
Реализация ветвления на машине Тьюринга.

Есть три машины Тьюринга:

- **ТР** — вычисляет предикат (условие) $P(x)$.
Результат: $P(x)=0$ ($P(x)=0$ (ложь) или $P(x)=1$ ($P(x)=1$ (истина)).
- **T₁** — вычисляет функцию $f_1(x)$.
- **T₂** — вычисляет функцию $f_2(x)$.

Все три машины работают в одном внешнем алфавите.

2. Переобозначение состояний

Состояния машин переименовываются, чтобы в объединённой машине они не пересекались. Например:

- Состояния ТРТР получают верхний индекс pp : $q_{1p}, q_{2p}, \dots, q_{1p}, q_{2p}, \dots$
- Состояния T₁T₁ — верхний индекс 11 : $q_{11}, q_{21}, \dots, q_{11}, q_{21}, \dots$
- Состояния T₂T₂ — верхний индекс 22 : $q_{12}, q_{22}, \dots, q_{12}, q_{22}, \dots$

3. Схема ветвления

Мы хотим создать новую машину **T**, которая вычисляет:

$$f(x) = \begin{cases} f_1(x), & \text{если } P(x)=0 \\ f_2(x), & \text{если } P(x)=1 \end{cases} \quad f(x) = \begin{cases} f_1(x), & \text{если } P(x)=0 \\ f_2(x), & \text{если } P(x)=1 \end{cases}$$

Алгоритм:

1. Дублирование данных

На ленте исходные данные xx копируются, получается две копии рядом.

2. Вычисление $P(x)$

Машина начинает с начального состояния q_i (которое было q_{ip} машины ТРТР) и обрабатывает **левую копию** данных, оставляя правую копию нетронутой.

3. Очистка левой части

После того как ТРТР завершит работу, результат предиката (0 или 1) известен внутри состояния машины.

Затем стираются все ячейки **левее правой копии** (включая левую копию и мусор от вычисления PP).

4. Переход к правой копии

Указатель ставится на первую непустую ячейку **правой копии** данных xx .

- Если $P(x)=0$, то переходим в **начальное состояние машины T_1** (q_1).
- Если $P(x)=1$, то переходим в **начальное состояние машины T_2** (q_2).

5. Вычисление f_1 или f_2

Машина выполняет соответствующую программу T_1 или T_2 на правой копии данных.

6. Общее завершение

Заключительные состояния q_{f1} и q_{f2} машин T_1 и T_2 отождествляются в одно общее **заключительное состояние q_f** машины T . На ленте — результат $f_1(x)$ или $f_2(x)$.

4. Зачем дублирование данных?

Если бы мы вычисляли $P(x)$ прямо на исходных данных xx , то после вычисления предиката исходные данные могли быть испорчены, и для вычисления f_1 или f_2 они были бы недоступны.

Дублирование позволяет сохранить **оригинальные данные** для второй фазы вычислений.

5. Итог

Таким образом, **ветвление** в теории машин Тьюринга реализуется через:

1. Дублирование входа.
2. Вычисление условия на одной копии.
3. Очистку рабочей области и выбор соответствующей машины для работы со второй копией.

Это аналог условного оператора `if $P(x)$ then $f_2(x)$ else $f_1(x)$` в обычных языках программирования.