Divisibility and State−Complexity

Klaus Sutner

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sutner@cs.cmu.edu

Requirements

Most of the computations in this notebook depend on `Automata`, a large package that implements finite state machines. The package is freely available at www.cs.cmu.edu/~sutner and contains installation instructions.

```
Needs["Automata`automata`"]
Get["Automata`experimental`"]
Get["Automata`divisibility`"]
MakeAbbrevs[]
Off[General::spell]
Off[General::spell1]
```

■ Automaticity and State−Complexity

□ Automaticity

A sequence $\boldsymbol{a} = (a_n)$ is B−automatic if it can be recognized by a finite state machine in the sense that, on input n, the machine outputs a_n. Here n is usually written in standard base B notation, though other numeration systems are also considered. A typical example is the Morse−Thue sequence (t_n) which is usually defined in terms of iterated morphisms. However, there is an alternative characterization that shows that t_n is the digit−sum of the binary expansion of n modulo 2. Thus, the Morse−Thue sequence is 2−automatic. The study of automaticity has lately attracted a lot of attention, see the excellent book by Allouche and Shallit [AS].

Alas, little is known about the state complexity of the finite state machines in question: given an automatic sequence, what is the size $\mu(\boldsymbol{a})$ of the smallest machine that recognizes it? Often there is a canonical machine that witnesses the automaticity of a sequence and provides an upper bound on $\mu(\boldsymbol{a})$, but the inherent complexity of the minimization process often makes it very difficult to pin down the exact state−complexity of the sequence. Using a suitable computational environment one can generate sample data that can lead to plausible conjectures. More computation can then help to prune false assumptions and produce answers, at least in a few selected cases. This is particulary important since the combinatorics of our problem will turn out to be fairly complicated so that in general no simple closed form solutions are available.

□ Divisibility and Horner Automata

In this paper we will study the state complexity of machines recognizing the set $m\mathbb{N}$ of all multiples of a fixed modulus m. It is not hard to see that $m\mathbb{N}$ is indeed B–recognizable for any base $B \geq 1$ (we will focus on the interesting case $B \geq 2$ in what follows). We denote the digit alphabet $\{0, 1, \ldots, B-1\}$ by Σ_B. There is a canonical DFA $\mathcal{A} = \mathcal{A}_{m,B}$ that accepts all strings $w \in \Sigma_B^*$ that denote numbers in base B notation that are divisible by m. The state set of $\mathcal{A}$ is the set $(m) = \{0, 1, \ldots, m-1\}$ of modular numbers and the right action is given by

$$p \cdot a = B\,p + a \bmod m.$$

If we fix the initial state to $q_0 = 0$ we have $q_0 \cdot w = \nu(w) \bmod m$ for any word w. Here $\nu(w)$ denotes the value of w: the value of word $w = w_{k-1}\, w_{k-2} \ldots w_1\, w_0$ is

$$\nu(w) = \Sigma_{i<k}\, w_i\, B^i.$$

Thus, the action corresponds to the standard Horner scheme of evaluating polynomials and we will refer to these machines as Horner automata. Hera are some sample Horner automata.

```
m = 5; B = 2;
M = DivisibilityDFA[ m, B, Full → True]
```

```
DFA[5, -2, {{1, 3, 5, 2, 4}, {2, 4, 1, 3, 5}}, 1, {1}]
```

We generate all words in the acceptance language of length 6 and compute their numerical values.

```
LanguageFA[M, 6]
```

```
{000000, 000101, 001010, 001111, 010100, 011001,
 011110, 100011, 101000, 101101, 110010, 110111, 111100}
```

```
WordToNumber[B][%]
```

```
{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60}
```

As required, the moduli are all 0. Other useful information about the associated language can be extracted from the automaton. For example, we can obtain generating function for the growth rate.

```
Clear[x]
gs = GrowthSeriesDFA[M, x]
```

$$\frac{1 - 2x + x^2 - x^3}{1 - 3x + 3x^2 - 3x^3 + 2x^4}$$

```
Series[ gs, {x, 0, 10} ]
```

$$1 + x + x^2 + 2x^3 + 4x^4 + 7x^5 + 13x^6 + 26x^7 + 52x^8 + 103x^9 + 205x^{10} + 0[x]^{11}$$

