Computation and Discrete Mathematics

Klaus Sutner
Carnegie Mellon University
Fall 2024

1. **Administrivia**

2. **CDM, the Idea**

3. **Example: Equivalence**

Dramatis Personae

- Prof:
  Klaus Sutner sutner@cs.cmu.edu
- TAs:
  Nicholas Kocurek nkocurek@andrew.cmu.edu
- Course secretary:
  Rosie Battenfelder rosemary@cs.cmu.edu

Website and Communication

Course Website: http://www.cs.cmu.edu/~cdm

Communication: Ed

Syllabus: Syllabus

Make sure to read the course syllabus carefully, I will assume you are familiar with all the rules and policies spelled out there. If you feel the instructions are not clear enough, talk to me or post on Ed.

Prerequisites

There is only one official prerequisite: 15-251.

You should interpret this as meaning: "utterly comfortable with all the 251 material, and with math in general."

There is a lot of material posted on the website, if you need to brush up on some particular topic (say, Turing machines or coding functions) take a look at this stuff. And ask if there are any questions.

A Deliberate Provocation

The discovery of the theory of computation and the evolution of digital computers is the single-most important development in mathematics in the last century.

Are You Insane?

What about Hilbert, Poincaré, Gödel, von Neumann, Eilenberg/MacLane, Erdős, Grothendiek, Perelman, Serre, Wiles, ...

Sure, enormous and often locally unexpected progress has been made in the last 100 years. But, ever since Leibniz and Newton, mathematics has been on an ever accelerating trajectory. Without trying to belittle anything, this is to be expected.

And von Neumann, arguably the most brilliant of them all, spent considerable energy on developing practical digital computers (together with Alan Turing, another giant).

Theorems vs Mathematics

More importantly, math is not just a list of theorems, even the most spectacular ones.

There is a fairly large infrastructure that supports the whole effort.

- knowledge management
- communication and cooperation
- numerical/symbolic computation
- examples/counterexamples
- proof checking

All of these have changed changed completely (a paradigm shift).

Thurston

> The standard of correctness and completeness necessary to get a computer program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.
>
> Bill Thurston

Fields Medal 1982, used computers extensively in his seminal work on low-dimensional topology. Supposedly taught himself to visualize 4-dim objects.

> I can't see how else it will go. I think the process will be first accepted by some small subset, then it will grow, and eventually it will become a really standard thing. The next step is when it will start to be taught at math grad schools, and then the next step is when it will be taught at the undergraduate level. That may take tens of years, I don't know, but I don't see what else could happen.
>
> Vladimir Voevodsky

In response to question about computer verified/generated proofs.

Fields Medal 2002. Astonishing connections between Martin-Löf type theory and classical homotopy theory.

> CS is the new "new math," and people are beginning to realize that CS, like math, is unique in the sense that many other disciplines will have to adopt that way of thinking. It offers a sort of conceptual framework for other disciplines, and that's fairly new.
>
> The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics.
>
> Bruno Chazelle

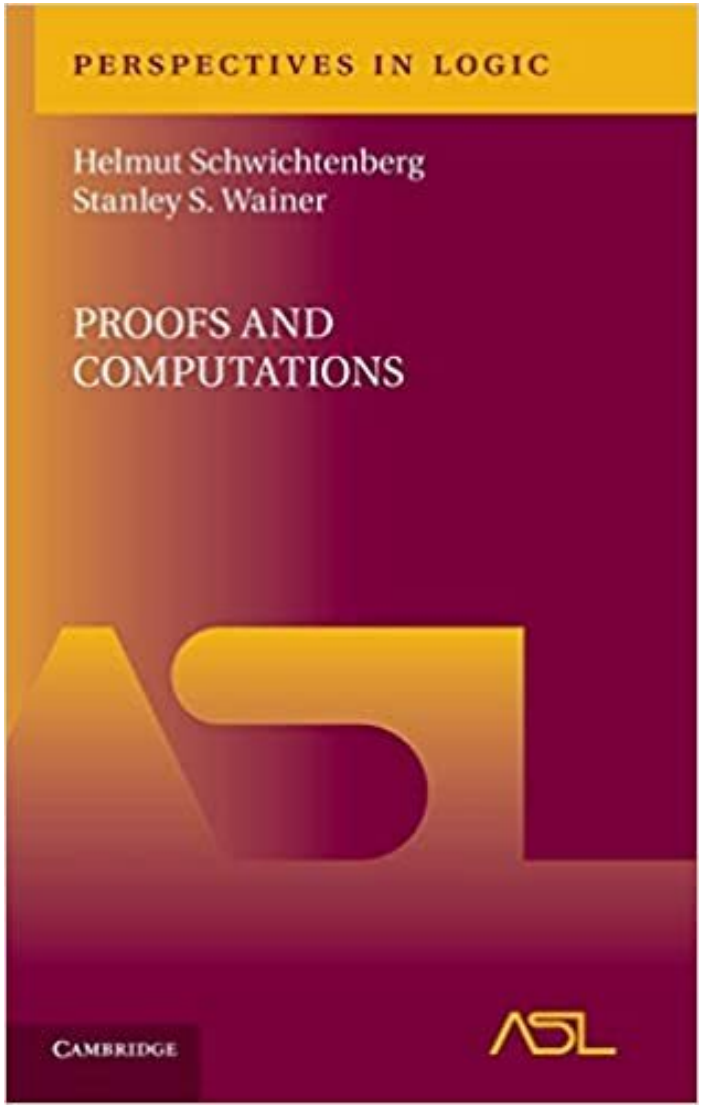
Princeton and Collège de France, used computational geometry to explain the behavior of bird flocks.

