RSD-TR-5-87

A GEOMETRIC APPROACH TO COUNTERFACTUAL REASONING AND ITS APPLICATION TO MODELING MANUFACTURING SYSTEMS

by

Arch W. Naylor

Mohammad B. Shadmehr

The Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-1109

March 1987

CENTER FOR RESEARCH ON INTEGRATED MANUFACTURING

Robot Systems Division
College of Engineering
The University of Michigan
Ann Arbor, Michigan 48109-1109

TABLE OF CONTENTS

A Geometric Approach to Counterfactual Reasoning and Its Application to Modeling Manufacturing Systems

Arch W. Naylor
University of Michigan

Mohammad B. Shadmehr
University of Michigan

February 28, 1987

Abstract

A geometric approach to logic-based formal models of manufacturing systems [1] is introduced and is used to solve the updating problem that arises in these logic-based models. The problem arises because it is relatively easy to formulate a model in which the next state is ambiguous. Conditions are presented which identify such undesirable models. The approach is first to show that the updating problem is very similar to the world selection problem in counterfactual reasoning and then to use functional analytic methods to develop conditions which can be viewed as geometric tests. Furthermore, in models which pass these tests, updating becomes an orthogonal projection. Although the application is to manufacturing systems, the results presented are applicable to many other kinds of systems. For example, they are applicable to situation-based planning problems and the frame problem in Artificial Intelligence, discrete-event systems, and rule-based systems. Furthermore, the approach used here may be of some use in the philosophical problem of counterfactuals.

Keywords: Manufacturing, Logic, Systems, Models, Discrete-Events, Counterfactual, Rule-Based, Expert, Planning, Situations, Frames

1 Introduction

If one is concerned, as we are, with software for the real-time control of the flow of parts, tools, material, and information on the factory floor, then one must also be concerned, in one way or another, with modeling what we will call the "logical view" of a manufacturing system. That is, one needs to keep track of logical conditions such as "part no. 28 is on machine 3", "vehicle 7 is at the loading dock", and "part no. 28 has finished the fourth step in its process plan". One also needs to follow changes in these conditions and model how long it takes to make these changes. For example, one should be able to model moving vehicle 7 from the loading dock to machine 3. In particular, one should merely have to say that is takes about 18 seconds to make the move, ignoring lower level details. The importance of such logical-view models is that they give precisely the needed perspective for the kind of real-time control of interest here: they present just enough detail, and no more, to describe what is essentially a real-time scheduling problem. Needless to say, other

kinds of models—for instance, robot kinematic models—are also important for other kinds of control, but from the logical viewpoint one merely needs to know that it takes the robot 10 seconds to move a part from the machine to the vehicle, not the equations of the path followed.

One approach to modeling the logical view of integrated manufacturing systems, the one we concentrate on in this article, is given in [1], where, among other things, a number of examples of its application to manufacturing systems are presented. Although it is shown there that the modeling approach is expressive enough to capture the real-time control-software issues, it is also pointed out that there is a fundamental problem—the updating problem—with the modeling system. In this article we present a complete analysis and solution of the problem. These results are important for at least two reasons.

First, the logic-based models of [1] are the "natural" way to describe manufacturing systems for purposes of real-time control. By "natural" we do not mean that the first-order logic notation used in [1] is required, that is just a very convenient standardized language. Other modes of description—for example, graphical—are clearly possible, but each will merely be another way of describing essentially the same perspective of the system being modeled, that is, the logical view; and each must deal with the updating problem. In other words, updating is a fundamental problem for modeling the logical view of a manufacturing system; and, therefore, its solution is important for real-time control software.

Second, a side effect of the solution to the updating problem presented here is to show that the geometric insights of functional analysis and linear spaces can be used to describe the dynamics of manufacturing systems and their control. In particular, planes, orthogonal projections, and linear transformations are applicable. This promises a new approach to the analysis of these systems, and subsequent articles will exploit this approach.

Next let us informally introduce the updating problem. Imagine a snapshot of a factory floor at some instant. It shows the locations of parts, tools, and material as well as other things. Further, imagine giving a command that will result in a new configuration of the factory floor. If the command is something like: "Load the part on the material transport system onto machine no. 9," it is fairly clear what is intended. However, if the command is "Load the part on the material transport system onto machine no. 9 or take it to the loading dock," what is intended is not clear. Presumably, there are two possibilities. One might argue that this simply reveals a nondeterministic system. Perhaps, but suppose that we are attempting to design or model a system that is supposed to be deterministic. How do we recognize and avoid this kind of ambiguity? Doing so, turns out to be relatively easy, but not quite so easy as our simple example might suggest. For instance, suppose that the command is "Unload all machines that can be unloaded." Is it clear what the next factory floor configuration will be? Not necessarily. There may be various ways to unload as many machines as possible. In other words, there are cases that are hard to call. Moreover, in a formal model the commands are probably not in English, they are sentences in a first-order language or something equivalent to that, and which ones lead to ambiguity is not immediately obvious.

Our problem, then, is to avoid this ambiguity. This problem does not, as we have said, arise because we have chosen to model manufacturing systems using first-order logic. The problem would be present even if we chose a different modeling method, and the advantage of our logic-based approach is that it allows us to formulate and treat the problem precisely.

The rest of the paper is organized as follows : First, we say just enough about counter-factual reasoning to present the spirit of this area. Then we review the logic-based modeling formalism developed in [1]. Next we mention how our work relates to counterfactuals. After

that we present a concept of relative closeness. Everything in this article hinges on this concept of closeness. In particular, updating comes down to picking, say, the next factory floor configuration to be the one (from among all the possibilities) that is the closest, and recognizing that sometimes there is no closest one. The next section shows that a functional analytic geometry follows naturally from our concept of closeness, and the remainder of the article exploits this geometry to get practical tests for unambiguous updating. Finally there is a section which describes, for the reader interested in situation-based planning or rule-based systems, how our work relates to these areas.

2 Overview of Counterfactual Reasoning

Counterfactual reasoning is concerned with explaining what is meant when someone says, for example, "If kangaroos had no tails, they would topple over." Such statements are counterfactual in the sense that the antecedent, in this case, *kangaroos have no tails*, is false [1] in the world that we live in. Such statements ask us to imagine a different world where kangaroos do not have tails and to contemplate the consequences of tailless kangaroos. If kangaroos would indeed topple over in such a different world, then we accept the statement "If kangaroos had no tails, they would topple over" as true in our world. Of course, specification of what the different world is becomes a key problem, and the usual approach is to imagine making as few changes as possible in the real world. That is, one imagines a world with tailless kangaroos that is "near" the real world. In fact, one imagines the set of all worlds with tailless kangaroos and picks the world in that set which is "nearest" to the real world. Not too surprisingly, this general approach has raised considerable controversy, as anyone familiar with metrics, topology, and minimization will appreciate. Simply stated, it is just not clear that there will always be a unique, nearest world to the real world. It is not even clear what we should mean by "near". The collection [2] is an excellent overview of counterfactuals. It contains reprints of classic seminal papers in the field together with relatively recent developments, and it presents the various approaches to the "nearness" problem.

We are interested in counterfactuals because they arise naturally in the logic-based models we employ. Consider the command "Unload the part from machine no. 4 and put it on the material transport system." After this command is executed the following statement will be true:"The part is not on machine no. 4 and the part is on the material transport system." Presumably this statement was previously false. Thus we went from a world (i.e., manufacturing system configuration) where the statement was false to a world where the statement is true. The problem is to determine what else has to change. That is, what counterfactuals of the form "If the part were not on machine no. 4 and the part were on the material transport system, then ..." are true. The consequents of these counterfactuals would characterize the additional changes that have to be made. The goal of this article is to say what these should be.

However, it is important to appreciate that we are saying what the changes "should" be, not what they "must" be. In other words, one may undertake any well defined scheme for updating. Presumably one could arrange things in our manufacturing system so that just about anything could happen, and we could somewhat arbitrarily require our models to follow suit. Instead, what we are doing here is to develop a simple, orderly, reasonable,

[1] The antecedent does not have to be false; however, true antecedents just are not interesting when one is studying counterfactuals.

even "logical" interpretation of how these logic-based models should work. So our updating scheme is not the only one possible, but rather one that works and makes sense. This parallels what those working on counterfactuals do: They try to formulate reasonable and consistent interpretations for counterfactuals and ignore peculiar, strained ones.

Stalnaker [2, page 87] argues that some counterfactual statements are neither true nor false. The classic example that he uses is contained in the following two statements: *If Bizet and Verdi had been compatriots, Bizet would have been Italian* and *If Bizet and Verdi had been compatriots, Verdi would have been French.* He argues that each of these counterfactual statements is neither true nor false. Basically, his argument is that there is no convincing way to choose between a world in which Bizet and Verdi are both Italians and one in which they are both French. We will use this point of view to argue that certain models are inappropriate because they raise this kind of ambiguity.

3 Review of the Modeling Formalism

We review just enough of [1] to allow the formulation of the needed framework. Intuitively, the models are made up of entities that are things such as parts, tools, machines, vehicles, and so forth. A part being on a machine would typically be modeled by putting these two entities—part and machine—in contact; and if the part were unloaded from the machine, this contact relationship would be broken in the model. Most of the formal details of the modeling approach follow in a fairly straightforward manner.

