Research Summary

Static and dynamic verification of object-oriented transactions

The most recent papers present a technology based on object-oriented assertion languages that overcomes a major limitation of persistent and database object systems. This technology allows specification of object-oriented integrity constraints, their static verification and dynamic enforcement. Proof strategies are based on static and dynamic verification techniques as they apply to verification of object-oriented transactions. The components of this technology presented in the paper *Verification technology for object-oriented/XML transactions* are an object-oriented constraint language, a verification system with advanced typing and logic capabilities, predefined libraries of object-oriented specification and verification theories, and an extended virtual platform for integrating constraints into the run-time type system and their management. The initial results are based on JML (Java Modeling language) and a higher-order verification system PVS. Most recent results presented in the paper *Object-oriented constraints for XML* are based on Spec#. Spec# technology comes with automatic static verification of code with respect to the specified constraints.

Integration of object-oriented languages and XML

Most of this work has been motivated by the problems of object-oriented interfaces to XML that have not been able to express typical XML Schema constraints, database constraints in particular. The structural features are represented within the limitations of object-oriented type systems including particles (elements and groups) and type hierarchies (simple and complex types and type derivations). The main novelty is that features of XML Schema that are not expressible in object-oriented type systems are specified in an object-oriented assertion language. Collaboration with Microsoft Research lead to two papers *Mapping XSD to OO schemas* and *An object-oriented core of XML Schema*. Further research based on assertion languages made it possible to use a prover technology to verify constraints associated with XML Schema. More importantly, object-oriented programs processing XML data are now subject to these constraints expressed in a language such

as JML or Spec#. The prover is interfaced with an extended virtual platform that manages object-oriented specifications of XML types and structures via reflection.

Verification theories for XML

Representation and verification techniques for XML Schema types, structures, and applications, in a program verification system PVS, are presented in the papers *Program verification techniques for XML Schema based technologies* and *Verification theories for XML Schema*. Type derivations by restriction and extension as defined in XML Schema are represented in the PVS type system using predicate subtyping. Powerful PVS logic capabilities are used to express complex constraints of XML Schema and its applications. Transaction verification methodology is grounded on declarative, logic-based specification of the frame constraints and the actual transaction updates. The overall approach has a model theory based on the view of XML types and structures as theories.

Virtual platforms

The architecture and applications of an extended virtual platform based on the Java Virtual Machine are described in the paper *Reflective constraint management for languages on virtual platforms*. Extending an object-oriented type system with assertions makes it possible for programs using reflection to rely on semantic information to ensure correct use of discovered types. Using extended reflective capabilities to access assertions in (dynamically) loaded class objects allows a variety of general and flexible verification techniques. The XVP (Extended Virtual Platform) implements these features by extending the Java Virtual Machine with the proposed functionalities. One of the goals of the XVP is to provide a virtual platform that supports JML and the programming by contract methodology.

Program verification technology

Using a higher-order verification system such as PVS makes it possible to develop verification technique based on specialized logics such as temporal logic or separation logic. In the paper *Temporal verification theories for*

Java-like classes the problem of a suitable logic for Java-related and other object-oriented assertion languages, is considered. A suitable prover technology is also presented and the required techniques for verifying properties of object types extended with logic-based constraints are developed. The temporal logic-based approach makes it possible to reason about properties of sequences of object states which allows verification of behavioral subtyping requirements that are based on history properties. This approach has a model theory based on the view of types as theories. The most recent results are presented in the paper *Algebraic specification techniques for parametric types with logic-based constraints*.

