Task (5. **Distance automata with more counters**). *Consider the following extension of a distance automaton. Instead of having a set of costly transitions, we have two counters* $\{1, 2\}$ *and each transition is labelled by an instruction from the following toolkit:*

- *do nothing;*
- *increment counter 1;*
- *reset counter 1;*
- *reset counter 1 and increment counter 2;*
- *reset both counters;*

The value of a run is the biggest value attained by any counter. Prove that limitedness is decidable for these automata, using the limitedness game.

We will base our proof on the proof for the standard distance automata. Let us recall that we needed to show equivalence of three conditions:

1. the automaton is limited;
2. there is some $m \in \{1, 2, \ldots\}$ such that the player Automaton wins the limitedness game with bound m;
3. player Automaton wins the limitedness game with bound $m = \omega$;

for an appropriately modified limitedness game. The Automaton player wins iff, after their every move:

1. at least one run imposed by the Automaton's play is accepting; and
2. no run has value exceeding m if $m \in \mathbb{N}$; if $m = \omega$ then this condition is instead that no run increases a counter infinitely many times while resetting it finitely many times.

The change to condition 2 for $m = \omega$ is needed to make it easily ω-regular.

As in the standard limitedness problem, implications 2 $\implies$ 1 and 2 $\implies$ 3 are obvious. We will show the two remaining implications.

Lemma 2. 1 $\implies$ 2

Proof. We will show that the automaton has a winning strategy if it is limited. Fix m as the limit of the automaton. Let us recall that for the standard limitedness game we have defined a notion of an *optimal run* as one with minimal cost among all runs. Then the Automaton player wins by playing a cost-optimal forest of runs.

We can easily extend this notion to an automaton with one resettable counter. We define the cost of a run as a function C from runs into $\mathbb{N} \cup \{\infty\}$ inductively.

- An empty run has cost 0.
- Let ρ be a run with cost k and t be a transition. Then we define cost of the run ρt as:

$$C(\rho t) = \begin{cases} k & \text{if } k = \infty \vee t \text{ is of type "do nothing"} \\ k+1 & \text{if } k < m \wedge t \text{ is of type "increment"} \\ \infty & \text{if } k = m \wedge t \text{ is of type "increment"} \\ 0 & \text{if } k < \infty \wedge t \text{ is of type "reset"} \end{cases}$$

This function has a few useful properties. First of all, it is monotone, i.e. for any runs ρ and ρ' and a transition t we have $C(\rho) \leq C(\rho') \implies C(\rho t) \leq C(\rho' t)$. Second of all, every run with value limited by m has a finite cost, and every run with a finite cost is limited by m. It can be shown that when the Automaton player constructs a forest of cost-optimal runs with respect to a function with these properties, then they win the limitedness game with bound m.

We will construct a similar cost function for our two-counter case, however, we need to loosen the second property a bit. We only require that every run with a finite cost be limited by $(m+1)^2$. First, let's show that such a function is sufficient.

Claim 1. *In the two-counter version of the limitedness problem, if there exists a cost function C from runs into $\mathbb{N} \cup \{\infty\}$ such that*

1. *it's monotone,*
2. *every run limited by m has finite score,*
3. *every run that has finite score is limited by $(m+1)^2$*

then the Automaton player wins the limitedness game with bound $(m+1)^2$ by choosing his sets of transitions to form a forest of cost-optimal runs with respect to C.

Proof. First, optimality of a run is local: if ρ is an optimal run, it means that for any ρ' we have $C(\rho) \leq C(\rho')$. Therefore, by monotonicity of C, after adding another transition t, we have $C(\rho t) \leq C(\rho' t)$. This shows that the strategy is at all possible.

