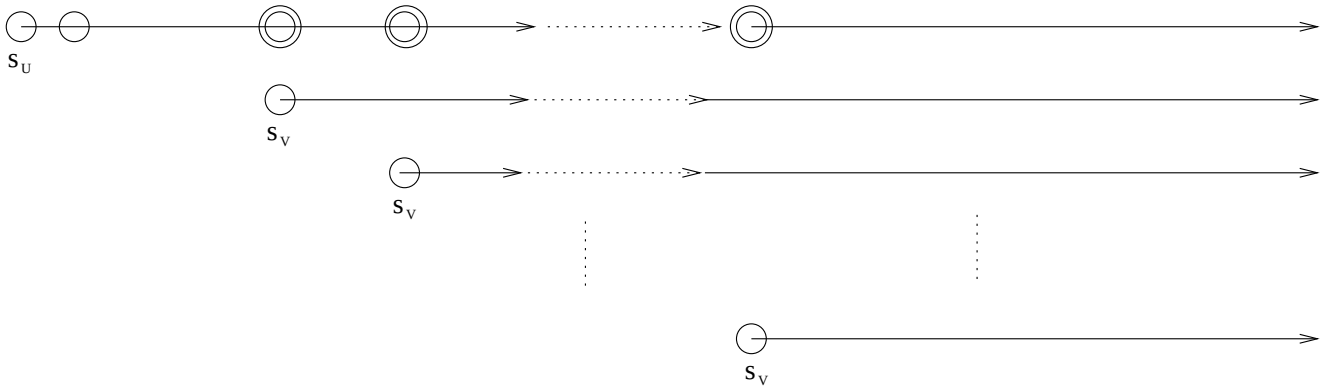
Lecture 8: Determinizing Büchi Automata

At the end of the last lecture we proved that every ω-regular language can be expressed as a finite union of languages of the form $U.\widehat{V}$. Let us try and design deterministic automata for a language of the form $U.\widehat{V}$ (This automaton cannot be a Büchi automaton. But it will differ from a Büchi automaton only in the definition of accepting runs.) Let $A_U = (Q_U, \Sigma, \delta_U, s_U, F_U)$ and $A_V = (Q_V, \Sigma, \delta_V, s_V, F_V)$ be two deterministic finite automata accepting the languages U and V respectively. Note that A_V, considered as a Büchi automaton, recognises $\widehat{V}$.

Exercise: Show that $\widehat{U.V}$ need not be equal to $U.\widehat{V}$ in general.

What we need is a sort of sequential composition of A_U and A_V. However, a given ω-word σ may have several (possibly infinitely many) prefixes that belong to U and not all of them may act as a witness to the membership of σ in $U.\widehat{V}$. That is, even though $\sigma \in U.\widehat{V}$, it may be possible to write $\sigma = u.\sigma'$ where $u \in U$ but $\sigma' \notin \widehat{V}$. So, our sequential composition must be able to pick the "right" prefix of σ. Of course, one can construct an automaton that nondeterministically picks some prefix of σ that belongs to U and verifies that the rest of the word is in $\widehat{V}$. But, we are looking for a deterministic automaton.

The right idea is to carry out a "powerset" construction on this nondeterministic automaton. Our automaton keeps a copy of A_U and whenever it identifies that a prefix of σ is in U, it forks a copy of A_V to read the rest of the input. A word is accepted if any one of these copies of A_V visits its final state infinitely often.

