

Лекция 9

Луя Yaroshevskiy

13 мая 2023 г.

Содержание

1 Теория вычислимости

1

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. Множество разрешимых языков счетно

```
1 fn p(x: Word) -> Res {
2   loop { }
3   return 1;
4 }
```

Примечание. Любой конечный язык является разрешимым

```
1 fn p(x: Word) -> Res {
2   if x == x_1 { // x_1 ∈ L
3     return 1;
4   }
5   if x == x_2 { // x_2 ∈ L
6     return 1;
7   }
8   // ...
9 }
```

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

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

Примечание. Множество полурешимых языков счетно

Примечание. Разрешимый \implies Полурешимый

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 лексикографическом 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. $\forall x \in L \exists t(x) \quad P|_{TL=t}$ выводит x

Пример.

```

1 fn zeroes() {
2   for i in [0..] {
3     let s = '0' * i;
4     println!("{}", s)
5   }
6 }
```

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

Доказательство.

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

```

1 fn inL(x: Word) -> Res {
2   async { listL(); }
3   if x.is_printed() {
4     return 1;
5   }
6   return 0;
7 }
```

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

```

1 fn listL() -> Res {
2   for t in [0..] {
3     for x in sigma[0..t] {
4       if inL(x).await(t) {
5         println!("{}", x)
6       }
7     }
8   }
9 }
```

□

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

Определение. Универсальный язык:

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

Свойство 1. U — полуразрешим — Тьюринг полный

Теорема 1.2. U — не разрешим

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

```

1 fn q(x: Word) -> Res {
2   if inU((x, x)) {
3     return 0;
4   } else {
5     return 1;
6   }
7 }

```

q никогда не зависит. 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, $A \cap B$ – разрешим

Свойство 3. A – разрешим, $\bar{A} = \Sigma^* \setminus A$ – разрешим

Свойство 4. A, B полурешимы, $A \cap B$ – полурешим

Свойство 5. A, B полурешимы, $A \cup B$ – полурешим

Доказательство.

```

1 fn p() -> Res {
2   for i in [0..] {
3     if inA.await(t) {
4       return 1;
5     }
6     if inB.await(t) {
7       return 1;
8     }
9   }
10 }

```

□

Теорема 1.3 (Поста). L и \bar{L} оба полурешимы $\implies L$ – разрешим

Доказательство.

```

1 fn inL(x: Word) -> Res {
2   for t in [0..] {
3     if inL(x).await(t) {
4       return 1;
5     }
6     if inCL(x).await(t) { // CL =  $\bar{L}$ 
7       return 0;
8     }
9   }
10 }

```

□

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

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