–a para regra inacabada, associando–a a duas condições específicas. Inadequação de objeto passa a ser prevista, se a condição (7) não é satisfeita, quando à variável recursão é associado o valor 1 e/ou à variável transf.total é associado 360.

regra (7'): (I((total/grau) * (transf.total + grau) = não múltiplo de 360) $\wedge$ ((recursão = 1) v (transf.total = 360)) $\triangleright F_p$

De fato, com base na condição (i) a regra (7) construída pela máquina é verdadeira em dois casos específicos.

– se recursão é igual a 1:

$(((\text{total}_0 + \text{grau}_1) \times \frac{\text{total}_1}{\text{grau}_1}) \neq (\overline{+}\ 360 * y)) \triangleright F$

– se a variável transf.total é associada a constante 360 (total de todas as chamadas recursivas não caudais de nível menor do que n é igual a 360)

$(((((((360 + \text{grau}_1) \times \frac{360}{\text{grau}_1}) + \text{grau}_2) \times \frac{360}{\text{grau}_2}) \ldots + \text{grau}_n) \times$

$\times \frac{\text{total}_n}{\text{grau}_n} \neq (\overline{+}\ 360*y)) \triangleright F$

No loop 6, a regra (7) reformulada para inacabada não foi usada como critério de poda, foi submetida e resistiu a provas de refutação, e foi reincorporada à poda (restrita às duas condições discriminadas).

A regra (3) corresponde a um caso particular em que a condição (i) não é satisfeita, mas igualmente restrita a condição específica.

Se a condição (3) é satisfeita e recursão é igual a 1, o resultado obtido em (i) é zero:

$$(\text{total}_0 + \text{grau}_1 = 0) \triangleright (\text{total}_0 + \text{grau}_1) * \frac{\text{total}_1}{\text{grau}_1} = 0$$

A reformulação dessa regra não chegou a ser efetuada por EXPERTENTE, uma vez que uma situação de desequilíbrio não chegou a ser instaurada. Apenas a construção de contra–exemplo acionaria sua reformulação e eventual transformação para regra inacabada, da forma:

Regra (3') $((I(\text{transf.total} + \text{grau}) = 0) \wedge (\text{recursão} = 1)) \triangleright F_p$

As regras construídas pela máquina a partir do passo (5) foram totalmente inesperadas. Dito de outra forma, consistem de descobertas que ainda não haviam sido atingidas na construção humana.

A regra (9) é verdadeira, com base na condição definida em (ii), se recursão é igual a 1. Substituindo tem–se:

$(I(\text{total}_0 + \text{grau}_1) \times 1 = \text{múltiplo de } 360) \triangleright F_p$

A formulação de EXPERTENTE, no entanto, é genérica a qualquer nível recursivo. Até esse ponto da experimentação nenhum contra–exemplo foi construído para essa formulação de EXPERTENTE. Se a condição da regra (9) implica em que a condição (ii) seja verdadeira, então (com base em (ii) ela é verdadeira

$C_{ii} \triangleright F$
$C_9 \triangleright C_{ii}$
$C_9 \triangleright F$

Para a regra (10) e para a regra (9), da mesma forma, não chegou–se a encontrar contra–exemplo. Ambas parecem implicar em que a condição (i), necessária ao resultado investigado, seja falsa. E, nesse caso, elas são corretas na previsão de objetos inadequados

$\sim C_i \triangleright F_v$
$C_8 \vee C_{10} \triangleright \sim C_i$
$C_8 \vee C_{10} \triangleright F_v$

Em síntese, as regras (8), (9) e (10) não foram integralmente compreendidas. Se elas são verdadeiras ou não, consiste de conhecimento a ser adquirido pela continuação desse processo de descoberta, desenvolvido por homem e máquina.

APÊNDICE B

LISTAGEM DO PROTÓTIPO

Neste anexo, apresenta–se a listagem da primeira versão de EXPERTENTE, e algumas limitações dessa versão.

As suas limitações mais graves decorrem da linguagem utilizada – LOGO (Dr. LOGO). Referem–se ao espaço de memória e ao tempo de processamento

1. O espaço de memória de Dr. LOGO, nos microcomputadores em que EXPERTENTE foi experimentado (IBM–PC–AT/80386, IBM–XT/8086) é de cerca de 155.000 "bytes". EXPERTENTE ocupa cerca de 130.000 "bytes". Em consequência do reduzido espaço disponível para sua execução, interrrupções ocorreram frequentemente.

Carregamento parcial do sistema e acesso periódico à disco não mostrou–se solução para esse problema. Com esse recurso, a memória virtual aparentemente indica espaço suficiente à execução de EXPERTENTE. Mas a ausência de comando que compacte a memória utilizada, resulta numa fragmentação do espaço disponível: formado por blocos dispersos, não admite o carregamento de procedimentos buscados em disco.

2. O tempo de processamento de EXPERTENTE implementado em Dr. LOGO mostrou–se adequado a uma interação homemmáquina no IBM PC–AT mas inadequado no IBM PC–XT. Embora sejam necessários afinamentos do protótipo, com o objetivo de agilizá–lo, a lentidão de execução de Dr. LOGO parece inviabilizá–la para sistemas relativamente complexos (ao menos se são utilizados processadores não tão rápidos quanto do IBM PC–AT).

A utilização de EXPERTENTE requer a solução desses problemas. Uma possibilidade consiste de sua versão para dialeto LISP, tal como MU–LISP.

Um segundo tipo de limitação refere–se a simplificações existentezs no proótipo.

1. A rotina INICIAL determina variáveis que serão investigadas dentre as existentes em procedimento qualquer, definido por usuário. Essa rotina supõe um número máximo (8) de variáveis e não admite variações nas designações dessas variáveis (grau, passo, total, recursão, transf.grau, transf.passo, transf.total, transf.rec.). Dentre elas, no máximo quatro variáveis constituem parâmetros do procedimento. Essas restrições,

estabelecidas com o objetivo de simplificar a primeira versão de EXPERTENTE são desnecessárias e devem ser eliminadas.

2. Outros aspectos, que interferem na interface, relativos a clareza de mensagens impressas na tela (tais como regras descobertas) não foram suficientemente cuidados.

3. Finalmente, dentre as meta–heurísticas descritas no Capítulo IV, a de número 3.2 do item IV.3.4 não está incluída na versão apresentada. Em consequência, na discriminação de condições que tornam a ação da máquina complementar à ação do sujeito, são consideradas apenas novas instâncias experimentadas pelo indivíduo (regra 3.1 do mesmo item).

