Logic Programming and Metadata Specifications

Antonio M. Lopez, Jr., Ph.D.*
Mathematical Sciences
Loyola University Box 51
New Orleans, LA 70118
(504) 865-3340
e-mail: lopez@loynovm.bitnet

Marguerite E. Saacks, Ph.D.*
Computer Science Department
Xavier University
New Orleans, LA 70125
(504) 483-7456
e-mail: saacks@comus.cs.tulane.edu

Abstract

Artificial intelligence (AI) ideas and techniques are critical to the development of intelligent information systems that will be used to collect, manipulate, and retrieve the vast amounts of space data produced by "Missions to Planet Earth." Natural language processing, inference, and expert systems are at the core of this space application of AI. This paper presents logic programming as an AI tool that can support inference (the ability to draw conclusions from a set of complicated and interrelated facts). It reports on the use of logic programming in the study of metadata specifications for a small problem domain of airborne sensors, and the dataset characteristics and pointers that are needed for data access.

Introduction

The National Aeronautics and Space Administration (NASA) is on the verge of a tremendous data explosion. By the end of this decade, the Earth Observing System (EOS), just one of NASA's projects, is expected to produce several terabytes of archival data each week. These data will be in a variety of formats and will "belong to" a variety of Earth and Science disciplines.

Although mass storage device technology, which makes megabyte data files practical and affordable, is keeping pace with current industrial and business demands, new innovative software systems will be required to organize, link, maintain, and properly archive the EOS data that is to be collected for the EOS Data and Information System (EOSDIS) (Dozier, 1990). Software problems associated with organizing, structuring, and managing these very large multi-format data files for efficient and timely access and update are being addressed. Artificial intelligence tools, techniques, and concepts offer great potential in solving many of the software problems that have already surfaced.

The Intelligent Data Management (IDM) project team at NASA's Goddard Space Flight Center (GSFC) is developing a prototype system for managing the terabytes of satellite imagery data that EOS is expected to produce. The research and development incorporates a number of

---

* This work was begun when the authors were 1991 NASA/ASEE Summer Faculty Researchers in the Information Systems Division of the Science and Technology Laboratory at the John C. Stennis Space Center in Mississippi.

state-of-the-art AI software methodologies in an effort to provide new insights and tools for building future intelligent information systems (Campbell and Cromp, 1990). Published works by members of the IDM project team discuss a high-level expert system for declarative and procedural knowledge acquisition (Cromp, 1988), an intelligent user interface for browsing satellite data catalogs (Cromp and Crook, 1989), the application of connectionism to query planning and scheduling (Short and Shastri, 1990), and an architecture for a large object-oriented database (Dorfman, 1991). At the heart of the work that is being done at GSFC is the Intelligent Information Fusion (IIF) concept, a structured approach to implementing the management and access to data, metadata (useful information about the data), and supporting information and knowledge (Roelofs and Campbell, 1990). An essential element of IIF is the semantic and knowledge-based representation that captures the essence of the data domain at all levels of knowledge representation, from the highest class structure, to the intermediate metadata, to the lowest level of data granule. The overall concept of implementing an Intelligent Information Fusion System (IIFS) for spatial data management has been described by Campbell et al. (1990).

An AI Tool

Logic programming is an outgrowth of the research that was done in the mid-1960's on automated inferencing and theorem proving. A logic program is constructed by describing what is true in a particular problem domain. It is equivalent to a set of logical axioms. These axioms are facts and rules that describe objects and the logical relationships between them. The execution of a logic program is equivalent to a constructive proof of a goal statement from the axioms, and it is carried out by an application-independent inference procedure (Genesereth and Ginsberg, 1985) embedded within the particular programming language implementation.

Logic programming provides an efficient mechanism to integrate data, metadata, and control into a domain-specific knowledge environment (Kerschberg, 1990). Recently, logic programming languages such as LDL (Naqvi and Tsur, 1989) and LOGIN (Ait-Kaci et al., 1990) have been designed for efficient access to very large collections of data and for using concepts such as inheritance. Both of these languages are extensions of PROLOG (PROgramming in LOGic), the flagship of logic programming languages.

PROLOG has been shown to be a useful AI tool in a wide range of applications such as expert systems (Moller-Jensen, 1990), relational databases (Lucas, 1988), knowledge representations (Goyal, 1989), and natural language processing (Tanaka, 1988). More recently, attention has been drawn to PROLOG as a specification language (Denney, 1991), and for use in declarative testing and debugging (Yan, 1991). It is these two aspects of this logic programming language that we intend to exploit in metadata specification.