The state of the model at any given time is characterized by a structure [3, page 79] for a first-order language. This language always has a one-place predicate symbol AE and two two-place predicate symbols CT and ST. In addition, it has a number of other, usually one-place, predicate symbols $V_\alpha, V_\beta, \ldots$, and it has constant symbols. Variable symbols are near the end of the alphabet: x, y, z, and constant symbols are in the alphabetic neighborhood of k.

A structure characterizing the state is referred to here as a **configuration** and, as usual for such a structure, it has

- A universe set, call it, E_u
- A subset of E_u associated with predicate symbol AE. We denote this set also by AE and rely on context to clarify whether the predicate symbol AE or the set AE is meant. We will treat all predicate symbols and their associated relations in the same manner.
- Subsets CT and ST of $E_u \times E_u$ associated with the predicate symbols CT and ST, respectively.
- Subsets $V_\alpha, V_\beta, \ldots$ of the appropriate cartesian product of E_u with itself associated with the predicate symbols $V_\alpha, V_\beta, \ldots$.
- Assignments of each constant symbol to an element of E_u.

The elements of E_u are the **entities**, and, as we have said, they are usually things such as parts, assemblies, tools, machines, vehicles and messages. That is, they usually have a direct and obvious connection to the real world. E_u is meant to be the set of all entities that will ever be of interest in the model. Those entities that are currently active are in the set AE. For example, a part may become inactive by leaving the factory floor. Usually active entities are the ones that are known explicitly, and the set of inactive ones, that is,

$E_u - AE$, is not described completely. [2] The sets $V_\alpha, V_\beta, \ldots$ are **value** sets. For example $x \in V_{Parti}$ or, equivalently, $V_{Parti}(x)$ designates that the entity x is a part of type i. In other words the value sets say what kind of entity x is. The entities that are active may change with time; however, the values of an entity are fixed forever. That is, a part is alway a part. Similarly, the element of E_u assigned to a particular constant symbol never changes. That is if an entity has a name, identifier, or serial number it never changes.

$CT \subset E_u \times E_u$ is the **contact relation**, and, again, it is used to model such things as the part being on the machine, that is, "the part and the machine in contact." The following sentences [3] are true at all times:

$$(\forall x)(\neg CT(x,x))$$

that is, CT is irreflexive, [4]

$$(\forall x)(\forall y)(CT(x,y) \rightarrow CT(y,x))$$

that is, CT is symmetric,

$$(\forall y)(\forall x)(CT(x,y) \rightarrow AE(x) \wedge AE(y))$$

that is, an inactive entity is never in contact with another entity.

The foregoing sentences as well as others that are always true are collected into a set which we will designate by Θ.

$ST \subset E_u \times E_u$ is the **substructure relation**, and it is used to model such things as the part being a substructure of the assembly. The following well-formed formulas are true at all times, that is, in Θ:

$$(\forall x)ST(x,x)$$

that is, ST is reflexive,

$$(\forall x, w_1, \ldots, w_s, z_1, \ldots, z_t, y)(ST(x,w_1) \wedge \ldots \wedge ST(w_s,y) \wedge ST(y,z_1) \wedge \ldots \wedge ST(z_t,x) \rightarrow x = y)$$

for s=0,1,...and t=0,1,.... This set of well-formed formulas means that the transitive closure of ST is antisymmetric. In other words, we do not want non-trivial "substructure loops."

Θ may also contain well-formed formulas that are peculiar to a particular system. For example, it might be that at most one part can be on machine m, that is,

$$V_{Part}(x) \wedge V_{Part}(y) \wedge CT(x,m) \wedge CT(y,m) \rightarrow x = y$$

As we will see below, the existence of Θ, significantly complicates the updating problem.

A configuration is, as we have said, a structure of the form just described, and, as we have also said, it is the state of the system. However, we know that some pieces of the structure never change— the values, the assignments of constants, and the universal set E_u— so these can be suppressed in the description of the state, and we can simply keep track of the ordered-triple (AE, CT, ST), where, of course, the interpretation of these symbols as relations is intended. It is possible to make the time at which the system enters

[2] Usually this lack of explicitness regarding $E_u - AE$ is the only reason that the model, i.e., structure, does not technically have a known finite size.

[3] Recall that a sentence is a well-formed formula with no free variables.

[4] Irreflexiveness is not really needed here, but it is convenient for CT to be that or reflexive.

configuration (AE, CT, ST) part of the state, that is, $((AE, CT, ST), \tau)$; however, this will not be needed in this article. [5]

The operation of the system, then, becomes a sequence (AE_k, CT_k, ST_k) of configurations or states. We now describe how the state changes come about.

The basic mechanism is a so-called **control** which is a collection of **input–logical variable pairs**, where the pairs are of the form (q, I) with q an input and I a logical variable. The control is used at time t by calling for one of its inputs. For example, if control C is the set $(q_1, I_1), \ldots, (q_n, I_n)$, then input q_1 is called for at time t by making I_1 true at time t. One can view the control as a set of buttons labeled I_1, through I_n from which one selection can be made. A system can have more than one control.

An **input**, in turn, is a collection of changes, and the purpose of a **change** is to transform some part of the current configuration. For example, $(k_1, k_2) \in CT$ may be transformed to $(k_1, k_2) \notin CT$. An input may be made up of more than one change because several parts of the configuration may have to be transformed, and these transformations may take different amounts of time. For example, moving a vehicle from A to B first separates the vehicle from A and then later puts it in contact with B, that is, the input Move contains two changes.

Eventually inputs and their changes should result in mappings of configurations into configurations, but trying to characterize them *directly* as mappings is impractical. The systems are highly parallel; consequently, many changes may be occurring simultaneously, and one would, in effect, need a mapping for each possible combination of changes. Further, the number of possible configurations is probably enormous. The alternative is to characterize these mappings indirectly by representing changes in a rather obvious way. A change is as follows:

$$L(t) \mapsto R(t_1)$$

where L and R are sets of well-formed formulas. The formulas in L refer to the configuration at the start of the change, and the formulas in R refer to the configuration after the change is finished. If all the formulas in L are satisfied, and the change is called for, then R, which is usually not satisfied by the current configuration, becomes satisfied by the new configuration. The effect of the change is to separate the entities k_1 and k_2. It is important not to confuse the symbol "$\mapsto$" with the first-order logic symbol "$\rightarrow$". That is, $L \mapsto R$ is <u>not</u> a well-formed formula. In other words, the modeling systems uses first-order logic, but it is not contained in it.

If the change starts at time t, then the response R becomes satisfied at time t_1. For example, $t_1 = t + 10$ shows that the response will occur 10 units of time after the change started.

Although changes are usually quite simple, we do have to go into a bit more detail about how changes work. This is required as a foundation for later results. In particular, we are developing a general theory here, and we have to start with a precise understanding of changes. The remainder of this section can be skimmed on first reading specially by a reader familiar with Skolemization in first order logic.

First we assume, with no loss in generality, that the formulas in L and R are in prenex normal form [3, page 150]. For example,

$$L = Q_1 x_1 \cdots Q_n x_n \gamma$$

[5] In fact, it is relatively easy to use entities to encode historical information into the current configuration. Thus, if we wanted to remember when we entered the current configuration, we could include an entity whose value was that time. This is important because it allows the state to contain, as it should, all we need to know about the past.

where the Q_i's are quantifiers (i.e., $\forall$ or $\exists$), the x_i's are variable symbols, and γ is a well-formed formula without any quantifiers. Occurrences of x_i, $1 \leq i \leq n$, in L are, of course, bound. If other variables symbols occur in L, they are free occurrences.

As long as there are no existential quantifiers in L or free variables, things are straightforward. Otherwise, we have to be a bit fussy. In particular, we have to state carefully how information is carried from the left to the right side of a change. First-order logic cannot help us here. We treat existential quantifiers first.

Our approach is to imagine [6] that we eliminate existential quantifiers by Skolemizing [3, page 274] L. We explain what this means through examples. If $L \longmapsto R$ is

$$(\exists x)(CT(x,k)) \longmapsto \neg CT(x,k)$$

we replaced it by

$$CT(k',k) \longmapsto \neg CT(k',k)$$

where k' is a constant symbol that is used nowhere else. The intuitive idea is that if L is satisfied by a configuration, there is at least one entity that can be associated with the quantified variable x. We simply assign an arbitrary one of these entities to the new constant symbol k'. Carefully note that if there is indeed more than one choice, the arbitrary selection introduces an element of randomness. The constant k' replaces x on the right side also, where, by convention, it occurs free.

Usually Skolem constants such as k' above handle all cases; however, if L is

$$(\forall y)(\exists x)(CT(x,y))$$

we replaced it by

$$(\forall y)(CT(F(y),y)$$

where F is a function symbol that is used nowhere else. This new symbol is assigned to a function, and if there is more than one choice, the selection is made arbitrarily, and it replaces x on the right side as well.

Similarly, if L is

$$(\exists u)(\forall v)(\exists w)(\forall x)(\forall y)(\exists z)\gamma(u,v,w,x,y,z)$$

we replaced it by

$$(\forall v)(\forall x)(\forall y)\gamma(k',v,F_w(v),x,y,F_z(v,x,y))$$

where k' is a new constant symbol, F_w and F_z are new function symbols, and γ is a well-formed formula without quantifiers. As before k', F_w, and F_z replace u, w, and z on the right side as well.

