INFORMATION PROCESSING FOR CAD/VLSI ON A GENERALIZED DATA MANAGEMENT SYSTEM

ADIBA M.
NGUYEN G.T

IMAG
Université de Grenoble
Laboratoire de Génie Informatique
B.P 68
38402 SAINT-MARTIN-D'HERES Cedex
(France)

Abstract :
We propose mechanisms for controlling in PROLOG the semantics of applications developed on a generalized data management system called TIGRE. It is intended to provide database management facilities for generalized data, i.e alphanumeric data, but also large and complex objects such as documents, graphics and voice. It is being implemented at IMAG (University of Grenoble), in cooperation with the BULL Corporate Research Center. We emphasize providing a powerful tool for constraint checking on the logical structure of data, on their relationships, as well as the static properties of objects, and their dynamic behavior. It is thus possible to represent, control and manipulate the semantics associated with sophisticated applications involving generalized data. It is used here to implement a validation sub-system in an integrated CAD environment for VLSI circuits.

I. INTRODUCTION.

Numerous research have been carried on recently to extend DBMSs with new data types, e.g graphics, voice, documents. Further, AI or CAD applications increase the role of knowledge bases and expert systems for the design of new sophisticated applications, e.g decision support systems /VAS83/, or CAD/VLSI /ZAU83/.

This paper is devoted to the design and operations of a logic programming component for a generalized data management system called TIGRE, and its use as a validation sub-system in CAD for VLSI circuits /ADI84/.

First, it is intended to enhance the data management server of TIGRE with an inference mechanism to implement a powerful data typing and semantics management capability. Next, it is intended to provide the users with a flexible tool for the design of sophisticated applications involving complex data, e.g VLSI circuits. A prototype of our proposal has been implemented on DEC LSI-11 machines, running the RT-11 operating system. It is now being integrated to TIGRE on a Honeywell DPS 8/70 computer.

We emphasize on the representation of the SEMANTICS of applications developed on TIGRE. By this we mean :
- the objects involved,
- their static properties,
- their processing,
- the dynamic constraints they must obey.

We use here logic programming to define :
- the semantics associated with the conceptual schema of an application,
- the access and manipulation of data stored in a relational-like DBMS or in a database of PROLOG clauses, at the internal schema level,
- the definition of the processing proceres at the external schema level.

Specifically, we show that implementing a validation component for controlling the semantics of CAD/VLSI applications can be entirely off-loaded to Prolog by a generalized DBMS, in a cooperative approach. It is consistent with a distributed environment where designers update partials designs at their own workstations, sharing knowledge and semantics information stored in public database servers /KAT83/.

Section II is devoted to a short presentation of the TIGRE generalized data management system. In Section III, we expose the guidelines for integrating TIGRE and a logic programming component such as PROLOG. Section IV is an overview of object definition for CAD/VLSI in such an environment. In Section V, we describe the object instantiation and manipulation facilities required by our approach. Section VI is devoted to the management of the semantics of CAD/VLSI applications. Section VII is made of conclusions.

II. THE GENERALIZED DBMS TIGRE.

TIGRE is developed at Laboratoire de Génie Informatique of Institut IMAG (University of Grenoble), in cooperation with the BULL Corporate Research Center. It is intended to provide a generalized database server for the management of graphics, voice, documents and usual business data. A full scale prototype is being implemented on a Honeywell DPS 8/70 machine, running the MULTICS operating system (Fig. 1).

We assume here that the DBMS TIGRE exists, and that it allows for the modelling of sophisticated applications, using a generalized data model /LOP83/. This one provides the user with a strong data typing capability, with such notions as entity, relationship, classes, aggregation and generalization.

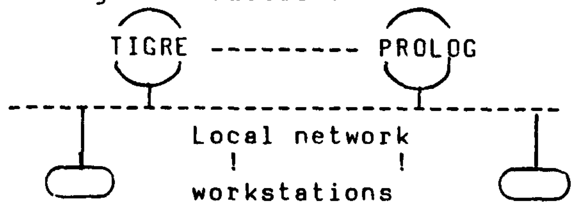

Fig. 1. Environment of the research.

On one hand, TIGRE will provide the data management facilities to store, retreive and modify large and/or complex objects, such as predefined cell components or circuit layouts. On the other hand, PROLOG will provide the semantics representation, control and manipulation capabilities for applications using these objects.

