Necktie knots
Formal languages
Information security

Mikael Vejdemo-Johansson
KTH Royal Institute of Technology, Stockholm
Jozef Stefan Institute, Ljubljana
Institute for Mathematics and its Applications, Minneapolis

What I hope you will learn today

- I want to demonstrate to you that algebra — especially as a means to abstracting from complex patterns to easy-to-understand systems of smaller combinable pieces — is interesting and has relevant repercussions.
- I want to show you a part of algebra you are not likely to see in a standard undergraduate curriculum.
- I will do this by showing you some of my research on tie knots, and how this ties in with information security and with linguistics.

AR 670-1

Wear and Appearance of Army Uniforms and Insignia

27-19. Neckties, male

c. Necktie, four-in-hand, black, service.

(3) How worn.

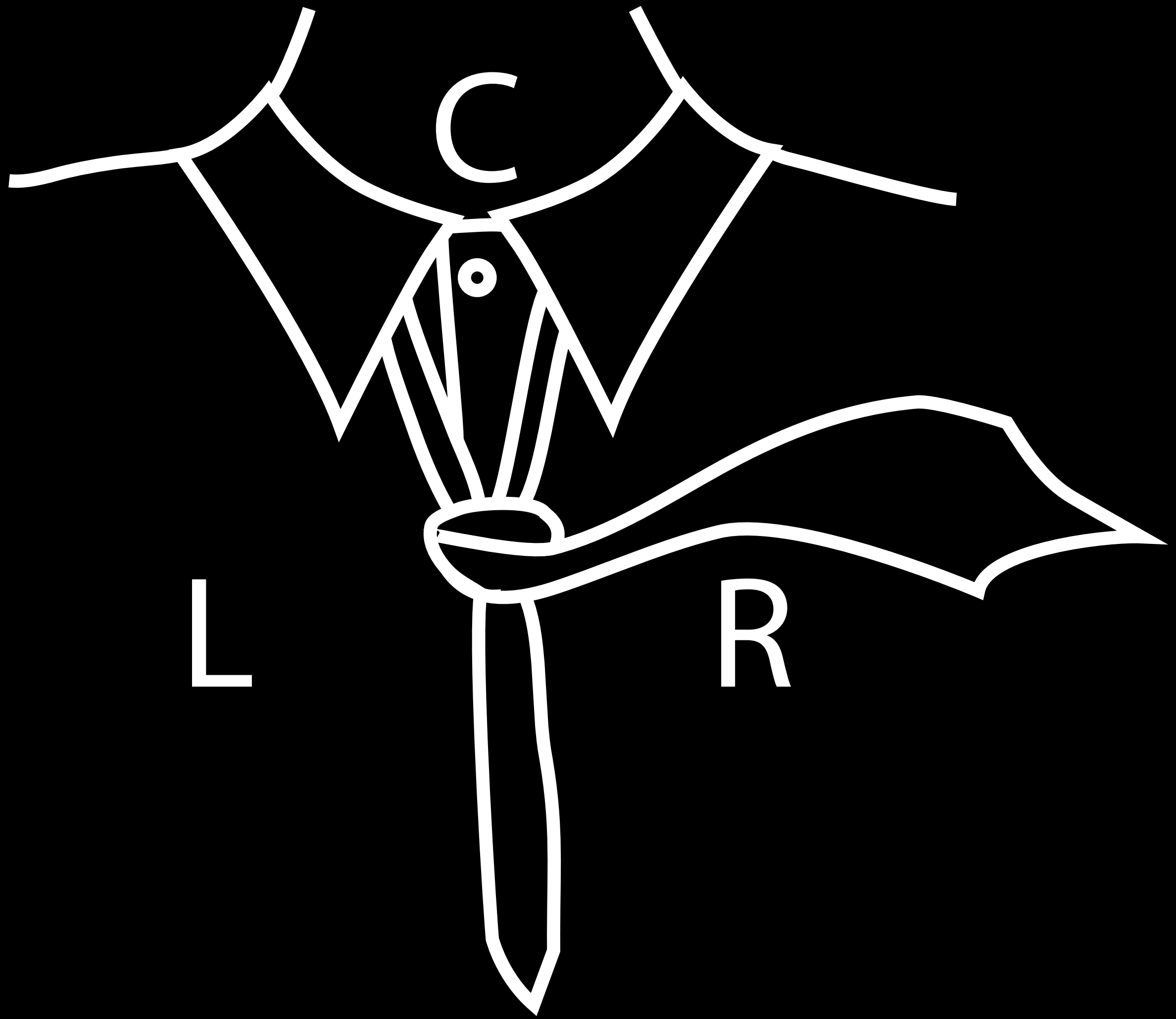
(a) Personnel may wear the tie in a Windsor, half-Windsor, or four-in-hand knot. Use of a conservative tie tack or tie clasp is authorized. The necktie is tied so it is no shorter than 2 inches above the top of the belt buckle, and so it does not extend past the bottom of the belt buckle.

