Grail+

What is Grail+?

Grail+ is a collection of programs for manipulating finite-state machines, regular expressions, and finite languages. Using Grail+, one can input machines, languages and expressions, convert them from one form to another, and output the result. Many other useful operations can be performed - for instance, with a finite-state machine it is possible to minimize, make deterministic, complement, enumerate its language, execute it on a given string, perform union with another machine, complete the machine, test various properties, and add and remove transitions. The newest addition to Grail+ is support for alternating finite automata.

How can Grail+ be used?

Grail+ is written in C++ and can be compiled and run under several different architectures. Many aspects of Grail+'s design are motivated by the philosophy that the product should be as flexible as possible to permit the greatest number and variety of applications. Modularity, expandability and compatibility were important concerns in the design of Grail+.

Here are three examples of ways in which Grail+ can be used:

- *As a set of command line filters* - When compiled, Grail+ consists of a set of 54 programs which can be run from the command line. Machines, expressions and languages can be inputted directly or redirected from files. Filters can be combined by piping the output from one filter to another filter as input. When employed in this manner, Grail+ can be very useful for testing the properties of machines and expressions or learning about formal language objects.

- *As a C++ library* - The functions and classes which exist in Grail+ can be included in any C++ program and used to store or manipulate machines, expressions and languages. Grail+'s code is modular and fully commented which makes it a valuable tool for programming.

- *As a test environment for new algorithms* - The existing Grail+ code can be easily expanded and modified to permit a researcher to test new algorithms or compare performance with existing algorithms. Member functions can be added to Grail+'s classes, filters which use the current classes can be written, or entirely new filters and classes can be designed.

What's new with Grail+?

We are in the process of putting the finishing touches on a new release of Grail+. The most substantial differences from the old version are the implementation of alternating finite automata and commenting of the source code.

The following filters for alternating finite automata will be supported by the new version of Grail+:

- **afacaten** - catenate two afas
- **afacomp** - compute the complement of an afa
- **afaexec** - execute an afa on a given input string
- **afareverse** - compute the reverse of an afa
- **afasize** - determine the size (number of states) of an afa
- **afastar** - compute the star of an afa
- **afatofm** - convert an alternating finite-state state machine to a finite-state machine
- **afaunion** - compute the union of two afas
- **fmtoafa** - convert a finite-state machine to an alternating finite-state machine
- **litafa** - convert an afa from bit-wise representation to literal representation

We are still working on documentation, automated testing for the afa code, and some bug fixes before the official release of the new version of Grail+. However, a pre-release version of Grail+ can be downloaded from our webpage (`http://www.csd.uwo.ca/research/grail`). If you obtain the pre-release, we would be happy to receive your feedback.

How to find out more about Grail+

The Grail+ project is being conducted in the Department of Computer Science at the University of Western Ontario in London, Ontario, Canada.

The Grail+ webpage, `http://www.csd.uwo.ca/research/grail`, contains additional information about the project, code and binaries for downloading, on-line man pages, and links to many other formal language theory resources.

The Grail+ project is currently coordinated by Dr. Sheng Yu (syu@csd.uwo.ca). Any comments, suggestions and inquiries regarding Grail+ are welcome.

The Grail+ Project *2000/02/09*