Finite Computational Structures and Implementations

Attila Egri-Nagy
www.egri-nagy.hu, twitter: @EgriNagy, GitHub:egri-nagy

国際教養大学
Akita International University

CANDAR'16
4th International Symposium on Computing and Networking
Hiroshima, Japan, November 22-25, 2016

Bad news There are still many important aspects of computation that we do not fully understand.

- *What is computable with limited resources?*
- *How can we verify the correctness of computations?*
- *How to measure computational power with precision?*

Good news We can leverage the available computational power to gain deeper understanding of computation.

Abstracting from models of computation

Turing-machine making the tape finite $\longrightarrow$ finite state automata

Finite state automata remove the distinction between initial, acceptor and ordinary states $\longrightarrow$ transformation semigroup

Transformation Semigroup remove distinction between state and event $\longrightarrow$ abstract semigroup (multiplication table)

In short:

computers – semigroups
implementations – structure preserving maps

FSA

Definition

By a finite state *automaton*, we mean a triple $\mathcal{A} = (X, \Sigma, \delta)$ where

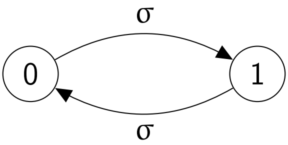
- X is the finite *state set*,
- Σ is the (finite) *input alphabet*,
- $\delta : X \times \Sigma \to X$ is the *transition function.*

No output function, initial and acceptor states are defined, we rather consider FA to be discrete dynamical systems.

Mod-2 counter

Example

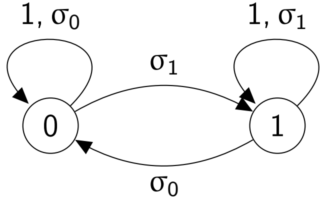
states $X = \{0, 1\}$, operations (input symbols) $\Sigma = \{\sigma\}$



Flip-flop

Example

states $X = \{0, 1\}$, operations (input symbols) $\Sigma = \{\sigma_0, \sigma_1, 1\}$

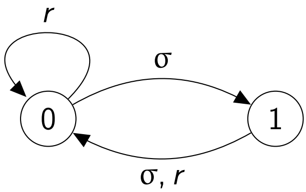


σ_0, σ_1 are resets, constant maps

Full Transformation Semigroup $\mathcal{T}_2$

Example

states $X = \{0, 1\}$, operations (input symbols) $\Sigma = \{\sigma, r\}$



r is a reset

How many finite state automata with a given number of states?

Seemingly simple question, not easy to answer.

Strategy: find a 'universal' automaton with n-states and find all sub-automata inside $\implies$ it is a search problem.

Semigroups

ABSTRACT

Definition

A *semigroup* is a set S with an associative binary operation $S \times S \to S$.

Example (Flip-flop monoid)

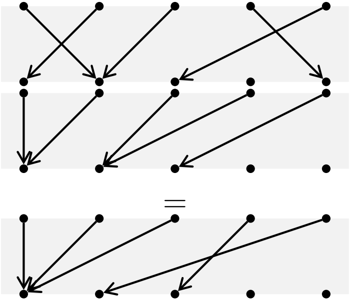
	1	a	b
1	1	a	b
a	a	a	b
b	b	a	b

TRANSFORMATION

Definition

A *transformation semigroup* (X, S) is a set of states X and a set S of transformations $s : X \to X$ closed under function composition.

Example (Transformations)



The current state is x, then event s happens changing the state to y.
Function notation:

$$y = s(x)$$

Operation sequence notation:

$$xs = y$$

Principle (State-event abstraction)

We can identify an event with its resulting state: state x is where we end up when event x happens.

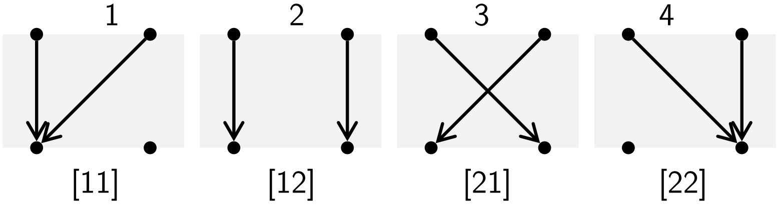
Action interpretation: $xs = y$ can be understood as event s changes the current state x to the next state y.
Event algebra notation: $xs = y$ can also be read as event x combined with event s yields the composite event y .