A different base and modulus.

```
m = 20; B = 15;
M = DivisibilityDFA[ m, B, Full → True]
```

```
DFA[20, -15,
 {{1, 16, 11, 6, 1, 16, 11, 6, 1, 16, 11, 6, 1, 16, 11, 6, 1, 16,
   11, 6}, {2, 17, 12, 7, 2, 17, 12, 7, 2, 17, 12, 7, 2, 17,
   12, 7, 2, 17, 12, 7}, <<12>>, {15, 10, 5, 20, 15, 10, 5,
   20, 15, 10, 5, 20, 15, 10, 5, 20, 15, 10, 5, 20}}, 1, {1}]
```

```
WordToNumber[B][LanguageFA[M, 2]]
```

```
{0, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220}
```

Horner automata provide an upper bound for the state complexity of divisibility: $\mu(m\mathbb{N}) \le m$, regardless of the base B. Alas, the bound is usually far from tight.

□ Minimal Divisibility Recognizers

Pinning Down Behavioral Equivalence

Though the canonical Horner DFAs fail to be minimal in general they are still helpful in determining the structure of the minimal recognizers. Recall that any DFA for a regular language covers the minimal DFA, so we need to determine the fibers of the covering map. Abstractly, we can describe the corresponding partition as follows. Let

$$\mathbb{N}_{k,B} = \{ c \in \mathbb{N} \mid 0 \le c < B^k \}.$$

It is well known that for radix B representation automaticity of divisibility follows from the fact that the following equivalence relation has finite index when $X = m\mathbb{N}$:

$$x \equiv_{m,B} y \iff \forall\, k \ge 0,\ c \in \mathbb{N}_{k,B}\ (\ B^k x + c \in X \longleftrightarrow B^k y + c \in X\ ),$$

see [BHMV]. The index of this equivalence relation is none other than the state complexity we are interested in.
However, this characterization does not readily produce a reasonable description of the state complexity.

By applying a standard minimization algorithm to our Horner automata we can generate some data that will guide the search for a description of the state complexity. In the following table m is the row−index and B is the column−index.

```
tab =
  Table[ Size[MinimizeFA[DivisibilityDFA[m, B, Full → True]]],
   {m, 12}, {B, 16}];
TableForm[ tab, TableHeadings → Automatic, TableSpacing → {1, 1}]
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	2	3	3	2	3	3	2	3	3	2	3	3	2	3
4	4	3	4	2	4	3	4	2	4	3	4	2	4	3	4	2
5	5	5	5	5	2	5	5	5	5	2	5	5	5	5	2	5
6	6	4	3	4	6	2	6	4	3	4	6	2	6	4	3	4
7	7	7	7	7	7	7	2	7	7	7	7	7	7	2	7	7
8	8	4	8	3	8	5	8	2	8	5	8	3	8	5	8	2
9	9	9	3	9	9	4	9	9	2	9	9	4	9	9	4	9
10	10	6	10	6	3	6	10	6	10	2	10	6	10	6	3	6
11	11	11	11	11	11	11	11	11	11	11	2	11	11	11	11	11
12	12	5	5	4	12	3	12	4	5	7	12	2	12	7	5	4

We will write $\mu(m, B)$ for the size of the minimal DFA that recognizes numbers divisible by m in base B notation. If we disregard $B = 1$ the table suggests that $\mu(m, m) = 2$. Moreover, for m and B coprime we have $\mu(m, B) = m$ so that the Horner automaton is already minimal. Both observations are easy to prove.

For the remaining cases, note that for any word $w \in \Sigma_B^k$ we have in $\mathcal{A}$:

$$p \cdot w = B^k\ p + \nu(w) \pmod m .$$

The behavior of state p in $\mathcal{A}$ is therefore

$$\{\, w \in \Sigma_B^* \mid B^{|w|}\ p + \nu(w) = 0 \pmod m \,\} .$$

Define the witness for p to be the length−lex minimal word w in the behavior of p.

Proposition: Two states are behaviorally equivalent if, and only if, they have the same witness.

Suppose p has a witness of length k. Then p is a solution of the linear equation

$$B^k\, x + c = 0 \pmod m \tag{1}$$

where the additive coefficient is bounded as $0 \le c < B^k$. Thus, in order to determine equivalence of states, we have to characterize the solution sets of this equation. Let

$\mathbb{S}_{k,c}$: the set of all solutions to equation (1)

and

$\mathbb{S}'_{k,c}$: the set of all solutions to equation (1) which are not in $\bigcup_{l<k} \mathbb{S}_{l,c}$.

We will refer to these solution sets as cumulative versus strict. Note that $\mathbb{S}_{k,c} \subseteq \mathbb{S}_{k+1,Bc}$. Since the length of the behavioral witness matters, rather than just the associated numerical value $c = \nu(w)$, we have to consider strict rather than just cumulative solution sets. Our goal is to determine the number of solution sets and the levels at which they appear. As it turns out, the combinatorics are somewhat complicated so that the easy availability of sample data is crucial.

Examples

Here are some examples of solution sets. We use a little wrapper function `SolveModEq` that solves modular equations in a useful format. The strict version removes solutions from lower levels. For coprime modulus and base all solution sets have size one; all parameters c are feasible in the sense that equation (1) has a solution at some level k.

