Connectionist Variable-Binding By Optimization

P. Anandan
Computer Science Department, Yale University

Stanley Letovsky
Computer Science Department, Carnegie-Mellon University

Eric Mjolsness
Computer Science Department, Yale University

Abstract

Symbolic AI systems based on logical or frame languages can easily perform inferences that are still beyond the capability of most connectionist networks. This paper presents a strategy for implementing in connectionist networks the basic mechanisms of variable binding, dynamic frame allocation and equality that underlie many of the types of inferences commonly handled by frame systems, including inheritance, subsumption and abductive inference. The paper describes a scheme for translating frame definitions in a simple frame language into objective functions whose minima correspond to partial deductive closures of the legal inferences. The resulting constrained optimization problem can be viewed as a specification for a connectionist network.

1 INTRODUCTION

Connectionist systems are attractive as an approach to computing because they promote such desirable properties as fine-grained parallelism, analog circuitry, fault tolerance, and automatic learning. One of the more potent ideas to have appeared in investigations of these systems, and one which underlies a large fraction of the work in the field, is the use of continuous "objective functions" or "distance metrics". Objective functions can serve as a perspicuous programming language, highly susceptible to analysis, and useful as a specification language for neural networks. This paper uses the objective-function paradigm to address a central limitation of most existing connectionist systems: their inability to perform the kind of inferences that are easy for symbolic AI systems based on logical or frame languages. We present an objective-function-based implementation of the basic mechanisms of variable-binding and dynamic frame allocation that underlie frame based inference. The result is a connectionist frame system with greater expressive and inferential power than previous systems.

Our concern in this paper is with supporting the types of inference that typically occur in frame systems. These include inheritance of properties along type hierarchies; classification of objects within type hierarchies (or *subsumption*[Brachman, 1983]); instantiating frame definitions for particular individuals; and recognizing instances of frames within complex scenes or descriptions. An important variant of this last process is *abductive inference* [Charniak and McDermott, 1987], which involves partial or near-miss recognition of frames. Abductive inference is used to generate possible explanations for observed phenomena in medical diagnosis, language understanding, visual scene interpretation, and other analysis tasks.

In conventional frame systems, the inference processes are built atop a layer of machinery that contains a few simple ingredients. The most important ingredient is variable binding – i.e., the ability to dynamically establish connections between the objects being reasoned about and the frames in the knowledge base. Another ingredient is frame allocation: the ability to dynamically conjure up new frame instances on demand. Much of the difficulty of building symbolic reasoning into connectionist

systems arises from the difficulty of implementing these basic underlying mechanisms. The problem is simple: connectionist systems tend to be hard-wired, at least over short timescales. This makes dynamic creation of nodes and links problematic.

The application of frame systems to model-based vision brings forth another set of problems – namely the representation of real-valued numerical parameters that are necessary to describe an instance of a model, and verification of the consistency between the parameters of an object and those of its parts. These checks typically involve coordinate-invariant computations. Doing coordinate transformations and coordinate-invariant recognition has been a difficult problem for connectionist systems (see [Hinton, 1981, Ballard, 1986] for discussion and solutions). Numerical parameters (which can be regarded as real-valued "slots") give rise to additional issues in the design of variable-binding machinery, including the need for ways to compute numeric slot values of frames, and notions of near-miss matching based on numeric differences.

Our approach to modeling frame-style inference involves a translation from a frame notation into real-valued equations whose solutions correspond to extensions of an initial set of axioms by sound or plausible inferences. The intuition underlying this translation scheme is that unification, the backbone of inference, can be viewed as a kind of graph matching on graphs containing variables. Graph matching can in turn be viewed as the minimization of an objective function which reflects the degree of mismatch between the two graphs. One advantage of this translation is that the distance metric representing the structural similarity between the graphs can be easily combined with other distance metrics which express the goodness-of-fit between the data and parametric models associated with specific classes of frames. In fact, there can be an entire database of such model-specific distance metrics.