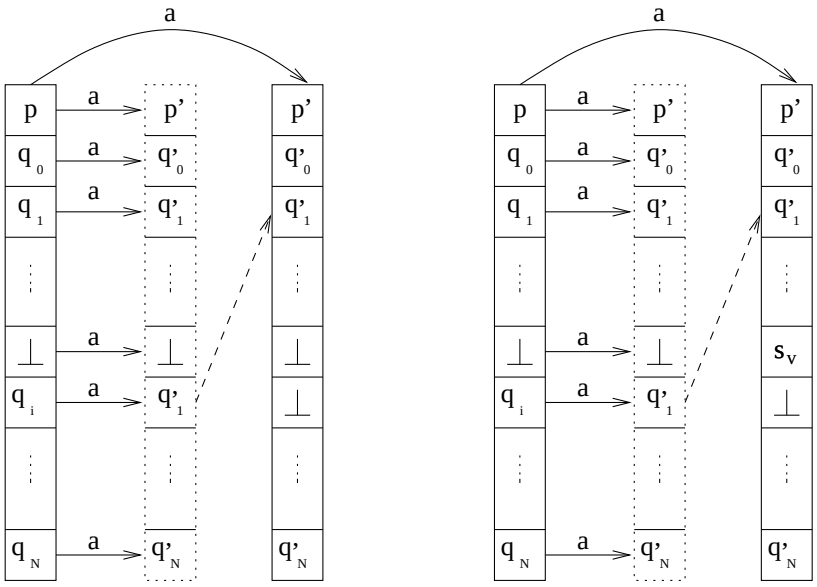


The top line is the "run" of A_U on σ. Note that at every prefix where A_U enters an accepting state, a new copy of A_V is started off. As it stands, this construction is not finite state as there may be even infinitely many prefixes of σ in U and so there would be unboundedly many copies of A_V.

The future behaviour of an automaton is completely determined by its current state. Thus, if two of these forked copies reach the same state at some point we might as well merge these two copies. Thus, at any point we will have at the most N copies where N

is the number of states in A_V. With these intuitions in mind, we propose the following construction. Let $A = (Q, \Sigma, \delta, s, \emptyset)$ be the deterministic automaton defined as follows: Q consists of elements of $Q_U \times (Q_V \cup \{\perp\})^{N+1}$ in which no element of Q_V repeats. We set $s = (s_U, \perp, \perp, \ldots, \perp)$ if $\epsilon \notin U$ and set $s = (s_U, s_V, \perp, \perp, \ldots, \perp)$ otherwise. The state is designed to carry one state of A_U and upto $N+1$ states of A_V (We only need N. The reason for using $N+1$ will be apparent soon.) We use the symbol $\perp$ to denote that a particular coordinate in the state is not currently carrying a copy of A_V.

Let $(p, q_0, q_1, \ldots, q_N)$ be an element of Q and a be in Σ. Suppose $p \xrightarrow{a} p'$ and $q_i \xrightarrow{a} q'_i$ (where we assume $\perp \xrightarrow{a} \perp$ for each $a \in \Sigma$). Note that, at least one of the q_is was $\perp$ and so at least one of the q'_is will also be $\perp$. Suppose j is the smallest index with $q'_i = \perp$. If $p' \in F_U$ then we set $q'_j = s_V$. Finally, if a state q appears more than once in $q'_0, q'_1, \ldots q'_N$ then, replace everything except the lowest index copy by $\perp$. Let the resulting tuple be $(p', q''_0, q''_1, \ldots, q''_N)$. We set $\delta((p, q_0, q_1, \ldots, q_N), a) = (p', q''_0, q''_1, \ldots q''_N)$.

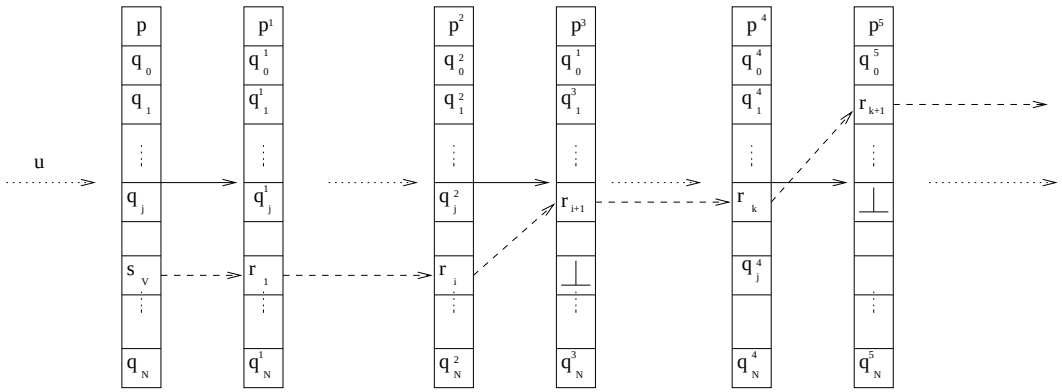


In the picture above, the left handside illustrates a move when $p' \notin F_U$ and the right handside illustrates the case when $p' \in F_U$. Note, how q'_i is replaced by $\perp$ since $q'_i = q'_1$. The run of the copy of A_V at coordinate i has been "handed over" to copy at coordinate 1. We shall refer to this as i *merges* with 1. Observe that whenever i merges with j, $i > j$.