The win condition 1 is satisfied, since an accepting run must exist by the automaton's limitedness, and property 2 guarantees that there is some optimal run. Conversely, optimal runs have finite score, thus win condition 2 is also satisfied for the bound $(m+1)^2$, by property 3. ■

Now we need to construct such a function. The idea is to keep the state of the counters as a number in base $m+1$. However, we cannot simply set the value to ∞ when any of them overflows, as it can be shown that such a function is not monotone. Therefore, we loosen the requirement for the bound to $(m+1)^2$ and say that when counter 1 overflows, it increments 2 once. Then we go to ∞ only when the counter 2 overflows. More formally, an empty run has value 0 and for a run ρ, transition t, when $C(\rho) = k = k_1 + k_2 \cdot (m+1)$ we have

$$C(\rho t) = \begin{cases} k & \text{if } k = \infty \vee t \text{ is of type "do nothing"} \\ k+1 & \text{if } k_1 < m \wedge t \text{ is of type "increment 1"} \\ (k_2+1)\cdot(m+1) & \text{if } k_1 = m \wedge k_2 < m \wedge t \text{ is of type "increment 1"} \\ \infty & \text{if } k_1 = m \wedge k_2 = m \wedge t \text{ is of type "increment 1"} \\ k_2 \cdot (m+1) & \text{if } k < \infty \wedge t \text{ is of type "reset 1"} \\ (k_2+1)\cdot(m+1) & \text{if } k_2 < m \wedge t \text{ is of type "reset 1, increment 2"} \\ \infty & \text{if } k_2 = m \wedge t \text{ is of type "reset 1, increment 2"} \\ 0 & \text{if } k < \infty \wedge t \text{ is of type "reset both"} \end{cases}$$

It remains to show the three properties from Claim 1. For monotonicity, the interesting cases are increments. Let's take any two runs ρ, ρ' and assume $C(\rho) = k_1 + k_2 \cdot (m+1) \leq C(\rho') = k_1' + k_2' \cdot (m+1)$. For a transition t of type "increment 1", if $C(\rho t) = \infty$ then clearly $C(\rho' t) = \infty$. Otherwise, $C(\rho t) = C(\rho) + 1 \leq C(\rho') + 1 \leq C(\rho' t)$, so monotonicity is preserved. For a transition t of type "reset 1, increment 2" again, if the first run overflows to ∞ then the second also must, otherwise we get $C(\rho t) = (k_2+1)\cdot(m+1) \leq (k_2'+1)\cdot(m+1) \leq C(\rho' t)$.

For a run limited by m we have no cases of incrementing 1 when $k_1 = m$, so it's easy to see that the values k_1, k_2 correspond to the values of the counters since their last reset. Therefore we get property 2.

To get property 3, assume that some counter exceeds $(m+1)^2$. Clearly if we do that with counter 2 we get a cost of infinity. If we do that with counter 1, we see that there will be $m+1$ "carry-overs" to the value k_2. Because we cannot reset the counter 2 without resetting the counter 1, we must get infinity as the cost when k_2 overflows. □

It remains to show $3 \implies 2$. We should first remark that the winning condition for $m = \omega$ is ω-regular. We can construct a nondeterministic Büchi automaton that first guesses the counter that will be incremented infinitely many times and then the moment at which the last reset occurs. So the win condition is the complement of that, and ω-regular languages are closed under complement. Of course we still need to check if the chosen transitions are legal and that there exists an accepting run, but that is trivial to simulate knowing the distance automaton. Having that, we can show the last implication:

Lemma 3. $3 \implies 2$

Proof. We will show that if the Automaton player has a winning strategy when $m = \omega$, then they have a winning strategy for some finite m. The argument is very similar to that for the standard case. By the Büchi-Landweber theorem, since the win condition is ω-regular, there exists a finite memory strategy for the Automaton player for the game with $m = \omega$. Assume the Automaton player using this strategy loses for a bound m equal to the number of states of the distance automaton times the number of states of the finite deterministic automaton recognising the strategy. Suppose the player Input plays $a_1, a_2, \ldots$ and the player Automaton plays $T_1, T_2, \ldots$, and the bound is violated. This means that for some $n \geq m$ there exists a run ρ using the transitions from the Automaton's play $T_1, T_2, \ldots, T_n$, that has value at least m. By the pigeon hole principle, there exist two indices $i, j, i < j$ such that the distance automaton is in the same state at positions i, j in ρ, and the player Automaton is in the same state of their strategy automaton at positions i, j in the game, and at least one of the counters is incremented without resetting between positions i, j in ρ. Therefore, the player Input can play $a_1 a_2 \ldots a_{i-1}(a_i \ldots a_j)^\omega$ and win the game with $m = \omega$. This contradicts the assumption of $3 \implies 2$. □

Task (6. Separation). *Prove that the following problem is decidable:*

- *Input: Regular word languages $L, K \subseteq \Sigma^*$, given say by deterministic automata.*
- *Question: Is there a language of star height 1 which contains L but is disjoint with K?*

A language of star height 1 is a language which can be defined by a regular expression, without complement, where the Kleene star is allowed, but it cannot be nested. As a hint, use the previous exercise.