Quine

> The utterly pure theory of mathematical proof and the utterly technological theory of machine computation are at bottom one, and the basic insights of each are insights of the other.
>
> W. V. O. Quine

Expressed a bit more directly: you cannot really understand proofs without understanding computation, and vice versa.

No, Seriously



> Generally, computer science, that no-nonsense child of logic, will exert growing influence on our thinking about the languages by which we express our vision of mathematics.
>
> Yuri Manin

This is closely connected to Chazelle's algorithmic thinking idea: computers are used in just about all fields of human discourse, and their use will inject the mathematical structure of computation and computers into these fields.

So What's The Problem?

Most current presentations of math in computer science ignore computation and are very tepid on logic. Including those that are supposedly custom designed for computer science majors, for the most part you get some rehash of the old Bourbaki style with a bit of combinatorics and graph theory thrown in; plus a bit of bad pseudo-code.

This is easy to check: take a textbook and mark all the parts that could easily have been written 50 years ago, before the computer revolution really took off. Make sure you have lots of markers at hand, you'll need them.

Never mind everything that has happened in the last 50 years. This sort of lag is not such a serious problem in a mature field like math, but in a rapidly developing area like CS one needs to be more careful.

CDM is simply an attempt to take these comments seriously and perhaps fix a few of the problems mentioned.

- Focus on logic and computation.
- Emphasize material that is most relevant for the CS/math connection.
- Use modern computational tools to help developing insights, search for examples/counterexamples, not waste time on routine computations, ...

Learning Style

This may sound like we are going to be concerned mostly with a rather formal and abstract development. Nothing could be further from the truth.

Far and away the most important challenge is for you to develop your

intuition

Technical details and formalization are necessary and indispensable, but intuition comes first—by a long shot. Don't even think about formalizing anything without first having developed a good intuitive grasp.

This holds for definitions, theorems, proofs, algorithms, data types.

Concepts and Definitions/Theorems

intuition understand the concept's meaning, its purpose, its intent

examples some objects that the definition applies to

counterexpl some objects where it does not, but almost

formalize pin things down in a semi-formal way, a definition

computation understand the computational aspects (if any)

results the basic theorems associated with the concept

links connections to other concepts

Proofs

intuition understand the objective precisely, develop a battleplan

formal refine the argument to a semi-formal level

examples what does the proof say about some concrete objects

counterexpl what goes wrong if we change hypotheses/conclusions

computation understand computational implications (if any)

