Report Number: WUCS-84-9

1984-12-01

Formal Specifications of Geographic Data Processing Requirements

Gruia-Catalin Roman

This paper establishes a formal foundation for the specification of Geographic Data Processing (GDP) requirements. The emphasis is placed on modeling data and knowledge requirements rather than processing needs. A subset of first order logic is proposed as the principal means for constructing formalizations of the GDP requirements in a manner that is independent of the data representation. Requirements executability is achieved by selecting a subset of logic compatible with the inference mechanisms available in Prolog. GDP significant concepts such as time, space and accuracy have been added to the formalization without losing Prolog implementabilty or separation of concerns.... Read complete abstract on page 2.

Formal Specifications of Geographic Data Processing Requirements

Gruia-Catalin Roman

Complete Abstract:

This paper establishes a formal foundation for the specification of Geographic Data Processing (GDP) requirements. The emphasis is placed on modeling data and knowledge requirements rather than processing needs. A subset of first order logic is proposed as the principal means for constructing formalizations of the GDP requirements in a manner that is independent of the data representation. Requirements executability is achieved by selecting a subset of logic compatible with the inference mechanisms available in Prolog. GDP significant concepts such as time, space and accuracy have been added to the formalization without losing Prolog implementabilty or separation of concerns. Rules of reasoning about time, space and accuracy (based on positional, temporal and fuzzy logic) may be compactly stated in a subset of second order predicate calculus and may be easily modified to meet the particular needs of specific application. Multiple views of the data and knowledge may coexist in the same formalization. The feasibility of the approach has been established with the aid of a tentative Prolog implementation of the formalism. The implementation also provides the means for graphical rendering of logical information on a high resolution color display.

FORMAL SPECIFICATION OF GEOGRAPHIC DATA PROCESSING REQUIREMENTS

Gruia-Catalin Roman

WUCS-84-9

December 1984

Department of Computer Science
Washington University
St. Louis, Missouri 63130

As appeared in Proceedings of the 2nd International Conference on Data Engineering, February 1986, pp. 434-446. *Outstanding Paper Award*

ABSTRACT

This paper establishes a formal foundation for the specification of Geographic Data Processing (GDP) requirements. The emphasis is placed on modelling data and knowledge requirements rather than processing needs. A subset of first order logic is proposed as the principal means for constructing formalizations of the GDP requirements in a manner that is independent of the data representation. Requirements executability is achieved by selecting a subset of logic compatible with the inference mechanisms available in Prolog. GDP significant concepts such as time, space and accuracy have been added to the formalization without losing Prolog implementability or separation of concerns. Rules of reasoning about time, space and accuracy (based on positional, temporal and fuzzy logic) may be compactly stated in a subset of second order predicate calculus and may be easily modified to meet the particular needs of a specific application. Multiple views of the data and knowledge may coexist in the same formalization. The feasibility of the approach has been established with the aid of a tentative Prolog implementation of the formalism. The implementation also provides the means for graphical rendering of logical information on a high resolution color display.

Acknowledgments: This work was supported in part by Defense Mapping Agency and by Rome Air Development Center under contract F30602-83-K-0065. Howard R. Lykins' effort and enthusiasm in implementing the formalism is also gratefully acknowledged.

TABLE OF CONTENTS

1. INTRODUCTION

Geographic Data Processing (GDP) systems are computer based systems that store and process information traditionally represented in the form of maps [NAGY79]. Within this broad application area, specialization to particular sets of requirements has led to great variability in the nature of the GDP systems. Some, such as the Landsat data bank [ZOBR81], are primarily repositories of image data while others store no images but use maps to render geographicly significant information such as census data (e.g., DIME [SILV77]). The most ambitious undertakings are in the areas of automated cartography and weapon systems support.

The high investment, risk and complexity associated with the development of GDP systems has kept their number relatively small. This, in turn, has resulted in a shortage of development methodologies and tools supporting specificly geographic data processing, especially when compared with what is available today in business data processing. Although it is generally accepted that the development of complex, mission critical, production critical and long-lived systems must start with a validated statement of requirements, the shortage is particularly acute in this area. The lack of any material dealing with GDP requirements in the November 1981 issue of *Computer* which was dedicated to "*pictorial information systems*" is not accidental.

An important component of any requirements specification is data modelling. However, traditional data modelling techniques [TSIC82], by and large, do not provide adequate means for dealing with geographic information. The difficulty stems from the fact that most geographic information must be qualified with respect to the location where it is valid, the time of the observation and its accuracy (i.e., trustworthiness). Furthermore, in the absence of unified theories of time, space and accuracy, different GDP systems (and often different users of the same system) employ different rules of reasoning about time, space and accuracy thus rendering inadequate any model that includes a fixed set of rules. These and other considerations have led us to investigating the use of logic as the formal foundation for a GDP requirements specification and validation methodology.

The approach described in this paper is essentially the same as the one adopted by Minker [MINK78] in his attempt to develop a database which supports logical inference. There are, however, three important differences. First, our formalization focuses specificly on geographic data processing. Second, the need to allow easy formulation of alternate reasoning rules about time, space and accuracy led to the introduction of elements of second order logic. Third, in the context of requirements specification, modularity and separation of concerns gain paramount significance.

Our formalization assumes that a GDP system perceives the real world as a collection of abstract *objects* corresponding to different user views of various geographic entities. Information about an individual object or a group of objects is captured by facts formally defined as first order predicates. The facts fall into two categories: *basic facts* which are simply assumed to be true and *virtual facts* which are defined in terms of other basic and virtual facts. For instance, "*Saint Louis is a large city*" could be a basic fact, when stated as such, or a virtual fact, if established by using a rule that says "*any city whose population exceeds one million is a large city.*"

Most often, a fact asserted about some geographic entity, i.e., object, consists of an attribute and a value from a set of acceptable values. In the fact "*Saint Louis average winter temperature is 45 F,*" for instance, a specific temperature is associated with the attribute average temperature. In the formalization the set of temperature values and operations over them forms a *semantic domain.*

Two semantic domains that are essential to dealing with geographic concepts, *space* and *time*, have been built into the formalism. Without ruling out alternate views of space and time, predefined spatial and temporal operators are provided. Based on positional and temporal logic,

they allow one to state that a fact is true at some point in time and in a particular place. *Accuracy* is another semantic domain discussed in the paper. A fuzzy operator is provided as the means of specifying the degree of trust assigned to individual facts.

