Say No to Case Analysis: Automating the Drudgery of Case-Based Proofs

Jeffrey Shallit

School of Computer Science, University of Waterloo
Waterloo, ON N2L 3G1 Canada
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit/

Case Analysis

Some proofs proceed by tedious case analysis (the *method of exhaustion*).

For example, here is part of Figure 61 (!) in the proof of the four-color conjecture by Appel-Haken-Koch in 1977.

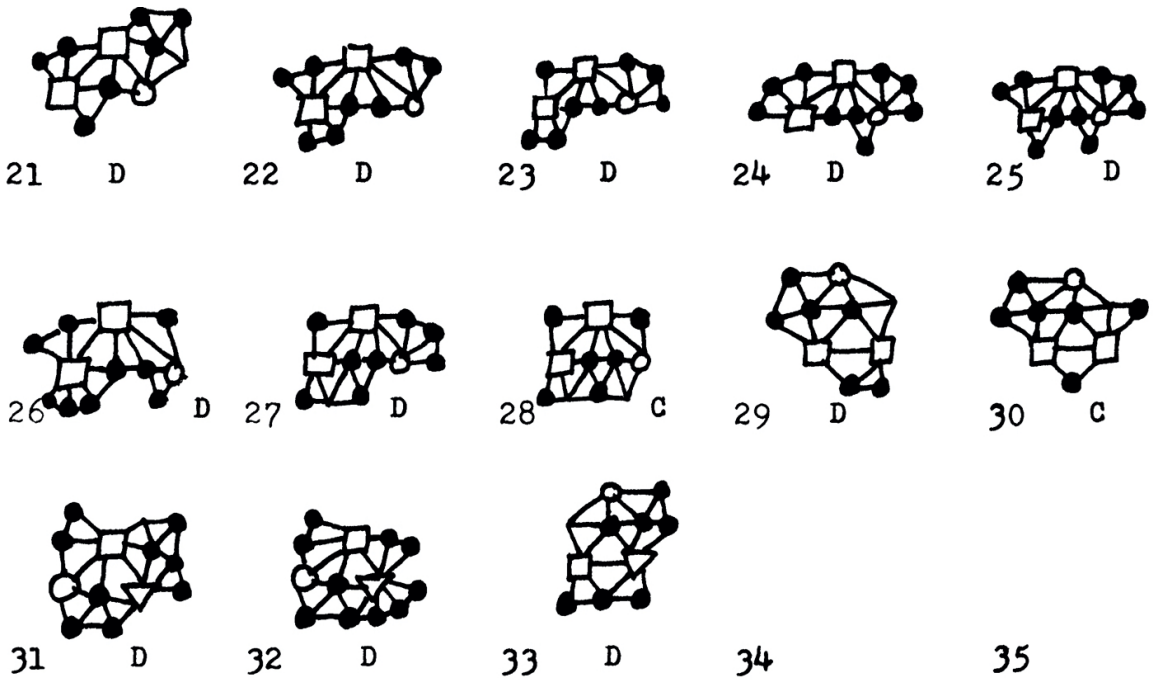


Figure 61

Case Analysis

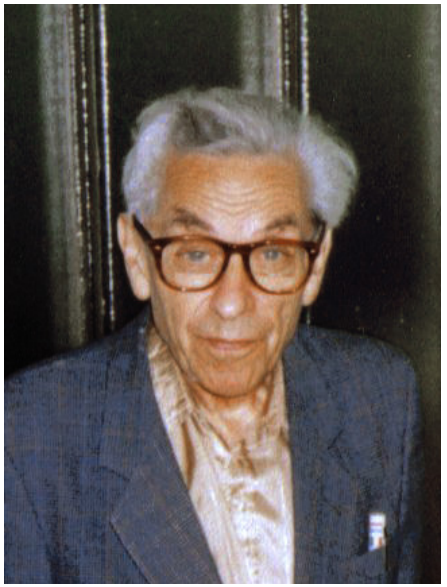
Other examples:

- There exists no finite projective plane of order 10 (Lam-Thie-Swiercz, 1989) (2000 hours of time on a Cray)
- Kepler's conjecture about sphere packing in three dimensions (Hales, 1998): no packing of equally-sized spheres can beat cubic packing in density (250 pages, 3 gigabytes of programs)
- Boolean Pythagorean triples problem (Heule-Kullmann-Marek, 2016): cannot avoid $\mathbf{x}[a] = \mathbf{x}[b] = \mathbf{x}[c]$ for a, b, c satisfying $a^2 + b^2 = c^2$ in infinite binary words $\mathbf{x}$ (4 CPU years on supercomputer, 200 terabyte proof)

Much as we might like to, we can't avoid the need for case analysis...

