▶ **[hello.tig]**

```
print("Hello World!\n")
```

▶ **[fibonacci.tig]**

```
let
  function fibonacci(n : int) : int =
    if n <= 0 then
      1
    else if n = 1 then
      1
    else
      fibonacci(n-1) + fibonacci(n-2)
in
  for i := 1 to 8 do
    (print_int(fibonacci(i)); print("\n"))
end
```

▶ **[read_unsigned.tig]**

```
let
  /* Read a positive integer from the standard input.
     Returns -1 on error */
  function read_unsigned() : int =
    let
      var result := 0
      var c := getchar()
      var valid := 1
    in
      /* Skip empty input */
      if c = "" then
        -1
      else (
        /* Process characters until newline or EOF */
        while c <> "" & c <> "\n" do (
          /* Check if character is a digit (0-9) */
          if ord(c) >= ord("0") & ord(c) <= ord("9") then
            result := result * 10 + (ord(c) - ord("0"))
          else
            valid := 0;
```

```
          c := getchar()
        );

        /* Return result or -1 on error */
        if valid = 1 then
          result
        else
          -1
      )
    end

  var a : int := read_unsigned()
in
  print_int(a*2);
  print("\n")
end
```

▶ [regexp1.txt]

```
(a(a*b*)*)*
```

▶ [regexp2.txt]

```
(b*ab*a)*b*
```

Automata Determinisation

▶ What is the language accepted by the automaton in the figure below?

Автомат приймає всі слова, які можна утворити шляхом переходів зі стартового стану **(1)** до кінцевого стану **(7)**.

Він містить **ε-переходи** (порожні переходи), що дозволяють рухатися між станами без зчитування символів.

Розглянемо всі можливі шляхи:

- Зі **стану 1** можна перейти по **x** до **стану 5**, потім по **ε** до **стану 6**, потім по **ε** до **стану 7**: це приймає **рядок "x"**

- Зі **стану 1** можна перейти по **ε** до **стану 2**, потім по **y** до **стану 6**, потім по **ε** до **стану 7**: це приймає **рядок "y"**.
- Зі стану **1** можна перейти по **ε** до **стану 2**, потім по **ε** до **стану 3**, потім по **ε** до **стану 4**, потім по **ε** назад до **стану 1**, потім слідувати **шляху "x"**: це приймає **рядки типу "x", "xx", "xxx" тощо**.
- Зі **стану 5** можна перейти по **z** до **стану 2**, потім слідувати **шляху "y"**: це приймає **рядок "xzy"**.

Мова, що приймається, складається з рядків, що починаються з **"x"**, **"y"** або **"xzy"**, які можуть бути доповнені повтореннями цих шаблонів з можливістю повернення назад через **епсилон-переходи**.

▶ Show that it is not deterministic.

Цей автомат не є детермінованим з декількох причин:

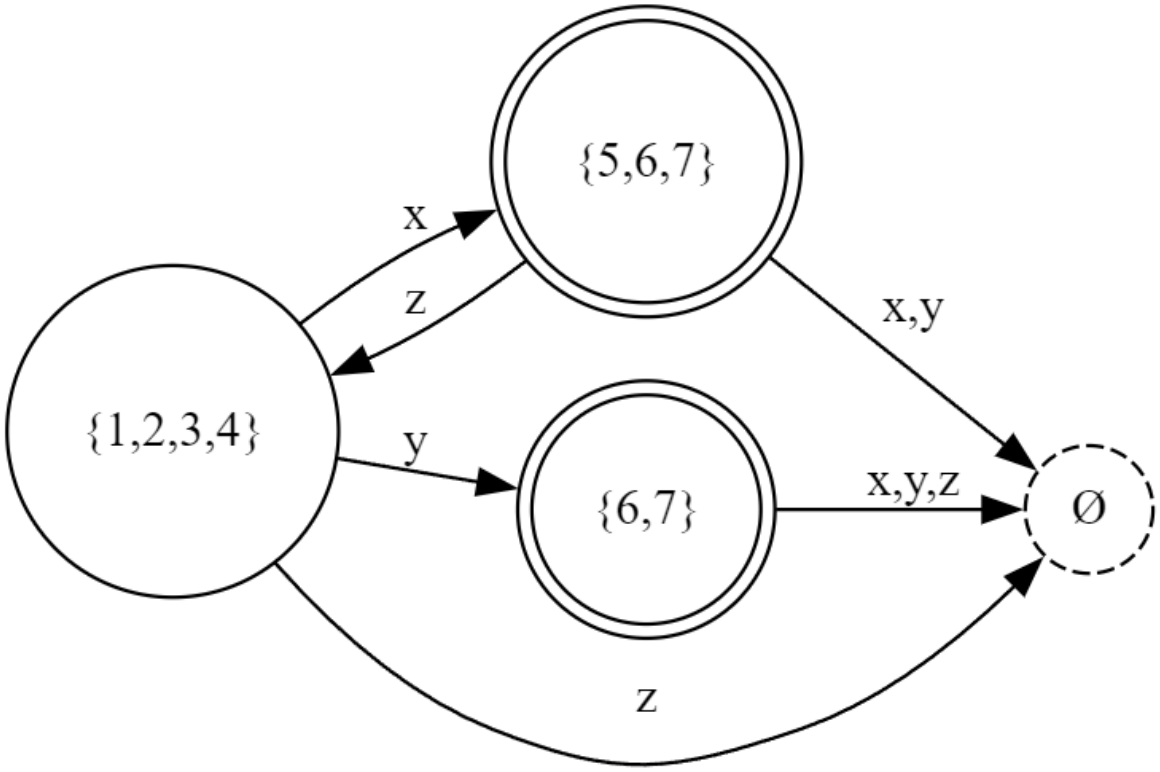
1. Він містить **епсилон-переходи (ε)**, які не дозволені в детермінованих скінченних автоматах (DFA).
2. Зі **стану 1** існує кілька можливих переходів при одному і тому ж вхідному символі:
    - Можна перейти до **стану 5** при вхідному символі **"x"**
    - Також можна перейти до **стану 2** по **ε**, і потенційно обробити інші вхідні символи
3. Зі **стану 2** можливі кілька **епсилон-переходів** (до **стану 3** або при вхідному символі **"y"** до **стану 6**)

Конкретний приклад недетермінізму: у **стані 1** автомат повинен "вгадати", чи слідувати **ε-переходу** до **стану 2** чи **x-переходу** до **стану 5** при обробці вхідного рядка.

▶ Determinise it.

Щоб зробити автомат детермінованим, потрібно:

1. Видалити **ε-переходи**, об'єднавши всі стани, які можуть бути досягнуті ними.
2. Створити нові стани, які представлятимуть множини станів NFA.
3. Побудувати переходи для кожного символу так, щоб кожен стан мав лише один наступний стан для кожного символу.




```
digraph StateTransitions {
    // Налаштування графа
    rankdir=LR;
    bgcolor="white";
    node [color=black, fontcolor=black];
    edge [color=black, fontcolor=black];

    // Вузли
    node1 [label="{1,2,3,4}", shape=circle];
    node2 [label="{5,6,7}", shape=doublecircle, peripheries=2];
    node3 [label="{6,7}", shape=doublecircle, peripheries=2];

    // Пуста множина
    empty [label="∅", shape=circle, style=dashed];

    // Переходи
    node1 -> node2 [label="x"];
    node1 -> node3 [label="y"];
    node1 -> empty [label="z"];
    node2 -> node1 [label="z"];
    node2 -> empty [label="x,y"];
    node3 -> empty [label="x,y,z"];
}
```