DD2242 Seminars on TCS: Proof Complexity Jan 1, 2016

Lecture 1

Lecturer: Jakob Nordström *Scribe: Name of scribe here*

*PLEASE READ THE TEXT BELOW **CAREFULLY** EVEN IF YOU ARE A SUPER-EXPERIENCED LaTeX HACKER! There are some notes here about style and notational conventions which we want all notes to follow. Once you have read the text below, please **erase it** in your own version of the lecture notes.*

Reviewer's comment 1: *Reviewers can use the command* `\reviewercomment` *to add meta-comments/-questions, which can easily be toggled off by defining* `\DONOTINSERTCOMMENTS` *at the top of the main LaTeX file.*

1 Briefly about Scribing Lectures

Our goal is to produce readable lecture notes in not too formal but still correct English. The handwritten lecture notes posted on the webpage should hopefully serve as a good starting point, but we want complete and correct sentences. Do not throw away information in the lecture notes, but rather *elaborate* on it. Also, if you have any questions which are answered in class or at Piazza, make sure to include any helpful explanations, or examples, or whatever, it the scribe notes so that other course participants can also get access to this information.

Note that you should expect to put a significant amount of work into scribing the lecture notes—they are supposed to be the equivalent of a problem set. See the scribe notes already posted on the course webpage to get a sense of the (high) standard expected.

Write the notes in the present tense. "In today's lecture, we will show that…" "We next present the lemma that will be the key to this argument." Et cetera. There is no "I" (lecturer) in the notes, only a "we."

Make your own mark on the lecture notes! Please fill in the details missing in the proofs. Do not just state a dry, formal definition, but if possible try to elaborate on what the definition *means* and *why* it looks the way it does. If you heard some nice explanation during the lecture, please add it to help others understand as well. If you heard some bad explanation, then come up with a better one and add it to the notes! Make sure that the proofs make sense. Remember that, as we learned in the first lecture, a proof is something with the help of which one can efficiently verify a statement…;-) Thus, the proofs you write should not just state that things are true, but explain *why* they are true. Adding a dose of humour to the notes never hurts.

A remark on style: While we do not want to get too formal, we prefer writing "it is" instead of "it's" and "do not" instead of "don't" and such like. Also, we DO NOT want to use ALL CAPS FOR HIGHLIGHTING and **also NOT bold text**. Instead, highlight important concepts by adding *emphasis* (`\emph{emphasis}`) with the standard LaTeX commands.

For consistent formatting, please use the macros presented in Section 2 below. Start editing the LaTeX file below the line

```
%%%% BODY OF LECTURE NOTES GOES HERE --- DO EDIT BELOW %%%%
```

but please do not touch anything above that line. Start the lecture notes by the command

```
\lecture{<no>}{<date>}{<lecturer's name>}{<your name>}
```

where the date is given in the format "Sep 13, 2016". In addition to the lecture number and the date, add the name of the lecturer (typically Jakob Nordström or Ilario Bonacina) and your name as a scribe. Also, please update the `XX` in the file name `ScribeNotesLecXX.tex` to the (two-digit) number of the lecture (lectures 1–9 have a leading 0).

In Section 2, you find some hopefully useful information how to typeset the lecture notes. This info is also meant to help us standardize notation in between lectures so please try to stick to this notation. In Section 3, there might be some information specific for "your" lecture.

2 Section with Useful Info

2.1 Subsection with General Text Editing Info

You can label sections by using `\label{sec:info}` and `\label{sec:general}` and then refer to Section 2 and Section 2.1 by using commands `\refsec{sec:info}` and `\refsec{sec:general}` (these and other `\refXXX` commands for references to structures of type XXX are specific to the macro packages used for these notes).

Theorem 2.1 (Optional name of theorem and/or reference). *This is a theorem.*

When you cite a reference for a result (theorem, lemma, et cetera), you should do it in the following way, with just the reference, unless there is a specific reason to do otherwise.

Lemma 2.2 ([Coo71]). *This is a lemma with an added reference to the paper which supposedly established the lemma in question (coded here as* `\cite{Cook71ComplexityTheoremProving}`*).*

For a definition with both a name and a reference, format like this.