Thus, a structure $\mathcal{A}$ satisfies, say, the sentence

$$(\forall v)(\forall x)(\forall y)\gamma(k',v,F_w(v),x,y,F_z(v,x,y))$$

if we can assign k' to some entity in the universe of $\mathcal{A}$ and associate F_w and F_z with functions such that the formula is, then, satisfied in the usual sense by $\mathcal{A}$. We do not consider k', F_w, and F_z part of the structure $\mathcal{A}$. Rather we view them as augmenting $\mathcal{A}$ to yield a structure for the language with the Skolem symbols added. Again, if there is more than one choice

[6] We use the word "imagine" because to use the modeling formalism it is not really necessary to make the substitutions that we will discuss. We are merely using Skolemizing as a vehicle for clarifying the operation of the model.

possible for k', F_w, and/or F_z, one is selected arbitrarily. This corresponds to selecting, say, any part on a pallet when there is at least one.

Note that any constants and functions introduced by Skolemizing refer to the same things in both L and R. Further, variables in L and R refer, by convention, to the same thing.

This finishes the discussion of the existential quantifiers. For the free variables that occur in L or R we have to use a different method, but since free variables rarely occur we omit their discussion here for the sake of brevity. Intuitively, when the free variables are used the change is to be instantiated at various places in the manufacturing system and take place at all these places simultaneously.

4 Updating Problem and Its Relation to Counterfactual Reasoning

The similarity of our updating problem to counterfactual reasoning is clear. Suppose that the currently active configuration is $\mathcal{A}$ and that the left side L of some change is satisfied. Suppose further that the corresponding right side R becomes true at time $t' = t + \Delta$. Further, suppose that a set Θ of sentences is always true. The updating problem is, as we have said, to determine what the next configuration $\mathcal{A}'$ should be. It is a problem because there may be no obvious and well-defined way to do it. The candidates for $\mathcal{A}'$ are in the class P of all configurations that satisfy R and Θ. The configuration $\mathcal{A}$ satisfies Θ but presumably not R. Thus, from a counterfactual reasoning point of view we are assuming that we are only interested in Θ-worlds. The counterfactual statements of interest are of the form "If R were true and Θ remained true, then ... "

The difference from counterfactual reasoning is that our worlds are far simpler and quite explicit. Still counterfactual reasoning raises exactly the same closeness problem, that is, which world in P, if any, is closest to the real world $\mathcal{A}$, and much of the philosophical literature [2] on counterfactual reasoning is concerned with this problem.

5 A Concept of Relative Closeness

Our approach is quite simple. First we require that the configurations all have the same universe, and that constants symbols, value predicate symbols, and Skolem symbols be interpreted in the same way. Then we say that $\mathcal{A}'$ is closer to $\mathcal{A}$ than $\mathcal{A}_s$ is if $\mathcal{A}' \neq \mathcal{A}_s$ and wherever $\mathcal{A}'$ disagrees with $\mathcal{A}$ so does $\mathcal{A}_s$. That is, $\mathcal{A}_s$ disagrees with $\mathcal{A}$ in all the ways that $\mathcal{A}'$ does and in some additional ones. (We formalize this definition in the next section.) Arguably, any concept of closeness must at least have this property. The difficulty comes when one tries to incorporate more. For example, one might compare factory floor configurations by means of metrics, topologies, neighborhood systems, or other more elaborate constructs. However, we would find this rather arbitrary and of questionable meaningfulness.

6 Geometric Analysis of Configuration Space

A geometric interpretation of the updating problem flows almost immediately out of our concept of relative closeness. This geometric viewpoint simplifies things immensely. The basic approach is to consider configurations to be vectors in a linear vector space, where

the space is usually infinite dimensional. We will show that selecting the next configuration can be interpreted as an orthogonal projection of the present configuration onto the set of all candidate next configurations.

Let $\mathcal{C}$ be a set of all configurations with a universe E_u which have a common interpretation for constant symbols, value predicates, and Skolem symbols. We focus our attention on the relations AE, CT, and ST, and build the geometric viewpoint based on them. $\mathcal{C}$, then, is the set of all triples (AE, CT, ST), where $AE \subseteq E_u, CT \subseteq E_u \times E_u, ST \subseteq E_u \times E_u$.

Let P be a subset of $\mathcal{C}$. P is meant to be the set of all configurations satisfying the right hand side, but for now we take it to be arbitrary. Given an arbitrary configuration $\mathcal{A}$ (which is meant to be the current configuration), our problem is to find a unique configuration $\mathcal{A}_o$ in P that is closest to $\mathcal{A}$.

The formalization of our concept of relative closeness is as follows:

Definition 1 *Let $\mathcal{A}, \mathcal{A}_o, \mathcal{A}_s$ be configurations in $\mathcal{C}$. We write $\mathcal{A} - \mathcal{A}_o \leq \mathcal{A} - \mathcal{A}_s$, meaning that $\mathcal{A}_o$ is closer to $\mathcal{A}$ than $\mathcal{A}_s$ is to $\mathcal{A}$, or $\mathcal{A}_o = \mathcal{A}_s$, when :*

$$AE \Delta AE_o \subseteq AE \Delta AE_s$$
$$CT \Delta CT_o \subseteq CT \Delta CT_s$$
$$ST \Delta ST_o \subseteq ST \Delta ST_s$$

where Δ denotes symmetric set difference (that is, the points in one of the sets but not both).

Now our problem is, for a given P and an arbitrary $\mathcal{A}$, to find $\mathcal{A}_o \in P$ such that

$$(\forall \mathcal{A}_s)(\mathcal{A}_s \in P \longrightarrow \mathcal{A} - \mathcal{A}_o \leq \mathcal{A} - \mathcal{A}_s)$$

This involves two stages, one to show that such an $\mathcal{A}_o$ exists, and another to characterize it in terms of $\mathcal{A}$ and P. The first problem is, of course, that this may not be possible for arbitrary P's. Therefore, the problem becomes to characterize the ones for which it is , that is, to characterize P's such that

$$(\forall \mathcal{A})(\exists \mathcal{A}_o)(\forall \mathcal{A}_s)(\mathcal{A}_o \in P \wedge (\mathcal{A}_s \in P \longrightarrow \mathcal{A} - \mathcal{A}_o \leq \mathcal{A} - \mathcal{A}_s)) \quad (1)$$

Notice that if such an $\mathcal{A}_o$ exists, it is necessarily unique. To see this let $\mathcal{A}_o$ and $\mathcal{A}'_o$ be two configurations in P that are closest to $\mathcal{A}$. By equation (1) we have :

$$(\mathcal{A} - \mathcal{A}_o \leq \mathcal{A} - \mathcal{A}'_o) \wedge (\mathcal{A} - \mathcal{A}'_o \leq \mathcal{A} - \mathcal{A}_o) \Longrightarrow \mathcal{A} - \mathcal{A}'_o = \mathcal{A} - \mathcal{A}_o \Longrightarrow \mathcal{A}'_o = \mathcal{A}_o$$

Now that our goal is formally stated, we assume that we have a set P which satisfies equation (1) and derive some necessary conditions that P must satisfy. Next we will show that these conditions are also sufficient. Finally, for all such P's, we characterize $\mathcal{A}_o$. Also for the purpose of simplicity we assume that a configuration consists of a single unary relation. Because we are going to treat configurations as points in a space this simplification is without loss of generality. It is quite easy to see how the arguments we present for the simple case of a single unary relation applies to the general case, where we consider AE, CT, and ST all at the same time, and, in fact, it is easy to generalize to other applications. This simplification means that the set of configurations becomes the power set of the set of entities. So from here on we will be concerned with the subsets of a given arbitrary set rather than configurations.

Lemma 1 *Given an arbitrary nonempty set S , the power set of S (denoted by 2^S) is a vector space over $GF(2)$ (that is, the binary field of 0 and 1, see* [4, page 8] *where vector addition and scalar multiplication are defined as follows :*

$$x_1, x_2 \in 2^S \; : \quad x_1 + x_2 \stackrel{\text{def}}{=} x_1 \Delta x_2 \;\; ; \;\; 1 \cdot x_1 \stackrel{\text{def}}{=} x_1 \quad 0 \cdot x_1 \stackrel{\text{def}}{=} \emptyset$$

Proof: The proof is straightforward. It only involves the verification of the axioms characterizing a linear vector space.
◇

Note 1 : In this vector space

$$x_1 + x_1 = x_1 \Delta x_1 = \emptyset \;\Rightarrow\; x_1 = -x_1 \;\Rightarrow\; x_1 + x_2 = x_1 - x_2$$

Thus, from here on '$-$' and '$+$' will be used interchangeably. The reader must be careful not to confuse '$-$' which is symmetric set difference here , with the usual definition of '$-$' for sets which is asymmetric set difference. The '$-$' is never used for the latter purpose in this article unless specifically mentioned.

Note 2 : $\subseteq$ defines a partial order in 2^S. We will be using $\leq$ and $\subseteq$ interchangeably. Furthermore, we will be using the fact that $(2^S, \cap, \cup, S, \emptyset)$ is a boolean algebra implicitly.