III. INTEGRATING TIGRE AND PROLOG.

TIGRE and PROLOG both operate on very different concepts. The generalized model of data of TIGRE is an extension of the entity-relationship model /CHE76/. It includes the notions of data type, class, generalization and aggregation. The associated data definition and manipulation language, called LAMBDA, is non procedural. PROLOG allows for the manipulation of complex data, structured in lists and trees.

Integrating both of them requires a level of consistency for the representation and the manipulation of common data. We represent in PROLOG all the data types defined with the model of TIGRE. We also guarantee that any constraint concerning the data types, the belonging of elements to classes, the inheritance of properties and the semantic constraints will be available and controlled in PROLOG. Moreover, we are able in such a case to write programs which are syntactically and semantically consistent with all applications developed on TIGRE.

IV. OBJECT DEFINITION FOR CAD/VLSI.

The first step corresponds to the definition in PROLOG of the basic data type operations. These include :
- basic type derivation, e.g scalar type,
- constructed types, e.g record, array,
- specialization of entities and associations,
- aggregation of entities.

These operators will guarantee the consistency of the data definitions and data manipulation operations already defined in TIGRE.

We automate the translation of data definition statements from LAMBDA in PROLOG clauses with a rewriting system. We use a CONSTRUCTOR concept. It provides for the definition of the logical structure of entities, independently of the operations on the data.

For our purpose, a circuit is the juxtaposition of a logic layer, of an electric layer and a physical layout. From the logic layer perspective, a circuit is composed of NODES which correspond to FUNCTIONS. Each one is connected to others by SIGNALS. From the electric perspective, the OPERATORS which implement the logic NODES are made of transistors, resistances and capacitances. They have specific characteristics. These are used to ensure that specific layout RULES are followed. A set of simplification rules are defined for this purpose. From the physical layout perspective, a circuit is an aggregate of CELLS, linked by CONNEC-

TIONS through PINS. Further details about circuit description and manipulation are given in /ADI84/.

V. OBJECT MANIPULATION.

In order to simplify the user interface, and alleviate the systematic scan of the entire search-space by the inference mechanisms of PROLOG, a few primitives are provided to allow for the data and the semantics manipulation, i.e :
- object instantiation,
- static properties control,
- processing procedures description,
- dynamic behavior control.

The primitives concerning the objects are
- create an object in the database (primitive object),
- derive an object from an existing one (dependent object),
- copy an object (making a dependent object independent),
- replace an object (making a (in-)dependent object dependent),
- modify an object (update the value of one or more attributes),
- delete an object.

The static properties are controlled by specific constructor clauses. The processing procedures must be written by the designers when the system is being implemented. They are application dependent. So far, the dynamic behavior definitions are not generated automatically. They must therefore be explicitly put down by the designers.

VI. SEMANTICS MANAGEMENT.

We have defined so far some basic mechanisms to implement in PROLOG a database schema which includes :
- basic operators on data types,
- constructors, which are clauses specifying semantic constraints on the logical structure of the data (Fig. 2).

Managing and controlling the semantics of sophisticated applications such as CAD for VLSI circuits can only be implemented if it is made EXPLICIT /KAT83/. We assume here that, except for data types and object structures, it can ENTIRELY be modelled and maintained through RELATIONSHIPS among the database objects. We provide for this purpose some primitives for the instantiation and manipulation of two specific types of relationships, namely : weak and strong relationships. They are used to account for the semantic links between objects, and particularly for the propagation of updates derived from the designers basic operations.

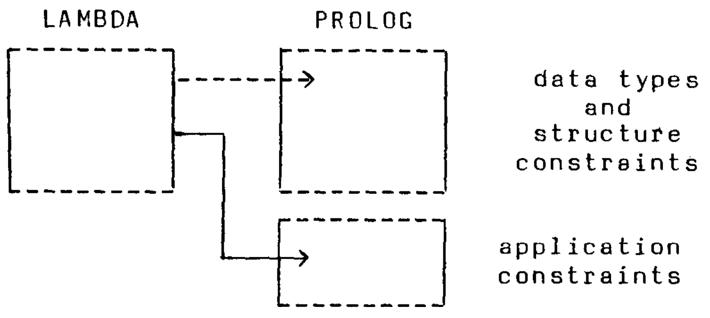

Fig. 2. Data types and application constraints.