results how does the proof help to clarify the given assertion

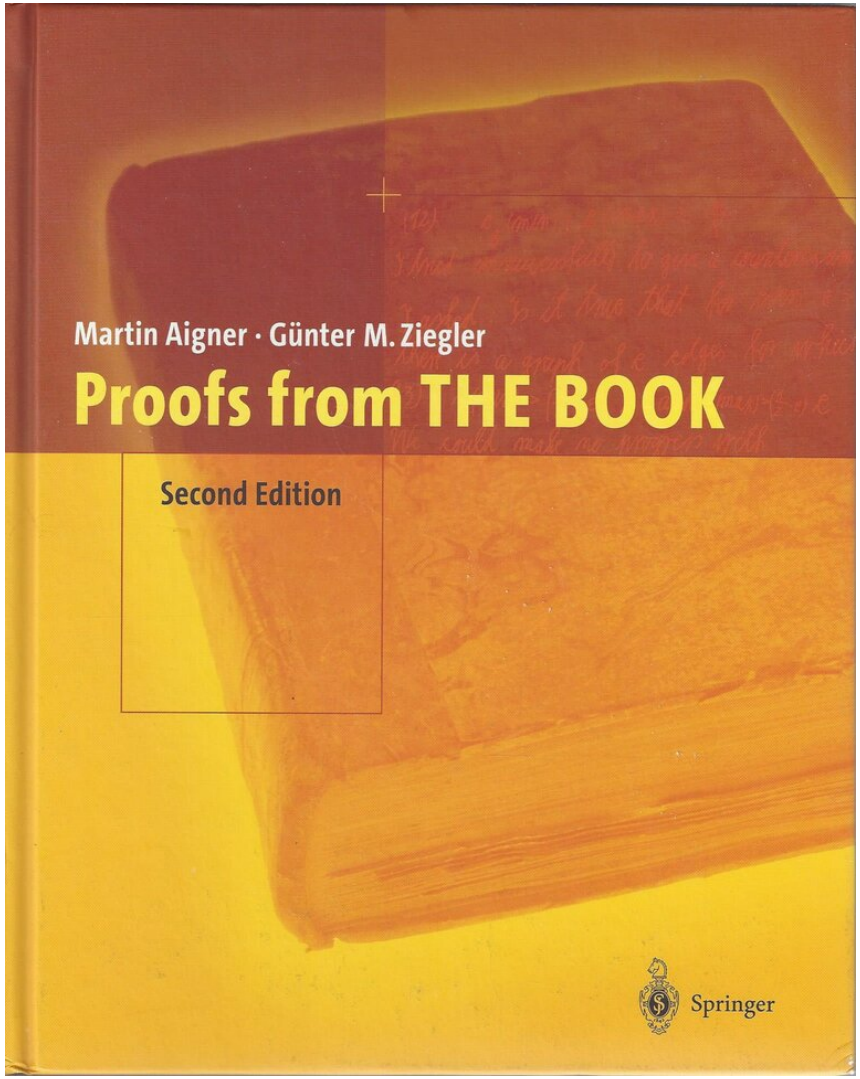
Good Proofs

The last item is particularly important: ideally, a good proof provides additional insights into the claim; it shows not just that it is true, but why it is true.

This is one of the reasons why it may be interesting to have multiple proofs for the same theorem—they all can shed additional light on the subject.

By a formal proof I will always mean an argument that can be verified by a proof checker. Such formal proofs are quite closely connected to our subject (one can think of proofs as computations), but we will never write out formal proofs. Instead, we use pseudo-formal proofs, proof sketches, sketches of proof sketches–exactly the stuff you are already used to.

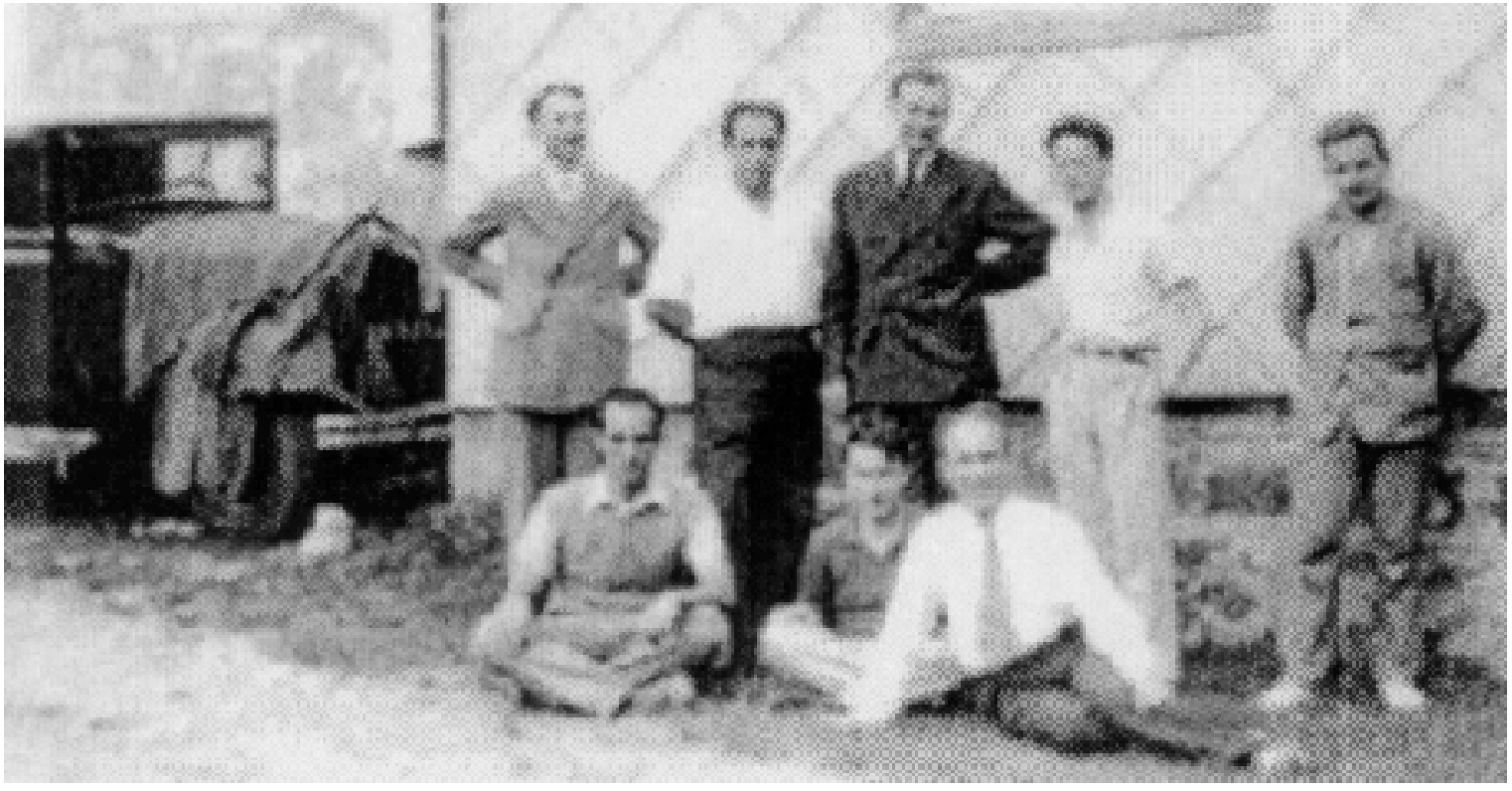
The Book of Proofs



Martin Aigner, Günter Ziegler
Proofs from THE BOOK
Springer 2014 (5th ed.)

