determined. When this generator triggers, more hypotheses are generated to get a finer partition of the range, and so on.

The enhancements to the linear programming paradigm of relaxation give some idea of the flexibility of the basic idea, but also reveal that the method is not at all cut-and-dried, and is still open to basic investigation. One of the questions about the method is exactly how to take advantage of parallel computation capabilities. Each constraint and hypothesis can be given its own processor, but how should they communicate? Also, there seems little reason to suppose that the optimization problems for this form of relaxation are any easier than they are for any other multidimensional search, so the method will encounter the usual problems inherent in such optimization. However, despite all these technical details and problems of implementation, the linear programming paradigm for the relaxation computation is a coherent formalization of the process. It provides a relatively "classical" context of results and taxonomy of problems [Hummel and Zucker 1980].

12.5 ACTIVE KNOWLEDGE

Active knowledge systems [Freuder 1975] are characterized by the use of procedures as the elementary units of knowledge (as opposed to propositions or data base items, for instance). We describe how active knowledge might work, because it is a logical extreme of the procedural implementation of propositions. In fact, this style of control has not proven influential; some reasons are given below.

Active knowledge is notionally parallel and heterarchical. Many different procedures can be active at the same time depending on the input. For this reason active knowledge is more easily applied to belief maintenance than to planning; it is very difficult to organize sequential activity within this discipline. Basically, each procedure is responsible for a "chunk" of knowledge, and knows how to manage it with respect to different visual inputs. Control in an active knowledge system is completely distributed. Active knowledge can also be viewed as an extension of the constraint relaxation problem; powerful procedures can make arbitrary detailed tests of the consistency between constraints.

Each piece of active knowledge (program module) knows which other modules it depends on, which depend on it, which it can complain to, and so forth. Thus the choice of "what to do next" is contained in the modules and is not made by an exterior executive.

We describe HYPER, a particular active knowledge system design which illustrates typical properties of active knowledge [Brown 1975]. HYPER provides a less structured mechanism for construction and exploration of hypotheses than does LP-relaxation. Using primitive control functions of the system, the user may write programs for establishing hypotheses and for using the conclusions so reached. The programs are "procedurally embedded" knowledge about a problem domain (e.g. how events relate one to another, what may be conjectured or inferred from a clue, or how one might verify a hypothesis).

When HYPER is in use on a particular task in a domain, hypotheses are created, or instantiated, on the basis of low-level input, high-level beliefs, or any reason in between. The process of establishing the initial hypotheses leads to a propagation of activity (creation, verification, and disconfirmation of hypotheses). Activation patterns will generally vary with the particular task, in heterarchical fashion. A priority mechanism can rank hypotheses in importance depending on the data that contribute to them. Generally, the actions that occur are conditioned by previous assumptions, the data, the success of methods, and other factors. HYPER can be used for planning applications and for multistep vision processing as well as inference (procedures then should generate parallel activity only under tight control). We shall thus allow HYPER to make use of a context-oriented data base (Section 13.1.1). It will use the context mechanism to implement "alternative worlds" in which to reason.

12.5.1 Hypotheses

A HYPER hypothesis is the attribution of a predicate to some arguments; its name is always of the form (PREDICATE ARGUMENTS). Sample hypothesis names could be (HEAD-SHAPED REGION1), (ABOVE A B), (TRIANGLE (X1,Y1) (X2,Y2) (X3,Y3)). A hypothesis is represented as a data structure with four components; the *status*, *contents*, *context*, and *links* of the hypothesis.

The *status* represents the state of the HYPER's knowledge of the truth of the hypothesis; it may be T(rue), F(alse), (in either case the hypothesis has been *established*) or P(ending). The *contents* are arbitrary; hypotheses are not just truth-valued assertions. The hypothesis was asserted in the data-base context given in *context.* The *links* of a hypothesis H are pointers to other hypotheses that have asked that H be established because they need H's contents to complete their own computations.

12.5.2 HOW-TO and SO-WHAT Processes

Two processes are associated with every predicate P which appears as the predicate of a hypothesis. Their names are (HOW-TO P) and (SO-WHAT P). In them is embedded the procedural knowledge of the system which remains compiled in from one particular task to another in a problem domain. (HOW-TO P) expresses how to establish the hypothesis (P arguments). It knows what other hypotheses must be established first, the computations needed to establish (P arguments), and so forth. It has a backward-chaining flavor. Similarly, (SO-WHAT P) expresses the consequences of knowing P: what hypotheses could possibly now be established using the contents of (P arguments), what alternative hypotheses should be explored if the status of (P arguments) is *F*, and so on. The feeling here is of forward chaining.