Note 3 : Note the consistency between Definition 1 and this lemma.

Note 4 : For the sake of clarity, in all subsequent equations $\cap$ and $\cup$ are given precedence over $+$. That is :

$$(x \cup y) + (y \cap z) = x \cup y + y \cap z$$

The following are two equalities that are very useful in proofs.

$$For \;\; x, y, z \in 2^S \; : \quad (x + y) \cap z = x \cap z + y \cap z \tag{2}$$

$$For \;\; x, y, z \in 2^S \; : \quad x + y = x \cup y + x \cap y \tag{3}$$

Definition 2 *A* **Plane** *is a subset P in a vector space U characterized by a generating subspace V and a vector d in the following way :*

$$(\forall x)(\exists y)(x \in P \longleftrightarrow (y \in V) \wedge (x = d + y))$$

It can be shown that the following simpler definition of a plane is equivalent to the one above when the underlying vector space is 2^S. We will be using this new one more often.

Definition 3 *A subset P in the vector space of 2^S is a* **Plane** *if it contains the sum of any 3 of its (not necessarily distinct) vectors.*

This finishes the preliminaries. Now we turn to the necessary conditions.

Lemma 2 *If $P \subseteq 2^S$ satisfies equation (1), then P is a plane.*

Proof:
Let x, y, z be any three (not necessarily distinct) elements of P. Let $A = x + y + z$. From equation (1) there exists $A_o \in P$ such that for any $p \in P$: $A - A_o \leq A - p$ But x, y, z are in P, hence :

$$\left.\begin{array}{l} A - A_o \leq A - x \\ A - A_o \leq A - y \\ A - A_o \leq A - z \end{array}\right\} \Longrightarrow A - A_o \;\leq\; (A - x) \cap (A - y) \cap (A - z) = \; (y + z) \cap (z + x) \cap (x + y)$$

$$= (y \cap z \cap x + y \cap z + y \cap x + y \cap x + z \cap x + z \cap y + z \cap x + z \cap x \cap y) = \emptyset$$

$$\Longrightarrow A - A_o = \emptyset \Longrightarrow A = A_o \Longrightarrow A \in P$$

It follows from Definition 3 that P is a plane.
$\diamond$

Lemma 3 *If $P \subseteq 2^S$ satisfies equation (1), then P is* **Cubic** [7], *that is* $(\forall x)(\forall x_1)(\forall x_2)(x_1 \in P \wedge x_2 \in P \wedge x_1 \leq x \leq x_2 \longrightarrow x \in P)$

Proof: Take $x_1, x_2 \in P$; $x_1 \leq x \leq x_2$. ¿From equation (1) there exists $x_o \in P$ such that for any $p \in P$: $x - x_o \leq x - p$. But x_1, x_2 are in P hence :

$$\left.\begin{array}{l} x - x_o \leq x - x_1 \\ x - x_o \leq x - x_2 \end{array}\right\} \Longrightarrow x - x_o \leq (x - x_1) \cap (x - x_2) = (x + x \cap x_2 + x \cap x_1 + x_1 \cap x_2)$$
$$= x + x + x_1 + x_1 = \emptyset \Longrightarrow x - x_o = \emptyset \Longrightarrow x = x_o \Longrightarrow x \in P$$

$\diamond$

Lemma 4 *If $P \subseteq 2^S$ satisfies equation (1),then P is* **Closed**, *that is the limit of any monotonically increasing (decreasing) sequence of elements of P is in P. The limit of a monotonically increasing (decreasing) sequence is defined in the usual way, that is, as the union (intersection) of the elements in the sequence.*

Proof :
a) monotonically increasing sequences. The statement of the lemma is :

$$(x_i \in P \;\; i = 1, 2, ...) \wedge (x_i \leq x_j \leftrightarrow i \leq j \;\; i, j = 1, 2, ...) \Longrightarrow x = \lim_{i \to \infty} x_i = \bigcup_{i=1}^{\infty} x_i \in P$$

Note that $x = \bigcup_{i=1}^{\infty} x_i$ is well defined because x_i's are in the power set of S;therefore, $x \subseteq S$. ¿From equation (1) there exists $x_o \in P$ such that for any $p \in P$: $x - x_o \leq x - p$. But x_i $i = 1, 2, ...$ are in P hence :

$$(\forall x_i)(x - x_o \leq x - x_i = (x \cap x_i') \cup (x' \cap x_i) = x \cap x_i') \Longrightarrow$$

$$x - x_o \leq \bigcap_{i=1}^{\infty} x \cap x_i' = x \cap \bigcap_{i=1}^{\infty} x_i' = x \cap (\bigcup_{i=1}^{\infty} x_i)' = x \cap x' = \emptyset \;\; \Longrightarrow x = x_o \Longrightarrow x \in P$$

b) monotonically decreasing sequences. The statement of the lemma is :

$$(x_i \in P \;\; i = 1, 2, ...) \wedge (x_i \geq x_j \leftrightarrow i \leq j \;\; i, j = 1, 2, ...) \Longrightarrow x = \lim_{i \to \infty} x_i = \bigcap_{i=1}^{\infty} x_i \in P$$

The proof of this part is analogous to that of part a) and hence omitted.
$\diamond$

Lemma 5 *If $P \subseteq 2^S$ satisfies equation (1), then every* **translation** *of P is a* **Closed Cubic Plane**, *where a translation of P is any set*

$$P_d = \{y \mid (y = x + d) \wedge (x \in P)\}$$

where d is a fixed vector.

[7] There are a number of terms that might be used here, for example, ideal, model complete, interval, or free. We borrowed "cubic" from switching theory.

Proof :
a) P_d is a plane
Let y_1, y_2, y_3 be any three vectors in P_d. By definition we have

$$y_i = x_i + d \ , \ x_i \in P \ ; \ i = 1, 2, 3$$

We have already shown that P must be a plane, hence $x_1 + x_2 + x_3 \in P$.

$$y_1 + y_2 + y_3 = (x_1 + x_2 + x_3) + d \implies y_1 + y_2 + y_3 \in P_d$$

b) P_d is cubic
Let $y_1 \leq y \leq y_2$, where y is any vector and y_1, y_2 are vectors in P_d. By definition we have $y_i = x_i + d \ , x_i \in P \ , i = 1, 2$. Let $x = y + d$. From equation (1) there exists $x_o \in P$ such that for any $p \in P$: $x - x_o \leq x - p$. But $x_i; \ i = 1, 2$ are in P, hence :

$$x - x_o \leq (x - x_1) \cap (x - x_2) = (y - y_1) \cap (y - y_2) = y + y + y_1 + y_1 = \emptyset \implies x = x_o \implies x \in P \implies y \in P_d$$

c) P_d is closed
Proof of this part is analogous to proof of part b) and hence omitted.
$\diamond$

Lemma 6 *If $P \subseteq 2^S$ and all its translations are* **Closed Cubic Planes** *, then P has a least element ℓ and a greatest element g.*

Proof : Only the proof for the existence of g is given. The other case is similar.
Being closed means that every monotonically increasing chain in P has a limit in P. Zorn's lemma [5, Page 556] states that :
If every chain in a partially ordered set has an upperbound, then the set has a maximal element.
Therefore, P must have a maximal element g. To show that g is the greatest element assume to the contrary that there exists $x \in P$, $x \not\leq g$. This means that $x \cup g > g$, which is a contradiction once we show that $x \cup g \in P$. To show $x \cup g \in P$ consider the set $P + g$ which is a translation of P and hence is cubic by hypothesis.

$$(x \cap g + g) \cap (x + g) = x \cap g + x \cap g + g \cap x + g = x \cap g + g \implies \emptyset \leq (x \cap g + g) \leq (x + g)$$

Since $\emptyset \in P + g$ and $x + g \in P + g$ the cubic property implies that $x \cap g + g \in P + g$ which results in $x \cap g \in P$. Using equation

$$x \cup g = x + g + x \cap g \implies x \cup g \in P$$

$\diamond$

The previous lemmas have provided all the necessary and sufficient conditions. Now we are ready to characterize the sets having a unique approximation for every point in the space.

Lemma 7 *$P \subseteq 2^S$ and all its translations are* **Closed Cubic Planes** *if and only if*

$$(\forall x)(\exists x_o)(\forall p)(x_o \in P \wedge (p \in P \longrightarrow x - x_o \leq x - p))$$

Proof : The *if* part is already done by previous lemmas. For the *only if* part take any $x \in 2^S$ and set $x_o = (x \cup \ell) \cap g$, where ℓ and g are the least and greatest elements of P (lemma 6) .

