

cs580Lib – a comprehensive library of Maple objects supporting instruction and research in formal languages and automata theory

Prof. Thomas F. Piatkowski (thomas.piatkowski@wmich.edu)
Prof. J. Donald Nelson (nelson@cs.wmich.edu)

Department of Computer Science
Western Michigan University
Kalamazoo MI 49008

keywords : computer science, education, formal languages, automata theory, collaborative education, research, Maple, cs580, Western Michigan University, software engineering, programming

intended audience : students, teachers, and researchers interested in formal languages and automata theory, the sophisticated use of Maple as a programming language, and the management of a large collaborative learning project

Abstract

cs580Lib is a comprehensive coherent well-planned collection of help pages, procedures, and other Maple objects, currently under creation at Western Michigan University, that supports research and instruction in formal languages and automata theory, in particular the upper-level course: cs580 *Theory of Computation* which uses T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science* (2e), Addison Wesley Longman, January 1998 as its text.

This paper is organized into the following sections:

- History and philosophy of the project
- Technical objectives and project structure and management
- Project status
- Commentary on experiences and future plans
- References

History and philosophy of the project

cs580 *Theory of Computation* is a required course in the Bachelor's and Master's curricula at Western Michigan University. The course assumes a student's familiarity with discrete mathematics (e.g., set theory, relations, functions, equivalence, countability, recursion, induction, graphs, and trees); and develops the mathematical foundations of formal languages as generated by grammars and recognized by automata and as categorized by the Chomsky hierarchy. In recent years, the instruction has been based on T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science* (2e), Addison Wesley Longman, January 1998, augmented by technical material on additional topics such as:

- extended regular expressions — the regular expression operators of *intersection*, *difference*, and *complement*;
- cartesian-style finite-directed graphs;
- a technique of directly constructing a DFA equivalent to any extended regular expression without use of NFAs;
- mnemonic naming conventions for certain entities, such as variables in grammars and states in automata.

For several years prior to last fall, Professor Piatkowski had been giving students the opportunity to use Maple in a major class project. Typically, students would develop a Maple worksheet as a tutorial *living document* in which they document the design and demonstration of a Maple procedure to:

- test if an arbitrary object is a particular standard data type defined for cs580, such as *finite directed graph* or *finite tree*, or
- perform an algorithm associated with cs580, such as *minimizing the state-set of a DFA*, or
- perform a useful utility related to cs580, such as drawing in elegant format a tree graph or DFA transition graph.

These early projects used Maple as a programming language (rather than as an interactive mathematical assistant), and were each fairly independent — each student having to design and implement Maple code that "did it all" (i.e., the student did not take immediate direct advantage of any other student's complementary work).

These initial experiments in using Maple in the cs580 context led to several observations; e.g.:

- Sometimes a student project was really terrific and the result was or nearly was something that could be confidently offered to others for embedded use in other computations.
- In a much clearer way than was revealed in homework exercises or exams, a surprising number of students showed in the projects that:
 - they did not really understand the concepts, mathematics, and/or algorithms associated with cs580;
 - they did not know how to program (or did not know how to break out of the stilted-style of the C-paradigm of programming).
- The Sudkamp text has errors and holes; e.g.:
 - ordered trees are used (in derivation trees, for example) but are never defined;
 - labeled graphs are not defined adequately enough to support their correct use in DFA transition graphs or parsing trees;
 - the state space of PDAs is dealt with inadequately vis-a-vis the atomicity of managing BOTH an external input tape and an internal stack.
- Maple does not have enough to offer in its extensive built-in library to support cs580 applications (both in scope and in the technical style of treating the material).
- There is little in the literature (articles or texts) to support using Maple (or similar symbolic mathematical software packages) in cs580 pedagogy.

In the summer of 2001, the authors decided to embark on a major and much more organized long-term pedagogical development project associated with using Maple in cs580 which would attempt to:

- develop a complete *industrial strength* library of Maple data structures, procedures, and help pages to support education in formal languages and automata;
- engage successive classes of cs580 students in a major effort of coordinated collaborative undergraduate and graduate research and projects.