A WEAK relationship between two objects can be created explicitly by the user. It can also be created by the system, when an object is instantiated by invocation of the COPY primitive. This enforces the sibling relationship between the original object and its copy. It permits the control of the logical structure and of the static properties of both objects when the constructor clauses are invoked by application programs. Further modifications of one of these objects do not imply modification of the other. Weakly related objects are INDEPENDENT.

A STRONG relationship can be created by the user. It can also be created by the system when :
- deriving an object from an existing one
- replacing an object by another.

This implies modification of the descendants of the parent objects. Further, modifications on strongly related objects are visible to the each other. They are DEPENDENT objects. In particular, dependent objects are used to model COMPONENTs of cells in a circuit, or derived objects relating a particular representation level to another. Similarly, independent objects are used to model various versions and alternatives of the same component.

Strong relationships thus represent COMPOSITION links between objects in CAD/VLSI applications. Weak relationships represent EQUIVALENCE links among objects or representations of the same object.

We assume here that the cooperation of TIGRE with a PROLOG validation sub-system can be implemented by protocols based SOLELY on coroutines. We wish indeed to make visible to the designer the derived updates related to his specific operations. We thus do not want them to be fully automatic, but allow him to interact, if he wishes.

With this assumption, it can be shown that the most complex semantics controls, i.e those that imply side-effect updates, can be implemented by two phase commit protocols. They are therefore consistent with distributed environments such as TIGRE.

We are involved in the development of a CAD/VLSI application, in which the knowledge associated to the logic, electric and physical representations of a circuit is stored as PROLOG clauses. More details about this approach are given in /ADI84/.

VII. CONCLUSION.

We emphasize on the representation of the SEMANTICS of sophisticated applications, such as CAD for VLSI circuits, to provide the users with :

- a generalized data model,
- powerful data manipulation and semantics representation and control facilities.

We show that extended entity-relationship like concepts can be easily modelled and manipulated in PROLOG. This offers to the users of TIGRE all the inference mechanisms of logic programming. It is thus possible to off-load the semantics definition and control to a validation sub-system, in a cooperative approach, whilst large and /or complex objects are handled by TIGRE. This leads to :

- the extension of the functionnalities of TIGRE,
- the cooperation of generalized database systems with knowledge bases developed in PROLOG,
- the integration of DBMS management facilities, specially those concerning the manipulation of large and/or complex objects, with artificial intelligence techniques, such as those using knowledge and semantics representation through logic programming.

Acknowledgments.

The authors are greatly indebted to Ms. Ofelia CERVANTES, Judith OLIVARES, Dominique RIALHE and Pascale WINNINGER, who experimented many of the ideas presented above during the last few months, in partial fulfillment of their respective Doctorate Degrees.

REFERENCES.

/ADI84/ ADIBA M., NGUYEN G.T
Knowledge Engineering for CAD/VLSI on a generalized data management system.
IFIP WG 5.2 Working Conference on Knowledge Engineering for Computer Aided Design.
Budapest (Hungary). September 1984.

/BER83/ BERGER-SABBATEL, J. IANESELLI, NGUYEN G.T
A Prolog database machine.
Third international workshop on database machines.
Munich (FRG). September 1983.

/CHE76/ CHEN P.P
The entity Relationship Model. Toward a unified view of data.
ACM TODS. Vol. 1, n° 1. 1976.

/KAT83/ KATZ R.H
Managing the chip design database.
Computer. December 1983.

/LOP83/ LOPEZ M., PALAZZO J., VELEZ F.
The TIGRE data model.
Research Report TIGRE n° 2.
IMAG. November 1983.

/NGU82/ NGUYEN G.T, L. FERRAT, H. GALY
A high level interface for a local network database system.
Proc. INFOCOM Conference.
Las Vegas (USA). March 1982.

/VAS83/ VASSILIOU Y. et al.
How does an expert system get its data ?
Proc. 9th Very Large Data Bases Conference.
Florence (Italy). October 1983.

/ZAU83/ ZAUMEN W.T
Computer assisted circuit evaluation in Prolog for VLSI.
ACM-IEEE Database Week.
San-Jose (USA). May 1983.