Incorporating Dynamic Control into the Model Generative Reasoning System

C. A. Fields, M. J. Coombs, E. S. Dietrich, and R. T. Hartley
Knowledge Systems Group
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003-0001, USA

Introduction.

Model -Generative Reasoning (MGR) is an automated problem solving architecture designed for application in task environments in which problems are novel and data arc noisy, incomplete, or of uncertain relevance (Hartley et al., 1987; Coombs and Hartley, 1987; Fields ct al., in press). The current prototype MGR system employs a sequential reasoning strategy, the *MGR algorithm* (Coombs and HartIcy, 1987). This problem solver is capable of generating models of events for which it has no stored schema by decomposing stored schemata representing similar events, and recombining the components into a new model. It can, therefore, function successfully in task environments in which a restricted class of novel events can arise, and in which irrelevant as well as relevant information is included in both the input and the knowledge base. The prototype has been applied to process control (Coombs and Hartley, 1988) and robot path planning (Eshner et al., in press) problems; current application areas include meteorological data fusion (Coombs et al., 1988) and DNA sequence analysis.

The problem solving strategy employed by the current MGR prototype is useful in some domains; however, it does not permit sufficiently plastic behavior in the face of some forms of novelty or incomplete data. If plasticity in the face of novelty is regarded as a measure of intelligence (Fields and Dietrich, 1987a), the use of a fixed strategy must be regarded as a deficit in a problem solver. One approach to alleviating the limitations imposed by a fixed problem solving strategy is to design a strategy that allows the problem solver to emulate the behavior that would result from applying a task-appropriate Method to each individual task. This is the general approach taken by Newell's group in the design of their "Universal Weak Method" (Laird and Newell, 1983; Laird et al., 1987). The approach that we have taken has similar motivations; however, instead of designing a "universal" symbolic weak method that allows emulation of alternative strategics, we have designed a concurrent, dynamic control structure that allows the arbitrary superposition of strategies within a single execution cycle. This structure will allow problem solvers based on the MGR architecture to, in effect, arbitrarily integrate alternative strategies into new methods customized to the particular problem at hand.

The MGR Architecture.

The MGR architecture is described in Fields et al. (in press). Viewed abstractly, the architecture comprises two subsystems, one of which constructs models from facts provided as input together with stored knowledge, and the other of which evaluates the resulting models with respect to addifional facts. This organization reflects the view that abductive problem solving is a process of *equilibration* between the perhaps conflicting demands of observations and expectations, and is motivated by general systems theoretic (Ashby, 1952; Pask, 1975), developmental (Piaget, 1977), and simulated neural network (Grossberg, 1980; 1987) analyses of problem solving and learning.

MGR uses a knowledge representation based on the conceptual graphs formalism of Sowa (1984). Conceptual graphs in MGR include executable representations of both qualitative spatio-temporal relations and continuous-valued constraints. Concepts in MGR are arranged in a type hierarchy, which implicitly defines specialization and generalization relations. No form of inheritance is used in MGR. Conceptual graphs are manipulated using four operators, **Specialize, Generalize, Merge, and Fragment,** which are implemented as modules using the maximal join and maximal project operations on conceptual graphs.

MGR is logically a shared-data parallel machine, in which each of the four graph- manipul ati on operators, as well as the ancillary constraint execution and model evaluation modules, is a single-instruction multiple-data (SIMD) machine. The constraint execution and model evaluation modules run whenever their input requirements arc satisfied, and are independent of the graph manipulation modules. The four graph manipulation operators are dynamically coupled, and are controlled by varying their rates of execution in response to changes in the model

population.

Concurrent Dynamic Control.

Let X(t) represent the state of the population of graphs available to the system at time t, i.e. $\mathbf{X}(t) = <\mathbf{D}(t), \mathbf{F}(t), \mathbf{M}(t)>$, where **D**(t), **F**(t), and **M**(t) represent the definition, fact, and model populations, respectively. This state is *represented* to the control structure as a state vector $\Psi(\mathbf{X})$. The components of $\Psi(\mathbf{X})$ are functions that measure properfies of the three populations of graphs.

Control can be either *deterministic,* in which case it is defined over particular graphs, or *stochastic,* in which case it is defined over populations of graphs that are taken to behave as statistical ensembles. If control is defined deterministically, the result of a particular execution cycle *will* be predictable, up to the nondeterminism of the individual operators. If control is stochastic, the result of a particular execution cycle will not be predictable, but the average results of an ensemble of cycles of execution on the same graph populations will be predictable. Stochastic control increases the plasticity of the system in the face of novel or incompletely-specified problems (Fields and Dietrich, 1987a; b); however, deterministic control is preferable from a software engineering perspective if the task environment is well-enough structured to choose appropriate selection functions to specify the ways in which graphs are selected for input into each operator.