A listagem de EXPERTENTE é apresentada a seguir.

```
TO RECOLHE :CONT
PR [] PR JUNTA [DEFINA] DECODIGO :CONT MAKE "L READLIST
IF :L = [] [PR [EXPERIMENTE ALGO ESTUDADO] OP RECOLHE
:CONT] [IF EQUALP CHECA :L "ERRO [PR [ACEITO UM NUMERO OU
UMA OPERACAO COM - - :GRAU :PASSO :TOTAL :RECURSAO
:TRANSF.GRAU :TRANSF.GRAU :TRANSF.PASSO :TRANSF.TOTAL
:TRANSF.REC] OP RECOLHE :CONT]]
LABEL "LOOP
PR JUNTA [- - - - - - - - - -] :L PR [( TECLE NAO SE VOCE
QUER MUDAR OU TECLE RETURN )]
MAKE "L. READLIST
IF OR :L. = [NAO] :L. = [N] [OP RECOLHE :CONT]
OP :L
END

TO CHECA :REC
IF :REC = [] [OP "T]
MAKE "A1 FIRST :REC
IF OR (OR (OR NUMBERP :A1 EQUALP ":GRAU :A1) (OR EQUALP
":PASSO :A1 EQUALP ":TOTAL :A1)) (EQUALP ":RECURSAO :A1)
[OP CHECA BF :REC]
IF OR (OR :A1 = ":TRANSF.GRAU :A1 = ":TRANSF.PASSO) (OR
:A1 = ":TRANSF.TOTAL :A1 = ":TRANSF.REC) [OP CHECA BF
:REC]
IF PRIMITIVEP FIRST :REC [OP CHECA BF :REC]
OP "ERRO
END

TO PONTO :PAR.
MAKE "L []
REPEAT (COUNT :PAR.) [MAKE "L LPUT WORD ": FIRST :PAR. :L
MAKE "PAR. BF :PAR.]
OP :L
END

TO EXCLUI :A :B
IF :B = [] [OP []]
IF :A = FIRST :B [OP EXCLUI :A BF :B] [OP FPUT FIRST :B
EXCLUI :A BF :B]
END

TO ABSOLUTO :N
IF :N < 0 [OP :N * -1] [OP :N]
END

TO ELIMINACAO :TOTAL :I :O
MAKE "V. [1]
LABEL "LOOP
IF :TOTAL = [] [OP []]
```

```
IF :I = [] [IF NOT :0 = [] [MAKE "V. [0] MAKE "I :0 MAKE
"0 [+ 1]] [OP :TOTAL]]
MAKE "II FIRST :I
LABEL "LOOP2
IF :II = [] [MAKE "I BF :I GO "LOOP]
MAKE "TAB TABELA -1 LPUT :V. FIRST :II FIRST :TAB LAST
:TAB [] [] MAKE "TOTAL ELIM LPUT :V. FIRST :II :TOTAL []
MAKE "II BF :II GO "LOOP2
END

TO INCLUIDO2 :N11 :C1 :NI
IF :N11 = [] [OP :NI]
MAKE "N111 FIRST FIRST :N11
IF AND MEMBERP FIRST :N111 :C1 MEMBERP LAST :N111 :C1 [IF
[] = EXCE BF FIRST :N11 :C1 [MAKE "NI LPUT FIRST :N11
:NI]]
OP INCLUIDO2 BF :N11 :C1 :NI
END

TO EXCE :QUAL :C1
IF :QUAL = [] [OP []]
IF MEMBERP FIRST :QUAL :C1 [OP "T]
OP EXCE BF :QUAL :C1
END

TO PALAVRA :LISTA
MAKE "L "A
REPEAT (COUNT :LISTA) [MAKE "L WORD :L FIRST :LISTA MAKE
"LISTA BF :LISTA]
OP BF :L
END

TO EQUIVALE :F.
IF :F. = [] [OP []]
IF FIRST :F. = "( [OP LPUT ") EQUIVALE BF :F.]
IF FIRST :F. = ") [OP LPUT "( EQUIVALE BF :F.]
OP LPUT FIRST :F. EQUIVALE BF :F.
END

TO EQUIVALEMEMBRO :F. :OBSERVA
IF OR MEMBERP "DIVIDIDO FIRST :F. MEMBERP "- FIRST :F. [OP
"F]
IF (COUNT :F.) > 1 [OP "F]
IF MEMBERP (FPUT EQUIVALE FIRST :F. []) :OBSERVA [OP "T]
[OP "F]
END

TO CL
CLEARSCREEN
```

```
END

TO PONDERA :PONTOS :PE
CT TEXTSCREEN
IF :PE = 1 [PR JUNTA [EU APRENDI COM VOCE :] DECODIGO :PE
PR [( TECLE RETURN )] MAKE "L READLIST OP :PONTOS + 2]
IF OR :PE = 3 :PE = 7 [PR JUNTA [EU APRENDI COM VOCE UMA
NOVIDADE :] JUNTA DECODIGO :PE [. ISSO E' OTIMO ! DESCO-
BERTAS NESSE PARAMETRO PODEM LEVAR A DESENHOS MUITO INTE-
RESSANTES] PR [( TECLE RETURN )] MAKE "L READLIST OP
:PONTOS + 8]
IF OR OR :PE = 4 :PE = 5 OR :PE = 6 :PE = 8 [PR JUNTA [EU
APRENDI COM VOCE UMA NOVIDADE :] JUNTA DECODIGO :PE [ESSA
DESCOBERTA E' IMPORTANTE] PR [( TECLE RETURN )] MAKE "L
READLIST OP :PONTOS + 5]
END

TO CRESCE
RECYCLE
ERASE "TENTE
RECYCLE
ERASE "INICIAL
RECYCLE
MAKE "CONHECIDOUM LAST TEXT "CONHECIDO
IF MEMBERP (WORD :MODELO :NOME) LAST TEXT "USUAR [(MAKE
"UML (WORD " WORD :MODELO :NOME)) (LOAD :UML) (RECYCLE)
(RECYCLE) (RECYCLE) (MAKE "U TEXT :UML) (ERASE :UML)
(RECYCLE)] [MAKE "U TEXT "UM]
MAKE "VISTO FIRST BF :U
MAKE "PODA FIRST BF BF :U
MAKE "INST FIRST BF BF BF :U
MAKE "DUVIDA FIRST BF BF BF BF :U
MAKE "U []
RECYCLE
LOAD "ESTUDA0
CRESCE2
END

TO EXIBE
MAKE "REALN [] MAKE "TC0 [] MAKE "PER "F
MAKE "COMBESP GERA [?] 2220 :VISTO
IF :COMBESP = [] [STOP]
MAKE "TAB [[[] [] [] [] [] [] [] []] [[] [] [] [] [] [] []
[]]] MAKE "RN []
MAKE "T: (COUNT :COMBESP) MAKE "POD1 REFUTUM :PODA [[]]
MAKE "CONT: 0
LABEL "LOOP
MAKE "R RANDOM :T:
MAKE "COMB ITEM (:R + 1) :COMBESP MAKE "COMBESP ELIM :COMB
```

```
:COMBESP [] MAKE "T: :T: - 1
IF (REDUZ :COMB :TAB 0.5) = "FRACASSO [GO "FORA]
IF AND AND NOT :POD1 = [] NOT :POD1 = [[]] AND NOT :POD1 =
[[] []] :PER = "T [MAKE "I REFUTACAO1 FIRST :POD1 LAST
:POD1 LPUT :COMB [] [[]] [[]] [[]] "UNICA IF :I = "FRAC
[GO "FORA]]
IF OR :CONT: = 0 NOT (:CONT: / 8) = INT (:CONT: / 8) [GO
"FORA1]
LABEL "LOOPI
PR [QUER QUE EU MOSTRE OUTROS DESENHOS? RESPONDA S / N]
MAKE "L READLIST
IF OR :L = [N] :L = [NAO] [STOP]
IF AND NOT :L = [S] NOT :L = [SIM] [PR [NAO ENTENDI.] GO
"LOOPI]
MAKE "CONT: :CONT: + 2
LABEL "FORA1
MAKE "COMB PARTE :MODELO :COMB
IF (LAST :COMB) = [1] [MAKE "TC1 LPUT BL :COMB :TC1] [MAKE
"TC0 LPUT BL :COMB :TC0]
MAKE "TAB TABELA 1 :COMB FIRST :TAB LAST :TAB [] []
MAKE "CONT: :CONT: + 1
LABEL "FORA
IF NOT :COMBESP = [] [GO "LOOP]
END

TO TENTE
ERNS
  RECYCLE
PR [JA CONHECO OS PROGRAMAS :]
LOAD "PROGRAMA
PR LAST TEXT "PROGRAMA PR [] PR []
PR [VOCE PODE EXPERIMENTAR UM PROCEDIMENTO QUE EU JA' CO-
NHECO OU QUALQUER OUTRO QUE VOCE TENHA DEFINIDO E ESTEJA
NO ESPACO DE TRABALHO] PR [] PR [] PR [DIGA O NOME DO PRO-
GRAMA QUE VAMOS EXPERIMENTAR.]
LABEL "LM
MAKE "MODELO READLIST
IF :MODELO = [] [PR [NAO ENTENDI] GO "LM] [MAKE "MODELO
FIRST :MODELO]
CT TEXTSCREEN
IF NOT MEMBERP :MODELO LAST TEXT "PROGRAMA [IF INICIAL =
"OK [GO "FORA1] [STOP]]
LOAD (WORD " :MODELO)
MAKE "TERMINO FIRST LAST TEXT "TERMINO
MAKE "VAZIO LAST TEXT "VAZIO
LABEL "LN
PR [] PR [DIGA SEU NOME] MAKE "NOME READLIST
IF :NOME = [] [GO "LN] [MAKE "NOME FIRST :NOME]
PR [] PR []
```

```
MAKE "L. TEXT (WORD " :MODELO)
MAKE "N (COUNT :L.) - 1
PR FPUT :MODELO PONTO FIRST :L.
REPEAT :N [PR FIRST BF :L. MAKE "L. BF :L.] PR [END] PR []
CRESCE
END

TO INVERSO :L :F
LOCAL "INV
LOCAL "L2
MAKE "L2 [[[1]] [[-1]] [[0]] [[VG.NEG]] [[VG.POS]]
[[INT.MULT360.NEG]] [[INT.MULT360.POS]] [[VG.MULT360.NEG]]
[[VG.MULT360.POS]] [[INT.DIVISOR360.NEG]]
[[INT.DIVISOR360.POS]] [[VG.DIVISOR360.NEG]]
[[VG.DIVISOR360.POS]] [[INT.NEG]] [[INT.POS]]]
IF (COUNT :F) = 2 [REPEAT (COUNT :L2) [MAKE "L2 LPUT (LPUT
LAST :L FIRST :L2) BF :L2]]
IF :L = [[]] [OP :L2]
MAKE "INV []
LABEL "LOOP
MAKE "L21 FIRST :L2
IF NOT MEMBERP :L21 :L [MAKE "INV LPUT :L21 :INV]
IF NOT BF :L2 = [] [MAKE "L2 BF :L2 GO "LOOP] [OP :INV]
END

TO ELIM :CB :COMB :FRENTE
IF :COMB = [] [OP :FRENTE]
IF NOT EQUALP :CB FIRST :COMB [MAKE "FRENTE LPUT FIRST
:COMB :FRENTE]
OP ELIM :CB BF :COMB :FRENTE
END

TO NINVAR :R.
LOCAL "L2
LOCAL "L
LOCAL "RES
MAKE "L2 [[[INT.MULT360.POS]] [[INT.MULT360.NEG]]
[[INT.DIVISOR360.POS]] [[INT.DIVISOR360.NEG]]
[[VG.DIVISOR360.POS]] [[VG.DIVISOR360.NEG]] [[INT.POS]]
[[INT.NEG]] [[VG.POS]] [[VG.NEG]] [[VG.MULT360.POS]]
[[VG.MULT360.NEG]] [[1]] [[-1]] [[0]]]
MAKE "L :L2
MAKE "RES :R.
IF OR MEMBERP [MULT.] :RES MEMBERP [NAONAOMULT.] :RES [MA-
KE "L ( ELIM ITEM 1 :L2 ( ELIM ITEM 2 :L2 :L2 [] ) [] )
MAKE "L ELIM ITEM 12 :L2 ELIM ITEM 11 :L2 :L [] []]
IF OR MEMBERP [DIV.] :RES MEMBERP [NAONAODIV.] :RES [MAKE
"L ELIM ITEM 3 :L2 ELIM ITEM 4 :L2 ELIM ITEM 5 :L2 ( ELIM
ITEM 6 :L2 :L [] ) [] [] [] MAKE "L ELIM [[1]]   ELIM [[-1]]
```

```
:L [] [][]
IF OR MEMBERP [NAOMULT.] :RES OR MEMBERP [NAODIV.] :RES
MEMBERP [NAODIV.OU.MULT.] :RES [MAKE "L ELIM ITEM 7 :L2
ELIM ITEM 8 :L2 ELIM ITEM 9 :L2 (   ELIM ITEM 10 :L2 :L [] )
[] [] [][]
IF MEMBERP [NAODIV.] :RES [MAKE "L ELIM ITEM 1 :L2 ( ELIM
ITEM 2 :L2 :L [] ) [] MAKE "L ELIM ITEM 12 :L2 ELIM ITEM
11 :L2 :L [] [][]
IF MEMBERP [NAOMULT.] :RES [MAKE "L ELIM ITEM 3 :L2 ELIM
ITEM 4 :L2 ELIM ITEM 5 :L2 ( ELIM ITEM 6 :L2 :L [] ) [] []
[] MAKE "L ELIM [[1]] ELIM [[-1]] ELIM [[0]] :L [] [] [][]
IF MEMBERP [NAOINTEIRO] :RES [MAKE "L ELIM ITEM 5 :L2 ELIM
ITEM 6 :L2 ELIM ITEM 9 :L2 ( ELIM ITEM 10 :L2 :L [] ) []
[] [] MAKE "L ELIM ITEM 11 :L2 ELIM ITEM 12 :L2 :L [] [][]
IF OR MEMBERP [INTEIRO] :RES MEMBERP [NAONAOINTEIRO] :RES
[MAKE "L ( ELIM ITEM 1 :L2 ( ELIM ITEM 2 :L2 ( ELIM ITEM 3
:L2 ( ELIM ITEM 4 :L2 ( ELIM ITEM 7 :L2 ( ELIM ITEM 8 :L2
:L [] ) [] ) [] ) [] ) [] ) [] ) MAKE "L ELIM [[1]] ELIM
[[-1]] :L [] [][]
IF OR MEMBERP [NEG.] :RES MEMBERP [NAOPOS.] :RES [MAKE "L
ELIM ITEM 2 :L2 ELIM ITEM 4 :L2 ELIM ITEM 6 :L2 ELIM ITEM
8 :L2 ELIM ITEM 10 :L2 :L [] [] [] [] [] MAKE "L ELIM ITEM
12 :L2 :L [] MAKE "L ELIM [[-1]] :L [][]
IF OR MEMBERP [POS.] :RES MEMBERP [NAONEG.] :RES [MAKE "L
ELIM ITEM 1 :L2 ELIM ITEM 3 :L2 ELIM ITEM 5 :L2 ELIM ITEM
7 :L2 ELIM ITEM 9 :L2 :L [] [] [] [] [] MAKE "L ELIM ITEM
11 :L2 :L [] MAKE "L ELIM [[1]] :L [][]
MAKE "L INVERSO :L [[]]
IF [[0]] = :RES [MAKE "L LPUT [[0]] :L]
IF [[1]] = :RES [MAKE "L LPUT [[1]] :L]
IF [[-1]] = :RES [MAKE "L LPUT [[-1]] :L]
OP :L
END

TO TIRA :F :T
LOCAL "L
IF OR (OR :T = [] :T = [[]]) :T = [[] []] [OP [[] []]]
IF :F = FIRST FIRST :T [OP LIST BF FIRST :T BF LAST :T]
[MAKE "L TIRA :F LIST BF FIRST :T BF LAST :T OP LIST FPUT
FIRST FIRST :T FIRST :L FPUT FIRST LAST :T LAST :L]
END

TO ABSTRAI :PARAM :UMA :Z :CONHL :NIVEL :TIPO :LIMITE
:FORM1 :FORM2
MAKE "LISTARUIM []
IF :NIVEL = 5 [OP "FRAC]
