EXAMENSARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Short Proofs of Finite Instances of Valid Sentences

av

Marko Djordjevic

No 4 – 1995

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Stockholms universitet
Reproenheten 1995

Acknowledgements

I wish to thank my supervisor Gunnar Stålmarck from whom I got the idea of writing this paper and who has given me much support and advice. I also thank Dag Prawitz and Filip Widebäck for valuable comments at a seminar, and Per-Erik Malmnäs for putting me into contact with G. Stålmarck.

Introduction

The study of proof lengths of logical calculi, or proof systems as we will call them, is related to the subject of computational complexity by the following result of Cook and Reckhow [4]:

NP is closed under complementation if and only if there is a proof system and a polynomial p such that every classical tautology is provable in this proof system with a proof which size is bounded by p, as a function of the length of the tautology.

A proof system that satisfies the above condition is said to be polynomially bounded.

It is conjectured that the answer is negative, that is, NP is not closed under complementation or equivalently there is no polynomially bounded proof system. Although this (or the converse) has not yet been proved some progress has been made. With respect to the notion of p-simulation a hiearchy of proof systems has appeared. We say that a proof system S_2 p-simulates a proof system S_1 if any proof of a tautology A in the system S_1 can be transformed within polynomial time into a proof of A in the system S_2. See [4], [5] or [13] for formal definitions. From the definition it follows that if a proof system S_2 p-simulates a polynomially bounded proof system S_1 then also S_2 is polynomially bounded. Also, if S is a proof system which is not polynomially bounded then no proof system that is p-simulated by S can be polynomially bounded. Hence, two proof systems that p-simulate each other can be regarded as equivalent with respect to the property of being polynomially bounded. A number of results comparing different proof systems have been obtained (see [10], [13]). For example, we know that $G_{d.a.g.}$ (Cut-free Gentzen systems with proofs represented as directed asyclic graphs) p-simulates G_{tree} (Cut-free Gentzen systems with proofs represented as trees) but not the other way around. We also know that Resolution and $G_{d.a.g.}$ p-simulates each other. Moreover, it is known that G_{tree} cannot be p-simulated by truth tables and truth tables cannot be p-simulated by G_{tree}, hence truth tables and G_{tree} are incomparable. All these proof systems have been proved not to be polynomially bounded. The truth table method is easily seen to be exponential. It was proved by Haken [6] that the size of minimal Resolution refutations of the formulas $\neg PHP_2, \neg PHP_3, \neg PHP_4, \ldots$ grows exponentially with respect to the length of PHP_n. The formulas PHP_n are translations into propositional logic of the assertion (the pigeonhole principle) that there is no one-one map from the set $\{1,2,\ldots,n\}$ into the set $\{1,2,\ldots,n-1\}$. It follows by p-simulation that the formulas PHP_n have exponential proofs also in G_{tree} and $G_{d.a.g.}$. With Frege systems and Natural Deduction systems we mean systems that can be fit into the definitions of Cook and Reckhow [4], so Frege systems correspond to what are usually called Hilbert systems, and in Natural deduction systems (in the sense of [4]) proofs are not presented as ordinary trees (as in Prawitz [9]) but rather as some structure where we need not derive the same formula several times in a proof. Frege systems and Natural Deduction systems p-simulate each other and p-simulate all the above named proof systems. We do not yet know if Frege systems and Natural Deduction systems are polynomially bounded. Ajtai [1] has shown that for a restricted form of Frege systems, called bounded depth Frege systems, where a restriction on the formulas occuring in proofs is placed there are no short (polynomially bounded) proofs of the formulas PHP_n. On the other hand the PHP_n formulas have short proofs in Frege systems without this restriction, as was shown by Buss [2]. In this paper we will show that for a certain Natural Deduction proof system, called Reductio, all finite instances of valid first order sentences (in a language with the quantifier $\forall$ only) have polynomially bounded proofs. If D is a finite domain and A is a first order sentence then the finite instance of A with respect to D is a propositional formula expressing A when it's variables range over D. Formal definitions, which essentially involve replacing occurences of $\forall$ by finite conjunctions, are given later. What will actually be proved is that any finite instance B of a valid sentence A has a Reductio proof which does not involve more than k assumptions simultaneously where k depends only on A. Then by a result of Stålmarck [11] it will follow that there is a polynomial p of degree k+1 such that the length of this proof is bounded by $p(|B|)$, where $|B|$ is the length of B. This will imply that for all proof systems which p-simulate Reductio, such as Frege systems, bounded depth Frege systems and Natural Deduction systems, finite instances of valid first order sentences have short proofs. Hence, if one wishes to prove that a Frege system, bounded depth Frege system or Natural Deduction system is not polynomially bounded then one cannot hope to find a sequence $A_1, A_2, A_3, \ldots$ of witnesses by letting A_n (n=1,2,3,...) be finite instances of a valid first order sentence. Before stating our main result rigorously we introduce our first order language and the Reductio proof system.

The Reductio proof system

The first order language that we will use consists of the following symbols:

One propositional constant: $\bot$.
A set of propositional symbols.
For every k=1,2,3,... a set of k-place predicate symbols.
Individual constants: 1,2,3,4,...
An infinite set of parameters, denoted by t,t',s,s' with or without indexes.
An infinite set of variables, denoted by x with or without indexes.
Logical connectives $\neg,\&,\vee,\rightarrow$.
One quantifier $\forall$.
Parantheses: (,).

Constants and parameters are also called terms. Terms are often denoted by t,t',s,s' with or without indexes.

Formulas are defined by:
$\bot$ is an atomic formula and all propositional symbols are *atomic formulas*.
If A is a k-place predicate symbol and $t_1,...,t_k$ are terms then $At_1...t_k$ is an *atomic formula*.
Atomic formulas are formulas.
If A and B are formulas then $\neg A$, $(A\&B)$, $(A\vee B)$, $(A\rightarrow B)$ and $(A\leftrightarrow B)$ are formulas.
If A is a formula and t is a term then $\forall xA[t/x]$ is a formula (where $A[t/x]$ is the expression obtained by substituting every occurence of t in A by x).

Parantheses will often be omitted, so that we write $A\&B, A\vee B, A\rightarrow B$ instead of $(A\&B),(A\vee B),(A\rightarrow B)$.
We will also use the abbreviation $A_1\&...\&A_k$ for $(A_1\&(...\&(A_{k-1}\&A_k)...))$.

The set of all formulas is denoted by F. If $\Delta \subseteq \mathsf{F}$ then we call Δ a formula set.

If $t_1,...,t_p$ are different terms and $s_1,...,s_m$ are terms then $A[t_1/s_1,...,t_m/s_m]$ is the expression obtained by substituting every occurence of t_i in A by s_i for i=1,...,m.

If a parameter/variable/individual constant occurs in a formula A then we say that A has a parameter/variable/individual constant. If no parameters/variables/individual constants occur in a formula A we say that A has no parameters/variables/individual constants.
If Δ is a formula set then the phrases 'Δ has a parameter/variable/individual constant' and 'Δ has no parameters/variables/individual constants' are to be understood in the obvious way.
A formula that has no parameters is called a *closed formula*.
A formula that has no parameters and no variables is called a *propositional formula*. (Hence $\forall$ cannot occur in a propositional formula)
A formula that has no individual parameters and no individual constants is called a *sentence*.
The *length* $|A|$ of a formula is the number of occurences in A of $\neg$, $\forall$ and logical connectives.