Our objective functions can be viewed as specifications for connectionist networks. The process of generating a network from such a specification is analogous to compiling, and can be formalized as the application of transformations to the specification. There are usually a number of ways of transforming an objective function into a network [Mjolsness and Garrett, 1989], and the different possible networks may have different efficiency properties, as measured in their use of nodes, connections and time. In this paper we focus on the translation from frames to objective functions, leaving the details of the translation from objective functions to networks for a later paper.

The system described in this paper is an extension of the Frameville system of Mjolsness, Gindi, and Anandan [Mjolsness *et al.*, 1988, Mjolsness *et al.*, 1989]. Our extension involves the representation of equality constraints between slots of a frame or those of its parts, yielding a degree of expressivity comparable to simple symbolic frame systems. This paper focuses solely on the theoretical part of our work – how to represent dynamically varying graph structures (Section 2), the description of the variable binding machinery (Section 3), our method for expressing soundness of inference using numerical constraints (Section 4), and our approach to controlling inference (Section 5). We do not describe any simulations or experimental results here. Preliminary experiments involving simple visual recognition and grouping problems are reported in [Mjolsness *et al.*, 1989]. New experiments involving the current extensions are also under way.

2 DYNAMIC GRAPH STRUCTURES

A key problem in doing symbolic inference in connectionist networks is providing mechanisms to dynamically create concepts and relations between them. In Frameville we divide the world into two parts: a static base of quantified knowledge, called *the model side*, and a dynamic set of ground formulae describing the objects of reasoning, called *the data side*. In an interpretation task, the data side would hold the observations and interpretations, while the model side would hold background knowledge. The model side does not change under our inference processes: dynamic allocation of frames and links occurs on the data side, and in the bindings between the data side and the model side. There are three types of dynamic objects: frame instances, which represent objects in the world, `inst`-links, which connect frame instances on the data side to frame types (or *models*, to borrow a term from model-based vision) on the model side, and `ina`-links, which represent slot-filling relationships between frame instances. In addition, there are three kinds of static links on the model side,

called **INA**, **ISA**, and **EQU**. These will be described later.

Each link-type is represented in our system by an array of numbers. For example **inst** links are represented by an $M \times D$ array, where M is the number of models on the model side, and D is the maximum number of frame instances that can be stored in the system at one time. The element α, j of the **inst** array represents an **inst**-link between the model α and frame instance j (denoted $\mathbf{inst}_{\alpha,j}$). If this array element is 1, then there is an **inst**-link between the two; if 0 there is no link. During the optimization process elements can take on real values in the range [0,1]. Numerical constraints in the objective function force these variables to settle on boolean values (section 3), so that when the energy reaches a minimum the state of these arrays describes a graph structure. Similar arrays exist for each of the dynamic link types[1].

If all the potential links leading to a frame instance are zero, then that frame instance effectively does not exist: it is not connected to the graph structure described by the arrays. Conversely, dynamic allocation of frame instances can be achieved by adding a link to a previously unallocated frame instance j. Hence no additional machinery is needed to represent dynamic frame allocation: the dynamic link arrays already imply the power to create new frame instances. The structure of the knowledge base dictates what frames we *could* create – specifically, slot fillers for known frame instances, and new instances whose slots may be filled by known instances. Whether such creation occurs is governed by the inferential control rules described in section 5.

3 FRAME INSTANTIATION

Consider the following frame-style definition of the concept *revenge*, which might occur in a story understander's knowledge base: [Birnbaum, 1986]

define **revenge**
  slots
    gte-1,gte-2: goal-thwarting-event
    aggressor,avenger: actor
  constraints
    thwarter(gte-1) = aggressor
    victim(gte-1) = avenger
    thwarter(gte-2) = avenger
    victim(gte-2) = aggressor

This definition states that an instance of **revenge** consists of two events of the type **goal-thwarting-event**, and two actors. A **goal-thwarting-event** is a kind of event (presumably defined in another frame definition) where one actor, called the **thwarter**, prevents the realization of a goal held by another actor, called the **victim**. In a **revenge** event, the thwarter of the first goal-thwarting event is the victim of the second, and vice versa. In the above notation, **slotA,slotB:type** means that the fillers of the **slots** must be of the type **type**.

