Fundamental Computer Science
Turing Machines (extensions)
Training session

Denis Trystram

MoSIG1-M1Info, 2021

Exercise (non deterministic TM)

- Consider a set $A = \{a_1, a_2, \ldots, a_n\}$ of positive integers and an integer $w \in \mathbb{N}$.

  Give a Non-deterministic Turing Machine that *recognizes* the language $L = \{A' \subseteq A : \sum_{a_i \in A'} a_i = w\}$.

Solution

1. choose non-deterministically a set $A' \subseteq A$
2. add the elements of A'
3. if they sum up to w, then *accept*

Solution

1. choose non-deterministically a set $A' \subseteq A$
2. add the elements of A'
3. if they sum up to w, then *accept*

- How to choose A' non-deterministically?
  - produce all binary numbers of n digits
  - start from $00\ldots0$ and add 1 at each iteration

Exercise RAM

- Write a program for a Random Access Turing Machine that multiplies two integers.
  Assume that the initial configuration is $(1; 0, a_1, a_2, 0; \emptyset)$

RAM (solution)

- Write a program for a Random Access Turing Machine that multiplies two integers.
  Assume that the initial configuration is $(1; 0, a_1, a_2, 0; \emptyset)$

while $R_1 > 0$ **do**
    $R_1 \leftarrow R_1 - 1$
    $R_3 \leftarrow R_3 + R_2$

RAM (solution)

- Write a program for a Random Access Turing Machine that multiplies two integers.
  Assume that the initial configuration is $(1; 0, a_1, a_2, 0; \emptyset)$

while $R_1 > 0$ **do**
    $R_1 \leftarrow R_1 - 1$
    $R_3 \leftarrow R_3 + R_2$

or (all computations should pass through R_0)

$R_0 \leftarrow R_1$
while $R_0 > 0$ **do**
    $R_0 \leftarrow R_0 - 1$
    $R_1 \leftarrow R_0$
    $R_0 \leftarrow R_3$
    $R_0 \leftarrow R_0 + R_2$
    $R_3 \leftarrow R_3$

RAM (solution)

- Write a program for a Random Access Turing Machine that multiplies two integers.
  Assume that the initial configuration is $(1; 0, a_1, a_2, 0; \emptyset)$

```
while R_1 > 0 do
    R_1 ← R_1 − 1
    R_3 ← R_3 + R_2
```

or (all computations should pass through R_0)

```
R_0 ← R_1
while R_0 > 0 do
    R_0 ← R_0 − 1
    R_1 ← R_0
    R_0 ← R_3
    R_0 ← R_0 + R_2
    R_3 ← R_3
```

```
1: load 1
2: jzero 9
3: sub =1
4: store 1
5: load 3
6: add 2
7: store 3
8: jump 1
9: halt
```