Turing machines over τ-structures

This presentation is inspired from presentations by B. Poizat and M. Prunescu:

```
@bookpoizat:95,
AUTHOR = B. Poizat,
TITLE = Les Petits Cailloux,
PUBLISHER = Aléas, Lyon,
SERIES = Nur Al-Mantiq War-Ma'rifah,
NUMBER = 3,
YEAR = 1995
```

```
@inproceedingsprunescu:06,
AUTHOR = Mihai Prunescu,
TITLE = Fast Quantifier Elimination Means P = NP,
BOOKTITLE = CiE,
YEAR = 2006,
PAGES = 459-470
```

First order structures $\mathfrak{A}$ over a vocabulary τ

- A vocabulary is a set of function symbols, relation symbols and constant symbols of arity $\mathrm{rf}(i), \mathrm{rr}(i) \in \mathbb{N}$.

- Constant symbols: c_i of arity 0
  Relation symbols: $R_{i,\mathrm{rr}(i)}$ of arity $\mathrm{rr}(i) \geq 1$.
  Function symbols: $F_{i,\mathrm{rf}(i)}$ of arity $\mathrm{rf}(i) \geq 1$.
  Vocabulary: $\tau \subseteq \{c_i : i \in \mathbb{N}\} \cup \{R_{i,\mathrm{rr}(i)} :\in \mathbb{N}\} \cup \{F_{i,\mathrm{rr}(i)} :\in \mathbb{N}\}$

- A τ-structure $\mathfrak{A}$ is an interpretation of a vocabulary.
  The universe of $\mathfrak{A}$ is a non-empty set A.
  The interpretation of c_i is an element $c_i^A \in A$.
  The interpretation of $R_{i,\mathrm{rr}(i)}$ is a relation $R^A_{i,\mathrm{rr}(i)} \subseteq A^{\mathrm{rr}(i)}$.
  The interpretation of $F_{i,\mathrm{rf}(i)}$ is a function $F^A_{i,\mathrm{rf}(i)} : A^{\mathrm{rf}(i)} \to A$.

Unless otherwise stated, vocabularies are **finite**.

Turing machines for τ-structures, I

In τ-structures $\mathfrak{A}$ we omit the superscript, when the context is clear, and do not distinguish between symbols and their interpretation. We also assume that each element of its universe A has a unique name.

The Turing machine $\mathbf{M}_\tau$ works as follows:

- $\mathbf{M}$ is a multi-tape machine with finitely tapes and states.
  The number of tapes as it least $\max_{i,j}\{\mathrm{rr}(i), \mathrm{rf}(j) : i, j \in \mathbb{N}\}$.
- Tape number i consists of cells indexed by $\mathbb{Z}$ and has a head H_i.
- Each cell of $\mathbf{M}$ contains either a name of an element $a \in A$
  or the symbol $\square$ for `blank`.
- Let x_i be the content of a cell on tape i with head position H_i.
- $\mathbf{M}$ is initialized with the input data.

Turing machines for τ-structures, II

The following program steps may occur:

- `stop`, the halting instruction;
- H_i+ or H_i-: Moving the head on tape i;
- $x_i := x_j$, where x_j is the content of cell j at head position H_j.
- $x_i := F_{i,\mathrm{rf}(i)}(x_{j_1}, \ldots, x_{j_{\mathrm{rf}(i)}})$, with x_{j_ℓ} as above.
- `If` $R_{i,\mathrm{rf}(i)}(x_{j_1}, \ldots, x_{j_{\mathrm{rf}(i)}})$`, goto state` q`, else goto` q' with x_{j_ℓ} as above.
- `If` $H_i = \square$ `goto` q `else to` q'.

Each such step is performed in one unit of time.

Decision problems and functions for $\mathbf{M}$.

Let $A^\star = \bigsqcup_{i\in\mathbb{N}} A^n$.

Elements of $A^\star$ are finite words over A, considered as an alphabet.

