Infinite Automata 2025/26
Lecture Notes 2

Henry Sinclair-Banks

Recall from the Exercise Sheet 1.

Lemma 2.1. Given a multiset A of n positive integers not greater than m and a multiset B of m positive integers not greater than n, there exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that $A', B' \neq \emptyset$ and

$$\sum_{a \in A'} a = \sum_{b \in B'} b.$$

Recall from Lecture 1.

Lemma 1.10. Let M be a 1-CM and let $(p, 0)$, $(q, 0)$ be two configurations. Let n be the number of states in M. There exist a polynomial f such that if $(p, 0) \xrightarrow{*}_M (q, 0)$, then there exist a run from $(p, 0)$ to $(q, 0)$ such that all configurations in the run have counter values at most $f(n)$.

Proof. Let $(p, 0) \xrightarrow{\pi} (q, 0)$ be the run (in M) which, among all other runs, has the least greatest counter value. Let (r, x) be the configuration in $(p, 0) \xrightarrow{\pi} (q, 0)$ with the greatest counter value. For the sake of contradiction, suppose that $x > 2n^2 + 2n$.

For convenience, suppose π_1 and π_2 are the prefix and suffix of π such that $(p, 0) \xrightarrow{\pi_1} (r, x) \xrightarrow{\pi_2} (q, 0)$. We will now examine $(p, 0) \xrightarrow{\pi_1} (r, x)$ in detail; symmetric arguments can be applied to $(r, x) \xrightarrow{\pi_2} (q, 0)$. Let $q_i(i)$ be the *last* configuration in $(p, 0) \xrightarrow{\pi_1} (r, x)$ with the counter value i. We shall call these configurations *marked configurations*.

Consider the n^2+n marked configurations $q_{n^2+n+1}(n^2+n+1), q_{n^2+n+2}(n^2+n+2), \ldots, q_{2n^2+2n}(2n^2+2n)$. We will group these marked configurations in n blocks, each consisting of $n+1$ marked configurations.

- Block 1: $q_{n^2+n+1}(n^2+n+1), q_{n^2+n+2}(n^2+n+2), \ldots, q_{n^2+2n+1}(n^2+2n+1)$.
- Block 2: $q_{n^2+2n+2}(n^2+2n+2), q_{n^2+2n+3}(n^2+2n+3), \ldots, q_{n^2+3n+2}(n^2+3n+2)$.
- $\cdots$
- Block n: $q_{2n^2+n}(2n^2+n), q_{2n^2+n+1}(2n^2+n+1), \ldots, q_{2n^2+2n}(2n^2+2n)$.

Now, using pigeonhole principle, observe that a cycle can be found in every block. Since there are $n+1$ marked configurations in a given block, there must be two marked configurations $q_i(i)$ and $q_j(j)$ with the same state $q_i = q_j$. Let $q = q_i = q_j$. Accordingly, the run from $q(i)$ to $q(j)$ is a cycle that adds $j - i$ to the counter. Importantly, observe that $1 \leq j - i \leq n$. This means that we can obtain n cycles with positive effects $1 \leq a_1, \ldots, a_n \leq n$, respectively.

We can symmetrically make the same arguments for the suffix $(r, x) \xrightarrow{\pi_2} (q, 0)$ to obtain n cycles with negative effects $-n \leq b_1, \ldots, b_n \leq -1$. For this, we define $q'_i(i)$ to be the *first* configuration in $(r, x) \xrightarrow{\pi_2} (q, 0)$ that has counter value i. Now, by using Lemma 2.1, we can find a non-empty collection of positive cycles and non-empty collection of negative cycles that overall have zero counter effect. Specifically, we use Lemma 2.1 with $m = n$, $A = \{a_1, \ldots, a_n\}$, and $B = \{-b_1, \ldots, -b_n\}$.