Subformulas are defined by:
A is a subformula of A.
If B is a subformula of A then B is a subformula of $\neg A$.
If C is a subformula of A or B then C is a subformula of $A\&B, A\vee B$ and $A\rightarrow B$.
If A is a subformula of B and t is a term then A is a subformula of $\forall xB[t/x]$.

For every $n\geq 1$ a function D_n: $\mathsf{F} \rightarrow \mathsf{F}$ is defined inductively on the definition of a formula:
$D_n(A) = A$ if A is atomic.
$D_n(\neg A) = \neg D_n(A)$.
$D_n(A*B) = D_n(A)*D_n(B)$ where $*$ is a connective.
$D_n(\forall xA[t/x]) = D_n(A)$ if t does not occur in A.
$D_n(\forall xA[t/x]) = D_n(A)[t/1]\&D_n(A)[t/2]\&...\&D_n(A)[t/n]$ if t occurs in A.

We say that $D_n(A)$ is the *finite instance* of A with respect to the domain $\{1,2,...,n\}$.
It is not difficult to prove by induction on the complexity of A that if t and t' are terms and $t\notin\{1,...,n\}$ then $D_n(A[t/t']) = D_n(A)[t/t']$.

We now present the Reductio proof system for first order logic which we will be working with. The propositional fragment of this system is due to Stålmarck and is studied in [11] and [12]. The rules of the Reductio proof system are divided into simple rules where no assumption is discharged, and the Reductio rule where assumptions are dicharged.

Simple rules

introduction rules: $\dfrac{A \quad B}{A\&B}\ \&I \qquad \dfrac{A}{A\vee B}\ \vee I1 \qquad \dfrac{B}{A\vee B}\ \vee I2 \qquad \dfrac{\neg A}{A\to B}\ \to I1 \qquad \dfrac{B}{A\to B}\ \to I2$

$\dfrac{A \quad \neg A}{\bot}\ \bot I \qquad \dfrac{A}{\forall x A[t/x]}$ $\forall$I with respect to t, where t is a parameter.

elimination rules: $\dfrac{A\&B}{A}\ \&E1 \qquad \dfrac{A\&B}{B}\ \&E2 \qquad \dfrac{A\vee B \quad \neg A}{B}\ \vee E1 \qquad \dfrac{A\vee B \quad \neg B}{A}\ \vee E2$

$\dfrac{A\to B \quad A}{B}\ \to E1 \qquad \dfrac{A\to B \quad \neg B}{\neg A}\ \to E2 \qquad \dfrac{\forall x A}{A[x/t]}$ $\forall$E where t is a term.

Let Δ be a finite formula set.

If $A_1,\ldots,A_k \in \Delta$ (k=1,2) and $\dfrac{A_1 \ldots A_k}{A}$ is an *instance* of a simple rule then we say that $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an *instance* (or *application*) of a simple rule. If $\dfrac{A_1 \ldots A_k}{A}$ is an instance of $\forall$I with respect to t then we say that $\dfrac{\Delta}{\Delta\cup\{A\}}$ depends on t. If $\Delta\cup\{A\} = \Delta$ then we say that $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an *improper instance* (or *improper application*) of a simple rule, otherwise we call $\dfrac{\Delta}{\Delta\cup\{A\}}$ a *proper instance* (or *proper application*) of a simple rule.

Now we simultaneously give the definition of *Reductio derivations* and of the *Reductio rule*.

(i) Every finite formula set Δ is a Reductio derivation with *premise set* Δ and *conclusion set* Δ.

(ii) If Π is a Reductio derivation with premise set Δ and conclusion set Δ' and and $\dfrac{\Delta'}{\Delta'\cup\{A\}}$ is a simple rule which does not depend on a parameter that occurs in Δ then $\dfrac{\Pi}{\Delta'\cup\{A\}}$ is a Reductio derivation with premise set Δ and conclusion set $\Delta'\cup\{A\}$.

(iii) If Π_1, Π_2, Π_3 are Reductio derivations with premise sets Δ, $\Delta_1\cup\{A\}$, $\Delta_1\cup\{\neg A\}$ respectively and conclusion sets $\Delta_1, \Delta_2, \Delta_3$ respectively and $\bot \in \Delta_i$ (i=$\overset{2}{\cancel{1}}$ or i=$\overset{3}{\cancel{2}}$) then $\dfrac{\dfrac{\Pi_1}{\Pi_2 \quad \Pi_3}}{\Delta_j}$ (j=$\overset{2}{\cancel{1}}$ or j=$\overset{3}{\cancel{2}}$, j≠i) is a Reductio derivation with premise set Δ_1 and conclusion set Δ_j.

We also say that $\dfrac{\dfrac{\Pi_1}{\Pi_2 \quad \Pi_3}}{\Delta_j}$ is an instance of the Reductio rule, that is, from the given premises we may conclude Δ_j which is now free from the assumptions A and $\neg$A.

Π is a Reductio derivation only if it can be obtained by succesive applications of (i),(ii) and (iii). From now on we will often just say derivation instead of Reductio derivation.

If a Reductio derivation Π has premise set Δ and conclusion set Δ' then we may write $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$ (instead of Π) to emphasize this. (Observe that for example $\dfrac{\begin{array}{c}\Delta\\ \Pi\end{array}}{\Delta'}$ and $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$ are different derivations). In this notation the above

Reductio derivation (and instance of the reductio rule) becomes

$$\dfrac{\dfrac{\begin{array}{c}\Delta\\ \Pi_1\\ \Delta_1\end{array}}{\begin{array}{cc}\Delta_1\cup\{A\} & \Delta_1\cup\{\neg A\}\\ \Pi_2 & \Pi_3\\ \Delta_2 & \Delta_3\end{array}}}{\Delta_3} \quad \text{if } \bot\in\Delta_2 \text{ or } \quad \dfrac{\dfrac{\begin{array}{c}\Delta\\ \Pi_1\\ \Delta_1\end{array}}{\begin{array}{cc}\Delta_1\cup\{A\} & \Delta_1\cup\{\neg A\}\\ \Pi_2 & \Pi_3\\ \Delta_2 & \Delta_3\end{array}}}{\Delta_2} \quad \text{if } \bot\in\Delta_3.$$

If $\Delta_1 = \Delta_3$ (or $\Delta_1=\Delta_2$) then we say that the instance of the Reductio rule above is *improper*, otherwise it is said to be *proper*.
Let Π be a derivation.

If Π has premise set Δ and conclusion set Δ' and $\bot\in\Delta'$ then we say that Π is a *refutation of* Δ and we write $\begin{array}{c}\Delta\\ \Pi\\ \bot\end{array}$.