The semantic consistency among asserted facts is specified by *constraints* that identify the circumstances under which the formalization is rendered inconsistent. A bridge, for instance, may not be both open and closed at the same instance of time. Semantic consistency, however, is relative to one's viewpoint. This realization led to the introduction of the *model* concept. A model is a grouping of facts and constraints that together define a particular interpretation of the data. As such, the model concept allows for multiple views of data to be held by different users and for data reinterpretation that occurs with the passage of time. The models recognized by a particular user establish his/her *world view.*

Meta-facts and *meta-constraints* play roles similar to their non-meta counterparts but they are expressed using second order logic. This allows them to deal with knowledge that transcends the specifics of individual facts, i.e., to define application specific rules of reasoning. The statements

> *"a fact known to be true at time t0 is still true at some later time t1 if no conflicting fact is known to be true at any time between t0 and t1, inclusively"*

and

> *"a fact may not be both true and false at the same instance of time and at the same physical location"*

illustrate the nature of meta-facts and meta-constraints, respectively. Meta-facts and meta-constraints may be packaged into *meta-models.* The inference rules for positional, temporal and fuzzy logic, for example, may be encapsulated into separate meta-models which may be activated on demand.

An important consideration in defining the formalization has been the desire to implement it in Prolog. To date, large portions of the formalism have been implemented on a VAX 11/750 using Prolog and C under Berkeley 4.2 UNIX operating system. Graphic rendering of logical information is provided by an interface to a Gould/DeAnza IP8500. The purpose of the experiment is to identify ways to overcome the technical difficulties relating to Prolog's computational inefficiency and the use of relatively large volumes of data. This paper, however, is concerned only with the formalism definition and not with its implementation.

The remainder of the paper discusses: the representation of simple factual data; the representation of world knowledge; the definition of general rules of reasoning; spatial, temporal and accuracy qualification of facts; and the ultimate objectives of our research.

2. REPRESENTATION OF SIMPLE FACTUAL DATA

This section introduces the concept of object and gives the definition for basic facts. Together they provide the means by which geographic entities are identified, attributes are attached to these entities and known relations between them are explicitly stated.

Graphic rendering of river basins information overlaid on a simulated aerial photograph.

2.1. Geographic Entities: Objects

The notion of object is a primitive concept that allows an individual or a community to distinguish between different geographic entities whose existence they acknowledge. Each object is uniquely identifiable by an object designator and all the facts referring to the same object use the same object designator. It must be pointed out, however, that the object designator is only a formalization convenience since one is expected to reference objects through the properties they exhibit, e.g., name and position.

2.2. Raw Data: Basic Facts

A basic fact is a property known to be true about some object or an n-ary relation existing between several objects. Some sample facts are listed below:

road(x1), road(x2), road_intersection(x1,x2)

In the proposed formalization only positive facts may be stated. Although logical negation is desirable to have, the inference mechanism of Prolog (our target implementation language) currently disallows its use. Thus, instead of asserting that a bridge is open or not open (i.e., open(x) or ~ open(x)) one has to state that the bridge is open or closed (i.e., open(x) or closed(x)). This creates the possibility for logical contradiction -- one may assert that the bridge is both open and closed since the inference mechanism assumes that the two predicates are independent of each other. As shown later, this problem may be alleviated by adding a constraint that states that the bridge may not be both open and closed. Nonetheless, the absence of logical negation is a nuisance.

3. WORLD KNOWLEDGE REPRESENTATION

Virtual fact, semantic domain, constraint and model are the key concepts introduced in this section. They are the means by which general knowledge about the world, rather than raw data, may be expressed. Explicit specification of the world knowledge, as part of the system requirements, is expected to reduce the occurrence of inconsistencies in the requirements specification, to unify and simplify the specification of the individual functions supported by the system, to improve enhanceability and to lead to a uniform treatment of GDP requirements whether the systems are "knowledge-based" or not.

3.1. Data Abstraction: Virtual Facts

A virtual fact is a fact whose truth value is determined by the truth of other facts, basic or virtual. A virtual fact is asserted by giving its definition. One may state, for instance, that a bridge has a known status if the bridge is known to be either open or closed:

(for-all X): (bridge(X) ^ (open(X) | closed(X)) => known_status(X))

(NOTE: Lower case words are used to denote constants while upper case words are generally used to denote quantified variables.)

Any fact that is not provable is said to be *undefined*, in conformity with what is generally known as the *Open World Assumption*. By contrast, the *Closed World Assumption* states that any fact that is not provable is automatically false. Such an assumption, extensively used in the database area, would be inappropriate for mission critical systems. Moreover, the closed world assumption may be explicitly stated, if necessary.

The definition of a virtual fact assumes the general form

(for-all X1)...(for-all Xn): (F(X1,...,Xm,...,Xn) => q(X1,...,Xm))

where Xi are variables over the set of object designators, F is a formula defined below, X1 through Xn are free variables in F, q is a constant predicate and X1 through Xm are some of the free variables in F. For a more compact notation, however, the general definition could be rewritten as

(for-all Xi): (F(Xi) => q(Xk))

where K = {1,..m} is a subset of I = {1,..m,..n}, i ranges over I and k ranges over K.

Using this compact notation, the formula F may be defined recursively as follows:

F(Xi) ::=

q1(Xi)
    where q1 is a constant predicate;
(F1(Xi1) ^ F2(Xi2))
    where i1 ranges over I1, i2 ranges over I2, and (I1 U I2) = I;
(F1(Xi1) | F2(Xi2))
    where i1 ranges over I1, i2 ranges over I2, and (I1 U I2) = I;
(F1(Xi) ^ (for-all Xj): (F2(Xi2,Xj) => F3(Xi3,Xj)))
    where i2 ranges over I2, i3 ranges over I3,
    (I2 U I3) is a subset of I, j ranges over J, and j is not in I;
(F1(Xi) ^ not(F2(Xi2)))
    where i2 ranges over I2, and I2 is a subset of I;

where i ranges over I, the symbols F1, F2 and F3 are instances of F, and the "not" operator is not

the logical negation but a test that a formula may not be shown to be true. Here again, the restrictions imposed over the definition of F are motivated by the need to achieve implementability of the formalism.

The examples below provide simple illustrations of virtual fact definitions:

A road is open if all bridges on that road are open.
(for-all X): (road(X) ^
(for-all Y): (bridge(Y,X) => open(Y)) => open_road(X))

A bridge that is not open is assumed to be closed.
(for-all X): (bridge(X) ^ not(open(X)) => closed(X))

A bridge that is open or closed has a known status.
(for-all X): (bridge(X) ^ (open(X) | closed(X)) => known_status(X))

3.2. Attribute Values: Semantic Domains

A semantic domain is defined as a set of values and operations over them, i.e., an abstract data type. These values are used to qualify properties of objects but they themselves may not be treated as objects. Semantic domain values do not stand for geographic entities and are not necessarily finite in number.