Note that if the ith coordinate merges with the jth at a move, the resulting state necessarily has $\perp$ at coordinate i (If we did not use $N+1$ copies this would not be guaranteed. If we had only N copies then we might have to use this position to start a new copy of A_V.) Another point to note is that a newly created copy might immediately merge and thus "may not start at all". This happens if one of the other components is already at state s_V.

We have not specified an acceptance condition as yet and left it as the empty set. Let us examine the runs of this autmaton on words in $U.\widehat{V}$. Suppose $\sigma = u.\sigma'$ with $u \in U$ and $\sigma' \in \widehat{V}$. After reading u, suppose our automaton is at state $(p, q_1, \ldots, q_{i-1}, s_V, \perp, \ldots, q_N)$ (the state reached on reading u is guaranteed to be of this form since $u \in U$ and from the

definition of δ). Suppose $s_V \longrightarrow r_1 \longrightarrow r_2 \longrightarrow \ldots$ is the run of A_V on σ'. This run visits the set F_V infinite often. The copy at coordinate i that reads σ' would also follow the run $r_1 \longrightarrow r_2 \longrightarrow$, however it is not guaranteed that this copy would stay at coordinate i as it may merge with a lower numbered coordinate and merge again and so on. Here is how the corresponding run might look:



Note how the dashed line of the "accepting run" of the copy of A_V forked on reading u merges with coordinate j and then from there with coordinate 1.

But the key point to note is the following, since a coordinate merges only with a lower numbered coordinate, a copy changes coordinate only finitely often. Thus there is a finite number of moves, say n_0, after which the copy of A_V forked on reading u stays at some coordinate j forever. Thus the states $r_{n_0}, r_{n_0+1}, \ldots$ appear consecutively as the jth component of the states in some suffix of the run of A on σ. For this j, we have that

1. States from F_V appear infinitely often at coordinate j. (In particular this happens infinitely after the copy forked on reading u stabilises at coordinate j).

2. The symbol $\perp$ appears only finitely often at coordinate j.

We shall now argue that the converse is also true. Suppose the run of A on σ satisfies these two properties w.r.t. some coordinate j. Then, the run can be written in the form $s \xrightarrow{u} s' \xrightarrow{\sigma'}$ where, $u \in U$, the jth component of s' is s_V and further the jth component of every state that appears to the right of this state is not $\perp$. (A proof of this is quite easy. There are two cases to consider. If the jth component was $\perp$ sometime along the run, then take the s' to be the successor of the state where $\perp$ appears for the last time at coordinate j. Otherwise, take $s' = s$.) Suppose the jth coordinate of the states in the run $s' \xrightarrow{\sigma'}$ are $r_1, r_2 \ldots$ then, it is clear from the definition of δ that $r_1 \xrightarrow{\sigma'_1} r_2 \xrightarrow{\sigma'_2} \longrightarrow$ is the run of A_V on σ'. By requirement 2, this run visits the set F_V infinitely often. Thus $\sigma' \in \widehat{V}$ and thus $\sigma \in U.\widehat{V}$.

For each coordinate j, we define two ω-regular languages W_i and W'_i as follows:

$$\begin{aligned} W_i &= \{\sigma \mid \text{In the run } \rho \text{ of } A \text{ on } \sigma, \text{ the } i\text{th component visits } F_V \text{ infinitely often}\} \\ W'_i &= \{\sigma \mid \text{In the run } \rho \text{ of } A \text{ on } \sigma, \text{ the } i\text{th component is } \perp \text{ infinitely often}\} \end{aligned}$$

By the above discussion, $U.\widehat{V} = \bigcup_{0 \leq i \leq N} W_i \cap \overline{W_i'}$. Further, both W_i and W_i' are accepted by deterministic Büchi Automata. For W_i we use the deterministic automaton A with $F_i = \{(p, v_0, \ldots, v_N \mid p \in Q_U,\ \forall j.\ v_j \in A_V \text{ and } v_i \in F_V\}$ as the set of good/accepting states and for W_i' we use $E_i = \{(p, v_0, \ldots, v_{j-1}, \perp, v_{j+1}, \ldots, v_N) \mid p \in Q_U \text{ and } \forall j.\ v_j \in Q_V\}$ as the set of good/accepting states.

