POP Seminar

Modular and Automated Type-Soundness Verification for Language Extensions

Sebastian Erdweg

Technische Universitaet Darmstadt

Abstract:

Language extensions introduce high-level programming constructs that protect programmers from low-level details and repetitive tasks. For such an abstraction barrier to be sustainable, it is important that no errors are reported in terms of generated code. A typical strategy is to check the original user code prior to translation into a low-level encoding, applying the assumption that the translation does not introduce new errors. Unfortunately, such assumption is untenable in general, but in particular in the context of extensible programming languages, such as Racket or SugarJ, that allow regular programmers to define language extensions

We present a formalism for building and automatically verifying the type-soundness of syntactic language extensions. To build a type-sound language extension with our formalism, a developer declares an extended syntax, type rules for the extended syntax and translation rules into the (possibly further extended) base language. Our formalism then validates that the user-defined type rules are sufficient to guarantee that the code generated by the translation rules cannot contain any type errors. This effectively ensures that an initial type check prior to translation precludes type errors in generated code. We have implemented a core system in PLT Redex and we have developed a syntactically extensible variant of System Fomega that we extend with let notation, monadic do blocks, and algebraic data types. Our formalism verifies the soundness of each extension automatically.

Bio:

Sebastian is post-doctoral researcher in the Software Technology Group at TU Darmstadt. Please see his personal website (http://erdweg.org/) for further information.

He is the lead developer of the extensible programming language SugarJ, which allows programmers to flexibly integrate new syntax, static analyses, and editor support by library import. SugarJ extensions are scoped via the module system and multiple extensions compose if activated in the same scope. Since SugarJ allows flexible and principled domain abstraction, it is particularly well-suited for the embedding of domain-specific languages.

His general research interests include: extensible programming languages, domain-specific languages, language design, language tooling, declarative parsing methodologies, module systems, type systems, and static analysis.

Appointments: Christian Kastner, kaestner@cs.cmu.edu

Thursday, May 2, 2013
Gates Hillman Center 6501
1:30 PM – 3:00 PM