The value 50 of the semantic domain temperature might be used, for instance, in a fact such as "*the average temperature in Saint Louis is 50 F.*" To state this formally, one needs to extend the scope of the predicates to include both object designators and semantic domain values:

average_temperature(50)(saint_louis)

For clarity sake, the notation distinguishes between the two types of arguments. Semantic domain specific operations that return boolean values (e.g., the characteristic function) may be used in the definition of virtual facts and constraints as if they were facts with the proviso that the value "false" is interpreted as "not provable."

The role of semantic domains in the definition of semantic consistency is discussed next under constraints while ways of reasoning about facts involving certain semantic domains receive extensive coverage under the headings of spatial, temporal and accuracy qualification of facts.

3.3. Semantic Consistency: Constraints

Constraint definitions assume a form very similar to virtual fact definitions

(for-all Xi): (F(Xi) => ERROR(type_of_violation,Xk))

with i in I, k in K, and K subset of I. If the distinguished predicate ERROR takes the value true for some set of arguments the facts are said to be *inconsistent* with respect to the set of constraints being considered.

Constraints may be employed in enforcing a many-sorted logic and general laws of nature and society. In a many-sorted logic, the arguments of each predicate may be restricted to certain predefined domains. (See [MINK78] for a brief overview.) An anomalous fact such as

average_temperature(green)(saint_louis),

for instance, may be flagged by defining the constraint

(for-all X,Y): (average_temperature(X)(Y) ^ not(TEMPERATURE(X))
=> ERROR(improper_temperature,X))

An example of a general law is "*each state has only one capital city*" which may be written as

(for-all X,Y,Z): (capital_of(X,Z) ^ capital_of(Y,Z) ^ (X≠Y)
=> ERROR(two_capital_cities,Z))

Because agreement on general laws is only rarely achievable in the community at large, constraints could be rendered useless unless one acknowledges that they are valid only relative to someone's point of view. This is accomplished by the introduction of the model concept below.

3.4. Knowledge Management: Models

The motivation for adding the concept of model to the formalism rests with the recognition that users have different views of the same data, that data is often reinterpreted because of changes in the application area, that simplifying assumptions about the data are frequently made for the purpose of satisfying particular data processing requirements and, above all, that general knowledge about the world is in a continuous state of flux. To deal with these hard realities of geographic data processing, facts are said to be true with respect to some point of view which is identified by using a model name as a qualifier for the fact:

celsius'freezing_point(0)(x).

This basic fact, for instance, states that the freezing point associated with the object x is 0 degrees, if one assumes the Celsius scale. By definition, any fact or constraint violation that is not explicitly qualified by some model is associated with a default model w.

3.5. Context Specification: World View

Each data processing activity involved in a GDP system operates under a particular set of assumptions. They represent its world view. The world view is defined as a finite set of models. By specifying a particular world view WV, one states that any fact that is true only with respect to models not present in WV is not of interest and, therefore, is assumed to be not provable. Because the same rule is applied to constraints, a constraint violation may occur in one world view but not in the other. A world view that exhibits no constraint violations is called *consistent*.

This concludes the discussion of features based on the first order predicate calculus. The formalization of more abstract knowledge, captured by meta-facts and meta-constraints, takes us to second order predicate calculus.

4. USER SPECIFIED RULES OF REASONING

This section is concerned with the specification and encapsulation of user defined rules of reasoning. This capability is essential because of the need to deal with alternate views of space, time and accuracy required by specific applications within the geographic data processing spectrum. Same capability, however, may be exploited to develop reasoning models for any other application specific semantic domains, e.g, magnetic field.

4.1. Inference Rules: Meta-Facts

Meta-facts provide the means by which one specifies new axioms and rules of inference, usually related to reasoning about a specific semantic domain. They involve higher order knowledge which, as a rule, tends to be independent of particular predicates. Predicate independence is accomplished by employing second order logic. In our formalization, this leads to the use of universally quantified variables that range over the set of defined predicates. In all other respects, the syntax of meta-facts is identical to that of facts. To illustrate a meta-fact definition, a formal statement of the *Closed World Assumption* is given below:

"any fact not known to be true is assumed to be false"
(for-all M,Q,X): (M'Q(X) => M'Q(true)(X))
(for-all M,Q,X): (MODEL(M) ^ PREDICATE(Q) ^ OBJECT(X)
^ not(M'Q(true)(X)) => M'Q(false)(X))

The use of meta-facts in defining rules of inference for reasoning about space, time and accuracy is illustrated in later sections.

4.2. Consistency Rules: Meta-Constraints

The meta-constraints play the same role as the constraints but at the meta-level. Syntacticly, they follow the same format but they are permitted to take advantage of the power available for meta-facts. This is illustrated by the formalization of the statement

"no fact may be both true and false:"
(for-all M,Q,X): (M'Q(true)(X) ^ M'Q(false)(X) => M'ERROR(contradiction,Q,X))

This meta-constraint and the related meta-facts introduced earlier (together referred as meta-rules) may be treated as a group and used only when needed. The way this is done is explained next.

4.3. Reasoning Rules Encapsulation: Meta-Models

One or more semantic domains, their associated operations and pertinent meta-rules may be grouped together to form a meta-model. The separation into meta-models enables the experimentation with different rules of inference without having to change the remainder of the formalization. A potential source of formalization errors is thus eliminated while also increasing the productivity of the requirements specification and validation process.

4.4. Viewpoint Selection: Meta-View

The meta-view consists of all the meta-models in use at one particular point in time. The ability to specify a subset of all the defined meta-models is needed to address specific concerns of the requirements evaluation process and to compare alternate formalizations of the semantic domains. Separation of concerns and incremental building of the requirements formalization are also encouraged.

The remaining sections develop sample meta-models for several important semantic domains.

5. SPATIAL QUALIFICATION OF FACTS

The concept of space is quintessential in geographic data processing. Geographic entities exist in space and each location on the earth surface corresponds to a position in the 3D space occupied by the planet. This section provides one particular formalization of the space concept. It starts by introducing the notion of absolute space as an abstraction of the physical space. The absolute space is used to define the concept of logical space. The latter captures the idea of finite resolution which is more representative for the way geographic data processing is actually carried out. Several spatial operators are defined using a framework similar to the one already established by positional logic. They allow one to state that an individual fact is realized only at particular points in the logical space. The relation between the spatial operators and their use in the definition of certain space dependent concepts is detailed under the last two headings of this section.

5.1. Absolute Space

The absolute space is an abstraction of the coordinate system being used. Each coordinate assumes values from the set of real numbers. In addition to the normal operations over reals, the definition of absolute space also includes a *distance* function and a *direction* function specific to the coordinate system being used, i.e., polar, cartesian, universal transverse Mercator, etc. This presentation, however, is not dependent on any particular coordinate system since changes in the coordinate system definition are expected to affect only the definition of the absolute space and not the rules of reasoning about spatial properties.

