Non-returning Turing machines

Marek Surovič
xsurov03 at stud.fit.vutbr.cz

Faculty of Information Technology
Brno University of Technology

December 12, 2012

Outline

- Turing machines and non-returning Turing machines

Outline

- Turing machines and non-returning Turing machines
- Non-returning Turing machines and Finite Automata

Outline

- Turing machines and non-returning Turing machines
- Non-returning Turing machines and Finite Automata
- Applications of non-returning Turing machines

Turing machines

Turing machines are a 6-tuple:

$$M = (Q, \Sigma, \Gamma, R, s, F)$$

where

- Q is a finite set of states
- Σ is a tape alphabet, such that $\Sigma \cap Q = \varnothing$
- Γ is an input alphabet, such that $\triangle \in \Gamma$ and $\Sigma \subseteq \Gamma$
- R is a finite set of rules of the form $qX \vdash pYt$ where
  - $p, q \in Q$
  - $X, Y \in \Gamma$
  - $t \in \{\rightarrow, \leftarrow, \downarrow\}$
- $s \in Q$ is the start state.
- $F \subseteq Q$ is a set of of final states

Non-returning Turing machines

Non-returning Turing machines are a 6-tuple:

$$M = (Q, \Sigma, \Gamma, R, s, F)$$

where

- Q is a finite set of states
- Σ is a tape alphabet, such that $\Sigma \cap Q = \varnothing$
- Γ is an input alphabet, such that $\triangle \in \Gamma$ and $\Sigma \subseteq \Gamma$
- R is a finite set of rules of the form $qX \vdash pYt$ where
  - $p, q \in Q$
  - $X, Y \in \Gamma$
  - $t \in \{\rightarrow, \downarrow\}$
- $s \in Q$ is the start state.
- $F \subseteq Q$ is a set of of final states

But I undestood none of this. . .

A Turing machine

But I undestood none of this. . .

A Turing machine

- is essentially a very simple computer.

But I undestood none of this...

A Turing machine

- is essentially a very simple computer.
- has a potentially infinite tape with symbols on it.

But I undestood none of this...

A Turing machine

- is essentially a very simple computer.
- has a potentially infinite tape with symbols on it.
- has a tape head that reads and writes symbols on the tape and can move left, right or stay put.

But I undestood none of this. . .

A Turing machine

- is essentially a very simple computer.
- has a potentially infinite tape with symbols on it.
- has a tape head that reads and writes symbols on the tape and can move left, right or stay put.
- has finite state control that tells the tape head what to do.

But I undestood none of this. . .

A Turing machine

- is essentially a very simple computer.
- has a potentially infinite tape with symbols on it.
- has a tape head that reads and writes symbols on the tape and can move left, right or stay put.
- has finite state control that tells the tape head what to do.

A non-returning Turing machine

But I undestood none of this. . .

A Turing machine

- is essentially a very simple computer.
- has a potentially infinite tape with symbols on it.
- has a tape head that reads and writes symbols on the tape and can move left, right or stay put.
- has finite state control that tells the tape head what to do.

A non-returning Turing machine

- is the same thing as the Turing machine, except the tape head can't move left.

So what difference does it make?

So what difference does it make?

The NRTM has almost no memory.

So what difference does it make?

The NRTM has almost no memory.

Almost?

So what difference does it make?

The NRTM has almost no memory.

Almost?

If the tape head writes something onto the tape and stays put, it will read the self-written symbol in the next step.

Finite Automaton

Finite automata are a 5-tuple:

$$M = (Q, \Sigma, R, s, F)$$

where

- Q is a finite set of states
- Σ is an input alphabet
- R is a finite set of rules of the form $pa \rightarrow q$ where
  - $p, q \in Q$
  - $a \in \Sigma \cup \{\varepsilon\}$
- $s \in Q$ is the start state
- $F \subseteq Q$ is a set of final states

Tell me more!

Finite automata

Tell me more!

Finite automata

- are something like computers as well, but not as powerful as Turing machines.

Tell me more!

Finite automata

- are something like computers as well, but not as powerful as Turing machines.
- have an input tape like the Turing machine.

Tell me more!

Finite automata

- are something like computers as well, but not as powerful as Turing machines.
- have an input tape like the Turing machine.
- have a tape head like the Turing machine, but they can't move left and they can't write anything onto the tape.

Why are we talking about these then?

Why are we talking about these then?

Because NRTMs are equivalent to finite automata, thus they are transformable between each other.

Why are we talking about these then?

