

CDM

Presburger Arithmetic

K. SUTNER

CARNEGIE MELLON UNIVERSITY

FALL 2024



Everybody who has worked in formal logic will confirm that it is one of the technically most refractory parts of mathematics. The reason for this is that it deals with rigid, all-or-none concepts, and has very little contact with the continuous concept of the real or of the complex number, that is, with mathematical analysis. Yet analysis is the technically most successful and best-elaborated part of mathematics. Thus formal logic is, by the nature of its approach, cut off from the best cultivated portions of mathematics, and forced onto the most difficult part of the mathematical terrain, into combinatorics.

John von Neumann, 1948

1 Numeration Systems

2 Automatic Structures

3 Deciding Presburger Arithmetic

A **numeration system** is a method to denote all natural numbers by words over a **digit alphabet** Δ .

The digits all directly correspond to particular numbers, typically 0, 1, 2, -1 and so forth. In a **positional** notation system the numerical value of a digit string $d = d_0d_1 \dots d_{k-1}$ is determined by **weights** $b_i \in \mathbb{N}_+$, $i \geq 0$, as follows. The **value map** $\text{val} : \Delta^* \rightarrow \mathbb{N}$

$$\text{val}(d) = \sum_{i < k} d_i b_i$$

must be surjective. Ideally, it is also injective, but that is not a requirement.

Interesting weights are simple, something like $b_i = B^i$ for some $B \geq 2$.

To define a numeration system $\mathcal{N} = \langle \Delta, D, \text{val} \rangle$, we need the following pieces.

- A regular language $D \subseteq \Delta^*$ that describes the digit strings over Δ that are admissible representations of numbers.
- A surjective **value map** $\text{val} : D \rightarrow \mathbb{N}$.

Note that val is **not** required to be injective; in fact, there may be infinitely many representations for the same number.

As a practical matter, whenever the value map is not injective, it is convenient to have a canonical choice of representation, a **representation map** $\text{rep} : \mathbb{N} \rightarrow D$ such that $\text{val}(\text{rep}(n)) = n$ for all $n \in \mathbb{N}$.

One natural choice is to pick the length-lex minimal string:

$$\text{rep}(n) = \min_{ll} (x \in D \mid \text{val}(x) = n)$$

If the value map is injective rep is simply the inverse of val .

Fix some integer $B \geq 2$. Arguably the easiest choice is

$$\begin{aligned}\Delta &= \{0, 1, \dots, B-1\} \\ D &= \Delta^* \\ \text{val}(x) &= \sum_{i < k} x_i B^i\end{aligned}$$

where $x = x_0 x_1 \dots x_{k-1}$. This is called **reverse base B** (or reverse radix B), since the LSD comes first in this system.

There are infinitely many representations for any number: $\text{rval}(\varepsilon) = 0$ and $\text{rval}(x0) = \text{rval}(x)$.

Addition is entirely straightforward in this system, it can be carried out by a suitable finite state transducer.

As stated, reverse base B is rather too permissive. One often restricts admissible representations.

NE No empty word

ε is not allowed as a representation for $0 \in \mathbb{N}$.

NTZ No trailing zeros

$x \in D$ implies $x_{-1} \neq 0$.

So for NE and, say, base $B = 2$ we have

$$D = (0 + 1)^+$$

and if we add NTZ we get

$$D = 0 + (0 + 1)^*1$$

It is customary to start numbers with the MSD, and use a more complicate value map: for $|x| = k$, 0-indexed,

$$\text{val}(x) = \sum_{i < k} x_i B^{k-i-1}$$

This is called **base B** or **radix B** notation, we are essentially evaluating x^{op} in reverse base B .

Note that the value map requires knowledge of the length of the string, a feature that often coexists uneasily with finite state machines. Since regular languages and rational relations are closed under reversal, there is no catastrophic difference, but things just tend to get more complicated.

Weird Digits

Some strange digit sets can be useful on occasion. For example, for base 2, we could alternatively use digits $\{-1, 0, +1\}$, so-called **trits**, or digits $\{0, 1, 2\}$, so-called **hyperbinary** digits. E.g., trits can speed up parallel addition.

Weird Weights

Defining weights as powers of B is natural, but we could also use more general recurrences. E.g., Fibonacci numbers can be useful to produce solutions to certain combinatorial problems.

There are occasions (combinatorics, coding theory, dynamical systems) where a weird numeration system helps. For example, one could use Fibonacci numbers F_k rather than powers of some base B and digits $\{0, 1\}$:

$$\text{val}(x) = \sum_i x_i F_{i+2}$$

For example, the following strings all have value 32: 111111, 1111001, 1100101, 0010101.

Exercise

Show that any natural number can be written in the Fibonacci system without using two consecutive 1s, so-called Zeckendorf form. Construct a transducer that converts to Zeckendorf form.

Fix some numeration system $\mathcal{N} = \langle \Delta, D, \text{val} \rangle$.

Suppose $A \subseteq \mathbb{N}$. What does it mean that some plain FSM \mathcal{A} **represents** A ?