Sometimes Things are True for No Good Reason

Paul Erdős (1913–1996) liked to talk about *The Book*, the place where a supreme being has recorded the most elegant proof of every mathematical theorem.



But we know this is a fantasy because

- Some true statements have no proofs
- Some true statements have no *short* proofs

Some True Statements Have No Proofs

For example, "This statement has no proof in Peano arithmetic."

Nicer example: Kirby-Paris version of Goodstein sequences: write m as a sum of powers of n. Write each exponent as the sum of powers of n. Repeat with exponents of exponents until a number $< n$ is reached. Then $G_n(m)$ is the number produced by replacing every n in the representation with $n+1$ and then subtracting 1.

Sequence starts with m, $G_2(m)$, then $G_3(G_2(m))$, then $G_4(G_3(G_2(m)))$, etc.

Some True Statements Have No Proofs

Example of Goodstein sequence:

$$\begin{aligned}
266 &= 2^8 + 2^3 + 2^1 = 2^{2^3} + 2^{2^1+1} + 2^1 = 2^{2^{2^1+1}} + 2^{2^1+1} + 2^1 \\
G_2(266) &= 3^{3^{3^1+1}} + 3^{3^1+1} + 3^1 - 1 = 3^{3^{3^1+1}} + 3^{3^1+1} + 2 \doteq 10^{38} \\
G_3(G_2(266)) &= 4^{4^{4^1+1}} + 4^{4^1+1} + 1 \doteq 10^{616} \\
G_4(G_3(G_2(266))) &= 5^{5^{5^1+1}} + 5^{5^1+1} \doteq 10^{10921} \\
&\vdots
\end{aligned}$$

Theorem. Every Goodstein sequence eventually ends at 1.

Not provable in Peano arithmetic! But provable in second-order arithmetic.

Sometimes Things are True for No Good Reason

Physicist Freeman Dyson (1923–2020) once gave a candidate for a statement that might be true "for no good reason":

Numbers that are exact powers of two are 2, 4, 8, 16, 32, 64, 128 and so on. Numbers that are exact powers of five are 5, 25, 125, 625 and so on. Given any number such as 131072 (which happens to be a power of two), the reverse of it is 270131, with the same digits taken in the opposite order. Now my statement is: it never happens that the reverse of a power of two is a power of five.

Sometimes Things are True for No Good Reason

Another example: abelian cubes in the Tribonacci word.

The Tribonacci word is **tr** $= 010201001020101\cdots$, the fixed point of the morphism sending

$$0 \to 01, \quad 1 \to 02, \quad 2 \to 0.$$

An abelian cube is a word of the form $xx'x''$, where x', x'' are permutations of x, like the English word `deeded`. The order of an abelian cube is $|x|$.

What are the orders of abelian cubes appearing in **tr**?

Answer: there is a Tribonacci automaton of 1169 states (!) recognizing the set of all these orders (expressed in the Tribonacci numeration system). Probably there is no simple description of what these orders are.

Some true statements have no short proof

For example:

"This statement has no proof in Peano arithmetic with less than 10^{100} symbols."

Friedman: constructed a statement about tree embeddings whose shortest proof in Peano arithmetic has length $A(1000)$, where A is the Ackerman function.

An Example from Automata Theory

Suppose we conjecture that all words satisfy some property.

If this property can be represented by an NFA $M = (Q, \Sigma, \delta, q_0, F)$, then this conjecture becomes the *universality problem:* does M recognize Σ^*?

But the universality problem for NFA's is PSPACE-complete, so probably there is no efficient algorithm to check universality.

Even worse, we may not even be able to check a possible counterexample in polynomial time, since there exist NFA's with $O(n)$ states where the shortest word *not* accepted is of length $\geq 2^n$.

An Example from Automata Theory

An example:

$$L_n = \{0, 1, \#\}^* - \{\, [0]\#[1]\#\cdots\#[2^n - 1]\,\},$$

where $[a]$ is the binary representation of a, padded on the left to make it n bits long.

It is not hard to construct an NFA M_n for L_n that has $O(n)$ states, while the shortest (and only) word M_n does not accept is clearly of length $(n+1)2^n - 1$.