The deterministic - stochastic decision determines the state functions used as the components of the state vector. In the case of deterministic control, the state functions are:

i. the *complexity* of a graph g_i, $\sigma(g_i)$ = *[# of arcs in* g_i*]*. This function provides a meaningful measure of complexity, since no two nodes in a graph constructed in CP can be connected by more than one arc.

ii. the *covering index* for the pair of graphs g_i and g_j, $\chi(g_i, g_j)$, *is* defined to be the number of node labels shared by the graphs g_i and g_j. *

in the case of stochastic control, the state functions are:

i. the *entropy* of the j^{th} population of graphs $\nu(\mathbf{X}_j) = -1n$ [# of graphs in $\mathbf{X}_j$], which measures the relative likelihood that any particular graph in the population will be operated upon in any cycle,

ii. the *average complexity* of the j^{th} population of graphs $\sigma(\mathbf{X}_j)$ = [# arcs in $\mathbf{X}_j$]/[# of graphs in $\mathbf{X}_j$].

iii. the *average covering index* $\chi(\mathbf{X}_i, \mathbf{X}_j)$ for the i^{th} and j_{th} populations of graphs is the average of the covering indices $\chi(g_i, g_j)$, where g_i is in the i^{th} population and g_j is in the j^{th} population.

iv. the *maximum covering index,* $\chi_m(\mathbf{X}_i, \mathbf{X}_j)$ for the i^{th} and j_{th} populations of graphs is the maximum of the covering indices $\chi(g_i, g_j)$, where g_i is a member of the i^{th} population and g_j is a member of the j^{th} population.

These functions are clearly linearly independent. The state vector in the deterministic case is, therefore:

$$\Psi(g_i, g_j) = <\sigma(g_i), \sigma(g_j), \chi(g_i, g_j)>,$$

for graphs g_i and g_j. In the stochastic case, the state vector is:

$$\Psi(<\mathbf{X}_i>) = <\nu(\mathbf{X}_i), \sigma(\mathbf{X}_i), \chi(\mathbf{X}_i, \mathbf{X}_j), \chi_m(\mathbf{X}_i, \mathbf{X}_j)>.$$

The control function itself is defined in terms of cost and response functions, which depend on the state functions. Each operation is assumed to require a time that depends on the complexity of the graphs being operated upon. This *cost* is represented for operator *i* as a function $\xi_i(\sigma(\oplus \mathbf{X}_i))$, where $\oplus \mathbf{X}_i$ represents the direct sum (disjoint union) of the graph populations on which the operator is defined, and $\sigma(\oplus \mathbf{X}_i)$ represents the sum of the complexities, or in the stochastic case, the average complexities of the graphs in these populations.

The MGR operators are coupled in their global effects on the populations of graphs. The coupling between operators *i* and *j* is represented by a function ξ_{ij}, which is assumed to act on the state vector Ψ. This correlation function represents the *response* of the correlated pair ij of operators to the current state of the system; the control function thus acts on correlated pairs of operators through the response functions, and on single operators through the cost functions.

Given these definitions, the change per unit time of the *firing rate* $\mathbf{R}_i$ of the i^{th} operator is given by:

$$\delta\mathbf{R}_i(\mathbf{X})/\delta t = \xi_i(\sigma(\oplus \mathbf{X}_i)) \Sigma\{[[\delta_i^{\alpha} + \delta_i^{\beta}]\xi_{\alpha\beta}(\Psi(\mathbf{X}_i))\},$$

where δ_i^{α} is the Kronecker delta function, i.e. $\Sigma\, \delta_i^{\alpha}\, \phi_{\alpha} = \phi_i$. The firing rate $\mathbf{R}_i$ of operator *i* is the time integral of this expression. The *rate vector* of the system is the vector $\mathbf{R}(t) = <\mathbf{R}_i(t)>$ of these operator firing rates. This vector describes the behavior of the system through time, and is taken to define the control structure of concurrent dynamic MGR.