The Culprit

Bourbaki, Charles Denis Sauter



Cartan, Dieudonné, Weil, Chevalley et al., 1935

Supposedly, this all started when Henri Cartan was badgering André Weil to produce a text from which would allow him to teach Stokes' theorem completely rigorously[†].

Thousands of pages were written, but the goal was never accomplished. And somehow the scope shifted to "all of mathematics," rather than just poor Stokes' theorem.

Rigor is certainly an invaluable goal, but, from a modern perspective, Bourbaki misses the mark, by a few light-years. The exposition is pseudo-formal at best, it uses a dubious logical foundation and there certainly is no proof checker in sight anywhere.

[†] According to S. S. Chern, Stokes' theorem is the only theorem in analysis.

Some Bourbaki Idiosyncrasies

- Logic is treated minimally, and badly.
- Problem solving is secondary to axiomatics.
- Combinatorial structure is non-essential.
- Algorithmic content is off-topic.
- Applications nowhere in sight.
- Presentation strictly linear, no external references.
- And, cela va sans dire, no pictures.

So What?

Why should anyone care? Just some math guys having innocent fun.

Because Bourbaki is still considered by many as the Holy Grail of Mathematics, even subconsciously.

This is actually a rather peculiar state of affairs. There are many omissions in Bourbaki's work: probability, mathematical physics, logic, category theory. Moreover, the style clashes head on with more recent developments. Just take a look at Knuth's *The Art of Computer Programming* or efforts to reconstruct math in the LEAN theorem prover.

Math Education

The most damaging impact of Bourbaki is probably in the world of math education. Their approach to math is highly idiosyncratic, yet it has became the more-or-less unchallenged framework for all of math education in the middle of the last century.

In other words: All the math classes you have ever taken are Bourbaki's fault, more or less.

Very, very few students benefit from this; for computer science majors in particular, it is nothing short of a catastrophe.

If $R\{x, y\}$ is both symmetric and transitive it is said to be an *equivalence relation* (with respect to the letters x and y). In this case the notation $x \equiv y \pmod{R}$ is sometimes used as a synonym of $R\{x, y\}$; it is read "*x is equivalent to y modulo* R". If R is an equivalence relation, we have $R\{x, y\} \Longrightarrow (R\{x, x\}$ and $R\{y, y\})$, because $R\{x, y\}$ implies $R\{y, x\}$, and $(R\{x, y\}$ and $R\{y, x\})$ implies $(R\{x, x\}$ and $R\{y, y\})$ by virtue of the definitions.

¶ Let $R\{x, y\}$ be a relation; it is said to be *reflexive on* E (with respect to the letters x, y) if the relation $R\{x, x\}$ is equivalent to $x \in E$. If there is no possible ambiguity about E, one says simply, by abuse of language, that R is reflexive.

¶ An *equivalence relation on* E is defined to be an equivalence relation which is reflexive on E. If $R\{x, y\}$ is an equivalence relation on E, we have $R\{x, y\} \Longrightarrow ((x, y) \in E \times E)$, hence R has a graph (with respect to the letters x, y). Conversely, suppose that the equivalence relation $R\{x, y\}$ has a graph G. Observe that the relation $R\{x, x\}$ is equivalent to the relation $(\exists y)R\{x, y\}$; for the former implies the latter (Chapter I, § 4, no. 2, scheme S5), and conversely, since $R\{x, y\}$ implies $R\{x, x\}$, $(\exists y)R\{x, y\}$ implies $(\exists y)R\{x, x\}$ and therefore also $R\{x, x\}$. Thus $R\{x, x\}$ is equivalent to $x \in \mathrm{pr}_1 G$, and hence R is an equivalence relation on $\mathrm{pr}_1 G$.

Equivalence Relations

An equivalence relation is supposed to express similarity between objects, a sort of generalized identity. For example, we may want to identify congruent triangles or triangles with the same area. Or functions with the same rate of growth.

To formalize this notion, one uses a binary relation $\rho \subseteq A \times A$ that satisfies the following conditions, written in pseudo first-order logic:

reflexive	$\forall\, x \in A \,(x\ \rho\ x)$
symmetric	$\forall\, x, y \in A \,(x\ \rho\ y \Rightarrow y\ \rho\ x)$
transitive	$\forall\, x, y, z \in A \,(x\ \rho\ y \wedge y\ \rho\ z \Rightarrow x\ \rho\ z)$

Equivalence Relations

One usually writes $[x]$ or $[x]_\rho$ for the equivalence class of $x \in A$, and $A/\rho = \{\, [x]_\rho \mid x \in A \,\}$ for the quotient set. The cardinality of A/ρ is the index of ρ (in particular in the finite case).

