Infinite Automata 2025/26

Lecture Notes 8

Henry Sinclair-Banks

Definition 8.1. A *bounded 1-VASS* is a tuple (Q, T, B) where (Q, T) is a 1-VASS and $B \in \mathbb{N}$ is the bound. A configuration in a bounded 1-VASS is a pair (q, x) where $q \in Q$ is the current state and $x \in [0, B]$ is the current counter value. This means that a transition $(p, u, q) \in T$ can be taken from a configuration (p, x) if $x + u \geq 0$ and $x + u \leq B$.

Theorem 8.2. Reachability in binary-encoded bounded 1-VASS is PSPACE-complete.

To get some understanding for why reachability in *bounded* 1-VASS can be PSPACE-hard while reachability in (unbounded) 1-VASS is in NP (Theorem 3.2), we will see that with the bound, one is able to simulate zero tests, equality tests, and resets. First, for zero tests, consider the following pair of transition (p, B, q) and $(q, -B, p')$; the only possible way to traverse from p to p' is from the configuration $(p, 0)$ without exceeding the bound B at state q. Second, for equality tests in general, suppose we would like to test whether the counter is equal to some value $z \in [0, B]$. Consider the following triplet of transitions $(p, -z, q)$, $(q, +B, r)$, and $(r, -B + z, p')$; the second transition can only be taken from the configuration $(q, 0)$, so the whole gadget can only be started from the configuration (p, z), as required. Third, for resets, consider the following two-state three-transition gadget: $(p, -1, p)$, $(p, +B, q)$, and $(q, -B, p)$. Like before, the transition from p to q can only taken from the configuration $(p, 0)$, this means that the loop at p with effect -1 is used to reset the counter value to 0.

Definition 8.3. A d-dimensional *counter-stack automaton* is a tuple (Q, T) where Q is a finite set of states and T is set of transitions. Each transition $t \in T$ is a 5-tuple $t = (q, E, \mathbf{u}, R, q')$ where:

- $q, q' \in Q$ are states;
- E is a set of *equality tests* represented as a partial function $E : [1, d] \to \mathbb{N}$ with the requirement that if $E(i)$ is defined, then for all $j > i$, $E(j)$ is also defined;
- $\mathbf{u} \in \mathbb{N}^d$ is the additive update to each of the d counters (note that all updates are nonnegative); and
- $R \subseteq \{1, \ldots, d\}$ is the set of indices of counters that will be *reset* with the restriction that if $R(i)$ is defined, then $E(i)$ must be defined.

A configuration of a d-dimensional counter-stack automaton is a pair $(q, \mathbf{x})$ where $q \in Q$ is a state and $\mathbf{x} \in \mathbb{N}^d$ are the current counter values. A configuration $(p, \mathbf{x})$ can transition to a configuration $(q, \mathbf{y})$ if there exists a transition $(p, E, \mathbf{u}, R, q) \in T$ such that

(i) for every $i \in \{1, \ldots, d\}$ such that $E(i)$ is defined, we have $\mathbf{x}[i] = E(i)$;

(ii) for every $i \in R$, we have $\mathbf{y}[i] = 0$; and

(iii) for every $i \notin R$, we have $\mathbf{x}[i] + \mathbf{u}[i] = \mathbf{y}[i]$.

Furthermore, we call a d-dimensional counter-stack automaton A is *b-safe* ($b \in \mathbb{N}$), from a configuration $(p, \mathbf{x})$ if it is impossible for the automaton to increase any of the d counters beyond b starting from the given configuration $(p, \mathbf{x})$. Formally, the requirement is

$$\{(q, \mathbf{y}) \in Q \times \mathbb{N}^d : (p, \mathbf{x}) \xrightarrow{*}_S (q, \mathbf{y})\} \subseteq Q \times [0, b]^d.$$

We say that a counter-stack automaton is *safe* if it is b-safe, for some $b \in \mathbb{N}$.

Definition 8.4. Reachability in safe counter-stack automata (problem).

Input. A counter-stack automaton $S = (Q, T)$ that is *promised* to be safe, an initial configuration $(p, \mathbf{x})$ and a target configuration $(q, \mathbf{y})$.

Question. Does there exist a run $(p, \mathbf{x}) \xrightarrow{*}_s (q, \mathbf{y})$?

We will now argue that, intuitively, why bounded 1-VASS can simulate safe counter-stack automata. Suppose we have a bounded 1-VASS $V = (Q, T, 15)$ and let c denote the one counter of V. We can store, inside of the one counter of V, two 2-bit counters c_1 and c_2 inside of c and perform some operations on them. Consider the encoding $c = 2^2 \cdot c_2 + c_1$; in other words, the higher two bits of c store c_2 an the lower two bits of c store c_1 (hence c is bounded by $15 = 2^3 + 2^2 + 2^1 + 2^0$). We can easily add to both counters c_1 and c_2; the operation $c_1 += 1$ corresponds to $c += 1$ and the operation $c_2 += 1$ corresponds to $c += 4$.