The Theory

Metadata provides systems such as EOSDIS and IIFS with a knowledge model that captures the data semantics (i.e., objects, properties, etc.) and the knowledge semantics (i.e., heuristics, scripts, etc.) of a particular domain. The truly creative and most difficult step in the development of metadata is the construction of an acceptable formalism from an intuitive understanding of the data domain, using design tools such as semantic networks, frame-based representations, and object-orientations (Cercone and McCalla, 1987). In specifying the metadata, the intelligent information system developers claim to know: (1) what knowledge to represent in an application, and (2) how to reason with that knowledge. Regardless of the tool used to specify the metadata, the question is, "Does the metadata provide an accurate knowledge model?"

In education theory, the deductive model of inquiry treats theories as: (1) a set of basic facts

and principles, and (2) a deductive logic that allows explanations and predictions to be derived. McEneaney (1990) has shown that there is a clear connection between theories in the deductive model of inquiry and logic programming. Logic programs can be used not only to test the validity of theoretical arguments, but also to make substantive contributions to theory development and revision. Logic programs can also be used to develop a metadata specification, because metadata is a theory about the relationships that exist among the data.

PROLOG can be used as an AI tool to construct and test metadata specifications given in terms of a semantic network, a frame-based representation, or an object-orientation. Specifications written in PROLOG are executable. Because PROLOG makes no distinction between data and program, it is a powerful tool for simulating the learning needed in intelligent information systems. In addition, this approach allows the developer to "ask questions" of the metadata, derive answers, and change the metadata if the answers are unacceptable. Through an iterative process of generate and test, the metadata specifications and the PROLOG program must eventually produce an accurate knowledge model.

The Application

NASA's John C. Stennis Space Center (SSC) has over the years collected data obtained by using the Thermal Infrared Multispectral Scanner (TIMS), and the Calibrated Airborne Multispectral Scanner (CAMS). The analog tapes produced by these sensors on different missions are stored in SSC's data holdings and digitized for Earth Scientists. In anticipation of EOSDIS and IIFS, the Information Systems Division at SSC was interested in developing metadata specifications for their TIMS and CAMS data sets.

An initial investigation revealed two important points. First, the information that was stored on the tape headers was not enough to support the types of queries that scientists in the Earth Science Division would want to make. For example, scientists suggested queries that needed information found on the Mission Flight Request Form, a five page document with possible attachments. The Mission Flight Request Form is not stored electronically with the data that was obtained by the mission. Second, people's understanding of metadata varied. Some proposed tables that could be implemented using a relational database management system; others produced the *NSSDC Directory Interchange Format Manual (Version 3.0, December 1990)* and indicated that a directory entry consists of collections of "metadata" fields; and yet others knew the purpose of metadata, but found it difficult to specify.

We decided to view metadata as a theory about the underlying data sets. Given facts and principles, the metadata would be used to "predict" the need for a particular data set. Viewed in this way, metadata would be analogous to a theory in the deductive model of inquiry, and it would be reasonable to build the theory as a logic program that would be analyzed, tested, and revised.

Two of the most successful approaches to building knowledge representation systems have been semantic networks and frames. One advantage of semantic networks is the simplicity with which logic can be used to answer questions. Frames have proven invaluable in organizing large numbers of facts. Both of these knowledge representation approaches were used to specify the requirements for the TIMS and CAMS metadata (Saacks and Lopez, 1992). Metadata was organized as a semantic network using explicit relationships between objects. Complex objects were represented as a single frame instead of a larger network.

An abbreviated portion of the semantic network developed to specify the metadata for the TIMS and CAMS data holdings at SSC is given in Figure 1. The data set pointer objects (**ssc100, ssc110, ..., ssc180**) are stacked and associated with the TIMS or CAMS sensor

that created it. This is done only for the convenience of presentation. Figure 1 shows that the **sensor** object inherits from both the **tool** object and the **platform** object. This is an instance of multiple inheritance. Similarly, the data set pointer objects inherit from the **flight** object, and from either the **tims** or **cams** object. What Figure 1 does not show is the complexity of each object.