12.5.3 Control Primitives

HYPER hypotheses interact through *primitive control statements*, which affect the investigation of hypotheses and the ramification of their consequences. The primitives are used in HOW-TO and SO-WHAT programs together with other general computations. Most primitives have an argument called priority, which expresses the reliability, urgency, or importance of the action they produce, and is used to schedule processes in a nonparallel computing environment (implemented as a priority job queue [Appendix 2]). The primitives are GET, AFFIRM, DENY, RETRACT, FAIL, WONDERIF, and NUDGE.

GET is to ascertain or establish the status and contents of a hypothesis. It takes a hypothesis H and priority PRI as arguments and returns the status and contents of the hypothesis. If H's status is T or F at the time of execution of the statement, the status and contents are returned immediately. If the status is P (pending), or if H has not been created yet, the current HOW-TO or SO-WHAT program calling GET (call it CURPROG) is exited, the proper HOW-TO job (i.e., the one that deals with H's predicate) is run at priority PRI with argument H, and a link is planted in H back to CURPROG. When H is established, CURPROG will be reactivated through the link mechanism.

AFFIRM is to assert a hypothesis as true with some contents. AFFIRM(H,CONT,PRI) sets H's status to T, its contents to CONT, activates its linked programs and then executes the proper SO-WHAT program on it. The newly activated SO-WHAT programs are performed with priority PRI.

DENY is to assert that a hypothesis with some contents is false. DENY(H,CONT,PRI) is like AFFIRM except that no activation though links occurs, and the status of H is of course set to F.

ASSUME is to assert a hypothesis as true hypothetically. ASSUME(H,CONT,PRI) uses the data base context mechanism to create a new context in which H is AFFIRMED; the original context in which the ASSUME command is given is preserved in the context field of H. H itself is stored into a context-dependent item named LASTASSUMED; this corresponds to remembering a decision point in PLANNER. By using the information in LASTASSUMED and the primitive FAIL (see below), simple backtracking can take place in a tree of contexts.

RETRACT(H) establishes as false a hypothesis that was previously ASSUMEd. RETRACT is always carried out at highest priority, on the principle that it is good to leave the context of a mistaken assumption as quickly as possible. Information (including the name of the context being exited) is transmitted back to the original context in which H was ASSUMEd by passing it back in the fields of H.

FAIL just RETRACTs the hypothesis that is the value of the item LASTASSUMED in the present context.

WONDERIF is to pass suggested contents to HOW-TO processes for verification. It can be useful if verifying a value is easier than computing it from scratch, and is the primitive that passes substantive suggestions. WONDERIF(H1, CONT, H2, PRI) approximates the notion "H2 wonders if H1 has contents CONT."

NUDGE is to wake up HOW-TO programs. NUDGE(H,PRI) runs the HOW-TO program on H with priority PRI. It is used to awaken hypotheses that might be able to use information just computed. Typically it is a SO-WHAT program that NUDGEs others, since the SO-WHAT program is responsible for using the fact that a hypothesis is known.

12.5.4 Aspects of Active Knowledge

The active knowledge style of computation raises a number of questions or problems for its users.

A hypothesis whose contents may attain a large range can be established for some contents and thus express a perfectly good fact (e.g., that a given location of an x-ray does not contain evidence for a tumor) but such a fact is usually of little help when we want to reason about the predicate (about the location of tumors). The SO-WHAT program for a predicate should be written so as to draw conclusions from such negative facts if possible, and from the conclusions endeavor to establish the hypothesis as true for some contents. Usually, therefore, it would set the status of the hypothesis back to P and initiate a new line of attack, or at its discretion abandon the effort and start an entirely new line of reasoning.

Priorities

A major worry with the scheme as described is that priorities are used to schedule running of HOW-TO and SO-WHAT processes, not to express the importance (or supposition value) of the hypotheses. The hypothesis being investigated has no way to communicate how important it is to the program that operates on it, so it is impossible to accumulate importance through time. A very significant fact may lie ignored because it was given to a self-effacing process that had no way of knowing it had been handed something out of the ordinary.

The obvious answer is to make a supposition value a field of the hypothesis, like its status or contents—a hypothesis should be given a measure of its importance. This value may be used to compute execution priorities for jobs involving it. This solution is used in some successful systems [Turner 1974].