LABEL "INICIO
IF NOT :FORM1 = [] [MAKE "PARAM LIST FIRST :PARAM FIRST
:FORM1 MAKE "FORM1 BF :FORM1]
```

```
IF :PARAM = [] [MAKE "LISTAFUNC :CONHL] [IF LAST :PARAM =
[] [MAKE "LISTAFUNC OPERA FIRST :PARAM [] :POD1 :LIMITE
MAKE "LISTAFUNC JUNTA FIRST :LISTAFUNC BF :LISTAFUNC] [IF
LISTP FIRST FIRST :PARAM [MAKE "PARAM1 FIRST FIRST :PARAM]
[MAKE "PARAM1 FIRST :PARAM] IF LISTP LAST LAST :PARAM [MA-
KE "PARAM2 LAST LAST :PARAM] [MAKE "PARAM2 LAST :PARAM]
MAKE "PARAM LIST :PARAM1 :PARAM2 MAKE "LISTAFUNC OPERA
:PARAM [] :POD1 :LIMITE MAKE "LISTAFUNC JUNTA FIRST
:LISTAFUNC BF :LISTAFUNC]]
IF :LISTAFUNC = [] [GO "FORA6]
LABEL "LOOP3
IF MEMBERP FPUT FIRST :LISTAFUNC [] FIRST :DUV1 [IF NOT
MEMBERP FIRST :LISTAFUNC :LISTARUIM [MAKE "LISTARUIM LPUT
FIRST :LISTAFUNC :LISTARUIM MAKE "LISTAFUNC LPUT FIRST
:LISTAFUNC :LISTAFUNC GO "FORA2]]
PR [- - - - - - - - - - - - - - - - - - - -]
PR JUNTA [VOU INVESTIGAR] FIRST :LISTAFUNC
MAKE "R0 REFUTACAO1 (FPUT FPUT FIRST :LISTAFUNC [] [])
(FPUT INVERSO [[]] FPUT FIRST :LISTAFUNC [] []) :Z [[]]
[[]] [[]] [[]]
MAKE "PROV0 FIRST LAST FIRST :R0
MAKE "NPROV0 INVERSO :PROV0 FPUT FIRST :LISTAFUNC []
IF :NPROV0 = [] [GO "FORA2]
MAKE "TESTE :NPROV0
MAKE "UMAL :UMA
MAKE "PROV1 []
LABEL "LOOP1
MAKE "R1 REFUTACAO1 (FPUT FPUT FIRST :LISTAFUNC [] [])
(FPUT :TESTE []) (FPUT FIRST :UMAL []) [[]] [[]] [[]] [[]]
IF NOT LAST FIRST :R1 = [] [IF :TIPO = "SENTAO [GO "PARE-
SENTAO]] [IF NOT MEMBERP FIRST FIRST LAST FIRST :R1 :PROV1
[MAKE "PROV1 FPUT FIRST FIRST LAST FIRST :R1 :PROV1]] [] GO
"FORA1]]
MAKE "PRO FIRST FIRST LAST FIRST REFUTACAO1 (FPUT FPUT
FIRST :LISTAFUNC [] []) (FPUT :PROV0 []) (FPUT FIRST :UMAL
[]) [[]] [[]] [[]] [[]]
MAKE "PROV1 FPUT :PRO :PROV1
MAKE "PROV0 ELIM :PRO :PROV0 []
IF :PROV0 = [] [GO "FORA2]
MAKE "TESTE INVERSO :PROV0 FPUT FIRST :LISTAFUNC []
LABEL "FORA1
IF NOT BF :UMAL = [] [MAKE "UMAL BF :UMAL GO "LOOP1]
IF NOT (EQUALP :TIPO "SESE) [GO "FORA2]
MAKE "HIPOT AGRUPA LIST :PROV1 :PROV0
IF OR FIRST :HIPOT = [] INVERSO FIRST :HIPOT [[]] = [] [GO
"FORA2]
PR [EPA! PARECE FAZER SENTIDO!]
OP LIST FPUT FIRST :LISTAFUNC [] :HIPOT
LABEL "PARESENTAO
```

```
IF OR (NOT MEMBERP FPUT FIRST #LISTAFUNC [] #CONHL) NOT
(SUBTRAI FPUT FIRST #LISTAFUNC [] #PROV1 #PODA) = #PODA
[PR [OPA!] OP LIST FPUT FIRST #LISTAFUNC [] #PROV0]
LABEL "FORA2
IF NOT BF #LISTAFUNC = [] [MAKE "LISTAFUNC BF #LISTAFUNC
GO "LOOP3] [MAKE "LISTARUIM []]
LABEL "FORA6
IF NOT #FORM1 = [] [MAKE "PARAM (LIST FIRST #PARAM FIRST
#FORM1) MAKE "FORM1 BF #FORM1 GO "INICIO]
OP "FRAC
END

TO SEPARA #POD
MAKE "CONHEC []
MAKE "A# 1 MAKE "C# 0 MAKE "D# (COUNT #POD)
LABEL "LOOP3
MAKE "C 0
IF ITEM #A# #POD = [] [GO "FORA3]
MAKE "AA# FIRST ITEM #A# #POD
IF OR MEMBERP #AA# #CONHEC MEMBERP EQUIVALE #AA# #CONHEC
[GO "FORA3]
  REPEAT (COUNT #AA#) [IF AND NOT PRIMITIVEP FIRST #AA# NOT
"DIVIDIDO = FIRST #AA# [MAKE "C #C + 1 IF #C > 2 [GO
"FORA3]] [] MAKE "AA# BF #AA#]
MAKE "AA# FIRST ITEM #A# #POD
MAKE "C# 1
REPEAT (COUNT #N3) [IF MEMBERP FIRST #N3 #AA# [MAKE "CO-
NHEC LPUT #AA# #CONHEC GO "FORA3] [MAKE "N3 LPUT FIRST #N3
BF #N3]]
LABEL "FORA3
IF NOT #A# = #D# [MAKE "A# #A# + 1 GO "LOOP3]
END

TO TRAZ #F #T
IF OR (OR #T = [] #T = [[]]) #T = [[] []] [OP []]
IF #F = FIRST FIRST #T [OP FIRST LAST #T] [OP TRAZ #F LIST
BF FIRST #T BF LAST #T]
END

TO ABS #PRN #PDESEQ #ZI #SC1 #SC0 #COMBINIC #NI
MAKE "N3 [] REPEAT (COUNT #PDESEQ) [MAKE "L FIRST DECESTU-
DA LAST FIRST #PDESEQ IF NOT MEMBERP #L #N3 [MAKE "N3 LPUT
#L #N3] [] MAKE "PDESEQ LPUT FIRST #PDESEQ BF #PDESEQ]
IF NOT #PRN = [] [MAKE "N3 LPUT FIRST DECESTUDA FIRST #PRN
#N3]
SEPARA (JUNTA [[[#GRAU]] [[#PASSO]] [[#TOTAL]]
[[#RECURSAO]] [[#TRANSF.GRAU]] [[#TRANSF.PASSO]]
[[#TRANSF.TOTAL]] [[#TRANSF.REC]]] FIRST #POD1)
IF NOT #PRN = [] [MAKE "CONHECPRN [] MAKE "DECPRN FIRST
```

```
DECODIGO FIRST :PRN REPEAT (COUNT :CONHEC) [IF MEMBERP
:DECPRN FIRST :CONHEC [MAKE "CONHECPRN LPUT FIRST :CONHEC
:CONHECPRN] []] MAKE "CONHEC LPUT FIRST :CONHEC BF
:CONHEC]] [MAKE "CONHECPRN :REFUTA MAKE "DECPRN :REFUTA
MAKE "CONHEC ELIM :REFUTA :CONHEC []]
LABEL "LOOP
IF :N3 = [] [OP :NI]
IF FIRST :N3 = :DECPRN [GO "FORA]
PR JUNTA LPUT :DECPRN [VOU INVESTIGAR] LPUT FIRST :N3 [E]
IF NOT :PRN = [] [IF (FIRST :PRN) = (DECODIGO FIRST :N3)
[MAKE "ELIM0N3 :SOMAELIM0] [MAKE "ELIM0N3 []] MAKE "M DECO-
DIGO FIRST :N3 REPEAT (COUNT :SOMAELIM0) [IF MEMBERP (LIST
(ITEM :M FIRST :SOMAELIM0) :M) :PDESEQ [MAKE "ELIM0N3 LPUT
FIRST :SOMAELIM0 :ELIM0N3] [] MAKE "SOMAELIM0 LPUT FIRST
:SOMAELIM0 BF :SOMAELIM0]]]
MAKE "CONHECN3 []
REPEAT (COUNT :CONHEC) [IF MEMBERP FIRST :N3 FIRST :CONHEC
[MAKE "CONHECN3 LPUT FIRST :CONHEC :CONHECN3 MAKE "CONHEC
LPUT FIRST :CONHEC BF :CONHEC] [MAKE "CONHEC LPUT FIRST
:CONHEC BF :CONHEC]]
IF AND NOT :ZI = "ZERO NOT :PRN = [] [MAKE "CONHEC ELIM
FIRST DECESTUDA FIRST :PRN :CONHEC []]
MAKE "TESTEPRN :CONHECPRN MAKE "CONH []
LABEL "LOOP3
MAKE "FRACHIP 1
IF OR :CONHECN3 = [] :TESTEPRN = [] [GO "FORA]
MAKE "ELIM0 :SOMAELIM0
IF :CONH = [] [MAKE "AZ ABSTRAI LPUT LPUT FIRST :TESTEPRN
[] [] :SC1 :ELIM0N3 :CONH 4 "SESE 0 ELIM FIRST :TESTEPRN
:CONHECN3 [] []] [MAKE "AZ ABSTRAI [] :SC1 :ELIM0N3 :CONH
4 "SESE 0 [] []]
MAKE "INAC "F
IF :AZ = "FRAC [MAKE "TESTEPRN BF :TESTEPRN MAKE "CONH []
GO "LOOP3]
IF EQUIVALEMEMBRO FIRST :AZ FIRST :HIPOTESE = "T [MAKE
"LI2 FPUT EQUIVALE FIRST FIRST :AZ []] [MAKE "LI2 FIRST
:AZ]
IF AND NOT MEMBERP FIRST :AZ FIRST :HIPOTESE :LI2 = FIRST
:AZ [PR [VEJA O QUE VERIFIQUEI:] MAKE "HIPOTESE LIST FPUT
FIRST :AZ FIRST :HIPOTESE FPUT LAST LAST :AZ LAST
:HIPOTESE MOSTTTRAPODA FPUT (LIST FIRST :AZ FIRST LAST :AZ)
[] MAKE "LL :TC1] [MAKE "LI TRAZ :LI2 :HIPOTESE MAKE "HI-
POTESE TIRA :LI2 :HIPOTESE MAKE "HIPOTESE LIST (FPUT :LI2
FIRST :HIPOTESE) (FPUT (INSERE LAST LAST :AZ :LI) LAST
:HIPOTESE) PR :HIPOTESE IF :LI = FIRST LAST :HIPOTESE [MA-
KE "LL []] [MAKE "LL :TC1]]
MAKE "PROVN3 :ELIM0N3 MAKE "PROVSOMA :SOMAELIM0 MAKE "PHIP
:HIPOTESE
REPEAT (COUNT :SOMAELIM0) [MAKE "CCOMB LPUT LPUT [0] FIRST
```

```
:SOMAELIM0 [] MAKE "I CHECAREFUTA (LIST FPUT :LI2 [] FPUT
FIRST LAST :AZ []) FPUT LIST :LI2 FIRST LAST :AZ [] IF AND
:I = "T NOT :ELIM0N3 = [] [MAKE "ELIM0N3 ELIM FIRST
:SOMAELIM0 :ELIM0N3 []] [] IF :I = "T [MAKE "SOMAELIM0 BF
:SOMAELIM0] [MAKE "SOMAELIM0 LPUT FIRST :SOMAELIM0 BF
:SOMAELIM0]]
MAKE "LL JUNTA :LL :TC1L
REPEAT (COUNT :LL) [MAKE "CCOMB LPUT LPUT [1] FIRST :LL []
MAKE "I CHECAREFUTA (LIST FPUT :LI2 [] FPUT FIRST LAST :AZ
[]) FPUT LIST :LI2 FIRST LAST :AZ [] IF :I = "T [IF :QUAL2
= [] [MAKE "INAC "T]] [] MAKE "LL LPUT FIRST :LL BF :LL]
MAKE "HIPOTESE :PHIP
IF :INAC = "T [MAKE "FRACHIP :FRACHIP + 1 MAKE "ELIM0N3
:PROVN3 MAKE "SOMAELIM0 :PROVSOMA IF :FRACHIP > 3 [PR [VO-
CE ACHA QUE DEVO PROSSEGUIR NESSA INVESTIGACAO ? RESPONDA
S / N] MAKE "L  READLIST IF OR :L = [N] :L = [NAO] [MAKE
"ELIM0N3]]]
IF NOT :ELIM0N3 = [] [IF BF :LISTAFUNC = [] [MAKE
"CONHECN3 BF :CONHECN3] [] MAKE "CONH BF :LISTAFUNC GO
"LOOP3]
LABEL "FORA
IF NOT :SOMAELIM0 = [] [MAKE "N3 BF :N3 GO "LOOP] [OP []]
END

TO INICIAL
MAKE "LL [GRAU PASSO TOTAL RECURSAO TRANSF.GRAU
TRANSF.PASSO TRANSF.TOTAL TRANSF.REC]
LABEL "LOOP3
PR [LEMBRETES IMPORTANTES:] PR [] PR [POSSO ESTUDAR NESSE
PROGRAMA UM MAXIMO DE OITO VARIAVEIS E UM MINIMO DE 2] PR
[] PR [AS VARIAVEIS PODEM RECEBER VALORES CONSTANTES OU
SER DEFINIDAS COMO OPERACOES DE OUTRAS VARIAVEIS. PODEM
SER EMPARELHADAS A PARAMETROS OU NAO.]
PR [EXISTEM DUAS LIMITACOES NESSE PROTOTIPO] PR JUNTA [-OS
NOMES DADOS `AS VARIAVEIS SO' PODEM SER :] :LL PR [-TODA
VEZ QUE UM VALOR ESTUDADO NAO FOR O RECEBIDO COMO ARGUMEN-
TO DA CHAMADA EXTERNA, DEVE SER PRECEDIDO DO COMANDO RUN.]
PR [] PR [SE ESTAS CONDICOES ESTAO SATISFEITAS TECLE SIM.
CASO CONTRARIO TECLE NAO E RETORNE DEPOIS DE MODIFICA -
LAS.] MAKE "L READLIST
IF OR :L = [NAO] :L = [N] [OP "PARE]
IF AND NOT :L = [S] NOT :L = [SIM] [PR [NAO ENTENDI.] GO
"LOOP3]
LABEL "LOOP1
PR [DIGA QUANTAS VARIAVEIS DEVEM SER ESTUDADAS EM SEU PRO-
GRAMA]
MAKE "TERMINO READLIST
IF NOT :TERMINO = [] [IF NOT NUMBERP FIRST :TERMINO [GO
"LOOP1] [IF 8 < FIRST :TERMINO [PR [MAXIMO 8] GO "LOOP1]]]
```

```
[GO "LOOP1]
MAKE "TERMINO FIRST :TERMINO MAKE "VAZIO []
DEFINE "USUAR [[]] [[]]
DEFINE "CONHECIDO [[]] [[]]
MAKE "CONT 1 MAKE "VISTO []
PR JUNTA [DE UM VALOR JA' EXPERIMENTADO PARA CADA VARIAVEL
DO PROGRAMA . CASO EU PECA UM VALOR PARA VARIAVEL INEXIS-
TENTE EM] FPUT :MODELO [TECLE -]
LABEL "LOOP
MAKE "L RECOLHE :CONT
IF :L = "ERRO [GO "LOOP]
IF :L = [-] [MAKE "TERMINO :TERMINO + 1 MAKE "VAZIO LPUT
:CONT :VAZIO MAKE "VISTO LPUT LIST :CONT [[]] :VISTO IF
:TERMINO > 0 [PR [HOUVE ENGANO NO NUMERO DE VARIAVEIS DO
PROGRAMA] GO "LOOP1]] [MAKE "VISTO LPUT LIST :CONT :L
:VISTO]
IF NOT :CONT = :TERMINO [MAKE "CONT :CONT + 1 MAKE "LL BF
:LL GO "LOOP]
IF :CONT < 0 [REPEAT (0 - :CONT) [MAKE "VISTO LPUT LIST