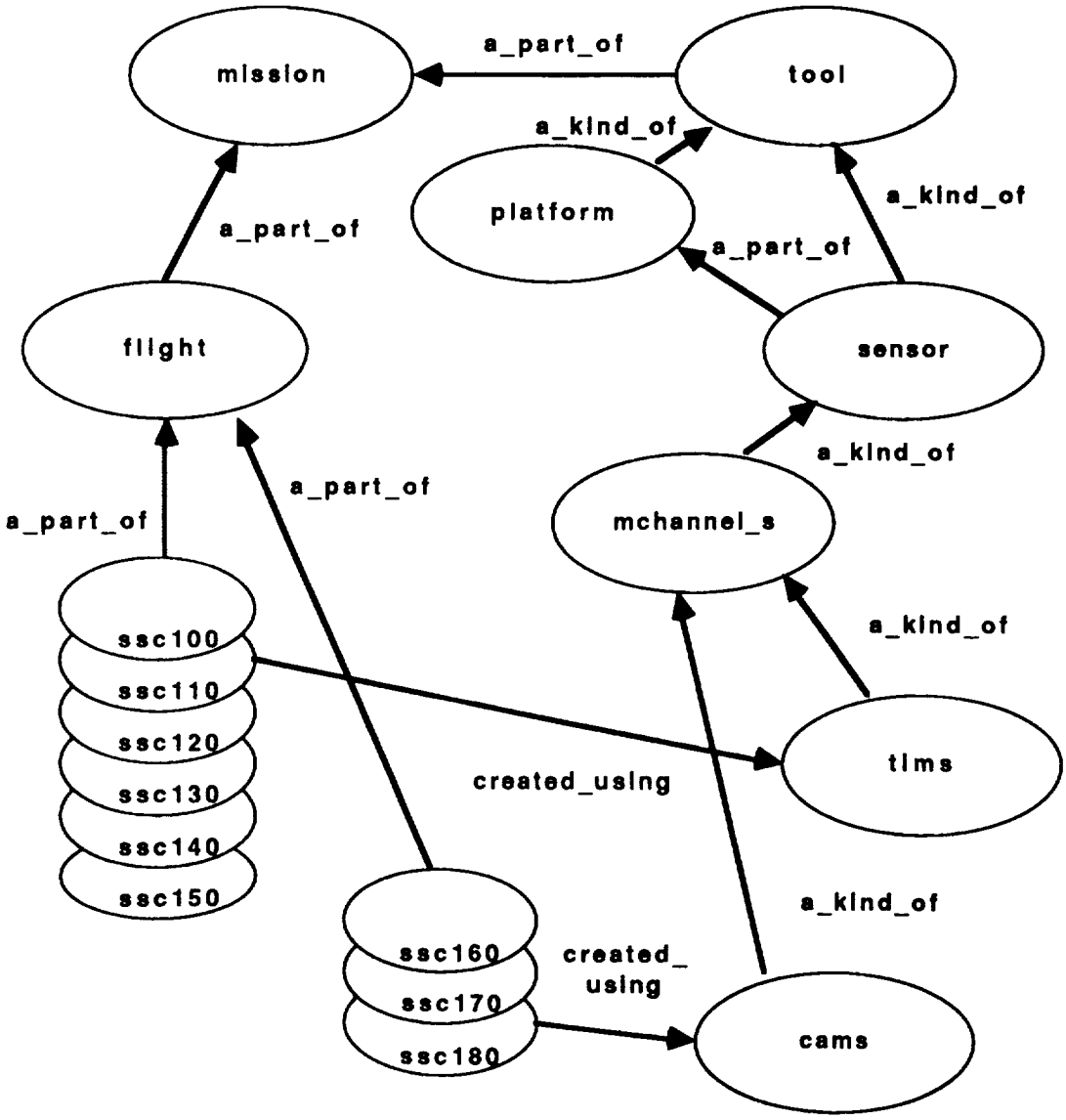

Figure 1. Metadata as a semantic network.

Figure 2 takes some of the objects in Figure 1 and develops them as frames with both unfilled and filled slots. This shows the complexity of the objects as well as the concept of slot inheritance. For example, all multichannel sensors (**mchannel_s**) have a **resolution** slot but it is not filled unless there is a specific data set pointer. Since the **mchannel_s** frame has a **resolution** slot, the **tims** frame, which is **a_kind_of mchannel_s**, has it, too.