Because NRTMs are equivalent to finite automata, thus they are transformable between each other.

So how do we transform them?

Why are we talking about these then?

Because NRTMs are equivalent to finite automata, thus they are transformable between each other.

So how do we transform them?

We take the NRTM apart and then build a FA out of the parts.

What tools are we going to need?

What tools are we going to need?

- A lhs() function that can take out the qX from the $qX \vdash pYt$ rule of the NRTM.

What tools are we going to need?

- A lhs() function that can take out the qX from the $qX \vdash pYt$ rule of the NRTM.
- A rhs() function that can take out the pYt from the $qX \vdash pYt$ rule of the NRTM.

What tools are we going to need?

- A lhs() function that can take out the qX from the $qX \vdash pYt$ rule of the NRTM.
- A rhs() function that can take out the pYt from the $qX \vdash pYt$ rule of the NRTM.
- An alph() function that can split various strings into symbols.

Any preparations?

Any preparations?

- Two sets that split the rules of the NRTM:

Any preparations?

- Two sets that split the rules of the NRTM:
  - R_{move} which contains the rules that move the tape head to the right.

Any preparations?

- Two sets that split the rules of the NRTM:
  - R_{move} which contains the rules that move the tape head to the right.
  - R_{stay} which contains the rules that do not move the tape head.

Any preparations?

- Two sets that split the rules of the NRTM:
  - R_{move} which contains the rules that move the tape head to the right.
  - R_{stay} which contains the rules that do not move the tape head.
- Two auxiliary sets: A and B.

The Algorithm - Inputs and outputs

The Algorithm - Inputs and outputs

- **Input**: A non-returning Turing machine $M_t = (Q_t, \Sigma_t, \Gamma_t, R_t, s_t, F_t)$

The Algorithm - Inputs and outputs

- **Input**: A non-returning Turing machine $M_t = (Q_t, \Sigma_t, \Gamma_t, R_t, s_t, F_t)$
- **Output**: A finite automaton $M_k = (Q_k, \Sigma_k, R_k, s_k, F_k)$, such that $L(M_t) = L(M_k)$

The Algorithm - Inputs and outputs

- **Input**: A non-returning Turing machine $M_t = (Q_t, \Sigma_t, \Gamma_t, R_t, s_t, F_t)$
- **Output**: A finite automaton $M_k = (Q_k, \Sigma_k, R_k, s_k, F_k)$, such that $L(M_t) = L(M_k)$
- **Idea**: Using lhs(), rhs() and alph() regroup and rebuild the states and rules of the NRTM so that they form a FA that simulates the NRTM and all of it's properties.

The Algorithm - Inputs and outputs

- **Input**: A non-returning Turing machine $M_t = (Q_t, \Sigma_t, \Gamma_t, R_t, s_t, F_t)$
- **Output**: A finite automaton $M_k = (Q_k, \Sigma_k, R_k, s_k, F_k)$, such that $L(M_t) = L(M_k)$
- **Idea**: Using lhs(), rhs() and alph() regroup and rebuild the states and rules of the NRTM so that they form a FA that simulates the NRTM and all of it's properties.

  Information about which symbol is under the tape head at a given moment is stored in the names of states of the FA.

The Algorithm - Inputs and outputs

- **Input**: A non-returning Turing machine $M_t = (Q_t, \Sigma_t, \Gamma_t, R_t, s_t, F_t)$
- **Output**: A finite automaton $M_k = (Q_k, \Sigma_k, R_k, s_k, F_k)$, such that $L(M_t) = L(M_k)$
- **Idea**: Using lhs(), rhs() and alph() regroup and rebuild the states and rules of the NRTM so that they form a FA that simulates the NRTM and all of it's properties.

  Information about which symbol is under the tape head at a given moment is stored in the names of states of the FA.

  Moves on the tape and operations with the symbols on the tape are simulated via transitions between various states of the FA.

The Algorithm - Set initializations

The Algorithm - Set initializations

- $Q_k := \{s_k\}; \Sigma_k := \Sigma_t; R_k := \{\}; F_k := \{\}; A := \{\}; B := \{\};$

The Algorithm - Set initializations

- $Q_k := \{s_k\}; \Sigma_k := \Sigma_t; R_k := \{\}; F_k := \{\}; A := \{\}; B := \{\};$
- Note that we made the starting state for the new FA beforehand.

The Algorithm - State constructions

- We need to construct source and destination states for every rule of our NRTM.