The above definition does several things: it establishes the slots of a frame, it places restrictions on the types of the fillers, and it requires that certain equality constraints hold among the slots, or among the slots of slots. In this paper we will not address slots with multiple fillers (but see [Mjolsness *et al.*, 1989]) or set inclusion relationships between slots, such as *the recipient is a member of the donor's family*. Thus the frame language considered here is not as expressive as possible.

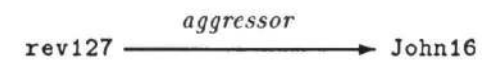

Figure 1: The Usual Graphic View of Role Filling

Applying the definition of **revenge** to an instance of it involves dynamically creating a correspondence between the instance and the **revenge** model. This is the variable-binding problem. Focusing on a single slot of a frame – say, the **aggressor** slot – we will show how to use the machinery of the previous section to bind it to a value. Suppose we want to represent the assertion that **John16** is the aggressor of a particular revenge instance, called **rev127**. The traditional approach would have us create a link of type **aggressor** going from the **rev127** node to the **John16** node (Figure 1). Such a scheme requires

[1] Arrays representing the static link types are compiled into the objective function; they are not variables as far as the optimization process is concerned.

arbitrarily many link types, whereas our machinery for dynamic graph structures requires a fixed, and preferably small number of dynamic link types, since each link-type gives rise to an array in the objective function. One solution to this problem is represent all slot-filling relationships using a single

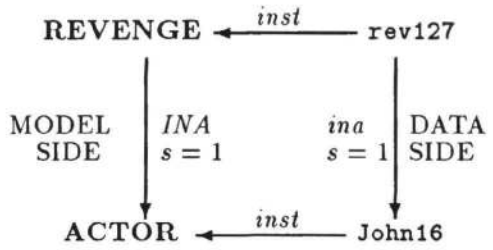

Figure 2: Rectangle Relationship Between `ina` and `inst`.

3-dimensional array called `ina`. The first two dimensions range over the set of dynamically allocatable frame instances, while the third ranges from 1 to S, the maximum number of slots in any frame. Typically S will be a fairly small number, probably between 5 and 10. The slots in each frame are assigned integers in the range [1,S]. $\mathtt{ina}_{i,j,s} = 1$), means that the s'th slot of the i'th frame instance is filled by the j'th frame instance.

In addition to `ina` links on the data side, and `inst` links between the data side and the model side, we have static `INA`, `ISA`, and `EQU` links on the model side. `INA` links express slot filler type restrictions, such as the fact that the `initiator` of a `revenge` must be an `actor`. $\mathtt{INA}_{\alpha,\beta,s} = 1$ means that any object that fills the s'th slot in an object of the model α must be an instance of the model β. `ISA` links on the model side encode class-subclass specializations and allow property inheritance and type subsumption. The `EQU` links will be discussed in the next section. As noted earlier, we also allow real-valued slots (or "analog neurons") $\mathtt{F}_{i,s}, \in [1, S]$ to be associated with each frame instance i. These are described in sections 4 and 5.

We can divide the various constraints incorporated into our objective function into those necessary to ensure the soundness of the inferences made in the network, and those needed for forward chaining and abductive reasoning. As explained in Section 5 such a separation is useful to control the proliferation of possibly correct but irrelevant inferences.

4 SOUNDNESS CONSTRAINTS

Soundness constraints are constraints that force the network to settle on states that describe meaningful frame structures. They are represented in the objective function by numerical equality constraints involving the dynamic variables. Although these constraints may be violated during the optimization process, they must be satisfied when the network reaches a fixedpoint. There is a variety of optimization techniques that can handle such "hard" constraints, some of which have been used in the context of neural networks [Mjolsness *et al.*, 1988].