Definition 2.3 (Propositional proof system [CR79]). Use emphasis to highlight new concepts that are being defined. For instance, a *propositional proof system* is something that we completely fail to define here, even though the heading kind of suggests that this was what this definition was supposed to be doing.

Proposition 2.4. *This is a template proposition. A "lemma" is (usually) a purely auxiliary result that is used to establish some other, more important result. A "theorem" is usually a major result, and might be the main focus of of a lecture or might be something we are citing from some research article. The way we use the term, a "proposition" is, loosely speaking, something in between—more independent than a lemma, but not as important as a theorem (but different people use these terms slightly differently).*

Proof. We write `\begin{proof}` when we want to start a proof, in this case of Proposition 2.4 (where this reference was coded as `\refpr{pr:template-proposition}`).

Note the box at the end of the proof. Do not leave blank space between the last line of a proof and the following `\end{proof}` line, or you will get a bad line break. □

Just a "proof" is assumed to prove the latest theorem-like statement. If you are proving some previous statement, specify which in the following way.

Proof of Lemma 2.2. This is a proof of some previously stated result, here Lemma 2.2. We coded this as `\begin{proof}[Proof of \reflem{lem:template-lemma}]`.

As already noted, a QED box is added at the end of each proof. If you do not get it where you want, e.g., if the proof ends with a list, you can force placement of the box by issuing the command `\qedhere` (but this is not needed here). □

Observation 2.5. *This is an observation.*

Proof sketch. This is not a full proof, but a proof sketch.

We coded this as `\begin{proof}[Proof sketch]` □

Example 2.6. This is a dummy example.

Just to summarize, you can write:

- `\refth{th:template-theorem}` to refer to Theorem 2.1.
- `\reflem{lem:template-lemma}` to refer to Lemma 2.2.
- `\refpr{pr:template-proposition}` to refer to Proposition 2.4.

- `\refobs{obs:template-observation}` to refer to Observation 2.5.
- `\refdef{def:template-def}` to refer to Definition 2.3.
- `\refex{ex:dummy}` to refer to Example 2.6.

The list above was coded by writing `\begin{itemize}` to start the list, then `\item` before each new item, and finally `\end{itemize}`. You can get a numbered list by writing `enumerate` instead of `itemize`.

2.2 Subsection with General Math Editing Info

Many typesetting commands with parantheses are available in three versions: `\command` with small parantheses, `\Command` with somewhat larger ones, and `\COMMAND` with very large parantheses (use only in displayed math).

For instance:

- `$\bigoh{n}$` yields $\mathrm{O}(n)$ and `$\Bigoh{n^k}$` yields $\mathrm{O}\big(n^k\big)$ with slightly larger parantheses.
- `$\bigomega{\log n}$` yields $\Omega(\log n)$ and `$\Bigomega{\frac{1}{n}}$` yields $\Omega\big(\frac{1}{n}\big)$.
- `$\bigtheta{n}$` becomes $\Theta(n)$ and `$\Bigtheta{\frac{n}{\log n}}$` becoms $\Theta\big(\frac{n}{\log n}\big)$.
- `$\abs{x}$` becomes $|x|$ and `$\Abs{x^y}$` becomes $\big|x^y\big|$.

Other commands that work the same way (i.e., have versions with parantheses of adaptable sizes) are:

1. `\set` denoting a set: $\{1, 2, 3, \ldots\}$.
2. `\setdescr` denoting a set described by some parameter (use `\Setdescr` for this example to get a nicer look): $\big\{2^i \,\big|\, i \in \mathbb{N}^+\big\}$.
3. `\setsize` denoting the size of a set: $|V(G)| = n$.
4. `\maxofset` and `\minofset` taking the maximum and minimum over a set as in, for instance, $\min\{Sp(\pi) \mid \pi : F \vdash \bot\}$.
5. `\maxofexpr` and `\minofexpr` taking the maximum/minimum over an "expression" as in, for instance, $\max_{D \in \pi}\{W(D)\}$.

You can use `\begin{equation*}` to display mathematics without a getting a numbered equation (and *not* `$$`), like this:

$$x^2 - x = 0 \ .$$

Since the spacing before periods and commas in displayed math is a science in its own right, we have macros `\eqperiod` and `\eqcomma` to get the spacing we like.