```
m = 10; B = 3;
ColumnForm[ Table[ SolveModEq[ m, B, k ], {k, 0, 2} ] ]
```

```
{{0}}
{{0}, {3}, {6}}
{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}}
```

Here modulus and base are not coprime. The size of the solution sets reaches a plateau at level $\sigma = 2$.

```
m = 15; B = 3;
ColumnForm[ Table[ Sort@SolveModEq[ m, B, k ], {k, 0, 3} ] ]
```

```
{{0}}
{{0, 5, 10}}
{{0, 5, 10}, {1, 6, 11}, {3, 8, 13}}
{{0, 5, 10}, {1, 6, 11}, {2, 7, 12}, {3, 8, 13}, {4, 9, 14}}
```

Note that the solution sets are of the form $a_0 + i\,d$ for $i = 0, 1, \ldots$ We refer to d as the stride of the solution set. In the example, $d = 5$.
Here the the size of the solution sets increases till all of $\mathbb{Z}_m$ appears as a solution somewhere.

```
m = 16; B = 6;
ColumnForm[ Table[ Sort@SolveModEq[ m, B, k ], {k, 0, 4} ]]
```

```
{{0}}
{{0, 8}, {2, 10}, {5, 13}}
{{0, 4, 8, 12}, {1, 5, 9, 13}, {2, 6, 10, 14}, {3, 7, 11, 15}}
{{0, 2, 4, 6, 8, 10, 12, 14}, {1, 3, 5, 7, 9, 11, 13, 15}}
{{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}}
```

Now the same examples with strict solutions. The coprime case is trivial, so we skip it.

```
m = 15; B = 3;
ColumnForm[
 Table[Sort@SolveModEq[m, B, k, Mode → Strict, Full → False],
  {k, 0, 3}]]
```

```
{{0}}
{{5, 10}}
{{1, 6, 11}, {3, 8, 13}}
{{2, 7, 12}, {4, 9, 14}}
```

Note how the size of some of the strict solutions sets deviates from $\gcd(B^k, m)$.

```
m = 16; B = 6;
ColumnForm[
 Table[Sort@SolveModEq[m, B, k, Mode → Strict, Full → False],
  {k, 0, 2}]]
```

```
{{0}}
{{8}, {2, 10}, {5, 13}}
{{1, 9}, {4, 12}, {6, 14}, {3, 7, 11, 15}}
```

Counting Cumulative Solutions

Let us tacitly assume that parameter c is feasible, i.e., that $\gcd(B^k, m) \mid c$ in which case the cardinality of $\mathbb{S}_{k,c}$ is $\gcd(B^k, m)$; the empty solution set will be dealt with separately. There are two natural parameters that are important for the description of all solution sets. First, the depth κ of m and B is the least level k for which $\bigcup_c \mathbb{S}_{k,c} = \mathbb{Z}_m$. Second, the saturation value σ is the least k such that $\gcd(B^k, m) = \gcd(B^{k+1}, m)$. Note that

$$\kappa = \lceil \log_B m \rceil.$$

Letting $\gamma(a, b) = a / \gcd(a, b)$, at level k there are $\gamma(B^k, m)$ solution sets of size $\gcd(B^k, m)$ each. Within each solution set the elements satisfy $x = y \pmod{\gamma(m, B^k)}$.

Lemma: Let $l = \min(\kappa, \sigma - 1)$ and set $N = \sum_{k=0}^{l} \gamma(B^k, m) + \sum_{k=l+1}^{\sigma} \gamma(m, B^k)$. If m and B are coprime then the total number of distinct solutions sets is N, otherwise it is $N + 1$.

We skip the proof.

Examples

We can get a short overview of the structure of the solution hierarchy with the command `ProfilemB`. The command prints out the key parameters, the stride of the solution sets, the number of solution sets and the size of the solution sets. Here is a case where $\sigma < \kappa$.