Theorem 1 *Every ω-regular language is a boolean combination of limit languages.*

Proof: From the above discussion, any language of the form $U.\widehat{V}$ can be written as a boolean combination of limit languages. We showed in the previous lecture that every ω-regular language is a finite union of languages of the form $U.\widehat{V}$. ■

1 Rabin and Müller Automata

We just showed that every ω-regular language can be expressed as a union of lanuages of the form $W_i \cap \overline{W_i'}$, where W_i and W_i' are limit languages accepted by Büchi automata which differ merely in the set of good/accepting states. Rabin suggested an accepting criterion that directly reflects this requirement.

Definition 2 *A Rabin automaton A is of the form $A = (Q, \Sigma, \delta, s, (E_1, F_1), (E_2, F_2) \ldots (E_k, F_k))$ where Q, Σ, δ and s are as in the case of Büchi automata. Each E_i and F_i is a subset of Q. Runs of such an automaton are defined as in the case of Büchi automata. A run ρ is accepting if there is an i, $1 \leq i \leq k$ such that $inf(\rho) \cap E_i = \emptyset$ and $inf(\rho) \cap F_i \neq \emptyset$. Thus a run is accepting if and only if there is an i such that, the set E_i is visited finitely often and F_i is visited infinitely often.*

A Rabin automaton is deterministic if δ is a transtion function. Thus the automaton constructed above, A, with $(E_0, F_0), \ldots (E_N, F_N)$ as the set of accepting pairs (where the definition of E_i and F_i is given at the end of the last section) is a deterministic Rabin automaton that accepts $U.\widehat{V}$.

Earlier Müller had proposed that acceptance be defined by directly specifying the $\inf(\rho)$s permitted.

Definition 3 *A Muller automaton A is of the form $A = (Q, \Sigma, \delta, s, \mathcal{F})$ where $\mathcal{F} \subseteq 2^Q$. A run ρ is accepting if $inf(\rho) \in \mathcal{F}$. As usual, such an automaton is said to be deterministic if δ is a function.*

We can equip A with the following $\mathcal{F}$

$$\mathcal{F} \quad = \quad \{X \subseteq Q \mid \exists i.\ X \cap E_i = \emptyset \ \wedge \ X \cap F_i \neq \emptyset\}$$

Quite clearly this automaton accepts a word precisely when the Rabin automaton described earlier accepts a word and that is precisely when the word is in $U.\widehat{V}$. (Notice that

this construction allows us to turn any Rabin automaton into an equivalent Müller automaton.) Thus, if we can show that deterministic Müller automata are closed under union, that would complete the proof showing that all ω-regular languages are accepted by deterministic Müller automata, giving us the desired determinzation construction.

Lemma 4 *If L_1 and L_2 are accepted by deterministic Müller automata then $L_1 \cup L_2$ is accepted by a deterministic Müller automaton.*

Proof: Let $A_1 = (Q_1, \Sigma, \delta_1, s_1, \mathcal{F}_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, s_2, \mathcal{F}_2)$ be deterministic Müller automata accepting L_1 and L_2 respectively. Let $A = (Q_1 \times Q_2, \Sigma, \delta_1 \times \delta_2, (s_1, s_2), \mathcal{F})$, where $\mathcal{F} = \{X \subset Q \mid X\downarrow_1 \in \mathcal{F}_1\} \cup \{X \subseteq Q \mid X\downarrow_2 \in \mathcal{F}_2\}$ (where $X\downarrow_i$ is the projection to coordinate i of X). It is quite easy to check that A accepts $L_1 \cup L_2$. ■

Exercise: Show that deterministic Rabin automata are closed under union.