By default, always display mathematics *with* a number, though—even if you do not need to refer to the equation, someone else who is reading the notes might need to do so. To display mathematics with a number, use `\begin{equation}`:

$$x + \overline{x} - 1 = 0 \tag{2.1}$$

You can use `\refeq{eq:example-equation}` to refer to this equation, which will give a reference that looks like (2.1).

2.3 Subsection with Proof Complexity Notation

Sometimes we want to specify in a subscript the proof system in question when discussing, e.g., proof complexity measures. In such a case we write `\resnot` to denote general resolution $\mathcal{R}$, `\treeresnot` to denote tree-like resolution $\mathcal{T}$, `\cpnot` to denote cutting planes $\mathcal{CP}$, `\pcnot` to denote polynomial calculus $\mathcal{PC}$, et cetera (where the suffix `not` is a mnemonic for "notation").

For size of refutations we write

- $S(\pi)$ (`\sizeofarg{\pi}`).
- $S_{\mathcal{PC}}(F \vdash \bot)$ (`\sizeref[\pcnot]{F}`).
- $S_{\mathcal{CP}}\big(PHP_n^m \vdash \bot\big)$ (`\Sizeref[\cpnot]{\phpnot{m}{n}}`) (note that the initial capital letter yields slightly larger parentheses).

To denote the length of refuting F in resolution we code `$\lengthref[\resnot]{F}$` which looks like $L_{\mathcal{R}}(F \vdash \bot)$. The length of refuting a CNF formula F in tree-like resolution is coded as `$\lengthref[\treeresnot]{F}$` which looks like $L_{\mathcal{T}}(F \vdash \bot)$.

The width of refuting F in resolution is written `$\widthref[\resnot]{F}$` and looks like $W_{\mathcal{R}}(F \vdash \bot)$, the clause space is written `$\clspaceref[\resnot]{F}$` and looks like $Sp_{\mathcal{R}}(F \vdash \bot)$, and the total space `$\totspaceref[\resnot]{F}$` looks like $TotSp_{\mathcal{R}}(F \vdash \bot)$.

When the proof system under discussion is perfectly clear from context, however, we might skip the optional argument in brackets and just write `$\lengththref{F}$` for $L(F \vdash \bot)$ instead, for instance.

When measuring a concrete proof π, we use the macros `$\lengthofarg{\pi}$`, i.e., $L(\pi)$, `$\sizeofarg{\pi}$`, i.e., $S(\pi)$, and `$\widthofarg{\pi}$`, i.e., $W(\pi)$, with the extra suffix `arg` appended because of unfortunate naming collisions with standard LaTeX packages, and `$\clspaceof{\pi}$`, i.e., $Sp(\pi)$, and `$\totspaceof{\pi}$`, i.e., $TotSp(\pi)$.

2.4 Some Symbols

Here is a list of some symbols (please tell me afterwards if there was something you really missed here and I will add it):

- $\sum_{i=1}^{n} a_n$ coded as `$\sum_{i=1}^{n} a_{n}$`
- $\prod_{i=1}^{n} p_i$ coded as `$\prod_{i=1}^{n} p_{i}$`
- $\to \infty$ coded as `$\to \infty$`
- $\models$ coded as `$\impl$` and $\not\models$ coded as `$\nimpl$`
- Implication arrow is `\limpl` ($\limpl$).
- Fatter arrows, which can be used in proofs, for instance, are coded as `$\Leftarrow$` ($\Leftarrow$) and `$\Rightarrow$` ($\Rightarrow$).
- Negation with a hook $\lnot x$ is coded `$\lnot x$`. Negation of literals with overline $\olnot{x}$ is coded as `$\olnot{x}$`.

2.5 Inference Rules and Derivations

Please typeset any inference rules and derivations as follows below, using the LaTeX tools by Sam Buss.