- A problem P is a subset of $A^\star$.
- $\mathbf{M}$ decides P if for all $x \in A^\star$ the machine halts in an accepting state if $x \in P$, and halts in a rejecting state if $x \notin P$.
- $\mathbf{M}$ computes a function $f : A^\star \to A$ if $\mathbf{M}$ halts on input $a \in A^\star$ with $f(a)$ in the output cell.
- Given a machine $\mathbf{M}$, halting sets are the sets

$$\text{Halt}_+(\mathbf{M}) = \{a \in A^\star : \mathbf{M} \text{ halts and accepts input } a\}$$
$$\text{Halt}_-(\mathbf{M}) = \{a \in A^\star : \mathbf{M} \text{ halts and rejects input } a\}$$
$$\text{Halt}(\mathbf{M}) = \text{Halt}_+(\mathbf{M}) \cup \text{Halt}_-(\mathbf{M}) = \{a \in A^\star : \mathbf{M} \text{ halts on input } a\}$$

Determinsitic complexity classes over $\mathfrak{A}$

- A problem P (a function f) is in the class $\mathrm{Comp}_{\mathfrak{A}}$ ($\mathrm{FComp}_{\mathfrak{A}}$) if there exists a machine $\mathbf{M}$ over $\mathfrak{A}$ which decides P (computes f).

- Let $t : \mathbb{N} \to \mathbb{N}$ be a function.'
  A problem P (a function f) is in the class $\mathrm{Comp}_{\mathfrak{A}}(t)$ ($\mathrm{FComp}_{\mathfrak{A}}(t)$) if there exists a machine $\mathbf{M}$ over $\mathfrak{A}$ which decides P (computes f) in time $t(n)$ for input of size n.

- A problem P is in the class $\mathbf{P}_{\mathfrak{A}}$ if the there a a machine $\mathbf{M}$ over $\mathfrak{A}$ which decides P in polynomial time (in unit cost).

- A function f is in the class $\mathbf{FP}_{\mathfrak{A}}$ if the there a a machine $\mathbf{M}$ over $\mathfrak{A}$ which computes f in polynomial time (in unit cost).

- Other deterministic complexity classes are defined similarly.

  Say exponential time and its variations.

Examples over the ordered field of reals $\mathcal{R}_{ofield} = \langle \mathbb{R}, +, \times, \leq, 0, 1 \rangle$

- Input a matrix $m \in \mathbb{R}^{n \times n}$. Compute the determinant $\det(m)$.
  This is in $\mathbf{FP}_{\mathcal{R}_{ofield}}$.

- Input a matrix $m \in \mathbb{R}^{n \times n}$. Compute the permanent $\mathrm{per}(m)$.
  This is in exponential time but not known to be in $\mathbf{FP}_{\mathcal{R}_{ofield}}$.

- The problem $P_{integer} = \{a \in \mathbb{R} : a \in \mathbb{Z}\}$ is in $\mathrm{Comp}_{\mathcal{R}_{ofield}}$.

  For positive input x we subtract 1 till the results is smaller than $-x$. If we passed through 0 we accept, else we reject. For negative input we first put $y := -x$ and test for y.

- There is no function $t(n)$ such that $P_{integer} \in \mathrm{Comp}_{\mathcal{R}_{ofield}}(t)$.

  To see this we first note that all inputs are of size 1.
  So a machine deciding $P_{integer}$ uses time bounded by a constant.

  We shall see in lectures SHORTQE and TABLE that this contradicts the characterization of halting sets.

$P_{integer}$ is NOT computable without order

Let $A(\bar{a})$ be a program with machine constants $\bar{a}$ which decides $\mathbb{N}$ over $\mathbb{R}$.

Let $c \in \mathbb{R}$ be non-algebraic over $\mathbb{Q}[\bar{a}]$. So $A(\bar{a})$ rejects c.

In particular, all tests of $A(\bar{a})$ test whether $c \neq t(\bar{a})$ where $t(\bar{a})$ is a rational function.

But there are only finitely many such tests in $A(\bar{a})$.

Therefore, there is $n_0 \in \mathbb{N}$ with the same test results.

This n_0 is rejected by $A(\bar{a})$, **a contradiction**! □

Why does this argument not work in the presence of order?