Let $L = \mathcal{L}(\mathcal{A}) \subseteq \Delta^*$. We want

$$\begin{aligned} L &\subseteq D \quad \text{only admissible strings accepted} \\ A &= \{ \text{val}(x) \mid x \in L \} \end{aligned}$$

So we have a notion of a **regular** or **recognizable** set of numbers with respect to numeration system \mathcal{N} .

We could add conditions, say, every vector in A has exactly one representative in L .

If the value map is a bijection, the last condition comes for free. However, if val is not injective one might be tempted to go in the opposite direction:

$$L = \text{val}^{-1}(A) = \bigcup \{ \text{val}^{-1}(n) \mid n \in A \}$$

In this case we will say that \mathcal{A} **strongly recognizes** A .

Again, fix a numeration system $\mathcal{N} = \langle \Delta, D, \text{val} \rangle$.

Suppose we have an arithmetic function $f : \mathbb{N} \rightarrow \mathbb{N}$ (the argument for k -ary functions is entirely similar). What does it mean that some transduction τ **represents** f ?

If $\text{val} : D \rightarrow \mathbb{N}$ is a bijection, the definition is fairly straightforward:

$\tau \subseteq D \times D$ total and single-valued

$$\tau(x, y) \Leftrightarrow f(\text{val}(x)) = \text{val}(y)$$

In general, a reasonable definition appears to be

$$\tau \subseteq D \times D$$

$$f = \{ (\text{val}(x), \text{val}(y)) \mid \tau(x, y), x, y \in D \}$$

So the transducer only accepts $x:y$ when both x and y represent numbers in our system. Also, whenever $f(n) = m$, there are strings $x, y \in D$ such that $\tau(x, y)$, $\text{val}(x) = n$ and $\text{val}(y) = m$. Lastly, whenever $\tau(x, y)$, we have $f(\text{val}(x)) = \text{val}(y)$.

As in the language case, there is a more restrictive definition: τ **strongly represents** f iff in addition, for all $x, y \in D$ such that $\text{val}(x) = n$ and $\text{val}(y) = m$ we already have $\tau(x, y)$.

This may seem more natural than our definition, but it makes the transducers more complicated.

Before we tackle rational relations and functions, here is a little finger exercise for regular sets of numbers.

Fix some numeration system $\mathcal{N} = \langle \Delta, D, \text{val} \rangle$. For any modulus $m \geq 2$, define the **divisibility language** as follows:

$$D_{m,\mathcal{N}} = \{ x \in D \mid m \mid \text{val}(x) \}$$

Since $D_{m,\mathcal{N}}$ is fairly simple, one might expect that for any reasonable numeration system all these divisibility languages are regular.

It turns out that plain base B is easier to deal with wrto divisibility.

Since we only care about values modulo m it is natural to pick $Q = \{0, 1, \dots, m-1\}$ as state set, $q_0 = 0$, $F = \{0\}$. In radix B we have

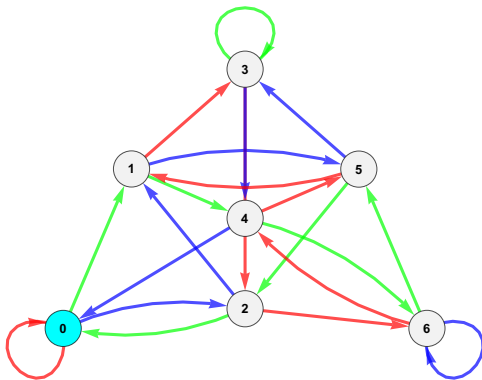
$$\text{val}(x d) = B \text{val}(x) + d \pmod{m}$$

which directly yields the deterministic transition function

$$\delta(p, d) = p B + d \bmod m$$

We will call these machines **Horner automata**, in symbols $\mathcal{H}_{m,B}$.

Careful, though, $\mathcal{H}_{m,B}$ is not minimal in general, just think about $\mathcal{H}_{B,B}$. It's the obvious machine that works, but it typically has redundant states.



Automaton $\mathcal{H}_{7,3}$; edge colors: red-0, green-1, blue-2.

Warning: the red edge from 4 to 2 actually starts at 3.

The infinitely superior LSD first system here causes a bit of a hiccup since

$$\text{val}(xd) = \text{val}(x) + d B^{|x|} \pmod{m}$$

so the machine needs to keep track of the numbers $B^{|x|} \bmod m$.

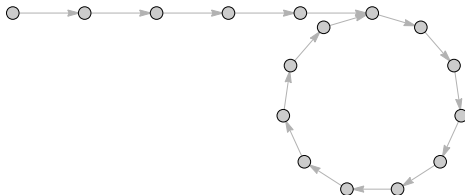
The sequence $(B^k \bmod m)_{k \geq 0}$, depends very much on divisibility properties of m and B . Without worrying about the details, it has to be ultimately periodic: there are finite sequences α and β over \mathbb{N} such that $(B^k \bmod m)_{k \geq 0} = \alpha \beta^\omega$. In fact, $|\alpha| + |\beta| \leq m$. We call $\alpha; \beta$ the fundamental sequence of B and m .