```
m = 15; B = 3;
ProfilemB[m, B]
```

```
m: 15 B: 3  σ: 1  κ: 2
```

	stride	# coeffs	# sols
0	15	1	1
1	5	1	3
2	5	3	3
3	5	9	3

A sanity check.

```
ColumnForm[ Table[ SolveModEq[m, B, k], {k, 0, 3} ]]
```

```
{{0}}
{{0, 5, 10}}
{{0, 5, 10}, {3, 8, 13}, {1, 6, 11}}
{{0, 5, 10}, {1, 6, 11}, {2, 7, 12}, {3, 8, 13}, {4, 9, 14}}
```

And here is $\sigma > \kappa$.

```
m = 16; B = 6;
ProfilemB[ m, B ]
```

```
m: 16 B: 6  σ: 4  κ: 1
```

	stride	# coeffs	# sols
0	16	1	1
1	8	3	2
2	4	9	4

Counting Strict Solutions

Let σ be the least k such that $\gcd(B^k, m) = \gcd(B^{k+1}, m)$ and let ρ, the rank of m and B, be the maximum k such that $\mathbb{S}_{k,c} \neq \emptyset$ for some c. It is easy to see that if m is a power of B we have $\rho = \kappa$, and $\rho = \kappa + 1$ otherwise; hence it suffices to consider levels $k \leq \rho$. Up to level σ the solution sets grow exponentially in size, so $\mathbb{S}_{k,c} \neq \emptyset$ for all feasible coefficients and there are $\gamma(B^k, m)$ solution sets at level k.

However, for $k > \sigma$ we have to contend with potentially empty solution sets. Recall that $\kappa = \lfloor \log_B m \rfloor$. Whenever $k > \kappa$ then the number of feasible coefficients c at level k is $\gamma(m, B^\kappa)$ rather than $\gamma(B^\kappa, m)$.

Lemma: Let $l = \min(\kappa, \sigma - 1)$ and set
$N = \sum_{k=0}^{l} \gamma(B^k, m) + \min(\gamma(m, B^\kappa) - \gamma(B^\kappa, m), \gamma(m, B^{\kappa+1}))$.

If $\kappa \geq \sigma$ then the number of disjoint solutions sets is N, otherwise it is $N + \gamma(B^\kappa, m)$.

Corollary: The size of the minimal DFA recognizing numbers in base B that are divisible by m is given by
$\mu(m, B) = N$ or $\mu(m, B) = N + \gamma(B^\kappa, m)$ depending on whether $\kappa \geq \sigma$.

Corollary: The length of the longest witness for any state in $\mathcal{A}$ is ρ.

B. Alexeev has found a way to avoid following the hierarchy of solution sets all the way to the end, at least in some cases. Let α be the least k such that
$\gamma(m, B^k) - \gamma(m, B^{k+1}) < \gamma(B^k, m)$. Note that $\alpha \leq \sigma$ and it may well happen that $\alpha < \sigma$.

Lemma: The number of disjoint solutions sets is $\sum_{k=0}^{\alpha-1} \gamma(B^k, m) + \gamma(m, B^\alpha)$.

For a proof see [A].

■ Reverse Base B

□ Brute Force

For reverse base B notation the construction of the minimal DFA $\mathcal{B} = \mathcal{B}_{m,B}$ that accepts all strings $w \in \Sigma_B^*$ that denote numbers divisible by m is somewhat more complicated. One possible choice of a canonical, though not necessarily minimal, DFA is to use as state set the Cartesian product $(m) \times P$ where P is the the multiplicative submonoid of $\mathbb{Z}_m$ generated by B. The right action is given by

$$(p, q) \cdot a = (q\,p + a,\ B\,q) \mod m,$$

the initial state is (0, 1) and the final states are of the form (0, _). Thus, the first component maintains the numerical value of the input string, modulo m, and the second provides the appropriate multiplier for the next digit.