How do you tie your tie?

- In particular: how do you describe how you tie your tie?

Fink & Mao (2000)

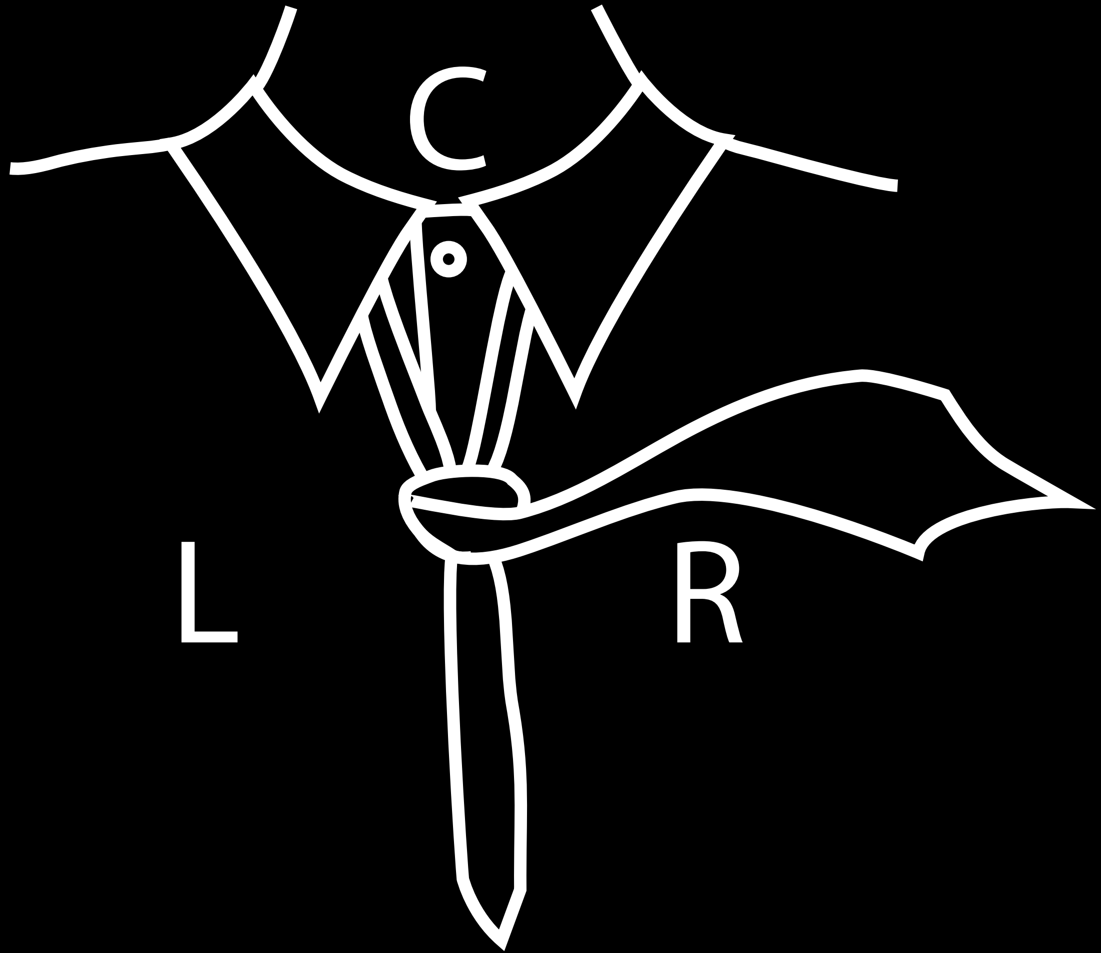
- Cambridge Physicists Thomas Fink and Yong Mao tried to count all possible necktie knots.
- To do this, they invented a mathematical shorthand for describing knots:
- Sequence of *zones* that the tie end goes through