We can wrap this up in a fairly nice algebraic way.

Here is a function that describes the position of a particle moving on a lasso with transient t and period p .

$$\text{rem}_{t,p}(i) = \begin{cases} i & \text{if } i < t, \\ t + (i - t) \bmod p & \text{otherwise.} \end{cases}$$

Write $\text{succ}(i) = \text{rem}_{t,p}(i + 1)$ and write $\mathbb{N}_{t,p}$ for the structure $\langle \{0, 1, \dots, t+p-1\}, \text{succ} \rangle$. A picture of $\mathbb{N}_{5,11}$:

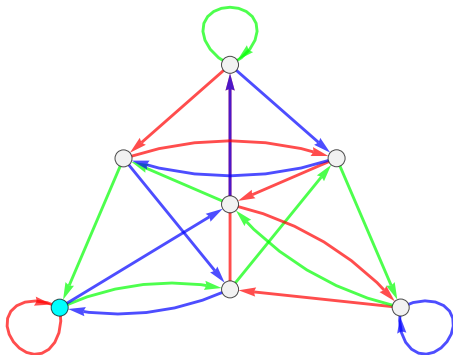


To build the divisibility machine for reverse base B modulo m , let $\alpha; \beta$ be the fundamental sequence, $t = |\alpha|$ and $p = |\beta|$.

We can now build a divisibility automaton $\mathcal{H}_{m,B}^r$ as follows:

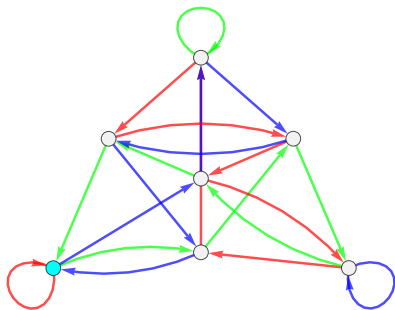
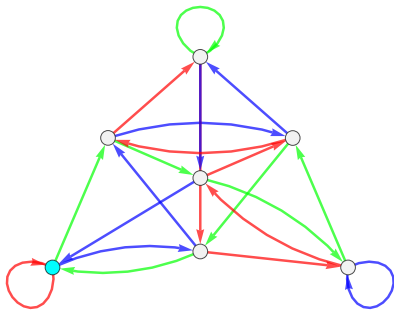
$$\begin{aligned} Q &= \{0, \dots, m-1\} \times \mathbb{N}_{t,p} \\ \delta((p, s), d) &= (p + d \beta_s \bmod m, \text{succ}(s)) \\ q_0 &= (0, 0) \\ F &= \{ (0, s) \mid s \in \mathbb{N}_{t,p} \} \end{aligned}$$

Note that this machine has size $O(m^2)$, though the accessible part can be smaller and the minimal machine can be smaller yet.



$\mathcal{H}_{7,3}^r$ has $42 = 7 \cdot 6$ states

fundamental sequence ; 1, 3, 2, 6, 4, 5 (transient 0, period 6).



Plain base 3 on the left, reverse base 3 on the right.

Both machines are minimal.

From the picture, $\min \mathcal{H}_{7,3}^r$ is the reversal of $\mathcal{H}_{7,3}$.

To see why, note that $\mathcal{H}_{7,3}$ is a **permutation automaton**: all the transition functions δ_a are permutations of the state set.

By Brzozowski, that means that $\text{pow}(\mathcal{H}_{7,3}^{\text{op}})$ is the minimal automaton for reverse base 3. But $\mathcal{H}_{7,3}^{\text{op}}$ is a DFA, done.

Careful, though, it is not true in general that $\mathcal{H}_{m,B}$ is a permutation automaton.

1 Numeration Systems

2 **Automatic Structures**

3 Deciding Presburger Arithmetic

Suppose we have some mathematical structure $\mathcal{X} = \langle X; R \rangle$ where R is a binary relation on X . Generalization to multiple relations/functions are obvious.

Definition

\mathcal{X} is **automatic** if there is a regular language $Nm \subseteq \Sigma^*$ and a surjective **naming function** $\nu : Nm \rightarrow X$ such that

1. the binary relation on Nm , $\nu(u) = \nu(v)$, is synchronous,
2. the binary relation on Nm , $R(\nu(u), \nu(v))$, is synchronous.

Nm is the set of names for the actual elements in \mathcal{X} .

One does not require the naming function to be injective, the same object in X may have several (even infinitely many) names. If ν is a bijection, then the first condition is trivially satisfied.

There is no condition on ν being computable in some simple manner, it just has to be defined on a regular set of words and be compatible with $=$ and R as in the definition.

In many concrete cases there is a “natural” choice for ν and N_m , but that is not part of the definition of automaticity.

This definition of automaticity is the correct one in the sense that it produces the desired result.