We're looking for any language L', such that it has star height 1 and $L \subseteq L' \subseteq K^{\complement}$. We assume $L \neq \emptyset$, because then the answer is trivially "YES" for $L' = \emptyset$ and emptiness is decidable for regular languages.

Fact 1. *Any star height 1 language can be defined by a regular expression that has no "concatenation-over-union" constructs, i.e. those of form $e_1(e_2 + \ldots + e_n)$ or $(e_1 + \ldots + e_{n-1})e_n$.*

Proof. Obviously we can turn such expressions into $e_1e_2 + \ldots + e_1e_n$ and $e_1e_n + \ldots + e_{n-1}e_n$, respectively. □

One can observe that this limits us to expressions of form

$$e_1(f_{1,1} + \ldots + f_{1,k_1})^* e_2(f_{2,1} + \ldots + f_{2,k_2})^* \ldots e_n(f_{n,1} + \ldots + f_{n,k_n})^*$$

We'll call such expressions *simple* or *simplified* regular expressions.

The idea is to construct a distance automaton that will try to guess the form of this simplified expression for L'. The less important counter 1 will keep the length of the current e_i or $f_{i,j}$ fragment, while the counter 2 will keep track of the number of fragments – n.

Limitedness requires the automaton to be universal. To ensure that we only go through the trouble of constructing an expression for words in L, we will use a finite automaton recognising $L^{\complement}$ (finding such is obviously decidable). All of its transitions will be of type "do nothing", thus ensuring that every word in $L^{\complement}$ will have a run of cost 0.

The more complicated part is ensuring $L' \subseteq K^{\complement}$. As we will prove later, a limited run over a word $w \in L$ will be equivalent to producing a simplified expression e such that $w \in L(e)$. We must also ensure that $L(e) \cap K = \emptyset$. To do that, our distance automaton will be constructing a transition function over the DFA recognising K as limited to words accepted by e. Assume $\mathcal{A} = \{Q, q_0, \Sigma, \delta, F\}$ is a DFA recognising K (finding such is obviously decidable). A transition function over it will be a function $f : \mathcal{P}(Q) \to \mathcal{P}(Q)$ that maps any set of states of $\mathcal{A}$ into the set of possible final states of runs of $\mathcal{A}$ over words in e. It follows from that description, that if $F \cap f(\{q_0\}) \neq \emptyset$, then there exists a word in $L(e) \cap K$ and our expression e cannot be a part of a union defining the language L', or it would violate the $L' \subseteq K^{\complement}$ requirement. To construct such a function during the distance automaton's run, we will require two operators over functions in $\mathcal{F}_{\mathcal{A}} = \mathcal{P}(Q) \to \mathcal{P}(Q)$, dependant on the automaton $\mathcal{A}$:

$$\begin{aligned}
\Phi_{\mathcal{A}} &: \Sigma \to \mathcal{F}_{\mathcal{A}} \\
\Phi_{\mathcal{A}}(a) &= \lambda X.\{\delta(x, a) \mid x \in X\} \\
\Psi_{\mathcal{A}} &: \mathcal{F}_{\mathcal{A}} \to \mathcal{F}_{\mathcal{A}} \\
\Psi_{\mathcal{A}}(F) &= \text{fix } F \\
\Upsilon_{\mathcal{A}} &: \mathcal{F}_{\mathcal{A}} \times \mathcal{F}_{\mathcal{A}} \to \mathcal{F}_{\mathcal{A}} \\
\Upsilon_{\mathcal{A}}(F, G) &= \lambda X.F(X) \cup G(X) \\
\Pi_{\mathcal{A}} &: \mathcal{F}_{\mathcal{A}} \times \mathcal{F}_{\mathcal{A}} \to \mathcal{F}_{\mathcal{A}} \\
\Pi_{\mathcal{A}}(F, G) &= G \circ F
\end{aligned}$$

Where fix means the least fixed-point of the function. In other words, since F defines runs over $\mathcal{A}$, operator $\Phi(a)_{\mathcal{A}}$ constructs runs of length 1 over a single-letter word a; $\Psi(F)_{\mathcal{A}}$ closes the runs over concatenation, analogous to an application of the Kleene star; $\Upsilon_{\mathcal{A}}(F, G)$ produces a union of runs from F and G; and $\Pi_{\mathcal{A}}(F, G)$ produces a concatenation of runs from F with runs from G. We're not going to prove the semantics of these operators formally – they are clear from the properties of DFAs.