(:CONT + 1) [[]] :VISTO MAKE "CONT :CONT + 1 MAKE "VAZIO
LPUT :CONT :VAZIO]]
DEFINE "UM FPUT [] FPUT :VISTO FPUT [[]] FPUT [[]] FPUT
[[]] FPUT [[]] FPUT [[[[] 1] [[] 2]]] []
DEFINE "TERMINO LIST [] FPUT :TERMINO []
DEFINE "VAZIO LIST [] :VAZIO
OP "OK
END

TO PARA
PR [DIGA] MAKE "L READLIST RUN :L
IF :L = [] [STOP] [PARA]
END

TO RELACIONA :PODAL1 :SUB1 :SUB0 :TCOMB1
MAKE "LIM 0.75
MAKE "N0 COUNT :SUB0
IF (COUNT :SUB1) / (:N0 + (COUNT :SUB1)) > 0.75 [STOP]
MAKE "C. 1
LABEL "LOOP3
IF OR :PODAL1 = [] :PODAL1 = [[]] [GO "FORA] [IF FIRST
:PODAL1 = [] [GO "FORA2]]
IF LAST ITEM :C. :VISTO = [[]] [GO "FORA2]
MAKE "F2 FIRST :PODAL1
IF NOT (COUNT :F2) = 1 [GO "FORA2]
CT PR [VOU VERIFICAR] PR FIRST :F1 PR [E] PR FIRST :F2
MAKE "RELAC1 REFUTACAO1 (FPUT :F2 []) (FPUT BF ITEM :C.
FIRST FIRST :VISTONAO []) :SUB1 [[]] [[]] [[]] [[]]
MAKE "B FIRST LAST FIRST :RELAC1 MAKE "CC BF ITEM :C.
FIRST FIRST :VISTONAO REPEAT (COUNT :B) [MAKE "CC ELIM
```

```
FIRST :B :CC [] MAKE "B BF :B] IF :CC = [] [GO "FORA2]
MAKE "RELAC2 REFUTACAO1 (FPUT :F2 []) (FPUT :CC []) :SUB0
[[]] [[]] [[]] [[]]
MAKE "NC ((COUNT (FIRST FIRST LAST :RELAC2)) / :N0) IF :NC
= 0 [GO "FORA2]
MAKE "PROV1: []
MAKE "RESREF :RESREF..
LABEL "LOOP0
MAKE "PROV FIRST LAST FIRST :RELAC2
LABEL "LOOP
MAKE "PROV1: FPUT LPUT FIRST :PROV FIRST :RESREF :PROV1:
IF NOT BF :PROV = [] [MAKE "PROV BF :PROV GO "LOOP]
IF NOT BF :RESREF = [] [MAKEEE "RESREF BF :RESREF GO "LOOP0]
REPEAT (COUNT :TCOMB1) [MAKE "RELAC3 REFUTACAO1 FPUT FPUT
FIRST :F1 :F2 [] FPUT :PROV1: [] FPUT FIRST :TCOMB1 []
[[]] [[]] [[]] [[]] IF NOT FIRST :RELAC3 = [[] []] [MAKE
"PROV1: ELIM FIRST FIRST LAST FIRST :RELAC3 :PROV1: [] IF
:PROV1: = [] [GO "FORA2]] [] MAKE "TCOMB1 LPUT FIRST
:TCOMB1 BF :TCOMB1]
PR []
PR [VEJA O QUE OBSERVEI]
PR []
PR JUNTA [SE] JUNTA FIRST :F1 JUNTA [=] INVAR :RESREF.. PR
JUNTA [ENTAO] JUNTA FIRST :F2 JUNTA [NAO E' =] FIRST LAST
FIRST :RELAC2
MAKE "DUVIDA FPUT LIST (JUNTA :F1 :F2) :PROV1: :DUVIDA
LABEL "FORA2
IF NOT BF :PODAL1 = [] [MAKE "PODAL1 BF :PODAL1 MAKE "C.
:C. + 1 GO "LOOP3] [MAKE "LIM :LIM - 0.5 IF NOT :LIM < 0
[MAKE "PODAL1 :PAR. MAKE "C. 1]]
END

TO ALGORITREF :F1 :RESREF :SUBAM1 :SUBAM0 :TCOMB1 :TCOMB0
MAKE "RESREF.. :RESREF
MAKE "PAR. [[:GRAU]] [[:PASSO]] [[:TOTAL]] [[:RECURSAO]]
[[:TRANSF.GRAU]] [[:TRANSF.PASSO]] [[:TRANSF.TOTAL]]
[[TRANSF.REC]]] MAKE "A 1 MAKE "B 1
RELACIONA :PAR. :SUBAM1 :SUBAM0 :TCOMB1
END

TO AGRUPA :R.
MAKE "INVUM INVAR FIRST :R.
IF NOT LAST :R. = [] [MAKE "INVT INVAR JUNTA FIRST :R.
LAST :R.] [MAKE "INVT [[[]]]]
MAKE "L []
MAKE "RES []
IF :INVUM = [] [MAKE "INVUM [[]] GO "FORA2]
MAKE "INV LAST :INVUM
IF NUMBERP FIRST :INV [MAKE "INV :INVUM GO "FORA1]
```

```
IF :INVT = [] [MAKE "INVT [[[]]]]
REPEAT (COUNT :INV) [IF NOT MEMBERP FIRST :INV LAST :INVT
[MAKE "RES LPUT LPUT FIRST :INV [] :RES] [] MAKE "INV BF
:INV]
MAKE "INV BL :INVUM
LABEL "FORA1
REPEAT (COUNT :INV) [IF AND NUMBERP FIRST FIRST :INV NOT
MEMBERP LPUT FIRST :INV [] LAST :R. [MAKE "L LPUT LPUT
FIRST :INV [] :L] [] MAKE "INV BF :INV]
LABEL "FORA2
MAKE "ZERO LAST :R. MAKE "PROV0 [] MAKE "INV LAST :INVUM
MAKE "RES2 []
REPEAT (COUNT :INVT) [IF NUMBERP ITEM 1 FIRST :INVT [IF
NOT MEMBERP LPUT FIRST :INVT [] FIRST :R. [MAKE "PROV0
LPUT LPUT FIRST :INVT [] :PROV0 MAKE "RES2 LPUT LPUT FIRST
:INVT [] :RES2]] [] MAKE "INVT LPUT FIRST :INVT BF :INVT]
REPEAT (COUNT :RES) [MAKE "RES2 FPUT (FPUT FPUT WORD "NAO
FIRST FIRST :RES [] []) :RES2 MAKE "PROV0 JUNTA NINVAR
FIRST :RES2 :PROV0 MAKE "RES LPUT FIRST :RES BF :RES]
IF AND :PROV0 = [] NOT :L = [] [OP LIST INVERSO :L [[]]
[]] [OP LIST :PROV0 :RES2]
END

TO SUBTRAI :F :R :ONDE
IF OR :ONDE = [] :ONDE = [[]] [OP :ONDE]
IF NOT :F = FIRST FIRST :ONDE [OP FPUT FIRST :ONDE SUBTRAI
:F :R BF :ONDE]
MAKE "PROV LAST FIRST :ONDE
LABEL "LOOP
MAKE "PROV EXCLUI FIRST :R :PROV
IF NOT BF :R = [] [MAKE "R BF :R GO "LOOP]
IF NOT :PROV = [] [OP FPUT LIST FIRST FIRST :ONDE :PROV BF
:ONDE] [OP BF :ONDE]
END

TO INTERPRETA :R
IF :R = "IND [OP :R]
IF :R < 0 [MAKE "Q :R * -1] [MAKE "Q :R]
IF (:Q - INT :Q) > 0.9999998 [IF :R < 0 [MAKE "R INT :R -
1] [MAKE "R INT :R + 1]]
IF (:Q - INT :Q) < 0.000001 [MAKE "R INT :R]
IF OR (OR :R = 1 :R = 0) :R = -1 [OP :R]
IF :R / 360 = (INT :R / 360) [IF :R > 0 [OP
"INT.MULT360.POS] [OP "INT.MULT360.NEG]]
MAKE "Q ((:R / 360) - (INT (:R / 360))) IF :Q < 0 [MAKE "Q
:Q * -1]
IF OR :Q < 0.00000001 :Q > 0.99999998 [IF :R > 0 [OP
"INT.MULT360.POS] [OP "INT.MULT360.NEG]]
IF 360 / :R = (INT 360 / :R) [IF (INT (:R / 1)) = :R [IF
```

```
:R > 0 [OP "INT.DIVISOR360.POJ [OP "INT.DIVISOR360.NEGJ]
[IF :R > 0 [OP "VG.DIVISOR360.POJ [OP
"VG.DIVISOR360.NEGJ]]
MAKE "Q ((360 / :R) - (INT (360 / :R))) IF :Q < 0 [MAKE "Q
:Q * -1]
IF OR :Q < 0.00001 :Q > 0.99998 [IF (INT (:R / 1)) = :R
[IF :R > 0 [OP "INT.DIVISOR360.POJ [OP
INT.DIVISOR360.NEGJ] [IF :R > 0 [OP "VG.DIVISOR360.POJ [OP
"VG.DIVISOR360.NEGJ]]]
IF NOT (INT (:R / 1)) = :R [IF :R > 0 [OP "VG.POJ [OP
"VG.NEGJ] [IF :R > 0 [OP "INT.POJ [OP "INT.NEGJ]
END

TO ALTERA :A :P :QUAL :CONT :VALOR
IF :QUAL = [] [OP []]
IF :P = :CONT [IF AND EQUALP "POE :VALOR NOT MEMBERP :A
FIRST :QUAL [OP FPUT (LPUT :A FIRST :QUAL) ALTERA :A :P BF
:QUAL :CONT + 1 :VALOR] [IF AND EQUALP "TIRA :VALOR MEM-
BERP :A FIRST :QUAL [OP FPUT ( ELIM :A FIRST :QUAL [] )
ALTERA :A :P BF :QUAL :CONT + 1 :VALOR]]]
OP FPUT FIRST :QUAL ALTERA :A :P BF :QUAL :CONT + 1 :VALOR
END

TO MUDAPODA
MAKE "I CHECAREFUTA :POD1
IF AND BF :CCOMB = [] NOT :FS. = [] [MAKE "POD1 REFUTUM
:PODA [[]]]
END

TO OPERA :PAR0 :PAR2 :CONH :LIMITE
MAKE "RECIPR []
MAKE "PAR :PAR0 MAKE "A FIRST :PAR
IF (COUNT :PAR0) = :LIMITE [OP []]
LABEL "LOOP1
MAKE "P JUNTA (LPUT "* LPUT ") FPUT "( :A) (LPUT ") FPUT
"( FIRST BF :PAR)
IF OR MEMBERP LPUT :P [] :POD1 EQUIVALEMEMBRO LPUT :P []
:POD1 = "T [MAKE "PAR2 FPUT :P :PAR2] [MAKE "PAR2 LPUT :P
:PAR2]
MAKE "P JUNTA [DIVIDIDO] (JUNTA LPUT ") FPUT "( :A LPUT ")
FPUT "( FIRST BF :PAR)
IF OR MEMBERP LPUT :P [] :CONH EQUIVALEMEMBRO LPUT :P []
:CONH = "T [MAKE "RECIPR FPUT :P :RECIPR] [MAKE "RECIPR
LPUT :P :RECIPR]
MAKE "P JUNTA (LPUT "+ LPUT ") FPUT "( :A) LPUT ") FPUT "(
FIRST BF :PAR
IF OR EQUIVALEMEMBRO LPUT :P [] :CONH = "T NOT MEMBERP
LPUT :P [] :CONH [MAKE "PAR2 FPUT :P :PAR2] [MAKE "PAR2
LPUT :P :PAR2]
```

```
MAKE "P JUNTA [DIVIDIDO] (JUNTA FPUT "( LPUT ") FIRST BF
#PAR LPUT ") FPUT "( #A)
IF AND EQUIVALEMEMBRO LPUT #P [] #CONH = "F NOT MEMBERP
LPUT #P [] #CONH [MAKE "PAR2 LPUT #P #PAR2] [MAKE "PAR2
FPUT #P #PAR2]
MAKE "P JUNTA (LPUT "- FPUT "( LPUT ") FIRST BF #PAR) LPUT
") FPUT "( #A
IF AND EQUIVALEMEMBRO LPUT #P [] #CONH = "F NOT MEMBERP
LPUT #P [] #CONH [MAKE "PAR2 LPUT #P #PAR2] [MAKE "PAR2
FPUT #P #PAR2]
MAKE "P JUNTA (LPUT "- FPUT "( LPUT ") #A) LPUT ") FPUT "(
FIRST BF #PAR
IF AND EQUIVALEMEMBRO LPUT #P [] #CONH = "F NOT MEMBERP
LPUT #P [] #CONH [MAKE "RECIPR LPUT #P #RECIPR] [MAKE "RE-
CIPR FPUT #P #RECIPR]
IF NOT BF BF #PAR = [] [MAKE "PAR BF #PAR GO "LOOP1] [OP
FPUT #PAR2 #RECIPR]
END

TO COPIA #AII #PII #VISTOII
IF EQUALP [] #VISTOII [OP []]
IF NOT EQUALP FIRST FIRST #VISTOII #PII [MAKE "COP FPUT
ALEATORIO BF FIRST #VISTOII #L#. COPIA #AII #PII BF
#VISTOII] [MAKE "COP FPUT (FPUT #AII []) COPIA #AII #PII
BF #VISTOII]
OP #COP
END

TO EXEC #MODELO #LISTA
MAKE "LL [GRAU PASSO TOTAL RECURSAO TRANSF.GRAU
TRANSF.PASSO TRANSF.TOTAL TRANSF.REC]
MAKE "C [] MAKE "T [] MAKE "CONT 0 MAKE "R (COUNT (FIRST
TEXT #MODELO))
LABEL "LOOP
REPEAT #R [IF NOT FIRST #LISTA = [[]] [MAKE FIRST #LL RUN
FIRST #LISTA MAKE "C JUNTA #C (JUNTA [(] JUNTA FPUT RUN
FIRST #LISTA [] [)])] [MAKE "CONT #CONT + 1] MAKE "LL BF
#LL MAKE "LISTA BF #LISTA]
IF #CONT > 0 [MAKE "R #CONT MAKE "CONT 0 GO "LOOP]
REPEAT (COUNT #LL) [IF NOT FIRST #LISTA = [[]] [MAKE FIRST
#LL FIRST #LISTA MAKE "T JUNTA #T (FPUT FIRST #LISTA [])]
[] MAKE "LL BF #LL MAKE "LISTA BF #LISTA]
PR #C PR #T
IF OR OR #MODELO = "POLIS #MODELO = "ARVORE BL #MODELO =
"POLIS [MAKE "MOD (WORD #MODELO "L) OP RUN FPUT #MOD PONTO
(FIRST TEXT #MODELO)] [RUN FPUT #MODELO PONTO (FIRST TEXT
#MODELO) OP POS]
END
```

```
TO TODOS :NT :NP
IF :NT = [] [OP "T] [IF :NP = [] [OP "F]]
IF NOT MEMBERP FIRST :NT :NP [OP "F]
OP TODOS BF :NT :NP
END

TO OLHA :TAB :COMB
MAKE "TAB1 FIRST :TAB
MAKE "TAB2 FIRST BF :TAB
MAKE "COP1 :TAB1
MAKE "COP2 :TAB2
MAKE "ZERO []
MAKE "SEPARA []
MAKE "PIOR 85
MAKE "MELHOR -1
MAKE "PPP 0
MAKE "FIM 0
LABEL "LOOP1
MAKE "PPP :PPP + 1
MAKE "SUB1 FIRST :COP1
IF :SUB1 = [] [GO "FORA2]
IF BF :SUB1 = [] [MAKE "FIM :FIM + 1 MAKE "T (FIRST FIRST
FIRST :COP2) * 100 / (LAST FIRST FIRST :COP2) GO "FORA]
MAKE "SUB2 FIRST :COP2
IF :SUB1 = [] [GO "FORA]
LABEL "LOOP2
IF LAST FIRST :SUB2 = 0 [[GO "JA]
MAKE "PROP (FIRST FIRST :SUB2) * 100 / LAST FIRST :SUB2
IF NOT (:PROP > 5) [MAKE "ZERO LPUT (LIST FIRST :SUB1
:PPP) :ZERO MAKE "PIOR :PROP] [IF :PROP < :PIOR [MAKE