Note that we require our FSM representation to be fairly strong: we insist on a synchronous automaton \mathcal{A}_R such that for any $x, y \in X$ we must have have

$$R(x, y) \iff \forall u \in \nu^{-1}(x), v \in \nu^{-1}(y) \ (\mathcal{A}_R(u, v) \downarrow)$$

In other words, any name works, we don't need to pick a nice one for the automaton to produce the right answer.

The idea of automaticity is half a century old.

Bernard R. Hodgson

Théories décidables par automate fini

Ph.D. thesis, 1976, Université de Montréal

Sadly, no one paid any attention until it was reinvented 20 years later.

A broad study of automaticity really started taking off around 2000.

Hodgson already had a number of interesting examples: dense and discrete linear orders, Presburger arithmetic, p -adic numbers.

He showed that automatic structures have certain closure properties wrto product constructions.

And, he realized that one can use the same approach based on automata operating on infinite words (ω -automata).

Clever title of a book published in 1992 by D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Patterson, and W. P. Thurston.

This approach uses the Cayley graph of a group rather than the usual first-order structure and there are subtle differences between the two methods.

They provide a quadratic time algorithm based on finite state machines that solves the word problem for certain groups that are important in low-dimensional topology.

1 Numeration Systems

2 Automatic Structures

3 **Deciding Presburger Arithmetic**

We want to think of

$$\mathfrak{N}_+ = \langle \mathbb{N}; + \rangle$$

as an automatic structure.

So, we need a naming map $\nu : \text{Nm} \rightarrow \mathbb{N}$ where $\text{Nm} \subseteq \Sigma^*$ is some regular language.

No problem, any of the standard numeration systems will work. And, it does not matter which one we choose as far as decidability results are concerned.

It does matter, though, for the actual algorithm, the machines will be somewhat different.

We will adopt the following conventions for the representation of \mathbb{N} in 2^* .

- reverse binary (LSD first)
- trailing zeros are allowed (TZ)
- $0 \in \mathbb{N}$ is denoted by 0, not the empty word (NE)

In this case, $Nm = (0 + 1)^+$. Since we allow trailing zeros, we can avoid padding and work over 2 (more precisely, we pad with zero). The naming map is typically called the value map here, it associates a digit string with a numerical value:

$$\text{val} : 2^+ \longrightarrow \mathbb{N}$$

This map is most natural, but infinite-to-one.

At any rate, in our case, there are only two possible atomic formulae:

- $x = y$
- $x + y = z$
- $x < y$

The order relation $x < y$ is definable, but it helps to improve the expressiveness of the language to add it as a primitive.

So we have 3 synchronous transducers $\mathcal{A}_{=}^{(2)}$, $\mathcal{A}_{<}^{(2)}$ and $\mathcal{A}_{+}^{(3)}$ (2, 2, and 3 tracks, respectively) that test these predicates, given arbitrary names for the natural numbers in question.

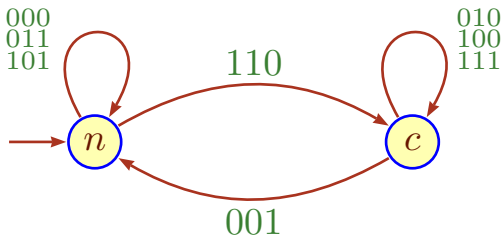
Clearly we could also handle the constants 0 and 1 (and arbitrary constants for that matter).

Again, we don't have to include $\mathcal{A}_{<}^{(2)}$ in our list of basic machines, we could replace every occurrence of an atomic formula $s < t$ by the formula

$$\forall d (s + d = t \wedge s \neq t)$$

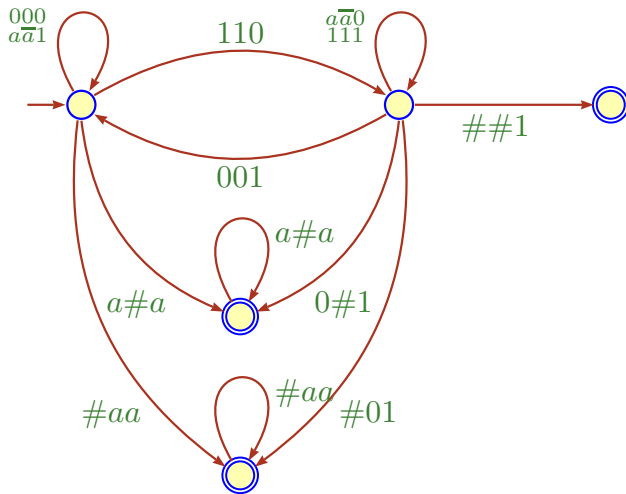
The decision algorithm would build a FSM that handles this condition, but it is better to lovingly handcraft an optimized machine once and for all.