Genericity for Java

A definitive paper *Genericity in Java: Persistent and database systems implications* provides experimental evidence of type violations in Java 5.0, develops a formal basis along with formal proofs showing the source of these problems, and it elaborates a correct implementation technique based on legacy compatible extension of the JVM. This paper was preceded by a paper on a reflective solution for the problem of extending Java with parametric polymorphism: *Parametric Polymorphism for Java: A Reflective Solution*. A follow-up paper *Parametric Polymorphism and Orthogonal Persistence* presents a specific implementation technique for this problem. Distinctive features of this solution are that the run-time type information and the persistent type information are correct. The paper *Parametric polymorphism for Java: Is there any hope in sight* shows a variety of unfortunate consequences of a solution to this problem accepted in Java 5.0 and its subsequent versions.

Model theory for object-oriented and XML models

A significant segment of recent research has been devoted to the development of various aspects of a general model theory for paradigms based on an advanced type system extended with logic-based constraints. The paper *Behavioral compatibility of self-typed theories* explores this model theory for the object-oriented paradigm with application to the problem of behavioral compatibility of classes. This model theory has proved to be a suitable general framework for schema and data integration as reported in the paper *A model theory for generic schema management* (research carried out at Mi-

crosoft Research) as well as for the integration of objects, XML and database models as reported in the paper *Institutions: Integrating objects, XML and databases*. A distinctive feature of this model theory is that it applies to a variety of logic paradigms and hence to a variety of constraint languages for either XML-related paradigms or the object-oriented paradigm or both. The paper *Semantics of objectified XML* applies this model theory to the integration of XML with typed object-oriented languages where both are equipped with logic-based constraints.

Persistent Java technology

This research covers persistence, parametric polymorphism, reflection, and assertions (constraints). Techniques for implementing a constraint language in the Java environment on top of a persistent extension of the Java Virtual Machine are presented in the paper *Orthogonal to the Java Imperative*. A reflective implementation technique developed for Java OQL is presented in the paper *Java and OQL: A Reflective Solution for the Impedance Mismatch*. Application of a modern program verification system to the problem of verification of consistency of Java transactions has been reported in the paper *Consistency of Java transactions*.

Object-oriented database technology

Research on object-oriented database technology has been largely related to the technology proposed by the ODMG Standard. This was the focus of the research grant from NSF: *A Family of the ODMG Object Models*. The grant was a follow-up to the paper *The ODMG Object Model: Does it Make Sense?*. The goal of this project was to provide research and technical solutions for major problems in the ODMG Standard, the proposed industrial standard for object-oriented databases. The main contributions of this project to the ODMG Standard are in the underlying type systems, the model of persistence, and the constraint language. The most important and definitive results of this project are presented in the paper *Type checking OQL queries in the ODMG type systems*. This paper proves that it is not possible to type check OQL queries in the type system underlying the ODMG Object Model. The second negative result is that OQL queries cannot be type checked in the type system of the Java binding of the ODMG

Standard either. A solution is to extend the ODMG Object Model with explicit support for parametric polymorphism. These results showed that Java cannot be a viable database programming language unless extended with parametric polymorphism. Problems of the accepted solution for Java 5.0 are established in the paper *Genericity in Java: Persistent and database systems implications*. The results of this project also include some implementation models and techniques applicable to the technology proposed by the ODMG Standard reported in the paper *Java and OQL: A reflective solution for the impedance mismatch*. Other related papers are *A Family of the ODMG Object Models*, and *O2 and the ODMG Standard: Do They Match?*

Typed and temporal object-oriented technology

This was the focus of the research supported by two research grants: *A Typed and Temporal Object-Oriented Technology (U.S Army Research Office and Kansas Technology Enterprise Corporation)* and *Integrated Object-Oriented Environment for Modeling, Simulation, Prototyping and Active Databases (DOD)*. The developed prototype technology is persistent and based on an advanced object-oriented type system. The user-oriented semantic model is declarative, based on temporal logic. The paradigm has a model theory presented in two major mathematical papers *Order-Sorted Model Theory for Temporal Executable Specifications* and *Semantics of Temporal Classes*. Database aspects of this generic technology including the underlying architecture are presented in the papers: *A Typed and Temporal Object-Oriented Database Technology* and *A Temporal Constraint System for Object-Oriented Databases*. The results of integration of the type system and the temporal constraint system are presented in the paper *Constrained Matching is Type Safe*. The main targeted application is object-oriented flight simulator technology. The relevant papers are: *Object-Oriented Flight Simulator Technology* and *Flight Simulator Database: Object-Oriented Design and Implementation*.

