Лекция 8

Ilya Yaroshevskiy

14 апреля 2021 г.

Содержание

1 Теория вычислимости

Σ — алфавит, $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$. $L \subset \Sigma^*$ — формальный язык.

Определение. L — **разрешимый (рекурсивный)**, если $\exists$ программа P, такая что:

- $x \in L \implies P(x) = 1$
- $x \notin L \implies P(x) = 0$

Примечание. Раньше называли рекурсивным, так как использовали рекурсивные функции. Это определение появилось до языков программирования

Примечание. Множетво разрешимых языков счетно

```
fn p(x: Word) -> Res {
    loop { }
    return 1;
}
```

Примечание. Любой конечный язык является разрешимым

```
fn p(x: Word) -> Res {
    if x == x_1 { // x_1 ∈ L
        return 1;
    }
    if x == x_2 { // x_2 ∈ L
        return 1;
    }
    // ...
}
```

Определение. L — **полуразрешимый (перечислимый, рекурсивно перечислмый)**, если $\exists p$, такая что:

1. $x \in L \implies p(x) = 1$
2. $x \notin L \implies p(x) \neq 1$

Примечание. Множество полуразрешимых языков счетно

Примечание. Разрешимый $\implies$ Полуразрешимый

Примечание. Σ^*, $\mathbb{N}^+$, $Prog$ — будут для нас эквивалентными понятиями, когда будем говорить про формальный язык.

- $\Sigma^* \leftrightarrow Prog$ — те строки, которые не являются программами, будем считать программами, которые ‘зависают‘ на любом входе

- $\Sigma^* \leftrightarrow \mathbb{N}^+$ — занумеруем строки в градуированном лексикографичеком порядке

$$\varepsilon, ‘0‘, ‘1‘, ‘00‘, ‘01‘, ‘10‘, ‘11‘, ‘000‘, ‘001‘$$

Определение. Арифметические операции, `if`, `for`, `while`, вызов функций. p — программа, x — слово — запустить программу p на слове x, запустить программу p на слове x с ограничением на время $TL = t$, и ограничением на память $ML = m$

Определение. L — перечислимый, если $\exists p$ которая на пустом входе выводит любое слово из языка хотя бы один раз. $\forall x \in L\ \exists t(x) \quad P\big|_{TL=t}$ выводит x

Пример.

```
fn zeroes() {
    for i in [0..] {
        let s = '0' * i;
        println!("{}", s)
    }
}
```

Теорема 1.1. L — перечислимый $\Leftrightarrow$ L — полуразрешимый

Доказательство.

($\Rightarrow$) Пусть `listL()` — перечисляет L

```
fn inL(x: Word) -> Res {
    async { listL(); }
    if x.is_printed() {
        return 1;
    }
    return 0;
}
```

($\Leftarrow$) `inL()` — полуразрешиель L. Если напишем `listL()`, то он зависнет. Введем таймер.

```
fn listL() -> Res {
    for t in [0..] {
        for x in sigma[0..t] {
            if inL(x).await(t) {
                println!("{}", x)
            }
        }
    }
}
```

$\square$

Примечание. Кодировать пару $\langle x, y \rangle$ можем

Определение. Универсальный язык:

$$U = \{\langle p, x \rangle \big| \text{программа } p(x) = 1\}$$

Свойство 1. U *— полуразрешим — Тьюринг полный*

Теорема 1.2. U — не разрешим

Доказательство. Допустим существует программа `rust#inU((p, x): (Word, Word))`, которая разрешает U

```
fn q(x: Word) -> Res {
    if inU((x, x)) {
        return 0;
    } else {
        return 1;
    }
}
```

`q` никогда не зависает. Вызовем `q(q)`

- $inU(\langle q, q\rangle) = 1$
  - $q(q) = 0$
  - $\langle q, q\rangle \in U \implies q(q) = 1$
- $inU(\langle q, q\rangle) = 0$
  - $\langle q, q\rangle \notin U \implies q(q) \neq 1$
  - $q(q) = 1$

Получается, что q не возвращает ни 0 ни 1. Противоречие □

Свойство 1. *A, B разрешимы, $A \cup B$ — разрешим*

Свойство 2. *A, B разрешимы, $A \cap B$ — резрешим*

Свойство 3. *A — разрешим, $\overline{A} = \Sigma^* \setminus A$ — разрешим*

Свойство 4. *A, B полуразрешимы, $A \cap B$ — разрешим*

Свойство 5. *A, B полуразрешимы, $A \cup B$ — разрешим*

Доказательство.

```
fn p() -> Res {
    for i in [0..] {
        if inA.await(t) {
            return 1;
        }
        if inB.await(t) {
            return 1;
        }
    }
}
```

□

Теорема 1.3 (Поста)**.** L и $\overline{L}$ оба полуразрешимы $\implies$ L — разрешим

Доказательство.

```
fn inL(x: Word) -> Res {
    for t in [0..] {
        if inL(x).await(t) {
            return 1;
        }
        if inCL(x).await(t) { // CL = \overline{L}
            return 0;
        }
    }
}
```

□

Теорема 1.4. Не существует языка программирования, который поддерживает все три свойства

1. Программа не зависает
2. Любой разрешимый язык, распознается программой на этом языке
3. Функция $\langle p, x \rangle \mapsto p(x)$ вычислима