Columbia University

COMS 4115 Programming Languages and Translators
Fall 2007

Prof Stephen Edwards

CRAWL

A Graph Language

Rajesh Venkataraman (rv2187@columbia.edu)
Carter Adams (carteradams@gmail.com)
Amoghavarsha Ramappa (ar2645@columbia.edu)

Chapter 1

Introduction

1.1 Overview

Motivation

Graphs play an important role in many applications in Computer Science. The theory of Graphs has been successfully exploited in many practical applications, with the most significant ones related to networks. In spite of such widespread interest in Graphs and their properties, all attempts at solving problems in Graph Theory computationally have almost always led to the creation of a new library which can perform certain operations on Graphs. We, as students of Computer Science, believe that there are umpteen advantages to having a language dedicated to describing and manipulating Graphs. Hence we want to create a language which can be used by Graph Theoreticians, Mathematicians and Computer Scientists to solve computational Graph problems

Description

CRAWL is a language which manipulates Graphs and Nodes as first order primitives. By this, we mean that operations on Graphs and Nodes in CRAWL are treated as fundamental as say C or C++ treats integer and string operations. By this, we hope that people can express their ideas about Graphs and operations on them succinctly and clearly and take advantage of a computer's inherent power in analysing Graphs.

Chapter 2

2.1 A Sample Program in CRAWL

As an example program, we provide the following pseudo code which describes the deadlock detection algorithm

```
ALGORITHM PARALLEL-DEADLOCK-DETECT( ResourceGraphNode, StartNode )

    if ( ResourceGraphNode equals StartNode ) return true
    while ResourceGraphNode.hasNeighbours()
        nodes <- ResourceGraphNode.allNeighbours()
        boolean found = false
        for each node in nodes
            found |= PARALLEL-DEADLOCK-DETECT(node, StartNode)
    return found
```