Synthesis of UVA CS Handbook Data from a Formalized UVA CS Curriculum

A Technical Report Submitted to the Department of Computer Science

Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Benjamin Barrett
Fall, 2020

Technical Project Team Members
Bugi Abdulkarim
Rahul Batra
Christine Cheng
Abdullah Mahmood

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature ______________________________ Date __________
Benjamin Barrett

Approved [illegible] Date 2020-12-02
Luther Tychonievich, Department of Computer Science

Abstract

This paper introduces the work of the data and handbook team for the Advising Assistant project. Notable technologies developed include: a standardized format for degree requirements, a program to parse students' transcripts, a program to generate a schedule based on constraints, and programs to convert degree requirements into formats usable in the CS handbook, the UVA undergraduate record, and the CS website. These programs are intended to benefit the Advising Assistant project as a whole: an application that students can use to schedule their classes based on their unique preferences and situation.

Introduction

At UVA, the number of CS students has grown rapidly and there has not yet been a comparable increase in the number of CS Advising Faculty. Additionally, the majority of advising time is taken by scheduling concerns. If the scheduling classes task could be automated, advisors would have more time to have conversations with their advisees covering more general topics. Advising Assistant was a project that sought to build a tool to help students schedule their classes. There were two major parts of this project, a data side, and a front-end side. I worked with Bugi Abdulkarim, Rahul Batra, Christine Cheng, and Abdullah Mahmood on the data team. Our goals covered a variety of backend tasks to aid in the creation of the scheduling task: Create some method of understanding and processing a student's current status; Given a student's current goals and classes, generate a schedule that fulfills those requirements as best as possible; Automate the dissimilation of CS course requirements to the CS website, the undergraduate record, and the CS handbook. This last goal was an attempt to ease the burden on professors when changing degree requirements for the major.

To accomplish our goals, we first set out to encode the CS curricula into something readable by both humans and machines. We decided to use the YAML[1] data serialization language as it is more human readable than other formats like JSON. We also created a way to parse students' transcripts to get a record of their courses taken. This is not an easy problem as transcripts are usually a pdf, which does not encode raw text data. The problem of getting the text off the pdf is not perfect, and many different tools exist (Yong, 2018). Additionally, the text needs to be parsed once it's read off the PDF. Using this data, we considered different methods to create schedules satisfying a number of different constraints. This was done via a constraint satisfaction algorithm in python (Kopec, 2019). We also explored automation techniques to facilitate the synchronization of the CS curriculum over three domains: The UVA CS undergraduate student handbook, the UVA undergraduate record, and the UVA CS website. Essentially this entailed programmatically converting the standard YAML format of the curriculum into the relevant output format. For the CS student handbook, there are a variety of output formats that need to be generated. The handbook itself is built in LaTex[2], and the graphs showing requirements are made using DOT. There are a few options to generate LaTex programmatically: using a built-in constructor like LuaTeX (Pégourié-Gonnard, 2012) or using a separate program to write the LaTex directly to the file. For the DOT[3] files, there is no

Architecture

There are three main modules created by the data and handbook team. There is a pdf reader, written in python, that takes a pdf as input and outputs a YAML file containing the class info of a student. This program takes into account classes the student has taken in the past and

[1] https://yaml.org/
[2] https://www.latex-project.org/
[3] https://graphviz.org/doc/info/lang.html

classes the student is taking currently. There is a course recommendation module that generates a schedule based on a given set of requirements and goals. Written in python and using a constraint satisfaction library, the course recommendation module accepts types of constraints like: credit max/min per semester, enemy courses, pinned courses, don't take certain classes at the same time, etc. Some constraints are manually encoded representing prior knowledge and some are only situationally applied based on user input. The last module includes methods to convert the CS curricula from its YAML format into formats for the UVA CS Handbook and the UVA undergraduate record. For the undergraduate record, there is a python script which will convert the Major requirements YAML file into a HTML file. This file is in the format of the undergraduate record's page for the BS in CS.

The second part of the YAML conversion module is code that generates a variety of components in the CS undergraduate handbook. This part of the module is written in Java. There are a variety of classes that are used with the SnakeYAML[4] library to parse the yaml file inputs. The project is then divided into three different functional sections. After parsing the input YAML files, the program has methods to: create a prerequisite graph, a master course list, a requirement checklist, or a list of elective information. These sections correspond to sections in the CS undergraduate handbook. The prerequisite graph is output in a dot format, it has options to include full class titles as well as only include information about CS courses. The graph handles multiple prerequisites by creating "OR" boxes that many classes can point at. The master course list is a list of all courses it can find in the CS courses YAML in the standard LaTeX format. The requirement checklist is a LaTeX figure that contains boxes to keep track of all classes needed for the BS in CS degree. The data needed for the list of elective info is taken from