5.2. Logical Space and Finite Resolution

The logical space is defined as a discrete subset of an absolute space. A convenient way to specify a logical space is to define a mapping R that reduces patches from the absolute space into single points in the logical space. This function is called the *resolution function*. A proper resolution function partitions the absolute space into intervals in a manner analog to the partitioning of the real line segment below:

[-----p1-----)[-----p2-----)[-----p3-----)[-----p4-----)[-----p5-----)...

Each point pi is also called the representative point for the corresponding interval because all the points in the interval are mapped into pi.

In the remainder of this section, the symbol R is used to denote both the resolution function for an individual logical space and the respective logical space with the intended interpretation being determined by the context. Moreover, a single common absolute space is assumed for all the logical spaces required by the exposition. Under these assumptions, a logical space R2 is said to be a refinement of another logical space R1, (i.e., R2>>R1), if and only if

(for-all P1,P2): (R2(P1)=R2(P2) => R1(P1)=R1(P2))

The figure below is a case in point:

```
[       p1       )[       p2       )[       p3       )...      R1
-------------------------------------------------------------------
[ p11  )[ p12  )[ p21  )[ p22  )[ p31  )[ p32  )...      R2
```

This definition is useful in relating properties stated with respect to different logical spaces, i.e., at different resolutions.

5.3. Spatial Operators

The *Simple Spatial Operator* provides the notational means to specify that a particular property is true at some position in space. The simplest spatial operator originates in work on positional logic [RESC71]. It merely states that *"property q(y) of object list x is true at position p:"*

@p q(y)(x).

(For the sake of clarity, the existence of the argument list over semantic domains, i.e., y, is omitted wherever its presence is superfluous.) The operator '@' is allowed to qualify only facts and not entire formulas; the position may be a universally quantified variable; the space is treated as any other semantic domain; and the resolution function may be applied to the position as in @R(p) q(x).

The semantics of the simple spatial operator are captured by the following meta-facts:

(for-all P,Q,X): @P Q(X) => Q(P)(X)
position may be treated as an additional qualifier

(for-all P,Q,X): Q(X) => @P Q(X)
space independent facts are true at every point in space

To illustrate the use of the spatial operator we provide the definitions for two facts, one basic the other virtual:

@p vegetation(pine)(hill)

(for-all P0,Z0,X): (@P0 elevation(Z0)(X) ^
(for-all P1,Z1): (@P1 elevation(Z1)(X) ^ (dist(P0,P1)<dist0) ^ (Z0>Z1))
=> @P0 elevation_peak(Z0)(X))

Although the resolution function may be used to assure that the specified position is within some required logical space, the simple spatial operator is independent of the concept of logical space. This is not so with its extensions proposed in this section: the area uniform, area sampled and area averaged operators. They are attempts to formalize the nature of the information being maintained when working with finite resolution.

The *Area Uniform Operator* states that a property that is true for some point of a logical space is true for each point of the absolute space mapped into that particular point of the logical space. Formally, this is stated as:

(for-all R,P,P0,Q,X): (@u[R]P Q(X) ^ R(P)=P0 => Q(R,u,P0)(X))
the operator introduces additional qualifications

(for-all R,P0,P,Q,X): (@u[R]P0 Q(X) ^ R(P)=R(P0) => @P Q(X))
the property is true for all points in the area

(for-all R1,R2,P1,P2,Q,X): ((R2 >> R1) ^
@u[R1]P1 Q(X) ^ R1(P2)=R1(P1) => @u[R2]P2 Q(X))
the property is inherited by the higher resolution subareas
of a low resolution area

(for-all R1,R2,P1,Q,X): ((R2 >> R1) ^
(for-all P2): (R1(P2)=R1(P1) => @u[R2]P2 Q(X)) => @u[R1]P1 Q(X))
the property is acquired by a low resolution area if all
its high resolution subareas share the same property

(NOTE: (1) Ri is treated here as a variable that ranges over the set of resolution functions. This is the case any time R is quantified. (2) In the second meta-fact above, because there is an infinity of points P satisfying the condition R(P)=R(P0), any attempt to find them is bound to fail unless the meta-fact is used in a context where the set of values taken by P is finite.)

One situation where the use of the area uniform operator is appropriate occurs when the property of a patch (defined by the resolution function R) is inherited by all points that make up the patch. This is the case with vegetation zones, for instance.

@u[R]P vegetation(ZONE)(land)

The area uniform operator also provides the means by which the underlying absolute space may be replaced by a finite resolution space for applications where this substitution is appropriate, e.g., when a maximum target resolution may be determined. A meta-fact of the form

(for-all P,Q,Y,X): (@P Q(Y)(X) => @u[R]P Q(Y)(X))

is all that is required to accomplish the transition to a finite resolution view of the world. For semantic domains where each point may be assigned a unique value, the uniqueness is guaranteed by the meta-model where the domain is defined and no additional meta-constraint is needed here.

The *Area Sampled Operator* may be employed whenever one needs to state that there is at least one point in the area having a certain property but the actual point is not necessarily known. The meta-facts defining this operator are the following:

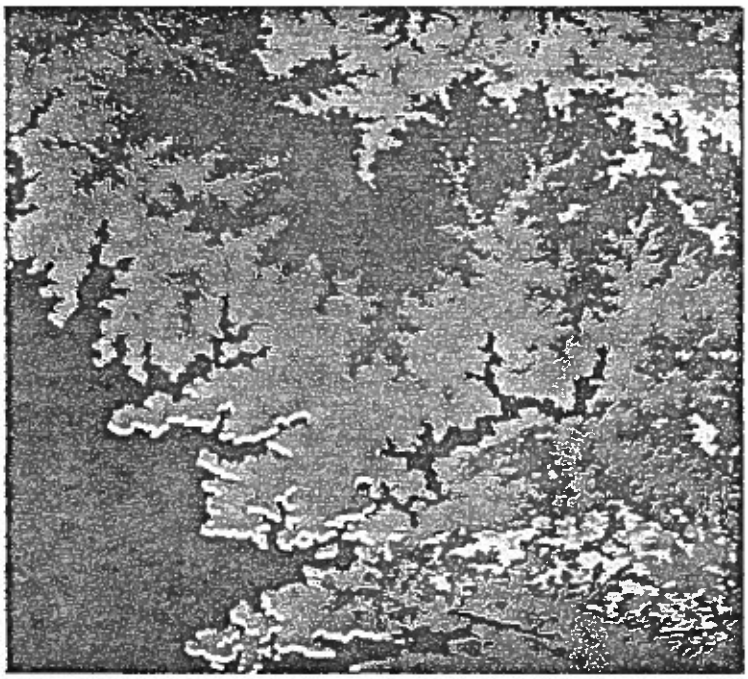
(for-all R,P,P0,Q,X): (@s[R]P Q(X) ^ R(P)=P0 => Q(R,s,P0)(X))
the operator introduces additional qualifications