Because $\mathcal{A}$ is a finite automaton, the domain $\mathcal{F}_{\mathcal{A}}$ is also finite, thus these operators are not only computable, but computable in finite memory for a fixed automaton $\mathcal{A}$. We can therefore assume that our distance automaton can use these operators to determine its next state.

During its run, the distance automaton must keep in its memory where in the expression being constructed it is and a fixed number of transition functions over $\mathcal{A}$ – call these functions $Word, Union, Full \in \mathcal{F}_{\mathcal{A}}$. Initially set all to $id \in \mathcal{F}_{\mathcal{A}}$. The possible actions of the automaton are:

1. If in a fragment e_i, end the fragment and begin a new fragment f_i, by beginning its first word $f_{i,1}$. This is a "reset 1" operation. Set $\mathit{Full} := \Pi(\mathit{Full}, \mathit{Word})$, $\mathit{Union} := \mathit{id}$, $\mathit{Word} := \mathit{id}$.

2. If in a word $f_{i,j}$, end the word and begin a new word $f_{i,j+1}$ of the current fragment f_i. This is a "reset 1" operation. Set $\mathit{Union} := \Upsilon(\mathit{Union}, \mathit{Word})$, $\mathit{Word} := \mathit{id}$.

3. If in a word $f_{i,j}$, end the current fragment f_i and begin a new fragment e_{i+1}. This is a "reset 1, increment 2" operation. Set $\mathit{Full} := \Pi(\mathit{Full}, \Psi(\mathit{Union}))$, $\mathit{Union} := \mathit{id}$, $\mathit{Word} := \mathit{id}$.

4. If there are no fragments, or we are in a word $f_{i,j}$ end the entire expression. This is a "do nothing" operation. Set $\mathit{Full} := \Pi(\mathit{Full}, \Psi(\mathit{Union}))$ and advance to a special end state.

5. If outside of the special end state, read the next letter of the input word, say $a \in \Sigma$, and extend the current word. This is an "increment 1" operation. Set $\mathit{Word} = \Pi(\mathit{Word}, \Phi(a))$.

The automaton accepts in the special end state if and only if $F \cap \mathit{Full}(\{q_0\}) = \emptyset$. A run of the distance automaton over a word $w \in \Sigma^*$ induces an expression consisting of a number of fragments e_i and f_i. We're not going to formally define how this expression is induced, but it's obvious from the above description of the automaton's semantics. For the corner case of the number of segments being 0, assume $L(e) = \{\epsilon\}$ (recall we have assumed $L \neq \emptyset$, so this is w.l.o.g.). It's also trivial to see that w must belong to the language of the induced expression. Of course, for runs that are not limited, this expression need not be simplified or even regular, as the fragments might be of infinite length. However, for limited runs we get correct expressions, by the following lemma:

Lemma 4. *If a run of the distance automaton is limited by $m \in \mathbb{N}$, then it induces a simplified expression $e_1(f_{1,1} + \ldots + f_{1,k_1})^* \ldots e_n(f_{n,1} + \ldots + f_{n,k_n})^*$ such that*

- *$n \leq m$; and*
- *$\forall_{i \leq n} |e_i| \leq m \wedge \forall_{j \leq k_i} |f_{i,j}| \leq m$.*
- *$\forall_{i \leq n} k_i < \infty$; and*

Proof.

- The number of fragments, n, depends on the number of times action 2 fires. Since it always increments the counter 2 and there is no operation resetting counter 2, from limitedness of the run we get $n \leq m$.
- We need to consider what can happen between any two "end word" actions, which are actions 1-4. The only remaining action is 5, which always increments the counter 1. The length of any word in the expression depends on the number of these actions, so from limitedness their length must be $\leq m$.
- From the previous bullet, for a fragment f_i the length of each word $f_{i,j}, j \leq k_i$ is limited by m. Since Σ is finite, there can only be $\sum_{1 \leq x \leq m} |\Sigma|^x$ such words, which is a finite number.