```
m = 10; B = 5;
M = DivisibilityDFA[m, B, Full → True, Direction → Backward]
```

```
DFA[20, -5,
 {{2, 2, 4, 4, 6, 6, 8, 8, 10, 10, 12, 12, 14, 14, 16, 16, 18,
   18, 20, 20}, <<3>>, {10, 2, 12, 4, 14, 6, 16, 8, 18,
   10, 20, 12, 2, 14, 4, 16, 6, 18, 8, 20}}, {1}, {1, 2}]
```

```
LanguageFA[M, 4]  // WordReverse // WordToNumber[B]
```

```
{0, 250, 500, 150, 400, 50, 300, 550, 200, 450, 100, 350, 600,
 130, 380, 30, 280, 530, 180, 430, 80, 330, 580, 230, 480, 10,
 260, 510, 160, 410, 60, 310, 560, 210, 460, 110, 360, 610,
 140, 390, 40, 290, 540, 190, 440, 90, 340, 590, 240, 490, 20,
 270, 520, 170, 420, 70, 320, 570, 220, 470, 120, 370, 620}
```

Note that the size of this automaton is a multiple of m and may be close to m^2. When m is prime and B is a generator of the multiplicative subgroup $\mathbb{Z}_m^*$ the machine will have $m(m-1)$ states.

```
M = DivisibilityDFA[11, 2, Full → True, Direction → Backward];
M  // Size
```

```
110
```

However, the minimal DFA here is much smaller.

```
M // MinimizeFA // Size
```

```
11
```

Some more computation suggests that indedd $\mu^R(m, B)$ is bounded by $m + 1$.

□ The Canonical Nondeterministic Automaton

We write $\mu^R(m, B)$ for the size of the minimal DFA that recognizes numbers divisible by m in reverse base B notation. Since the deterministic machine from the last section appears to be overly large, it is tempting to consider a nondeterministic one in an effort to determine $\mu^R(m, B)$: the reversal of the canonical DFA $\mathcal{A} = \mathcal{A}_{m,B}$ for standard base B. This machine has size m and the transitions again are determined by linear equations modulo m. Recall that in $\mathcal{A}$ the transitions given by $p \cdot a = B\,p + a \pmod m$. Hence we have in $\mathcal{A}^R$:

$$q \in \delta(p, w^R) \iff q \text{ solves } B^{|w|}\,x + \nu(w) - p = 0 \pmod m \qquad (2)$$

Thus, the reachable states in the full power automaton of $\mathcal{A}^R$ are going to be the solution sets of $B^k\, x + c \pmod m$ where $c < B^k$. Note that this time we are dealing with cumulative solutions, not the strict hierarchy from the first section.

```
m = 15; B = 3;
M = ReverseFA@DivisibilityDFA[m, B, Full → True];
MM = ToDFA[ M, Normalize → 2 ]
```

```
DFA[7, -3, {{2, 2, 3, 5, 7, 4, 6},
  {3, 4, 3, 6, 2, 5, 7}, {3, 5, 3, 7, 4, 6, 2}}, 1, {1, 2}]
```

The command `DivisibilityDFA` with option `Full->True` preserves the structure of the state set:

```
States[MM]
```

```
{{0}, {0, 5, 10}, {}, {3, 8, 13}, {1, 6, 11}, {4, 9, 14}, {2, 7, 12}}
```

which state set is none other than the solution sets for equation (1). Since function `SolveModEq[m,B,k]` disregards the empty set as a possible solution set we only get 6 states out of 7 in the next computation.

```
Union@FlattenOne@Table[ SolveModEq[m, B, k], {k, 0, 3}]
```

```
{{0}, {0, 5, 10}, {1, 6, 11}, {2, 7, 12}, {3, 8, 13}, {4, 9, 14}}
```

When m and B are coprime there is no sink since the equation has a solution for all choices of the coefficients.

```
DivisibilityDFA[11, 6, Full → True] // ReverseFA // ToDFA //
 TrapStatesFA
```

```
{}
```

Otherwise there are one or two trap states, in the latter case only one of the two is final.

```
DivisibilityDFA[6, 4, Full → True] // ReverseFA // ToDFA //
 TrapStatesFA
```

```
{3}
```

```
DivisibilityDFA[8, 6, Full → True] // ReverseFA // ToDFA //
 TrapStatesFA
```

```
{3, 8}
```

Since we need not be concerned with the strict hierarchy it is actually a little easier to count the number of all solution sets in this case.