Letting a Computer Do the Work

Entringer, Jackson, and Schatz proved that every binary word containing no squares xx with $|x| \geq 2$ is of length ≤ 18. They do so by a case-based analysis that is displayed in a diagram that takes up an entire page of their paper:

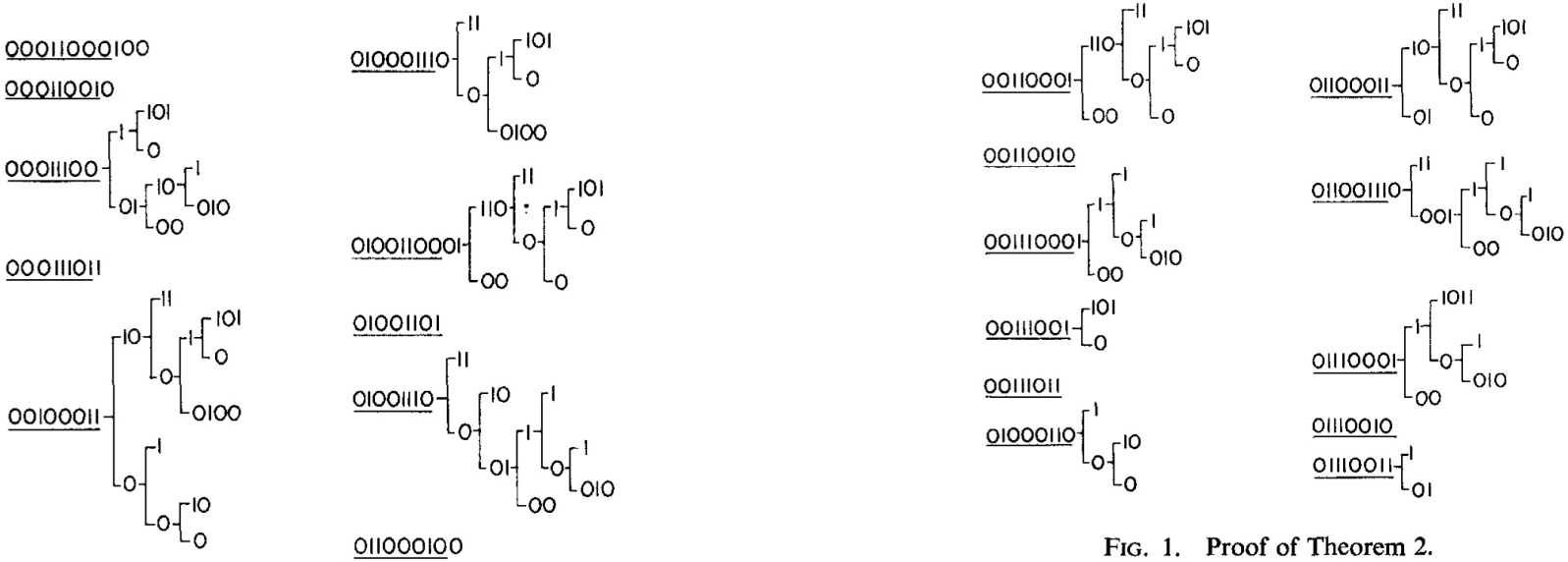


FIG. 1. Proof of Theorem 2.

Letting a Computer Do the Work

But why do this?

In general, to solve avoidability problems, one can use breadth-first search to search the tree of possibilities.

Better to have a general-purpose implementation of breadth-first search, make it publicly available, and report various parameters about the search (tree depth, number of nodes, maximal examples, etc.).

In the case where the tree is very large, this will be the only sensible way anyway.

Letting a Computer Do the Work

Example: in a 2008 paper, we studied self-similarity of words. For x, y equal-length words, we define $A(x, y) = t/|x|$, where t is the number of positions where x and y agree.

A word w is α-similar if

$$\alpha = \sup\{A(x, y) : |x| = |y| \text{ and } xy \text{ is a factor of } w\}.$$

We showed that no infinite word over a 5-letter alphabet can be $< 2/5$-similar, by breadth-first search. Here we had to examine over 200,000 words to prove the result, and this probably has no short proof.

Algorithmic Case Analysis Prevents Errors

Consider a recent theorem by Cilleruelo and Luca: for every integer base $b \geq 5$, every natural number is the sum of three natural numbers whose base-b representations are palindromes.

Their 30-page proof required examining a very large number of cases (one case was labeled IV.5.v.b), and would be rather challenging to verify by hand.