With this we have established that all ω-regular languages are accepted by Deterministic Müller automata. But can they accept more? Notice that the language accepted by a deterministic Müller automaton $A = (Q, \Sigma, \delta, s, \mathcal{F})$ is a boolean combination of the languages accepted by the Deterministic Büchi automata A_q, $q \in Q$, given by $A_q = (Q, \Sigma, \delta, s, \{q\})$. In particular,

$$L(A) \quad = \quad \bigcup_{F \in \mathcal{F}} \left(\bigcap_{q \in F} L(A_q) \cap \bigcap_{q \notin F} L(A_q)\right)$$

Exercise: Why does this construction fail in the case of Nondeterministic Muller automata?

Thus we have established that deterministic Müller automata accept precisely the class of ω-regular languages. We now show that even nondeterministic Müller automata accept ω-regular languages.

Lemma 5 *Let $A = (Q, \Sigma, \delta, s, \mathcal{F})$ be a Müller automaton. Then, $L(A)$ is ω-regular.*

Proof:(Sketch) We construct a NBA accepting $L(A)$ as follows: the NBA simulates the A. Further it nondeterministically guesses a set $X \in \mathcal{F}$ and a position in the run and verifies that after this point, every state in X is hit infinitely often and no state outside of X is visited. How do we check that everything in X is hit infinitely often? Order the set X as say $x_1, x_2, \ldots x_k$. The automaton keeps a current index from the set $\{0, 1, 2, \ldots k\}$. Whenever the automaton visits a state x_i and the current index is $i > 0$, the current index is incremented to $i + 1$ (modulo k+1) otherwise the index is left unchanged. If the current index is 0, the index is always incremented to 1 in the next move. We use a Büchi condition to simply check that the current index is 0 infinitely often. ■

Exercise: Write down the construction described in the proof of Lemma 5 precisely.

Theorem 6 *The following statements are equivalent:*
1. L is a ω-regular language.
2. L is a finite union of languages of the form $U.\widehat{V}$.
3. L is a boolean combinated of languages of the form $\widehat{V}$.
4. L is accepted by some nondeterministic Büchi automaton.
5. L is accepted by some nondeterministic Müller automaton.
6. L is accepted by some nondeterministic Rabin automaton.
7. L is accepted by some deterministic Müller automaton.
8. L is accepted by some deterministic Rabin automaton.

Proof: The equivalence of the first 4 statements was proved earlier. We have just shown that NBA can be translated to DMA and that NMA can be translated to NBA. Thus NBA, NMA and DMA are equivalent. We also showed that NRA can be translated to NMA. One of the exercises (showing that DRAs are closed under union) shows that NBAs can be translated to DRAs. That established the equivalence of NRAs and DRAs with ω-regular languages. ■

2 MSO over infinite words

We can interpret the logic MSO over infinite words. The first order variables $x, y, \ldots$ are interpreted as positions in the word, second order variables $X, Y, \ldots$ are interpreted as sets of positions, $x \in X$, $a(x)$ etc. have the usual interpretations. For instance, the following formula

$$\forall x.\exists y.\ (x < y) \wedge a(x)$$

asserts that there are infinitely many a's.

Exercise: Write down a formula in MSO that asserts that there is some subword of the form $ba^{2*n}b$ for some $n > 1$.