Database programming languages

The initial research in the area of the object-oriented database technology was mainly related to the design and implementation of *Modulex*, a database programming environment supporting multiple paradigms (object-oriented and

relational in particular). The associated publications are the book *Object-Oriented Database Programming* and the papers
Object-Oriented Database Programming Environment Based on Modula-2, *Persistent Meta-Objects* and *Toward Multiparadigm Database Interfaces*. Applications of the developed technology have been in the area of production management systems and spatial data management with the associated publications: *Object-Oriented Geo-Information Processing in Modulex* and *Advanced Database Programming Languages: A Geo-Information Processing Prospective*.

Database Type Systems

The follow-up research was largely devoted to further developments of the strongly typed database technology with the goal to introduce high-degrees of polymorphism, a sophisticated meta-level support, polymorphic facilities based on kinds (of types), higher-order polymorphism and reflection. The associated publications are: *Generic Modules, Kinds and Polymorphism for Modula-2*, *Joins as Pullbacks*, *Integrating Inheritance, Subtype and Parametric Polymorphism in Database Type Systems*, *Object-oriented Type Evolution Using Reflection*, *Type-Safe Linguistic Reflection: A Generator Technology*, *Polymorphic and Reflective Type Structures*, and *F-bounded Polymorphism for Database Programming Languages*.

Typed logic-based object-oriented technology

Further developments are reflected in the papers: *Declarative Object-Oriented Programming: Inheritance, Subtyping and Prototyping*, *A Typed Object-Oriented Database Technology with Deductive and Reflective Capabilities*, *Expressibility of Typed Logic Paradigms for Object-Oriented Databases*, *Typed Declarative Object-Oriented Database Programming*, *A Typed and Temporal Object-Oriented Database Technology*. These papers deal with a logic-based, typed object-oriented technology. A variety of logic paradigms are explored as a basis for declarative object-oriented languages, prototyping tools, and a strongly typed object-oriented database technology.

Relational database technology

The early work on the integration of database and programming languages and systems produced a complete, relational, strongly typed, multi-user database programming environment including a high-level, non-procedural definition, query, manipulation and control language, transaction support, dynamic indices, concurrency control and recovery. This project was supported by the NSF grant *Extended Relational Database Programming Environment*. The system was in actual commercial use at some ten sites. The research visit to Japan included presentation of the above work in a number of research institutions involved in the projects of the new generation of computer systems (ETL (Electrotechnical Laboratory), Tokyo University, Tsukuba University, NEC Research Laboratory, Hitachi Systems Laboratory, NTT Computer and Communications Laboratory) as well as a visit to ICOT (Tokyo Institute for the New Generation of Computer Systems). The integration of the major published results on the relational database technology together with the results on the development of a specific, strongly typed relational database technology have been presented in the book *Relational Database Technology*. The relevant papers are: *Relational Pascal Database Interface* and an application paper *Relational Pascal Model of Soil Database*.

Programming languages and programming methodology

Early work in this area, starting with the Ph.D. dissertation (*Algebraic Aspects of Programming and Formal Languages*) and the follow-up papers (*Natural State Transformations*, *Categorical Theory of Tree Processing*) dealt with the algebraic approach to some problems in formal language theory (tree transformations) and programming languages (abstract data types). The first book co-authored with Michael A. Arbib (*The Design of Well-Structured and Correct Programs*) and related papers (such as *Proof Rules for Gotos*) belong to the area of programming methodology based on the axiomatic approach (Hoare's logic) and formal verification of program correctness. The postdoctoral work at the University of Edinburgh was largely devoted to the related problems.