An important "syntactic" constraint is that there be at most one object (i.e., frame instance) which fills a given slot of any other frame instance. That is, for any given i and s, at most one $\mathtt{ina}_{i,j,s} = 1$. This can be expressed as: [2]

$$\forall\, i, s \quad (1 - \sum_j \mathtt{ina}_{i,j,s}) \sum_j \mathtt{ina}_{i,j,s} = 0 \qquad (1)$$

The meaning of `INA` described in the previous section can be expressed as

$$\forall\, \alpha, \beta, i, j, s \ \text{ s.t. } \mathtt{INA}_{\alpha,\beta,s}$$
$$\mathtt{inst}_{\alpha,i}\ \mathtt{ina}_{i,j,s}\ (1 - \mathtt{inst}_{\beta,j}) = 0 \qquad (2)$$

This says that if there is an `INA`-link between α and β for slot s, then whenever any i is an instance of α, and the s'th slot of i is j, then j must be an instance of β. The combination of the $(1 - \mathtt{inst}_{\beta,j})$ term on the left hand side, equated to 0 on the right, is an idiom that means the term $\mathtt{inst}_{\beta,j}$ must be 1.

Usually, definitions of frames will also contain equality constraints. For instance, we may require that the same object should fill two different slots of the same frame. Since slot-fillers themselves are frame instances, such equality relations may be nested. For example, the definition of **revenge** given in Section 3 requires that `thwarter(gte-1)`

[2] Each constraint consists of a generative portion and the actual constraint. The generative portion is universally quantified over a set of variables and may contain restrictions on them expressed in terms of model-side links (where for conciseness, we have used $\mathtt{x}$ and $\neg\mathtt{x}$ to represent $\mathtt{x} = 1$ and $\mathtt{x} = 0$ respectively). The constraint may be a hard-constraint of the form $h(\textit{dynamic-links}) = 0$, or a term (soft-constraint) of the form $f(\textit{dynamic-links})$ that is included in the objective function to be minimized.

= **aggressor**. Equality constraints can be expressed as predicate calculus assertion by treating slots as functions of their frames[Charniak, 1988]. They have the following general form:

$$\forall\, i \in \alpha \quad t(s(i)) = u(i)$$

This represents the assertion that the same object (or frame instance) that fills slot t of slot s of any instance i of model α must fill slot u of i as well. We can denote this in terms of an EQU matrix as $\mathtt{EQU}_{\alpha,s,t,u} = 1$. Equalities are then expressed by constraints of the form:

$$\begin{aligned} &\forall\, i,j,k,\alpha,\beta,\gamma,s,t,u \ \text{ s.t.} \\ &\quad \mathtt{INA}_{\alpha,\beta,s} \wedge \mathtt{INA}_{\beta,\gamma,t} \wedge \mathtt{INA}_{\alpha,\gamma,u} \wedge \mathtt{EQU}_{\alpha,s,t,u} \\ &\mathtt{inst}_{\alpha,i}\ \mathtt{ina}_{i,j,s}\ (\mathtt{ina}_{j,k,t} - \mathtt{ina}_{i,k,u}) = 0 \end{aligned} \quad (3)$$

Equality constraints between two sibling slots of a frame instance can also be expressed. Equality constraints between slots that are deeply nested in compositional hierarchies can be transformed into a set of equality constraints none of which involve nesting of depth greater than 2, as in Equation 3 This transformation involves introducing additional "dummy" slots for each of the intervening frames and "copying" the slot of a child frame instance into its parent frame instance. The copy mechanism is itself expressible as an equality constraint.

As noted in Section 2, ISA links allow frames to be organized into a specialization hierarchy. Thus $\mathtt{ISA}_{\alpha,\beta} = 1$ means that model β is a specialization of α. A model is allowed to be a specialization of multiple "higher-level" models, so the specialization hierarchy forms a directed acyclic graph. If frame instance i is an instantiation of model α, then it must be an instantiation of exactly one of the specializations of α, unless α is a leaf node in the specialization hierarchy. This is expressed as:

$$\begin{aligned} &\forall\, i,\alpha \ \text{ s.t. } \ \alpha \text{ is not a leaf} \\ &\mathtt{inst}_{\alpha,i} - \sum_{\beta \text{ s.t. } \mathtt{ISA}_{\alpha,\beta}} \mathtt{inst}_{\beta,i} = 0 \end{aligned} \quad (4)$$

This rule implements both inheritance up the type hierarchy, because in inst link to model β tends to turn on inst links to the ISA-parents of β; and subsumption, or discrimination down the type hierarchy, because an inst link to α tends to turn on an inst link to one of α's ISA-children. The constraints associated with the children will rule out inconsistent specializations. More specialized frames must use the same slot-numbering conventions as their parents. In this paper, we do not address the issue of exceptions [Derthick, 1988].