Just one resolution inference can be done like this:

$$\frac{C_1 \qquad C_2}{C_3} \tag{2.2}$$

A couple of resolution inferences "in parallel" can be coded like this:

$$\frac{C_1 \qquad C_2}{C_3} \tag{2.3}$$

$$\frac{D_1 \qquad D_2}{D_3} \tag{2.4}$$

$$\frac{E_1 \qquad E_2}{E_3} \tag{2.5}$$

And finally, to code a derivation with more structure, do like this:

$$\dfrac{\dfrac{\dfrac{C_1 \qquad C_2}{C_3} \qquad C_4}{C_5} \qquad C_6}{\begin{array}{c} C_7 \\ \vdots \\ \dfrac{C_{n-2} \qquad C_{n-1}}{C_n} \end{array}} \tag{2.6}$$

3 Section with Info for Your Lecture

3.1 Notation (Mainly for Lectures 7-12)

3.1.1 Generic

`\formf` F: CNF formula to be refuted
`\proofstd` π: notation for proofs/refutations.
`\refof{\proofstd}{\formf}` $\pi : F \vdash \bot$: refutation (in proof system under consideration)
`\formf \impl \clc` $F \models C$: formula F implies clause C
`\R, \Z, \Nplus` $\mathbb{R}, \mathbb{Z}, \mathbb{N}^+$: reals, integers, and positive integers
`\F` $\mathbb{F}$: generic field
`\F_2 = \set{0,1}` $\mathbb{F}_2 = \{0,1\}$: Galois field with two elements
`\set{a,b,c}` $\{a,b,c\}$: a set with elements a, b, c
`\setdescr{m}{\exists n \in \Nplus\, m=2n}` $\{m \mid \exists n \in \mathbb{N}^+ \, m = 2n\}$: set of all even positive numbers (just to demonstrate macro)
Use `\veca`, `\vecb`, `\vecu`, `\vecv` et cetera to denote vectors (or sets of variables) $\mathbf{a}$, $\mathbf{b}$, $\mathbf{u}$, $\mathbf{v}$, et cetera.

3.1.2 d-DNF Resolution

`\resk` d-DNF resolution or, for short, `\reskshort` $\mathcal{R}(d)$: the proof system performing resolution/cut over d-DNF formulas
`\termsize` d: the "d" or "k" in d-DNF resolution and in d-DNF formulas.
Just do `\renewcommand{\termsize}{k}` to get a k if it makes it easier to scribe the notes.
`\dnfstd` or `\dnfphi` φ: generic DNF formula
`\dnfalt` or `\dnfpsi` ψ: second generic DNF formula
`\termt` t: generic term in a DNF formula
`\clc` C and `\cld` D: generic clauses
`\lita` a and `\litb` b: generic literals
Just do `\renewcommand{\lita}{\ell}` to get an ℓ for literals if it makes it easier to scribe the notes.
`\xorl` L: XOR clause. Please use the macro `\xorl` as we will redefine this notation later to avoid collisions.
`\vars{\formf}` $Vars(F)$: set for variables of a formula.
`\tsupp{\termt}` $supp(t)$: Support of term in the sense of [Ale11].

`\tsuppstar{\termt}` $supp^*(t)$: "Extended support" of term in the sense of [Ale11].
`\covnum{\dnfstd}` $cov(\varphi)$: covering number of a DNF formula
`\treet` T: binary tree
`\dnfheight{\dnfstd}` $h(\varphi)$: min height of a tree strongly representing the DNF formula φ
`\tbranch{\treet}{b}` $br_b(T)$: b-branches in representing tree
Let us also use the notation $br(T) = br_0(T) \cup br_1(T)$
`treeheight{\treet}` $d(T)$: height of a binary tree T
`\restr` ρ: restriction, i.e., partial assignment
`\restr = \set{x \mapsto 1, y \mapsto 0}` $\rho = \{x \mapsto 1, y \mapsto 0\}$: explicit representation of restriction
`\restrict{\dnfstd}{\restr}` $\varphi\restriction_\rho$: DNF formula restricted by a partial assignment
`\domainof{\restr}` $\mathrm{dom}(\rho)$: domain of a partial assignment
`\negclause{\restr}` $C_{\neg\rho}$: unique maximal clause falsified by restriction ρ

3.1.3 Probability Theory

`\distrd` $\mathcal{D}$: probability distribution (in our d-DNF resolution lectures this distribution is on partial assignments)
`\prob{\eventa}` $\Pr[\mathcal{A}]$: probability of the event $\mathcal{A}$
`\prob[\sample{\restr}{\distrd}]{\eventa}` $\Pr_{\rho\sim\mathcal{D}}[\mathcal{A}]$: probability of the event $\mathcal{A}$ over restrictions ρ sampled from distribution $\mathcal{D}$