[4] https://bitbucket.org/asomov/snakeyaml/wiki/Documentation

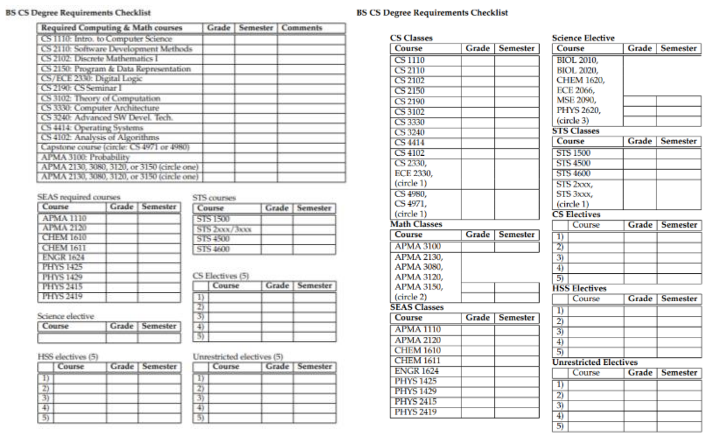
the elective info YAML, this is a LaTeX generation of the text with formatting consistent with the handbook. There is a command line utility that uses some of this functionality. In particular, given some YAML files and the location of the CS handbook github repo, the utility will automatically generate and input the requirement checklist and prerequisite graphs into the handbook .tex file. All that is left to do is compile the handbook repo with LaTeX.

My contributions

I created a standard format to store degree information. I also created a procedure to automatically generate artifacts that express info about the CS degree and various CS classes. The information format we used was the YAML file type. This file type allowed the information to be human and machine readable. We created three different YAML files containing information about: requirements for the BS CS degree, info about CS classes, and CS electives. These data sources were designed to maximize the amount of information while also reducing the amount of parsing required. Using these three data sources, I created a command line utility that would automatically update the CS undergraduate handbook. The program takes as input the yaml files and the location cs-ugrad handbook github repository. Specifically, the utility generates requirement checklists and pre-requisite graphs and inserts them into the undergraduate handbook automatically based on user inputs. The program was built in Java using the SnakeYAML library. Samples of the generated graphs and checklists are shown below. The generated graphs and checklists (As seen in Figure 1 and 2) are not necessarily as compact or informative as the manually generated graphs, but they do still contain all the relevant information.

Future work

In this project, I show it is possible to automatically generate informative graphs with information about the CS degree and automatically add it to the handbook. For future work, this functionality could be expanded to other sections of the handbook. In particular, I believe the format lends itself well to the course description section and sections that are basically just lists without a lot of text. For the BS CS degree, the elective information section could also be automated, though the large amount of text may make that process more cumbersome than helpful. Additionally, the same methods can potentially be expanded for use with requirements for the BA and MS CS degrees.



BS CS Degree Requirements Checklist

Required Computing & Math courses	Grade	Semester	Comments
CS 1110: Intro. to Computer Science			
CS 2110: Software Development Methods			
CS 2102: Discrete Mathematics I			
CS 2150: Program & Data Representation			
CS/ECE 2330: Digital Logic			
CS 2190: CS Seminar I			
CS 3102: Theory of Computation			
CS 3330: Computer Architecture			
CS 3240: Advanced SW Devel. Tech.			
CS 4414: Operating Systems			
CS 4102: Analysis of Algorithms			
Capstone course (circle: CS 4971 or 4980)			
APMA 3100: Probability			
APMA 2130, 3080, 3120, or 3150 (circle one)			
APMA 2130, 3080, 3120, or 3150 (circle one)			

SEAS required courses

Course	Grade	Semester
APMA 1110		
APMA 2120		
CHEM 1610		
CHEM 1611		
ENGR 1624		
PHYS 1425		
PHYS 1429		
PHYS 2415		
PHYS 2419		

STS courses

Course	Grade	Semester
STS 1500		
STS 2xxx/3xxx		
STS 4500		
STS 4600		

CS Electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Science elective

Course	Grade	Semester

HSS electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Unrestricted electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

BS CS Degree Requirements Checklist

CS Classes

Course	Grade	Semester
CS 1110		
CS 2110		
CS 2102		
CS 2150		
CS 2190		
CS 3102		
CS 3330		
CS 3240		
CS 4414		
CS 4102		
CS 2330, ECE 2330, (circle 1)		
CS 4980, CS 4971, (circle 1)		

Math Classes

Course	Grade	Semester
APMA 3100		
APMA 2130, APMA 3080, APMA 3120, APMA 3150, (circle 2)		

SEAS Classes

Course	Grade	Semester
APMA 1110		
APMA 2120		
CHEM 1610		
CHEM 1611		
ENGR 1624		
PHYS 1425		
PHYS 1429		
PHYS 2415		
PHYS 2419		

Science Elective

Course	Grade	Semester
BIOL 2010, BIOL 2020, CHEM 1620, ECE 2066, MSE 2090, PHYS 2620, (circle 3)		

STS Classes

Course	Grade	Semester
STS 1500		
STS 4500		
STS 4600		
STS 2xxx, STS 3xxx, (circle 1)		