"PIOR :PROP MAKE "SEPARA LPUT (LIST FIRST :SUB1 :PPP) []]
[IF :PROP = :PIOR [MAKE "SEPARA LPUT (LIST FIRST :SUB1
:PPP) :SEPARA]]]
IF :PROP > :MELHOR [MAKE "MELHOR :PROP]
LABEL "JA
MAKE "SUB1 BF :SUB1 MAKE "SUB2 BF :SUB2
IF NOT = :SUB1 [] [GO "LOOP2]
LABEL "FORA
MAKE "COP1 BF :COP1 MAKE "COP2 BF :COP2
IF NOT = :COP1 [] [GO "LOOP1]
LABEL "FORA2
IF :MELHOR = :PIOR [IF :MELHOR < 75 [OP "FIM] [OP LIST
(LIST "SIM :MELHOR) (LIST :TAB :COMB)]]
MAKE "PAR COUNT FIRST :TAB
IF OR (AND :FIM = :PAR :T < 75) (AND :FIM = (:PAR - 1)
:MELHOR < 75) [OP "FIM]
IF NOT EQUALP :ZERO [] [MAKE "TAB RETIRA :COMB :TAB1 :TAB2
:ZERO OP LIST (LIST "ZERO :ZERO) :TAB]
IF NOT EQUALP :SEPARA [] [MAKE "TAB RETIRA :COMB :TAB1
```

```
:TAB2 :SEPARA OP LIST (LIST "SEPARA :SEPARA) :TAB]
OP LIST (LIST "SIM :MELHOR) (LIST :TAB :COMB)
END

TO RETIRA :COMB :TAB1 :TAB2 :QUAL
MAKE "SOMAELIM0 []
MAKE "TAB LIST :TAB1 :TAB2
REPEAT (COUNT :COMB) [IF LAST FIRST :COMB = [1] [MAKE
"COMB BF :COMB] [MAKE "COMB LPUT FIRST :COMB BF :COMB]]
LABEL "LOOP1
MAKE "COP :COMB
LABEL "LOOP2
MAKE "CC FIRST :COP
REPEAT LAST FIRST :QUAL [MAKE "LP FIRST :CC MAKE "CC BF
:CC]
IF EQUALP :LP FIRST FIRST :QUAL [MAKE "TAB TABELA -1 FIRST
:COP FIRST :TAB LAST :TAB [] []] MAKE "COMB ELIM FIRST :COP
:COMB [] MAKE "SOMAELIM0 LPUT BL FIRST :COP :SOMAELIM0]
MAKE "COP BF :COP
IF NOT EQUALP :COP [] [GO "LOOP2]
MAKE "QUAL BF :QUAL
IF AND NOT EQUALP :QUAL [] NOT :COMB = [] [GO "LOOP1]
OP LIST (LIST FIRST :TAB FIRST BF :TAB) :COMB
END

TO ALEATORIO :ELEM :N
MAKE "N. :N MAKE "ELEML []
IF (COUNT :ELEM) < :N [OP :ELEM]
MAKE "R.R (COUNT :ELEM)
LABEL "LOOP
REPEAT :N. [MAKE "SORTE ITEM ((RANDOM :R.R) + 1) :ELEM IF
NOT MEMBERP :SORTE :ELEML [MAKE "ELEML LPUT :SORTE
:ELEML]]
IF COUNT :ELEML < :N [MAKE "N. :N - COUNT :ELEML GO "LOOP]
OP :ELEML
END

TO SEPARADO
IF OR :PODA = [] :PODA = [[]] [MAKE "SELEC [[] [] [] [] []
[] [] [] []] STOP]
MAKE "L [:PASSO :TOTAL :RECURSAO :TRANSF.GRAU
:TRANSF.PASSO :TRANSF.TOTAL :TRANSF.REC]
MAKE "N3 [:GRAU]
MAKE "SELEC [[]]
REPEAT (COUNT :L) [MAKE "N3 LPUT FIRST :L :N3 SEPARADO2
MAKE "SELEC LPUT :CONHEC :SELEC MAKE "L BF :L]
END

TO SEPARADO2
```

```
MAKE "CONHEC []
MAKE "A# 1 MAKE "D# (COUNT #PODA)
LABEL "LOOP3
IF (COUNT ITEM #A# #PODA) = 0 [STOP]
MAKE "C# []
MAKE "AA# FIRST FIRST ITEM #A# #PODA
IF NOT MEMBERP LAST #N3 #AA# [GO "FORA3]
REPEAT (COUNT #AA#) [IF AND NOT PRIMITIVEP FIRST #AA# NOT
FIRST #AA# = "DIVIDIDO [MAKE "C# LPUT FIRST #AA# #C#] []
MAKE "AA# BF #AA#]
MAKE "AA# FIRST FIRST ITEM #A# #PODA
REPEAT (COUNT #C#) [IF NOT MEMBERP FIRST #C# #N3 [GO
"FORA3] [] MAKE "C# BF #C#]
MAKE "CONHEC LPUT ITEM #A# #PODA #CONHEC
LABEL "FORA3
IF NOT #A# = #D# [MAKE "A# #A# + 1 GO "LOOP3]
END

TO CRESCE2
ERASE "CRESCE
RECYCLE
MAKE "RESULTADO NIVELDOIS #MODELO #CONHECIDOUM
DEFINE (WORD #MODELO #NOME) FPUT [] FPUT FIRST FIRST BF
#RESULTADO LAST LAST #RESULTADO
IF NOT MEMBERP (WORD #MODELO #NOME) LAST TEXT "USUAR [DE-
FINE "USUAR LIST [] LPUT (WORD #MODELO #NOME) LAST TEXT
"USUAR] [ERASEFILE (WORD " WORD #MODELO #NOME)]
PACKAGE (WORD " #NOME) (FPUT WORD #MODELO #NOME [])
SAVE (WORD " WORD #MODELO #NOME) (FPUT #NOME [])
IF NOT MEMBERP #MODELO LAST TEXT "PROGRAMA [DEFINE "PRO-
GRAMA LIST [] LPUT #MODELO LAST TEXT "PROGRAMA DEFINE "UM
FPUT [] FPUT #VISTO FPUT [[]] FPUT [[]] FPUT [[]] [] PAC-
KAGE "PROGRAMA [PROGRAMA] ERASEFILE "PROGRAMA SAVE "PRO-
GRAMA [PROGRAMA] DEFINE "CONHECIDO [[] [[]] [(ERASEFILE
(WORD " #MODELO)) IF (FIRST FIRST #RESULTADO) > 0 [DEFINE
"CONHECIDO LIST [] FIRST BF TEXT (WORD " WORD #MODELO
#NOME)]]
MAKE "L (JUNTA FPUT #MODELO [] JUNTA [USUAR] JUNTA [UM]
JUNTA [CONHECIDO] JUNTA [VAZIO] [TERMINO])
IF OR OR #MODELO = "POLIS #MODELO = "ARVORE BL #MODELO =
"POLIS [MAKE "L LPUT WORD #MODELO "L #L]
PACKAGE (WORD " #MODELO) #L
SAVE (WORD " #MODELO) (FPUT #MODELO [])
RECYCLE
LOAD "INICIO
RECYCLE
END

TO INSERE #N #N1
```

```
IF :N = [] [OP :N1]
IF NOT MEMBERP FIRST :N :N1 [MAKE "N1 LPUT FIRST :N :N1]
OP INSERE BF :N :N1
END

TO REFUTACAO1 :CFUNC :CRES :COMB1 :REFUT :RESUL :COMBREF
:UNICA
LOCAL "PROV2
IF :CFUNC = [] [OP LIST LIST BL :REFUT BL :RESUL BL
:COMBREF]
IF FIRST :CRES = [[]] [GO "OUT]
MAKE "CCOMB1 :COMB1
MAKE "SUBCOMBL [] MAKE "COMBREFL []
LABEL "LOOP1
IF OR (OR :MODELO = "POLIS :MODELO = "ARVORE) BL :MODELO =
"POLIS [IF FIRST BF BF BF BF FIRST :CCOMB1 = [] [GO
"FORA5]]
MAKE "FUNC FIRST :CFUNC MAKE "RES FIRST :CRES MAKE "PROV2
[] MAKE "C111 FIRST :CCOMB1
MAKE "LP [GRAU PASSO TOTAL RECURSAO TRANSF.GRAU
TRANSF.PASSO TRANSF.TOTAL TRANSF.REC]
REPEAT :TERMINO [MAKE FIRST :LP RUN FIRST :C111 MAKE "LP
BF :LP MAKE "C111 BF :C111]
LABEL "LOOP5
IF :PROV2 = [] [MAKE "PROV2 LPUT LPUT INTERPRETA RUN FIRST
:FUNC [] :PROV2] [MAKE "PROV2 LPUT LPUT RUN FIRST :FUNC []
:PROV2]
IF NOT BF :FUNC = [] [MAKE "FUNC BF :FUNC GO "LOOP5]
IF NOT MEMBERP :PROV2 :RES [MAKE "PROV2 LPUT RUN FIRST
:FUNC :PROV2 IF NOT MEMBERP :PROV2 :RES [MAKE "PROV2 LPUT
RUN FIRST FIRST :CFUNC [] IF NOT MEMBERP :PROV2 :RES [MAKE
"SUBCOMBL FPUT FIRST :CCOMB1 :SUBCOMBL GO "FORA5]]]
IF :UNICA = "UNICA [OP "FRAC]
IF NOT EQUALP FIRST :CFUNC FIRST :REFUT [MAKE "REFUT FPUT
FIRST :CFUNC :REFUT MAKE "RESUL FPUT FPUT :PROV2 [] :RESUL
MAKE "COMBREFL FPUT FIRST :CCOMB1 :COMBREFL GO "FORA5]
IF AND NOT MEMBERP :PROV2 FIRST :RESUL NOT EQUALP :PROV2
FIRST FIRST :RESUL [MAKE "RESUL FPUT (FPUT :PROV2 FIRST
:RESUL) BF :RESUL]
MAKE "COMBREFL FPUT FIRST :CCOMB1 :COMBREFL
LABEL "FORA5
IF NOT BF :CCOMB1 = [] [MAKE "CCOMB1 BF :CCOMB1 GO "LOOP1]
MAKE "COMBREF FPUT LIST :COMBREFL :SUBCOMBL :COMBREF
LABEL "OUT
OP REFUTACAO1 BF :CFUNC BF :CRES :COMB1 :REFUT :RESUL
:COMBREF :UNICA
END

TO CLASSIFICA :MODELO :COMBSUJ :COMBSIST :NAOC :VISTOC :AC
```

```
:PC :PONTOS :PROVIS
MAKE "VERMS [] MAKE "FS. [] MAKE "RS. [] MAKE "CS0. [] MA-
KE "CS1. [] MAKE "COMBESP [] MAKE "HIPOTESE [[[]] [[]]]
MAKE "RES INVESTIGA JUNTA :COMBSUJ :COMBSIST :VISTOC :NAOC
:AC :PC
MAKE "REALN FIRST :RES MAKE "RAP FIRST BF :RES MAKE "RAM-
BOS FIRST BF BF :RES MAKE "RDIF LAST BL :RES
IF AND :REALN = [] :T. = "F [OP "NADA]
IF OR :RAP = "T :RAMBOS = "T [PR [VOCE TEVE IDEIA ( S )
DIFERENTE ( S ) . VAMOS ESTUDALA -LA ( S ) .] MAKE "RES
ESTUDA :MODELO :REALN "T (JUNTA :COMBSUJ :COMBSIST)
:VISTOC :NAOC :AC :PC :PONTOS :PROVIS MAKE "PROVIS LAST
:RES MAKE "PONTOS LAST BL :RES OP LPUT :PROVIS LIST (LIST
  "RECOMECA :PONTOS) FIRST :RES]
PR [VOCE PENSOU EM COISA ( S ) DIFERENTE ( S ) . VAMOS
ANALISA -LA ( S ) .] MAKE "RES ESTUDA :MODELO :REALN "F
:COMBSUJ :VISTOC :NAOC :AC :PC :PONTOS :PROVIS MAKE "PRO-
VIS LAST :RES MAKE "PONTOS LAST BL :RES OP LPUT :PROVIS
LIST (LIST "CONTINUA :PONTOS) FIRST :RES
END

TO IMPRINIC :VISTO :PROV
TEXTSCREEN CT
PR [] PR [] PR [] PR [] PR []
MAKE "PAR [[- - - - - - - - - :GRAU - - - - - - - - -] [- - -
- - - - - - :PASSO - - - - - - - - -] [- - - - - - - - - :TOTAL
- - - - - - - - -] [- - - - - - - - - :RECURSAO - - - - - -]
[- - - - - - - - - :TRANSF.GRAU - - - - -] [- - - - - - - - -
:TRANSF.PASS - - - - -] [- - - - - - - - - :TRANSF.TOTAL - -
- -] [- - - - - - - - - :TRANSF.REC - - - - -]]
PR [PARA EXPERIMENTAR O PROGRAMA, VOCE PRECISA ASSOCIAR A
CADA VARIAVEL UMA CONSTANTE OU UMA TRANSFORMACAO DE OUTRA
VARIAVEL. TECLE ALGO QUE JA' FOI ESTUDADO OU UMA NOVA
IDEIA QUE VOCE QUEIRA EXPERIMENTAR.]
MAKE "CVISTO :VISTO MAKE "CONT 0 MAKE "COMB [] MAKE "CPROV
:PROV MAKE "M []
LABEL "LOOP1
IF NOT :COMB = [] [IF NOT LAST :COMB = [[]] [PR :M]]
MAKE "UM BF FIRST :CVISTO MAKE "CONT :CONT + 1 MAKE "UM-
PROV FIRST :CPROV
IF MEMBERP :CONT :VAZIO [MAKE "COMB LPUT [[]] :COMB GO
"VZ]
PR [] PR [] PR FIRST :PAR PR [ESTUDADOS - - - - - - EM ES-
TUDO]
LABEL "LOOP2
IF AND :UMPROV = [] :UM = [] [GO "FORA2]
IF NOT :UM = [] [PR FIRST :UM] [PR []]
IF NOT :UMPROV = [] [IF (LAST CURSOR) > 19 [SETCURSOR LIST
(FIRST CURSOR) 22] [SETCURSOR LIST ((FIRST CURSOR) - 1)
```

```
22] PR FIRST :UMPROV]
IF NOT :UM = [] [MAKE "UM BF :UM]
IF NOT :UMPROV = [] [MAKE "UMPROV BF :UMPROV]
GO "LOOP2
LABEL "FORA2
MAKE "COMB LPUT (RECOLHE :CONT) :COMB
IF NOT LAST :COMB = [[]] [MAKE "M LPUT LAST :COMB :M]
LABEL "VZ
MAKE "CVISTO BF :CVISTO
IF OR :MODELO = "POLIS :MODELO = "ARVORE [IF :CONT = 4 [IF
EQUALP LAST :COMB [0] [OP JUNTA :COMB [[]] [] [] [[]]]] [IF
:TERMINO < 8 [IF :CONT = :TERMINO [REPEAT (8 - :TERMINO)
[MAKE "COMB LPUT [[]] :COMB]] OP :COMB]]]
IF NOT :CVISTO = [] [MAKE "PAR BF :PAR MAKE "CPROV BF
:CPROV GO "LOOP1]
OP :COMB
END

TO MOSTRAPODA :PODAL
MAKE "N.... COUNT :PODAL
IF :N.... = 1 [GO "LOOP]
TEXTSCREEN CT
PR [AS SEGUINTES CONDICOES DEVEM SER SATISFEITAS PARA QUE
O DESENHO DE CERTO :]
LABEL "LOOP
PR [- - - - - - - - - - - - - - - - - - - -] PR []
IF FIRST CURSOR > 18 [PR [( TECLE RETURN )] MAKE "L REA-
DLIST CT]
IF LAST FIRST :PODAL = [[]] [PR FIRST FIRST :PODAL GO "FO-
RA]
IF (COUNT FIRST FIRST :PODAL) > 1 [GO "OUT]
