IPFW

MyIPFWAdvisor
Sugar Interface Documentation

Version 1.0

Revision History

Date	Version	Description	Author
22/2/2012	1.0	Initial version	Trent Forkert

Table of Contents

Sugar Interface Documentation

1. Overview

1.1 Brief Description

The purpose of this document is to describe our interface to Sugar CSP Solver.

1.2 Assumptions

This document is based on a revision of the project named “Advisor” available in one of two ways:

1.2.1 *Git access*

Checkout revision 5a0798fad5c6 from our Git repository [https://bitbucket.org/trentforkert/myipfwadvisor.git]

1.2.2 *Zip archive*

Download the zip file at https://bitbucket.org/trentforkert/myipfwadvisor/get/5a0798fad5c6.zip

2. SugarSupport API

The project contains a class named SugarSupport, which has methods for three useful actions. These methods serve as a wrapper to the actual Sugar API, reducing the work involved in calling Sugar. The methods of this class are static.

2.1 getTempFilename: Get a Temporary Filename

This returns a filename as a string, based on two inputs.

2.1.1 *prefix – the prefix to be used in the filename*

2.1.2 *suffix – the suffix to be used in the filename*

2.2 writeCSP: Write a Constraint Satisfaction Problem

This generates a Constraint Satisfaction Problem in the format Sugar needs, and writes it to a file, according to the parameters:

2.2.1 *student – represents the student the CSP is being generated/solved for*

2.2.2 *reqCourses – represents the courses student is required to take*

This is ultimately defined in the file 'req.txt', which lists course requirements one per line. Each line can specify multiple courses, in which case (at least) one of them will be selected.

2.2.3 *minHours – the minimum number of credit hours a student wishes to take*

2.2.4 *maxHours – the maximum number of credit hours a student wishes to take*

2.2.5 *cspFilename – the filename to write the generated CSP to*

2.3 callSugar: Invoke Sugar

This is where the current codebase tells Sugar to solve a CSP. This method takes two parameters, and returns true if the CSP can be satisfied.

2.3.1 *cspFilename – the filename containing the CSP to be solved*

2.3.2 *sugarOutputFile – the filename Sugar will write its results to*

3. SugarMain API

This class is provided by the developers of Sugar at Japan's Kobe University. It is the direct approach to calling Sugar. This should be avoided unless necessary – prefer the SugarSupport approach for improved maintainability.

3.1 encode: Translate CSP into solvable form

This method takes a few filenames as parameters, and writes a Boolean Satisfiability Testing Problem (SAT) for further computation. The SAT will be solved by another library, detailed in section 4.

3.1.1 *cspFilename – the filename of the CSP file to be encoded*

3.1.2 *cnfFilename – the filename of the SAT file to be generated*

3.1.3 *mapFilename – the filename for mappings between the SAT and the CSP*

3.2 decode: Decode the solved SAT data

This method takes two filenames as input, and prints the decoded solved results. However, in SugarSupport.callSugar, System.out is redirected to the output file specified in that method's parameters.

3.2.1 *satOutputFilename – the filename where the solutions to the SAT are saved*

3.2.2 *mapFilename – the filename to let Sugar map the SAT data back to CSP data*

4. SAT4J

This is the SAT solver referenced in the previous section. Online JavaDoc for the version we are using (v2.2) can be found at http://www.sat4j.org/maven22/org.sat4j.core/apidocs/index.html if it is needed.