$\mathcal{T}_2$ the full transformation semigroup on 2 points (states)

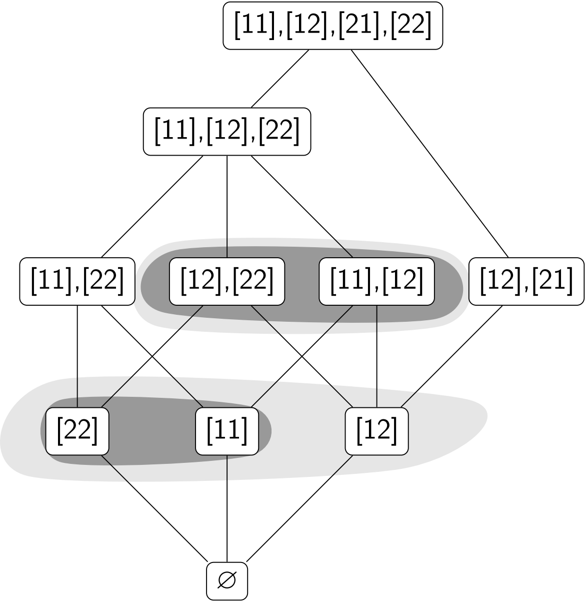
Abstract semigroup

	1	2	3	4
1	1	1	4	4
2	1	2	3	4
3	1	3	2	4
4	1	4	1	4

Transformation semigroup



Enumeration: subsemigroups of the full transformation semigroup $\mathcal{T}_2$



Enumerating finite state automata

Number of subsemigroups of full transformation semigroups.

	#subsemigroups	#conjugacy classes	#isomorphism classes
$\mathcal{T}_0$	1	1	1
$\mathcal{T}_1$	2	2	2
$\mathcal{T}_2$	10	8	7
$\mathcal{T}_3$	1 299	283	267
$\mathcal{T}_4$	3 161 965 550	132 069 776	131 852 491

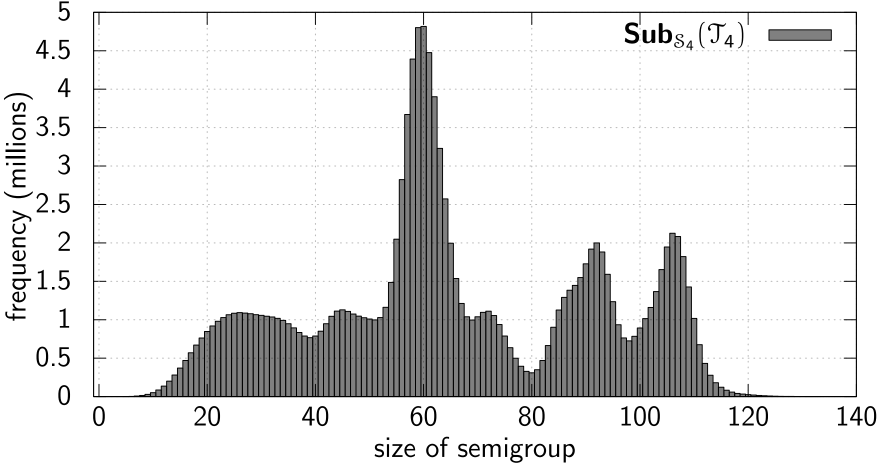


Figure: The six peaks of $\mathcal{T}_4$. The main bulk of the size distribution of transformation semigroups of degree 4.

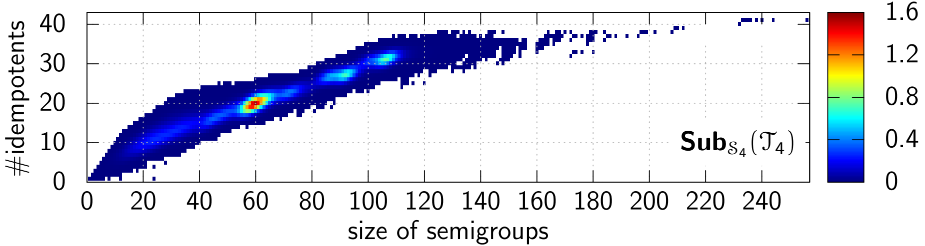


Figure: Size versus the number of idempotents for transformation semigroups up to degree 4. Frequency values in millions.

