On the Polynomial Transparency of Resolution[4]

Reinhold Letz
Institut fur Informatik
Technische Universitat Munchen
Arcisstr. 21, 8000 Munchen 2
Germany

Abstract

In this paper a framework is developed for measuring the complexities of deductions in an abstract and computationally perspicuous manner. As a notion of central importance appears the so-called *polynomial transparency* of a calculus. If a logic calculus possesses this property, then the complexity of *any* deduction can be correctly measured in terms of its inference steps. The resolution calculus lacks this property. It is proven that the number of inference steps of a resolution proof does not give a representative measure of the *actual* complexity of the proof, even if only shortest proofs are considered. We use a class of formulae which have proofs with a polynomial number of *inference steps,* but for which the *size* of any proof is exponential. The polynomial intransparency of resolution is due to the renaming of derived clauses, which is a fundamental deduction mechanism. This result motivates the development of new data structures for the representation of logical formulae.

1 Introduction

The competitiveness of logic calculi is essentially determined by two complementary factors; on the one hand, by the ability to provide compact proofs, and on the other, by the effort needed for finding such proofs, that is, by the search spaces induced by the indeterminism inherent in the calculi. In this paper, we shall systematically address the problem how to measure the first of these two capabilities of a calculus—its *in deterministic power*—in an abstract and nevertheless computationally reliable manner.

The paper consists of two parts, a conceptual part and an application part. In the conceptual part, we present a general framework for measuring the computational complexities of arbitrary transition relations and deductions, which are treated as particular transition relations. In order to be able to compare complexities on a level which is as abstract as possible, we subscribe to abstractions modulo *polynomials,* as usual in complexity theory. The central notions emerging this way are the properties of *polynomial transparency* and *weak polynomial transparency.* The polynomial transparency of a transition relation guarantees that the number of rewrite steps in *any* transition sequence represents an adequate measure for the actual computational complexity of the sequence. Weak polynomial transparency is the adequate concept for evaluating the *tndetermintstic powers* of special transition relations, called *proof relations,* by restricting attention to *shortest proofs* only.

The benefit of the framework is twofold, not only does it facilitate the classification of deduction systems, it also may give advice how to improve the systems. This is illustrated in the application part of this paper where the developed notions are used on the resolution calculus. It is proven that resolution lacks polynomial transparency, both in the strong and in the weak sense. As a consequence, the number of inference steps of a shortest resolution proof does not give a representative measure of the *actual* complexity of the proof. The use of *number terms* in the object language can remedy this weakness for a certain class of formulae. Whether a general remedy exists remains an open problem.

2 Complexity Measures for Deductions

The indeterministic power of a calculus is determined by the complexities of shortest proofs. This raises the fundamental question how the complexities of proofs and deductions in general should be measured.

2.1 Deduction Processes as Transition Relations

For investigations into the computational complexity of logic calculi, it is important to realize that deductions can be given a *declarative* and a *process* interpretation. According to the declarative reading, deductions are typically defined as sequences of formulae $D = F_0, \ldots, F_n$ where each F_i, $1 \leq i \leq n$, is derivable by applying an inference rule to formulae in D with an index stricly less than t. Another popular interpretation is to define deductions as trees of formulae where each parent node can be derived from its immediate successors. There is no limitation to further ways of defining deductions. De-

*This work was financially supported by the Esprit Basic Research Actions 3125 Medlar I and 6471 Medlar II.

ductions as static objects of the type mentioned above tend to be non-operational, in the sense that they do not prescribe the precise methodology according to which they have to be constructed. A deduction *process* can be viewed as one particular way of building up a deduction object. From a complexity point of view, the deduction process is the more fundamental notion and the deduction object is just a—even though extremely useful—by-product of the deduction process.[1]

The most natural and general specification approach for the description of deduction processes is to model them as binary *transition relations* between *structures* of logical expressions, which play the role of the *states* in the transition relations. Typically, these states are finite *sets* of formulae, in general, they may be arbitrary finitistic set-theoretic objects, or strings encoding such objects. The binary transition relation $\vdash$ of the deduction processes defined by a calculus is determined by the inference rules of the calculus.[2] Like for the logical consequence relation, the notions of deductions—we will call them *derivations*—and *proofs* can be introduced for the state transition relation $\vdash$. A *derivation in* a transition relation $\vdash$ is a sequence D of states such that for each pair of successive states S_i and S_{i+1} in D: $S_i \vdash S_{i+1}$. *Proofs* can be defined by associating with a given transition relation $\vdash$ a collection of *input states* Σ, and a distinguished state Ω, named the *success state*. We shall call such transition relations *proof relations*. Then, any finite derivation in $\vdash$ with an initial state $S \in \Sigma$ and terminal state Ω is said to be a *proof of* S in $\vdash$. Using some notation from the field of *reduction relations*, given a transition relation $\vdash$, we abbreviate with $\vdash^k$ the k-fold composition of $\vdash$. The *(minimal) distance* between any pair $\langle S, S' \rangle$ of states in the transitive closure of a transition relation $\vdash$ is defined as $\min(\{k \mid S \vdash^k S'\})$, and abbreviated with $\delta(S, S', \vdash)$.