(for-all R,P,Q,X): (@P Q(X) => @s[R]P Q(X))
the area acquires the sample if any point in the area has the property

(for-all R1,R2,P1,P2,Q,X): ((R2 >> R1) ^
@s[R2]P2 Q(X) ^ R1(P2)=R1(P1) => @s[R1]P1 Q(X))
the area acquires the sample if any subarea has it

The area sampled operator is essential for dealing with map making where, for instance, a road may still have to be drawn even when its actual thickness is much less then the map resolution:

(for-all P) (@P road(x) => @s[R]P road(x))



Graphic rendering of vegetation zones.
@u[R]P vegetation(ZONE)(land)

The *Area Average Operator* allows the association of a value to a point in the logical space with the understanding that the respective value characterizes not the point but the area it represents. When working with finite resolutions, one deals with average values rather than precise measurements. The semantics of the area average operator are captured by the following meta-facts:

(for-all R,P,P0,Q,Y,X): (@a[R]P Q(Y)(X) ^ R(P)=P0 => Q(R,a,P0,Y)(X))
the operator introduces additional qualifications

(for-all R1,R2,P1,Q,Y0,X): ((R2 >> R1) ^
(Y0 = average("Y","R1(P2)=P1 ^ @u[R2]P2 Q(Y)(X)"))
=> @a[R1]P1 Q(Y0)(X))
the average may be computed if values are known for each subarea

(for-all R1,R2,P1,Q,Y0,X): ((R2 >> R1) ^
(Y0 = average("Y","R1(P2)=P1 ^ @a[R2]P2 Q(Y)(X)"))
=> @a[R1]P1 Q(Y0)(X))
the average may be computed if averages are known for each subarea

where the function called *average* determines which points in the logical space R2 are mapped into the point P1, by the resolution function R1, and for these points computes the average value of any of the arguments of the fact Q for a particular object or group of objects. The average is weighted if the resolution function partitions the space in unequal areas. (We assume this not to be the case.)

The concept of average makes sense only when the values of the semantic domain involved in the averaging computation are of numeric nature and each point may have associated with it a

Graphic rendering of "invisible" road and average elevation
@s[R]P road(x) and @a[R]P elevation(Z)(land)

single value. Such is the case, for instance, when one considers elevation

@a[R]P elevation(Z)(land)

where P is given in terms of latitude and longitude.

The four spatial operators introduced in this section provide the ability to specify that a fact is true at a specific point, at some point of a particular area, over all points of a particular area and on the average over some area. They also establish a framework which is instrumental in defining a variety of concepts that reflect spatial properties of geographic entities. This use of the spatial operators is illustrated next.

5.4. Formalization of Spatial Properties

This subsection provides examples of how spatial operators may be used to define formally geometric properties of individual objects, spatial relations between objects and the loss of information that takes place when moving to increasingly lower levels of resolution. The ability to formally define these properties is an indicator of the spatial operators' power and usefulness and leads to the development of very powerful primitives that ought to simplify the formalization task.

Geometric properties, as defined here, include type of geographic feature (point, line, area, volume), general topology (e.g., single or multiple points), dimensionality (2D or 3D), geometric constraints (e.g., multiple points on a line, on an area, or in some volume), shape (e.g., square) and size (with respect to some metric). The simplest illustration is given by the definition of a point type feature:

(for-all X,Q1,P1): (@P1 Q1(X) ^ not(Q1(X)) ^
((for-all P2,Q2): (@P2 Q2(X) ^ not(Q2(X)) => P1=P2)) => point_type(X))

Informally, this definition states that all position dependent properties of the object are true at a single point in space.

Spatial relations between objects cover concepts such as relative position, relative orientation, relative size, adjacency (usually, at some given resolution) and overlap. The definition of overlap, for instance, takes the form:

(for-all X1,X,X2,Y1,Y,Y2,P,Qa,Qb): (@P Qa(X1,X,X2) ^ @P Qb(Y1,Y,Y2) ^
not(Qa(X1,X,X2)) ^ not(Qb(Y1,Y,Y2)) => overlap(X,Y))

(NOTE: X1, X2, Y1 and Y2 represent arbitrary lists of object designators.) Since facts formulated in a space independent manner are true at every point in space, they are excluded from consideration. Otherwise, the concept of overlap would become meaningless.

It is generally accepted that a high resolution map provides more information than a low resolution map covering the same area. When the map generation is automated there is the need to specify the nature of the information loss incurred in the process of interpreting the data with regard to a lower resolution than originally formulated. The rules governing this process are a special case of abstraction rules. Four types of abstraction rules have been identified as particularly useful in this context: copying, thresholding, averaging and composition. They refer to the operations applied to properties of the original object in order to determine the properties of the same object when considered at a lower resolution.

Copying rules state the conditions under which a high resolution point passes on its properties to the low resolution point into which it is mapped. The type of property and the size of the object are two possible considerations. Thresholding rules state the conditions under which properties of the high resolution points are ignored during the transition to the lower resolution. A combined copying/thresholding rule may be applied, for instance, to determine the presence of some island on the map:

(for-all R1,R2,P,X): ((R2>>R1) ^
@R2(P) island(X) ^ (size(X,R2) > delta) => @R1(P) island(X))

where R1 is the logical space having the lower resolution and "size" is a function that determines the number of points covered by some object at a specified resolution.

Averaging rules state the conditions under which the average of the semantic domain values associated with some property of high resolution points that map into the same low resolution point is assigned to the latter. The definition of the average operator, @a[R], shows one example of how this rule works.

Finally, composition rules are used to generate new properties for the low resolution point based on the properties of the high resolution points that map into it. One example is the determination of the shore line through the use of the following rule:

(for-all R1,R2,P1,P2,X): (R1(P1)=R1(P2) ^
@R2(P1) lake(X) ^ @R2(P2) shore(X) => @R1(P1) shore_line(X))

where R1 and R2 are defined as before.

This concludes the discussion of spatial qualification of facts. Next section, covering temporal qualification of facts, shows how some of the concepts and operators introduced so far are also applicable to reasoning about time, with some minor reformulation.

6. TEMPORAL QUALIFICATION OF FACTS

The contents of this section parallels in intent recent data modelling efforts concerned with the formalization of temporal concepts used by database designers. However, while Clifford and Warren [CLIF83], for instance, attempt to extend the relational model to include historical relations, we start with a logic-based formalization and thus we are able to assimilate directly some of the work on temporal logic [RESC71], mostly through the adaptation of the spatial operators introduced previously. This approach is possible because, as noted by others, temporal logic may be seen as a special case of the positional logic.