The idea is now to construct a new run $(p, 0) \xrightarrow{\pi'} (q, 0)$ where π' is obtained by removing the non-empty collections of positive and negative effect cycles from π. It is important to not that the positive effect cycles are only removed after the configuration $q_{n^2+n}(n^2+n)$ in the first half of the run $(p, 0) \xrightarrow{\pi_1} (r, x)$ and negative effect cycles are only removed before the configuration $q'_{n^2+n}(n^2+n)$ in the second half of the run $(r, x) \xrightarrow{\pi_1} (q, 0)$. This means that the prefix of π' until $q_{n^2+n}(n^2+n)$ and the suffix from $q'_{n^2+n}(n^2+n)$ are both the same as in π. We therefore only need to verify that the counter does not drop below zero in the middle part of the run between $q_{n^2+n}(n^2+n)$ and $q'_{n^2+n}(n^2+n)$.

Recall that $q_{n^2+n}(n^2+n)$ is the *last* configuration with the counter value n^2+n, so all configurations between $q_{n^2+n}(n^2+n)$ and (r,x) have counter value at least n^2+n. Let a' be the total effect of the removed positive cycles; $a' \leq a_1 + \ldots + a_n \leq n^2$. This means the counter is always nonnegative in modified run $(p,0) \xrightarrow{\pi'_1} (r, x-a')$.

Lastly, it is true that by removing negative effect cycles can only increase the counter so it only remains to check that before $q'_{n^2+n}(n^2+n)$, the new run is nonnegative. Indeed, by definition of $q'_{n^2+n}(n^2+n)$, we know that all configurations in the run from (r,x) to $q'_{n^2+n}(n^2+n)$ will have counter value at least n^2+n. Thus, by the same argument as before, after the removal of positive cycles in the first half of the run, the counter will still be above zero.

Now, we have obtained a run $(p,0) \xrightarrow{\pi'} (q,0)$ which observes a maximum counter value of $x-a'$. Since $x - a' < x$, this contradicts the choice of $(p,0) \xrightarrow{\pi} (q,0)$ being the run from $(p,0)$ to $(q,0)$ that had the least greatest counter value.

We conclude this proof by stating that of course there exists such a polynomial f; $f(n) := 2n^2+2n$. □

Recall also from Lecture 1.

Theorem 1.9. Reachability in 1-CMs is in NL.

Proof sketch. The following is a corollary of Lemma 1.10 and the proof of Theorem 1.8.

Let $(M,(p,0),(q,0))$ be a given arbitrary instance of reachability in a 1-CM M, with an initial configuration $(p,0)$, and a target configuration $(q,0)$. By Lemma 1.10, we need not consider configurations with a counter value greater than $f(n)$; thus there are $f(n)+1$ many possible counter values $0,1,2,\ldots,f(n)$. This means that we can design a non-deterministic algorithm that explores the space of $f(n)$-bounded configurations; there are $n(f(n)+1)$ many such configurations. This also means that if there is a run from $(p,0)$ to $(q,0)$ of length at most $n(f(n)+1)$ because one need not repeat configurations in a run. So the same algorithm as in the proof of Theorem 1.8 can be used. We only need $\log(n) + \log(f(n)+1) = \mathcal{O}(\log(n))$ space to store the current configuration and we only need $\log(n) + \log(f(n)+1) = \mathcal{O}(\log(n))$ space to store the variable ℓ that tracks the length of the run. □

Definition 2.2. A *d-dimensional vector addition system (d-VAS)* is a set of vectors

$$V = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\} \subseteq \mathbb{Z}^d.$$

One should think that a d-VAS is an automaton with d many nonnegative integer counters and which *does not* have any states. A configuration of a d-VAS V is just a vector $\mathbf{c} \in \mathbb{N}^d$. A VAS V can step from configuration $\mathbf{c}$ to configuration $\mathbf{c'}$ if $\mathbf{c'} - \mathbf{c} \in V$.