Regulation tie knots

L - Left
R - Right
C - Center
U - tuck Under

4-in-hand	LRLCU
half-Windsor	LRCLRCU
Windsor	LCRLCRLCU

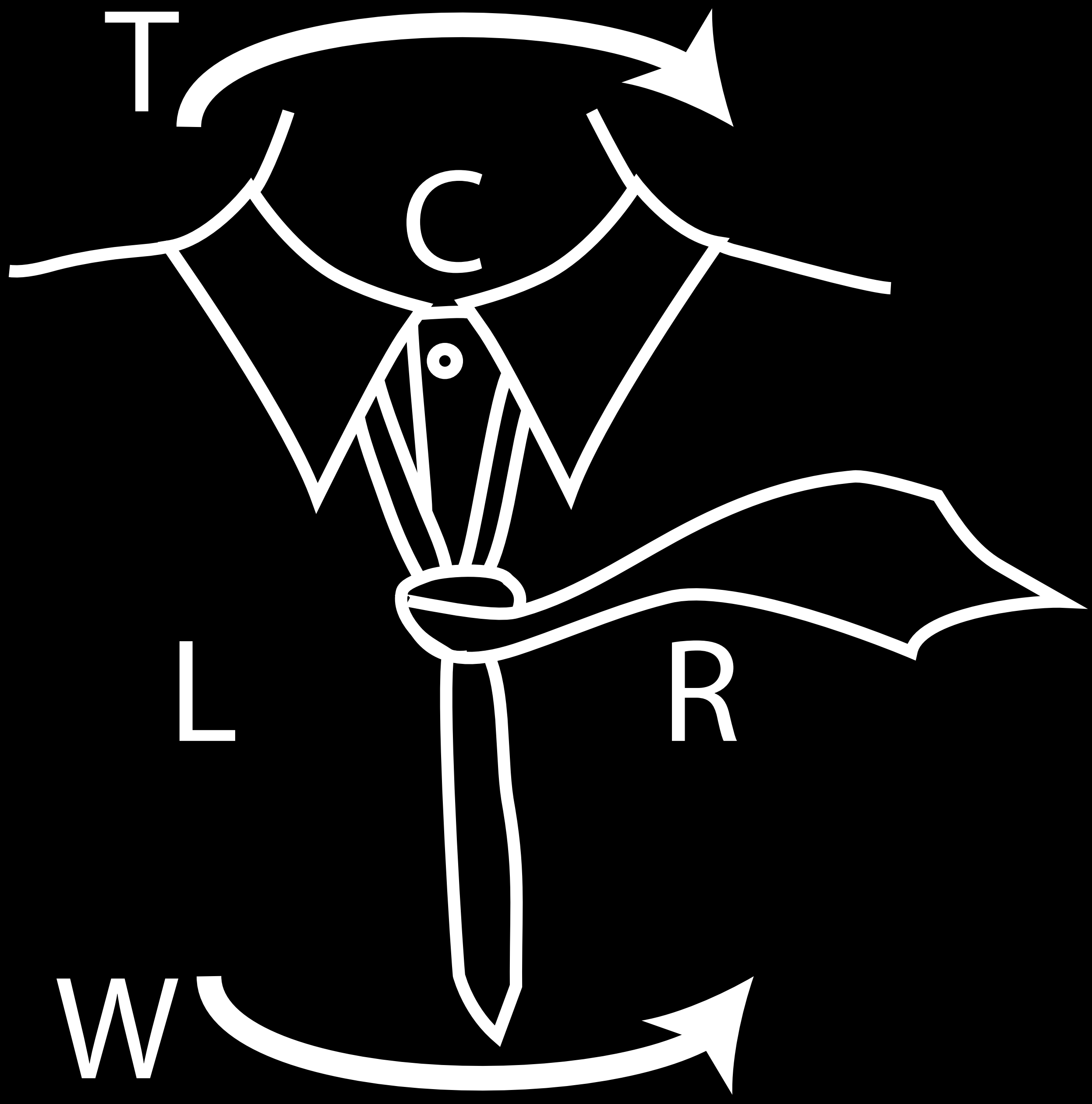


So… how many are there?