In addition to treating time as a uni-dimensional space, this section also considers the issue of reasoning about arbitrary time intervals and proposes an appropriate set of temporal operators.

6.1. Time as a Uni-Dimensional Space

If time is treated as a uni-dimensional space, the absolute time is reduced to the real line and logical time, in turn, is introduced with the help of the same resolution function R. (Potential confusion between logical time and space is avoided by context.) All the spatial operators have temporal counterparts:

&t q(x)	*Simple Temporal Operator*
&u[R]t q(x)	*Interval Uniform Temporal Operator*
&a[R]t q(x)	*Interval Average Temporal Operator*
&s[R]t q(x)	*Interval Sampled Temporal Operator*

which play identical roles.

The concept of temporal resolution, however, is not as important as its spatial counterpart. More often than not, reasoning about time involves dealing with arbitrary time intervals during which one property or another holds true for some object. The interval uniform operator, as defined so far, while able to capture this notion, is not convenient to use – a separate resolution function would have to be defined for each interval. One way to overcome this impediment is explored next.

6.2. Time Intervals

This subsection extends the scope of the interval uniform operator, introduces a definition for the concept of *now*, and discusses two models useful in reasoning about time. The proposed extension allows one to suply an interval definition in place of the resolution function as in the following examples (t2 > t1):

&u[t1,t2] q(x) &u(t1,t2] q(x) &u[t1,t2) q(x) &u(t1,t2) q(x)

Because of the similarity between their definitions, only the definition for the closed interval case is provided below.

(for-all T,T1,T2,Q,X): (&u[T1,T2] Q(X) ^ (T1 $\leq$ T $\leq$ T2) => &T Q(X))

The interval uniform operator was also extended to deal with cyclic phenomena but this extension is not be discussed in this paper.

Illustrations of the interval uniform operator are provided by the formulation of two models important in reasoning about time. The first one, called in [CLIF83] the *Comprehension Principle*, is a variation on the closed world assumption. It says that *"although some fact may not be uniformly true over some interval of interest, it is often expedient to assume that it is:"*

(for-all T,Q,X): (&T Q(X) ^ (t1 $\leq$ T $\leq$ t2) => &u[t1,t2] Q(X))

The second model, also discussed in [CLIF83], is the *Continuity Assumption.* It applies to cases when only one value of some semantic domain may qualify any given object at any one moment in time and allows one to *"assume that a fact holds true as long as no conflicting fact has been asserted."*

(for-all T1,T2,Q,Y1,Y2,Y,X): (&T1 Q(Y1)(X) ^ &T2 Q(Y2)(X) ^
(for-all T): ((T1 < T < T2) ^ not(&T Q(Y)(X))) => &u[T1,T2) m'Q(Y1)(X))

One complication brought about by dealing with time is the need to consider the concept of present moment. Staticly, the present moment is a unique point in time separating past from future. In this context, it is important to be able to tell if some arbitrary point in time is part of the past, present or future. Three functions bearing these respective names could be provided for this purpose:

past(1971) -- is provable, the year is 1984;
present(1971) -- is not provable;
future(1971) -- is not provable;

They are not adequate, however, for dealing with the dynamics of time, i.e., the present becoming past while the future is becoming present.

A special place holder, call it *now,* must to be introduced in order to express facts whose truth changes as the present moves into the future. In the simplest case, one must be able to state that some fact is always true in the present as in

&now q(x)

whose semantics is given by

(for-all T,Q,X): (&now Q(X) ^ present(T) => &T Q(X)).

Similarly, one may allow *now* to appear as interval boundaries. The use of unevaluated expressions becomes necessary, however, if one wants to go so far as to permit intervals such as */now-5,now+5/.*

The discussion on temporal qualification of facts ends here. Although treated independently, temporal operators may be used in conjunction with the spatial operators defined so far and other operators introduced in sections to come. Operator commutativity may be assumed at all times.

7. ACCURACY QUALIFICATION OF FACTS

The world about which GDP systems maintain information is full of uncertainties and the means by which the information is gathered are imperfect. Consequently, much of the information a GDP system provides to its users ought to be qualified in a manner that indicates the extent to which the information may be viewed as accurate. If this is not done, decisions taken under the assumption that the information is absolutely true may have disastrous consequences.

To qualify the accuracy of the information maintained by a GDP system presupposes the ability to specify the uncertainly level of some facts and the ability to evaluate the impact of logical inference on the accuracy of facts derived from them. This section shows how fuzzy logic may be used to accomplish these two goals. The presentation starts with a brief review of fuzzy logic. It is followed by the definition of a fuzzy operator and illustrations of how the operator allows one to specify uncertainty originating from several sources. The presentation then turns to a brief

discussion of fuzzy constraints. The section concludes with the definition of a simple fuzzy inference model used for uncertainty level derivation in virtual facts.

7.1. Fuzzy Logic

In two-valued logic the truth value of a formula may be 1 or 0, i.e., true or false. Fuzzy logic [LEE72], however, allows the truth value of a formula to take any value in the closed interval [0, 1]. The rules by which a truth value is assigned to a formula are modified accordingly.

The table below summarizes the truth value assignments corresponding to one of the most widely used rules, the min-max rule.

TRUTH(F) =

```
TRUTH(q)
        -- if F is the atomic formula 'q'
1 - TRUTH(F1)
        -- if F is "~ F1"
min(TRUTH(F1),TRUTH(F2))
        -- if F is "F1 ^ F2"
max(TRUTH(F1),TRUTH(F2))
        -- if F is "F1 | F2"
inf{TRUTH(F1(X)) for X in D}
        -- if F is "(for-all X): (F1(X))" with X in D
sup{TRUTH(F1(X)) for X in D}
        -- if F is "(there-exist X): (F1(X))" with X in D
```

Using the min-max rule above, the sentence

flooded(plain) ^ frozen(plain)

is assigned (1) the truth value 0.45 when flooded(plain) has the truth value 0.45 and frozen(plain) has the truth value 0.65 and (2) the truth value 0.00 (i.e., false) when flooded(plain) is false and frozen(plain) is true.

The min-max rule is not the only rule that may be used in fuzzy logic and, like all the others, is limited in the extent to which it is able to combine symbolic logic and probability theory. In particular, it ignores possible logical dependencies between facts. Nevertheless, fuzzy logic provides an elegant and often intuitive way of assigning a measure of accuracy to information. The compatibility with two-valued logic (which may be seen as a special case of fuzzy logic) and with its inference mechanisms is also very attractive.