IV.5.v.b) $\boldsymbol{y_{m-1} = 0}$. Note that $x_{m-2} = 1$. Indeed, if $x_{m-2} = 0$, then in order to have $c_{m-1} = 1$, we would need that $c_{m-2} = 1$ and so $x_{m-3} \geq g - 2$, which is false.

δ_{m+1}	δ_m	δ_{m-1}	δ_{m-2}	$*$
x_{m-2}	1	1	x_{m-2}	$*$
$*$	0	0	0	0
$*$	$*$	0	$g-1$	0

$\longrightarrow$

δ_{m+1}	δ_m	δ_{m-1}	δ_{m-2}	$*$
$x_{m-2}-1$	1	1	$x_{m-2}-1$	$*$
$*$	$g-1$	$g-2$	$g-2$	$g-1$
$*$	$*$	1	1	1

IV.5.v.c) $\boldsymbol{y_{m-1} = 1}$. Then $x_{m-2} = 1$. Indeed, this follows as before, namely since $y_{m-2} \equiv \delta_m - z_{m-3} - 1 \pmod{g}$, and $\delta_m \neq 0$, $y_{m-2} = 0$, we get that $z_{m-3} \leq \delta_m - 1$, so $x_{m-2} \neq 0$ (even when $m = 4$). Then:

Algorithmic Case Analysis Prevents Errors

As it turns out, however, the initial proof had some small, easily-repaired flaws that were only discovered when the case analysis was programmed up in Python by Baxter.

Replacing Case Analysis with Automata

Cilleruelo et al. resolved the sum-of-palindromes problem for bases $b \geq 5$.

We decided to tackle the remaining cases $2 \leq b \leq 4$.

Instead of a long case-based argument, our goal was to solve the problem using automata.

Basic approach: use nondeterminism to "guess" a representation of a number as a sum of palindromes, then verify that your guess is correct.

If we create an automaton M that implements this, then we need to show that $L(M)$ is universal: the base-b representation of every number is accepted.

Replacing Case Analysis with Automata

Trouble: for CFL's universality is undecidable, and for NFA's universality is PSPACE-complete.

We succeeded with the following ideas:

- Instead of arbitrary palindromes, we restricted our attention to palindromes of approximately the same length.
- Instead of guessing the palindromes in the sum, we guessed the first halves of the palindromes.
- This makes adding them more complicated, but the trick is to represent N you are trying to write as a sum of palindromes in a "folded" manner, where the input consists of blocks of two symbols. The first coordinates correspond to the most-significant digits and the second coordinates correspond to the least-significant digits, in reverse.

Replacing Case Analysis with Automata

With these ideas, we were able to prove:

Theorem. Every natural number is the sum of 4 numbers whose base-2 representation is a palindrome.

We also proved similar results for bases 3 and 4.

All the case analysis was replaced by basic operations on automata.

Replacing Case Analysis with Automata

Here's another example where one can replace case analysis with automata.

Theorem. (Fici-Zamboni) There are exactly 40 distinct infinite binary words containing, as factors, 10 distinct palindromes.

Their proof required some case analysis.

Instead, let's view this as an automaton problem.

Replacing Case Analysis with Automata

Step 1: Let S be a finite set of palindromes over an alphabet Σ.

Then

$$C_\Sigma(S) := \{x \in \Sigma^* \ : \ \textsc{PalFac}(x) \subseteq S\}$$

is regular.

To see this, note that if ℓ is the length of the longest palindrome in S, then

$$\overline{C_\Sigma(S)} = \bigcup_{t \in P_{\leq \ell+2}(\Sigma) \setminus S} \Sigma^* t \Sigma^*,$$

where $P_{\leq r}(\Sigma)$ is the set of all palindromes of length $\leq r$ over Σ.

Replacing Case Analysis with Automata

Step 2: Let $D_\ell(\Sigma)$ be the set of finite words over Σ containing at most ℓ distinct palindromes as factors.

Then

$$D_\ell(\Sigma) = \bigcup_{\substack{|S| \leq \ell \\ S \subseteq P_{\leq 2\ell-1}(\Sigma)}} C_\Sigma(S)$$

is regular.

Replacing Case Analysis with Automata

Step 3: Setting $\ell = 10$, and computing $D_\ell(\{0,1\})$, we get an automaton of 280 states recognizing the set of all finite binary words having at most 10 distinct palindromic factors.