We say that Π is a *proof* of A if Π is a refutation of $\{\neg A\}$.
We say that a formula A is in Π (or that A is a formula of Π) if $A\in\Delta$ for some formula set Δ that occurs in Π.
If every formula in Π is a propositional formula then we say that Π is a *propositional derivation* , and if in addition Π is a proof (of some formula A) then we say that Π is a *propositional proof.* If Π is a propositional proof of A then of course A must be propositional formula.
We define $F(\Pi)$ to be the set of formulas in Π.

For every formula A we define *sub*(A) by $B\in$ *sub*(A) if and only if (1) B is a subformula of A or (2) $B=\neg C$ and C is a subformula of A or (3) $B = \bot$.

If Π is a proof of A and $F(\Pi) \subseteq$ *sub*(A) then we say that Π has the *subformula property* or that Π is a *subformula proof* of A.

The *depth* $d(\Pi)$ of a derivation Π is defined inductively by:
$d(\Delta) = 0$.
$d(\dfrac{\Pi}{\Delta}) = d(\Pi)$.
$d\left(\dfrac{\dfrac{\Pi_1}{\Pi_2 \quad \Pi_3}}{\Delta_3}\right) = \max(d(\Pi_1), d(\Pi_2)+1, d(\Pi_3)+1)$.
The *length* $|\Pi|$ of a derivation Π is defined to be the number of occurences of formula sets in Π.

It is easy to verify soundness of Reductio. Completeness is proved in the last section of this paper. To prove it we use the relationship between Reductio and the system KEQ of Mondadori and D'Agostino (see [5], [7], [8]) and show that for any KEQ-proof T of a formula A there is a Reductio proof Π of A such that every formula in Π is a subformula of a formula in T. KEQ is complete, and remains complete also when we impose the restriction that a proof of a formula A may only contain formulas that belong to sub(A). Hence, the same is true for Reductio, or in other words, for every valid sentence A there is a Reductio proof Π of A with the subformula property.

Short Reductio proofs of finite instances of valid sentences

Now we state the main result.

Theorem. If A is a valid sentence then there is a polynomial p and subformula proofs $\Pi_1, \Pi_2, \Pi_3, \ldots$ of $D_1(A), D_2(A), D_3(A), \ldots$ respectively so that $|\Pi_n| \leq p(|D_n(A)|)$ for n=1,2,3,... .

The theorem will follow from the these two propositions:

Proposition 1. If Π is a subformula proof of a sentence A then there are propositional subformula proofs $\Pi_1, \Pi_2, \Pi_3, \ldots$ of $D_1(A), D_2(A), D_3(A), \ldots$ such that $d(\Pi_n) = d(\Pi)$ for n=1,2,3,... .

Proposition 2.(Stålmarck) For every $n \geq 1$ there is a polynomial p_n of degree n such that:
If Π is a propositional subformula proof of a formula A and $d(\Pi) \leq n$ ($n \geq 1$) and only proper applications of rules occur in Π then $|\Pi| \leq p_{n+1}(|A|)$.

Indeed, if A is a valid sentence then by the argument in the previous section there is a subformula proof Π of A and by proposition 1 there are propositional subformula proofs $\Pi_1, \Pi_2, \Pi_3, \ldots$ of $D_1(A), D_2(A), D_3(A), \ldots$ such that $d(\Pi_n) = d(\Pi)$ for n=1,2,3,... . Since applications of improper rules can always be removed without increasing the depth we may assume that only applications of proper rules occur in $\Pi_1, \Pi_2, \Pi_3, \ldots$. Proposition 2 now implies that there is a polynomial p of degree $d(\Pi)+1$ such that $|\Pi_n| \leq p(|D_n(A)|)$ for n=1,2,3,... .
We first give the proof of the second proposition.

Proof of proposition 2. We will prove the following assertion which implies the proposition.
For every $n \geq 1$ there is a polynomial p_n of degree n such that:
If Π is a propositional derivation with $d(\Pi) \leq n$ and only proper applications of rules occur in Π and $F(\Pi) \subseteq$ *sub*(A) for some propositional formula A, then $|\Pi| \leq p_{n+1}(|A|)$.
The proof is by induction on n. First let n=0.
Let Π be a propositional derivation in which only proper applications of rules occur and assume $d(\Pi) = 0$, $F(\Pi) \subseteq$ *sub*(A). Then Π must look like

$$\begin{array}{c} \Delta_1 \\ \hline \Delta_2 \\ \vdots \\ \vdots \\ \Delta_{k-1} \\ \hline \Delta_k \end{array}$$

It is easy to see that *sub*(A) contains at most $3|A|+1$formulas (remember that A is a propositional formula). Since Δ_{i+1} contains one more formula than Δ_i and Δ_k contains at most $3|A|+1$ formulas we must have $k \leq 3|A|+1$ which implies $|\Pi| \leq 3|A|+1$. Set $p_1(x)=3x+1$.
Now suppose that the assertion is true for $m \leq n-1$ and let Π be a propositional derivation in which only proper applications of rules occur and assume $d(\Pi) = n$, $F(\Pi) \subseteq$ *sub*(A). Then Π looks like

$$\begin{array}{c}
\Pi_1 \\ \hline
\Pi'_1 \qquad\qquad \Pi''_1 \\ \hline
\Pi_2 \\ \hline
\Pi'_2 \qquad\qquad \Pi''_2 \\ \hline
\vdots \\
\vdots \\
\Pi_k \\ \hline
\Pi'_k \qquad\qquad \Pi''_k \\ \hline
\Pi_{k+1}
\end{array}$$

where $d(\Pi_i) \le n\text{-}1$, $d(\Pi'_i) \le n\text{-}1$, $d(\Pi''_i) \le n\text{-}1$, $i=1,\dots,k$ and $d(\Pi_{k+1}) \le n\text{-}1$.
Since the derivations Π_i, Π'_i, Π'_i ,$i=1,\dots,k$ and Π_{k+1} satisfy the conditions of the assertion the induction hypothesis gives $|\Pi_i| \le p_{n-1}(|A|)$, $|\Pi'_i| \le p_{n-1}(|A|)$, $|\Pi''_i| \le p_{n-1}(|A|)$ $i=1,\dots,k$ and $|\Pi_{k+1}| \le p_{n-1}(|A|)$ where p_{n-1} is a polynomial of degree n-1. Since the instances of the reductio rule are proper we must also have $k \le 3|A|+1$. Hence we get $|\Pi| \le (3|A|+1)\cdot 3\cdot p_{n-1}(|A|) + p_{n-1}(|A|)$. Set $p_n(x) = (9x+3)\cdot p_{n-1}(x) + p_{n-1}(x)$. This completes the induction step.

We now aim at proving proposition 1. In order to avoid dealing with some uninteresting details an extension of the reductio proof system will be introduced in the following maner:

Extended rules are defined by:
(1) All simple rules are extended rules.
(2) For $k \ge 3$ $\dfrac{A_1 \dots A_k}{A_1\&\dots\&A_k}$ and $\dfrac{A_1\&\dots\&A_k}{A_i}$ $i=1,\dots,k$ are extended rules.
If $\Delta \subseteq \Delta'$ are finite formula sets and for every $A \in \Delta'$ either $A \in \Delta$ or we can find formulas $A_1,\dots,A_k \in \Delta$ such that $\dfrac{A_1 \dots A_k}{A}$ is an instance of an extended rule then we say that $\dfrac{\Delta}{\Delta'}$ is an instance of an extended rule.
If $\dfrac{A_1 \dots A_k}{A}$ is $\forall I$ with respect to t for some $A \in \Delta'$ then we say that $\dfrac{\Delta}{\Delta'}$ *depends* on t.