7.2. Uncertainty Level Specification

In general, there is no objective way of assigning an accuracy level to an arbitrary fact. While general models may be used to establish the propagation of the accuracy through the inference process, an initial set of accuracies must be provided by the user. The accuracy supplied by the user may vary in the degree of subjectivity depending on the uncertainty source affecting a particular fact. There are several important sources of uncertainty: user confidence, measurement error, extrapolation and statistical sampling.

Before illustrating the way to specify the uncertainty level originating with each of the sources, we will introduce a logical operator called the *Simple Fuzzy Operator*. Its syntax takes the form "%a q(x)" where "a" is an accuracy value in the closed interval [0,1], zero is interpreted

as absolutely false, one is interpreted as absolutely true and the values in between correspond to degrees of truth. The formal semantic definition is captured by the meta-fact

(for-all A,Q,X): (%A Q(X) ^ (0 ≤ A ≤ 1) => Q(A,X)

which states that accuracy is just an other qualification of a fact.

For reasons dealing with the separation between accuracy and other concerns, the fuzzy operator may not qualify a basic fact. Instead, a separate virtual fact must be specified as in the definition

q(x) => %a q(x)

This will allow one to use different models of accuracy or to ignore accuracy all together.

The fuzzy operator allows the user to define rules by which the accuracy of certain classes of facts is determined. When accuracy is an expression of the user's trust in the data or in the measuring device, it may be derived from the fact in question. In other words, the accuracy becomes a function of the predicate, semantic domain values and the objects involved:

(for-all Q,Y,X,A): (Q(Y)(X) ^ (A = f(Q,Y,X)) => %A Q(Y)(X)).

The function f may be viewed, in some cases, as a model for estimating the precision of a measurement or sensing device and, in other cases, as a human judgment based on expertise that is not quantifiable. Any changes in the way accuracy is computed are limited to the redefinition of the function f.

Even when measurements are precise, uncertainty may still be introduced when extrapolations are made to fill in gaps in the data. Consider, for instance, a geological survey or an ocean depth study. In both cases, a limited set of points are sampled and the value attached to the points in between is computed using some mathematical formula. Given the frequency of the samples and some knowledge of the application, the extent to which the computed values differ from the real world may be determined and used as an accuracy estimate. The formula below is representative of the general form such accuracy definitions take:

```
(for-all A,P,P1,P2,Z,Z1,Z2): (@P1 depth(Z1)(ocean) ^ @P2 depth(Z2)(ocean) ^
        nearest_samples(P1,P,P2) ^ ((A,Z) = f(P,P1,P2,Z1,Z2))
                                    => %A @P depth(Z)(ocean))
```

where f is the depth interpolation function.

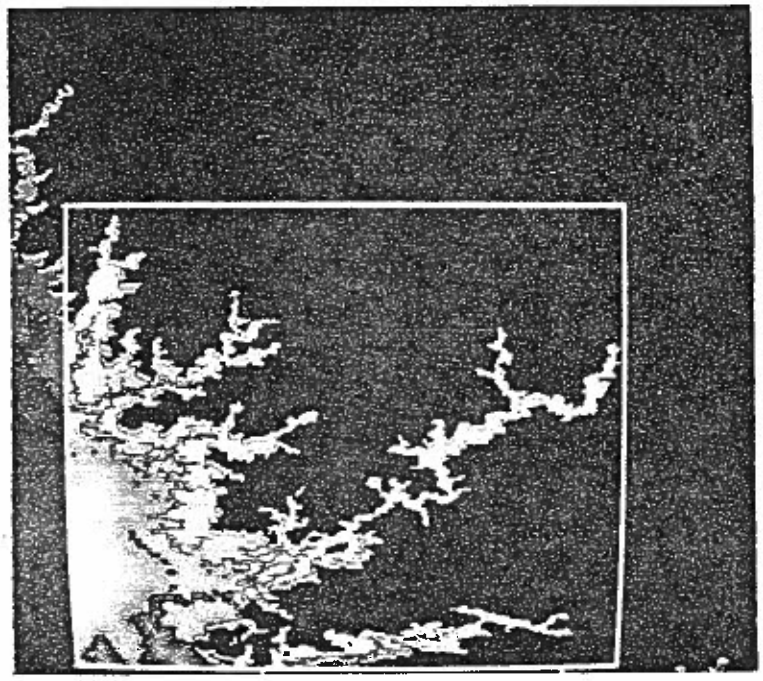
If the user wishes to define accuracy as a statistical property of some group of sample facts (e.g., picture clarity may be expressed as one minus the percentage of cloud cover), any formalism based on pure logic fails. The reason is the inability to count the number of provable instantiations of some given formula. There is no way, for instance, to get a count of the number of pixels P that are white, i.e., render provable the formula "@P white(image)" where P is a free variable. To do this one needs to go outside pure logic and introduce a primitive such as

card("F(Xi)")

where F is a formula having the free variables Xi and *card* (read "cardinality") is a function that returns a count of the distinct instances of F(Xi) that are provable, if the number of instances is finite. Using *card*, picture clarity could be defined by the formula:

```
(for-all A): ((n  = card("@P white(image)")) ^
        (n0 = card("@P any_color(image)")) ^
                                (A = 1 - n/n0) => %A clarity(image))
```

Same paradigm may be actually used for any statistically defined accuracy.



Graphic rendering of ocean depth accuracy
%A @P depth(Z)(ocean)

We turn our attention next to some of the pragmatics of uncertainty level specification. There are four important points to consider. First, in those situations when accuracy qualifications are not relevant one should not have to contend with potential logical consequences of their unwanted presence. Second, it is possible for the same fact to be qualified in two different ways thus raising the issue of selecting the "right" accuracy. Third, constraints may also involve accuracy. Forth, the lack of a general accuracy model applicable to all facts suggests that alternate models are needed when the same data is used in different contexts.

7.3. Ignoring Accuracy Qualifications

There are two ways of ignoring accuracy qualifications: the user is interested only in the facts that are absolutely true or the user choses to view as true any facts whose accuracy exceeds certain threshold. In the first case, any definition supplied by the user simply ignores the presence of the fuzzy operator. As a consequence, since a formula such as q(x) is not provable from facts of the form %a q(x), regardless of the values taken by a, all the facts for which an accuracy is specified are automatically ignored. In the second case, the user must suply a meta-model defining the threshold rule as in

(for-all A,Q,X) : (%A Q(X) ^ (A > 0.80) => m'Q(X)).

A model must be specified in order to separate the facts of interest from all the other facts. Otherwise, the rule above may have absolutely no effect since, usually, each fact for which an accuracy is specified also exists without any accuracy.