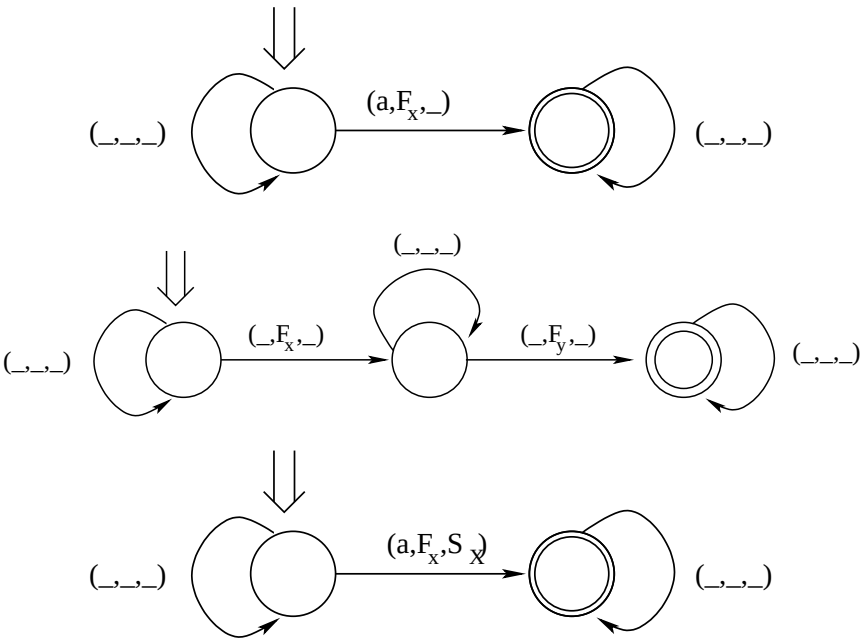
Let $A = (Q, \Sigma, \delta, s, F)$ be a NBA. Using one variable X_q for each state $q \in Q$, it is quite easy to write down a formula in MSO, that describes the set of words accepted by A. There are some differences w.r.t. to the proof for the finite case: Firstly, A may be nondeterministic. It is quite trivial to generalize the construction there to work over nondeterministic automata. Secondly, we need to describe the Büchi acceptance rather than asserting that the automaton enters a final state at the end. Büchi acceptance is described by the following formula:

$$\forall x.\exists y.(x < y) \wedge \bigvee_{q \in F} (y \in X_q)$$

Thus, every ω-regular language can be described using sentences in MSO.

In order to describe the languages defined by formulas with free variables, we shall consider (V, W)-words where V is a set of first order variables and W is a set of second order variables (i.e. ω-words over the alphabet $\Sigma \times 2^V \times 2^W$ where each $x \in V$ appears exactly in one letter in the word). It is quite easy to construct a NBA accepting precisely the set of (V, W)-words. We now sketch an argument showing that for any MSO formula ϕ, and sets V and W with $\text{free}_1(\phi) \subseteq V$ and $\text{free}_2(\phi) \subseteq W$, the set of (V, W)-words satisfying ϕ is a ω-regular language.

The construction of a NBA automaton accepting $L(\phi)$ proceeds by induction on the structure of the formula. In all cases we assume that the autmaton is intersected with an automaton recognising the set of (V, W)-words. Here are the automata for $a(x)$, $x \in X$ and $x < y$ in that order:



These are the same automata as used in Lecture 3. We simply interpret then as NBAs here.

Since we have already established that Büchi automata are closed under boolean operations, translating the boolean operators is quite trivial and thus we are left with the quantifiers. As usual, we can express the universal quantifiers using the existential quantification and negation.

Suppose $\phi = \exists x.\alpha$. Let $\text{free}_1(\phi) \subseteq V$ and $\text{free}_2(\phi) \subseteq W$. Then $\text{free}_1(\alpha) \subseteq V \cup \{x\}$ and $\text{free}_2(\alpha) \subseteq W$. By the induction hypothesis, the set of $(V \cup \{x\}, W)$-words satisfying α is a ω-regular language. Further, $(a_1, F_1, S_1)(a_2, F_2, S_2) \ldots$ satisfies α if and only if $(a_1, F_1 \setminus \{x\}, S_1), (a_2, F_2 \setminus \{x\}, S_2) \ldots$ satisfies $\exists x.\alpha$. In the finite word case, we simply applied closure under homomorphism at this point. Since homomorphic images of infinite words might be finite words, we can't directly use homomorphism closure. However, it is rather trivial to construct a Büchi automaton that guesses a position for x and verifies that resulting word satisfies α. We leave the details as an exercise. The construction for $\exists X.\alpha$ also proceeds similarly.

Thus we have the following theorem due to J.R.Büchi.

Theorem 7 *A language L is ω-regular if and only if it can be described by a sentence in MSO.*

As an immediate corollary of this result we have:

Corollary 8 *The problem of checking whether a formula ϕ in MSO is satisfied by any word is decidable. The problem of checking whether a formula ϕ in MSO satisfied by all words is also decidable.*

Proof: We can translate ϕ into a Büchi automaton A such that $L(\phi) = L(A)$. The first question corresponds to checking emptiness of the automaton A, and the second corresponds to checking that $L(A) = \Sigma^*$. Both of which are decidable. ■

3 Applications of MSO

As an application of this theorem, we shall show that Presburger arithmetic, that is, the theory of natural numbers with addition, is a decidable theory.