Extended derivations are defined exactly as reductio derivations but with 'simple rule' replaced by 'extended rule'. All notions (except length of proof) that where introduced for derivations carry over to extended derivations in the obvious way. As with derivations we may denote an extended derivation Π with premise set Δ and conclusion set Δ' by $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$. In particular, if $\begin{array}{c}\{\neg A\}\\ \Pi\\ \perp\end{array}$ is an extended derivation then we say that Π is an *extended proof* of A. If Π is an extended derivation in which an instance of an extended rule depending on a parameter t occurs then we say that Π *depends* on t.
It is clear that every derivation is also an extended derivation.
Every instance of an extended rule $\dfrac{\Delta}{\Delta'}$ can be replaced by a finite sequence

$$\begin{array}{c}
\Delta_1 \\ \hline
\Delta_2 \\
\vdots \\
\vdots \\
\Delta_{k-1} \\ \hline
\Delta_k
\end{array}$$

where $\Delta=\Delta_1$, $\Delta'=\Delta_k$ and $\dfrac{\Delta_i}{\Delta_{i+1}}$ is an instance of a simple rule for $i=1,\dots,k$.

This means that every extended derivation $\begin{matrix}\Delta\\ \Pi\\ \Delta'\end{matrix}$ can be transformed into a derivation $\begin{matrix}\Delta\\ \Pi'\\ \Delta'\end{matrix}$ with depth preserved

($d(\Pi') = d(\Pi)$) and where every formula in Π' is a subformula of a formula in Π.

We now need some definitions and lemmas concerning properties of formulas and extended rules.

If A is a formula then par(A) is the set of parameters that occur in A and con(A) is the set of individual constants that occur in A. If Δ is a formula set then we define $par(\Delta) = \bigcup_{A\in\Delta} par(A)$ and $con(\Delta) = \bigcup_{A\in\Delta} con(A)$. If Π is a derivation then $con(\Pi) = \bigcup_{A\in\Pi} con(A)$ where $A\in\Pi$ means that A is a formula of Π.

A function from a finite set of parameters into the set of (individual) constants $\{1,2,3, \dots\}$ is called a *substitution instance*. If σ is a substitution instance with domain $\{t_1,\dots,t_k\}$ then $A[\sigma]$ abbreviates $A[t_1/\sigma(t_1),\dots,t_k/\sigma(t_k)]$.
Let σ be a substitution instance.
If A is a formula then we define $D_n^\sigma(A) = D_n(A)[\sigma]$, (by an earlier comment we have $D_n(A)[\sigma] = D_n(A[\sigma])$).
If Δ is a finite formula set then we define:
$\Delta[\sigma] = \{A[\sigma] : A\in\Delta\}$.
$D_n(\Delta) = \{D_n(A): A\in\Delta\}$.
$D_n^\sigma(\Delta) = \{D_n^\sigma(A): A\in\Delta\}$.

$D_n^*(\Delta) = D_n^{\sigma_1}(\Delta) \cup \dots \cup D_n^{\sigma_m}(\Delta)$ where $\sigma_1,\dots,\sigma_m$ are *all* substitution instances from $par(\Delta)$ into $\{1,\dots,n\}$.

The first lemma states that under certain conditions D_n^* preserves extended rules.

Lemma 1. For any n=1,2,3,... the following holds:

If $\frac{\Delta}{\Delta'}$ is an instance of an extended rule and $con(\Delta') \subseteq \{1,\dots,n\}$ then $\dfrac{D_n^*(\Delta)}{D_n^*(\Delta')}$ is an instance of an extended rule.

Proof. Fix some $n\geq 1$.
We want to show that for every $A \in D_n^*(\Delta')$ there are formulas $A_1,\dots,A_k\in D_n^*(\Delta)$ so that $\dfrac{A_1 \dots A_k}{A}$ is an instance of an extended rule.
Let $A \in D_n^*(\Delta')$.
Then $A = D_n^\sigma(B)$ for some $B \in \Delta'$ and some substitution instance $\sigma{:}par(\Delta') \to \{1,\dots,n\}$. Since $\frac{\Delta}{\Delta'}$ is an instance of an extended rule there are formulas $B_1,\dots,B_k \in \Delta$ such that $\dfrac{B_1 \dots B_k}{B}$ is an instance of an extended rule.
We now get three cases.
1) Suppose $\dfrac{B_1 \dots B_k}{B}$ is not $\forall$I or $\forall$E.
The fact that $D_n(\neg C) = \neg D_n(C)$ and $D_n(C*C') = D_n(C)*D_n(C')$ if C and C' are formulas and $*$ is a connective then implies that $\dfrac{D_n(B_1) \dots D_n(B_k)}{D_n(B)}$ is an instsance of an extended rule. But then $\dfrac{D_n(B_1)[\sigma] \dots D_n(B_k)[\sigma]}{D_n(B)[\sigma]}$ is an instance of an extended rule and since $par(\Delta) \subseteq par(\Delta')$ (because $\Delta \subseteq \Delta'$) we also have $D_n(B_1)[\sigma],\dots,D_n(B_k)[\sigma] \in D_n^*(\Delta)$. Since $A = D_n^\sigma(B) = D_n(B)[\sigma]$ we are finished with case 1.

2) Suppose $\dfrac{B_1 \dots B_k}{B}$ is $\forall$I.
Then k=1 and $B = \forall x B_1[t/x]$ for some parameter t.

If t does not occur in B_1 then $A = D_n^\sigma(B) = D_n(B)[\sigma] = D_n(\forall x B_1[t/x])[\sigma] = D_n(B_1)[\sigma] = D_n^\sigma(B_1) \in D_n^\sigma(\Delta)$.

If t occurs in B_1 then $A = D_n^\sigma(B) = D_n(B)[\sigma] = D_n(\forall x B_1[t/x])[\sigma] = (D_n(B_1)[t/1] \;\&\; ... \;\&\; D_n(B_1)[t/n])[\sigma] =$

$D_n(B_1)[t/1][\sigma] \;\&\; ... \;\&\; D_n(B_1)[t/n][\sigma]$ and $D_n(B_1)[t/1][\sigma],...,D_n(B_1)[t/n][\sigma] \in D_n^*(\Delta)$.

Since $\dfrac{D_n(B_1)[t/1][\sigma] \;...\; D_n(B_1)[t/n][\sigma]}{D_n(B_1)[t/1][\sigma] \;\&\; ... \;\&\; D_n(B_1)[t/n][\sigma]}$ is an instance of an extended rule we are finished with case 2.

3) Suppose $\dfrac{B_1 \;...\; B_k}{B}$ is $\forall E$.