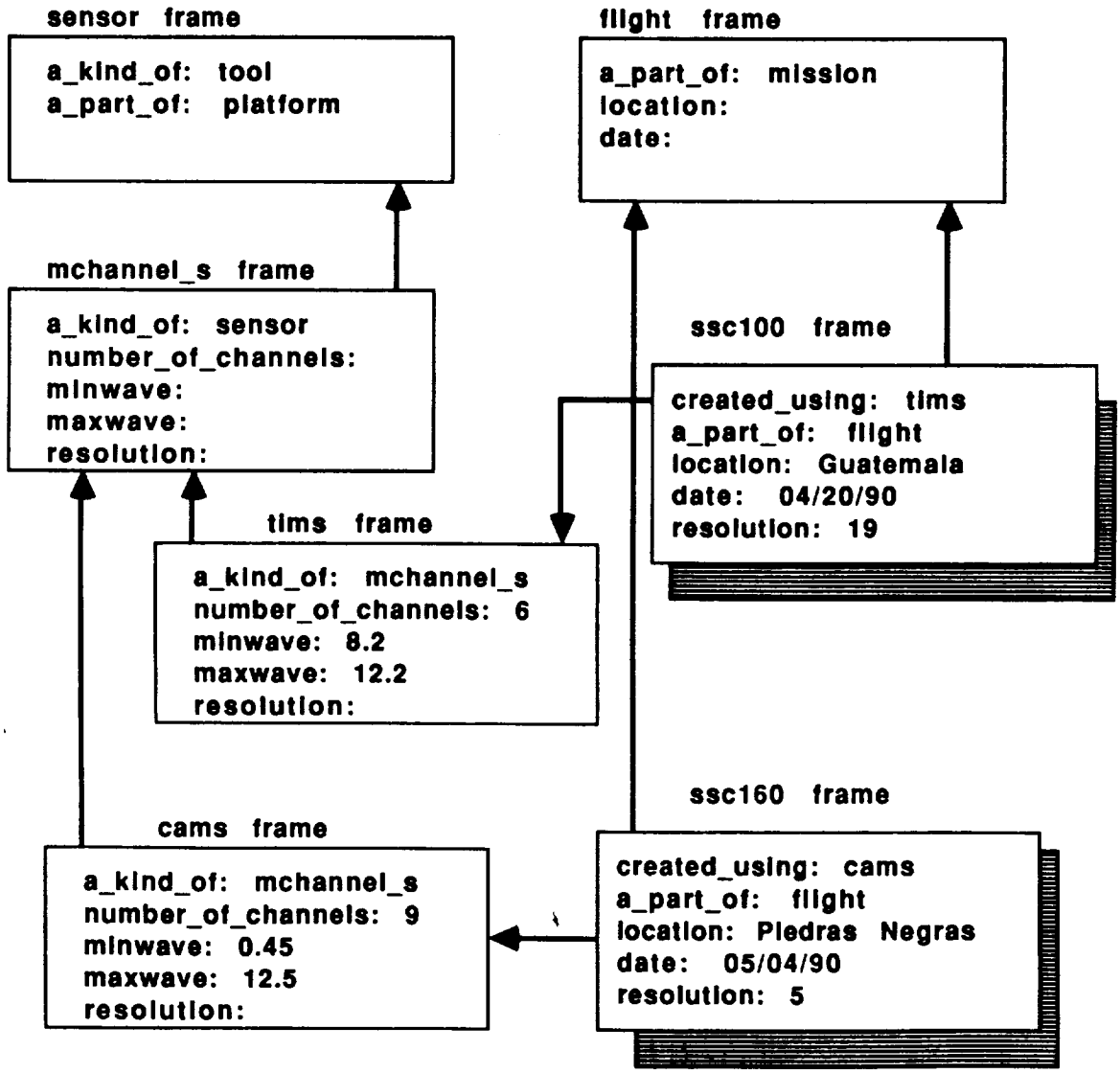

Figure 2. Metadata as frames.

The semantic network and frames indicated facts and principles that had to be represented in the metadata. However, we still needed to know if we could reason with this knowledge. Could the metadata provide a deductive logic that would allow predictions to be derived about what data to retrieve? The work of McEneaney (1990) suggested the use of logic programming to address this question.

