The primary area of my research is the design, optimization, and implementation of programming languages and dependable software systems. My graduate studies mainly focused on one aspect of software reliability: meaning preservation of program optimizations, in particular of cross-module transformations. My research interests also include programming languages for Web technologies (Java and XML) and Internet security.

Overview of my graduate research. Program optimization performed by the compiler allows a programmer to write elegant, readable code without sacrificing the program’s efficiency. For instance, a programmer can write a program which consists of many short functions without having to worry about the run-time overhead of a function call, since in all likelihood most such functions will be inlined by the compiler.

However, it is essential that compiler transformations do not change the meaning of the program, i.e. the original and the transformed programs behave exactly the same way. A rigorous proof is needed to guarantee that this is indeed the case. Such proofs are non-trivial even for simple program transformations, and are especially challenging in the presence of modules and separate compilation.

I have studied meaning preservation of program transformations in a formal framework given by a calculus for modeling programs constructed of separate fragments, or modules, in the style of the $\lambda$-calculus.

The property that justifies a certain class of program transformations in such a framework is called computational soundness. It states that any transformation expressible by the calculus rewrite rules\footnote{such transformations are referred to as local.} preserves meaning with respect to the small-step operational semantics. The usual technique for proving computational soundness is based on two well-known properties of a calculus: confluence and standardization. However, the traditional technique for proving computational soundness is not applicable to calculi which lack confluence, such as the module calculus mentioned above.

I have overcome this difficulty by developing a new technique for proving computational soundness based on two new properties, \textit{lift} and \textit{project}. This
approach does not require other properties commonly used in computational soundness proofs, such as left-linearity of the calculus and finiteness of developments. The technique and its application to the calculus of records (the core of the module calculus) are presented in detail in my dissertation. The earlier version of the calculus and the technique was presented in Machkasova, Turbak “A Calculus for Link-time Compilation”, ESOP2000.

**Future directions based on the graduate research.** The results in my dissertation open a wide range of directions for future research, such as:

1. Extending the module calculus to represent more features of separate compilation. I am especially interested in dynamic linking or arbitrary nesting of modules, and proving cross-module transformations in this framework.

2. Studying types, in particular intersection types, in relation to module calculi. This direction is inspired by the use of the reducibility method for proving standardization of a calculus.

3. Exploring applicability of the lift and project technique to other calculi. Likely candidates include certain calculi with explicit substitution, calculi with a letrec rule, and, eventually, calculi for modeling imperative features of programming languages.

4. Combining this technique with other program analyses, such as termination analysis, and extending it to other program transformations.

**Other future research directions.** Below are other areas in which I have significant research interest and background:

1. Design and development of dependable software systems. This includes development and use of program verification tools and proof systems for proving properties of programs. Program verification tools and proof systems facilitate reasoning about a program. For instance, such a system would allow us to prove that the program does not use certain system resources or a certain combination of resources at the same time.

Reliability of software systems requires rigorous formal justification. This challenging area combines mathematical logic, theoretical aspects of programming language research, such as type systems and formal
reasoning, and software engineering approach. This is an exciting re-
search direction for me, and it is an attractive area for grant funding.
This research direction opens numerous possibilities for a student project
because of its practical nature and the fact many components of such
a project do not require prior theoretical background, so it can be a
perfect starting point for an undergraduate student.

2. Study of programming language and program optimization issues spe-
cific to Web applications. Programming languages and technologies
used in Web applications include Java servlets, XML, and mobile agents.
I have an extensive experience with servlets through the course on E-
commerce which I have developed an taught at Wellesley College, and
with XML and calculi for mobile computations through seminar “Pro-
gramming the Internet” at Boston University.

An example of a problem in this realm is program optimizations for
Java servlets. The factors that may influence optimizations of servlets
are high level of multi-threading and the need for synchronized access
to shared resources, such as the access to databases.

3. I am interested in research related to E-commerce and Internet security.
Possible directions of my research include automated verification tools
for security of Internet communications, designing dependable tools for
handling “electronic money” (such as DigiCash), and security of XML.
I will detail my research plans related to XML security, since XML is an
especially “hot” research topic, and is also very attractive to students.

In the last couple of years XML has become a standard for structured
information interchange in Web-based applications, in particular in e-
business. While XML has been extended with security features, such as
“XML Signature” and “XML Encryption,” experts agree that security
of XML-based Web applications still cannot be relied on.

One possible approach to this problem is through a type system which
carries security information. Such a system would guarantee that a
program with a certain type has the desired security properties. An-
other possible approach is through proof-carrying code, i.e. a program
that carries a formal proof of its own correctness which can be verified
on the client machine.