Then k=1 and $B_1 = \forall x C[t/x]$ for some formula C and parameter t and $B = C[t/t']$ where t' is a parameter or $t' \in \{1,...,n\}$ (because $con(\Delta') \subseteq \{1,...,n\}$ by assumption).

If t does not occur in C then B = C so $A = D_n^\sigma(B) = D_n(B)[\sigma] = D_n(C)[\sigma] = D_n(B_1)[\sigma] = D_n^\sigma(B_1) \in D_n^\sigma(\Delta)$.

Now suppose t occurs in C.

If t' is a parameter then $\sigma(t')=i$ for some $i \in \{1,...,n\}$ and $D_n(C[t/t'])[\sigma] = D_n(C)[t/t'][\sigma] = D_n(C)[t/i][\sigma]$. If $t' \in \{1,...,n\}$ then $D_n(C[t/t']) = D_n(C)[t/t'] = D_n(C)[t/i]$ for some $i \in \{1,...,n\}$. In both cases

$\dfrac{D_n(C)[t/1][\sigma] \;\&\; ... \;\&\; D_n(C)[t/n][\sigma]}{D_n(C[t/t'])[\sigma]}$ is an instance of an extended rule. Since $A = D_n^\sigma(B) = D_n(B)[\sigma] =$

$D_n(C[t/t'])[\sigma]$ and $D_n(C)[t/1][\sigma] \;\&\; ... \;\&\; D_n(C)[t/n][\sigma] = (D_n(C)[t/1] \;\&\; ... \;\&\; D_n(C)[t/n])[\sigma] = D_n(\forall x C[t/x])[\sigma]$

$= D_n(B_1)[\sigma] = D_n^\sigma(B_1) \in D_n^\sigma(\Delta)$ we are finished with case 3.

The following lemma tells that under certain conditions the property of being a subformula of another formula is preserved by D_n^σ.

Lemma 2. For every n=1,2,3,... the following holds:

Let $\sigma: P \to \{1,...,n\}$ be a substitution instance (P is a finite set of parameters).

If A is a subformula of B, $par(A) \subseteq P$, $con(A) \subseteq \{1,...,n\}$ and $par(B) = \emptyset$ then $D_n^\sigma(A)$ is a subformula of $D_n(B)$.

Proof. Fix some $n \geq 1$. For the given n we will prove the lemma by induction on the complexity of B.

If B is atomic then A=B and so A has no parameters and $D_n^\sigma(A) = D_n(A) = D_n(B)$.

If B is not atomic and A=B then A has no parameters so $D_n^\sigma(A) = D_n(A) = D_n(B)$.

Now suppose that B is not atomic and $A \neq B$.

If $B = B_1 * B_2$ where $*$ is a connective then A is a subformula of B_1 or B_2. Since $par(B_1) = \emptyset$ and $par(B_2) = \emptyset$ the induction hypothesis gives that $D_n^\sigma(A)$ is a subformula of $D_n(B_1)$ or $D_n(B_2)$ which means that $D_n^\sigma(A)$ is a

subformula of $D_n(B_1) * D_n(B_2) = D_n(B_1 * B_2) = D_n(B)$. If $B = \neg C$ we can reason similarly.

Suppose $B = \forall x C[t/x]$ where t is a term. Since A is a subformula of $\forall x C[t/x]$ there is a term t' such that A is a subformula of $C_1 = C[t/t']$ and since t is the only parameter of C (because $par(\forall x C[t/x]) = \emptyset$) we also have $B = \forall x C_1[t'/x]$. Since $con(A) \subseteq \{1,...,n\}$, either t' is a parameter or $t \in \{1,...,n\}$.

If $t' \in \{1,...,n\}$ then C_1 has no parameters (because otherwise B would have parameters) and by the induction hypothesis $D_n^\sigma(A)$ is a subformula of $D_n(C_1)$ and since $D_n(C_1) = D_n(C_1)[t'/t']$ is a subformula of $D_n(C_1)[t'/1] \;\&$

$... \;\&\; D_n(C_1)[t'/n] = D_n(\forall x C_1[t'/x]) = D_n(B)$ we have that $D_n^\sigma(A)$ is a subformula of $D_n(B)$.

If t' is a parameter and t' does not occur in A then A is a subformula of $C_1[t'/1]$ (for instance) and $par(C_1[t/1]) =$

$\emptyset$ so by the induction hypothesis $D_n^\sigma(A)$ is a subformula of $D_n(C_1[t'/1]) = D_n(C_1)[t'/1]$ which is a subformula of

$D_n(C_1)[t'/1] \;\&\; ... \;\&\; D_n(C_1)[t'/n] = D_n(\forall x C_1[t'/x]) = D_n(B)$.

Now suppose that t' is a parameter that occurs in A. Then $t' \in P$ by assumption so $A[t'/\sigma(t')]$ is a subformula of $C_1[t'/\sigma(t')]$. We also have $par(A[t'/\sigma(t')]) \subseteq P$, $con(A[t'/\sigma(t')]) \subseteq \{1,...,n\}$ and $par(C_1[t'/\sigma(t')]) = \emptyset$. Hence by the induction hypothesis $D_n^\sigma(A[t'/\sigma(t')])$ is a subformula of $D_n(C_1[t'/\sigma(t')])$. We also have $D_n^\sigma(A) = D_n(A)[\sigma] =$

$D_n(A)[t'/\sigma(t')][\sigma] = D_n(A[t'/\sigma(t')])[\sigma] = D_n^\sigma(A[t'/\sigma(t')])$ and $D_n(C_1[t'/\sigma(t')]) = D_n(C_1)[t'/\sigma(t')]$. Since $D_n(C_1)[t'/\sigma(t')]$ is a subformula of $D_n(C_1)[t'/1]\ \&\ ...\ \&\ D_n(C_1)[t'/n] = D_n(\forall x C_1[t'/x]) = D_n(B)$ we conclude that $D_n^\sigma(A)$ is a subformula of $D_n(B)$.

If Π is an extended derivation and σ a substitution instance then $\Pi[\sigma]$ is defined to be the result of replacing every formula set Δ of Π by $\Delta[\sigma]$.

Lemma 3. For every n=1,2,3,... the following holds:

Let $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$ be an extended derivation which does not depend on any of the parameters $t_1,...,t_k$ and assume $con(\Pi) \subseteq \{1,...,n\}$ and $F(\Pi) \subseteq sub(C)$ for some sentence C. Then for every substitution instance $\sigma: \{t_1,...,t_k\} \rightarrow \{1,...,n\}$, $\begin{array}{c}\Delta[\sigma]\\ \Pi[\sigma]\\ \Delta'[\sigma]\end{array}$ is an extended derivation and $con(\Pi[\sigma]) \subseteq \{1,...,n\}$, $F(\Pi[\sigma]) \subseteq sub(C)$ and $d(\Pi[\sigma]) = d(\Pi)$.

Proof. By induction on the complexity of derivations.

Let $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$ be a derivation and Δ_0 a formula set.

If Π' is the result of replacing every formula set Δ of Π by $\Delta \cup \Delta_0$ then clearly $\begin{array}{c}\Delta\cup\Delta_0\\ \Pi'\\ \Delta'\cup\Delta_0\end{array}$ is a derivation and we say that it is an imitation of $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$.