To complete the proof $x_{\prime\prime}$ must be shown to be in P and it must be shown to be the closest element of P to x.

$$\left.\begin{array}{l} x_{\prime\prime} \cap g = x_{\prime} \Longrightarrow x_{\prime} \leq g \\ x_{\prime\prime} \cap \ell = \ell \Longrightarrow \ell \leq x_{\prime\prime} \\ (\ell \in P) \wedge (g \in P) \end{array}\right\} \Longrightarrow \quad x_{\prime\prime} \in P$$

To show that $x_{\prime\prime}$ is the closest element of P to x take any $p \in P$
$(x - x_{\prime}) \cap (x - p) = x + x \cap p + x_{\prime} \cap x + x_{\prime\prime} \cap p = x + x \cap p + x \cap g + (x \cup \ell) \cap p$
Now apply equation 3 to the last two terms to get

$$x + x \cap p + x \cap p + (x \cap g) \cup ((x \cup \ell) \cap p) = x + (x \cup \ell) \cap ((x \cap g) \cup p) = x + (x \cup \ell) \cap ((x \cup p) \cap g) =$$

$$x + (x \cup (\ell \cap p)) \cap g = x + x_{\prime} \quad \Longrightarrow x - x_{\prime} \cap (x - p) = x - x_{\prime} \Longrightarrow x - x_{\prime} \leq x - p$$

$\diamond$

The lemma just proved characterizes our desired sets as sets all of whose translations are closed cubic planes. Let's briefly investigate the independence of these four conditions. It can be shown that being a plane is implied by staying cubic under translations but the other three conditions are independent. The following examples show what goes wrong if any of the three properties of closedness, cubic-ness, and invariance under translations is omitted.

Example 1 *Let S have an infinite number of elements. Let $P \in 2^S$ be the set of all finite subsets of S. P and all its translations are obviously planes. Furthermore, P is obviously cubic. Translations of P are also cubic; however, this may not be so obvious. Consider $P + d$, where d is a vector (reminder : vectors are sets) of infinite cardinality (if d is of finite cardinality things are obvious). Let $x_1, x_2 \in P + d$ and $x_1 \leq x \leq x_2$. x_1, x_2 are different from d at a finite number of places. This implies that x can be different from d at only a finite number of places. Hence $x + d$ is finite which means that $x \in P$.*
So we see that P and all its translations are cubic planes but they are obviously not closed. Moreover, it is not difficult to see that P does not have a closest approximation for any infinite vector.

Example 2 *Let $S = \{a, b, c\}, P = \{\emptyset, S\}$. It is easy to verify that P and all its translations are closed planes but not cubic. Moreover, P does not have a closest approximation for any point outside of it. For example consider $\{a\}$, neither point in P is a closest approximation for this point because $\{a\} - \emptyset$ and $\{a\} - S$ are not even comparable.*

Example 3 *Let $S = \{1, 2, 3, 4, 5\}, P = \{\{1, 3, 5\}, \{2, 3, 5\}, \{1, 4, 5\}, \{2, 4, 5\}\}$. It is easy to verify that P is a closed cubic plane. However, not all its translations are. Consider the translation of P under the vector $d = \{1, 3, 5\}$*

$$P + d = \{\emptyset, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}$$

Obviously this translation is not cubic, and it is easy to see that P has no closest approximation for many points, for example the point $\{5\}$.

The following theorem is the major result of this section. So far we have shown that the sets possessing the desired property of having a closest approximation for every point in the space are sets all of whose translations are closed cubic planes. The next theorem shows that these translation invariant closed cubic planes can be characterized as the set of all points that contain as a subset (remember points in the space are subsets of S) a given

point and have no intersection with another given point. In terms of configurations, it says that the set of all configurations such that every one of the three relations AE, CT, ST of every configuration is true on a given portion of the space, false on another given portion of the space, and arbitrary elsewhere; characterizes a desired right hand side. This was conjectured in [1].

Theorem 8 *For $P \subseteq 2^S$ the following three statements are equivalent :*
1) P and all its translations are **Closed Cubic Planes**
2) $(\forall x)(\exists x_o)(\forall p)(x_o \in P \wedge (p \in P \longrightarrow x - x_o \leq x - p))$
3) There exist sets $P_T, P_F \in 2^S$, $P_T \cap P_F = \emptyset$ such that

$$(\forall x)(x \in P \longleftrightarrow (P_T \subseteq x) \wedge (x \cap P_F = \emptyset))$$

Proof: Previous lemma showed that 1) if and only if 2). Here we will show that 1) $\rightarrow$ 3) and 3) $\rightarrow$ 2) to finish the proof.
1) $\rightarrow$ 3)
P has a least element ℓ and a greatest element g as shown by Lemma 6. Set $P_T = \ell, P_F = g'$ (where g' is complement of g). To show that 3) holds, consider any $x \in P$. Because P is cubic and because of the way ℓ, g, P_T, P_F are defined we have :

$$x \in P \Longleftrightarrow \ell \leq x \leq g \Longleftrightarrow (\ell \subseteq x) \wedge (x \subseteq g) \Longleftrightarrow (P_T \subseteq x) \wedge (x \cap P_F = \emptyset)$$

3) $\rightarrow$ 2)
To show that 2) holds set $x_o = (x \cup P_T) - P_F$, where '$-$' denotes the asymmetric set difference here. To complete the proof, x_o must be shown to be in P, and it must be shown to be the closest element of P to x. To show these simply replace P_T for ℓ and P'_F for g in the proof of the previous lemma.
$\diamond$

So far all the attention has been centered on the characteristics of sets that possess a unique approximation for every point in the space. Now we move on to show that this unique approximation is an orthogonal projection.

Definition 4 *Two sets $A, B \in 2^S$ are called* **orthogonal** *if $A \cap B = \emptyset$.*

Theorem 9 *Let P and all its translations be closed cubic planes. Lemma 6 shows that P has a least element ℓ and a greatest element g. Define :*

$$\pi : 2^S \longrightarrow P \ , \ \pi(x) = x_o = (x \cup \ell) \cap g$$

The mapping $(\pi - \ell)$ is an orthogonal projection on the subspace $P - \ell$

The statement of the theorem needs a word of explanation. Because $\ell \in P$, the plane $P - \ell$ is the translated version of P that is a subspace. Recall Definition 2. It should be obvious that $P - \ell$ is the generating subspace of P. It can be shown that the generating subspace of a plane is always unique and does not depend on the choice of the vector (here ℓ).
With the above in mind it should not be difficult to see what the statement of the theorem is saying. One question remains however: why are we talking about $(\pi - \ell)$ and $P - \ell$ and not directly about π and P ? The answer is that π fails to be linear whereas $(\pi - \ell)$ is linear and this is anything but unusual. In the Cartesian plane only lines through the origin are subspaces and only a projection on these lines is linear. Projection on a line that does not go through the origin can be in a sense orthogonal but not linear. This is exactly the case

here.
Proof of Theorem 9 :
Three things need to be established: $\pi - \ell$ is linear, $\pi - \ell$ is a projection on $P - \ell$, and $\pi - \ell$ is an orthogonal projection.
But first an elementary result needed in all three subproofs :

$$x \in 2^S : \quad (\pi-\ell)(x) = (x \cup \ell) \cap g + \ell \cap g = (x \cup \ell + \ell) \cap g = (((x \cup \ell) \cap \ell') \cup ((x \cup \ell)' \cap \ell)) \cap g = x \cap \ell' \cap g$$

1) Since 0 and 1 are the only scalars in this vector space the following two conditions suffice for linearity.

$$(\pi - \ell)(\emptyset) = (\emptyset \cup \ell) \cap g + \ell = \emptyset$$

$$x, y \in 2^S : \quad (\pi - \ell)(x + y) = (x + y) \cap \ell' \cap g = x \cap \ell' \cap g + y \cap \ell' \cap g = (\pi - \ell)(x) + (\pi - \ell)(y)$$

2) The previous theorem showed that $x_o = \pi(x) \in P$. Immediate consequence of this is that $(\pi - \ell)(x) \in (P - \ell)$ which implies that $P - \ell = Range(\pi - \ell)$. For $\pi - \ell$ to be a projection it must also be shown that $(\pi - \ell)^2 = (\pi - \ell)$.

$$x \in 2^S : \quad (\pi - \ell)^2(x) = (\pi - \ell)(x \cap \ell' \cap g) = (x \cap \ell' \cap g \cap \ell' \cap g) = (x \cap \ell' \cap g) = (\pi - \ell)(x)$$

3) To show that $(\pi - \ell)$ is an orthogonal projection on $P - \ell$ it must be shown that $x - (\pi - \ell)(x) \perp (\pi - \ell)(x)$.

$$x \in 2^S : \quad (x - (\pi - \ell)(x)) \cap (\pi - \ell)(x) = (x + x \cap \ell' \cap g) \cap (x \cap \ell' \cap g) = x \cap \ell' \cap g + x \cap \ell' \cap g = \emptyset$$

$\diamond$

7 Conditions for Unambiguous Updating

By now the necessary and sufficient conditions for unambiguous updating would appear to be obvious: the set of candidate next configurations must form a translation invariant closed cubic plane. However, this is not quite the case. In Section 6 we ignored the role of Θ, that is, the sentences that are always true, and the role of L, that, is the sentences in the left hand side of the change. We allowed the current configuration to be any configuration in $\mathcal{C}$, and we know that only those configurations which satisfy all the sentences in Θ and L need be considered. Furthermore we did not require the resultant configuration to satisfy Θ, and we should have. This section is devoted to these two further considerations and how they affect our scheme for updating.

Needless to say, if R, the sentences in the right hand side, together with Θ characterize a translation invariant closed cubic plane, there is nothing new to say. The problem occurs when they do not, and this can happen in realistic cases.