```
m = 45; B = 3;
ProfilemB[m, B]
```

```
m: 45 B: 3  σ: 2  κ: 3
```

	stride	# coeffs	# sols
0	45	1	1
1	15	1	3
2	5	1	9
3	5	3	9
4	5	9	9

```
ColumnForm[ SolveModEq[ m, B, Range[0, 4] ] ]
```

```
{{0}}
{{0, 15, 30}}
{{0, 5, 10, 15, 20, 25, 30, 35, 40}}
{{0, 5, 10, 15, 20, 25, 30, 35, 40}}
{{0, 5, 10, 15, 20, 25, 30, 35, 40}}
```

As before let σ be the least k such that $\gcd(B^k, m) = \gcd(B^{k+1}, m)$ and let $\kappa = \lfloor \log_B m \rfloor$. Also let ρ be the maximum k such that some new solution appears at level k. Thus if m is a power of B we have $\rho = \kappa$, and $\rho = \kappa + 1$ otherwise.

Lemma: Let $l = \min(\kappa, \sigma - 1)$ and set $N = \sum_{k=0}^{l} \gamma(B^k, m) + \sum_{k=l+1}^{\sigma} \gamma(m, B^k)$.

If m and B are coprime then the number of solutions sets is N, otherwise it is $N + 1$.

Corollary: The size of the power automaton obtained from $\mathcal{A}^R_{m,B}$ is N or $N + 1$ depending on whether m and B are coprime.

Corollary: The length of the longest witness for any state in $\mathrm{pow}(\mathcal{A}^R_{m,B})$ is ρ.

□ Minimal Recognizers

We claim that the power automaton obtained from $\mathcal{A}^R$ is always reduced and thus already minimal. To see this, call a state p in a machine M rich if its behavior contains at least one word not in the behavior of $Q - \{p\}$. Clearly, any state $P \subseteq Q$ in the power automaton of M that contains a rich state cannot be equivalent to any other state. Hence it suffices to prove the following.

Lemma: All states in $\mathcal{A}^R$ are rich.

Proof:
Let p be any state in $\mathcal{A}^R$ and choose a word w such that $\nu(w) = p \pmod m$, whence w^R is in the behavior of p in $\mathcal{A}^R$. Suppose w^R lies also in the behavior of state q. Then 0 solves $B^{|w|} x + \nu(w) - q = 0 \pmod m$ and $p = q$, as required.

Corollary: The power automaton obtained from $\mathcal{A}^R_{m,B}$ is minimal.

■ Fibonacci Base

□ Fibonacci Base

Any strictly increasing sequence (U_n) of positive integers where $U_0 = 1$ gives rise to a numeration system. In order to keep the number of distinct digits finite one usually imposes the condition that $\sup U_{n+1} / U_n$ be bounded. The digits in this case can be chosen to be $\Sigma_D = \{0, 1, \ldots, D-1\}$ where $D-1$ is the largest integer less than the supremum. In general, numbers will admit multiple representations in such a numeration system and one can define the normalized representation to be the one obtained by the natural greedy algorithm.

A classical example is given by the Fibonacci sequence (starting at the third term):

$(U_n) = 1, 2, 3, 5, 8, 13, \ldots$ In this case $\lim U_{n+1} / U_n$ is the Golden Ratio, $\phi = \frac{1+\sqrt{5}}{2}$. Hence there are only digits $\{0, 1\}$ and one can compute the normalized representation as follows. We need a little auxiliary function that returns the largest Fibonacci number not greater than a given number.

```
n = RandomInteger[10^4];
LargestFibonacci[n]
```

```
17
```

```
Fibonacci[%] ≤ n < Fibonacci[% + 1]
```

```
True
```

Then a standard greedy algorithm will produce the Fibonacci representation of a number.