Now we are ready for the main lemma from which proposition 2 will easily follow.

Lemma 4. For every n=1,2,3,... the following holds:

If $\begin{array}{c}\Delta\\ \Pi\\ \Delta'\end{array}$ is an extended derivation such that $F(\Pi) \subseteq sub(C)$ for some sentence C and $con(\Pi) \subseteq \{1,...,n\}$ then there exists a propositional extended derivation $\begin{array}{c}D_n^*(\Delta)\\ \Pi'\\ D_n^*(\Delta')\end{array}$ such that $F(\Pi') \subseteq sub(D_n(C))$, $con(\Pi') \subseteq \{1,...,n\}$ and $d(\Pi') = d(\Pi)$.

Proof. We first fix some $n \geq 1$ and then prove the result for this n by induction on the complexity of extended derivations. Let Π be a derivation such that $F(\Pi) \subseteq sub(C)$ for a sentence C and $con(\Pi) \subseteq \{1,...,n\}$.
Basis: $\Pi = \Delta$.
Let $\Pi' = D_n^*(\Delta)$. Then Π' is a propositional extended derivation and $d(\Pi') = d(\Pi)$. If A is a formula of Π' then A $= D_n^\sigma(B)$ for some formula $B \in \Delta$ and substitution instance $\sigma: par(\Delta) \rightarrow \{1,...,n\}$. Since $B \in sub(C)$ lemma 2 gives that $A \in sub(D_n(C))$. By the definition of D_n^* we also have $con(\Pi') \subseteq \{1,...,n\}$.

Induction step:

1) Suppose $\Pi = \dfrac{\begin{matrix}\Delta \\ \Pi_1 \\ \Delta_1\end{matrix}}{\Delta'}$. By the induction hypothesis there exists a propositional extended derivation $\begin{matrix}D_n^*(\Delta) \\ \Pi'_1 \\ D_n^*(\Delta_1)\end{matrix}$ such that $F(\Pi'_1) \subseteq sub(D_n(C))$, $con(\Pi') \subseteq \{1,...,n\}$ and $d(\Pi'_1) = d(\Pi_1)$. Since $\dfrac{\Delta_1}{\Delta'}$ must be an instance of an extended rule and $con(\Delta') \subseteq \{1,...,n\}$ lemma 1 gives that $\dfrac{D_n^*(\Delta_1)}{D_n^*(\Delta')}$ is an instance of an extended rule.

Hence $\Pi' = \dfrac{\begin{matrix}D_n^*(\Delta) \\ \Pi'_1 \\ D_n^*(\Delta_1)\end{matrix}}{D_n^*(\Delta')}$ is a propositional extended derivation and clearly $d(\Pi') = d(\Pi)$. If $A \in D_n^*(\Delta')$ then $A = D_n^{\sigma}(B)$ for some formula $B \in \Delta'$ and substitution instance $\sigma: par(\Delta') \to \{1,...,n\}$. But $B \in sub(C)$ so lemma 2 gives that $A \in sub(D_n(C))$. Hence $D_n^*(\Delta') \subseteq sub(D_n(C))$ and since we already have $F(\Pi'_1) \subseteq sub(D_n(C))$ we get $F(\Pi') \subseteq sub(D_n(C))$. By the definition of D_n^* we also have $con(\Pi') \subseteq \{1,...,n\}$.

2) Suppose $\Pi = \dfrac{\dfrac{\begin{matrix}\Delta \\ \Pi_1 \\ \Delta_1\end{matrix}}{\begin{matrix}\Delta_1 \cup \{A\} & \Delta_1 \cup \{\neg A\} \\ \Pi_2 & \Pi_3 \\ \Delta_2 & \Delta_3\end{matrix}}}{\Delta_3}$ where $\perp \in \Delta_2$.

Let $\sigma_1,...,\sigma_m$ be all substitution instances from $par(A)$ into $\{1,...,n\}$. Since $\begin{matrix}\Delta_1 \cup \{A\} \\ \Pi_2 \\ \Delta_2\end{matrix}$ and $\begin{matrix}\Delta_1 \cup \{\neg A\} \\ \Pi_3 \\ \Delta_3\end{matrix}$ do not depend on any parameter in $par(A)$ lemma 3 gives that $\begin{matrix}\Delta_1[\sigma_i] \cup \{A[\sigma_i]\} \\ \Pi_2[\sigma_i] \\ \Delta_2[\sigma_i]\end{matrix}$ and $\begin{matrix}\Delta_1[\sigma_i] \cup \{\neg A[\sigma_i]\} \\ \Pi_3[\sigma_i] \\ \Delta_3[\sigma_i]\end{matrix}$ are extended derivations for $i=1,...,m$ and $d(\Pi_2[\sigma_i]) = d(\Pi_2)$, $d(\Pi_3[\sigma_i]) = d(\Pi_3)$ and $F(\Pi_2[\sigma_i]) \subseteq sub(C)$, $F(\Pi_3[\sigma_i]) \subseteq sub(C)$ and $con(\Pi_2[\sigma_i]) \subseteq \{1,...,n\}$, $con(\Pi_3[\sigma_i]) \subseteq \{1,...,n\}$. By the induction hypothesis there are propositional extended derivations

$$\begin{matrix}D_n^*(\Delta) \\ \Pi'_1 \\ D_n^*(\Delta_1)\end{matrix} \quad \text{and} \quad \begin{matrix}D_n^*(\Delta_1[\sigma_i]) \cup \{D_n(A[\sigma_i])\} \\ \Pi_{2,i} \\ D_n^*(\Delta_2[\sigma_i])\end{matrix} \quad \text{and} \quad \begin{matrix}D_n^*(\Delta_1[\sigma_i]) \cup \{\neg D_n(A[\sigma_i])\} \\ \Pi_{3,i} \\ D_n^*(\Delta_3[\sigma_i])\end{matrix}$$

such that $d(\Pi'_1) = d(\Pi_1)$, $d(\Pi_{2,i}) = d(\Pi_2)$, $d(\Pi_{3,i}) = d(\Pi_3)$ and $F(\Pi'_1) \subseteq sub(D_n(C))$, $F(\Pi_{2,i}) \subseteq sub(D_n(C))$, $F(\Pi_{3,i}) \subseteq sub(D_n(C))$ and $con(\Pi_{2,i}) \subseteq \{1,\ldots,n\}$, $con(\Pi_{3,i}) \subseteq \{1,\ldots,n\}$ for $i=1,\ldots,m$. Since $\perp \in D^*_n(\Delta_2[\sigma_i])$ we can form the extended derivations

$$\Sigma_i = \cfrac{\cfrac{D^*_n(\Delta_1[\sigma_i])}{\begin{array}{cc} D^*_n(\Delta_1[\sigma_i]) \cup \{D_n(A[\sigma_i])\} & D^*_n(\Delta_1[\sigma_i]) \cup \{\neg D_n(A[\sigma_i])\} \\ \Pi_{2,i} & \Pi_{3,i} \\ D^*_n(\Delta_2[\sigma_i]) & D^*_n(\Delta_3[\sigma_i]) \end{array}}}{D^*_n(\Delta_3[\sigma_i])} \qquad \text{for } i=1,\ldots,m.$$

$$\text{Let}\quad \Pi' = \begin{array}{l} D^*_n(\Delta) \\ \Pi'_1 \\ D^*_n(\Delta_1) \\ \Sigma'_1 \\ D^*_n(\Delta_1) \cup D^*_n(\Delta_3[\sigma_1]) \\ \Sigma'_2 \\ D^*_n(\Delta_1) \cup D^*_n(\Delta_3[\sigma_1]) \cup D^*_n(\Delta_3[\sigma_2]) \\ \Sigma'_3 \\ \vdots \\ \Sigma'_m \\ D^*_n(\Delta_1) \cup \bigcup_{i=1}^{m} D^*_n(\Delta_3[\sigma_i]) \end{array} \qquad \text{where } \Sigma'_i \text{ are the appropriate imitations of } \Sigma_i.$$

