Theory of Computer Science

F3. GOTO-Computability

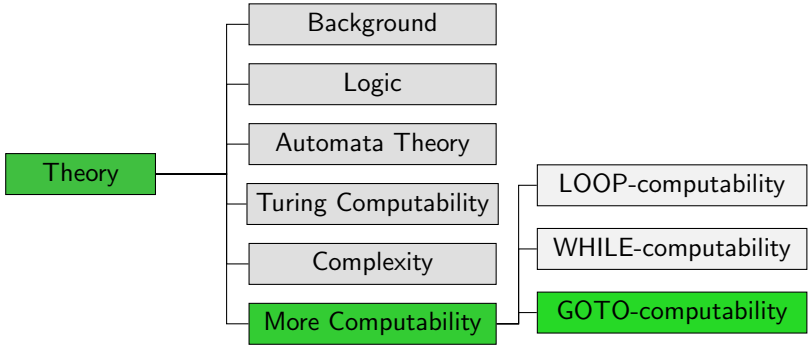
Gabriele Röger

University of Basel

May 22, 2019

F3.1 GOTO Programs

Course Overview



Motivation

We now know:

- WHILE programs are strictly more powerful than LOOP programs.
- Deterministic Turing machines are at least as powerful as WHILE programs.

Are DTMs strictly more powerful than WHILE programs or equally powerful?

To answer this question, we make a detour over one more programming formalism.

GOTO Programs: Syntax

Definition (GOTO Program)

A GOTO program is given by a finite sequence $L_1 : A_1, L_2 : A_2, \ldots, L_n : A_n$ of labels and statements.

Statements are of the following form:

- $x_i := x_j + c$ for every $i, j, c \in \mathbb{N}_0$ (addition)
- $x_i := x_j - c$ for every $i, j, c \in \mathbb{N}_0$ (modified subtraction)
- HALT (end of program)
- GOTO L_j for $1 \leq j \leq n$ (jump)
- IF $x_i = c$ THEN GOTO L_j for $i, c \in \mathbb{N}_0$, $1 \leq j \leq n$ (conditional jump)

German: GOTO-Programm, Marken, Anweisungen, Programmende, Sprung, bedingter Sprung

GOTO Programs: Semantics

Definition (Semantics of GOTO Programs)

- Input, output and variables work exactly as in LOOP and WHILE programs.
- Addition and modified subtraction work exactly as in LOOP and WHILE programs.
- Execution begins with the statement A_1.
- After executing A_i, the statement A_{i+1} is executed. (If $i = n$, execution finishes.)
- exceptions to the previous rule:
  - HALT stops the execution of the program.
  - After GOTO L_j execution continues with statement A_j.
  - After IF $x_i = c$ THEN GOTO L_j execution continues with A_j if variable x_i currently holds the value c.

GOTO-Computable Functions

Definition (GOTO-Computable)

A function $f : \mathbb{N}_0^k \to \mathbb{N}_0$ is called GOTO-computable if a GOTO program that computes f exists.

German: GOTO-berechenbar

F3.2 GOTO vs. WHILE

GOTO-Computability vs. WHILE-Computability

Theorem

Every GOTO-computable function is WHILE-computable.

If we allow IF statements, a single WHILE loop is sufficient for this.

(We will discuss the converse statement later.)

GOTO-Computability vs. WHILE-Computability

Proof sketch.

Given any GOTO program, we construct an equivalent WHILE program with a single WHILE loop (and IF statements).

Ideas:

- Use a fresh variable to store the number of the statement to be executed next.
  - $\rightsquigarrow$ The variable of course has the form x_i, but for readability we write it as *pc* for "program counter".
- GOTO is simulated as an assignment to *pc*.
- If *pc* has the value 0, the program terminates.

...

GOTO-Computability vs. WHILE-Computability

Proof sketch (continued).

Let $L_1 : A_1$, $L_2 : A_2$, ..., $L_n : A_n$ be the given GOTO program.

basic structure of the WHILE program:

```
pc := 1;
WHILE pc ≠ 0 DO
  IF pc = 1 THEN (translation of A_1) END;
  ...
  IF pc = n THEN (translation of A_n) END;
  IF pc = n + 1 THEN pc := 0 END
END
```

...

GOTO-Computability vs. WHILE-Computability

Proof sketch (continued).

Translation of the individual statements:

- $x_i := x_j + c$
  $\rightsquigarrow x_i := x_j + c;\ pc := pc + 1$
- $x_i := x_j - c$
  $\rightsquigarrow x_i := x_j - c;\ pc := pc + 1$
- HALT
  $\rightsquigarrow pc := 0$
- GOTO L_j
  $\rightsquigarrow pc := j$
- IF $x_i = c$ THEN GOTO L_j
  $\rightsquigarrow pc := pc + 1$; IF $x_i = c$ THEN $pc := j$ END

□

Intermediate Summary

We now know:

- WHILE programs are strictly more powerful than LOOP programs.
- Deterministic Turing machines are at least as powerful as WHILE programs.
- WHILE programs are at least as powerful as GOTO programs.

We now show that GOTO programs are at least as powerful as DTMs, closing the cycle DTM–WHILE–GOTO.

F3.3 Turing vs. GOTO

Turing-Computability vs. GOTO-Computability

Theorem (Turing-Computability vs. GOTO-Computability)

Every Turing-computable numerical function is GOTO-computable.

Proof.

$\rightsquigarrow$ blackboard. □

Final Result

Corollary

Let $f : \mathbb{N}_0^k \to_p \mathbb{N}_0$ be a function.

The following statements are equivalent:

- *f is Turing-computable.*
- *f is WHILE-computable.*
- *f is GOTO-computable.*

Moreover:

- *Every LOOP-computable function is Turing-/WHILE-/GOTO-computable.*
- *The converse is not true in general.*

F3.4 Summary

Summary

results of the investigation:

- another new model of computation: GOTO programs
- Turing machines, WHILE and GOTO programs are equally powerful.
  - Whenever we said "Turing-computable" or "computable" in parts D or E, we could equally have said "WHILE-computable" or "GOTO-computable".
- LOOP programs are strictly less powerful.