1. Introduction

Term Rewrite Systems (Termersetzungssysteme)

Applications

- Specification of programs
  (specify and execute algebraic specifications)
- Program analysis and verification
- Execution of programs
  (term rewriting is already a full programming language, is the basis of functional programming)
- Symbolic computation

Example: Addition of natural numbers

Express natural numbers by terms over 0 and succ

\[ 0 = 0, \quad \text{succ}(0) = 1, \quad \text{succ}(\text{succ}(0)) = 2, \ldots \]

Addition algorithm:

\[ \text{plus}(0, y) \equiv y \]
\[ \text{plus}(\text{succ}(x), y) \equiv \text{succ}(\text{plus}(x, y)) \]

(Functional program, looks like Haskell)

Execution of such a program:
apply equations from left to right = term rewriting

Compute "2+1."

\[
\begin{align*}
\text{plus (succ(succ(0)), succ(0))} & \rightarrow \\
\text{succ (plus (succ(0), succ(0)))} & \rightarrow \\
\text{succ (succ (plus (0, succ(0))))} & \rightarrow \\
\text{succ (succ (succ(0)))} & \\
\end{align*}
\]

Topics of the lecture:
- Is the result of a program deterministic? (Confluence)

\[
\begin{align*}
\text{plus (0, plus (succ(0), 0))} \\
\text{plus (0, succ (plus (0, 0)))} \\
\text{plus (succ (0), 0)}
\end{align*}
\]

This example is confluent. How can one check confluence/determinism of programs automatically?
• Does the program always stop after finitely many steps? (Termination)

Our plus-example is terminating.

But one could add:

\[ \text{plus}(x, y) \equiv \text{plus}(y, x) \]

Now the program is no longer terminating:

\[ \text{plus}(0, \text{succ}(0)) \rightarrow \text{plus}(\text{succ}(0), 0) \rightarrow \text{plus}(0, \text{succ}(0)) \rightarrow \ldots \]

How can one check termination automatically?

• Does a program satisfy its specification? (Correctness)

The plus-program could satisfy properties like:

\[ \text{plus}(\text{succ}(\text{succ}(0)), x) \equiv \text{plus}(\text{succ}(0), \text{succ}(x)) \]

\[ \text{plus}(\text{plus}(x, y), z) \equiv \text{plus}(x, \text{plus}(y, z)) \]

Do these properties hold for the plus-program, i.e., do they follow from the 2 plus-equations?

A related question are properties of abstract data types.

Ex: Data types of groups
associativity: \[ f(x, f(y, z)) = f(f(x, y), z) \]
neutral element: \[ f(x, e) = x \]
inverse fct. \( i \): \[ f(x, i(x)) = e \]

Does every group satisfy \[ i(i(v)) = v \]?

I.e.: Does this equation follow from the 3 group axioms? (Yes)

How can one find this out automatically?

• How can one make an "incomplete" program complete? (Completion)

If a program or a data type are incompletely specified, how can one extend them in a suitable way?

Structure of the lecture:

1. Introduction
2. Preliminaries (Syntax + Semantics of Terms and equations)
3. Term Rewriting and Deduction of Equations
4. Termination of Term Rewriting
5. Confluence — n —

6. Completion — n —

**Organisation**

- Lecture in English
- German Course Notes (on the web)
- English Notes from the lecture available on the web
- Very old video of parts of the lecture (on the web)
- Website: [http://verify.rwth-aachen.de/tes15](http://verify.rwth-aachen.de/tes15)

- Lecture on **TUE + THU**
  - Ex. Course on **FRI1** (Florian Frohn)
  - This week: 3 lectures (lecture instead of ex. course on FRI1)

- Lecture for Master (Informatik, SSE, Math) and Bachelor (Informatik)
  - Wahl/ Pflicht Theorie or
  - Vorgezogene Masterprüfung
  - Bachelor: register for exam at EPA,
Exercises
- weekly exercise sheets
- solve in groups of 2
- submit at beginning of ex. course or in box (EN 2nd floor)
- 50% of points needed to participate in the exam
- date of exam (con’t): Feb 22, 2016
  2nd exam: March 18, 2016
- to participate in exercises:
  please register via our website
  (≠ registering for exam ≠ L2P)
  until Friday, Oct. 23