Π' is an extended derivation because $D^*_n(\Delta_1) = \bigcup_{i=1}^{m} D^*_n(\Delta_1[\sigma_i])$ and the premise set of Σ_i is $D^*_n(\Delta_1[\sigma_i])$.

We also have $\bigcup_{i=1}^{m} D^*_n(\Delta_3[\sigma_i]) = D^*_n(\Delta_3)$, because $\sigma_1,\ldots,\sigma_m$ are all substitution instances from $par(A)$ into $\{1,\ldots,n\}$, and $D^*_n(\Delta_1) \subseteq D^*_n(\Delta_3)$, because $\Delta_1 \subseteq \Delta_3$. Hence $D^*_n(\Delta_1) \cup \bigcup_{i=1}^{m} D^*_n(\Delta_3[\sigma_i]) = D^*_n(\Delta_3)$ so Π' is a propositional extended derivation with premise set $D^*_n(\Delta)$ and conclusion set $D^*_n(\Delta_3)$. By the construction of Π' we see that $d(\Pi') = d(\Pi)$. Since $F(\Pi') = F(\Pi'_1) \cup \bigcup_{i=1}^{m} F(\Pi_{2,i}) \cup \bigcup_{i=1}^{m} F(\Pi_{3,i})$ and

$$con(\Pi') = con(\Pi'_1) \cup \bigcup_{i=1}^{m} con(\Pi_{2,i}) \cup \bigcup_{i=1}^{m} con(\Pi_{3,i})$$

(by the construction of Π') we also have $F(\Pi') \subseteq sub(D_n(C))$ and $con(\Pi') \subseteq \{1,...,n\}$.

Now proposition 1 is an easy consequence of lemma 4.

Proposition 1. If Π is a subformula proof of a sentence A then there are propositional subformula proofs Π_1, Π_2, Π_3,... of $D_1(A)$, $D_2(A)$, $D_3(A)$,... such that $d(\Pi_n) = d(\Pi)$ for n=1,2,3,... .

Proof. Since Π is a proof of a sentence A we can, without loss of generality, assume that no individual constants occur in any formula of Π, with other words we may assume $con(\Pi) = \varnothing$. Since Π is a subformula proof of A we also have $F(\Pi) \subseteq sub(A)$. Let Δ be the conclusion set of Π, (the premise set of Π is $\{\neg A\}$). Then by lemma 4 (because a derivation is also an extended derivation) there are extended derivations

$$\begin{array}{c} D^*_1(\{\neg A\}) \\ \Pi'_1 \\ D^*_1(\Delta) \end{array}, \quad \begin{array}{c} D^*_1(\{\neg A\}) \\ \Pi'_2 \\ D^*_1(\Delta) \end{array}, \quad \begin{array}{c} D^*_3(\{\neg A\}) \\ \Pi'_3 \\ D^*_3(\Delta) \end{array}, ...$$

such that $d(\Pi'_n) = d(\Pi)$ and $F(\Pi'_n) \subseteq sub(D_n(A))$ for n=1,2,3,... .

Since no parameters occur in A we have $D^*_n(\{\neg A\}) = \{ D_n(\neg A) \} = \{ \neg D_n(A) \}$ and since $\perp \in \Delta$ we also have $\perp \in D^*_n(\Delta)$. Hence Π'_1, Π'_2, Π'_3,... are extended proofs of $D_1(A)$, $D_2(A)$, $D_3(A)$,... with the subformula property and $d(\Pi'_n) = d(\Pi)$ for all n. As was pointed out earlier these can be transformed into proofs Π_1, Π_2, Π_3,... of $D_1(A)$, $D_2(A)$, $D_3(A)$,... with the subformula property and with $d(\Pi_n) = d(\Pi)$ for n=1,2,3,... .

Completeness of Reductio

To prove completeness of Reductio we will use its relationship with the proof system KEQ. We will give the system KEQ in a slightly different form than it appears in [5], [7], [8]. In [5], [7], [8] KEQ-refutations are presented as trees of signed formulas, here we will present them as trees of formula sets. The rules of KEQ include all the elimination rules of Reductio and, in addition, the following rules:

$$\frac{\neg(A\&B) \quad A}{\neg B}\ \neg\&E1 \qquad \frac{\neg(A\&B) \quad B}{\neg A}\ \neg\&E2 \qquad \frac{\neg(A\vee B)}{\neg A}\ \neg\vee E1$$

$$\frac{\neg(A\vee B)}{\neg B}\ \neg\vee E2 \qquad \frac{\neg(A\to B)}{A}\ \neg\to E1 \qquad \frac{\neg(A\to B)}{\neg B}\ \neg\to E2 \qquad \frac{\neg\neg A}{A}\ \neg\neg E$$

$\dfrac{\neg\forall xA}{\neg A[x/t]}$ $\neg\forall E$ introducing t, where t is a parameter.

KEQ as presented in [5], [7], [8] operates on a language which contains the symbol $\exists$ and has an elimination rule for $\exists$ ($\exists E$) and one elimination rule for $\neg\exists$ ($\neg\exists E$). But the fact that KEQ is complete also when we impose the restriction that only formulas in sub(A) may occur in a KEQ-proof of A implies that for a language without the symbol $\exists$ (as our language) the fragment of KEQ without the rules $\exists E$ and $\neg\exists E$ is complete (also with the restriction that a proof of A may contain only formulas in sub(A)).

If Δ is a finite formula set and there are $A_1,...,A_k \in \Delta$ such that $\dfrac{A_1 \ldots A_k}{A}$ is an instance of a rule of KEQ then we say that $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance (or an application) of a rule. KEQ-*refutations* are defined by:

(i) If Δ is a finite formula set such that $B\in\Delta$ and $\neg B\in\Delta$ for some formula B then Δ is a KEQ-refutation of Δ.

(ii) If T is a refutation of $\Delta\cup\{A\}$ and $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance of a rule, but not $\neg\forall E$ introducing t if $t\in\Delta$, then $\dfrac{\Delta}{T}$ is a refutation of Δ.