Let P_R, P_Θ, P_L, $P_{L,\Theta}$, and $P_{R,\Theta}$ be, respectively, the set of configurations that satisfy R, Θ, L, $L \cup \Theta$, and $R \cup \Theta$. Clearly, $P_{R,\Theta} = P_R \cap P_\Theta$, and $P_{L,\Theta} = P_L \cap P_\Theta$. Presumably, we are interested in the case where $P_{R,\Theta}$, and $P_{L,\Theta}$ are not empty, that is, R and L are both consistent with Θ .

Let us consider an example in which we restrict our attention to just CT. If $R = CT(k_1, k_2) \wedge CT(k_2, k_1)$, we know that P_R is a translation invariant closed cubic plane (characterized by $P_{R_T} = \{(k_1, k_2), (k_2, k_1)\}, P_{R_F} = \emptyset$, where P_{R_T} and P_{R_F} are the sets introduced in the third part of Thereom 8.). If Θ says that CT is symmetric, that is, $(\forall x)(\forall y)(CT(x, y) \rightarrow CT(y, x))$, then P_Θ is a closed subspace, but it is not cubic. $P_{R,\Theta}$

is all symmetric configurations satisfying $CT(k_1,k_2) \wedge CT(k_2,k_1)$, and it is a closed plane but not a cubic one. However, in this case updating is still unambiguous. If CT is any symmetric CT-relation, its orthogonal projection onto the translation invariant closed cubic plane P_R is also symmetric, and, therefore, in $P_{R,\Theta}$.

On the other hand, if $R = CT(k_1,k_2)$, P_R is still a translation invariant closed cubic plane. However, now the orthogonal projection of a symmetric CT onto P_R will not be symmetric. Thus, for a given Θ, there are certain translation invariant closed cubic planes that lead to unambiguous updating and ones that do not.

Our task is, therefore, to go from a current configuration $A \in P_{L,\Theta}$ to the next configuration $A_o \in P_{R,\Theta}$ such that A_o is the closest element of $P_{R,\Theta}$ to A. Our problems are :
1) How to make sure there exists such an A_o and that it is unique
2) How to find such an A_o

Of course we can get around these problems by requiring $P_{R,\Theta}$ to be a translation invariant closed cubic plane, but that is a stronger than necessary requirement which is not satisfied in many realistic cases like the examples above. So it might appear that we have to discard all the results of the last section regarding translation invariant closed cubic planes and start over again. But, although we have to start over again, the results of the last section turn into very useful tools that simplify our problems immensely. In fact, updating is still carried out by an orthogonal projection onto a translation invariant closed cubic plane.

Again, for the sake of simplicity, we are going to consider the subsets of a given set S instead of configurations. We keep on using P_R, P_Θ, $P_{R,\Theta}$, P_L, and $P_{L,\Theta}$, but interpret them in the new context. Next some definitions and lemmas are presented which culminate in a test for verification of the existence of A_o . Specification of A_o, when it exists, is taken care of next. Finally we draw some general conclusions using illustrative examples.

Definition 5 *Consider an arbitrary set $Q \subseteq 2^S$. The* **Kin** *of $q, q \in Q$, with respect to Q is the set of all points in the space (that is 2^S) that q is closer to than any point in $Q - \{q\}$ is. That is*

$$Kin(q) = \{x \mid x - q \leq x - q' \; ; \; q' \in Q\}$$

The following is a set-theoretic identity that will be used later.

Lemma 10 *For $R \subseteq 2^S, x \in 2^S$ let $x + R = \{x + r | r \in R\}$, then*

$$\bigcap(x+R) = x + x \cap (\bigcap R + \bigcup R) + \bigcap R$$

Proof : Using the fact that $A \cap B' = A + A \cap B$ we get

$$x + x \cap (\bigcup R + \bigcap R) + \bigcap R = (x + x \cap \bigcup R) + (\bigcap R + x \cap \bigcap R) = (x \cap (\bigcup R)') + (\bigcap R \cap x')$$

Now apply equation (3) of last section
$= (x \cap (\bigcup R)') \cap (\bigcap R \cap x') + (x \cap (\bigcup R)') \cup (\bigcap R \cap x')$
$= \emptyset + (x \cap (\bigcup R)') \cup (\bigcap R \cap x') \cup (x \cap x') \cup (\bigcap R \cap (\bigcup R)') = (x' \cup (\bigcup R)') \cap (\bigcap R \cup x)$
$= (x' \cup \bigcap \{r' | r \in R\}) \cap (\bigcap \{r | r \in R\} \cup x) = \bigcap \{x' \cup r' | r \in R\} \cap \bigcap \{x \cup r | r \in R\}$
$= \bigcap \{(x' \cup r') \cap (x \cup r) | r \in R\} = \bigcap \{x + r | r \in R\} = \bigcap (x + R)$
$\diamond$

Lemma 11 *The kins of two different points are disjoint.*

Proof : Let k be a point that is in kins of both q_1, and q_2.

$$(k \in kin(q_1) \wedge k \in kin(q_2)) \Longrightarrow (k-q_1 \leq k-q_2) \wedge (k-q_2 \leq k-q_1) \Longrightarrow k-q_1 = k-q_2 \Longrightarrow q_1 = q_2$$

$\diamond$

Lemma 12 *Consider an arbitrary set $Q \subseteq 2^S$. The kin of $q, q \in Q$, with respect to Q is a translation invariant closed cubic plane. Furthermore, the kins of $q_1 \in Q$, and $q_2 \in Q$ with respect to Q are simply shifted versions of each other.*

Proof :
a) Let K = kin of $q = \{k | k \in 2^S, k - q \leq k - q', q' \in Q\}$.
Use Theorem 8 to show that K is a translation invariant closed cubic plane.

$$Let \quad K_T = q + \bigcap Q \quad ; \quad K_F = q + \bigcup Q$$

First we have to verify that $K_T \cap K_F = \emptyset$

$$K_T \cap K_F = (q + \bigcap Q) \cap (q + \bigcup Q) = q + q + \bigcap Q + \bigcap Q = \emptyset$$

What remains to be shown is that the following well-formed formula is true :

$$(\forall x)(x \in K \longleftrightarrow (K_T \subseteq x) \wedge (x \cap K_F = \emptyset))$$

By definition of K we have $(x-q) = \bigcap(x-Q)$ for $x \in K$. Apply Lemma 10 to this equality to get $q = x \cap \bigcup Q + x \cap \bigcap Q + \bigcap Q$. For the right implication let $x \in K$.

$$x \cap K_T = x \cap (q + \bigcap Q) = x \cap (x \cap \bigcup Q + x \cap \bigcap Q) = (x \cap \bigcup Q + x \cap \bigcap Q) = q + \bigcap Q = K_T$$
$$\Longrightarrow K_T \subseteq x$$
$$x \cap K_F = x \cap (q + \bigcup Q) = x \cap (x \cap \bigcup Q + x \cap \bigcap Q + \bigcap Q + \bigcup Q) = x \cap (\bigcup Q + \bigcap Q + \bigcap Q + \bigcup Q)$$
$$\Longrightarrow K_F \cap x = \emptyset$$

For the left implication let $K_T \subseteq x$, $x \cap K_F = \emptyset$

$$(x \cap (q + \bigcap Q) = q + \bigcap Q) \wedge (x \cap (q + \bigcup Q) = \emptyset) \Rightarrow (x \cap q + x \cap \bigcap Q = q + \bigcap Q) \wedge (x \cap q = x \cap \bigcup Q)$$
$$\Rightarrow x \cap \bigcup Q + x \cap \bigcap Q = q + \bigcap Q \Rightarrow q = x \cap \bigcup Q + x \cap \bigcap Q + \bigcap Q \Rightarrow x - q = \bigcap(x - Q) \Rightarrow x \in K$$

b) Let K_1 and K_2 be the kins of q_1 and q_2 respectively. We claim that $K_2 = K_1 + q_1 + q_2$.
Recall that by definition :

$$K_i = \{k | k \in 2^S, k - q_i \leq k - q', q' \in Q\} \quad i = 1, 2$$

To prove the claim let $x_1 \in K_1$, and q' be any element in Q then :

$$((x_1 + q_1 + q_2) + q') \cap ((x_1 + q_1 + q_2) + q_2) = ((x_1 + q_1) + (x_1 + q_2) + (x_1 + q')) \cap (x_1 + q_1) =$$
$$(x_1 + q_1) + (x_1 + q_1) + (x_1 + q_1) = (x_1 + q_1) = ((x_1 + q_1 + q_2) + q_2) \Rightarrow (x_1 + q_1 + q_2) \in K_2$$

Similarly it can be shown that : $(x_2 + q_1 + q_2) \in K_1$ which proves the claim.
$\diamond$

Now we are ready for the test for the verification of the existence of A_o

Definition 6 *P_R, P_L, and P_Θ are called* **compatible** *when for any point in $P_{L,\Theta}$ there is a unique point in $P_{R,\Theta}$ that is the closest point of $P_{R,\Theta}$ to that point.*

Lemma 13 *P_R, P_L, and P_Θ are* **compatible** *if and only if $P_{L,\Theta}$ is a subset of the union of the kins of the points of $P_{R,\Theta}$ (with respect to $P_{R,\Theta}$).*

Proof : If P_R , P_L , and P_Θ are compatible, then the claim of the lemma is an immediate consequence of the definition of the kin. If $P_{L,\Theta}$ is a subset of the union of the kins of the points of $P_{R,\Theta}$, then take any point $A \in P_{L,\Theta}$. A is in the kin of one (and only one) point of $P_{R,\Theta}$ and hence that unique point is the closest element of $P_{R,\Theta}$ to A. Therefore compatibility holds.