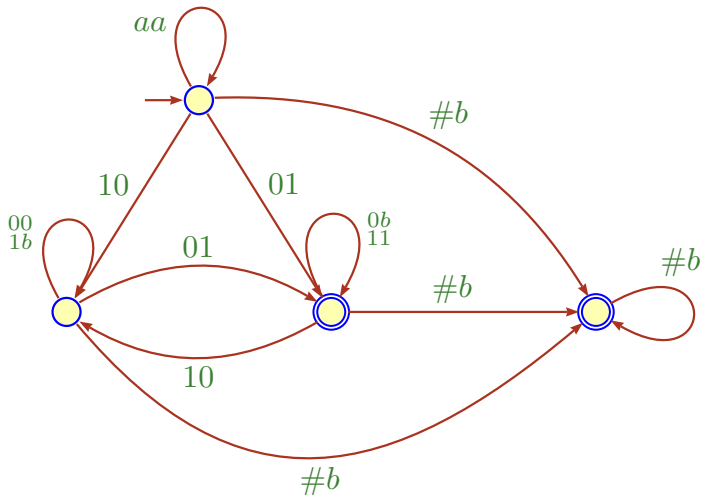
More generally, one probably would like to have a little library of optimized machines for phrases that appear often.



This is the core of the transducer for addition, with states “no carry” and “carry,” using reverse binary as the numeration system.

Alas, this transducer ignores all other aspects of a numeration system (empty-word, trailing-zeros).





The machines on the next few slides do not use a padding symbols; instead we cheat by padding with 0s. They are correct in spirit, but actually wrong as far as our definitions go.

To wit, they require the right number of trailing zeros in the names chosen to represent the natural numbers in question. That violates the definition of automaticity.

Exercise

Figure out how to fix this problem by using a padding symbol.

Suppose we want to express relational composition $\exists y (R(x, y) \wedge R(y, z))$. Suppose we have corresponding names $u, v, w \in \text{Nm}$.

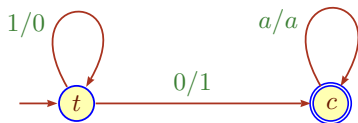
With our type of machine it may happen that

$$\mathcal{A}_R \text{ accepts } u:v \text{ and } v0^{42}:w$$

But the 3-track product machine does not accept $u:v:w$.

The construction fails since we cannot cope with 2 different representations for the witness.

The key component for the successor transduction is the following machine:

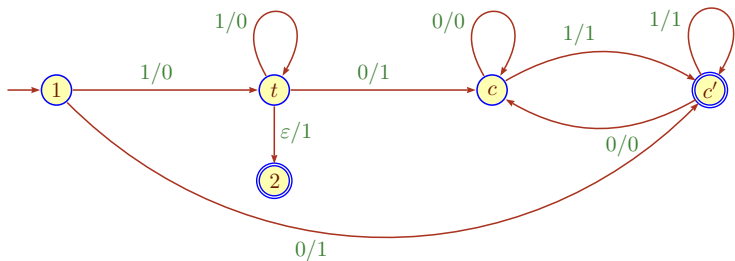


Input is supposed to be in reverse base 2.

Here t stands for toggle, c for copy.

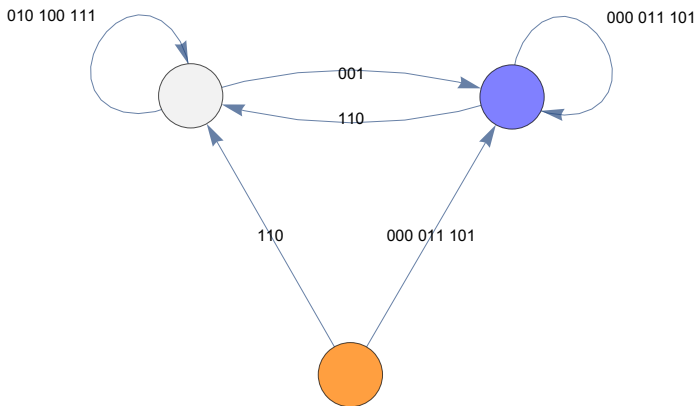
This transducer allows for trailing zeros.

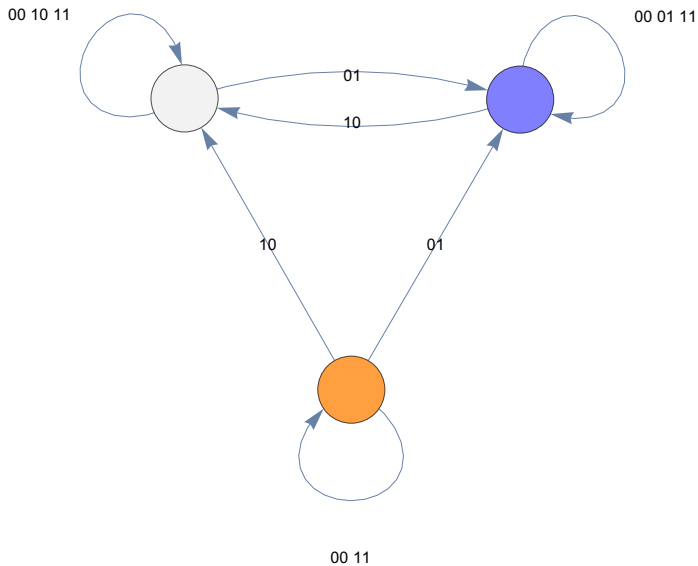
Alas, it implements a cyclic counter: $1^k \mapsto 0^k$.