Non-deterministic complexity classes

In Turing machines using binary strings there are two definitions of non-determinism for polynomial time computations:

Let $p(x) \in \mathbb{Z}[x]$ be a polynomial with $p(a) \in \mathbb{N}$ for $a \in \mathbb{N}$.

Coin tosses: On input of size n we are allowed $p(n)$ coin tosses and use their results in the machine.

This defines the non-deterministic complexity class $\mathbf{NP}^{binary}$.

Guesses: On input of size n we are allowed to guess $p(n)$ values in $\{0, 1\}$ and use their results in the machine.

This defines the non-deterministic complexity class $\mathbf{NP}^{guess}$.

Theorem 1 (Folklore)
For Turing machines using binary strings the two classes coincide:

$$\mathbf{NP}^{binary} = \mathbf{NP}^{guess}$$

Infinite guess space

If $\mathfrak{A}$ is infinite the two definitions differ:

- $\mathbf{NP}_{\mathfrak{A}}^{binary}$ gives $2^{p(n)}$ choices for coin tosses on input of size n, which is finite.
- $\mathbf{NP}_{\mathfrak{A}}^{guess}$ allows for infinitely many guesses, even on fixed input size.

It follows that

- Problems $P \in \mathbf{NP}_{\mathfrak{A}}^{binary}$ are always decidable in exponential time, hence $P \in \mathrm{Comp}_{\mathfrak{A}}$.
- Problems $P \in \mathbf{NP}_{\mathfrak{A}}^{guess}$ are not always decidable, hence it is not obvious that $P \in \mathrm{Comp}_{\mathfrak{A}}$.

Typical problems in $\mathbf{NP}_{\mathbb{R}_{ofield}}$

HN($\mathbb{R}$): (Hilbert's Nullstellensatz): Given the coefficients in $\mathbb{R}$ of a finite set F of polynomials with $F \in \mathbb{R}[x_1, \ldots, x_n]$ for each $f \in F$, decide whether there is $\bar{a} \in \mathbb{R}^n$ such that for each $f \in F$ we have $f(\bar{a}) = 0$.

4-FEAS($\mathbb{R}$): Given one polynomial $F \in \mathbb{R}[x_1, \ldots, x_n]$ of degree ≤ 4, is there $F \in \mathbb{R}[x_1, \ldots, x_n]$ such that $f(\bar{a}) = 0$.

LPF($\mathbb{R}$): (Linear programming feasibility): Given a set of m inequalties in n variables

$$A_i \cdot x = \sum_j a_{i,j} \cdot x_j \geq b_i, \ \ i = 1, \ldots, m$$

with $a_{i,j} \in \mathbb{R}$, decide whether there is $c \in \mathbb{R}^n$ such that $A_i \cdot c \geq b_i$ for all $i = 1, \ldots, m$.

Is every problem $P \in \mathbf{NP}_{\mathfrak{A}}$ also in $\mathrm{Comp}_{\mathfrak{A}}$?

For coin tossing we always have $P \in \mathbf{NP}_{\mathfrak{A}}^{binary}$ implies $\mathrm{Comp}_{\mathfrak{A}}$.

For guessing, whether $P \in \mathbf{NP}_{\mathfrak{A}}^{guess}$ implies $\mathrm{Comp}_{\mathfrak{A}}$.
is not obvious at all.

- For the ring of integers $\mathbb{Z}_{ring}$ the problem $\mathrm{HN}(\mathbb{Z}_{ring})$ is not in $\mathrm{Comp}_{\mathbb{Z}_{ring}}$.

  This is due to the undecidability of existential formulas in $\mathbb{Z}_{ring}$ by the Davis-Putnam-Robinson-Matyasevich (DPRM) Theorem.

- For the ordered field of reals $\mathbb{R}_{ofield}$ the problem $\mathrm{HN}(\mathbb{R}_{ofield})$ is in $\mathrm{Comp}_{\mathbb{R}_{field}}$.

  This is due to the decidability of the theory of real closed fields by the Tarski-Seidenberg Theorem.

Both are highly non-trivial results.