◇

Now that we are assured of the existence and the uniqueness of A_o we turn to the second problem of how to find A_o. Had $P_{R,\Theta}$ been a translation invariant closed cubic plane we would have found A_o by taking the orthogonal projection of A onto $P_{R,\Theta}$. Here $P_{R,\Theta}$ is not, but we claim that A_o is the orthogonal projection of A onto the smallest translation invariant closed cubic plane containing $P_{R,\Theta}$ as a subset.

Lemma 14

$$Let \quad \ell = \bigcap P_{R,\Theta} \ , \quad g = \bigcup P_{R,\Theta} \ , \quad P = \{p | \ell \leq p \leq g\}$$

P is the smallest translation invariant closed cubic plane containing $P_{R,\Theta}$.

Proof : let $P_T = \ell$, $P_F = g' = 2^S - g$, then we have :

$$P = \{p | (P_T \subseteq p) \ \wedge \ (p \cap P_F = \emptyset)\}$$

Hence by Theorem 8 of the previous section P is a translation invariant closed cubic plane. The fact that P is the smallest one is obvious by the construction of P and the choices for ℓ and g.

◇

The following theorem sums up the results of this section. In simple words, it is saying that unambiguous updating is possible when P_R, P_L, and P_Θ are compatible and updating is done by taking an orthogonal projection.

Theorem 15 *When P_R, P_L, and P_Θ are compatible , the closest element of $P_{R,\Theta}$ to a given $A \in P_{L,\Theta}$ is the orthogonal projection of A onto the smallest translation invariant closed cubic plane containing $P_{R,\Theta}$.*

Proof : $A \in P_{L,\Theta}$ plus the fact that P_R, P_L, and P_Θ are compatible implies that there exists $A_o \in P_{R,\Theta}$ such that A is in the kin of A_o, hence A_o is the closest element of $P_{R,\Theta}$ to A. What remains to be shown is that A_o is the orthogonal projection of A onto plane P defined below , or to be precise $(A_o - \ell)$ is an orthogonal projection onto $P - \ell$ (see the discussion of Theorem

$$P = \{p | \ell \leq p \leq g \ \ , \ \ell = \bigcap P_{R,\Theta} \ , \ g = \bigcup P_{R,\Theta}\}$$

Using Theorem 9 all we need to do is to show that the following relation between A and A_o holds : $A_o = (A \cup \ell) \cap g$

To show this we use Lemma 12. Let $K = kin(A_o)$ with respect to $P_{R,\Theta}$. Since $A \in K$ we have :

$$(A \cap K_T = K_T) \wedge (A \cap K_F = \emptyset) \implies (A \cap (A_o + \ell) = A_o + \ell) \wedge (A \cap (A_o + g) = \emptyset) \implies$$
$$(A \cap A_o + A \cap \ell = A_o + \ell) \wedge (A \cap A_o = A \cap g) \implies (A \cap g + A \cap \ell = A_o + \ell) \implies$$
$$((A \cap g) + (A \cap g \cap \ell) + \ell = A_o) \implies ((A \cap g) \cup \ell = A_o) \implies ((A \cup \ell) \cap g = A_o)$$

$\diamondsuit$

Now that we have characterized the condition for unambiguous updating (namely requiring P_R, P_L, and P_Θ to be compatible) and showed how to carry out the updating (by taking an orthogonal projection), let us go back to the two examples we discussed at the beginning of the section and see how each can be treated.

Let $R = CT(k_1, k_2) \wedge CT(k_2, k_1)$, and let Θ say that CT is symmetric, that is, $(\forall x)(\forall y)(CT(x, y) \rightarrow CT(y, x))$. We ignore L in this example. Let us see whether P_Θ and P_R are compatible. $P_{R.\Theta}$ is all symmetric configurations satisfying $CT(k_1, k_2) \wedge CT(k_2, k_1)$. Hence the kin of any point in $P_{R.\Theta}$ consists of four points which agree everywhere except on the ordered pairs (k_1, k_2) and (k_2, k_1) (where they are arbitrary). The union of all the kins is the set of all configurations whose CT is symmetric everywhere with the possible exception of locations (k_1, k_2) and (k_2, k_1). This union obviously includes P_Θ as a subset. Hence P_Θ and P_R are compatible.

To update a current configuration A here, we take its orthogonal projection on the smallest translation invariant closed cubic plane through $P_{R.\Theta}$. The result is

$$A_o = (A \cup \ell) \cap g \quad \text{where :} \quad \ell = \bigcap P_{R.\Theta} \ , \ g = \bigcup P_{R.\Theta}$$

Which is equivalent to saying : The CT of the next configuration is true on locations (k_1, k_2) and (k_2, k_1) , but everywhere else it is exactly like the CT of the current configuration. A result that is in perfect agreement with our intuition of what the above right hand side is trying to do.

The next example is more interesting. Let $R = CT(k_1, k_2)$, Θ be as before (all symmetric configurations), and as before ignore L. Let us see whether P_Θ and P_R are compatible. $P_{R.\Theta}$ is, as before, the set of all symmetric configurations satisfying $CT(k_1, k_2) \wedge CT(k_2, k_1)$. Notice that although R only specifies $CT(k_1, k_2)$, because of Θ we are forced to include $CT(k_2, k_1)$ as well. From here on this example is just like the previous one and updating under the right hand sides of either of these examples yields the same "next configuration".

The above examples illustrate a very important point. They show that the modeling system developed here is powerful enough to keep Θ always satisfied without being explicitly told (in the right hand sides of the changes) how to do so. This means that the designer of the changes need not worry about making sure that his right hand sides are in harmony with the things that are always true (Of course this presupposes compatibility). This may be an insignificant advantage when Θ is simple but it becomes a very significant one with a complex Θ.

The next example shows another important point. It demonstrates the capability of the modeling system to detect bad combinations of L, R, and Θ.

Let $R = ST(k_1, k_2)$, and let Θ be the following well-formed formulas :

$$(\forall x)(ST(x, x))$$
$$(\forall x_1) \ldots (\forall x_n)(ST(x_1, x_2) \wedge \ldots \wedge ST(x_{n-1}, x_n) \wedge ST(x_n, x_1) \longrightarrow x_1 = x_n)$$

In other words, Θ is saying that ST is reflexive and its transitive closure is antisymmetric, both of which make eminently good sense given the intuitive notion of a substructure relationship because we do not want to allow non-trivial substructure loops.

Assume that the current configuration is A and the following are true in A :

$$ST(k_2, k_3) \wedge ST(k_3, k_1) \quad k_1 \neq k_2 \neq k_3$$

Let L be arbitrary otherwise, as long as $A \in P_{L,\Theta}$ (for example L can be $CT(k_1, k_2)$). Let us see whether L, R, and Θ are compatible.
Let q be an arbitrary element of $P_{R,\Theta}$. $Kin(q)$ is a translation invariant closed cubic plane characterized by K_T, and K_F where :

$K_T = q + \bigcap P_{R,\Theta} =$
$q + \{x \mid x$ is a configuration in which $ST(k_1, k_2)$ is true, and everything else is false$\}$
$K_F = q + \bigcup P_{R,\Theta} =$
$q + \{x \mid x$ is a configuration in which $ST(k_2, k_1)$ is false, and everything else is true$\}$

Since for any $y \in Kin(q)$ we must have $(K_T \subseteq y \wedge y \cap K_F = \emptyset)$, we conclude that $Kin(q)$ consists of four points (configurations) which are all exactly like q everywhere except on the ordered pairs (k_1, k_2), and (k_2, k_1) (where they are arbitrary). The union of these kins is the set of all configurations that are arbitrary on the ordered pairs (k_1, k_2), and (k_2, k_1) , and at other locations are exactly like a point (configuration) in $P_{R,\Theta}$. Now consider the current configuration A which is true on the ordered pairs (k_2, k_3), and (k_3, k_1). If A is in the union of the kins of the points of $P_{R,\Theta}$, there must be a point of $P_{R,\Theta}$ which is exactly like A at all locations except for ordered pairs (k_1, k_2), and (k_2, k_1). Thus at this point both $ST(k_2, k_3)$, and $ST(k_3, k_1)$ are true. Moreover, at this point $ST(k_1, k_2)$ is also true because this point belongs to P_R. But the truth of the above three substructures at this point is in clear contradiction with Θ. Hence we have a contradiction, which implies that A is not in the union of the kins of the points of $P_{R,\Theta}$ which means that L, R, and Θ are not compatible.

When a situation like that in the last example happens, it warns the system designer of an ambiguity in the system. In the above example ambiguity arises because the specified action of the right hand side creates a non-trivial loop of length three of substructure relations. Since such loops are prohibited (by Θ) the loop must be broken somewhere, but where? It can be broken at any of the three places and that is ambiguous. The course of action that the system designer should take when encountering such a situation is to expand L to prevent this change from taking place when there is a possibility of trouble (here creating a loop); or to expand R to remedy the situation should the trouble occur (here by specifying how to break the loop).