Taking the frames, we coded the metadata into PROLOG. For example, the **flight**, **mchannel_s**, and **tims** frames became the following:

```
% Flight Frame ----------------------------------------

value(flight,a_part_of,mission).

slot(flight,location).
slot(flight,date).


% Multichannel Sensor Frame ---------------------------

value(mchannel_s,a_kind_of,sensor).

slot(mchannel_s,number_of_channels).
slot(mchannel_s,minwave).
slot(mchannel_s,maxwave).
slot(mchannel_s,resolution).

units(mchannel_s,minwave,micron).
units(mchannel_s,maxwave,micron).
units(mchannel_s,resolution,meter).


% TIMS Frame ------------------------------------------

value(tims,a_kind_of,mchannel_s).
value(tims,number_of_channels,6).
value(tims,minwave,8.2).
value(tims,maxwave,12.2).
```

Note that the **slot** predicate is used for those slots that are unfilled, while the **value** predicate is used for those slots that are filled. This approach makes writing inference rules simpler. Also, since slots having numeric values can have associated information, say about the units of measurement, we have included a **units** predicate.

The data set pointer frames here require the use of the **value** predicate only. However, in a completely developed metadata system, the frame would contain the rules by which the underlying data could be retrieved (i.e., E-mail addresses, login accounts, telenet machine numbers, etc.). The frame name would be the "entry_id" as defined in the *NSSDC Directory Interchange Format Manual*. For our prototype, the data set frame names are keys that we want our metadata to predict. Some examples of data set pointer frames written in PROLOG are:

```
% Data Set Pointer Frames ---------------------------------

value(ssc110,created_using,tims).
value(ssc110,a_part_of,flight).
value(ssc110,location,'Site 1/Peten').
value(ssc110,date,'04/21/90').
value(ssc110,resolution,5).

value(ssc120,created_using,tims).
value(ssc120,a_part_of,flight).
value(ssc120,location,'Site 1/Peten').
value(ssc120,date,'04/22/90').
value(ssc120,resolution,5).

value(ssc130,created_using,tims).
value(ssc130,a_part_of,flight).
value(ssc130,location,'Piedras Negras').
value(ssc130,date,'04/23/90').
value(ssc130,resolution,5).
```

The inference rules that can be applied to these frames can be written independently of the particular application. To be able to test and debug the metadata specification, we need value inheritance rules, slot inheritance rules, rules enabling qualifying slots to be inherited, and a good deal more. An example of the slot inheritance rule is:

```
has_slot(Object,Slot) :- slot(Object,Slot).
has_slot(Object,Slot) :- value(Object,a_kind_of,Superclass), has_slot(Superclass,Slot).
```

It should be mentioned at this point that the principle control mechanism of the "standard" PROLOG interpreter is depth first searching. Since PROLOG is an extensible language, other control strategies may be substituted.

The following are some examples of queries to and responses from the PROLOG code:

```
?- has_value(tims,Metadata_slot,Slot_value).
Metadata_slot = a_kind_of                Slot_value = mchannel_s
Metadata_slot = number_of_channels       Slot_value = 6
Metadata_slot = minwave                  Slot_value = 8.20
Metadata_slot = maxwave                  Slot_value = 12.20
Metadata_slot = a_kind_of                Slot_value = sensor
Metadata_slot = a_kind_of                Slot_value = tool
Metadata_slot = a_part_of                Slot_value = platform

?- has_slot(What,minwave).
What = mchannel_s   What = tims   What = cams

?- has_value(Entry_id,created_using,tims), has_value(Entry_id,location,'Site 1/Peten').
Entry_id = ssc110   Entry_id = ssc120
```

?- has_slot(Frame,Something), has_value(Entry_id,Something,'Piedras Negras').
Frame = flight Something = location Entry_id = ssc130
Frame = flight Something = location Entry_id = ssc160

?- has_value(ssc110,created_using,Sensor),has_value(Sensor,minwave,Minwave),
has_units(Sensor,minwave,Units).
Sensor = tims Minwave = 8.2 Units = microns

The first query demonstrates a browse of the TIMS frame. The responses reveal the filled slots in the TIMS frame, as well as the filled slots in the Multichannel Sensor and Sensor frames, since the TIMS frame inherits those values. The second query looks for those frames that have a particular slot, filled or unfilled. The third query is the command, "Give the key for any TIMS data set on Site 1/Peten." This is a more complex query than the previous ones in that it involves two constraints on the data set pointer frames. The fourth query is an example of partial information obtaining a result. This query seeks to find ancillary information about Piedras Negras data sets as well as keys. Finally, the fifth query represents the question, "What is the minwave and its units for the sensor used in creating the data set with key ssc110?"