□

This will be enough to prove $L \subseteq L'$. To prove that we preserve $L' \subseteq K^{\complement}$ we need an invariant that should be intuitively visible by looking at the construction of the distance automaton:

Lemma 5. *Consider a prefix of a run of the distance automaton ending in action 1, 3 or 4. Let e be the expression induced by this prefix. For any two states $p, q \in Q$ we have $q \in \mathit{Full}(p)$ if and only if there exists a word $w \in L(e)$, such that there exists a run in the automaton $\mathcal{A}$ over the word w starting in p and ending in q.*

Proof. For a prefix of length 1 the property is obvious, as $L(e) = \{\epsilon\}$ and the only possible action is 4. $\mathit{Full} = \Pi(\mathit{id}, \Psi(\mathit{id})) = \mathit{id}$, so Full describes runs over the empty word in $\mathcal{A}$.

Assume the property holds for prefixes of lengths up to n. Consider a prefix of length $n+1$. There are two cases.

- The final action was of type 1. Then the expression is of form $e \circ e_i, e_i = a_1 \ldots a_k$ and the only possible actions performed after we finished the previous fragments was action 5. We know that the lemma held when we began the e_i fragment. The Word function is then a composition of functions $\Phi(a_1), \ldots, \Phi(a_k)$. It's clear that this function represents possible runs over e_i in $\mathcal{A}$. By the induction hypothesis, Full represents possible runs over words in $L(e)$. Then the action 1 sets Full to the runs over concatenations of such words with e_i with the $\Pi_{\mathcal{A}}$ operator.

- The final action was of type 3 or 4. Then the expression is of form $e(f_{i,1}+\ldots+f_{i,k_i})^*$ and we have performed k_i sequences of actions 5 separated by actions 2. Then $Union = \Upsilon_{\mathcal{A}}(\Upsilon_{\mathcal{A}}(\ldots(\Upsilon_{\mathcal{A}}(id, W_1),\ldots,W_{k_i-1}),W_{k_i})$, where W_j is a function representing possible runs over $f_{i,j}$ in $\mathcal{A}$ (by a similar argument as in the previous bullet). It's clear from the operator $\Upsilon_{\mathcal{A}}$ that $Union$ represents possible runs over a single word selected from $f_{i,1},\ldots,f_{i,k_i}$. Applying the $\Psi_{\mathcal{A}}$ operator yields a function representing possible runs over any concatenation of these words. Then, as in the previous bullet, by the induction hypothesis we see that the operator $\Pi_{\mathcal{A}}$ yields all possible runs over $e(f_{i,1}+\ldots+f_{i,k_i})^*$.

$\square$

These lemmata suffice to prove decidability.

Theorem 1. *The above distance automaton is limited if and only if some suitable language L' exists.*

Proof.
$\Longrightarrow$ Assume the bound is $m \in \mathbb{N}$. Let S be the set of all simplified expressions induced by words in L. By applying Lemma 4 we get $|S| < \infty$ – we have at most m fragments of finite length. The expression $e = \bigcup S$ is a finite regular expression, every word in L is contained in $L(e)$ trivially from the construction, and by Lemma 5 none of the induced expressions admit any words from K, so e also does not. Therefore $L \subseteq L(e) \subseteq K^{\complement}$.

$\Longleftarrow$ To get the equivalence, assume L' exists. By Fact 1 there must exist a suitable union of simplified expressions defining L'. For each word in L' choose a single simplified expression e admitting that word. The distance automaton must have a run inducing that expression, not necessarily an accepting run. That run does not cause any counter to exceed the length of the run, which is $O(|e|)$. If it doesn't end with an action 4, we can extend it so. By Lemma 5, if $F \cap Full(\{q_0\}) \neq \emptyset$, then there exists a word $w \in L(e) \cap K$. But $L(e) \subseteq L'$, thus $L' \cap K \neq \emptyset$, which contradicts the assumption. So all such runs over words in L are accepting and have their cost bounded by some constant. Words outside of L have trivial runs of cost 0. Therefore the automaton is limited. $\square$

Since the limitedness problem is decidable and we have shown the separation problem to be equivalent, it is also decidable. Note that the proof is constructive, since given a bound m one can check all possible unions of simplified expressions bounded by m in the sense of Lemma 4.