Alternatively, one can also employ a calculus of relations:

reflexive	$I_A \subseteq \rho$
symmetric	$\rho^{\mathsf{op}} \subseteq \rho$
transitive	$\rho \circ \rho \subseteq \rho$

Partitions

Alternatively, we can consider partitions of A, subsets $A_i \subseteq A$, $i \in I$, that are non-empty, pairwise disjoint and cover A:

- $A_i \neq \emptyset$
- $i \neq j \Rightarrow A_i \cap A_j = \emptyset$
- $\bigcup_{i \in I} A_i = A$

The A_i are called the blocks of the partition.

The Connection

Proposition

For any equivalence relation on A and $x, y \in A$:

$$[x]_\rho \cap [y]_\rho = \emptyset \quad \textit{or} \quad [x]_\rho = [y]_\rho$$

Hence, we can think of A/ρ as a partition, a set of blocks.

So we can interpret an equivalence relation as a partition.

And Back

On the other hand, given a partition (A_i), we can define an equivalence relation ρ on $A = \bigcup A_i$:

$$x \; \rho \; y \Leftrightarrow \exists \, i \, (x, y \in A_i)$$

Lemma

We have two mutually inverse operations:

$$\text{toblk} : \textit{equivalence relations} \longrightarrow \textit{partitions}$$
$$\text{toequ} : \textit{partitions} \longrightarrow \textit{equivalence relations}$$

Application: Counting

It is natural to ask how many equivalence relations there are on a set of cardinality n.

It is far easier to do this by counting partitions, rather than using the original definition. These numbers are called Bell numbers B_n. For $n = 0, \ldots, 10$ the values are

$$1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$$

Note that $B_0 = 1$ according to our definitions.

There is no nice closed form for B_n, but we have the following recurrence.

Lemma

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

Proof

Write ℓ for the number of blocks, so $1 \leq \ell \leq n$ for $1 \leq n$.

Suppose we have a partition $(A_i \mid i \in [\ell])$ of $[n+1]$.

We may safely assume that $n{+}1 \in A_\ell$. Let $|A_\ell| = k+1$ for $0 \leq k \leq n$.

But then $(A_i \mid i \in [\ell{-}1])$ is a partition of a set of cardinality $n-k$.

By induction, the number of such partitions is B_{n-k}. Clearly, the number of choices for block A_ℓ is $\binom{n}{k}$.

Done by summation. $\square$

Computational Content

We can extract a (pretty lousy) recursive algorithm for generating all partitions from the proof.

The exponential generating function for the Bell numbers

$$\sum \frac{B_n}{n!}\, x^n = e^{e^x - 1}$$

Asymptotics are quite tricky, a fairly simple result is

$$B_n < \left(\frac{.792\, n}{\ln(n+1)} \right)^n$$

Digression: Set Theory

The proof argument really leans heavily on (baby) set theory: one defines all the basic mathematical concepts (relation, equivalence relation, partition, block, natural number, cardinality, ...) purely in terms of set theory.

One can then construct very precise proofs such as the last one leaning on an axiomatic system of set theory, say, Zermelo-Fraenkel with Choice. Most of the axioms are perfectly reasonable and intuitive, so almost everyone would accept such an argument as compelling.

Alas, if every math concept is translated into sets, some weirdness can result and produce some amount of cognitive dissonance.

Implementation

Generic CS Question:
How would one go about implementing some math concept?

The definition as a set of pairs translates directly into a possible implementation: we can think of an equivalence relation as a container type whose elements are pairs, a record type.

Using the characterization as partitions, we could also store a list of blocks, using containers of containers.

Exercise

Compare these implementations. How useful would they be?

Digression: Efficiency

For those of you who cringe at the thought of implementing an equivalence relation by a hashtable-of-hashtables, remember Knuth's dictum:

> Premature optimization is the source of all evil.

Having said that, you have to be able to put on your hacker goggles when necessary.

Cue Java horror story.

Going Infinite

Standard data structures work only for finite relations. What should we do with infinite relations?

A relation is a subset of a Cartesian product:

$$\rho \subseteq A \times B$$

which is "the same" as a map

$$\widehat{\rho} : A \times B \longrightarrow \mathbf{2}$$

So we could implement a function that, on input $(a, b) \in A \times B$, returns true/false depending on whether $(a, b) \in \rho$.

This is just GANS (general abstract nonsense), but functions actually are hugely useful in computing with equivalence relations.

In many cases, equivalence of two objects is determined by measuring some particular property of the objects: length, area, number of components, acceptance language and so on.

We can model this by a function $f : A \to B$.
Define a binary relation on A, the kernel relation K_f of f, by

$$x \, K_f \, y \iff f(x) = f(y)$$

It is easy to see that K_f is always an equivalence relation: in essence, we just use the fact that equality is an equivalence relation.

If the function f is easily computable, we have a way to implement K_f without direct use of any container data structures: a program that computes f is will do.

This works for infinite carrier sets, at least in principle. For example, a function $f : \mathbb{N} \to \mathbb{N}$ could be used to represent an equivalence relation on the naturals.

Exercise

Discuss how this succinct representation differs from the more traditional data structure approach when the carrier set is finite.

Innocent Question:
Are all equivalence relations kernel relations?

Sure, use the function $f(x) = [x]_\rho$.

That works, but it is not really a good answer: the codomain of f is the power set of A, computationally we'd like to avoid power sets if at all possible.