Minimal degree problem

Given an abstract semigroup, what is the minimal number of states we need to realize it as a transformation semigroup?

This is different from FSA minimization, where we want to keep the recognized language the same but reduce the number of states.

First attempt: try to embed the abstract semigroup into a full transformation semigroup of degree n, $\mathcal{T}_n$.

Emulation/Implementation algebraically

Definition (Isomorphic relations of computational structures)

Let S and T be computational structures (semigroups). A relation $\varphi : S \to T$ is an *isomorphic relation* if it is

1. homomorphic: $\varphi(s)\varphi(t) \subseteq \varphi(st)$,
2. fully defined: $\varphi(s) \neq \varnothing$ for all $s \in S$,
3. lossless: $\varphi(s) \cap \varphi(t) \neq \varnothing \implies s = t$

for all $s, t \in S$. We also say that T *emulates*, or *implements* S.

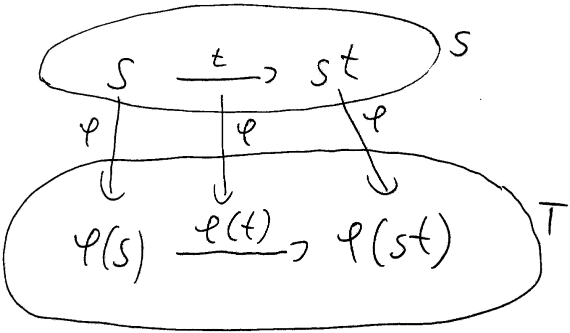
Embedding

Isomorphism into a substructure.

$\varphi : S \to T$

$$\varphi(s)\varphi(t) = \varphi(st)$$

where $\varphi(S) \subseteq T$



Finding embeddings is done by a specialized (partitioned) backtrack search algorithm.

Embedding $\mathcal{T}_n$ into $\mathcal{T}_{n+k}$

$\mathcal{T}_m \overset{k}{\hookrightarrow} \mathcal{T}_n$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
$m = 1$	1	2	3	5	7	11
$m = 2$	0	1	3	12	35	110
$m = 3$	0	0	1	4	17	64
$m = 4$	0	0	0	1	2	6
$m = 5$	0	0	0	0	1	$\geqslant 2$
$m = 6$	0	0	0	0	0	1

Table: Number of embeddings of full transformation semigroups.

Embedding the trivial monoid is equivalent to finding idempotent elements ($e^2 = e$) up to conjugation in the target semigroup.

3-state machines implemented as 4-state and 5-state machines

There are 282 non-empty transformation semigroups on 3 points up to conjugation, and 132069775 on 4 points.

How many isomorphic copies of the degree 3 transformation semigroups can we find inside $\mathcal{T}_4$?

Counting the embeddings up to conjugation we find only 2347 subsemigroups of $\mathcal{T}_4$ isomorphic to some subsemigroup of $\mathcal{T}_3$; so most degree 4 transformation semigroups are 'new'.

Calculating the same number for $\mathcal{T}_5$ yields 18236; a modest increase compared to the still unknown, but expected-to-be gigantic, number of degree 5 transformation semigroups.

Software packages:

SubSemi	`github.com/gap-packages/subsemi`
Kigen	`https://github.com/egri-nagy/kigen`

Host systems/languages:

GAP `www.gap-system.org` Groups, Algorithms, Programming – a System for Computational Discrete Algebra

Clojure general purpose Lisp-like language with immutable data structures for the JVM `clojure.org`

Non-conclusion

Ongoing investigations on the semigroup level:

1. exploring the space of possible computations;
2. measuring finite computational power;
3. computational correctness.

Thank You!

www.egri-nagy.hu
compsemi.wordpress.com
github.com/egri-nagy/
twitter.com/EgriNagy