3.1.4 Graphs and PHP Formulas

`\graphg` G: notation for generic graph
`\gleft` L: left bipartition in a bipartite graph G ("pigeons" in Lecture 9)
`\gright` R: right bipartition in a bipartite graph G ("holes")
`\edgese` E: edge set of a graph
`\varnumberofpigeons` Δn: the default number of pigeons in this lecture
`\varratiopigeonholes` Δ: ratio between the number of pigeons and holes
`\nholes` n: number of holes
`\phpnot{\npigeons}{\nholes}` PHP_n^m: pigeonhole principle formula
`\varphp{i}{j}` $x_{i,j}$: the Boolean variables used to express the pigeonhole principle
`\pigeonclause{i}` P^i: clause in PHP_n^m saying that pigeon i flies somewhere
`\holeclause{i}{i'}{j}` $H_j^{i,i'}$: clause in PHP_n^m saying that i, i' do not both fly to hole j
`\graphphpnot[\graphg]` $PHP(G)$: pigeonhole principle on a graph G (note the square brackets)
`\gexpsize` r: the size of the sets up to which we have expansion
`\gexpfactor` c: expansion factor for a bipartite boundary expander
`\boundaryexpander{\gexpsize}{\gexpfactor}` (r, c)-boundary expander: boundary expander with default parameters
`\Neighbours{\gleft'}` $N\big(L'\big)$: neighbours on the right of left vertex set L' (note that writing `\Neighbours` instead of `\neighbours` gives slightly larger parentheses, so that the content inside fits nicely).
`\Uniqueneighbours{\gleft'}` $\partial\big(L'\big)$: unique neighbours on the right of left vertex set L' (again capital "U" makes parentheses slightly larger)

3.2 References (Mainly for Lectures 7-12)

You should probably only need a subset of these.

Definition of d-DNF resolution [Kra01]
Segerlind-Buss-Impagliazzo paper on which Lectures 8-10 are mainly based: [SBI04]
Alekhnovich 3-CNF lower bound (Lectures 11-12) [Ale11]

[BW01]: reference for Theorem 2 in the handwritten notes for Lecture 9.

[Raz15] Razborov's improvement of the lower bound for the pigeonhole principle

Below follow some general references for results on PHP formulas and random k-CNF formulas that were mentioned in Lecture 7 and sometimes also in later lectures.

Haken's PHP paper [Hak85]

Random k-CNFs hard for resolution [CS88]

Maciel-Pitassi-Woods quasipolynomial size proofs [MPW02]

Ran Raz on WPHP [Raz04a]

Alexander Razborov follow-ups on WPHP [Raz03, Raz04b]

PHP upper bound for Frege [Bus87]

PHP lower bound for bounded-depth Frege [KPW95] and [PBI93] independently (but they had a joint conference version [BIK$^+$92]).

References

[Ale11] Michael Alekhnovich. Lower bounds for k-DNF resolution on random 3-CNFs. *Computational Complexity*, 20(4):597–614, December 2011. Preliminary version in *STOC '05*.

[BIK$^+$92] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, Pavel Pudlák, and Alan R. Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC '92)*, pages 200–220, May 1992.

[Bus87] Samuel R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52(4):916–927, 1987.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[CR79] Stephen A. Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[KPW95] Jan Krajíček, Pavel Pudlák, and Alan R. Woods. An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1):15–40, 1995. Preliminary version in *STOC '92*.

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-3):123–140, 2001.

[MPW02] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *Journal of Computer and System Sciences*, 64(4):843–872, 2002. Preliminary version in *STOC '00*.

[PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993. Preliminary version in *STOC '92*.

[Raz03] Alexander A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theoretical Computer Science*, 1(303):233–243, June 2003.

[Raz04a] Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *Journal of the ACM*, 51(2):115–138, March 2004. Preliminary version in *STOC '02*.

[Raz04b] Alexander A. Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69(1):3–27, August 2004. Preliminary version in *CCC '02*.

[Raz15] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472, March 2015.

[SBI04] Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004. Preliminary version in *FOCS '02*.