Formulas in Presburger arithmetic use first order variables $x, y, \ldots$ that range over natural numbers and the predicate $x + y = z$. Here is a valid sentence in this logic:

$$\forall x.\forall y.\exists z.(x + z = y) \vee (y + z = x)$$

Here is a sentence that is not true:

$$\forall x.\forall y.\exists z.(x + z = y)$$

Notice that the relation $x < y$, $x = y + 1$, and so on can be defined in Presberger arithmetic. Presberger showed that the problem of determining whether a given sentence in Presburger arithmetic is valid (or true) is decidable, by the method of quantifier elimination. Büchi showed that one can translate sentences in Presburger arithmetic to sentences in MSO (over the one letter alphabet) preserving validity. We can then use Corollary 8 to conclude the decidability of Presburger arithmetic.

There is exactly one word over the one letter alphabet. Let the positions of this word be $0, 1, 2, \ldots$. Büchi's idea as to denote a natural number n by a subset of positions. Suppose $n = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k}$. Then, n is represented by the set $\{i_1, i_2, \ldots, i_k\}$. Conversely, every finite subset of $0, 1, 2, \ldots$ represents a number. The translation map $\mathcal{T}$ is defined as follows: $\mathcal{T}(\exists x.\phi) = \exists X.Fin(X) \wedge \mathcal{T}(\phi)$, $\mathcal{T}(\forall x.\phi) = \forall X.Fin(X) \Rightarrow \mathcal{T}(\phi)$, $\mathcal{T}(\phi \vee \phi') = \mathcal{T}(\phi) \vee \mathcal{T}(\phi')$, $\mathcal{T}(\neg\phi) = \neg\mathcal{T}(\phi)$, and $\mathcal{T}(x + y = z) = \mathsf{Add}(X, Y, Z)$. Here, $\mathsf{Fin}(X)$ simply asserts that the set X is finite and it is quite easy to write a formula saying this. That leaves the definition of $\mathsf{Add}(X, Y, Z)$. We need to show that there is MSO formula that verifies that the number reprepresented by the set Z is the sum of the numbers represented by the sets X and Y. One can write this down directly, but we prefer the route via automata. We need

to show that the set of $(\emptyset, \{X, Y, Z\})$-words satisfying the sum property described above is a ω-regular language. Each input letter can be thought of as consisting of 3 bits, one bit indicating whether the position is in X or not, and similarly one each for Y and Z. With this interpretation, the automaton is reading three bit sequences starting at the least significant bit and it needs to verify that one sequence is the sum of the other two. This is an easy programming exercise. Thus, we can translate formulas in Presburger arithmetic to MSO.

Theorem 9 *(Presburger/Büchi) The theory of natural numbers with addition is decidable.*

Things change quite drastically if we allow multiplication and this is one of the implications of Gödel's famous theorem.

As a second application of the decidability of MSO we turn to model checking. This is a problem that arises in computer science. You are given a program and a property of runs of programs and you wish to check that every run of the program satisfies this property. For example, the property could say "It is never the case that more than one process is in the critical section" or " whenever a process requests for a resource, it is eventually granted access to the resource". Such properties can be quite easily seen to expressible in MSO. On the program side, we know that almost everything is undecidable if we work with general programs. However, if we restrict the set of programs to finite state programs then we can think of the program as Buchi automaton (where all states are accepting) A_p and the property as a MSO formula that can also be translated in to a Büchi Automaton A_ϕ and the model checking problem reduces to checking if $L(A_p) \subseteq L(A_\phi)$. But this is quite easily seen to be decidable.

Exercise: What is the complexity of the complementation construction for NBAs described in this lecture?

It turns out that translating MSO into automata is prohibitively expensive. On the other hand, there are restricted logics, in which properties such as the two listed above are expressible, and that can be translated to automata with reasonable cost. These will be some of the topics we shall study later in this course.

It turns out that it is possible to construct a complement BA whose size is bounded by $O(2^{O(nlogn)})$. We shall examine this in the coming lectures. Note that the translations from one kind of ω-automata to another is also an expensive operation. We shall examine the exact complexity of these translations later.