We conclude by saying that the modeling system provided here not only answers some theoretical questions regarding the problem of updating, but it also is powerful enough to help the designer of the system with his design. In fact for complex systems a powerful tool like this is indispensable.

8 Syntactic Characterizations of Conditions

We see now that it is important that we be able to recognize translation invariant closed cubic planes, and we have given a definition in semantic (i.e., set-theoretic) terms. Now we have to connect this semantic definition to the syntax ; otherwise, we will not be able to recognize *sentences* that yield translation invariant closed cubic planes.

Perhaps it is well to repeat what the situation is. We are not concerned with *all* configurations which satisfy a set of sentences. At least we are not concerned with them all at once. Rather we are concerned with special classes of configurations. In particular, any one of these classes is made up of configurations that have the same universe and that interpret constant symbols, value predicate symbols, and Skolem symbols in the same way. That is,

they differ only in the ways that they interpret AE, CT, and ST. Within one of these classes we are interested in the configurations that satisfy a set of sentences, and we want to know if they form a translation invariant closed cubic plane.

We also know from Theorem 8 that a translation invariant closed cubic plane is characterized by sets AE_T, AE_F, CT_T, CT_F, ST_T, and ST_F. Thus, we need sentences that will define these sets, and, of course, these sets have to be defined either explicitly or implicitly using symbols from our first-order language that are not AE, CT, or ST.

We define these sets explicitly [8] as follows:

$$(\forall x)(AE_T(x) \leftrightarrow \phi_{AET}(x)) \qquad (\forall x)(AE_F(x) \leftrightarrow \phi_{AEF}(x))$$

where $\phi_{AET}(x)$, and $\phi_{AEF}(x)$ are well-formed formulas with x as the only free variable and which do not contain AE, CT, or ST, and, of course, the two sets must be disjoint which means that

$$(\forall x)(\neg(\phi_{AET}(x) \wedge \phi_{AEF}(x)))$$

The sets CT_T, CT_F, ST_T, and ST_F are taken care of similarly.

We then describe translation invariant closed cubic planes with three sentences as follows:

$$(\forall x)((\phi_{AET}(x) \wedge AE(x)) \vee (\phi_{AEF}(x) \wedge \neg AE(x)) \vee \neg(\phi_{AET}(x) \vee \phi_{AEF}(x))$$

Any AE which satisfies this sentence must be true on AE_T, false on AE_F, and arbitrary elsewhere. The other two sentences are

$$(\forall x)(\forall y)((\phi_{CTT}(x,y) \wedge CT(x,y)) \vee (\phi_{CTF}(x,y) \wedge \neg CT(x,y)) \vee \neg(\phi_{CTT}(x,y) \vee \phi_{CTF}(x,y))$$

and

$$(\forall x)(\forall y)((\phi_{STT}(x,y) \wedge ST(x,y)) \vee (\phi_{STF}(x,y) \wedge \neg ST(x,y)) \vee \neg(\phi_{STT}(x,y) \vee \phi_{STF}(x,y))$$

Example 4 *Suppose that the right side is*

$$AE(k_1) \wedge AE(k_2) \wedge \neg CT(k_1,k_2) \wedge \neg CT(k_2,k_1)$$

In the above canonical form this becomes

$$\begin{aligned}
\phi_{AET} &= ((x = k_1) \vee (x = k_2)) \\
\phi_{AEF} &= False \\
\phi_{CTT} &= False \\
\phi_{CTF} &= ((x = k_1) \wedge (y = k_2)) \vee ((x = k_2) \wedge (y = k_1))
\end{aligned}$$

Of course, the original form of R in the foregoing example is preferable to the canonical form, and, in fact, many translation invariant closed cubic planes are characterized by such conjunctions. The canonical form just provides us with a convenient test. Finally, a technical point, there are sets that cannot be defined using our first-order language: therefore, there are translation invariant closed cubic planes that cannot be defined using our first-order language. However, they are of no practical interest.

[8] Beth's Theorem [6, page 87] guarantees that an implicit definition of AE_T etc is not necessary.

9 The Connection to Situation-based Planning, the Frame Problem, and Rule-based Systems

Robot planning problems are discussed in most introductory texts on Artificial Intelligence (for example, see [7, page 252]). In brief the problem is that given the current status of the world(that is the current 'situation'), represented in the context of first-order logic, how do we go about changing the world to attain a goal status. One popular method for going about solving this problem is by using automatic theorem proving. But since the facts change from situation to situation a new kind of variable called a situation variable is introduced and added to the arguments of all predicates (This idea is due to Green [8]). The axioms of the theorem proving system are the operations or actions that can be performed by the robot (plus the facts that are true in the initial situation). A typical operation looks like :

$$L(s) \longrightarrow R(s')$$

where L and R are well-formed formulas and s and s' are situation variables. The above operation says how the status of the world change as the transition from situation s to situation s' is made. However, that is all it says; more specifically it does not say what happens to other things in the world, and this is where the frame problem arises. The problem is to specify what the next situation will be unambiguously.

One way to do this is to add more axioms for every operation. These added axioms, called frame axioms, simply assert what happens to each each part of a situation. It then becomes possible to use automatic theorem proving to find a sequence of operations (axioms) that change the current situation to a situation in which a desired goal statement is true. However, there are usually so many axioms in the system now that this method becomes impractical for all but small problems. That is why some robot problem-solving systems such as STRIPS [9] abandon the idea of situation variables and explicit frame axioms and solve the frame problem implicitly by requiring every operation to have a 'delete' list and an 'add' list on the right hand side rather than simply a statement that is to be true in the next situation. The status of the situation at any time is represented by a global database. When any action is performed the database is updated by making true the predicates in the 'add' list and making false the ones in the 'delete' list and leaving everything else intact (this is the implicit frame axiom).

We are interested in 'situations' and the frame problem because situations and our configurations are essentially the same kinds of semantic objects. However, a situation is usually viewed as part of a larger semantic object that is a collection of situations. Furthermore, our modeling system does not explicitly allow deduction to be used as a tool for planning. Rather than one logical system, we have many. At any given time deduction is applicable to the current configuration only. The frame problem can obviously be viewed as a kind of updating problem. In particular, one wants to specify a sufficient number of first-order sentences to characterize the next situation uniquely. Obviously, this is what we are interested in too. In fact, we analyze the problem in a context that goes beyond first-order logic, but, given our results, we can put everything back into a first-order system as is done in situation-based planning. It is an easy matter to write down a well-formed formula that characterizes an orthogonal projection which, we now know, is the solution to the updating (and frame) problem. This well-formed formula is the required frame axiom. Thus, there is not an explosion of frame axioms, we just need a few. Further, since we know that each orthogonal projection is characterized by an add and delete list (see, part 3 of 8), we see that the STRIPS approach fits into our theoretical framework also.

Anybody familiar with rule-based systems (see the Handbook of Artificial Intelligence for a good overview [10]) can see that our changes are similar to the rules of a rule-based system and that our configuration is reminiscent of the 'working memory'. Rule-based systems being more a programming paradigm than a modeling tool, however, never give a systematic treatment to the updating problem, and solve it implicitly by also restricting the right-hand side to an 'add,delete' list of the elements to be added to or deleted from the working memory.

Our analysis confirms, then, the intuitive idea of the 'add, delete' list, but it also shows that more can be achieved within our system (see discussion at the end of section 7). Another relevant, difference between our system and Rule-based systems is the presence of Θ (the set of sentences that must always be true) in our system.

10 Summary

We have presented a practical test for determining which models have unambiguous updating and which do not. Presumably, ones that do not are inappropriate models. In addition and perhaps more importantly, we have developed a geometric approach to the analysis of our logic-based model. It is fairly clear that this approach can be used to address a number of other questions regarding these models, and it is our intent to address them in subsequent articles.

References

[1] Naylor, A.W. and Maletz, M.C.,"The Manufacturing Game: A Formal Approach to Manufacturing Software," IEEE Trans. on Systems, Man, and Cybernetics, May-June, 1986

[2] Harper, W.L., Stalnaker, R., and Pearce, G. (eds.), "IFS Conditionals, Belief, Decision, Chance, and Time," D. Reidel, Dordrecht, Holland, 1981

[3] Enderton, H.B.,"A Mathematical Introduction to Logic," Academic Press, 1972

[4] Gill, A.,"Linear Sequential Circuits," McGraw-Hill, 1966

[5] Naylor, A. W. and Sell, G. R., "Linear Operator Theory in Engineering and Science," (2nd Edition), Springer-Verlag, 1982

[6] Chang, C.C. and Keisler, H.J., "Model Theory," North Holland Publishing Co., 1973

[7] Rich, E., "Artificial Intelligence", New York : McGraw-Hill, 1983

[8] Green, C., "Theorem Proving by Resolution as a basis for Question-Answering Systems" in "Machine Intelligence 4" editted by Meltzer and Michie, 1969

[9] Fikes,R.E. and Nilsson,N.J. "STRIPS : A new approach to the application of Theorem Proving to problem solving," Artificial Intelligence, Vol. 2, 1971

[10] Barr,A. and Feigenbaum,E.A. "The Handbook of Artificial intelligence," (Vol. 1), Kaufman, Los Altos, California, 1981