PR FIRST FIRST FIRST :PODAL MAKE "L1 INVERSO LAST FIRST
:PODAL [[]] MAKE "L2 []
REPEAT (COUNT :L1) [MAKE "L2 LPUT FIRST FIRST FIRST :L1
:L2 MAKE "L1 BF :L1]
MAKE "L2 INVAR :L2
REPEAT (COUNT :L2) [IF NOT NUMBERP FIRST FIRST :L2 [GO
"OUT0]] [] MAKE "L2 LPUT FIRST :L2 BF :L2]
PR JUNTA [NAO E'] INVAR LAST FIRST :PODAL GO "FORA
LABEL "OUT0
PR JUNTA [E'] :L2 GO "FORA
LABEL "OUT
MAKE "PROV [] MAKE "POD LAST FIRST :PODAL
LABEL "LOOP1
IF :POD = [] [GO "FORA1]
IF NOT MEMBERP FPUT FIRST FIRST :POD [] :PROV [MAKE "PROV
FPUT FPUT FIRST FIRST :POD [] :PROV]
MAKE "POD BF :POD GO "LOOP1
LABEL "FORA1
```

```
IF LAST FIRST FIRST :PODAL = [] [MAKE "CONEC []] [MAKE
"CONEC [SE]]
PR JUNTA :CONEC JUNTA FIRST FIRST FIRST :PODAL JUNTA [E']
INVAR :PROV
MAKE "PROV [] MAKE "POD LAST FIRST :PODAL MAKE "N 2
LABEL "LOOP2
IF :POD = [] [GO "FORA2]
IF AND LAST FIRST FIRST :PODAL = [] :N > (COUNT FIRST
:POD) [GO "FORA]
IF NOT MEMBERP FPUT ITEM :N FIRST :POD [] :PROV [MAKE
"PROV FPUT FPUT ITEM :N FIRST :POD [] :PROV]
MAKE "POD BF :POD GO "LOOP2
LABEL "FORA2
IF (COUNT FIRST FIRST :PODAL) = :N [MAKE "CONEC [ENTAO]
MAKE "R.. :PROV] [MAKE "CONEC [E] MAKE "R.. :PROV]
PR JUNTA :CONEC JUNTA ITEM :N FIRST FIRST :PODAL JUNTA
[NAO E'] :R..
IF (COUNT FIRST FIRST :PODAL) > :N [MAKE "N :N + 1 MAKE
"POD LAST FIRST :PODAL MAKE "PROV [] GO "LOOP2]
LABEL "FORA
IF AND NOT BF :PODAL = [] NOT BFFF :PODAL = [[]] [MAKE "PO-
DAL BF :PODAL GO "LOOP]
PR [- - - - - - - - - - - - - - - - - - - -]
IF :TERMINO = 0 [STOP]
IF :N... = 1 [PR [VOCE CONCORDA?]] [PR [VOCE CONCORDA COM
TODAS ESSAS CONDICOES ?]]
PR [RESPONDA S / N] MAKE "L READLIST
IF :N... = 1 [STOP]
IF OR :L = [N] :L = [NAO] [PR [E' MESMO ? ENTAO ME ENSINE.
MOSTRE O QUE ESTA' ERRADO USANDO UM ARGUMENTO AINDA NAO
ESTUDADO. SERA' QUE VOCE CONSEGUE ?] PR [] PR [] PR [TECLE
RETURN] MAKE "L READLIST]
END

TO CHECAREFUTA :QUAL :QUAL1
IF OR :QUAL = [] :QUAL = [[]] [[]] [OP "F]
MAKE "VERM REFUTACAO1 FIRST :QUAL LAST :QUAL LPUT BL FIRST
:CCOMB [] [[]] [[]] [[]] [[]]
IF FIRST FIRST :VERM = [] [OP "F]
MAKE "F. FIRST FIRST :VERM MAKE "R. LAST FIRST :VERM
LABEL "CONF
IF :F. = [] [OP "T]
MAKE "TIRA FIRST :R. MAKE "HIP FPUT INVAR FIRST :R. []
IF AND (COUNT FIRST :F.) = 1 NOT NUMBERP FIRST FIRST FIRST
FIRST :R. [MAKE "LI TRAZ FIRST :F. :QUAL MAKE "PROVC2 LAST
INVAR INVERSO :LI [[]] IF NUMBERP FIRST :PROVC2 [GO "OUT9]
[MAKE "TIRA [] REPEAT (COUNT :PROVC2) [MAKE "TIRA LPUT
LPUT LPUT WORD "NAO FIRST :PROVC2 [] [] :TIRA MAKE "PROVC2
LPUT FIRST :PROVC2 BF :PROVC2] MAKE "HIP [] REPEAT (COUNT
```

```
:TIRA) [MAKE "L: NINVAR FIRST :TIRA IF MEMBERP FIRST FIRST
:R. :L: [MAKE "HIP LPUT FIRST :TIRA :HIP MAKE "TIRA JUNTA
BF :TIRA :L:] [MAKE "TIRA BF :TIRA]]]
IF AND NOT NUMBERP FIRST FIRST FIRST FIRST :R. (COUNT
FIRST :F.) > 1 [MAKE "TIRA TRAZ FIRST :F. :QUAL MAKE "HIP
LPUT LPUT LAST INVAR :TIRA [] []]
IF LAST FIRST :CCOMB = [1] [MAKE "QUAL2 SUBTRAI FIRST :F.
:TIRA :QUAL1 MAKE "QUAL1 :QUAL2] [MAKE "QUAL2 :QUAL1]
IF AND NOT FIRST :QUAL = FPUT FIRST FIRST :HIPOTESE [] NOT
:QUAL = LIST :FS. :RS. [MAKE "LI TRAZ FIRST :F. :HIPOTESE
MAKE "HIPOTESE TIRA FIRST :F. :HIPOTESE MAKE "HIPOTESE
LIST FPUT FIRST :F. FIRST :HIPOTESE FPUT ( INSERE :HIP :LI
) LAST :HIPOTESE]
LABEL "OUT9
MAKE "R. FPUT :TIRA BF :R.
IF MEMBERP FIRST :F. :FS. [GO "EMPAR]
MAKE "FS. FPUT FIRST :F. :FS. MAKE "RS. FPUT FIRST :R.
:RS.
IF LAST FIRST :CCOMB = [0] [MAKE "CS0. FPUT FPUT BL FIRST
:CCOMB [] :CS0. MAKE "CS1. FPUT [] :CS1.] [MAKE "CS1. FPUT
FPUT BL FIRST :CCOMB [] :CS1. MAKE "CS0. FPUT [] :CS0.]
GO "FORA
LABEL "EMPAR
MAKE "CFS :FS. MAKE "FRENTER [] MAKE "FRENTEC0 [] MAKE
"FRENTEC1 []
LABEL "LOOP
IF NOT FIRST :F. = FIRST :CFS [MAKE "FRENTER LPUT FIRST
:RS. :FRENTER MAKE "FRENTEC1 LPUT FIRST :CS1. :FRENTEC1
MAKE "FRENTEC0 LPUT FIRST :CS0. :FRENTEC0 MAKE "RS. BF
:RS. MAKE "CS1. BF :CS1. MAKE "CS0. BF :CS0. MAKE "CFS BF
:CFS GO "LOOP]
MAKE "RS. JUNTA (LPUT ( INSERE FIRST :R. FIRST :RS.)
:FRENTER ) BF :RS.
IF OR MEMBERP BL FIRST :CCOMB FIRST :CS1. MEMBERP BL FIRST
:CCOMB FIRST :CS0. [MAKE "CS1. JUNTA :FRENTEC1 :CS1. MAKE
"CS0. JUNTA :FRENTEC0 :CS0. GO "FORA]
IF LAST FIRST :CCOMB = [1] [MAKE "CS1. JUNTA LPUT (FPUT BL
FIRST :CCOMB FIRST :CS1.) :FRENTEC1 BF :CS1. MAKE "CS0.
JUNTA :FRENTEC0 :CS0.] [MAKE "CS0. JUNTA :FRENTEC0 (FPUT
(FPUT BL FIRST :CCOMB FIRST :CS0.) BF :CS0.) MAKE "CS1.
JUNTA :FRENTEC1 :CS1.]
LABEL "FORA
MAKE "F. BF :F. MAKE "R. BF :R. GO "CONF
END

TO PARTE :MODELO :COMBI
CLEARSCREEN PR []
MAKE "POSIN POS
MAKE "EXEC EXEC :MODELO :COMBI
```

```
IF :EXEC = "FRACASSO [GO "ERRO]
LABEL "LOOP
PR [ESSE DESENHO DEU CERTO , E E' INTERESSANTE ? RESPONDA
S / N .]
MAKE "RES READLIST
IF OR :RES = [S] :RES = [SIM] [MAKE "COMBI LPUT [1] :COMBI
OP :COMBI]
IF :RES = [?] [PR [DIGA] RUN READLIST GO "LOOP] [IF AND
NOT :RES = [N] NOT :RES = [NAO] [PR [NAO ENTENDI] GO
"LOOP]]
LABEL "ERRO
PR [VEJA O QUE FIZ .] PR :C PR :T
RECYCLE
OP LPUT [0] :COMBI
END

TO REDUZ :RCOMB :RTAB :PESO
IF OR :RCOMB = [] :RTAB = [[]] [OP "SUCESSO]
MAKE "L1 FIRST FIRST :RTAB
IF FIRST :RCOMB = [[]] [GO "FORA2]
IF :L1 = [] [OP "SUCESSO]
MAKE "L2 FIRST LAST :RTAB
LABEL "LOOP
IF NOT :RN = [] [IF EQUALP FIRST FIRST :RN FIRST :RCOMB
[GO "FORA2]]
IF NOT EQUALP FIRST :RCOMB FIRST :L1 [GO "FORA]
IF (LAST FIRST :L2) > 1 [IF (FIRST FIRST :L2) / (LAST
FIRST :L2) < :PESO [OP "FRACASSO]]
LABEL "FORA
MAKE "L1 BF :L1
IF NOT :L1 = [] [MAKE "L2 BF :L2 GO "LOOP]
LABEL "FORA2
OP REDUZ BF :RCOMB (LIST BF FIRST :RTAB BF LAST :RTAB)
:PESO
END

TO GERA :AG :PG :VISTOI
RECYCLE
SEPARADO
IF :TERMINO > 6 [MAKE "L:. 3] [IF "TERMINO = 0 [MAKE "L:.
5] [MAKE "L:. 4]]
IF :REALN = [] [MAKE "L:. 5]
MAKE "J 0
LABEL "JA
MAKE "COP COPIA :AG :PG :VISTOI
RECYCLE
IF :REALN = [] [GO "FORA]
MAKE "L :REALN
LABEL "LOOP
```

```
MAKE "L1 FIRST :L
IF AND NOT (FIRST :L1) = :AG NOT (LAST :L1) = :PG [MAKE
"COP ALTERA FIRST :L1 LAST :L1 :COP 1 "POE]
IF NOT BF :L = [] [MAKE "L BF :L GO "LOOP]
LABEL "FORA
IF OR (AND NOT :MODELO = "POLIS NOT :MODELO = "ARVORE) NOT
MEMBERP [0] FIRST BF BF BF :COP [MAKE "GERADO COMB FIRST
:COP [] [] :COP :SELEC 0] [IF NOT EQUALP [[0]] FIRST BF BF
BF :COP [MAKE "COP JUNTA (FPUT FIRST :COP FPUT FIRST BF
:COP FPUT FIRST BF BF :COP []) JUNTA (LPUT ( ELIM [0]
FIRST BF BF BF :COP []) [] ) BF BF BF BF :COP MAKE "GERADO
COMB FIRST :COP [] [] :COP :SELEC 0] [MAKE "GERADO COMB
FIRST :COP [] [] (LPUT [[]] LPUT [[]] LPUT [[]] LPUT [[]]
LPUT [[0]] BL BL BL BL BL :COP) :SELEC 0]]
IF AND :GERADO = [] :J = 0 [MAKE "J 1 MAKE "SELEC [[]] MA-
KE "PER "T GO "JA]
OP :GERADO
END

TO AVALIA :TAB :COMBINIC :A.A :P.P :VISTO :NAO :OBS
:PROVIS
MAKE "PASSO OLHA :TAB :COMBINIC
RECYCLE
IF :PASSO = "FIM [OP LIST LIST :VISTO [] :PROVIS]
MAKE "COMB. LAST LAST :PASSO
MAKE "TAB FIRST LAST :PASSO
MAKE "RESPI FIRST :PASSO
IF AND NOT EQUALP FIRST :RESPI "ZERO NOT EQUALP FIRST
:RESPI "SEPARA [IF :A.A = [] [STOP] [GO "ERA]]
PR [VOU TENTAR DESCOBRIR ALGO SOBRE O QUE NAO ESTA' DANDO
CERTO.]
IF NOT :P.P = [] [MAKE "P LPUT :P.P []] [MAKE "P []]
MAKE "NI ABS :P LAST :RESPI FIRST :RESPI :SC1 [] :COMBINIC
[]
GO "FORA4
LABEL "ERA
MAKE "VISTO ALTERA :A.A :P.P :VISTO 1 "POE
MAKE "POD1 REFUTUM :PODA [[]]
OP LPUT :OBS LPUT :PROVIS LPUT :TAB (LIST (LIST :VISTO
:NAO) :COMB.)
LABEL "FORA4
IF :A.A = [] [IMPRIMA :TAB :REFUTA 1 "CONT] [IMPRIMA :TAB
:A.A :P.P "CONT]
OP AVALIA :TAB :COMB. :A.A :P.P :VISTO :NAO :OBS :PROVIS
END

TO IMPRIMA :TABELA :A :P :VEZ
CT
MAKE "TAB11 FIRST :TABELA
```

```
MAKE "TABI2 LAST :TABELA
MAKE "PAR [[- - - - - - - - - :GRAU - - - - - - - -] [- - -
- - - - - :PASSO - - - - - - - -] [- - - - - - - - :TOTAL
- - - - - - - -] [- - - - - - - - :RECURSAO - - - - - -]
[- - - - - - - - :TRANSF.GRAU - - - - -] [- - - - - - - -
:TRANSF.PASSO - - - -] [- - - - - - - - :TRANSF.TOTAL - -
- -] [- - - - - - - - :TRANSF.REC - - - - -]]
TEXTSCREEN CT
PR [FOI EXPERIMENTADO :] PR JUNTA :A JUNTA [COMO] DECODIGO
:P
PR [PERCENTUAIS DE EXITO]
LABEL "LOOP1
IF FIRST CURSOR > 23 [PR [( TECLE RETURN )] MAKE "L REA-
DLIST CT]
MAKE "UM FIRST :TABI1 MAKE "DOIS FIRST :TABI2
IF :UM = [[]] [GO "FORA.]
PR FIRST :PAR
LABEL "LOOP2
IF :UM = [] [GO "FORA.]
IF LAST FIRST :DOIS = 0 [PR JUNTA FIRST :UM [- - - - - -
ELIMINADO]] [PR JUNTA FIRST :UM JUNTA [=] FPUT ((FIRST
FIRST :DOIS) * 100 / (LAST FIRST :DOIS)) FPUT "% []]
MAKE "UM BF :UM
MAKE "DOIS BF :DOIS GO "LOOP2
LABEL "FORA.
MAKE "TABI1 BF :TABI1 MAKE "TABI2 BF :TABI2 MAKE "PAR BF
:PAR
IF NOT :TABI1 = [] [GO "LOOP1]
END

TO TABELA :S :RES :TAB1 :TAB2 :SSUB1 :SSUB2
MAKE "SUB1 FIRST :TAB1
MAKE "SUB2 FIRST :TAB2
IF OR :RES = [] FIRST :RES = [] [GO "NL]
MAKE "FRENTE []
IF NOT MEMBERP FIRST :RES :SUB1 [MAKE "SUB1 FPUT FIRST
:RES :SUB1 MAKE "SUB2 FPUT (LIST FIRST LAST :RES "1) :SUB2
GO "NL]
LABEL "LOOP1
IF EQUALP FIRST :RES FIRST :SUB1 [MAKE "T ((LAST FIRST
:SUB2) + :S) MAKE "M (FIRST FIRST :SUB2) + (:S * FIRST