Performing equality tests is more challenging. Suppose we want to test that $c_2 = 2$. The possible values for c (with $c_2 = 2$) are $2^3 + 0 = 8$, $2^3 + 1 = 9$, $2^3 + 2 = 10$, or $2^3 + 3 = 11$. Testing $8 \leq c \leq 11$ is possible with a bounded 1-VASS: first subtract and add 8 to test $c \geq 8$ and then add and subtract $B - 11 = 4$ to test $c \leq 11$. However, now suppose that we want to test that $c_1 = 1$. The possible values for c (with $c_1 = 1$) are $0 + 2^0 = 1$, $4 + 2^0 = 5$, $8 + 2^0 = 9$, and $12 + 2^0 = 13$. Testing whether $c \in \{1, 5, 9, 13\}$ is not possible without enumerating each test and if there were a greater number of counters $c_1, c_2, \ldots, c_d$ inside of c, this leads to an exponential blow up. Instead, we shall accordingly enforce the restriction that if we wish to test $c_1 = 1$, we must also know (or test) the value of c_2. For example, if $c_1 = 1$ and $c_2 = 2$, then we just need test $c = 9$.

With the above ideas, it is possible to prove that binary-encoded 1-VASS can simulate safe counter-stack automata. Note that the bound roughly corresponds to 2 to the power of the dimension of the counter-stack automata.

Lemma 8.5. *Fearnley and Jurdziński 2013.* Reachability in safe counter-stack automata is polynomial-time reducible to reachability in binary-encoded bounded 1-VASS

Definition 8.6. *Subset-sum Game.* A subset sum game is played between two players: the *universal player* and the *existential player*. The game itself is specified by a pair (ψ, T) where $T \in \mathbb{N}$ is the *target value* and ψ defines the choices for the players:

$$\forall\{A_1, B_1\}\ \exists\{E_1, F_1\}\ \cdots\ \forall\{A_n, B_n\}\ \exists\{E_n, F_n\}.$$

Here $A_1, B_1, E_1, F_1, \ldots, A_n, B_n, E_n, F_n \in \mathbb{N}$. The game consists of n rounds and each round consists of two turns, one for the universal player and one for the existential player. In the i-th round, the universal player takes a turn to first choose whether to add A_i or B_i to the current sum, and then the existential player takes a turn to reacts and choose to add E_i or F_i to the current sum. It is important to note that the choices made in rounds $1, \ldots, i-1$ can affect the decision of the players choices in the i-th round. A play is *winning for the existential player* if, after all $2n$ turns, the sum of all chosen values A_i or B_i and E_i or F_i sums to T, otherwise the play is *winning for the universal player*.

The subset-sum game decision problem takes as input the pair (ψ, T) and asks whether the existential player has a winning strategy. In other words, this problem is to decide, no matter what decisions are made by the universal player, does the existential player have a strategy (a procedure for making decisions during the game) to always win the game?

The subset-sum game problem trivially belongs to PSPACE. There are two ways to see this: the first is that it can solved by a polynomial time alternating Turing machine, and the second is that one can imagine iterating through all possible plays in the game and the current configuration only needs to track a linear stack of current decisions and the current sum (which cannot be large). The quantified version of subset-sum (which has the same spirit as QSAT) is a PSPACE-hard problem and solving a subset-sum game basically amounts to deciding a restricted instance of quantified subset-sum. The hardness of quantified subset-sum can be used to prove the following theorem.

Theorem 8.7. Deciding the winner of a subset-sum game is PSPACE-complete.

Now, in order to prove Theorem 8.2, we will use Lemma 8.5 and the following lemma.

Lemma 8.8. Reachability in safe counter-stack automata is PSPACE-hard.

Proof sketch. We will reduce from the subset-sum game decision problem. However, for the sake of simplicity, we will work on the following arbitrary *two-round* instance of the subset-sum game to demonstrate the key ideas behind this proof.

$$(\psi = \forall\{A_1, B_1\}\ \exists\{E_1, F_1\}\ \forall\{A_2, B_2\}\ \exists\{E_2, F_2\},\ T)$$

In order to simulate this game with a safe counter-stack automata, we will attempt to evaluate every possible play (from the perspective of the existential player). We would like to know, for either decision A_1 or B_1 of the universal player, whether there is a choice between E_1 and F_1 such that, for either decision A_2 or B_2 of the universal player, whether there is a decision between E_2 and F_2 such that the sum of all chosen values is always equal to T. For this, we would like to evaluate the following:

Play number	u_1	e_1	u_2	e_2	Sum
1	A_1	E_1 or F_1	A_2	E_2 or F_2	$= T$?
2	A_1	*Same as play 1*	B_2	E_2 or F_2	$= T$?
3	B_1	E_1 or F_1	A_2	E_2 or F_2	$= T$?
4	B_1	*Same as play 3*	B_2	E_2 or F_2	$= T$?

If all plays indeed evaluate to have sum T, then we know that the subset-sum game (ψ, T) is winning for the existential player.

End of lecture, to be continued.