Implementing Distributed Reversible Computation

Peter Browning[1] , Clément Aubert[1]

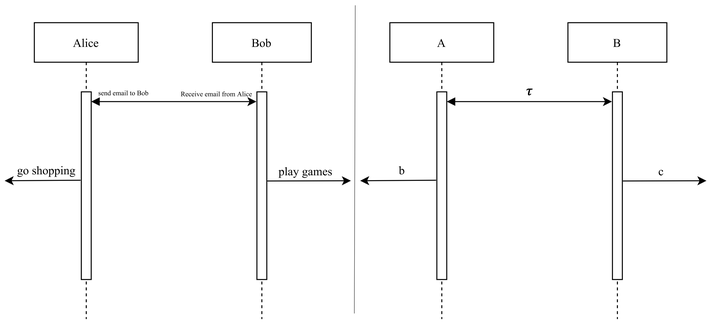
Augusta University[1]

What is it?

- Computers need to communicate
- How can we verify that communicating systems work the way they should?
- Calculus of Communicating Systems (CCS) models their correctness

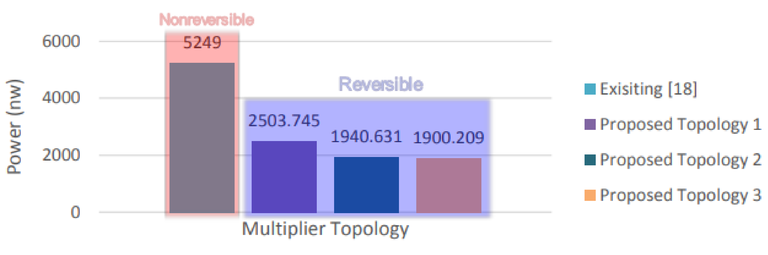
Cat = (sleep + eat)
Person = (walk | talk)
Vending machine = (receive money . give drink)
Alice = (send email . go shopping) → Alice = (a . b)
Bob = (receive email . play games) → Bob = (ā . c)

Represented as a sequential diagram:



Why do we care about reversibility?

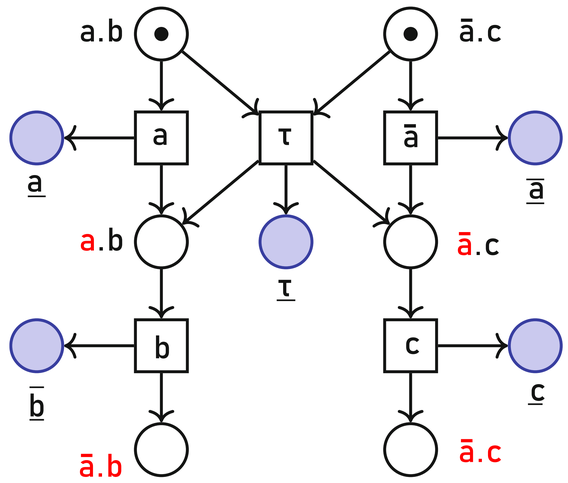
- Capability to reverse cryptographic functions
- “Rewind” a computer that was compromised
- Reversible cryptography
- Computers are natively capable of reverse program flow, removing the need to manually implement it
- Energy efficient adiabatic circuits



Abstracting away complexities

We can unravel processes into Petri nets. In reversible systems, we can observe the following:

(a.b | ā.c)



Implementation

- First public program to implement a reversible form of CCS
- Coded completely from scratch, including string and parsing libraries
- ~3.5k lines of code, fully open source.
- Implements all the core components of CCS(K):

- Operator precedence
- Summation
- Concurrency
- Action prefixing
- Tau synchronizations
- Full-scope key-based reversibility
- Restrictions
- Tokenized processes

```
[debug] Starting parsing of a.b|'a.c
[debug] Begin matching with memory a, counter: 0, set: false
[debug] Found match: a Grammar: LABEL_COMBINED
[debug] Adding prefix: a
[debug] Begin matching with memory b, counter: 0, set: false
[debug] Found match: b Grammar: LABEL_COMBINED
[debug] Adding prefix: b
[debug] Begin matching with memory |, counter: 0, set: false
[debug] Found match: | Grammar: OP_CONCURRENT
[debug] Begin matching with memory ', counter: 0, set: false
[debug] Begin matching with memory 'a, counter: 0, set: false
[debug] Found match: 'a Grammar: LABEL_COMBINED
[debug] Adding prefix: 'a
[debug] Begin matching with memory c, counter: 0, set: false
[debug] Found match: c Grammar: LABEL_COMBINED
[debug] Adding prefix: c
(a.b)|('a.c)
[debug] Checking if a.b can act on Tau{a, 'a}
[debug] Checking if 'a.c can act on Tau{a, 'a}
[Tau{a, 'a}]([Tau{a, 'a}]a.b)|([Tau{a, 'a}]'a.c)
[debug] Checking if [Tau{a, 'a}]a.b can act on b
[debug] Checking if [Tau{a, 'a}]'a.c can act on b
[b]([b]0)|([Tau{a, 'a}]'a.c)
[debug] Checking if [b]0 can act on c
[debug] Checking if [Tau{a, 'a}]'a.c can act on c
[c]([b]0)|([c]0)
```

What’s next?

- New features
- Documentation
- CCS Extensions
- Program Equivalences

Conclusion

- Distributed reversibility is **viable**
- Our program may server as a base for future reversible protocols

You can view the source code here:

SCAN ME

References

Eva Graversen, Iain Phillips, Nobuko Yoshida:Event Structure Semantics of (controlled) Reversible CCS. RC 2018: 102-122

Ivan Lanese, Iain Phillips. Forward-Reverse Observational Equivalences in CCSK. RC 2021 - 13th Conference on Reversible Computation, Jul 2021, Nagoya, Japan. pp.126 - 143, ff10.1007/978-3-030-79837-6_8ff. ffhal-03338669f

N Radha and M Maheswari 2020 J. Phys.: Conf. Ser. 1706 012066

Acknowledgements

Thank you to the following for the help and support:
Dr. Clément Aubert, Dr. St Louis, Dr. Quentin Davis, Augusta University CURS.