The name currently used to denote both our project effort and the collection of materials contained in it is **cs580Lib**.

The motivation for undertaking this effort includes:

- The materials included in **cs580Lib** are useful in their own right.
- The activities involved in creating the materials in **cs580Lib** provide an excellent learning opportunity for students of formal languages and automata theory.
- The total project is a good educational exercise in using Maple in an elegant large-scale application... and is, thus, an excellent vehicle for demonstrating software engineering principles and practice.
- The successful project library will enable the assignment to cs580 students of homework and project tasks that are larger and more interesting and realistic than those amenable to manual methods currently employed.

This paper describes the philosophy of our **cs580Lib** undertaking, provides a status report on work accomplished to date, and indicates the planned direction of future efforts.

Technical objectives and project structure and management

The main technical objective of the **cs580Lib** project is to produce a growing, coherent, integrated, high-quality MapleSoft publishable library of

- Maple data structure definitions
- online Maple procedures
- online Maple help pages
- online Maple tutorial worksheets and other documents

that support research and instruction in cs580.

As much as possible, the mathematical objects included in **cs580Lib** will be treated in a technical manner identical to that employed in the course text; e.g.: a *function* from domain X to range Y will be an explicit set of 2-tuples constituting a subset of the cartesian product $X \times Y$, etc.

The main management objective of the **cs580Lib** project is to successfully engage over a sequence of semesters a large group of students in a collaborative effort to develop **cs580Lib** content and in the process to be faithful to the pedagogical objectives of the cs580 course.

The faculty will provide strong technical management guidance that promotes, among other objectives:

- independence of student projects, primarily through modulizing the overall project scope into small interrelated projects
- a rational implementation order on the modules.

The faculty (either as authors or editors) will totally control the online library content, thereby facilitating a high level of programming and expository product quality.

Each student's efforts will be narrowly focused and integrated with the work of other students through the provision by faculty of:

- all needed Maple data structure definitions
- all online help pages — these are complete and robust; for each procedure they define
 - the calling prototype and
 - the functional characteristics
- dependencies relating all help pages (thus inducing the precedence order in which projects should be implemented).

Each student for a project attempts to:

- implement one of the procedures assigned through negotiation with the instructor
 - for which a help page exists and
 - for which no good current implementation is available and
 - for which all predecessor procedures are available; and
- document his/her implementation in a high-quality tutorial Maple worksheet.

Good student work will be edited and incorporated by the **cs580Lib** faculty into the ongoing growing project library.

cs580Lib currently includes:

- a comprehensive set of help files, written in Maple-style, that are accessible through the Maple online browser, and which describe the data structures and procedures associated with the library;
- a set of procedures which are compatible via use of the specified shared data structures of the library;
- a set of graphic utilities that assist in drawing (in Maple) general elegantly formatted
 - finite directed graphs,
 - finite trees,
 - block diagrams, and
 - cartesian state transition graphs;
- a set of tutorial worksheets explaining the implementation of selected procedures;
- a set of guidelines associated with programming and documentation conventions and style;
- a readme file providing status information and installation instructions for the library.

A small glimpse of the scope and style of **cs580Lib** can be had from the following set of lists that presents the names of the defined data structures and procedures in the current version of **cs580Lib**. The subject matter of these objects reflects that priority order of topic implementation began with topics focused on discrete mathematics, strings and languages, regular expressions and grammars.