Polynomial time Turing reducibility

We want to introduce oracle calls or, less mystic, subroutine calls.

Let P be a problem.

Let $Oracle(P)$ be a device (oracle, subroutine call) which on input $x_1, \ldots, x_n$ solves problem P at unit cost.

We are not too precise here: We have to define an oracle machine $M(P)$.

- It has an oracle-input tape.
- It has an oracle call instruction, which reads the input from the oracle-input tape and returns 1 if the input is accepted by P, and 0 otherwise.
- Now the result of an oracle call can be used in the test.

Given to problems P_1 and P_2 we say that P_1 is
polynomial time Turing reducible to P_2 and write $P_1 <_{PT} P_2$,
if there is a polynomial time oracle machine $M(P_2)$ which solves P_1.

P_2 is **NP**-complete if for every problem $P_1 \in \mathbf{NP}$ we have $P_1 <_{PT} P_2$.

$\mathbf{NP}_{\mathfrak{A}}$-complete problems

HN($\mathbb{R}$): (Hilbert's Nullstellensatz): Given the coefficients in $\mathbb{R}$ of a finite set F of polynomials with $F \in \mathbb{R}[x_1, \ldots, x_n]$ for each $f \in F$, decide whether there is $\bar{a} \in \mathbb{R}^n$ such that for each $f \in F$ we have $f(\bar{a}) = 0$.

4-FEAS($\mathbb{R}$): Given one polynomial $F \in \mathbb{R}[x_1, \ldots, x_n]$ of degree ≤ 4, is there $F \in \mathbb{R}[x_1, \ldots, x_n]$ such that $f(\bar{a}) = 0$.

CSAT: The satisfiability problem for τ-circuits (to be defined below).

FSAT: The satisfiability problem for τ-formulas (to be defined below).

The following is not known to be in $\mathbf{P}_{\mathfrak{A}}$, nor $\mathbf{NP}_{\mathfrak{A}}$-complete:

LPF($\mathbb{R}$): (Linear programming feasibility): Given a set of m inequalties in n variables $A_i \cdot x = \sum_j a_{i,j} \cdot x_j \geq b_i,\ i = 1, \ldots, m$ with $a_{i,j} \in \mathbb{R}$, decide whether there is $c \in \mathbb{R}^n$ such that $A_i \cdot c \geq b_i$ for all $i = 1, \ldots, m$.

τ-circuits, I

Let τ have at least two constant symbols 0 and 1.
A τ-circuit C is a finite labeled directed acyclic graph (DAG).

- The vertices of C are called **gates**.
- The directed edges of C are called **arrows**.
- There are **input**, **output** gates and **selection** gates.
  There are **constant, function** and **relation** gates.
- The gates input and output elements of a τ-structure $\mathfrak{A}$.
- There is at least one input gate and one output gate.
- The in-degree (fan-in) and out-degree (fan-out) of the gates are finite and bounded by the maximal arities of relation and function symbols in τ.

τ-circuits, II

- An input gate has fan-in 1. It copies the input element and sends it along the arrows.
- A constant gate has fan-in 0. It sends its constant along the arrows.
- The relation and function gates have fan-in given by the corresponding arities.
  The relation gate outputs the truth value of its relation, i.e., 0 (false) or 1 (true).
  The function gates outputs the value of its function.
- The selection gate has fan-in 3 and computes the selection function $s(x, y, z)$ with $(s(0, y, z) = y$, $(s(1, y, z) = z$ and $s(x, y, z) = x$ otherwise.
- Decision circuits have one output gate which outputs $x \in \{0, 1\}$.

The complexity measure of a circuit C is the number of gates.

Algebraic vs branching circuits

- τ-circuits, as defined here, are **branching circuits**.
- **Algebraic** circuits have only function gates and do **not allow** selection gates, tests and case distinctions.

The theory of algebraic circuits is well developed and was initiated by L. Valiant (born 1948) in 1992.

L. G. Valiant. Why is Boolean complexity theory difficult?

Proceedings of the London Mathematical Society symposium on Boolean function complexity, pp. 8494, 1992.