Structuring Knowledge

One has a wide choice in how to structure the "theory" of a complex problem in terms of HYPER primitives, predicates, arguments, and HOW-TO and SO-WHAT processes. The set of HOW-TO and SO-WHAT processes specify the complete theory of the tasks to be performed; HYPER encourages one to consider the interrelations between widely separated and distinct-sounding facts and conjectures about a problem, and the structure it imposes on a problem is minimal.

Since HOW-TO and SO-WHAT processes make explicit references to one another via the primitives, they are not "modular" in the sense that they can easily be plugged in and unplugged. If HOW-TO and SO-WHAT processes are invoked by patterns, instead of by names, some of the edge is taken off this criticism. Removing a primitive from a program could modify drastically the avenues of activation, and the consequences of such a modification are sometimes hard to foresee in a program that logically could be running in parallel.

Writing a large and effective program for one domain may not help to write a program for another domain. New problems of segmenting the theory into predicates, and quantifying their interactions via the primitives, setting up a priority structure, and so forth will occur in the new domain, and it seems quite likely that little more than basic utility programs will carry over between domains.

EXERCISES

12.1 In the production system example, write a production that specifies that blue regions are sky using the opponents color notation. How would you now deal with blue regions that are lakes (a) in the existing color-only system; (b) in a system which has surface orientation information?

12.2 This theorem was posed as a challenge for a clausal automatic theorem prover [Henschen et al. 1980]. It is obviously true: what problems does it present?

$$\begin{gathered}\{[(\exists x)(\forall y)(P(x) \Longleftrightarrow P(y))] \\ \Longleftrightarrow [[(\exists x)Q(x)\} \Longleftrightarrow [(\forall y)(P(y))]]\} \Longleftrightarrow \\ \{[(\exists x)(\forall y)(Q(x) \Longleftrightarrow Q(y))] \\ \Longleftrightarrow [[(\exists x)P(x)] \Longleftrightarrow [(\forall y)(Q(y))]]\}\end{gathered}$$

12.3 Prove that the operator of Eq. (12.18) takes probability vectors into probability vectors, thus deriving the reason for Eq. (12.19).

12.4 Verify (12.23).

12.5 How do the c_{ij} of (12.18) affect the labeling? What is their semantics?

12.6 If events X and Y always co-occur, then $p(X, Y) = p(X) = p(Y)$. What is the correlation in this case? If X and Y never co-occur, what values of $p(X)$ and $p(Y)$ produce a minimum correlation? If X and Y are independent, how is $p(X, Y)$ related to $p(X)$ and $p(Y)$? What is the value of the correlation of independent X and Y?

12.7 Complete Table 12.3.

12.8 Use only the labels of Fig. 12.9b and c to compute covariances in the manner of Table 12.3. What do you conclude?

12.9 Show that Eq. (12.29) preserves the important properties of the weight vectors.

12.10 Think of some rival normalization schemes to Eq. (12.29) and describe their properties.

12.11 Implement the linear and nonlinear operators of Section 12.4.3 and 12.4.4 and investigate their properties. Include your ideas from Exercise 12.10.

12.12 Show a case that the nonlinear operator of Eq. (12.29) assigns nonzero weights to maximally incompatible labels (those with $r_{ij} = -1$).

12.13 How can a linear programming relaxation such as the one outlined in sec. 12.4.5 cope with faces or edges of the feasible solution solid that are normal to the preference direction, yielding several solutions of equal preference?

12.14 In Fig. 12.11, what (P, Q) solution is optimal if the preference vector is $(1, 4)$? $(4, 1)$? $(-1, 1)$? $(1, -1)$?

REFERENCES

AIKINS, J. S. "Prototypes and production rules: a knowledge representation for computer consultations." Ph.D. dissertation, Computer Science Dept., Stanford Univ., 1980.

BAJCSY, R. and A. K. JOSHI. "A partially ordered world model and natural outdoor scenes." In *CVS*, 1978.

BARROW, H. G. and J. M. TENENBAUM. "MSYS: a system for reasoning about scenes." Technical Note 121, AI Center, SRI International, March 1976.

BRACHMAN, R. J. "On the epistemological status of semantic networks." In *Associative Networks: Representation and Use of Knowledge by Computers*, N. V. Findler (Ed.). New York: Academic Press, 1979, 3–50.