We then simply list the infinite paths through this automaton.

Since there are no birecurrent states, all the infinite paths are of the form uv^ω for some finite words u, v.

Some of these have only 9 distinct palindromic factors; the remaining ones give us the list of 40.

Heuristics Plus Algorithms Can Create Proofs

Idea: use heuristics to find possible routes to a proof. Then use an algorithm to complete the proof.

Consider the following problem: choose a finite set of unary operations on languages, such as $S = \{$ Kleene closure, complement $\}$.

Start with a language L, and apply the operations of S to L as many times as you like, and in any order.

(This is the *orbit* of L under the set S.) How many different languages can you get?

For the particular S above, the answer is 14; this is a version of the Kuratowski 14-theorem from topology.

Heuristics Plus Algorithms Can Create Proofs

We can then try different sets of operations. In 2012, we proved the following result:

Theorem. For the set of eight operations

$$S = \{ \text{Kleene closure, positive closure, complement, prefix, suffix, factor, subword, and reverse} \}$$

the size of the orbit of every language is at most 5676.

Idea: certain finite sequences of composed operations generate the same language as shorter sequences.

Heuristics Plus Algorithms Can Create Proofs

For example, if k denotes Kleene closure and c denotes complement, then $kckckck$ has the same effect as kck.

By generating an extensive list of identities like $kckckck \equiv kck$, we can do a breadth-first search over the tree of all sequences of operations, demonstrating that there is a finite set of sequences that covers all possibilities.

But which identities are true? Here is where heuristics can help us.

Heuristics Plus Algorithms Can Create Proofs

We can model all languages with the class of regular languages.

To find an identity, we can apply one list of operations to some randomly-generated set of regular languages and compare it to the result of some other list.

If the results agree everywhere, we have a candidate identity we can try to prove.

When implemented, our procedure generated dozens of identities, most of which had trivial proofs.

Once we had these identities, we used breadth-first search to prove that the size of the orbit was finite.

Decision Procedures

Best of all possible worlds: replace case analysis by a claim that can be verified with a decision procedure.

One domain where this has been very successful is the combinatorics of automatic sequences.

A sequence $(s_n)_{n \geq 0}$ over a finite alphabet is *automatic* if, roughly speaking, there is a deterministic finite automaton with output (DFAO) that, on input the representation of the natural number n in some form, ends in a state with output s_n.

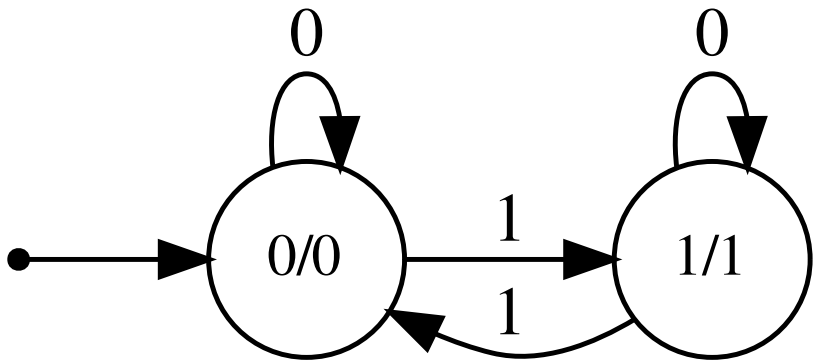
Decision Procedures

A typical example of the kind of representation we are talking about is base-2 representation. For automatic sequences, thanks to Büchi and others there is a decision procedure to answer questions about these sequences that are phrased in first-order logic.

Let's look at a specific example. The *Thue-Morse sequence*

$$\mathbf{t} = (t_n)_{n \geq 0} = 0110100110010110 \cdots$$

is an automatic sequence and is generated by the following very simple automaton. Here the label a/b on a state means that the state is numbered a and the output associated with the state is b.



Decision Procedures

A word x has *period* $p \geq 1$ if $x[i] = x[i+p]$ for all indices i that make sense.

Currie and Saari proved that $\mathbf{t}$ has a factor of least period p for all integers $p \geq 1$. Their proof required 3 lemmas, 6 cases, and 3 pages, e.g.:

Case 2. $r \equiv 3 \pmod 6$. First, let $s = (r+3)/2$. Then either $s \equiv 0$, or $3 \pmod 6$, and $s < r$. Thus there is a factor w of $\mathbf{t}$ of the form $00z101$ having length s and least period s. Let $u = {\sim}\mu(w^R)'$. Then u is a factor of $\mathbf{t}$, it is of length $r+1$, and it is of the form $u = 00y010$, where $y = 110\mu(z^R)$.