- Fink and Mao found 85 tie knots
- Used their symbols + rules
    - Forbidden sequences:
      LL, CC, RR
      inwards after inwards, outwards after outwards
    - Prescribed ending:
      LRCU or RLCU
    - Prescribed length:
      9 moves

More ties than we thought

- In a recent preprint of mine, we modify these symbols and rules
- Instead of L/R/C, use Widdershins (CCW) / Turnwise (CW)
- Inwards/outwards and no repeat regions fixed in symbol choices
- Only remaining rules guide when Under moves are possible

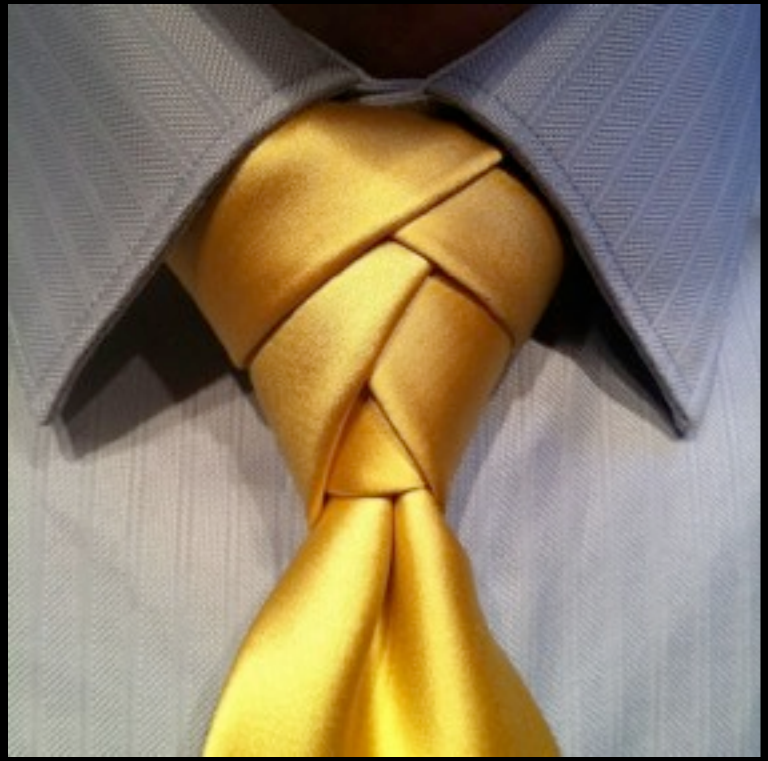


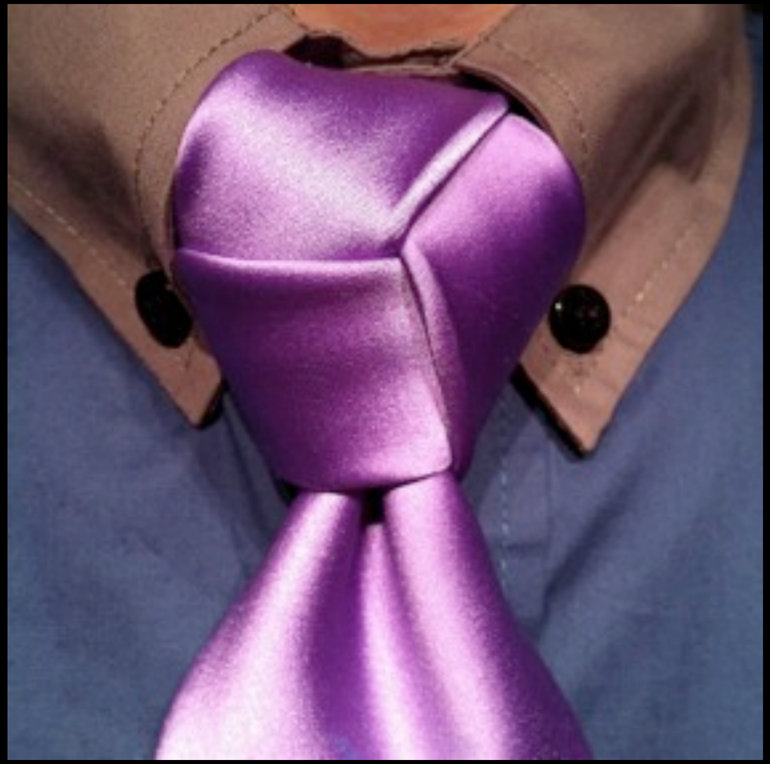
New notation: new knots

Removing assumptions from Fink&Mao opens up for new knots.

Writing down rules and axioms, we can use the algebra of *formal languages* to count the new knots.

We found 174 147 possible new tie knots.


