GRI: Interpreter of a dynamic language for GRaph algorithms

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

ABSTRACT

we decided to work on a dynamically typed language to represent graphs and apply various computations on them. With a dynamically typed language the user do not have to worry much about declaring types and can focus mostly on his/her experiments.

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

3. A MOTIVATING EXAMPLE

4. DEFINITIONS AND NOTATIONS

4.1 Interprocedural Analysis

.