- We need to construct source and destination states for every rule of our NRTM.
- for each $r \in R_t$ do begin
  $$Q_k := Q_k \cup \{lhs(r)\} \cup \{rhs(r)\};$$
  end;

The Algorithm - State constructions

- We need to construct source and destination states for every rule of our NRTM.
- for each $r \in R_t$ do begin

  $$Q_k := Q_k \cup \{lhs(r)\} \cup \{rhs(r)\};$$

  end;
- But since the NRTM retains the tape and it's properties from the original TM we need some extra states as well.

The Algorithm - State constructions

- We need to construct source and destination states for every rule of our NRTM.
- for each $r \in R_t$ do begin
  $Q_k := Q_k \cup \{lhs(r)\} \cup \{rhs(r)\};$
  end;
- But since the NRTM retains the tape and it's properties from the original TM we need some extra states as well.
- for each $r \in R_t$ do begin
  if $r \in R_{move}$ then
  $Q_k := Q_k \cup \{qa : q \in Q_t, q \in alph(rhs(r)), a = \triangle\};$
  end;

The Algorithm - Rule constructions

The Algorithm - Rule constructions

- Since the NRTM can start with any symbol under the tape head, we need a new starting state and rules that get us from the starting state to the various states that represent possible starting symbols on the tape.

The Algorithm - Rule constructions

- Since the NRTM can start with any symbol under the tape head, we need a new starting state and rules that get us from the starting state to the various states that represent possible starting symbols on the tape.
- $R_k := R_k \cup \{pa \to q : p \in Q_k, p = s_k, q \in Q_k, s_t \in alph(q), a = \varepsilon\};$

The Algorithm - Rule constructions

The Algorithm - Rule constructions

- The NRTM retains the ability to write onto the tape, which is a problem if it stays on the same symbol afterwards. This can affect the course of the computation process done by the NRTM and has to be simulated by the FA. We simulate it by reading nothing at all, but making a transition to another state.

The Algorithm - Rule constructions

- The NRTM retains the ability to write onto the tape, which is a problem if it stays on the same symbol afterwards. This can affect the course of the computation process done by the NRTM and has to be simulated by the FA. We simulate it by reading nothing at all, but making a transition to another state.
- for each $r \in R_{stay}$ do begin
  $$A := A \cup \{t : t \in R_t, lhs(t) = rhs(r)\};$$
  end;

The Algorithm - Rule constructions

- The NRTM retains the ability to write onto the tape, which is a problem if it stays on the same symbol afterwards. This can affect the course of the computation process done by the NRTM and has to be simulated by the FA. We simulate it by reading nothing at all, but making a transition to another state.
- for each $r \in R_{stay}$ do begin
  $A := A \cup \{t : t \in R_t, lhs(t) = rhs(r)\}$;
  end;
- for each $r \in A$ do begin
  $R_k := R_k \cup \{pa \to q : p \in Q_k, p = lhs(r), q \in Qk, q = rhs(r), a = \varepsilon\}$;
  end;

The Algorithm - More rule constructions!

- The moving transitions that the NRTM makes are divided into two transitions in the FA. The read and write part...

The Algorithm - More rule constructions!

- The moving transitions that the NRTM makes are divided into two transitions in the FA. The read and write part. . .
- for each $r \in R_t$ do begin
  $R_k := R_k \cup \{pa \rightarrow q : p \in Q_k, p = lhs(r), q \in Q_k, q = rhs(r), a \in alph(lhs(r)), a \in \Sigma_k\}$;
  end;

The Algorithm - Even more rule constructions!

- ...and the move part.

The Algorithm - Even more rule constructions!

- ...and the move part.
- for each $r \in R_t$ do begin

$B := \{qX \vdash pYt : qX \vdash pYt \in R_t, q \in Q_t, q \in alph(rhs(r))\}$;

for each $t \in A$ do begin

$R_k := R_k \cup \{pa \to q : p \in Q_k, p = rhs(r), q \in Q_k, q = lhs(t), a = \varepsilon\}$;

end;

end;

The Algorithm - And finally the final states.

The Algorithm - And finally the final states.

- Quite simply, we take the states that are derived from the final states in the NRTM and make them the final states of the FA.

The Algorithm - And finally the final states.

- Quite simply, we take the states that are derived from the final states in the NRTM and make them the final states of the FA.
- for each $q \in F_t$ do begin
  $F_k := F_k \cup \{p : p \in Q_k, q \in alph(p)\}$; end;

Applications!

Applications!

- Anyone got any ideas?