Definition 2.3. A *d-dimensional vector addition system with states (d-VASS)* consits of a finite set of states Q and a set of transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$. A configuration of a d-VASS (Q,T) is a pair containing the current state $q \in Q$ and the current counter values $\mathbf{v} \in \mathbb{N}^d$; denoted $(q,\mathbf{v})$ or $q(\mathbf{v})$. A VASS (Q,T) can transition from $p(\mathbf{u})$ to $q(\mathbf{v})$ if $(p, \mathbf{v}-\mathbf{u}, q) \in T$.

Before talking about the encoding of the encoding CMs, VAS, and VASS. We shall compare VASS to pushdown automata as language recognisers. We can define *letter-labelled d-VASS* (Σ, Q, T, q_0, F) where Σ is finite alphabet, Q is a finite set of states, $T \subseteq Q \times (\Sigma \cup \varepsilon) \times \mathbb{Z}^d \times Q$ are the transitions, q_0 is the starting state, and F is the set of final states. The only difference is that every transition of the VASS is labelled by a letter $\sigma \in \Sigma$ (or ε). The acceptance condition is just whether a final state $f \in F$ is reached (this is called "the coverability acceptance condition"). One can modify the acceptance condition so that you must reach a final state $f \in F$ with exactly zero on every counter value (this is called "the reachability acceptance condition").

Lemma 2.4. Fix a finite alphabet Σ. Let V be a letter-labelled 1-VASS. There exists a pushdown automaton P that recognises the same language as V.

Proof sketch. Just use a pushdown automaton with a unitary stack alphabet $\Gamma = \{a\}$. Whenever the counter of V is incremented, an 'a' is pushed onto the stack and whenever the counter of V is decremented, an 'a' is popped off the stack. $\square$

Lemma 2.5. Fix a finite alphabet Σ. There exists a pushdown automaton P such that there does not exist a letter-labelled 1-VASS V that recognises the same language as P.

Proof idea. Consider the pushdown automaton P that recognises the language D_2 (the second-order Dyck language). $D_2 \subseteq \{\texttt{[}, \texttt{]}, \texttt{(}, \texttt{)}\}$ is the language of balanced parentheses of two different types, square brackets `[ ]` and round brackets `( )`. The pushdown automata just puts opening brackets on the stack to maintain the current bracket nesting depth and whenever it sees a closed bracket, it check whether the top of the stack has a matching opening bracket and then pops that open bracket off the top of the stack and continues.

To see that there does not exist a letter-labelled 1-VASS V that recognises D_2, then consider the fact that the word $\texttt{[}^x \, \texttt{(}^y \, \texttt{)}^y \, \texttt{]}^x \in D_2$. Roughly speaking, the 1-VASS will use the counter value to store the depth of the open brackets, but it will not be able to distinguish between the two different types of brackets. This means that V can "cheat" and read a word of the form $\texttt{[}^x \, \texttt{(}^y \, \texttt{)}^z$ where $z > y$ and thus $\texttt{[}^x \, \texttt{(}^y \, \texttt{)}^z \notin D_2$. $\square$

Lemma 2.6. Fix a finite alphabet Σ. There exists a letter-labelled 2-VASS V such that there does not exist pushdown automaton P that recognises the same language as V.

Proof idea. Let $\Sigma = \{a, b, c\}$. Consider a letter-labelled 2-VASS that recognises the language $L = \{a^x b^y c^z : z \leq \min(x, y)\}$. Such a letter-labelled 2-VASS can consist of three states, the first state has a cycle reading 'a's with the effect $(1, 0)$, the second state has a cycle reading 'b's with effect $(0, 1)$, and the third state has cycle reading 'c's with effect $(-1, -1)$. For now, it is left as a challenge to prove that not exist a language-equivalent pushdown automaton. One can use the pumping lemma for pushdown automata while considering the words $a^x b^{2x} c^x$, $a^{2x} b^x c^x$, and $a^{2x} b^{2x} c^{2x}$. $\square$

A corollary of Lemma 2.5 and Lemma 2.6 is that, as language recognising automata, it is not true that VASS are "more powerful" than pushdown automata, nor is it true that pushdown automata are "more powerful" than VASS.