Currently defined data structures

arc path	node path	relation
CFG derivation	ordered finite tree	string
context free grammar	partition	unordered finite tree
finite directed graph	proposition	
function	regular expression	

Currently defined procedures

Ancestors()	isCFG()	isRLG()
cartesianProduct()	isCFGDerivation()	isRR()
Children()	isCompleteBinaryFT()	isSentence()
concatenate()	isCycle()	isSentential()
convertERToPartition()	isDerivationTree()	isSR()
convertFunctionToTable()	isDerived()	isStrictlyBinaryFT()
convertPartitionToER()	isDerivedOne()	isString()
convertTableToFunction()	isER()	isSubset()
depth()	isF()	isTR()
Descendants()	isFDG()	isUFT()
Domain()	isFL()	minCommonAncestor()
drawFDG()	isLeaf()	outDegree()
drawFT()	isNullArcPath()	parent()
equivalenceClass()	isOneToOneF()	partitionBlock()
frontier()	isPartition()	Range()
generateFDG()	isProperSubset()	ReachableOne()
generateFT()	isProposition()	ReachableZero()
implicitSet()	isR()	reConcatenate()
inDegree()	isRE()	reStar()
isAcyclicFDG()	isREMember()	reUnion()
isArcPath()	isRG()	Siblings()

A feel for the projected size of the final **cs580Lib** project can be had from the measurements and projections presented in the following table.

size of the cs580Lib project	number of pages covered in Sudkamp	number of data structures defined in help pages	number of procedures defined in help pages	number of procedures implemented
at the start of the fall semester 2002	85	13	63	39
projections for a complete one-semester cs580 course	173	34	176	
projections for covering the complete Sudkamp text	540	105	551	

Commentary on experiences and future plans

Some observations deriving from the effort to date include:

- The process of rigorously specifying the detailed data structures for the library has exposed errors and incompleteness in the course text material. (This has been a very important beneficial side effect to this project!)
- Students' Maple worksheets can quickly display if a student really understands the theoretical material involved.
- Maple is an excellent programming language and environment for this kind of application. The programming guidelines and specification style we have adopted conforms in a very natural way to the mathematical constructs in Sudkamp (unlike some of Maple's built-in library data structures and procedures dealing with graphs and sets).
- (Surprisingly) some of our students really don't know how to program.
- Many students (computer science majors) bring a C-language programming style to their work... and this often is clumsy and inappropriate for the Maple environment; e.g.:
 - flag constructs are introduced when conditional functions are more appropriate,
 - unbounded sets and lists are represented as bounded arrays,
 - indices are used as pointers to set or list members instead of directly accessing the members, and these indices are used in controlling loops, etc.
- The word-processor front end of Maple is very difficult to use for sophisticated text formatting.
- The project is quite large and exposes the students indirectly to a number of good software engineering issues, including:
 - implementing in conformance with rigorous semi-formal specifications,
 - testing with respect to rigorous semi-formal specifications,
 - dealing with configuration management concerns,
 - producing high quality implementation documentation.
- The material in the **cs580Lib** project will be used experimentally this fall in our cs661 graduate course, *Software Engineering II: Verification and Validation*. Maple provides a good workable environment in which verification testing, white-box validation, and black-box validation testing can be explored.
- The topics in the **cs580Lib** project can be used as targets in our cs660 graduate course, *Software Engineering I: Requirements and Specification*.
- The current versions of **cs580Lib** are limited to constructs based on finite sets. In the future we would like to extend the functionality of our Maple objects to include infinite sets and constructs built on them.
- At some point we would like to begin illustrating the power and elegance of **cs580Lib** by applying it to several illustrative problems that are larger and more realistic than those treated in the text.

References

J.K. Kurma, *Drawing Finite Directed Graphs and Finite Trees in Maple*, Technical Report TR/01-101, Department of Computer Science, Western Michigan University, Kalamazoo MI, December 6, 2001.

Maple 7.00 Help System, Waterloo Maple, Waterloo ON Canada, 2001.

Maple 7 Learning Guide, Waterloo Maple, Waterloo ON Canada, 2001.

Maple 7 Programming Guide, Waterloo Maple, Waterloo ON Canada, 2001.

T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science* (2e), Addison Wesley Longman, January 1998.

S.M. Vengaiah, *Generating Finite Directed Graph and Finite Tree Objects in Maple*, Technical Report TR/02-10, Department of Computer Science, Western Michigan University, Kalamazoo MI, May 20, 2002.