In developing the PROLOG model of the metadata, our goal was to show what would be obtained by browsing, and to verify that certain keys would be obtained when particular facts were given in a query. Hence in a very empirical manner addressing the question, "Does the metadata provide an accurate knowledge model?" Earth scientists could propose questions and we could query the PROLOG model to determine if the metadata produced usable results.

Conclusion

PROLOG is an AI tool that can be used to write and test metadata specifications. A PROLOG model of the metadata can be used to gain insights into the relationships that the metadata attempts to capture. By querying the PROLOG model built for the TIMS and CAMS data sets at SSC, we were able to confirm relationships and access paths to data set pointers. Furthermore, we gained new insights into relationships, and realized the existence of relationships that had gone unnoticed, such as the need for a **created_using** relationship. PROLOG can be used to quickly prototype metadata before it is embedded in an intelligent information system, thus saving time and money by insuring that needs are met. Furthermore, since PROLOG is at the heart of logic programming languages such as LDL and LOGIN, it is conceivable that the work done with metadata specification can flow directly into an intelligent information system designed for accessing very large collections of data.

References

Ait-Kaci, H., Nasr, R., and Seo J. (1990). Implementing a Knowledge-based Library Information System with Typed Horn Logic. *Information Processing and Management,* 26(2), 249-268.

Campbell, W. and Cromp, R. (1990). Evolution of an Intelligent Information Fusion System. *Photogrammetric Engineering and Remote Sensing*, 56(6), 867-870.

Campbell, W., Cromp, R., Hill, S., Goettsche, C. and Dorfman, E. (1990). Intelligent Information Fusion for Spatial Data Management. *Proceedings of the 4th International Symposium on Spatial Data Handling,* 2, 567-578.

Cercone, N. and McCalla, G. (1987). *The Knowledge Frontier: Essays in the Representation of Knowledge,* New York: Springer Verlag.

Cromp, R. (1988). The Advice/Inquirer: A System for High-level Acquisition of Expert Knowledge. *Telematics and Informatics,* 5(3), 297-312.

Cromp, R. and Crook, S. (1989). An Intelligent User Interface for Browsing Satellite Data Catalogs. *Telematics and Informatics,* 6(3/4), 299-312.
Denney, R. (1991). Test-case Generation from PROLOG-based Specification. *IEEE Software,* 8(2), 49-57.
Dozier, J. (1990). Looking Ahead to EOS: The Earth Observing System. *Computers in Physics,* 4(3), 248-259.
Dorfman, E. (1991). Architecture of a Large Object-oriented Database for Remotely Sensed Data. *Proceedings of ACSM/SPRS/Auto Carto 10.*
Genesereth, M. and Ginsberg, L. (1985). Logic Programming. *Communications of the ACM,* 28(9), 933-941.
Goyal, P. (1989). Intelligent Information Systems: The Concept of an Intelligent Document. *Information Systems,* 14(4), 351-358.
Kerschberg, L. (1990). Expert Database Systems: Knowledge/Data Management Environments for Intelligent Information Systems. *Information Systems,* 15(1), 151-160.
Lucas, R. (1988). *Database Applications Using PROLOG.* New York: John Wiley and Sons.
McEneaney, J. (1990). Logic Programming as a Theoretical Tool in Educational Research. *Journal of Artificial Intelligence in Education,* 2(1), 63-78.
Moller-Jensen, L. (1990). Knowledge-based Classification of Urban Area Using Texture and Context Information in Landsat-TM Imagery. *Photogrammetric Engineering and Remote Sensing,* 56(6), 899-904.
Naqvi, S. and Tsur, T. (1989). *A Logical Language for Data and Knowledge Bases.* Rockville: Computer Science Press.
Roelofs, L. and Campbell, W. (1990). Using Expert Systems to Implement a Semantic Data Model of a Large Mass Store System. *Telematics and Informatics,* 7(3/4), 361-377.
Saacks, M. and Lopez, A. (1992). A Frame-Based Design for the TIMS and CAMS Metadata for a Stennis Information Management System. Under review.
Short, N. and Shastri, L. (1990). The Application of Connectionism to Query Planning/Scheduling in Intelligent User Interfaces. *Telematics and Informatics,* 7(3/4), 209-220.
Tanaka, H. (1988). DCKR-Knowledge Representation in PROLOG and its Application to Natural Language Processing. In *Proceedings of the First Franco-Japanese Symposium in Programming of Future Computers,* Fuchi, K. and Nivat, M. (Eds.), 427-439. Amsterdam: Elsevier Science Publishing.
Yan, S. (1991). *Foundations of Declarative Testing and Debugging in Logic Programming.* New Jersey: Ablex Publishing.