Better Question: Can we use a map $f : A \to A$?

Heavy Guns

The natural approach to finding a map $f : A \to A$ for an equivalence relation ρ is to pick a particular representative $x' \in [x]_\rho$ for all equivalence classes.

We can then define $f(x) = x'$.

This sort of function is called a choice function or a selector. Alas, to do this in general, we need the Axiom of Choice. This axiom is fairly uncontroversial at this point, Zermelo-Fraenkel with Choice (ZFC) is the standard foundational system in math.

Bourbaki strikes again. It is interesting that most mathematicians would point to ZFC as the background theory that all their work is grounded in, without actually knowing in detail exactly what the axioms are.

The Axiom of Choice

Here is one way to describe this axiom:

Definition (Axiom of Choice)

For any family $(A_i)_{i \in I}$ of pairwise disjoint, non-empty sets, there is a choice set C such that $|C \cap A_i| = 1$ for all i.

Here I is an arbitrary index set and there are no additional constraints on the A_i beyond non-emptiness.

To construct our choice function, given a choice set C for the family of blocks, we can simply define

$$f(x) = y \iff [x]_\rho \cap C = \{y\}$$

Axiom Schmaxiom

Question: Can't we just ignore this GANS kind of stuff?

In mathematics, absolutely not.

Theorem (Ash, 1973)

The additive groups $\langle \mathbb{R}, + \rangle$ and $\langle \mathbb{C}, + \rangle$ are isomorphic iff the Axiom of Choice holds.

OK, but how about just plain computer science?

My claim is, the answer is still a loud **NO**. GANS is directly important for programming language semantics and verification, but even ignoring this part, one can still learn a lot from abstract ideas.

Oh, Come On

If the damn axiom is so important, why don't we just assume it's true and be done with it, once and for all?

> The Axiom of Choice is obviously true,
> the Well-Ordering Principle obviously false,
> and who can tell about Zorn's Lemma?
>
> Jerry Bona

Logic sometimes coexists very uneasily with psychology.

Theorem (Banach-Tarski Paradox (AC))

The unit sphere can be decomposed into finitely many pieces, which pieces can be reassembled to form a sphere of radius 2.

This sounds utterly wrong: the volume won't work out. Alas, the pieces in the Banach-Tarski decomposition are very strange and cannot be visualized. In particular we cannot assign qualities such as "volume" to these pieces.

So, this result is a correct theorem of ZFC, but it is very, very counterintuitive.

Projecting Down to Earth

The sets we encounter in discrete math and CS typically carry a natural order. But then we can define a canonical choice function

$$f(x) = \min [x]_\rho$$

In set theory, this requires a well-ordering to make sure the minimal element exists. In practice, this is not an issue, everything we touch in CS is well-ordered (famous last words).

For example, congruence modulo 4 on $[10]$ produces the canonical classifier

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	1	2	3	4	1	2	3	4	1	2

So we can implement f as an integer array with very fast $O(1)$ equivalence test.

Going Dynamic

Our canonical representation is good, but what if the equivalence relation is being generated dynamically, so we don't yet know what the least element of $[x]$ is (a smaller representative might pop up later)[†]?

More precisely, suppose we are performing some construction that discovers elements $x, y \in A$ that have to be equivalent. If we already have $f(x) = f(y)$, there is nothing to do.

But what if currently $a = f(x) \neq b = f(y)$? We could set $f(y) = a$, but that is not enough: we need to update all z such that $f(z) = b$.

Easy to do and correct, but not efficient.

[†]This is not the type of question typically discussed in combinatorics.

A Brilliant Trick

We can still use a selector-like function $f : A \to A$, but relax the conditions a bit:

$$x \ \rho \ y \iff \mathsf{FP}(f, x) = \mathsf{FP}(f, y)$$

Here $\mathsf{FP}(f, x)$ is the (unique) fixed point on the orbit of x under f. Of course, we have to make sure that this fixed point really exists.

The drawback is that an equivalence test is no longer $O(1)$: it depends on the length of the transients under f. This is the key for optimizations, see below.