BROWN, C. M. "The HYPER system." DAI Working Paper 9, Dept. of Artificial Intelligence, Univ. Edinburgh, July 1975.

BUCHANAN, B. G. and E. A. FEIGENBAUM. "DENDRAL and meta-DENDRAL: their applications dimensions." *Artificial Intelligence 11*, 2, 1978, 5–24.

BUCHANAN, B. G. and T. M. MITCHELL. "Model-directed learning of production rules." In *Pattern Directed Inference Systems,* D. A. Waterman and F. Hayes-Roth (Eds.). New York: Academic Press, 1978.

COLLINS, A. "Fragments of a theory of human plausible reasoning." *Theoretical Issues in Natural Language Processing-2*, Univ. Illinois at Urbana-Champaign, July 1978, 194–201.

DAVIS, R. and J. KING. "An overview of production systems." AIM-271, Stanford AI Lab, October 1975.

DAVIS, L. S. and A. ROSENFELD. "Applications of relaxation labelling 2. Spring-loaded template matching." Technical Report 440, Computer Science Center, Univ. Maryland, 1976.

DELIYANNI, A. and R. A. KOWALSKI. "Logic and semanatic networks." *Comm. ACM 22*, 3, March 1979, 184–192.

ERMAN, L. D. and V. R. LESSER. "A multi-level organization for problem solving using many, diverse, cooperating sources of knowledge." *Proc.*, 4th IJCAI, September 1975, 483–490.

FELDMAN, J. A. and Y. YAKIMOVSKY. "Decision theory and artificial intelligence: I. A semantics-based region analyser." *Artificial Intelligence 5*, 4, 1974, 349–371.

FIKES, R. E. "Knowledge representation in automatic planning systems." In *Perspectives on Computer Science*, A. Jones (Ed). New York: Academic Press, 1977.

FIKES, R. E. and N. J. NILSSON. "STRIPS: a new approach to the application of theorem proving to problem solving." *Artificial Intelligence 2*, 3/4, 1971, 189–208.

FREUDER, E. C. "A computer system for visual recognition using active knowledge." Ph.D. dissertation, MIT, 1975.

FREUDER, E. C. "Synthesizing constraint expressions." *Comm. ACM 21*, 11, November 1978, 958–965.

GARFINKEL, R. S. and G. L. NEMHAUSER. *Integer Programming.* New York: Wiley, 1972.

GOMORY, R. E. "An algorithm for integer solutions to linear programs." *Bull. American Mathematical Society 64*, 1968, 275–278.

HARALICK, R. M. "The characterization of binary relation homomorphisms." *International J. General Systems 4*, 1978, 113–121.

HARALICK, R. M. and J. S. KARTUS. "Arrangements, homomorphisms, and discrete relaxation." *IEEE Trans. SMC 8*, 8, August 1978, 600–612.

HARALICK, R. M. and L. G. SHAPIRO. "The consistent labeling problem: Part I." *IEEE Trans. PAMI 1*, 2, April 1979, 173–184.

HARALICK, R. M., L. S. DAVIS, and A. ROSENFELD. "Reduction operations for constraint satisfaction." *Information Sciences 14*, 1978, 199–219.

HAYES, P. J. "In defense of logic." *Proc.*, 5th IJCAI, August 1977, 559–565.

HAYES, P. J. "Naive physics: ontology for liquids." Working paper, Institute for Semantic and Cognitive Studies, Geneva, 1978a.

HAYES, P. J. "The naive physics manifesto." Working paper, Institute for Semantic and Cognitive Studies, Geneva, 1978b.

HAYES, P. J. "The logic of frames." *The Frame Reader.* Berlin: DeGruyter, in press, 1981.

HENDRIX, G. G. "Encoding knowledge in partitioned networks." In *Associative Networks: Representation and Use of Knowledge by Computers*, N. V. Findler (Ed.). New York: Academic Press, 1979, 51–92.

HENSCHEN, L., E. LUSK, R. OVERBEEK, B. SMITH, R. VEROFF, S. WINKER, and L. WOS. "Challenge Problem 1." *SIGART Newsletter 72*, July 1980, 30–31.

HERBRAND, J. "Recherches sur la théorie de la démonstration." *Travaux de la Société des Sciences et des Lettres de Varsovie, Classe III, Sciences Mathématiques et Physiques, 33,* 1930.