Evidently, the word u has period r. Corollary 1 implies that it has no even period strictly shorter than $|u| = r+1 = 2s-2$. Writing $u = u_0u_1u_2\cdots u_r$, we see that $u_0 \neq u_{r-1}$, $u_1 \neq u_{r-1}$, $u_3 \neq u_r$, showing that u does not have period $r-1$, $r-2$, or $r-3$. By Lemma 1, u can have no odd period less than r. Thus the least period of u is r, witnessing the item (iii).

Next, let $s = (r+1)/2$. Then either $s \equiv 2$, or $5 \pmod 6$. There is a factor v of $\mathbf{t}$ of the form $00z101$ having length s and least period s. Let $u = {\sim}\mu(v^R)$. Then u is a factor of $\mathbf{t}$, it is of length r, and it has the form $u = 00y101$, where $y = 110\mu(z^R)0$.

Evidently, the word u has period $|u| = r$. Corollary 1 implies that it has no even period strictly shorter than $r+1 = 2s$. Writing $u = u_0u_1u_2\ldots u_{r-1}$, we see that $u_0 \neq u_{r-1}$, $u_1 \neq u_{r-1}$, $u_0 \neq u_{r-3}$, showing that u does not have period $r-1$, $r-2$, or $r-3$. By Lemma 1, u can have no odd period less than r. Thus the least period of u is r, witnessing the item (ii).

Decision Procedures

The Currie-Saari claim about least periods can be phrased in a certain logical system that is algorithmically decidable, and there is a decision procedure for it.

This procedure has been implemented in the Walnut theorem prover written by Hamoon Mousavi, and so we can enter the commands

```
def tmperi "(p>0) & (p<=n) & Aj (j>=i & j+p<i+n) => T[j]=T[j+p]":
def tmlper "$tmperi(i,n,p) & (Aq (q>=1 & q<p) => ~$tmperi(i,n,q))":
eval currie_conj "Ap (p>=1) => Ei,n (n>=1) & $tmlper(i,n,p)":
```

which returns TRUE in a matter of .062 seconds of CPU time. Here

- `tmperi` asserts that $\mathbf{t}[i..i+n-1]$ has period p, and
- `tmlper` asserts that the least period of $\mathbf{t}[i..i+n-1]$ is p.

Decision Procedures

A factor is said to be *bordered* if it begins and ends with the same word in a nontrivial way, like the English word entanglement. If it is not bordered, we call it *unbordered*.

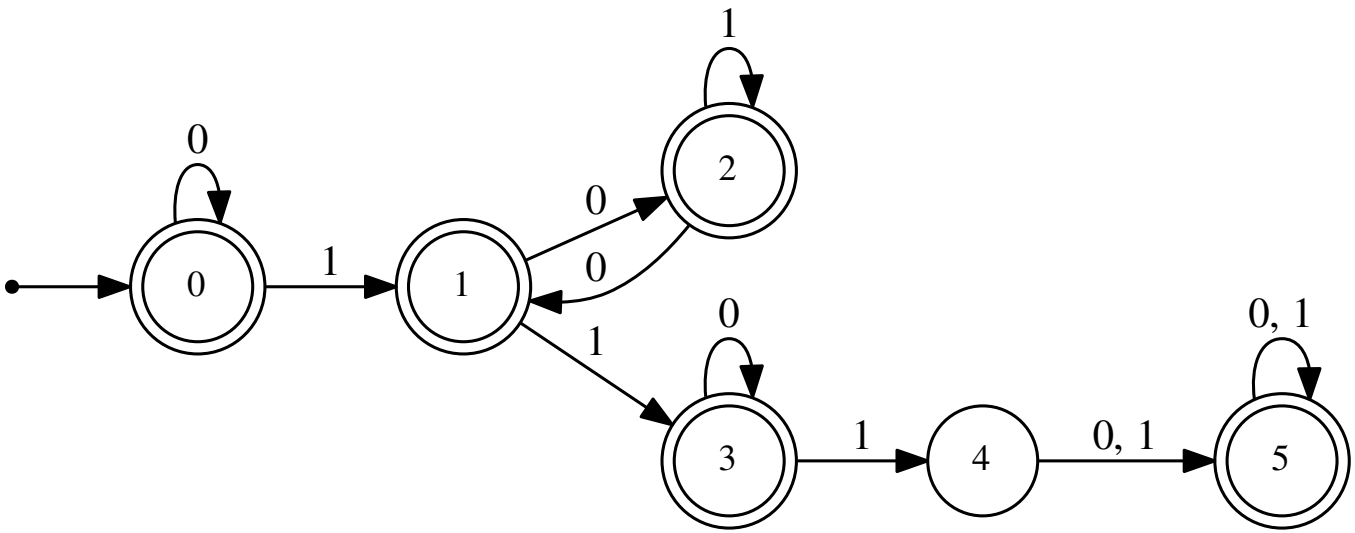
Currie and Saari were also interested in determining all lengths of unbordered factors in **t**.

