The first step in designing software is to pick the right language for the task. A single programming language, however, is rarely sufficient to meet the needs of a growing project. For example, Facebook began with a PHP script but has recently developed their own language, Hack [4], to gradually augment PHP code with static type checking. Similarly, Jane Street Capital primarily used Visual Basic until one employee gradually rewrote the core trading software in the safer OCaml language, and Bank of America maintains a trusted layer of COBOL underneath its Java modules. This phenomena of combining languages is widespread, and it arises because software projects encompass a spectrum of *changing* goals.

My goal is to build a *full spectrum programming language* that accomodates degrees of safety, ranging from the high-assurance required for kernel software to powerful flexibility of scripts. More importantly, this language will guarantee the interaction of its component languages. Type system invariants will be preserved during interactions with untyped modules, and scripts that grow into full-fledged programs will be straightforward to port into typed or even dependently typed code.

General Approach

Full-spectrum programming is not a novel idea; in fact, the research community has been working on combined-language systems for the past decade. Pioneering work by Matthews and Findler studied a joint language of Scheme and ML [5], and more recently Osera et. al examined the interaction of simple and dependent types [6]. Other projects have explored topics such as certified foreign-function interfaces [1] and safe interoperability of untyped code with Java [2]. The time has come for a full-spectrum language that unifies these efforts.

My proposal is to study the interaction between dependently-typed and (un)typed code, building on the gradual typing work done by Sam Tobin-Hochstadt while at Northeastern University [9]. Tobin-Hochstadt gave a full proof-of-concept for safe transitions between untyped and simply-typed modules. I plan to create safe transitions between his languages and a certified, dependently-typed language of my own design. The feedback loop for realizing this design will include a theoretical framework—in particular, a constructive system with dynamically-checkable guarantees—and an implemention within the Racket ecosystem, which offers an extensible core language for development and diverse libraries to translate. My passion for the beautiful concepts in programming languages research and prior experience bringing these concepts to industry recommend me for this task.

My studies will be directed by a group of experienced faculty. Matthias Felleisen has spent the past eight years working on Typed Racket with three of his students. Amal Ahmed has built the theoretical foundations for a multi-language compiler [8]. Jan Vitek has studied the performance of gradually-typed languages, bridging the gap between academia and industry. I am extremely lucky to have found such a diverse set of mentors at one institution.

Broader Impacts

The three motivating examples listed in my introduction are interesting, but relatively benign. Much more compelling reasons for building a full spectrum language are the Swedish Pension system, which consists of 320,000 lines of Perl [7], and the DARPA I2O HACMS project, which seeks to provide a simple interface for programming Unmanned Autonomous Vehicles [3].

The pension system quickly grew from a script when the government's hired IT firm failed to deliver a product. Sweden needs a plan to gradually certify individual modules and protect its retirees because a complete rewrite of this system is infeasible. Conversely, the High-Assurance Cyber Military Systems (HACMS) project is seeking to de-classify sections of a secure codebase. To create a simple API for autonomous machines, they essentially need a method of linking scripts to dependently typed programs. These are precisely the issues I seek to address with my research.

Beyond its applications to real-world systems, a full-spectrum language is ideal for teaching programming concepts to novices because it offers smooth transitions from beginning to advanced topics. I hope to introduce my full spectrum language to the Bootstrap program, an afterschool initiative that teaches algebra to middle school students. Introducing dependent types in Bootstrap would allow a natural transition from algebra to logic and proofs. Dependent types allow extremely powerful reasoning about programs, enough for students to write lemmas and theorems they might find in a textbook. Being able to rewrite theorems as programs will help students better understand the ideas they see on the blackboard and improve their formal reasoning skills; programming is a more intuitive and comprehensive introduction to proofs than the rigid "checklists" that U.S. high-school students learn in geometry class. Thus a full-spectrum language is the key to an engaging and comprehensive program for mathematics education.

[1] Michael Furr and Jeffrey S Foster. Checking type safety of foreign function calls. In *TOPLAS*, 2005.

[2] Kathryn E Gray, Robert Bruce Findler, and Matthew Flatt. Fine-grained interoperability through mirrors and contracts. In *OOPSLA*, 2005.

[3] High-Assurance Cyber Military Systems (HACMS). `http://www.darpa.mil/Our_Work/I2O/Programs/High-Assurance_Cyber_Military_Systems_(HACMS).aspx`. Accessed: 2014-10-20.

[4] HHVM and Hack manual. `http://docs.hhvm.com/manual/en/index.php`. Accessed: 2014-10-27.

[5] Jacob Matthews and Robert Bruce Findler. Operational semantics for multi-language programs. In *POPL*, 2007.

[6] Peter-Michael Osera, Vilhelm Sjöberg, and Steve Zdancewic. Dependent interoperability. In *PLPV*, 2012.

[7] Of scripts and programs: Tall tales, urban legends, and future prospects. Invited talk at DLS '09: `https://www.cs.purdue.edu/homes/jv/talks/dls09.pdf`. Accessed: 2014-10-20.

[8] James T Perconti and Amal Ahmed. Verifying an open compiler using multi-language semantics. In *ESOP*, 2014.

[9] Sam Tobin-Hochstadt. *Typed scheme: From scripts to programs.* PhD dissertation, Northeastern University, 2010.