(iii) If T_1 is a refutation of $\Delta\cup(A)$ and T_2 is a refutation of $\Delta\cup\{\neg A\}$ then $\dfrac{\Delta}{T_1 \quad T_2}$ is a refutation of Δ.

If T is a KEQ-refutation of Δ then we may denote T by $\dfrac{\Delta}{T}$. A KEQ-*proof* of a formula A is a KEQ-refutation of $\{\neg A\}$. As it was already pointed out, our variant of KEQ is complete for our language and if A is a valid sentence then there is a KEQ-refutation of $\{\neg A\}$ in which only formulas of sub(A) occur. Hence, if we want to prove that for every valid sentence A there is a subformula Reductio-proof of A, it is sufficient to prove the following lemma.

Lemma. If T is a KEQ-refutation of Δ then there is a Reductio-refutation $\begin{array}{c}\Delta\\ \Pi\\ \bot\end{array}$ of Δ such that every formula in Π that is different from $\bot$ is a subformula of a formula in T.

Proof. By induction on the complexity of T.

(i) If T is Δ then some formula A both A and $\neg A$ belong to Δ. Hence $\dfrac{\Delta}{\Delta\cup\{\bot\}}$ is a reductio-refutation of Δ which satisfies the claim of the lemma.

(ii) Suppose that T = $\begin{array}{c}\Delta\\ \hline \Delta\cup\{A\}\\ T_1\end{array}$. By induction there is a Reductio-refutation $\begin{array}{c}\Delta\cup\{A\}\\ \Pi_1\\ \bot\end{array}$ of $\Delta\cup\{A\}$ such that every formula in $\begin{array}{c}\Delta\cup\{A\}\\ \Pi_1\\ \bot\end{array}$ that is different from $\bot$ is a subformula of a formula in T_1. We now get different cases depending on $\dfrac{\Delta}{\Delta\cup\{A\}}$. If $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance of one of the elimination rules of reductio (&E1,&E2,$\vee$E1,$\vee$E2,$\rightarrow$E1,$\rightarrow$E2,$\forall$E) then $\Pi = \begin{array}{c}\Delta\\ \hline \Delta\cup\{A\}\\ \Pi_1\\ \bot\end{array}$ is a Reductio-refutation of Δ such that every formula in Π that is different from $\bot$ is a subformula of a formula in T.

Now suppose $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance of one of the rules $\neg$&E1,$\neg$&E2,$\neg\vee$E1,$\neg\vee$E2,$\neg\rightarrow$E1,$\neg\rightarrow$E2,$\neg\neg$E. We will only do the case $\neg$&E1, the others are handled similarly. If $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance of $\neg$&E1 then there are formulas B and C such that $\neg$(B&C),B$\in\Delta$ and A=$\neg$C so if we put

$$\Pi = \dfrac{\dfrac{\Delta}{\dfrac{\dfrac{\Delta\cup\{C\}}{\Delta\cup\{C,B\&C\}}}{\Delta\cup\{C,B\&C,\bot\}} \qquad \Delta\cup\{\neg C\}}}{\begin{array}{c}\Delta\cup\{\neg C\}\\ \Pi_1\\ \bot\end{array}}$$

then Π is a Reductio-refutation of Δ such that every formula in Π that is different from $\perp$ is a subformula of a formula in T.

The last case to consider is if $\dfrac{\Delta}{\Delta\cup\{A\}}$ is an instance of $\neg\forall$E. Then there is a formula $\neg\forall xB\in\Delta$ such that $A = \neg B[x/t]$ where t is a parameter and $t\notin par(\Delta)$. If we put

$$\Pi = \dfrac{\dfrac{\dfrac{\dfrac{\Delta}{\Delta\cup\{B[x/t]\} \qquad \begin{matrix}\Delta\cup\{\neg B[x/t]\}\\ \Pi_1 \\ \perp\end{matrix}}}{\Delta\cup\{B[x/t]\}}}{\Delta\cup\{B[x/t],\forall xB\}}}{\Delta\cup\{B[x/t],\forall xB,\perp\}}$$

then Π is a Reductio-refutation of Δ such that every formula in Π that is different from $\perp$ is a subformula of a formula in T.

(iii) Suppose that $T = \dfrac{\Delta}{\begin{matrix}\Delta\cup\{A\}\\ T_1\end{matrix} \quad \begin{matrix}\Delta\cup\{\neg A\}\\ T_2\end{matrix}}$. By induction there are Reductio-refutations $\begin{matrix}\Delta\cup\{A\}\\ \Pi_1 \\ \perp\end{matrix}$ and $\begin{matrix}\Delta\cup\{\neg A\}\\ \Pi_2 \\ \perp\end{matrix}$ of $\Delta\cup\{A\}$ and $\Delta\cup\{\neg A\}$ respectively, such that every formula in Π_1 (Π_2) different from $\perp$ is a subformula of a formula in T_1 (T_2). If we put

$$\Pi = \dfrac{\dfrac{\Delta}{\begin{matrix}\Delta\cup\{A\}\\ \Pi_1 \\ \perp\end{matrix} \qquad \begin{matrix}\Delta\cup\{\neg A\}\\ \Pi_2 \\ \perp\end{matrix}}}{\perp}$$

then Π is a Reductio-refutation of Δ such that every formula in Π that is different from $\perp$ is a subformula of a formula in T.

References

[1] M. Ajtai, The complexity of the pigeonhole principle, *Proceedings of the 29th Annual Symposium on the Foundations of Computer Science*, 1988.

[2] S.R. Buss, Polynomial size proofs of the propositional pigeonhole principle, *The Journal of Symbolic Logic*, vol. 52, 916-927, 1987.

[3] S.A. Cook, The complexity of theorem proving procedures, *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, 151-158, 1971.

[4] S.A. Cook and R.A. Reckhow, The relative efficiency of propositional proof systems, *The Journal of Symbolic Logic*, vol. 44, 36-50, 1979.

[5] M. D'Agostino, *Investigations into the complexity of some propositional calculi*, Ph.d. thesis, Oxford University Computing Laboratory, Oxford University, 1990.

[6] A. Haken, The intractability of resolution, *Theoretical Computer Science*, vol. 39, 297-308, 1985.

[7] M. Mondadori, Classical analytical deduction, *Annali dell'Università di Ferrara*, Discussion paper n.1, Università di Ferrara, 1988.

[8] M. Mondadori, Classical analytical deduction, part II, *Annali dell'Università di Ferrara*, Discussion paper n.5, Università di Ferrara, 1988.

[9] D. Prawitz, *Natural deduction. A proof theoretical study*, Almqvist & Wiksell, Uppsala, 1965.

[10] R.A. Reckhow, *On the lengths of proofs in the propositional calculus*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1976.

[11] G. Stålmarck, A proof theoretic concept of tautological hardness, unpublished manuscript, 1994.

[12] G. Stålmarck and M. Säflund, Modelling and verifying systems and software in propositional logic, *Safety of Computer Control Systems*, 31-36, 1990.

[13] A. Urquhart, Complexity of proofs in classical propositional logic, *Logic from computer science*, Proceedings Workshop, Berkeley/CA 1989, Publ., Math. Sci. Res. Inst. 21, 597-608, 1992.