Finally, the requirement that inst and ina links are boolean-valued can be expressed as:

$$\begin{aligned} \forall\, i,\alpha \quad & \mathtt{inst}_{\alpha,i}\,(1 - \mathtt{inst}_{\alpha,i}) = 0 \\ \forall\, i,j,s \quad & \mathtt{ina}_{i,j,s}\,(1 - \mathtt{ina}_{i,j,s}) = 0 \end{aligned} \quad (5)$$

5 INFERENTIAL CONTROL

We suppose that a reasoning problem is posed to a Frameville network by establishing an initial data-side graph structure; array elements describing this graph are "clamped" to a value of 1, so the network must settle into a state which is minimal subject to the restriction that the input is a subgraph of the final graph. The constraints described in the previous section rule out certain types of meaningless network states, but they by no means completely determine the behavior of a network. For instance, if the data only constrain inst and F variables, a consistent solution is to set all ina variables to zero; many variations of this trivial solution are possible. Furthermore, our frame language is rich enough to allow a variety of inferences, which, if applied willy-nilly, will rapidly use up the supply of dynamically allocatable frame instances and links, without necessarily drawing any interesting conclusions. Two particularly "dangerous" types of inference may be termed *recognition* and *slot-filling*.

By *slot-filling* we mean the allocation of new frame instances to fill unfilled slots of existing frames, or using existing instances if they are consistent with the constraints on the slot. For example, if the data says there is a **revenge**, we can create instances for the **initiator**, the **avenger**, the **gte-1** and the **gte-2**, and establish the appropriate relationships between them. This is a legitimate inferential step, since if the **revenge** exists, the slot fillers must exist. Alternatively, we could put an existing instance of the appropriate type – **John17**, say – into the **aggressor** slot. This would be a plausible but not necessary inference. When new instances are created to fill slots, the slot fillers may require their own slots to be filled, leading to an explosion of allocation until the capacity of the network is exhausted.

In Frameville, slot-filling is achieved by minimizing "penalty terms" in the objective function (as opposed to the use of "hard" constraints, which

must be satisfied). Corresponding to every slot of every frame-definition, we have an additive term in the objective function of the form:

$$\forall\, \alpha, \beta, i, s \quad \text{s.t.} \quad \mathbf{INA}_{\alpha,\beta,s} \\ (\mathbf{inst}_{\alpha,i} - \sum_j \mathbf{ina}_{i,j,s}\ \mathbf{inst}_{\beta,j})^2 \qquad (6)$$

The lowest energy state of the network, which is zero, will be achieved if and only if all the slots of each instance of a frame are properly filled. By themselves, these terms will tend to produce the explosion mentioned above. To counteract this tendency, we add an additional term, called a *parsimony* term, which penalizes the network for the creation of new frames:

$$\sum_i \sum_{\alpha \text{ is a leaf}}^{\alpha \text{ s.t.}} \mathbf{inst}_{\alpha,i} \qquad (7)$$

One consequence of this parsimony term is that the network will prefer using existing frame instances to fill unfilled slots over creating new ones.

Recognition means creating an instance of a frame when we observe a set of frame instances that satisfy the constraints on the frames definition. For example, if we see two goal thwarting events where the thwarters and victims are reversed, we can create a revenge. Pure recognition is not dangerous: it inevitably terminates and is not typically explosive. However, partial recognition, in which the frame definition is partially but not completely satsfied, is a useful variant of pure recognition in a world where the input data is incomplete, and partial recognition tends to be explosive and potentially nonterminating.