They proved that **t** has a length-n unbordered factor if $n \not\equiv 1 \pmod 6$, but were unable to find a necessary condition. We can achieve this with Walnut by writing

```
def tmfactoreq "At (t<n) => T[i+t]=T[j+t]":
def tmbord "(m>=1) & (m<n) & At (t<m) => $tmfactoreq(i,(i+n)-m,m)":
def tmunbordlength "Ei Am ~$tmbord(i,m,n)":
```

Decision Procedures

Running this in `Walnut` produces the following automaton, which recognizes the base-2 representation of all n for which $\mathbf{t}$ has a length-n unbordered factor:



Theorem. The Thue-Morse sequence $\mathbf{t}$ has an unbordered factor of length n if and only if $(n)_2 \notin 1(01^*0)^*10^*1$.

Decision Procedures

Let's look at one more problem, this time from additive number theory.

The *upper Wythoff set* $U = \{2, 5, 7, 10, 13, \ldots\}$ is defined to be $\{\lfloor \alpha^2 n \rfloor : n \geq 1\}$, where $\alpha = (1 + \sqrt{5})/2$ is the golden ratio.

Recently Kawsumarng et al. studied the sumset

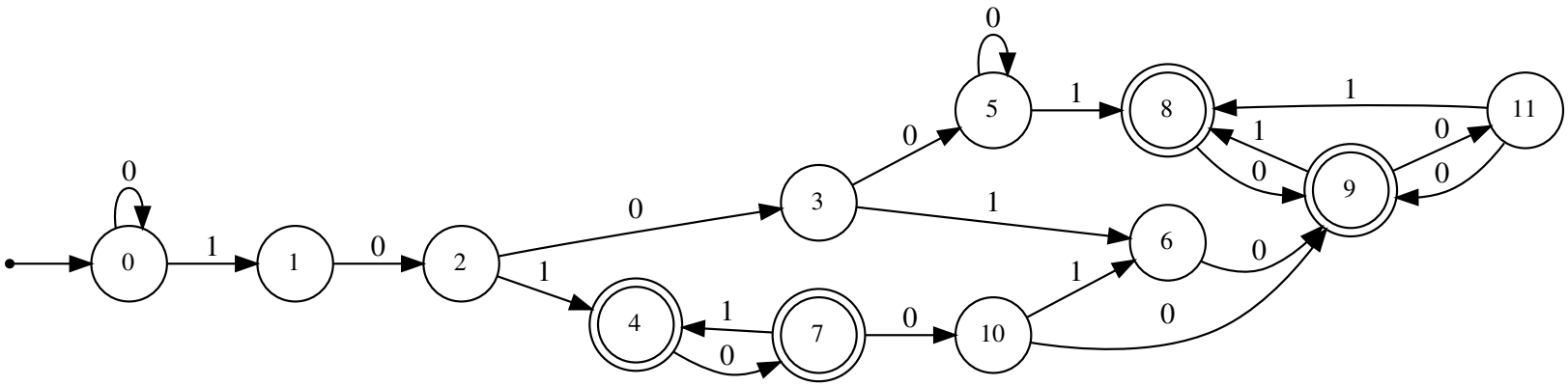
$$U + U = \{x + y : x, y \in U\}.$$

Using a case-based argument, they constructed a rather complicated description of this set, noting that it "has some kinds of fractal and palindromic patterns".

However, it turns out that the assertion $n \in U + U$ is first-order expressible in a decidable logical theory; this allows us to give a complete description of $U + U$ as the set of natural numbers whose Fibonacci representation.

Decision Procedures

$U + U = \{x + y : x, y \in U\}$ is recognized by the following automaton:



Here no explicit breakdown into cases was necessary; instead, the decision procedure "automatically" constructs the automaton from a description of U.

