19.04

На этой паре мы занимались тем, что ввели определения, обсудили связь между ними и попытались мотивировать изучение эвристик из класса $\mathrm{1DSPACE}'$.

Определение 1. *Машина Тьюринга называется онлайн машиной Тьюринга, если она может двигать (но не обязана) головку на входной ленте только в одном направлении.*

Определение 2. *Класс всех языков, распознаваемых онлайн машиной Тьюринга, использующей не более $f(n)$ памяти, будем обозначать* $\mathrm{1DSPACE}(f)$.

Определение 3. *Класс языков, распознаваемых онлайн машиной Тьюринга, которая может узнать длину входа и использует не более $f(n)$ памяти, будем обозначать* $\mathrm{1DSPACE}'(f)$.

Лемма 1. *Существует язык $L \in \mathrm{1DSPACE}'(\log n)$ такой, что $L \notin \mathrm{1DSPACE}(o(n))$.*

Здесь в качестве L подходит множество всех таких двоичных слов s, что двоичная запись $|s|$ является некоторым префиксом s.

Лемма 2. *Для любой функции f такой, что $f(n) < \frac{n}{2}$ при всех достаточно больших n, и f может быть вычислено с использованием $O(\log f)$ памяти, существует язык L такой, что $L \in \mathrm{DSPACE}(\log f) \cap \mathrm{1DSPACE}'(f)$ и $L \notin \mathrm{1DSPACE}'(o(f))$.*

Тут подходит множество всех f-периодичных строк.

В лемме 2 у нас возникла трудность с вычислением f и на семинаре я предложил просто взять $f = n/4$, но после мы выяснили, что в качестве f подходит любая достаточно разумная функция, просто нужно внимательно следить за определениями. Подробнее в разделе с замечаниями.

Основным результатом была такая теорема.

Теорема 1. *Для любой функции $f = \Omega(\log\log n)$ и языка L, распознаваемого оффлайн машиной Тьюринга M с использованием $f(n)$ памяти и рабочим алфавитом Γ, $L \in \mathrm{1DSPACE}'(f(n) \cdot |\Gamma|^{f(n)})$, если $f(n)$ может быть вычислено с использованием $O(f(n) \cdot |\Gamma|^{f(n)})$ памяти.*

Доказательство этой теоремы очень похоже на сведение двусторонних конечных автоматов к односторонним (обычным), с одной небольшой тонкостью, связанной с зацикливанием. Дело в том, что машина

Тьюринга, даже если она останавливается на любом входе, может зациклиться, если ее запустить из неправильной конфигурации. В автоматах такое тоже бывает, но там об этом можно не думать, поскольку все функции переходов можно вычислить заранее (они же конечные!). Я эту проблему обходил при помощи техники baby-step giant-step (то есть запуска двух симуляций с разными скоростями), но на самом деле ее можно решать как угодно, например, просто добавлением счетчика (но тогда надо внимательно следить за памятью).

Algorithm 1 DPLL

```
procedure DPLL_{A,B}(φ)
    if φ is empty then
        return satisfiable
    if φ contain empty clause then
        return unsatisfiable
    x ← A(φ)
    b ← B(φ, x)
    if DPLL_{A,B}(φ[x = b]) = satisfiable then
        return satisfiable
    return DPLL_{A,B}(φ[x = ¬b])

procedure DPLL_H(φ)
    if φ is empty then
        return satisfiable
    if φ contain empty clause then
        return unsatisfiable
    (x, b) ← H(φ)
    if DPLL_H(φ[x = b]) = satisfiable then
        return satisfiable
    return DPLL_H(φ[x = ¬b])
```

Еще мы определили два вида DPLL (классический с двумя эвристиками, а нужный для наших целей — с одной) и поняли, что они друг от друга в терминах сложности по памяти почти ничем не отличаются.

26.04

Мы всю пару доказывали экспоненциальную нижнюю оценку на DPLL_H с $H \in \mathrm{1DSPACE}'(o(\frac{n}{\log n}))$. Подробности напишу позже.

Общие замечания

Здесь чуть позже появятся замечания, которые могут быть интересны тем, кто был на семинаре.