The above expression can be simplified considerably if the correlation functions $\xi_{\alpha\beta}$ are assumed to be linear. In this case, $\xi_{\alpha\beta}(\Psi)$ can be represented as the inner product $\xi_{\alpha\beta} * \Psi$. The sum of the correlation functions involving the i^{th} operator, $\Sigma[\delta_i^{\alpha} + \delta_i^{\beta}]\xi_{\alpha\beta}$, can then be represented

as a single correlation vector ξ_i. The expression for the inner products of the correlation vectors with the state vec-
- tor reduces to $\Sigma\, \xi_i^{\gamma} \Psi_{\gamma}(\mathbf{X}_i)$, and the expression for the change in firing rate of the i^{th} operator becomes:

$$\delta \mathbf{R}_i(\mathbf{X})/\delta t = \xi_i\, (\sigma(\oplus\, \mathbf{X}_i))\, \Sigma\, \xi_i^{\gamma} \Psi_{\gamma}(\mathbf{X}_i).$$

This expression is taken to define the control structure of *linear* concurrent dynamic MGR.

The above expression is semilinear, and is indeed isomorphic to the semilinear node functions that characterize most parallel distributed processing (PDP) systems (Rumelhart et al., 1986). The correlation vectors ξ_{ij} can, therefore, be regarded as the components of the weight matrix of a PDP network. The approach to dynamic control outlined above can thus be thought of, in the special case of linear correlation functions, as equivalent to using a PDP machine to calculate the rates of application for a set of concurrent symbolic operators.

This fusion of symbolic and dynamic reasoning strategies in a single system represents a novel approach to problem solving architecture. We expect that hybrid systems such as this one, in which strategies can be superposed in' real time while solving a problem, will be required for robust automated problem solving in open task environments.

References.

Ashby, W. Ross (1952) *Design Jbr a Brain.* London: Chapman and Hall.

Coombs, M. J. and R. T. Hartley (1987) The MGR algorithm and its application to the generation of explanations for novel events. *Int. J. Man-Machine Studies* 27: 1-30

Coombs, M. J. and R. T. Hartley (1988) Explaining novel events in process control through model generative reasoning. *Int. J. Expert Systems: Research and Applications 1:* 89-109.

Coombs, M. J., C. A. Fields, and G. McWilliams (1988) *Artificial Intelligence Methods for Optimizing the Use of Meteorological Databases: Recommendations for Implementing the MERCURY System.* Final Report, WAO 87-2.10-5, Contract # DAAD07-86-C-0034. U.S. Army Atmospheric Sciences Laboratory.

Eshner, D. P., R. T. Hartley, R. C. Lennox, and M. M. Moya (in press) Conceptual representation for robot task planning. In: J. Sowa, N. Foo, and P. Rao (Eds) *Applications of Conceptual Graphs.* Reading, MA: Addison-Wesley.

Fields, C. and E. Dietrich (1987a) Multi-domain problem solving: A test case for computational theories of intelligence. *Proc. Second Rocky Mountain Conf. on Al.* University of Colorado. pp. 205-223.

Fields, C. and E. Dietrich (1987b) A stochastic computing architecture for multi-domain problem solving. *Proc. Second Int. Symp. on Methodologies for Intelligent Systems.* CoIloquirn Volume. Oak Ridge National Laboratory (ORNL-6417). pp. 227-238.

Fields, C. A., M. J. Coombs, and R. T. Hartley (in press) MGR: An architecture for problem solving in unstructured environments. *Proc. Third Int. Symp. on Methodologies for Intelligent Systems.*

Grossberg, S. (1980) How does the brain build A cognitive code? *Psych. Rev.* 87: 1-51.

Grossberg, S. (1987) Competitive learning: From interactive activation to adaptive resonance. *Cog. Sci. 11:* 2363.

Hanley, R. T., M. J. Coombs, and E. Dietrich (1987) An algorithm for open-world reasoning using model gcncration. *Proc. Second Rocky Mountain* Conf. *on AI.* University of Colorado. pp. 193-203.

Laird, J. E. and A. Newell (1983) A universal weak method: Summary of results. Proc. IJCAI-83. pp. 771-773.

Laird, J. E., A. Newell, and P. S. Rosenbloom (1987) SOAR: An architecture for general intelligence. *Art. Intell.* 33: 1-64.

Pask, G. (1975) *Conversation, Cognition, and Learning.* Amsterdam: Elsevier.

Piaget, J. (1977) *The Development of Thought.- Equilibration of Cognitive Structures.* New York: Viking.

Rumelhart, D. E., G. E. Hinton, and J. L. McClelland (1986) A general framework for parallel distributed processing. In: D. Rumclhart and J. McClelland (Eds) *Parallel Distributed Processing.* Vol. 1. Cambridge, MA: MIT/Bradford. pp. 45-76.

Sowa, J. (1984) *Conceptual Structures.* Reading, MA: Addison-Wesley.