Poizat's Theorem

Let $\mathbf{M}_\tau$ be a machine over $\mathfrak{A}$ with k tapes working in time bounded by $t(n)$, where $t(n) : \mathbb{N} \to \mathbb{N}$ is a function of the input size n with $t(n) \geq n$.

Theorem 2 (B. Poizat, 1995)
There is a recursive sequence of circuits $C_n(x_1, \ldots, x_n)$ of size $\leq t(n)^{k+1}$ and a polynomial $p(x) \in \mathbb{Z}[x]$ with $p(n) \in \mathbb{N}$ for $n \in \mathbb{N}$, such that

(i) for an input $a \in A^\star$ of size n both the machine $\mathbf{M}_\tau$ and $C_n(x_1, \ldots, x_n)$ gives the same result, and

(ii) the circuits C_n (encoded as a binary string) are uniformly constructed by a classical Turing machine from (an encoding of) $\mathbf{M}_\tau$ and n in time $p(n)$.

Proof sketch:

- The computation of $\mathbf{M}_\tau$ on input of size n can represented by an execution graph G_n with cycles.
- To get C_n unwind the execution graph of the machine $\mathbf{M}_\tau$ and turn it into a DAG.
- To compute the sequence C_n uniformly note that both G_n and hence C_n can be computed uniformy in polynomial time.

□

C-Sat

$\mathrm{C-SAT}_{\mathfrak{A}}$ is the following problem:

Input $w(C)a \in A^\star$, with $w(C)$ a binary encoding of a circuit $C(x, y)$.

Problem Is there $b \in A^\star$ such that

$$\langle \mathfrak{A}, a, b \rangle \models C(a, b)$$

Theorem 3
For every τ-structure $\mathfrak{A}$ the problem $\mathrm{C-SAT}_{\mathfrak{A}}$ is $\mathbf{NP}_{\mathfrak{A}}$-complete.

Proof: Clearly $\mathrm{C-SAT}_{\mathfrak{A}} \in \mathbf{NP}_{\mathfrak{A}}$.

Conversely, let $P \in \mathbf{NP}_{\mathfrak{A}}$. Using Theorem 2, there are τ-circuits C_n^P solving P for inputs of size n.
Let $w(C_n^P)$ be the binary string encoding C_n^P. □

Outline of the ESSLLI-course given by

J.A. Makowsky (Haifa) and K. Meer (Cottbus)

LECTURE 1 (JAM): Introduction INTRO (5 slides)
Turing machines over relational structures, NEWBSS, (19 slides)
Short quantifier elimination. SHORTQE (16 slides)

LECTURE 2 (JAM): Introduction to quantifier elimination QE (26 slides)
Fields, rings and other structures TABLE (incomplete, 20 slides)

LECTURE 3 (JAM): Computing with the reals: Removing order or multiplication;
Adding `Fortran`-libraries. FORTRAN (24 slides)
Comparing Poizat's Theorem with descriptive complexity FAGIN (20 slides)

LECTURE 4 (KM): Inside $\mathbf{NP}_{\mathbb{R}}$ and analogues to Ladner's Theorem, Meer-1 (149 slides)

LECTURE 5 (KM): PCP-Theorem over $\mathbb{R}$, Meer-2 (139 slides)

ADDITIONAL MATERIAL see next slide.

Additional material for the ESSLLI-course

LECTURE 6 (JAM): Quantifier elimination in algebraically closed fields, ACF-0 (21 slides)
By JAM after Kreisel and Krivine.

LECTURE 7 (JAM): $\mathbf{P}_G \neq \mathbf{NP}_G$ for all abelian groups. ABELIAN (18 slides)
By JAM After M. Prunescu

LECTURE 8 (JAM): $\mathbf{P}_G \neq \mathbf{NP}_G$ for all boolean algebras. BOOLEAN (23 slides)
By I. Bentov after M. Prunescu

LECTURE 9 (JAM): $\mathbf{P}_G \neq \mathbf{NP}_G$ for all real matrix rings. MATRIX (70 slides)
By N. Labai after A. Rybalov