UACalc: Directions

Ralph Freese

University of Hawaii

Boulder, May 19, 2016

UACalc Web Site: http://uacalc.org/

Source: http://github.com/UACalc

My Web Page: http://math.hawaii.edu/∼ralph

Outline

Outline

Getting it

- Go to **http://uacalc.org** and follow the directions

Getting it

- Go to **http://uacalc.org** and follow the directions
- Source code at **https://github.com/UACalc**

Getting it

- Go to **http://uacalc.org** and follow the directions
- Source code at **https://github.com/UACalc**
- Go to **uacalcsrc**
- Instructions at the bottom (by DeMeo) for
  - viewing the source in your web browser
  - down loading a copy
  - contributing code and pull requests

Outline

Using it

I will demonstrate

- the 3 ways of getting an algebra into UACalc
- switching the Current Algebra
- the Con, Sub and Drawing tabs
- testing Maltsev conditions and other tasks

Outline

Applicataions

This slide isn't big enough to hold them all!

Applicataions

This slide isn't big enough to hold them all!

Well, maybe one example.

Outline

How to contribute

How to contribute

Send money!

How to contribute

Send money! (Lots of it.)

How to contribute

Send money! (Lots of it.)

(small, unmarked bills please)

How to contribute

- Report bugs

How to contribute

- Report bugs
- Request features

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.
  - It's used for testing for an edge term and for an NU term.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.
  - It's used for testing for an edge term and for an NU term.
- Download the source; code your algorithm; send it to me.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.
  - It's used for testing for an edge term and for an NU term.
- Download the source; code your algorithm; send it to me.
  - **M. Valeriote** contributed many things this way including
  - J. Horowitz's algorithm for testing for a k-ary NU in the idempotent case.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.
  - It's used for testing for an edge term and for an NU term.
- Download the source; code your algorithm; send it to me.
  - **M. Valeriote** contributed many things this way including
  - J. Horowitz's algorithm for testing for a k-ary NU in the idempotent case.
- Write code and generate a **git pull request**.
  - See `https://github.com/UACalc/uacalcsrc` for instructions.

How to contribute

- Report bugs
- Request features
- If you have a nice algorithm, write it up and send it to me.
  - **A. Kazda** gave a fast algorithm for testing for a cube term blocker in idempotent algebras.
  - It's used for testing for an edge term and for an NU term.
- Download the source; code your algorithm; send it to me.
  - **M. Valeriote** contributed many things this way including
  - J. Horowitz's algorithm for testing for a k-ary NU in the idempotent case.
- Write code and generate a **git pull request**.
  - See `https://github.com/UACalc/uacalcsrc` for instructions.
  - **M. Maroti** contributed his Mace4 reader this way.

Outline

Viewing the algorithms

Time permitting I'll demonstrate this.

Outline

The future

- Parallel algorithms:

The future

- Parallel algorithms:
  - Fun and interesting to design,

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)
- All types of centrality and commutators and the user interface.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)
- All types of centrality and commutators and the user interface.
- TCT minimal sets and traces.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)
- All types of centrality and commutators and the user interface.
- TCT minimal sets and traces.
- Write an expository paper explaining the algorithms.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)
- All types of centrality and commutators and the user interface.
- TCT minimal sets and traces.
- Write an expository paper explaining the algorithms.
  - and a Manual.

The future

- Parallel algorithms:
  - Fun and interesting to design,
  - but difficult to debug. So we are suspending it for now.
  - Well, we're proceeding with caution.
- Some refactoring (code clean-up and improvement)
- All types of centrality and commutators and the user interface.
- TCT minimal sets and traces.
- Write an expository paper explaining the algorithms.
  - and a Manual.
- **Suggestions ???**