Recognition is also achieved by minimization. The terms corresponding to recognition are of the form

$$\forall \alpha, j, \beta \ \text{s.t.} \ \mathbf{INA}_{\alpha,\beta,s} \\ (\mathbf{inst}_{\beta,j} - \sum_i \mathbf{ina}_{i,j,s}\ \mathbf{inst}_{\alpha,i})^2 \qquad (8)$$

This rule has the following interpretation. If an instance j of model β is appropriate to fill slot s of frames of type α , then an instance i of α may be created and the appropriate slot-filler binding established. This kind of mechanism is often used to propose hypotheses in abductive inference [Charniak, 1988]. This rule also tends to penalize the occurence of multiple instances of the same frame type having identical fillers for a given slot. In frame systems it is usually desirable to prevent the occurrence of distinct frames having identical fillers in *all* slots. A direct expression of this constraint gives rise to a very high order ($O(S)$) energy function, which is expensive to implement in a network. The recognition term above is a limited attempt to achieve a similar effect.

6 REAL-VALUED PARAMETERS

In addition to the machinery for requiring structural correspondences between data and models, Frameville allows numeric slots constrained by *model-specific* objective function terms whose algebraic form will depend on the models involved. These terms may be idiosyncratic functions of both numerical and frame-valued slots, but for ease of exposition, we restrict our attention to numerical slots here:

$$\forall\, \alpha, \beta, i, j, s \quad \text{s.t.} \quad \mathbf{INA}_{\alpha,\beta,s} \\ \mathbf{inst}_{\alpha,i}\ \mathbf{ina}_{i,j,s}\ \mathbf{inst}_{\beta,j}\ H^{\alpha,s}(\vec{\mathbf{F}}_i, \vec{\mathbf{F}}_j) \qquad (9)$$

where $\vec{\mathbf{F}}_i = (\mathbf{F}_{i1}, \mathbf{F}_{i2}, \ldots)$ represents the vector of numerical parameters of i. This constraint relates the numerical parameters of two frame instances, one a slot-filler of the other. Similar model-specific terms can be used to express relationships between numerical parameters of the fillers of sibling slots of a frame instance. The $H^{\alpha,s}$ functions given above are specific to a model α and may express coordinate system invariant relationships between the parameters of a frame instance and those of its fillers. By doing so, we eliminate the need for explicitly storing the transformation matrices between the coordinate system of an object and those of its parts. If, however, it is useful to explicitly have such a transformation matrix, it can also be represented using numerical slots.

The propagation of constraints among numeric slot values can be expressed either by hard constraints (i.e., by requiring that the expression in Equation 9 be zero) or as penalty terms in the objective function. In the latter case, they affect inferential control in a model-specific way. Our approach also has the advantage that it is not necessary to separate the computation of the numerical parameters of a high-level object from the recognition of the object itself. The optimization process simultaneously determines the object identity and best choice of object parameters to fit the data. When higher-level information is available,

top-down propagation of that information is also achieved by the same optimization process. Model-specific constants may also be incorporated into frame-valued slots, thereby biasing the likelihood that particular slots will be filled or that particular types will trigger recognition. Such constants may provide a basis for implementing certainty or probability-like mechanisms.

Each type of penalty term described in this section gets added into the overall objective function, but with possibly different multiplicative coefficients. It is the relative values of these coefficients that determines the competition between parsimony on the one hand and forward-inferencing and abductive "guessing" on the other. Appropriate values of these coefficients may be determined experimentally, or they may be set dynamically by external mechanisms that control the Frameville network.

7 RELATIONSHIP TO OTHER CONNECTIONIST FRAME SYSTEMS

Our approach resembles Derthick's μKLONE system [Derthick, 1988] in several ways, notably in our general method for translating logical assertions into objective functions and numerical constraints, and in our use of a static model-base that is used by a compiler whose output is a network designed for optimization. (Note that this separation of knowledge into "data" and "models" is also present in Shastri's approach [Shastri, 1987].) However, we have introduced a mechanism for variable-binding which greatly increases the expressive and inferential power of the system, and that deals with real-valued parameters and with constraints involving such parameters. In particular, our variable binding mechanism, in combination with the separation of frame types from frame-instances, allows us to have multiple instances of a frame, and to dynamically create new instances and relationships among them. We share with Dolan and Dyer [Dolan and Dyer, 1988] the advantage of being able to perform chains of inferences in parallel, although in both our cases, considerable experimental work needs to be done before a complete evaluation can be made. Our approach differs from that of Dolan and Dyer (as well from that of Touretzky, *et al.* [Touretzky and Geva, 1987]) in some fundamental ways: first our use of objective functions as a specification language allows us to specify the desired properties of the network – namely its fixedpoints – and allows us to perform algebraic fixedpoint-preserving transformations that can lead to efficient networks. Second, the same methodology also allows the modular and incremental design of the system (prior to any algebraic transformations). One more distinction: while in many of these existing systems (including μKLONE) microfeature-based coarse coding mechanisms are used to represent the similarity of concepts, our use of a data-base of model-specific distance metrics $H^{\alpha,s}$, indexed by the `ISA` links, points to a fundamentally new approach to the design of connectionist frame systems.