LAST :RES) MAKE "FRENTE LPUT (LIST :M :T) :FRENTE] [MAKE
"FRENTE LPUT FIRST :SUB2 :FRENTE]
MAKE "SUB2 BF :SUB2 MAKE "SUB1 BF :SUB1
IF NOT EQUALP :SUB1 [] [GO "LOOP1] [MAKE "SUB2 :FRENTE MA-
KE "SUB1 FIRST :TAB1]
LABEL "NL
IF BF :TAB1 = [] [MAKE "VOLTA LIST (LPUT :SUB1 :SSUB1)
(LPUT :SUB2 :SSUB2) OP :VOLTA]
```

```
IF BF :RES = [] [MAKE "RES [[]]]
MAKE "VOLTA TABELA :S BF :RES BF :TAB1 BF :TAB2 LPUT :SUB1
:SSUB1 LPUT :SUB2 :SSUB2 OP :VOLTA
END

TO INCLUIDO :N11 :C2
MAKE "PP 0 MAKE "C1 []
LABEL "LOOP
MAKE "PP :PP + 1 MAKE "C1 LPUT (LIST FIRST :C2 :PP) :C1
IF NOT BF BF :C2 = [] [MAKE "C2 BF :C2 GO "LOOP]
OP INCLUIDO2 :N11 :C1 []
END

TO INVAR :R0
LOCAL "L
LOCAL "L2
IF OR :R0 = [] OR :R0 = [[]] :R0 = [[[]]] [OP []]
IF LISTP FIRST :R0 [REPEAT (COUNT :R0) [MAKE "R0 LPUT
FIRST FIRST FIRST :R0 BF :R0]]
MAKE "L []
MAKE "R :R0 REPEAT (COUNT :R) [IF NOT NUMBERP FIRST :R
[MAKE "R LPUT FIRST :R BF :R] [MAKE "R BF :R]]
IF :R = [] [GO "FORA]
IF AND AND NOT MEMBERP "INT.DIVISOR360.PO :R NOT MEMBERP
"INT.DIVISOR360.NEG :R AND NOT MEMBERP "VG.DIVISOR360.PO
:R NOT MEMBERP "VG.DIVISOR360.NEG :R [MAKE "L LPUT [NAO-
DIV.] :L]
IF AND AND NOT MEMBERP "INT.MULT360.PO :R NOT MEMBERP
"INT.MULT360.NEG :R AND NOT MEMBERP "VG.MULT360.NEG :R NOT
MEMBERP "VG.MULT360.PO :R [MAKE "L LPUT [NAOMULT.] :L]
IF AND AND NOT MEMBERP "VG.PO :R NOT MEMBERP "VG.NEG :R
AND NOT MEMBERP "INT.PO :R NOT MEMBERP "INT.NEG :R [IF :L
= [] [MAKE "L LPUT [DIV.OU.MULT] :L] [IF (COUNT :L) = 2
[MAKE "L [[NAODIV. NAOMULT.]]] [IF :L = [[NAODIV.]] [MAKE
"L [[MULT.]]] [MAKE "L [[DIV.]]]]]]
IF AND AND AND NOT MEMBERP "INT.DIVISOR360.PO :R NOT MEM-
BERP "VG.DIVISOR360.PO :R AND NOT MEMBERP "INT.MULT360.PO
:R NOT MEMBERP "VG.MULT360.PO :R AND NOT MEMBERP "VG.PO :R
NOT MEMBERP "INT.PO :R [MAKE "L LPUT [NEG.] :L] [IF AND
AND AND NOT MEMBERP "INT.DIVISOR360.NEG :R NOT MEMBERP
"VG.DIVISOR360.NEG :R AND NOT MEMBERP "INT.MULT360.NEG :R
NOT MEMBERP "VG.MULT360.NEG :R AND NOT MEMBERP "VG.NEG :R
NOT MEMBERP "INT.NEG :R [MAKE "L LPUT [POS.] :L]]
IF AND AND AND NOT MEMBERP "INT.DIVISOR360.PO :R NOT MEM-
BERP "INT.DIVISOR360.NEG :R AND NOT MEMBERP
"INT.MULT360.PO :R NOT MEMBERP "INT.MULT360.NEG :R AND NOT
MEMBERP "INT.PO :R NOT MEMBERP "INT.NEG :R [MAKE "L LPUT
[NAOINTEIRO] :L] [IF AND AND AND NOT MEMBERP
"VG.DIVISOR360.PO :R NOT MEMBERP "VG.DIVISOR360.NEG :R AND
```

```
NOT MEMBERP "VG.PO :R NOT MEMBERP "VG.NEG :R AND NOT MEM-
BERP "VG.MULT360.PO :R NOT MEMBERP "VG.MULT360.NEG :R [MA-
KE "L LPUT [INTEIRO] :L]]
IF NOT :L = [] [REPEAT ((COUNT :L) - 1) [MAKE "L FPUT JUN-
TA FIRST :L FIRST BF :L BF BF :L]]
LABEL "FORA
MAKE "L2 []
MAKE "R :R0
REPEAT (COUNT :R) [IF NUMBERP FIRST :R [MAKE "L2 LPUT
FIRST :R :L2] []] MAKE "R BF :R]
IF :L2 = [] [OP :L]
IF MEMBERP 0 :L2 [MAKE "L FPUT [0] :L]
IF MEMBERP 1 :L2 [MAKE "L FPUT [1] :L]
IF MEMBERP -1 :L2 [MAKE "L FPUT [-1] :L]
OP :L
END

TO DECODIGO :PP
IF NOT NUMBERP :PP [GO "FORA]
IF :PP = 1 [OP [:GRAU]]
IF :PP = 2 [OP [:PASSO]]
IF :PP = 3 [OP [:TOTAL]]
IF :PP = 4 [OP [:RECURSAO]]
IF :PP = 5 [OP [:TRANSF.GRAU]]
IF :PP = 6 [OP [:TRANSF.PASSO]]
IF :PP = 7 [OP [:TRANSF.TOTAL]]
IF :PP = 8 [OP [:TRANSF.REC]]
LABEL "FORA
IF :PP = "GRAU [OP 1]
IF :PP = "PASSO [OP 2]
IF :PP = "TOTAL [OP 3]
IF :PP = "RECURSAO [OP 4]
IF :PP = "TRANSSSF.GRAU [OP 5]
IF :PP = "TRANSF.PASSO [OP 6]
IF :PP = "TRANSF.TOTAL [OP 7]
IF :PP = "TRANSF.REC [OP 8]
END

TO JUNTA :K :L
IF :L = [] [OP :K]
OP JUNTA (LPUT FIRST :L :K) BF :L
END

TO COMB :PC :TCOMBC :CB :LISTA :SEPARADO :CONT
IF OR (OR :PC = [] (COUNT :TCOMBC) > 15) :CONT > 2 [OP
:TCOMBC]
MAKE "CB LPUT FIRST :PC :CB
IF FIRST :SEPARADO = [] [GO "OUT]
MAKE "C. 1
```

```
MAKE "CB1 :CB
REPEAT (8 - (COUNT :CB)) [MAKE "CB1 LPUT [[]] :CB1] MAKE
"A. 1
REPEAT (COUNT FIRST :SEPARADO) [MAKE "L REFUTACAO1 FPUT
FIRST ITEM :A. FIRST :SEPARADO [] FPUT LAST ITEM :A. FIRST
:SEPARADO [] FPUT :CB1 [] [[]] [[]] [[]] "UNICA IF :L =
"FRAC [GO "OUT2] [MAKE "A. :A. + 1]]
LABEL "OUT
IF NOT EQUALP BF :LISTA [] [MAKE "TCOMBC COMB FIRST BF
:LISTA :TCOMBC :CB BF :LISTA BF :SEPARADO 0] [MAKE "TCOMBC
LPUT :CB :TCOMBC]
MAKE "TCOMBC COMB BF :PC :TCOMBC BL :CB :LISTA :SEPARADO
:CONT + 1 OP :TCOMBC
LABEL "OUT2
MAKE "TCOMBC COMB BF :PC :TCOMBC BL :CB :LISTA :SEPARADO
:CONT OP :TCOMBC
END

TO INVESTIGA :TCOMBI :VISTOI :NAOI :AI :PI
MAKE "REALN [] MAKE "VELHO [] MAKE "RAP "F MAKE "RAMBOS "F
MAKE "RDIF "F MAKE "T. "F
MAKE "POD1 REFUTUM :PODA [[]]
LABEL "LOOP2
MAKE "RN []
MAKE "COMB1 FIRST :TCOMBI
MAKE "CVISTO :VISTOI
IF LAST :COMB1 = [0] [GO "FORA1]
LABEL "LOOP3
MAKE "V FIRST :CVISTO
MAKE "C FIRST :COMB1
IF NOT MEMBERP :C :V [MAKE "RN LPUT (LIST :C FIRST :V)
:RN]
IF AND NOT BF :CVISTO = [] NOT BF :COMB1 = [] [MAKE "CVIS-
TO BF :CVISTO MAKE "COMB1 BF :COMB1 GO "LOOP3]
IF NOT :RN = [] [IF MEMBERP (LIST :AI :PI) :RN [IF NOT
EQUALP (LPUT (LIST :AI :PI) []) :RN [MAKE "RAMBOS "T] [MA-
KE "RAP "T]] [MAKE "RDIF "T] MAKE "REALN INSERE :RN
:REALN]
LABEL "FORA1
MAKE "CCOMB LPUT FIRST :TCOMBI []
IF NOT :FS. = [] [MAKE "I CHECAREFUTA LIST :FS. :RS. [[]]
IF CHECAREFUTA :DUV1 :DUVIDA = "T [MAKE "T. "T MAKE "DUVI-
DA :QUAL2 MAKE "DUV1 REFUTUM :DUVIDA [[]]]
IF CHECAREFUTA :POD1 :PODA = "T [MAKE "T. "T MAKE "PODA
:QUAL2 MAKE "POD1 REFUTUM :PODA [[]]]
IF NOT BF :TCOMBI = [] [MAKE "TCOMBI BF :TCOMBI GO "LOOP2]
[OP FPUT :REALN FPUT :RAP FPUT :RAMBOS FPUT :RDIF FPUT
:VELHO []]
END
```

```
TO CONVERSA :MODELO :A :P
MAKE "DUV1 REFUTUM :DUVIDA [[]]
MAKE "PODU [[] []]
MAKE "TOT 0 MAKE "MAIS 0 MAKE "ACAO 0
MAKE "COMBSIST []
MAKE "TCOMB [] MAKE "TCOMBS [] MAKE "PROVIS [[] [] [] []
[] [] [] [] [[]] MAKE "COMB5 [] MAKE "TC0L [] MAKE "TC1L []
LABEL "INICIO
IF NOT :PODA = [[]] [PR [VOCE QUER VER A QUE CONCLUSOES EU
CHEGUEI ? RESPONDA S / N] MAKE "L READLIST IF OR :L = [S]
:L = [SIM] [MOSTRAPODA :PODA] [IF AND NOT :L = [N] NOT :L
= [NAO] [PR [NAO ENTENDI] GO "INICIO]]]
PR LPUT "? LPUT :MODELO [VOCE QUER EXPERIMENTAR O PROGRA-
MA] PR [RESPONDA S / N]
MAKE "R READLIST
IF AND (NOT (:R = [S])) (NOT (:R = [SIM])) [IF OR (:R =
[N]) (:R = [NAO]) [GO "FORA] [PR [NAO ENTENDI] GO "INI-
CIO]]
MAKE "COMB IMPRINIC :VISTO :PROVIS
CT MAKE "COMB PARTE :MODELO :COMB
MAKE "PROVIS PROVISORIO :COMB :PROVIS
MAKE "COMB5 LPUT :COMB :COMB5
IF LAST :COMB = [1] [MAKE "TC1L LPUT BL :COMB :TC1L] [MAKE
"TC0L LPUT BL :COMB :TC0L]
MAKE "TCOMB LPUT :COMB :TCOMB GO "INICIO
LABEL "FORA
IF :COMB5 = [] [GO "SISTEMA]
MAKE "DEC CLASSIFICA :MODELO :COMB5 :TCOMBS :NAO :VISTO :A
:P :PONTOS :PROVIS
MAKE "PROVIS [[] [] [] [] [] [] [] []]
MAKE "TC1L [] MAKE "TC0L []
IF EQUALP :DEC "NADA [MAKE "COMB5 [] GO "SISTEMA]
MAKE "DEC BL :DEC
MAKE "VISTO FIRST LAST :DEC
MAKE "NAO LAST LAST :DEC
MAKE "D FIRST FIRST :DEC
MAKE "PONTOS LAST FIRST :DEC
IF :D = "RECOMECA [GO "SISTEMA]
IF :D = "CONTINUA [MAKE "COMB5 [] GO "INICIO]
LABEL "SISTEMA
IF :CONHECZ = [] [STOP]
IF OR :A = [] :A = [[]] [STOP]
LABEL "LOOP1
PR JUNTA LPUT :MODELO [VOCE QUER VER UM DESENHO FEITO POR]
JUNTA [COM] JUNTA :A JUNTA DECODIGO :P [? RESPONDA S / N]
MAKE "L READLIST
IF OR :L = [N] :L = [NAO] [STOP]
MAKE "ACAO 1
```

```
IF :COMBSIST = [] [MAKE "COMBSIST GERA :A :P :VISTO]
LABEL "LOOP2
IF :COMBSIST = [] [PR [NAO CONSEGUI.] STOP]
IF OR MEMBERP FIRST :COMBSIST :TC1 MEMBERP FIRST :COMBSIST
:TC0 [MAKE "COMBSIST BF :COMBSIST GO "LOOP2]
MAKE "COMBS PARTE :MODELO FIRST :COMBSIST MAKE "TOT :TOT +
1
IF LAST :COMBS = [1] [MAKE "MAIS :MAIS + 1 MAKE "TC1 LPUT
BL :COMBS :TC1] [MAKE "TC0 LPUT BL :COMBS :TC0]
IF (:MAIS / :TOT) < 0.7 [STOP]
MAKE "COMBSIST BF :COMBSIST MAKE "TCOMBS LPUT :COMBS
:TCOMBS GO "INICIO
END

TO NIVELDOIS :MODELO :CONHECZ
PR [VOU MOSTRAR O QUE JA APRENDI.] MAKE "TC1 [] MAKE "TC0
[] EXIBE ERASE "EXIBE (RECYCLE)
MAKE "VISTOX :VISTO MAKE "NAO []
MAKE "PONTOS 0
LABEL "LOOP1
IF :CONHECZ = [] [MAKE "CONHECZ :VISTO]
MAKE "PZ FIRST FIRST :CONHECZ
MAKE "VALORES BF FIRST :CONHECZ
IF :VALORES = [] [GO "FORA1]
LABEL "LOOP2
MAKE "AZ FIRST :VALORES
IF OR :AZ = [] :AZ = [[]] [GO "FORA1]
CONVERSA :MODELO :AZ :PZ
MAKE "RESULTADO LIST FPUT :PONTOS [] LIST :VISTO FPUT
:PODA FPUT :INST FPUT :DUVIDA []
LABEL "LOOP3
PR [VOCE QUER EXPERIMENTAR MAIS ESSE PROGRAMA ? RESPONDA S
/ N] MAKE "L READLIST IF OR :L = [N] :L = [NAO] [OP
:RESULTADO]
LABEL "FORA2
IF NOT :VALORES = [] [MAKE "VALORES BF :VALORES]
IF NOT :VALORES =   [] [GO "LOOP2]
LABEL "FORA1
MAKE "CONHECZ BF :CONHECZ
GO "LOOP1
END

TO REFUTUM :CPODA :CDUVIDA
IF AND :CPODA = [[]] :CDUVIDA = [[]] [OP [[] []]]
MAKE "FUNC [] MAKE "RES []
LABEL "LOOP1
IF OR :CPODA = [] :CPODA = [[]] [GO "FORA]
IF LAST FIRST :CPODA = [] [GO "FORA]
IF NOT LAST FIRST :CPODA = [[]] [MAKE "FUNC LPUT FIRST
```

```
FIRST :CPODA :FUNC MAKE "RES LPUT FIRST BF FIRST :CPODA
:RES]
MAKE "CPODA BF :CPODA GO "LOOP1
LABEL "FORA
IF NOT :CDUVIDA = [[]] [MAKE "CPODA :CDUVIDA MAKE "CDUVIDA
[[]] GO "LOOP1]
OP LIST :FUNC :RES
END