7.4. Dealing With Conflicting Accuracies

Because it is possible for several uncertainty level definitions to qualify the same fact as having several different accuracies, the *Unified Fuzzy Operator* is introduced as a way of resolving such conflicts. This operator is restricted to appearing only to the left hand side of fact definitions and provides a way of referring to the highest accuracy assigned to some fact. For instance, to state that a fact is considered true whenever there is at least one accuracy qualification of this fact exceeding the value 0.75 one may write

(for-all A,Q,X) : (%[A] Q(X) ^ (A > 0.75) => m'Q(X)).

(NOTE: other definitions of the unified fuzzy operator, e.g., minimum or average value, may be needed for specific types of facts.)

7.5. Fuzzy Constraints

Any constraint that explicitly involves a fuzzy operator is called a fuzzy constraint. Two special cases are particularly relevant. The first one is when the error is not qualified by an accuracy but is triggered by the accuracy of some other fact as in the definition

(for-all A,X): (%A clarity(X) ^ (A < 0.80) => ERROR(bad_image,X)).

The second case is when some accuracy is actually associated with the error. It might be important, for instance, to know the percentage of river crossings where no bridge appears to be present:

%A ERROR(missing_bridge)

A high accuracy value associated with this error may indicate possible problems with the data being processed.

7.6. Uncertainty Level Propagation via Logical Inference

In a previous section we have shown how an uncertainty level may be assigned to certain classes of facts. These facts in turn may be used to derive new virtual facts whose uncertainty level is dependent upon the accuracy of the original facts. The question addressed in this section is how to assign automatically an accuracy to facts derived from accuracy qualified facts.

We assume that each fact has a definition which does not involve any references to the accuracies of the facts involved. (We have adopted this approach earlier in order to separate the accuracy issues from the other issues and in order to permit easy substitution of one accuracy model for another.) Let us also assume the existence of a function AC which computes an accuracy for valid formulas. Since virtual fact definitions take the form

(for-all Xi): (F(Xi) => q(Xk))
where i ranges over I, k ranges over K and K is a subset of I,

the accuracy for q(xk) (i.e., an instance of q(Xk)) is given by AC(F(xi)). This may be stated generally as

(for-all Xi): (F(Xi) ^ (A = AC(F(Xi))) => %A q(Xk))

(NOTE: These types of formulas may be generated mechanically.)

The definition of AC follows the basic rules of fuzzy logic introduced earlier

AC(F(xi)) =

a
-- when F(xi) is "q1(xi)," and %[a] q1(xi) is provable
failure
-- when F(xi) is "q1(xi)," and %[a] q1(xi) is not provable
min(AC(F1(xi1)),AC(F2(xi2)))
-- when F(xi) is "(F1(xi1) ^ F2(xi2))"
max(AC(F1(xi1)),AC(F2(xi2)))
-- when F(xi) is "(F1(xi1) | F2(xi2))"
min(AC(F1(xi)), inf{max(1-AC(F2(xi2,Xj)), AC(F3(xi3,Xj))) for all Xj})
-- when F(xi) is "(F1(xi) ^ (for-all Xj): (F2(xi2,Xj) => F3(xi3,Xj)))"
min(AC(F1(xi)),1)
-- when F(xi) is "(F1(xi) ^ not(F2(xi2)))" and F2(xi2) is not provable
failure
-- when F(xi) is "(F1(xi) ^ not(F2(xi2)))" and F2(xi2) is provable

(NOTE: By supplying a different definition for AC, one in effect changes the rules of reasoning for accuracy. The definition above is simply a default.)

Because of the reliance on fuzzy logic the approach described here inherits many of its limitations. The most important one is the inability to handle dependencies between facts. The conservative manner in which accuracies are computed does guarantee, however, that no fact will be given an accuracy greater than the one that would result from considering fact dependencies. Furthermore, if the only two accuracies used are 0 (false) and 1 (true) the results are consistent with the two-valued logic.

8. CONCLUSIONS

Due to the extreme requirements placed on the GDP systems needed today (size, data volume, production throughput, etc.) and due to limitations in the current state-of-the-art, these systems are expensive and difficult to design. Our ultimate goal is to reduce significantly the risk associated with building GDP systems through the development of inexpensive rapid prototypes. Our strategy is to build a realistic simulations of proposed system, or parts there of, and to record intermediary results of the simulation for the purpose of an interactive real time play-back of the simulation under user control. During play-back, built-in adjustable delays reproduce the performance characteristics of the technical solution under consideration. In this way both the functional adequacy and the potential production impact may be clearly identified prior to committing to the actual purchase or development of the system. The simulations may be used both as a means for technical evaluation by experts or production personnel and as an avenue for building and evaluating training materials for new production systems.

The work described in this paper is the first step toward developing a Geographic Data Processing Simulation and Evaluation Facility: *the establishment of a formal foundation for GDP requirements specification.* The emphasis is placed on modelling data and knowledge requirements rather than processing needs. A subset of first order logic is proposed as the principal means for constructing formalizations of the GDP requirements in a manner that is independent of the data representation. Requirements executability is achieved by selecting a subset of logic compatible with the inference mechanisms available in Prolog. GDP significant concepts such as time, space and accuracy have been added to the model without losing Prolog implementability. Rules of reasoning about time, space, accuracy and other application specific concepts may be compactly stated in second order predicate calculus and may be easily modified to meet the particular needs of a specific application. Multiple views of the data and knowledge may coexist in the same formalization.

The feasibility of the approach has been established with the aid of a tentative implementation of the formalism using Prolog and C. Key components of the facility and the requirements specification and validation methodology are currently under development.

9. REFERENCES

[CLIF83] Clifford, J. and Warren, S., "Formal Semantics for Time in Databases," *ACM Trans. on Database Systems* 8, No. 2, pp. 214-254, June 1983.

[LEE72] Lee, R. C. T., "Fuzzy Logic and the Resolution Principle," *JACM* 19, No. 1, pp. 109-119, January 1972.

[MINK78] Minker, J., "An Experimental Relational Data Base System Based on Logic," pp. 107-147, *Logic and Data Bases,* H. Gallaire and J. Minker editors, Plenum Press, 1978.

[NAGY79] Nagy, G. and Wagle, S., "Geographic Data Processing," *Computing Surveys* 11, No. 1, pp. 139-181, June 1979.

[RESC71] Rescher, N. and Urquhart, A., *Temporal Logic,* Springer-Verlag, 1971.

[SILV77] Silver, J., "The GBF/DIME System: development, design and use," U.S. Bureau of the Census, Washington, D.C., 1977.

[TSIC82] D. C. Tsichritzis and F. H. Lochovsky, *Data Models,* Prentice-Hall, Inc. 1982.

[ZOBR81] Zobrist, A. L. and Nagy, G., "Pictorial Information Processing of Landsat Data for Geographic Analysis," *Computer* 14, No. 11, pp. 34-41, November 1981.