```
FibonacciDigitsRaw[1001]
FibonacciDigits[1001]
```

```
{16, 7, 2}
```

```
{1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}
```

A famous open problem related to the Fibonacci sequence is to determine the period length of the sequence modulo m. These numbers are sometimes referred to as Pisano numbers, see Sloane's database of sequences at A001175.
We write per(m) for the length of the sequence modulo m. It is easy to see that the function is multiplicative. It seems that for primes p we have $\text{per}(p^e) = p^{e-1}\,\text{per}(p)$ but no proof is known. Moreover, the behavior of per on primes is not well–understood either. It is known that $\text{per}(p) = p - 1$ exactly when there is a primitive root α modulo p such that $\alpha^2 = \alpha + 1 \pmod p$. Also, $\text{per}(m) = m$ if and only if $m = 24 \cdot 5^e$.

```
Pisano[{11, 19, 31, 41}]
```

```
{10, 18, 30, 40}
```

□ Brute Force

The obvious brute−force construction of a finite state machine for numbers in Fibonacci base (not necessarily normalized) that are divisible by m is to use as state set the Cartesian product $Q = \{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, n-1\}$ where $n = \text{per}(m)$ is the Pisano number of m. The right action on Q is given by

$$(p, q) \cdot 0 = (p, q - 1 \bmod n) \quad \text{and}$$
$$(p, q) \cdot 1 = (r + F_{q+1} \bmod m, q - 1 \bmod n)$$

where F_n denotes the nth Fibonacci number. It is straightforward to construct this automaton, assuming for the time being that (0, 0) is the initial and final state. The method employed below is based on the construction of a cyclic semi−module which is then interpreted as a DFA. `Automata` contains a command GenerateDFA that implements the necessary machinery.

```
Clear[dot]
m = 3;
n = Pisano[m];
P0 =
  Mod[RotateRight[Reverse@Fibonacci@Range[0, n - 1], 2], m];
dot[{p_, q_}, "0"] := { p, Mod[q + 1, n] };
dot[{p_, q_}, "1"] :=
  { Mod[ p + P0[[q + 1]], m], Mod[q + 1, n] };
q0 = {0, 0};
M = GenerateDFA[ q0, dot, -2, # === q0 &, Normalize → 1 ]
```

```
DFA[24, -2,
 {{2, 4, 5, 6, 7, 9, 11, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 1,
   22, 21, 3, 23, 24, 8}, {3, 4, 5, 7, 8, 10, 9, 11, 13, 14, 12,
   15, 16, 17, 19, 20, 18, 21, 1, 22, 23, 2, 24, 6}}, 1, {1}]
```

At present, the automaton only accepts words of length a multiple of $8 = \text{per}(3)$.

```
LanguageFA[M, -40, SizeOnly → True]
```

```
{1, 0, 0, 0, 0, 0, 0, 0, 88, 0, 0, 0, 0, 0, 0, 0,
 21856, 0, 0, 0, 0, 0, 0, 0, 5592448, 0, 0, 0, 0, 0,
 0, 0, 1431655936, 0, 0, 0, 0, 0, 0, 0, 366503876608}
```

In order to get words of arbitrary length we need to add more initial states.

```
PositionList[ States[M], Cases[ States[M], {0, _}] ];
MM = SetInitialFA[ M, % ]
```

```
FA[24, -2, {{1, 1, 2}, {2, 1, 4}, {3, 1, 5}, {4, 1, 6},
  {5, 1, 7}, {6, 1, 9}, {7, 1, 11}, {8, 1, 10}, {9, 1, 12},
  <<30>>, {16, 2, 20}, {17, 2, 18}, {18, 2, 21}, {19, 2, 1},
  {20, 2, 22}, {21, 2, 23}, {22, 2, 2}, {23, 2, 24}, {24, 2, 6}},
 {1, 2, 4, 6, 9, 12, 15, 18}, {1}]
```

A small sanity check: the numerical values are all multiples of 3 −− and all such values seem to appear.

```
FromFibonacciWord[LanguageFA[ MM, -6 ]]
```

```
{{0}, {0}, {0, 3}, {0, 3, 3, 6}, {0, 3, 3, 6, 6, 9},
 {0, 3, 3, 6, 6, 9, 9, 12, 15, 18}, {0, 3, 3, 6, 6, 9, 9, 12,
  15, 18, 15, 18, 18, 21, 21, 24, 21, 24, 24, 27, 27, 30}}
```

A Canonical Automaton

The automaton MM from the last section is nondeterministic because of its multiple initial states though all transitions are deterministic. The question arises what the size of the corresponding power automaton and minimal automaton might be.