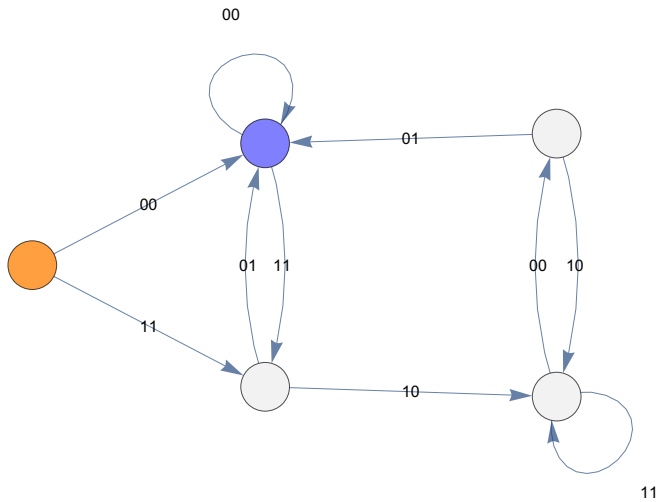


Think of the ϵ as a $\#$.

This version is obtained by a bit of surgery on the basic machine, and does not use the empty word for 0, nor allows for trailing zeros.

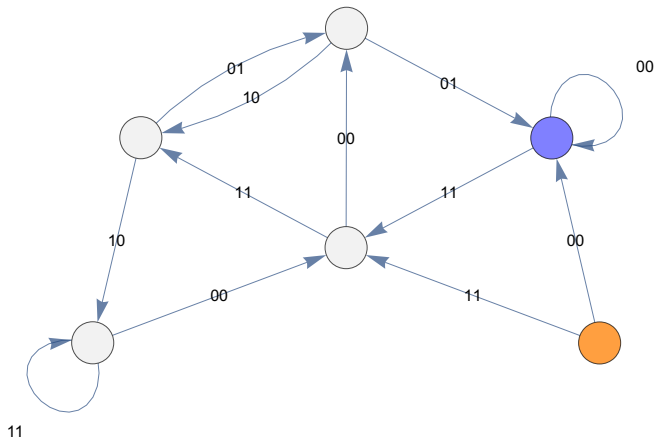






Multiply by 5

46



Consider the sentence

$$\Phi \equiv \exists x \forall y (x < y \Rightarrow \exists u, v (3 * u + 5 * v = y))$$

Thanks to our brilliant choice of coefficients, Φ is actually true.

We use a 5-track machine \mathcal{A} with the intended meaning:

u existential quantifier

v existential quantifier

u' $u' = 3u$

v' $v' = 5v$

z $z = u' + v'$

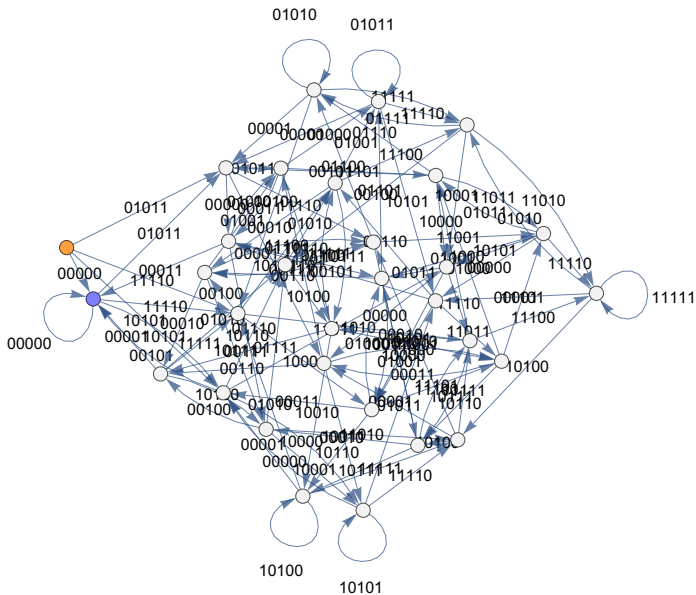
We combine a multiply-by-3 and a multiply-by-5 machine with an adder:

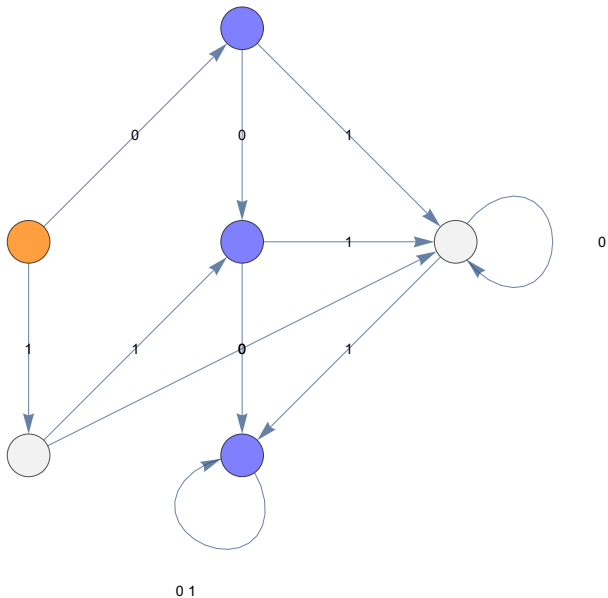
$$\mathcal{A} = \text{emb}_{1,3}^{(5)}(\text{mult}_3) \times \text{emb}_{2,4}^{(5)}(\text{mult}_5) \times \text{emb}_{3,4,5}^{(5)}(\mathcal{A}_+)$$