HEWITT, C. "Description and theoretical analysis (using schemata) of PLANNER" (Ph.D. dissertation). AI-TR-258, AI Lab, MIT, 1972.

HINTON, G. E. "Relaxation and its role in vision." Ph.D. dissertation, Univ. Edinburgh, December 1979.

HUMMEL, R. A. and S. W. ZUCKER. "On the foundations of relaxation labelling processes." TR-80-7, Computer Vision and Graphics Lab, Dept. of Electrical Engineering, McGill Univ., July 1980.

KOWALSKI, R. A. "Predicate logic as a programming language." *Information Processing 74.* Amsterdam: North-Holland, 1974, 569–574.

KOWALSKI, R. A. *Logic for Problem Solving.* New York: ElsevierNorth-Holland (AI Series), 1979.

LINDSAY, R. K., B. G. BUCHANAN, E. A. FEIGENBAUM, and J. LEDERBERG. *Applications of Artificial Intelligence to Chemistry: The DENDRAL Project.* New York: McGraw-Hill, 1980.

LOVELAND, D. "A linear format for resolution." *Proc.*, IRIA 1968 Symp. on Automatic Demonstration, Versailles, France. New York: Springer-Verlag, 1970.

LOVELAND, D. *Automated Theorem Proving: A Logical Basis.* Amsterdam: North-Holland, 1978.

MCCARTHY, J. "Circumscription induction—a way of jumping to conclusions." Unpublished report, Stanford AI Lab, 1978.

MCCARTHY, J. and P. J. HAYES. "Some philosophical problems from the standpoint of artificial intelligence." In *MI4*, 1969.

MCDERMOTT, D. "The PROLOG phenomenon." *SIGART Newsletter 72*, July 1980, 16–20.

MENDELSON, E. *Introduction to Mathematical Logic.* Princeton, NJ: D. Van Nostrand, 1964.

MINSKY, M. L. "A framework for representing knowledge." In *PCV*, 1975.

NEWELL, A., J. SHAW, and H. SIMON. "Empirical explorations of the logic theory machine." In *Computers and Thought*, E. Feigenbaum and J. Feldman (Eds.). New York: McGraw-Hill, 1963.

NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

NILSSON, N. J. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.

REITER, R. "On reasoning by default." *Theoretical Issues in Natural Language Processing-2*, Univ. Illinois at Urbana-Champaign, July 1978, 210–218.

ROBINSON, J. A. "A machine-oriented logic based on the resolution principle." *J. ACM 12*, 1, January 1965, 23–41.

ROSENFELD, A., R. A. HUMMEL and S. W. ZUCKER. "Scene labelling by relaxation operations." *IEEE Trans. SMC 6*, 1976, 420.

RYCHNER, M. "An instructable production system: basic design issues." In *Pattern Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (Eds.). New York: Academic Press, 1978.

SHORTLIFFE, E. H. *Computer-Based Medical Consultations: MYCIN.* New York: American Elsevier, 1976.

SLOAN, K. R. "World model driven recognition of natural scenes." Ph.D. dissertation, Moore School of Electrical Engineering, Univ. Pennsylvania, June 1977.

SLOAN, K. R. and R. BAJCSY. "World model driven recognition of outdoor scenes." TR40, Computer Science Dept., Univ. Rochester, September 1979.

SUSSMAN, G. J. and D. MCDERMOTT. "Why conniving is better than planning." AI Memo 255, AI Lab, MIT, 1972.

TURNER, K. J. "Computer perception of curved objects using a television camera." Ph.D. dissertation, School of Artificial Intelligence, Univ. Edinburgh, 1974.

WARREN, H. D., L. PEREIRA, and F. PEREIRA. "PROLOG: The language and its implementation compared with LISP." *Proc.*, Symp. on Artificial Intelligence and Programming Languages, SIGPLAN/SIGART, 1977; *SIGPLAN Notices 12*, 8, August 1977, 109–115.

WATERMAN, D. A. and F. HAYES-ROTH (Eds.). *Pattern-Directed Inference Systems.* New York: Academic Press, 1978.

WINOGRAD, T. "Extended inference modes in reasoning by computer systems." *Proc.*, Conf. on Inductive Logic, Oxford Univ., August 1978.

ZADEH, L. "Fuzzy sets." *Information and Control 8*, 1965, 338–353.

ZUCKER, S. W. "Relaxation labelling and the reduction of local ambiguities." Technical Report 451, Computer Science Dept., Univ. Maryland, 1976.