```
MM // ToDFA
```

```
DFA[9, -2, {{1, 3, 4, 6, 2, 8, 5, 9, 7}, {2, 4, 5, 7, 8, 1, 3, 6, 9}},
 1, {1, 4, 7}]
```

```
MM // MinimizeFA // Size
```

```
9
```

The power automaton is already minimal and much smaller than one might expect. To see why, note that MM is a permutation automaton. Hence the size of all the states in the power automaton is n, the Pisano number of m and the number of initial states. In fact, all these state contain exactly one element (p, r) for each $0 \le p < n$.

```
ToDFA[ MM, Normalize → 2] // States
```

```
{{{0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {0, 5}, {0, 6}, {0, 7}},
 {{1, 1}, {0, 2}, {1, 3}, {2, 4}, {2, 5}, {0, 6}, {2, 7}, {1, 0}},
 {{1, 1}, {1, 2}, {0, 3}, {1, 4}, {2, 5}, {2, 6}, {0, 7}, {2, 0}},
 {{0, 0}, {1, 2}, {1, 3}, {0, 4}, {1, 5}, {2, 6}, {2, 7}, {2, 1}},
 {{0, 1}, {1, 2}, {2, 3}, {2, 4}, {0, 5}, {2, 6}, {1, 7}, {1, 0}},
 {{0, 1}, {1, 3}, {1, 4}, {0, 5}, {1, 6}, {2, 7}, {2, 0}, {2, 2}},
 {{0, 0}, {1, 1}, {2, 3}, {0, 4}, {2, 5}, {1, 6}, {1, 7}, {2, 2}},
 {{0, 2}, {2, 3}, {1, 4}, {1, 5}, {0, 6}, {1, 7}, {2, 0}, {2, 1}},
 {{0, 3}, {2, 4}, {1, 5}, {1, 6}, {0, 7}, {1, 0}, {2, 1}, {2, 2}}}
```

But then we might as well use sequences of length n of remainders modulo m as states. The action on these sequences can be chosen to be

$$P \cdot 0 = \mathrm{rot}(P)$$
$$P \cdot 1 = \mathrm{rot}(P) + F \pmod m$$

where $F = (F_0, F_1, \ldots, F_{n-1}) \bmod m$ is a period of the Fibonacci sequence modulo m and rot indicates a cyclic shift to the left. It is straightforward to implement this automaton, using again the command GenerateDFA from Automata.

```
M = FibonacciDivDFA[ 3 ]
```

```
DFA[9, -2, {{1, 3, 4, 6, 2, 8, 5, 9, 7}, {2, 4, 5, 7, 8, 1, 3, 6, 9}},
 1, {1, 4, 7}]
```

While the states of these automata depend on the Pisano numbers, the size of the automata appears simply to be m^2.

```
Table[ {k, Size@FibonacciDivDFA[k]}, {k, 2, 6}] // TableForm
```

```
2    4
3    9
4    16
5    25
6    36
```

It is easy to establish this conjecture. The states are all Fibonacci type sequences modulo m but with different initial conditions. All initial conditions occur since the standard sequence has the form (0, 1, …, 1).
It follows that the minimal automaton can have size at most m^2 and to show that this bound is tight it suffices to prove that the canonical automaton is already reduced. To

this end, write 2 for the input $0^{n-1}\ 1$, so that $P \cdot 2 = P + F \pmod m$. Letting $P = (p_0, p_1, \ldots, p_{n-1})$ we have $P \cdot 0^{i-1}\ 2^j\ 0^{n-1}$ is final iff $p_i = -j \pmod m$. Hence all states have distinct behavior and we are done.

■ References

AS J.−P. Allouche, J. Shallit, Automatic Sequences: Theory, Applications, Generalizations; Cambridge UP 2003.

A B. Alexeev, Minimal DFAs for Testing Divisibility, to appear in JCSS.

BH V. Bruyère, G. Hansel, Recognizable Sets of Numbers in Nonstandard Bases, LNCS 911 1995, 167−197.

BHMV V. Bruyère, G. Hansel, C. Michaux. R. Villmaire, Logic and p−recognizable Sets of Integers, Bull. Belg. Math. Soc., 1994 (1) 191−−238.

S K. Sutner, Automata: a Hybrid System for Computational Automata Theory, Proceedings CIAA 2002,
J−M. Champarnaud, D. Maurel, eds., 2002, 217−222.