Acknowledgements - The authors would like to thank Kirk Sharp, Gay Irby, Bobby Junkin, and Terry Jackson of the Information Systems Division (NASA/SSC), Tom Sever and Doug Rickman of the Earth Science Division (NASA/SSC), and Bill Campbell and Bob Cromp of the NSSDC IDM project team (NASA/GSFC) for their comments and support.

Call for Papers

NASA 1993

Goddard Conference on Space Applications of Artificial Intelligence

May 1993
NASA Goddard Space Flight Center
Greenbelt, Maryland

The Eighth Annual Goddard Conference on Space Applications of Artificial Intelligence will focus on AI research and applications relevant to space systems, space operations, and space science. Topics will include, but are not limited to:

- Knowledge-based spacecraft command & control
- Expert system management & methodologies
- Distributed knowledge-based systems
- Intelligent database management
- Fault-tolerant rule-based systems
- Simulation-based reasoning
- Fault isolation & diagnosis
- Planning & scheduling
- Knowledge acquisition
- Robotics & telerobotics
- Neural networks
- Image analysis

Original, unpublished papers are now being solicited for the conference. Abstracts should be 300–500 words in length, and must describe work with clear AI content and applicability to space-related problems. Two copies of the abstract should be submitted by September 1, 1992 along with the author's name, affiliation, address and telephone number. Notification of tentative acceptance will be given by September 16, 1992. Papers should be no longer than 15 pages and must be submitted in camera-ready form for final acceptance by November 16, 1992.

Accepted papers will be presented formally or as poster presentations, which may include demonstrations. All accepted papers will be published in the Conference Proceedings as an official NASA document, and select papers will appear in a special issue of the international journal *Telematics and Informatics*. There will be a Conference award for Best Paper.

No commercial presentations will be accepted

Sponsored by NASA/GSFC

Mission Operations and Data Systems Directorate

1993 Goddard Conference on Space Applications of Artificial Intelligence
May 1993 — NASA/GSFC, Greenbelt, MD

❑ ***Abstracts due:*** Sept. 1, 1992
❑ ***Papers due:*** Nov. 16, 1992
❑ ***For further info call:*** (301) 286-3150

❑ ***Send abstracts to:***
Carl F. Hostetter
NASA/GSFC
Code 531.1
Greenbelt, MD 20771

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1992	Conference Publication

4. TITLE AND SUBTITLE
1992 Goddard Conference on Space Applications of Artificial Intelligence

5. FUNDING NUMBERS
JON-530-030-09-01-25

6. AUTHOR(S)
James L. Rash, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA-Goddard Space Flight Center
Greenbelt, Maryland 20771

8. PERFORMING ORGANIZATION REPORT NUMBER
92B00045
Code 530

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CP-3141

11. SUPPLEMENTARY NOTES
Rash: NASA-Goddard Space Flight Center, Greenbelt, Maryland, 20771

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 63

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This publication comprises the papers presented at the 1992 Goddard Conference on Space Applications of Artificial Intelligence held at the NASA/Goddard Space Flight Center, Greenbelt, Maryland, on May 5-6, 1992. The purpose of this annual conference is to provide a forum in which current research and development directed at space applications of artificial intelligence can be presented and discussed. The papers in this proceedings fall into the following areas: Planning and Scheduling, Control, Fault Monitoring/Diagnosis/Recovery, Information Management, Tools, Neural Networks, and Miscellaneous Applications.

14. SUBJECT TERMS
Artificial Intelligence, expert systems, planning, scheduling, fault isolation, fault diagnosis, control, neural networks.

15. NUMBER OF PAGES
264

16. PRICE CODE
A12

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

National Aeronautics and
Space Administration
Code JTT
Washington, D.C.
20546-0001

Official Business
Penalty for Private Use, $300

NASA

National Aeronautics and
Space Administration

Washington, D.C.
20546

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

Official Business
Penalty for Private Use $300

U.S.MAIL

SPECIAL FOURTH CLASS MAIL
BOOK

L3 001 CP-3141 72030289005698
NASA
CENTER FOR AEROSPACE INFORMATION
ACCESSIONING DEPT
P O BOX 8757 BWI ARPRT
BALTIMORE MD 21240