GRI: Interpreter of a dynamic language for GRaph algorithms

Sandeep Dasgupta
University Of Illinois at Urbana Champaign.
sdasgup3@illinois.edu

ABSTRACT

As graphical models are increasingly become popular in various field, the domain experts often struggle to represent and compute on such model in a convenient and efficient way. In this project we develop a dynamically typed language to represent and compute on the graphical models which provides both the desired convenience without loosing much on the efficiency.

Keywords

Graph Algorithm, dynamically typed language, interpreter

1. INTRODUCTION

As graphical models are increasingly being used in various fields like biochemistry (genomics), electrical engineering (communication networks and coding theory), computer science (algorithms and computation) and operations research (scheduling), organizational structures, social networking, there is a need to represent and allow computation on them in a convenient and efficient way. This involves (but not limited to)

- Designing a language which provide an convenient interface to the programmer to program those models. This is essential so that even for domain experts who are not coding experts can code and reason about their implementation. Ease of interface could be due to:
  - Expressive power of the language representing those models.
  - Intuitive extensibility of the language.
  - Ability of the language to provide exploratory programming, where the user may experiment with different ideas (without dwelling much into the language syntax) before coming to a conclusive one.
- Designed language need to be efficient in the following sense.
  - Underlying design decisions including data structures need to be carefully crafted to achieve expected runtime w.r.t the input size.
  - Implementation need to be scalable w.r.t the space/time requirements. This is important because most of the graph algorithm typically work on huge input sizes.

2. RELATED WORK

Our work in mostly inspired by the line of work by GUESS [**?**] and Graphal [**?**]. GUESS, a novel system for graph exploration that combines an interpreted language with a graphical front end that allows researchers to rapidly prototype and deploy new visualizations. GUESS also contains a novel, interactive interpreter that connects the language and interface in a way that facilities exploratory visualization tasks. Our language, Gython, is a domain-specific embedded language which provides all the advantages of Python with new, graph specific operators, primitives, and shortcuts.

Graphal is an interpreter of a programming language that is mainly oriented to graph algorithms. There is a command line interpreter and a graphical integrated development environment. The IDE contains text editor for programmers, compilation and script output, advanced debugger and visualization window. The progress of the interpreted and debugged graph algorithm can be displayed in 3D scene.

Our language design is very similar to above two work. But we additionally provided built-in functions for basic graph algorithms. This not only help us getting convinient short hand notations to compute those basic algorithms, but also we gain on performance due the fact that those basic algorithms are now compiled.

3. A MOTIVATING EXAMPLE

4. DEFINITIONS AND NOTATIONS

4.1 Interprocedural Analysis

.