8 CONCLUSIONS

We have presented a connectionist frame system called Frameville, which can represent dynamically varying graph structures and thereby can inherit some of the representational and reasoning power of symbolic frame systems. In particular, we have described machinery to dynamically instantiate frames and perform variable-binding. All of this has been done within the paradigm of objective function minimization, which is used both as a programming language and a neural-network specification language. Our approach also extends the traditional use of distance metrics by allowing us to systematically integrate constraints involving pointers and real-valued variables into a single objective function, and by allowing us to utilize a data-base of model-specific distance metrics. This work is an extension of previous work described in [Mjolsness *et al.*, 1989], where preliminary experiments are also reported. The present paper has laid out the theoretical ground work necessary to perform new experiments, and we expect that further refinements of our theory will be shaped by these experiments.

ACKNOWLEDGEMENTS

We would like to thank Gene Gindi for his continued collaboration in Frameville research. We would also like to thank Chris Riesbeck for discussions about our work and Denys Duchier for commenting on an earlier draft of this paper. This work was supported in part by AFOSR grant AFOSR-88-0240 and by DARPA grant DAAA15-87-K-00001.

References

[Ballard, 1986] Dana Ballard. Cortical connections and parallel processing: Structure and function. *Behavioral and Brain Sciences*, vol 9:67–120, 1986.

[Birnbaum, 1986] Lawrence Birnbaum. Integrated processing in planning and understanding. Technical Report YALEU/CSD/RR-489, Yale University, 1986. PhD Dissertation.

[Brachman, 1983] Ronald J. Brachman. What **is-a** is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer, Special Issue on Knowledge Representation*, pages 30–36, October 1983.

[Charniak and McDermott, 1987] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, 1987.

[Charniak, 1988] Eugene Charniak. Motivation analysis, abductive unification, and nonmonotonic equality. *Artificial Intelligence*, 34(3):275–296, 1988.

[Derthick, 1988] Mark Derthick. Mundane reasoning and parallel constraint satisfaction. Technical Report CMU-CS-88-182, Carnegie-Mellon University, September 1988. PhD Dissertation.

[Dolan and Dyer, 1988] Charles P. Dolan and Michael G. Dyer. Parallel retrieval and application of conceptual knowledge. In D. Touretzky, G. Hinton, and T. J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann, 1988.

[Hinton, 1981] Geoffrey E. Hinton. Shape representation in parallel systems. In *Proceedings of 7th IJCAI.* IJCAI, 1981.

[Mjolsness and Garrett, 1989] Eric Mjolsness and Charles Garrett. Algebraic transformations of objective functions. Technical Report YALEU/DCS/RR-686, Yale University, March 1989.

[Mjolsness *et al.*, 1988] Eric Mjolsness, Gene Gindi, and P. Anandan. Optimization in model matching and perceptual organization: A first look. Technical Report YALEU/DCS/RR-634, Yale University, June 1988.

[Mjolsness *et al.*, 1989] Eric Mjolsness, Gene Gindi, and P. Anandan. Optimization in model matching and perceptual organization. *Neural Computation*, 1989. to appear.

[Shastri, 1987] Lokendra Shastri. *Semantic Networks: An Evidential Formulation and its Connectionist Realization.* Morgan Kaufmann, 1987.

[Touretzky and Geva, 1987] David S. Touretzky and Shai Geva. A distributed connectionist representation for concept structures. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 155–164. Lawrence Earlbaum Associates, 1987.