Projecting away all but the z -track

$$\mathcal{B} = \text{prj}_{1,2,3,4}^{(5)}(\mathcal{A})$$

produces a plain FSM that recognizes all linear combinations of 3 and 5.





It now suffices to check that

$$\Phi \equiv \exists x \forall y (x < y \Rightarrow \mathcal{B}(y))$$

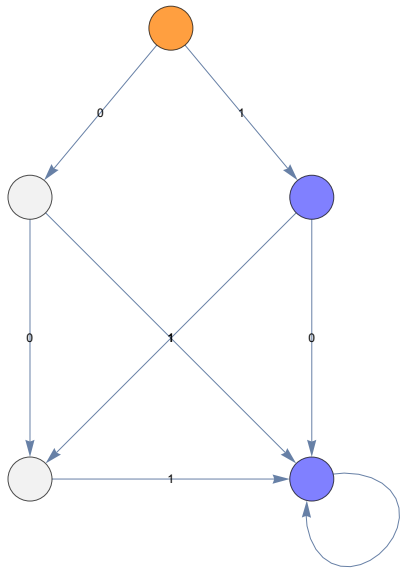
The universal quantifier needs to be rewritten:

$$\Phi \equiv \exists x \neg \exists y (x < y \wedge \neg \mathcal{B}(y))$$

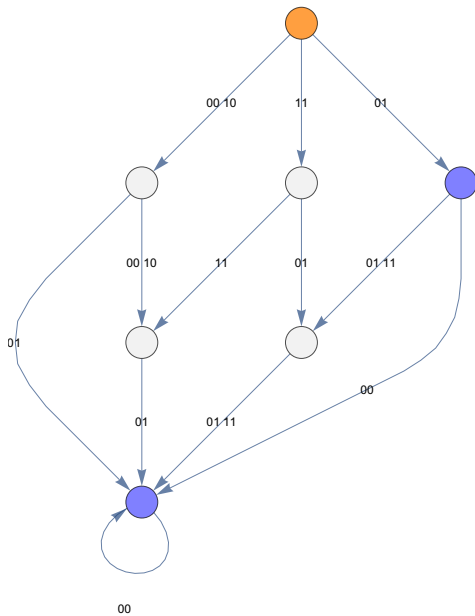
The $\exists y (x < y \wedge \neg \mathcal{B}(y))$ part is handled by more embeddings:

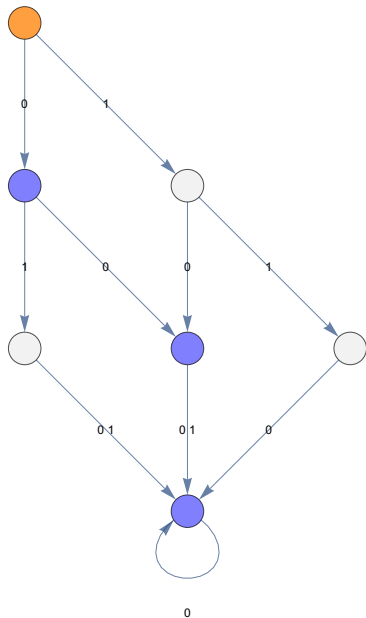
$$\mathcal{C} = \text{emb}_{1,2}^{(2)}(\text{less}) \times \text{emb}_2^{(2)}(\neg \mathcal{B})$$

For the complement we have to deal with the numeration system; only strings in Nm are allowed.



$\neg\mathcal{B}$ recognizes 1, 2, 5, 6.





least number not accepted:
7

hence witness $x = 7$ works

Since automatic presentations depend on a naming function $\nu : Nm \rightarrow X$ it is far from clear whether two automatic presentations describe the same underlying first-order structure. This is known as the **isomorphism problem**.

Theorem

The isomorphism problem for automatic structures is undecidable.

In fact, the problem is outside of the arithmetical hierarchy and belongs to the analytical hierarchy, at level Σ_1^1 .

By the same token, it is fairly difficult to make sure that a given structure \mathcal{X} fails to be automatic. Here are some examples of non-automatic structures:

- Additive rational numbers $\langle \mathbb{Q}; + \rangle$
- Divisibility of naturals $\langle \mathbb{N}; | \rangle$
- Skolem arithmetic $\langle \mathbb{N}; \cdot \rangle$
- Free semigroup $\{a, b\}^*$
- Structures with a pairing function

Let \mathcal{A} and \mathcal{B} be two relational, first-order structures with signatures $\tau(\mathcal{A})$ and $\tau(\mathcal{B})$. The signatures may well be different, so their first-order theories use a different vocabulary.