2.2 Three Natural Measures for Derivations

For evaluating the complexity of a derivation $S_0, \ldots, S_n$ in a transition relation $\vdash$, three different measures are the obvious alternatives, which correspond to three different degrees of precision. The finest measure charges the minimal *computing cost* needed in a *basic machine model* to come from the initial state S_0 to the terminal state S_n via the given intermediate states in the derivation. The computing cost of rewriting a state S_i to a state S_{i+1} may be, for example, the minimal number of configurations of a *nondeterministic Turing machine* (or the machine operations of the indeterministic version of any alternative *realistic* machine model)[3] to transform S_i into S_{i+1}. Conceptually, the chosen basic machine model can be viewed as another (more elementary) transition relation, written $\longrightarrow$. Then, the *elementary computing cost* of the derivation $D = S_0, \ldots, S_n$ can be defined as

$$\mathrm{cost}(D) = \sum_{i=0}^{n-1} \delta(S_i, S_{i+1}, \rightarrow).$$

Taking the elementary computing cost of a derivation as the measure of its complexity has certain disadvantages. First, for the standard realistic machine models, the measure is too detailed to be interesting as a quantity of comparison on a higher level of abstraction. Second, its value may vary strongly, depending on the chosen realistic machine model—even though only up to polynomials. Lastly, it may be very difficult to *actually obtain* the realistic computing cost, because the mapping down of high-level transition steps into basic machine operations is normally not carried out explicitly, instead one is satisfied with *knowing about the possibility* of such a transformation and its computational invariances.

An advance is offered by disregarding the elementary computing cost and restricting oneself to a higher level of representation, by only considering the *size* of a derivation $D = S_0, \ldots, S_n$, which is simply the sum of the (string) sizes of the states in D:

$$\mathrm{size}(D) = \sum_{i=0}^{n} \mathrm{size}(S_i).$$

The highest abstraction level even ignores the *size* of a derivation $D = S_0, \ldots, S_n$ and considers only the *number of rewrite steps* in the top-level transition relation $\vdash$, in terms of logic calculi, the number of inference steps:

$$\mathrm{steps}(D) = n.$$

Eventually, it is this measure that is being striven for. It has been used successfully for analyzing the indeterministic powers of many propositional calculi, for example, in [Reckhow, 1976], [Haken, 1985], and various other papers. The abstraction performed by these authors is an abstraction *modulo polynomials;* they make plausible that the elementary computing cost is polynomially bounded by the number of inferences. Such an abstraction is very natural in that it takes into account the problem area of *NP* vs *coNP,* on the one hand, and additionally leaves aside uninteresting subpolynomial differences which result from the choice of the realistic machine model, on the other.

2.3 Polynomial Size- and Step-Transparency

The following two notions are fundamental for a general theory of the abstraction modulo polynomials. First, we consider the abstraction step from the elementary computing cost to the *size* of a derivation, and state under which condition such an abstraction is permissible.

Definition 1 [Polynomial size-transparency] A transition relation $\vdash$ is called *polynomially size-transparent* if

[1]This evaluation can be justified by recalling under which conditions a given object is accepted as a deduction of a type S, namely, if there exists a *procedure* which decides whether the object has type S. And the true complexity of a deduction object is the complexity of this verification procedure (for further motivation, consult [Letz, 1993]).

[2]As a matter of fact, $\vdash$ does *not* denote the standard logical consequence relation.

[3]A *realistic* machine model can be defined to be any machine model in which the elementary operations are polynomially related to the configurations of Turing machines (cf. the further remarks in [van Emde Boas, 1990] and [Letz, 1993]).

there is a polynomial p such that for every derivation D $= S_0, \ldots, S_n$ in $\vdash$:

$$\mathrm{cost}(D) < p(\mathrm{size}(D)).$$

If a transition relation h is polynomially size-transparent, then the size of any derivation gives a representative complexity measure of its elementary computing cost, as long as we are interested in complexities modulo polynomials. Polynomial size-transparency generalizes a basic concept introduced by Cook and Reckhow in [Cook and Reckhow, 1974]. They define a (complete) proof system as a (surjective) in polynomial time computable function from the set of strings to the set of valid formulae. Apparently, any proof system is polynomially size-transparent.

In order to define a general criterion which guarantees that we can even abstract from the size of a derivation, it is necessary to use polynomials in two arguments.

Definition 2 [Polynomial (step-)transparency] A transition relation I- is called polynomially step-transparent or just polynomially transparent if there is a polynomial p in two arguments such that for every derivation D $= S_0, \ldots, S_n$ in $\vdash$:

$$\mathrm{cost}(D) < p(\mathrm{size}(S_0), n).$$

If a transition relation is polynomially transparent, then the input size and the number of rewrite steps of any derivation give a representative measure of the complexity of the derivation.

Note It is apparent that indeed a polynomial in two arguments is needed, demanding that cost(D) < p(n) does not result in a useful notion. As an example, consider a calculus which solely can check whether a logical formula has the structure $F \vee \neg F$. According to the intended reading of inference steps, we wish to say •that the calculus can verify its input in a single inference step. However, there is no complexity function (and hence no polynomial) which bounds the elementary computing cost for verifying formulae of arbitrary size that have the shape $F \vee \neg F$.