Eldredge
LCRLRCRLUCRCLU
TTTWWTTUTTWWU


Trinity
LCLRCRLCURLU
TWWWTTTUTTU

Try some non-regulation knots

Thin blade active

Edeity	LCLRCRLCU
Trinity	LCLRCRLCURLU
Eldredge	LCRLRCRLUCRCLU
Allwin	LCLRCLRLRCUULRU
Atlantic	LCRLCU

Thick blade active (start inverted)

Kelvin	LRLRCU
Pratt	LCLRCU
Hanover	LRCLRCLRCU
Balthus	LCRCLCRLCU
Oriental	LRCU

UU in the Allwin means go under the 2nd to last bow

Patterns and abstraction

- Mathematics is all about finding patterns and extracting rules
- We see a complex system of things:
  Planetary motion… necktie knot variations… bee hive dances…
  Terracotta soldiers… traffic flow…
- Break the system down into simple pieces.
  Study rules for how to combine these pieces.
- Algebra studies different rulesets for combining simple pieces:
  Vector spaces and matrices for… signal processing… logistics planning… encryption/decryption… geometry… mechanics…

Formal languages: the abstraction of text patterns

- A language is
  - Some *alphabet* of allowed symbols
  - Some collection of allowed *words*
  - Some collection of *grammatical rules*
- Formal languages study the levels of complexity that emerge from rule sets.

Hierarchy of languages

- Noam Chomsky proposed a hierarchy of complexity levels of formal languages:

  Type 0 Unrestricted. Anything you could program. Recognized by Turing Machines.

  Type 1 Context-sensitive. Rules can depend on left- and right-contexts. Recognized by bounded Turing Machines

  Type 2 Context-free. Rules can not depend on contexts. Rules can chain together. Rules can nest in other rules. Recognized by stack machines.

  Type 3 Regular. Rules can not depend on contexts. Rules can chain with strong restrictions. Recognized by state machines.

Regular languages

- Generate extremely fast computer code to generate or recognize.
- Used a lot in search engines: *regular expressions*
- Very restricted; no memory.
- No backtracking in checking validity.
- Can recognize, e.g. {b, ab, aab, aaab, ...}
  Can not recognize, e.g. {ab, aabb, aaabbb, aaaabbbb, ...} or palindromes.

Context-free languages

- Basis for the syntax of many programming languages.
- No backtracking, but has limited memory.
- Can recognize {ab, aabb, aaabbb, aaaabbbb, ...} and palindromes.
  Can not recognize {aba, aabbaa, aaabbbaaa, ...}

Context-sensitive languages

- All natural human languages seem to fall in this category.
- Many fundamental protocols and computer systems fall here. ASN.1, crypto certificates, communication protocols...

Recursively Enumerable languages

- May require the full attention of an infinitely large computer.
- Can express all computable things.
- Might not be computable with finite resources.

So where do the tie knots fit in?

- What do you think?
    - Regular?
    - Context-free?
    - Context-sensitive?
    - Recursively enumerable?

So where do the tie knots fit in?

- If we limit the length of the tie?
  Regular
- If we limit the number of steps back (UU et.c.) we can take?
  Regular
- The unrestricted language of infinitely long infinitely thin neckties?
  Either context-free and context-sensitive

Undecidable problems

- Some questions can be proven not to have answers always computable in finite time:
    - Does this Turing Machine ever stop? (the Halting problem)
    - Does this context-sensitive language have any valid strings?
    - Are these two context-free languages the same?
- If there are variations in the language *designed* and the language *implemented* we might not be able to accurately predict all possible results. There are *weird machines* where any bug or malformed input can disrupt the system — including exploits or security holes.

Security ramifications

- Computer communications rely on tightly specified protocols
  TCP/IP, Email (SMTP), Web (HTTP), encryption (SSL/TLS)…
- If the implementation of a protocol uses too weak a parser, it will actually implement a *different* protocol.
  If the equivalence of languages is undecidable, two different implementations might actually implement *different* protocols.
- Kaminsky, Patterson and Sassaman (2010): showed that one can play out differences in protocol implementations to produce arbitrarily bad encryption certificates. The internet trust networks cannot be trusted…
  We could, e.g., get the means to impersonate www.amazon.com or www.usma.edu or www.army.mil without the browser warning for the deception.

Do you want to know more?

- www.langsec.org for formal languages based security research
- tieknots.johanssons.org for some pointers on tie knot languages and a random tie knot generator

  The research here has been joint work with:
  Anders Sandberg, Oxford University
  Meredith L Patterson and Dan Hirsch, Upstanding Hackers LLC