Construction

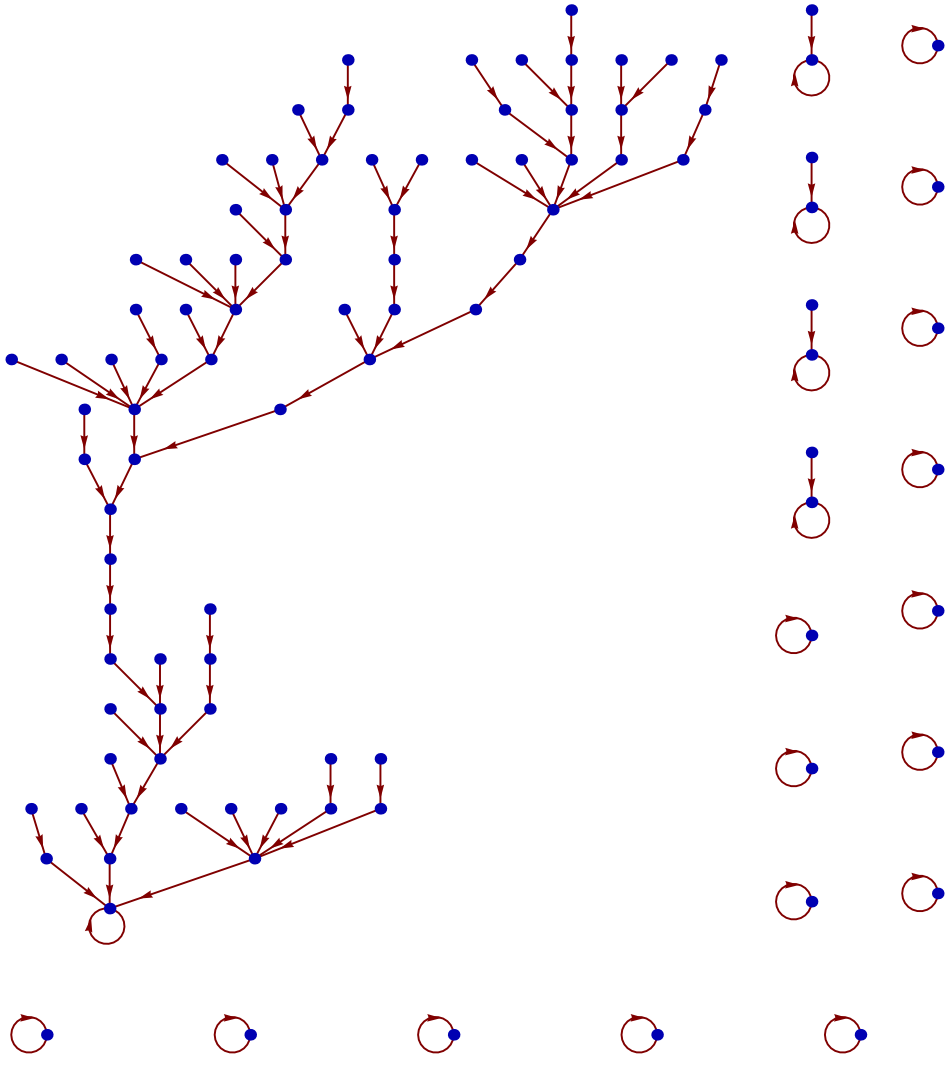
Initially, we set $f(x) = x$ for all x: every equivalence class consists of just the element itself.

If a new equivalence between x and y is discovered, we proceed almost in the same way as for the canonical selector:

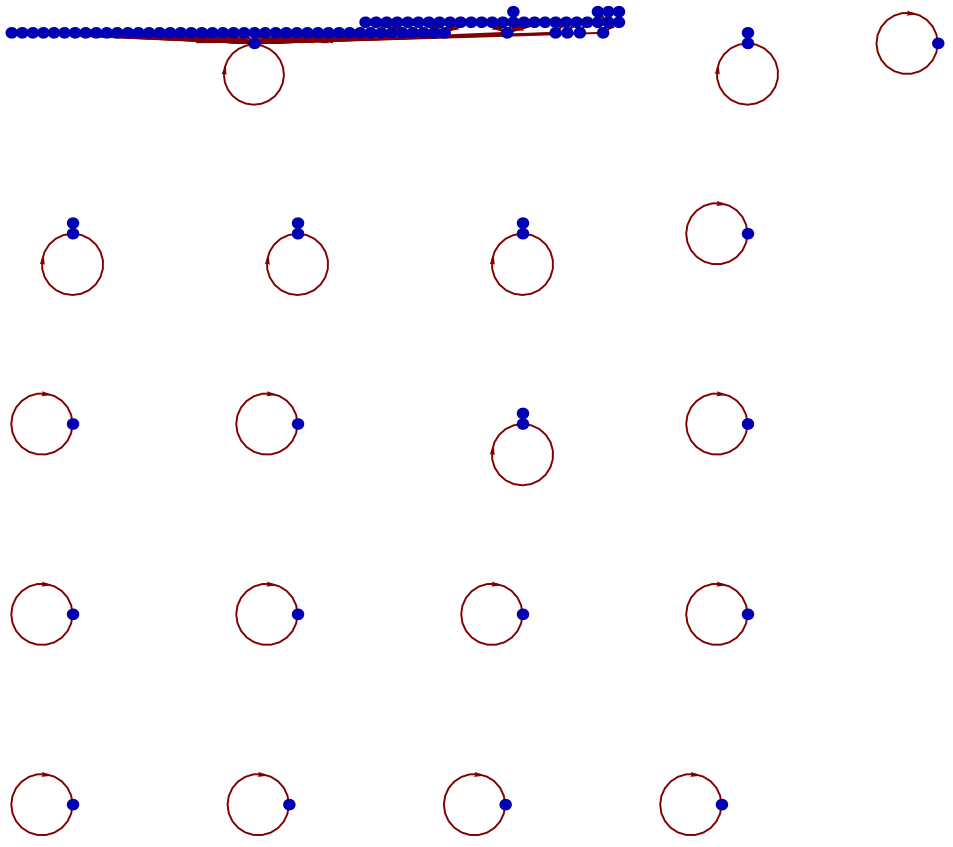
- compute $a = \mathsf{FP}(f, x)$ and $b = \mathsf{FP}(f, y)$
- set $f(b) = a$

We only update one function value, but we have to compute fixed points first. The only obvious bound for traversing the transient part of an orbit is $O(n)$. Still, things look mildly promising.