TO PROVISORIO :RES :PROV.
MAKE "SUB1 FIRST :PROV.
IF NOT MEMBERP FIRST :RES :SUB1 [MAKE "SUB1 LPUT FIRST
:RES :SUB1]
IF OR (FIRST BF :RES) = [] (BF BF :RES) = [] [OP FPUT
:SUB1 BF :PROV.]
OP FPUT :SUB1 PROVISORIO BF :RES (BF :PROV.)
END

TO GENE
IF OR FIRST :LII = [[POS.]] FIRST :LII = [[NAONEG.]] [MAKE
"TRI FPUT FPUT [[1]] FIRST :TRI BF :TRI STOP]
IF OR FIRST :LII = [[NEG.]] FIRST :LII = [[NAOPOS.]] [MAKE
"TRRRI FPUT FPUT [[-1]] FIRST :TRI BF :TRI STOP]
IF OR FIRST :LII = [[DIV.]] FIRST :LII = [[INTEIRO]] [MAKE
"TRI FPUT JUNTA [[[1]] [[-1]]] FIRST :TRI BF :TRI]
END

TO DIVIDIDO :AAA :B
IF :B = 0 [OP 0]
IF :AAA = 0 [OP 0]
OP :AAA / :B
END

TO LIMPAREF
MAKE "TC1L. [] MAKE "TC0L. [] MAKE "TLII [] MAKE "TFI []
LABEL "INICIO
MAKE "TFS. FIRST :FS. MAKE "TRS. FIRST :RS. MAKE "SC1
FIRST :CS1. MAKE "SC0 FIRST :CS0.
IF AND :SC0 = [] NOT :SC1 = [] [MAKE "CCOMB JUNTA :TC0
:TC0L MAKE "SC0 FIRST FIRST LAST ( REFUTACAO1 FPUT :TFS.
[] FPUT INVERSO FPUT LAST FIRST :TRS. [] :TFS. [] :CCOMB
[[]] [[]] [[]] [[]] )]
LABEL "NOVA
PR [ESSA CONDICAO PARECE TER ALGUM SIGNIFICADO.] MOSTRAPO-
DA FPUT LIST (LPUT [] :TFS.) :TRS. [] PR [VAMOS VER.]
MAKE "LII TRAZ :TFS. :HIPOTESE
IF :LII = [] [MAKE "LII INVAR :TRS. IF :LII = [] [MAKE
"LII [[INTEIRO]] [[NAOINTEIRO]] [[POS.]] [[NEG.]]
[[MULT.]] [[NAOMULT.]] [[DIV.]] [[NAODIV.]]]] [MAKE "LLL
```

```
[] MAKE "LI1 [] REPEAT (COUNT :LII) [IF NUMBERP FIRST
FIRST :LII [MAKE "LI1 LPUT LPUT FIRST :LII [] :LI1] [MAKE
"LI2 FIRST :LII REPEAT (COUNT :LI2) [MAKE "LLL FPUT LPUT
LPUT FIRST :LI2 [] [] :LLL MAKE "LI2 BF :LI2]] MAKE "LII
BF :LII] MAKE "LII JUNTA :LLL :LI1]]
MAKE "TRI []
REPEAT (COUNT :LII) [MAKE "TRI LPUT NINVAR FIRST :LII :TRI
MAKE "LII LPUT FIRST :LII BF :LII]
MAKE "C1L [] REPEAT (COUNT :LII) [MAKE "C1L LPUT [] :C1L]
MAKE "C0L :C1L MAKE "RI :C1L MAKE "CL :C1L MAKE "SC :SC1
MAKE "CONTRL. 1
LABEL "INL
REPEAT (COUNT :SC) [MAKE "RESL REFUTACAO1 FPUT :TFS. []
(FPUT :TRS. []) FPUT FIRST :SC [] [[]] [[]] [[]] [[]] MAKE
"SC LPUT FIRST :SC BF :SC IF NOT FIRST :RESL = [[] []]
[REPEAT (COUNT :TRI) [IF MEMBERP LPUT FIRST FIRST FIRST
LAST FIRST :RESL [] FIRST :TRI [MAKE "C (FIRST FIRST FIRST
LAST :RESL) IF NOT MEMBERP :C FIRST :CL [MAKE "CL FPUT
FPUT :C FIRST :CL BF :CL] [] MAKE "R (FIRST FIRST LAST
FIRST :RESL) IF NOT MEMBERP :R FIRST :RI [MAKE "RI FPUT
FPUT :R FIRST :RI BF :RI] [[]] [] MAKE "CL LPUT FIRST :CL
BF :CL MAKE "RI LPUT FIRST :RI BF :RI MAKE "TRI LPUT FIRST
:TRI BF :TRI]]]
IF :CONTRL. = 1 [MAKE "C1L :CL MAKE "CONTRL. 0 MAKE "CL
:C0L MAKE "SC :SC0 GO "INL] [MAKE "C0L :CL]
MAKE "FI :TFS.
MAKE "AC "F MAKE "CONT 1
REPEAT (COUNT :C1L) [IF AND (ITEM :CONT :C1L) = [] NOT (I-
TEM :CONT :C0L) = [] [IF NOT NUMBERP FIRST FIRST ITEM
:CONT :LII [MAKE "AC "T]] [] MAKE "CONT :CONT + 1]
IF :AC = "F [MAKE "TC0L. LPUT :C0L :TC0L. MAKE "TC1L. LPUT
:C1L :TC1L. MAKE "TFI LPUT :FI :TFI MAKE "TLII LPUT :LII
:TLII]
LABEL "FORA6
MAKE "RII FIRST :RI MAKE "SC1L FIRST :C1L MAKE "SC0L FIRST
:C0L
IF (COUNT :FI) = 2 [IF NUMBERP FIRST FIRST FIRST :RII [MA-
KE "TRI FPUT :RII BF :TRI] [MAKE "Q [] MAKE "L 1 REPEAT
(COUNT FIRST :TRI) [MAKE "Q LPUT (LPUT LAST FIRST :RII
ITEM :L FIRST :TRI) :Q MAKE "L :L + 1] MAKE "TRI FPUT :Q
BF :TRIII]]
LABEL "AQ
IF AND NOT :SC0L = [] :SC1L = [] [PR [VEJA O QUE CONCLUI]
MOSTRAPODA FPUT LIST :FI FIRST :TRI [] IF (COUNT :FI) = 1
[GENE] [] IF AND (COUNT :FI) = 1 MEMBERP :FI FIRST :POD1
[MAKE "L TRAZ :FI :POD1 MAKE "PODA SUBTRAI :FI :L :PODA]
[MAKE "L []] MAKE "DUVIDA SUBTRAI FPUT FIRST :FI [] [[]]
:DUVIDA MAKE "L INSERE :L FIRST :TRI MAKE "PODA FPUT LIST
:FI :L :PODA MAKE "POD1 LIST FPUT :FI FIRST :POD1 FPUT
```

```
LAST FIRST :PODA LAST :POD1 MAKE "DUVIDA SUBTRAI :FI :L
:DUVIDA]
LABEL "FORA3
PARA
IF NOT BF :TRI = [] [MAKE "TRI BF :TRI MAKE "C1L BF :C1L
MAKE "C0L BF :C0L MAKE "RI BF :RI MAKE "LII BF :LII GO
"FORA6]
LABEL "FORA4
MAKE "PODA ELIM LIST :FI [[[0]] [[-1]] [[1]]] :PODA []
MAKE "DUVIDA ELIM LIST :FI [[[0]] [[-1]] [[1]]] :DUVIDA []
MAKE "POD1 REFUTUM :PODA [[]] MAKE "DUV1 REFUTUM :DUVIDA
[[]]
IF AND NOT MEMBERP :TFS. FIRST :POD1 NOT MEMBERP :FI FIRST
:DUV1 [MAKE "DUVIDA FPUT LIST :TFS. [[]] :DUVIDA]
IF NOT BF :FS. = [] [MAKE "FS. BF :FS. MAKE "RS. BF :RS.
MAKE "CS1. BF :CS1. MAKE "CS0. BF :CS0. GO "INICIO]
INACABADA :TC0L. :TC1L. :TFI :TLII
MAKE "V. [] MAKE "VERMS. [] MAKE "TC1 JUNTA :TC1 :TC1L MA-
KE "TC1L [] MAKE "TC0 JUNTA :TC0 :TC0L   MAKE "TC0L []
RECYCLE
END

TO INACABADA :TC0L. :TC1L. :TFI :TLII
LABEL "LOOP1
IF :TFI = [] [STOP]
MAKE "C0L FIRST :TC0L. MAKE "C1L FIRST :TC1L. MAKE "FI
FIRST :TFI MAKE "LII FIRST :TLII
LABEL "LOOP2
IF :LII = [] [GO "FORA3]
IF AND FIRST :C1L = [] NOT FIRST :C0L = [] [GO "FORA]
MAKE "SC0L FIRST :C0L MAKE "PROVSC0L []
REPEAT (COUNT :SC0L) [IF (FIRST REFUTACA01 FIRST :POD1
LAST :POD1 FPUT FIRST :SC0L [] [[]] [[]] [[]] "UNICA) =
[[]] [[]] [MAKE "PROVSC0L FPUT FIRST :SC0L :PROVSC0L] [] MA-
KE "SC0L BF :SC0L]
IF (COUNT :PROVSC0L) > 1 [ALGORITREF FPUT FIRST :FI []
NINVAR FIRST :LII FIRST :C1L :PROVSC0L ( JUNTA :TC1 :TC1L
) :TC0L]
LABEL "FORA
MAKE "LII BF :LII MAKE "C0L BF :C0L MAKE "C1L BF :C1L GO
"LOOP2
LABEL "FORA3
MAKE "TLII BF :TLII MAKE "TC0L. BF :TC0L. MAKE "TC1L. BF
:TC1L. MAKE "TFI BF :TFI GO "LOOP1
END

TO ESTUDA :MODELO :RN :V :COMB :VISTO :NAO :A :P :PONTOS
:PROVIS
LABEL "LOOP1
```

```
MAKE "N3 [] MAKE "SOMAELIM0 [] MAKE "SC1 []
IF :RN = [] [IF NOT :FS. = [] [LIMPAREF OP LPUT :PROVIS
LIST (LIST :VISTO :NAO) :PONTOS MAKE "TC1 JUNTA :TC1
:TC1L] [OP LPUT :PROVIS LIST (LIST :VISTO :NAO) :PONTOS]]
MAKE "RN1 LPUT LPUT (LIST FIRST :RN FIRST :RN) [] []
MAKE "N1 [] MAKE "COMBJA [] MAKE "COMB4 [] MAKE "TAB [[[]
[] [] [] [] [] [] [[] [[] [] [] [] [] [] [] [[]] MAKE "OBS
[]
IF :COMB = [] [GO "OUTROS]
PR JUNTA [VAMOS VER RESULTADOS DE] JUNTA FIRST FIRST :RN
JUNTA [COMO] DECODIGO LAST FIRST :RN
MAKE "CCOMB :COMB
LABEL "LOOP2
MAKE "IN INCLUIDO :RN1 FIRST :CCOMB
IF NOT EQUALP :IN :RN1 [GO "FORA..]
IF LAST FIRST :CCOMB = [0] [IF CHECAREFUTA :POD1 :PODA =
"T [GO "FORA..]]
MAKE "COMBJA LPUT FIRST :CCOMB :COMBJA
MAKE "TAB TABELA 1 FIRST :CCOMB FIRST :TAB LAST :TAB [] []
MAKE "COMB4 LPUT FIRST :CCOMB :COMB4
IF LAST FIRST :CCOMB = [1] [MAKE "SC1 LPUT BL FIRST :CCOMB
:SC1]
LABEL "FORA..
IF NOT BF :CCOMB = [] [MAKE "CCOMB BF :CCOMB GO "LOOP2]
IF :COMB4 = [] [GO "OUTROS]
IMPRIMA :TAB FIRST FIRST :RN LAST FIRST :RN "CONT
MAKE "VISTONAO AVALIA :TAB :COMB4 FIRST FIRST :RN LAST
FIRST :RN :VISTO [] [] :PROVIS
MAKE "TAB LAST BL BL :VISTONAO MAKE "VISTONAO BL BL BL
:VISTONAO MAKE "COMB4 LAST :VISTONAO
MAKE "VISTO FIRST FIRST :VISTONAO
LABEL "OUTROS
MAKE "N3 [] MAKE "SOMAELIM0 [] MAKE "PER "F
PR JUNTA [VOU EXPERIMENTAR OUTROS DESENHOS COM] FIRST
FIRST :RN
MAKE "COMB2 GERA FIRST FIRST :RN LAST FIRST :RN :VISTO MA-
KE "SUSSECO 0 MAKE "FRACASSO 0 MAKE "PER "F
LABEL "LOOP3
IF :COMB2 = [] [GO "FORA23]
MAKE "COMB3 ITEM ((RANDOM COUNT :COMB2) + 1) :COMB2 MAKE
"COMB2 ELIM :COMB3 :COMB2 []
IF OR (MEMBERP (LPUT [1] :COMB3) :COMBJA) (MEMBERP (LPUT
[0] :COMB3) :COMBJA) [GO "FORA2]
IF EQUALP ( REDUZ :COMB3 :TAB ((FIRST FIRST ITEM LAST
FIRST :RN LAST :TAB) / (LAST FIRST ITEM LAST FIRST :RN
LAST :TAB)) ) "FRACASSO [GO "FORA2]
IF AND (NOT :SUSSECO = 0) (:SUSSECO / 3 = INT :SUSSECO /
3) [PR JUNTA [VOCE QUER QUE EU INVESTIGUE MAIS] JUNTA
FIRST FIRST :RN JUNTA [COMO] JUNTA DECODIGO LAST FIRST :RN
```

```
[?] PR [RESPONDA S / N] MAKE "L READLIST IF OR :L = [N] :L
= [NAO] [GO "FORA23]]
IF MEMBERP :COMB3 JUNTA :TC1 :TC1L [MAKE "COMB3 LPUT [1]
:COMB3] [IF MEMBERP :COMB3 JUNTA :TC0 :TC0L [MAKE "COMB3
LPUT [0] :COMB3] [CT PR JUNTA [VEJA UM DESENHO COM] JUNTA
FIRST FIRST :RN JUNTA [COMO] DECODIGO LAST FIRST :RN MAKE
"COMB3: PARTE :MODELO :COMB3]]
MAKE "CCOMB LPUT :COMB3: []
MAKE "I CHECAREFUTA LIST :FS. :RS. []
MAKE "I CHECAREFUTA :DUV1 :DUVIDA IF :I = "T [MAKE "DUVIDA
:QUAL2 MAKE "DUV1 REFUTUM :DUVIDA [[]]]
IF LAST :COMB3: = [1] [MAKE "TC1L LPUT BL :COMB3: :TC1L]
MAKE "SC1 LPUT BL :COMB3: :SC1 MAKE "CCOMB LPUT :COMB3: []
IF :PER = "T [MAKE "I CHECAREFUTA :POD1 :PODA IF :I = "T
[MAKE "PODA :QUAL2 MAKE "POD1 REFUTUM :PODA [[]]]] [MAKE
"TC0L LPUT BL :COMB3: :TC0L]
IF OR :PER = "F LAST :COMB3: = [1] [MAKE "TAB TABELA 1
:COMB3: FIRST :TAB LAST :TAB [] [] MAKE "COMB4 LPUT
:COMB3: :COMB4]
MAKE "SUSSECO :SUSSECO + 1
LABEL "FORA2
GO "LOOP3
LABEL "FORA23
PR JUNTA [VEJAMOS RESULTADOS DE] JUNTA FIRST FIRST :RN
JUNTA [COMO] DECODIGO LAST FIRST :RN
IMPRIMA :TAB FIRST FIRST :PN LAST FIRST :RN "CONT
RECYCLE
MAKE "VISTONAO AVALIA :TAB :COMB4 FIRST FIRST :RN LAST
FIRST :RN :VISTO [] [] :PROVIS
IF EQUALP FIRST :VISTONAO "FRACASSO [PR LPUT FIRST FIRST
:RN [NAO DESCOBRI QUANDO DA CERTO] GO "FORA3]
MAKE "VISTO FIRST FIRST :VISTONAO
IF NOT EQUALP FIRST :RN LIST :A :P [MAKE "PONTOS PONDERA
:PONTOS LAST FIRST :RN]
LABEL "FORA3
RECYCLE
MAKE "RN BF :RN GO "LOOP1
END
```