CS Electives

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

HSS Electives

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Unrestricted Electives

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Figure 1: Current CS degree requirement checklist (left), and generated checklist (right)

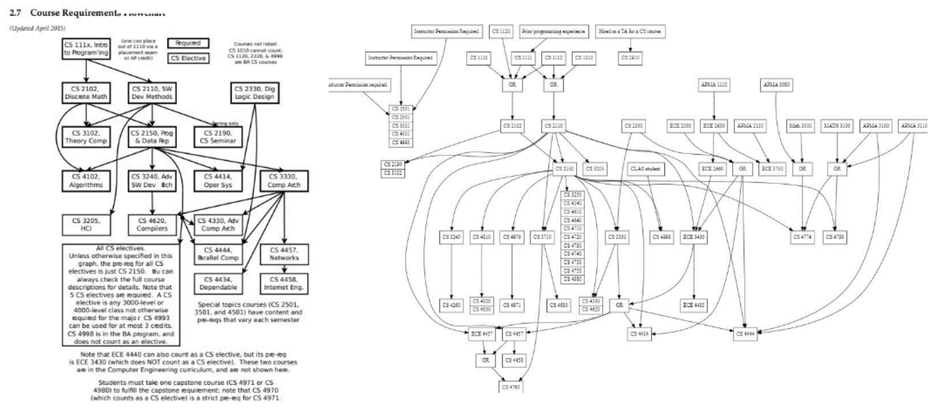

2.7 Course Requirements Flowchart
(Updated April 2015)
CS 111x, Intro to Program'ing
(one can place out of 1110 via a placement exam or AP credit)
Required
CS Elective
Courses not listed: CS 1010 cannot count; CS 1120, 2220, & 4998 are BA CS courses
CS 2102, Discrete Math
CS 2110, SW Dev Methods
CS 2330, Dig Logic Design
Spring only
CS 3102, Theory Comp
CS 2150, Prog & Data Rep
CS 2190, CS Seminar
CS 4102, Algorithms
CS 3240, Adv SW Dev Tech
CS 4414, Oper Sys
CS 3330, Comp Arch
CS 3205, HCI
CS 4620, Compilers
CS 4330, Adv Comp Arch
CS 4444, Parallel Comp
CS 4457, Networks
CS 4434, Dependable
CS 4458, Internet Eng.
All CS electives. Unless otherwise specified in this graph, the pre-req for all CS electives is just CS 2150. You can always check the full course descriptions for details. Note that 5 CS electives are required. A CS elective is any 3000-level or 4000-level class not otherwise required for the major. CS 4993 can be used for at most 3 credits. CS 4998 is in the BA program, and does not count as an elective.
Special topics courses (CS 2501, 3501, and 4501) have content and pre-reqs that vary each semester
Note that ECE 4440 can also count as a CS elective, but its pre-req is ECE 3430 (which does NOT count as a CS elective). These two courses are in the Computer Engineering curriculum, and are not shown here.
Students must take one capstone course (CS 4971 or CS 4980) to fulfill the capstone requirement; note that CS 4970 (which counts as a CS elective) is a strict pre-req for CS 4971.
Instructor Permission Required
CS 1120
Prior programming experience
Hired as a TA for a CS course
Instructor Permission Required
CS 1110
CS 1111
CS 1112
CS 1010
CS 2910
Instructor Permission required
OR
OR
APMA 1110
APMA 3080
CS 1501
CS 2501
CS 3501
CS 4501
CS 4993
CS 2102
CS 2110
CS 2330
ECE 2330
ECE 2630
APMA 2130
Math 3350
MATH 3100
APMA 3100
APMA 3110
CS 2190
CS 3102
CS 2150
CS 3205
CLAS student
ECE 2660
OR
ECE 3750
OR
OR
CS 3250
CS 4240
CS 4610
CS 4640
CS 4710
CS 4720
CS 4730
CS 4740
CS 4750
CS 4753
CS 4980
CS 3240
CS 4810
CS 4970
CS 3710
CS 3330
CS 4998
ECE 3430
CS 4774
CS 4780
CS 4260
CS 4820
CS 4830
CS 4971
CS 4630
CS 4330
CS 4620
OR
ECE 4435
ECE 4457
CS 4457
CS 4414
CS 4444
OR
CS 4458
CS 4760

Figure 2: Current CS course requirement flowchart (left) and generated flowchart (right)

References

Yong, Tien & Azad, Saiful & Rahman, Mohammed & Zamli, Kamal & Rabby, Gollam. (2018). A Highly Accurate PDF-To-Text Conversion System for Academic Papers Using Natural Language Processing Approach. Advanced Science Letters. 24. 7844-7849. 10.1166/asl.2018.13029.

Kopec, D. (2019). Classic computer science problems in python. Shelter Island, NY: Manning.

Pégourié-Gonnard, M. (2012). The luacode package. Retrieved from http://mirrors.ibiblio.org/CTAN/macros/luatex/latex/luacode/luacode.pdf