The fact that this automaton has so many states and a complicated structure partially explains why the set $U + U$ is difficult to describe explicitly.

Heuristics Plus Decision Procedures Provide Proofs

We can combine the ideas of depth-first or breadth-first search over a space with a decision procedure to (a) figure out a good candidate for a solution and then (b) prove it is correct.

As an example, let's return to automatic sequences.

In 1965, Richard Dean studied the *Dean words:* squarefree words over $\{x, y, x^{-1}, y^{-1}\}$ that are not reducible (that is, there are no occurrences of $xx^{-1}, x^{-1}x, yy^{-1}, y^{-1}y$).

Heuristics Plus Decision Procedures Provide Proofs

Let us use the coding $0 \leftrightarrow x$, $1 \leftrightarrow y$, $2 \leftrightarrow x^{-1}$, $3 \leftrightarrow y^{-1}$.

We can use "automatic breadth-first search" to find a candidate for an infinite Dean word.

In automatic breadth-first search, you guess that the infinite word you want to construct is k-automatic for some integer $k \geq 2$, and generated by a DFAO of $\leq \ell$ states.

You then use BFS to explore the tree of all w obeying the particular constraints, such that the smallest k-DFAO generating w has $\leq \ell$ states.

Heuristics Plus Decision Procedures Provide Proofs

If you are lucky, BFS will converge to the prefixes of a single k-automatic infinite word (or small number of such words).

When implemented for Dean words, breadth-first search quickly converges on the sequence

$$0121032101230321\cdots,$$

which (using the Myhill-Nerode theorem) we can guess as the fixed point of the morphism

$$0 \to 01,\ 1 \to 21,\ 2 \to 03,\ 3 \to 23.$$

Heuristics Plus Decision Procedures Provide Proofs

Then we carry out the following Walnut commands:

```
morphism d "0->01 1->21 2->03 3->23":
promote DE d:
eval dean1 "Ei,n (n>=1) & At (t<n) => DE[i+t]=DE[i+n+t]":
# check if there's a square
eval dean02 "Ei DE[i]=@0 & DE[i+1]=@2":
eval dean20 "Ei DE[i]=@2 & DE[i+1]=@0":
eval dean13 "Ei DE[i]=@1 & DE[i+1]=@3":
eval dean31 "Ei DE[i]=@3 & DE[i+1]=@1":
# check for existence of factors 02, 20, 13, 31
```

All of these return FALSE, so this word is a Dean word. We have thus proved the existence of Dean words with essentially no human intervention.

Objections and Answers

You've replaced a case-based proof with an algorithm, but how do you know the algorithm is correct?

Answer: Sometimes an implementation will be much simpler than the record of the cases it examines, so it will actually be *easier* to verify the program than the case-based argument.

In other cases, the algorithm can produce a *certificate* that another, simpler program can easily verify.

Finally, in addition to formal correctness, there is also empirical correctness. With a program in hand, we can test it on a wide variety of different inputs to look for oversights and omissions.

Objections

Running a program provides no insight as to *why* a result is true.

Answer: Sometimes there just *won't be* a simple reason why a result is true.

In situations like this, it's better just to accept the result and move on.

Objections

Some of the decision procedures you've talked about have astronomical worst-case running times.

Answer: Don't pay much attention to the worst-case running time of decision procedures! Even if the running time is nonelementary, it will often run in a reasonable length of time for the instances we are interested in.

A Final Word

This talk has been an argument against the following recommendation:

"A computation is a temptation that should be resisted as long as possible." – John P. Boyd, *Amer. Math. Monthly* **123** (2016) 241.

Not only should you give into the temptation, you should actively seek it in some cases!

For Further Reading

- E. Charlier, M. Domaratzki, T. Harju, J. Shallit: Composition and orbits of language operations: finiteness and upper bounds. *Int. J. Comput. Math.* **90** (2013), 1171–1196. [link]
- P. Ochem, N. Rampersad, and J. Shallit, Avoiding approximate squares, *Int. J. Found. Comput. Sci.* **19** (2008), 633–648. [link]
- A. Rajasekaran, J. Shallit, and T. Smith, Additive number theory via automata theory, *Theor. Comput. Sys.* **64** (2020) 542–567. [link]
- J. Shallit, Sumsets of Wythoff sequences, Fibonacci representation, and beyond, *Per. Math. Hung.* (2021). [link]