For simplicity, assume that \mathcal{A} has just one binary relation R .

Definition

An *n -dimensional interpretation of \mathcal{A} in \mathcal{B}* consists of first-order formulae in the language of \mathcal{B}

$$\delta(\mathbf{x}) \quad \phi_R(\mathbf{x}, \mathbf{y})$$

where \mathbf{x} and \mathbf{y} are n -vectors of distinct variables. Furthermore, there is a surjective *coordinate map* $f : \delta^{\mathcal{B}} \rightarrow A$ such that

$$\mathcal{A} \models R(f\mathbf{b}_1, f\mathbf{b}_2) \iff \mathcal{B} \models \phi_R(\mathbf{b}_1, \mathbf{b}_2)$$

We can associate any formula Φ in the vocabulary of \mathcal{A} with an interpretation $\Phi^{\mathcal{I}}$, a formula in the vocabulary of \mathcal{B} defined as follows:

- Replace any atomic formula $R(u, v)$ by $\phi_R(\mathbf{x}, \mathbf{y})$.
- Keep the propositional part in tact.
- Relativize every quantifier to δ :

$$\forall u \phi(u) \rightsquigarrow \forall \mathbf{x} (\delta(\mathbf{x}) \Rightarrow \phi(\mathbf{x}))$$

$$\exists u \phi(u) \rightsquigarrow \exists \mathbf{x} (\delta(\mathbf{x}) \wedge \phi(\mathbf{x}))$$

Theorem

Let \mathcal{B} interpret \mathcal{A} with coordinate map \mathcal{I} . Then for any first-order formula $\Phi(x)$ in the vocabulary of \mathcal{A} and $b \in \mathcal{B}$:

$$\mathcal{A} \models \Phi(fb) \iff \mathcal{B} \models \Phi^{\mathcal{I}}(b)$$

In particular for a sentence we have

$$\mathcal{A} \models \Phi \iff \mathcal{B} \models \Phi^{\mathcal{I}}$$

So all first-order questions about \mathcal{A} can be answered in \mathcal{B} .

In particular, if we have a decision algorithm for \mathcal{B} we can also apply it to \mathcal{A} : just check $\Phi^{\mathcal{I}}$ instead of Φ .

Here is an extension of Presburger arithmetic based on the special divisibility relation for $m \geq 2$:

$$x \mid_m y \iff x \mid y \wedge \exists k (x = m^k)$$

$$\mathfrak{N}_m = \langle \mathbb{N}; +, \mid_m \rangle$$

\mathfrak{N}_m is still automatic: we can write numbers in reverse base m .

Addition is synchronous as always. The divisibility relation is also easily synchronous.

Write $\Delta_m = \{0, 1, \dots, m-1\}$ for the usual digit alphabet in base m .

We can think of all words over Δ_m as an m -ary tree. Let $\sigma_d(x) = xd$ be the d th child of x in that tree, write \prec for the prefix partial order on Δ_m^* and write $\text{el}(x, y)$ if x and y have the same length.

The **tree structure** of order m is defined by

$$\text{tree}_m = \langle \Delta_m^*; \sigma_0, \dots, \sigma_{m-1}, \prec, \text{el} \rangle$$

Theorem

The structures \mathfrak{N}_m and tree_m are mutually interpretable in each other.

Sketch of proof. To interpret, say, tree_m in \mathfrak{N}_m , associate a word $u = u_0 u_1 \dots u_{k-1}$ in Δ_m^* (i.e., a branch in the tree) with the numerical value

$$m^k + \sum_i u_i m^i$$

The m^k forces the MSD to be 1 and we have to choose δ accordingly. First note that for all x there is exactly one y such that

$$\alpha(x, x') \equiv x' \mid_m x' \wedge x' \leq x < mx'$$

Hence we can set

$$\delta(x) \equiv \exists x' (\alpha(x, x') \wedge x < 2x')$$

Using α , we can express $\sigma_d(u) = v$ as

$$\psi_d(x, y) \equiv \exists x' (\alpha(x, x') \wedge y = x + (m + d - 1)x')$$

The prefix relation translates into

$$\pi(x, y) \equiv \exists x', z (\alpha(x, x') \wedge x' \mid_m z \wedge y = x + z)$$

Lastly, equal length/equal level translates into

$$\lambda(x, y) \equiv \exists x', y' (\alpha(x, x') \wedge \alpha(y, y') \wedge x' = y')$$

Done.



Here is a rather surprising result that states that, in the sense of interpretability, the structures \mathfrak{N}_m and tree_m are the most complicated automatic structures.

Theorem

A structure is automatic iff it can be interpreted in \mathfrak{N}_m iff it can be interpreted in tree_m .

So, as a matter of principle, we could solve all first-order queries about automatic structures by a translation to a carefully optimized decision algorithm for \mathfrak{N}_m and/or tree_m .