Using Type Systems to Enable Program Interoperability

Rod Moten

PROARC, Inc.

rmoten@proarc-inc.com

Program interoperability is when two or more programs interchange data in a meaningful way. To achieve program interoperability requires that the developers of the programs agree upon a communication protocol to interchange data and agree upon the structure and meaning of the data. In computing today, implementing interoperability isn't difficult and there are many approaches that help automate the creation of code to interchange data. A benefit of achieving interoperability automatically is empowering people not trained as programmers to create new applications by using existing applications. In other words, people can take two or more programs and combine them to create a new application. The new application is referred to as a *mashup* and is one way of performing *high-level information fusion*. To achieve this capability on a large scale with various types of applications requires resolving *semantic heterogeneity*. One cause of semantic heterogeneity is when two programs have different representation of data with the same meaning. The usual approach to achieve this is for the developers to agree on a common representation of all data that can be interchanged between their programs. However to create mashups or perform high-level information fusion in a large-scale heterogeneous environment requires resolving semantic heterogeneity dynamically.

In this talk, we will present research we are conducting to utilize type systems as a data interchange language that has built-in support for detecting and resolving semantic heterogeneity. We use a language we have invented, called TTIQ, to demonstrate how types systems may be more beneficial in large-scale heterogeneous environments than the start-of-the-art.

The talk assumes that the audience isn't familiar with type systems, but is familiar with set theory and topology. Type systems are formal languages based on type theory. Type theory was developed as an alternative method to set theory for formalizing mathematics. Recently, computer scientists and mathematicians have shown that type systems can be interpreted as topological spaces. Therefore, in the talk we will provide a brief introduction to type theory and its relationship to topology.

Biography

Rod Moten was born and raised in Long Island, NY. He obtained a BS in Computer Science and Mathematics from the State University of NY (SUNY) at Stony Brook and a PhD in Computer Science from Cornell University. Rod's dissertation focused on the implementation of parallel high-order functional programming languages. Although, high-order functional programming languages have a small following, Rod has used the theory and algorithms to implement these languages to develop techniques for improving software interoperability. Rod has successfully applied these techniques to create software utilized in the private and public sectors. Rod is actively pursuing research utilizing type systems to improve interoperability in the Semantic Web and Big Data applications and has several publications in this area. Rod has over 15 years of experience as a researcher, professor, trainer, developer, software architect, tech lead, systems engineer and scrum master. Rod is currently the Chief Scientist of a Maryland-based start-up called PROARC, Inc.