A Random Relation on $[100]$



Optimization



Using natural hacks like path compression and ranking.

Exercise

Read up on the Union-Find algorithm and implement it.
Compare the running time to the non-optimized version above.

Exercise

It is easy to foil the non-optimized version.
Can you foil the full Union-Find method?

Exercise

Read up on the running time analysis of Union-Find.

Visualization

While we are looking at pretty pictures, here is another way of plotting relations.

We can think of an endorelation ρ on A as an $A \times A$ Boolean matrix B_ρ:

$$B_\rho(a, b) = 1 \iff a \ \rho \ b$$

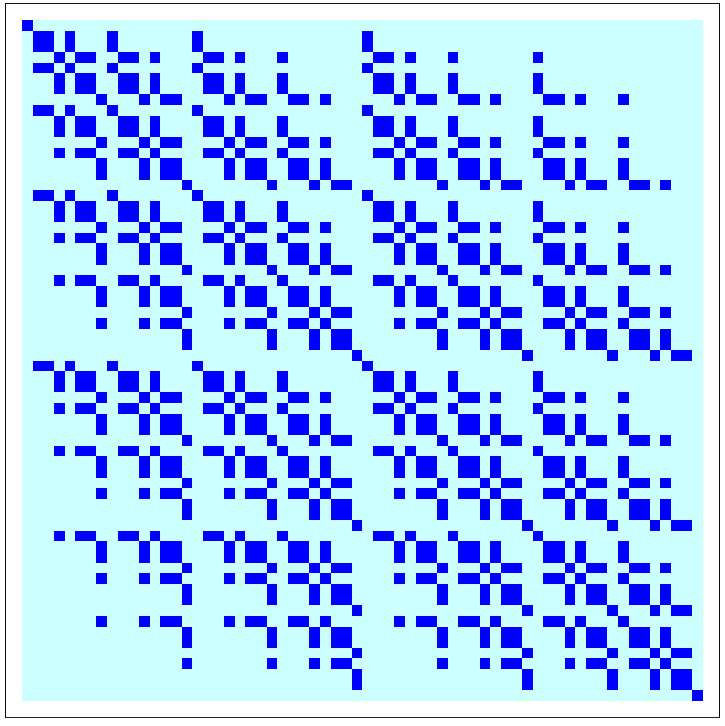
Fine, but to get an $A \times A$ Boolean matrix we need a map $A \times A \to \mathbf{2}$ together with a *total order* on A. In the finite case, we want to think of A as a list

$$A = a_1, a_2, \ldots, a_{n-1}, a_n$$

so that we are dealing with an $n \times n$ matrix.

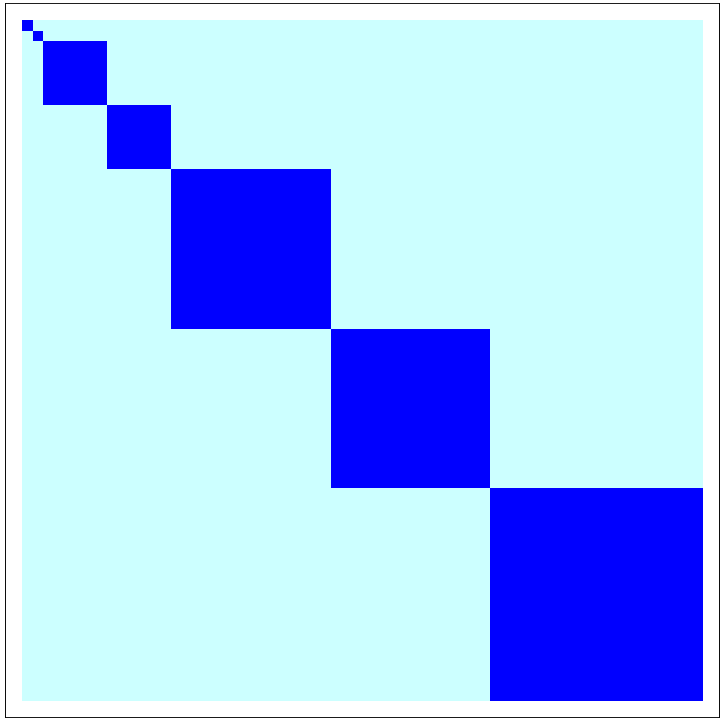
Bad Picture

Consider the equivalence relation on $\mathbf{2}^n$ induced by the map $f(x) = \#_1\, x$. Here is a picture for $n = 6$.



The picture clearly shows reflexivity and symmetry, but transitivity is utterly opaque.

Good Picture



The first picture used standard lexicographic order. By adjusting the order of the carrier set, we get the “right” picture.

Exercise

What is the order used in the last picture?
Is there another order that might be more natural?

Exercise

Define $x, y \in \mathbf{2}^n$ to be equivalent if x can be cyclically rotated to get y.
What would the nice picture look like for $n = 6$?