Clearly, if a transition relation I- is polynomially transparent, then I- is polynomially size-transparent.

2.4 A Sufficient Condition for Polynomial Transparency

It is apparent that polynomial transparency is a highly desirable property.4 The question is now how to determine whether a transition relation is polynomially transparent. Polynomial transparency is a property defined on derivations of arbitrary lengths. It would be very comfortable if the polynomial transparency of a transition relation could be derived from more elementary properties of the transition relation. A very useful sufficient condition for polynomial transparency, which only takes into account the step behaviour of a transition relation, can be defined as follows.

Definition 3 [Polynomial time step-reliability] A transition relation $\vdash$ is called polynomial time step-reliable if there is a polynomial p such that for any one-step derivation D = S,S' in $\vdash$:

$$\mathrm{cost}(D) < p(\mathrm{size}(S)).$$

Note The development of data structures and algorithms for polynomial unification can be viewed as the attempt to achieve the polynomial time step-reliability of deduction systems using unification.

Unfortunately, polynomial time step-reliability is not sufficient for guaranteeing polynomial transparency. Additionally, a size increase condition is needed. The following very general one will do.

Definition 4 [Logarithmic polynomial size step-reliability] A transition relation I- is called logarithmic polynomial size step-reliable, or just logp size step-reliable, if there is an integer 6 > 1 and a polynomial p such that for every pair {S,S') €I-:

$$\mathrm{size}(S') < (\log_b p(\mathrm{size}(S))) + \mathrm{size}(S).$$

The following proposition (a proof can be found in [Letz, 1993]) is fundamental for the theory of abstraction modulo polynomials.

Proposition 1 If a transition relation I- is polynomial time step-reliable and logp size step-reliable, then I- is polynomially transparent.

2.5 Weak Polynomial Transparency

There are transition relations for which polynomial transparency cannot be guaranteed for arbitrary derivations, so that not in any case the input size and the steps of a derivation give a representative measure of its complexity. But, one may argue, when the indeterministic power of the transition relation defined by a logic calculus is concerned, then it is legitimate to consider solely those derivations which are shortest proofs of the inputs. The question is how to define 'short', in terms of elementary computing cost, in terms of derivation size, or number of steps. Also, the shortest proof, in anyone of these models, may violate the condition of polynomial transparency, but the second shortest may fit. In order to facilitate the formulation of reasonably tolerant generalizations of polynomial transparency, we define minimal proofs with respect to polynomials.

Definition 5 [p-(step-)minimal proof] Given a proof relation $\vdash$ and a polynomial p. A proof D of an input state S in $\vdash$ is said to be p-minimal in $\vdash$ if for any proof D' of 5 in I-:

$$\mathrm{cost}(D) < p(\mathrm{cost}(D'));$$

and D is called p-step-minimal in I- if for any proof D' of S in $\vdash$:

$$\mathrm{steps}(D) < p(\mathrm{size}(S), \mathrm{steps}(D')).$$

4Also, the concept of polynomial transparency leads to a natural generalization of the notion of a realistic machine model. By a generalized realistic machine model we can understand any computation model which, as a transition relation, is polynomially transparent and has the expressive power of Turing machines.

Now, polynomial difference in complexity poses no problems, not the *absolutely* shortest proof must be taken, any proof will do which p-simulates the shortest one. Using p-step-minimal proofs the notion of polynomial transparency can be weakened as follows.

Definition 6 [Weak polynomial (step-)transparency] A proof relation I- is called *weakly polynomially step-transparent* or just *weakly polynomially transparent* if there are polynomials *p* and *p'* such that for every input state *S* there exists a p-step-minimal proof *D* of *S* in I- with

cost(D) < p'(size(5),steps(D)).

Note One could even be more liberal and only demand the existence of p-minimal proofs in the definition above. We think that the resulting notion would become too weak, for the following reason. With the notion of weak polynomial transparency we intend to express that the abstraction level of inference steps indeed provides a representative complexity measure for the indeterministic power of a proof relation, even though not for the *absolutely* shortest proofs, so at least for *one* of the short proofs. But the class of short proofs should be *defined* in terms of inference steps, this way demonstrating the usefulness of the abstraction level.

3 Transparency Properties of Resolution

As a typical representative of a logic calculus relying on the use of *lemmata,* we shall study the transparency properties of the *resolution calculus,*[5] The resolution calculus [Robinson, 1965] can be formulated as a system of a single but complex inference rule of the shape

$$\frac{\{A_1,\ldots,A_m\}\cup r_1 \qquad\qquad \{\neg B_1,\ldots,\neg B_n\}\cup r_2}{r_1\theta \cup r_2\theta}$$

where θ is a most general unifier for the set of atomic formulae $\{A_1,\ldots,A_m,B_1,\ldots,B_n\}$, and r_1 and r_2 are clauses; implicitly assumed is the renaming of the variables in one of the input clauses of the rule.

Let us illustrate at this example the distinction between the declarative and the process interpretation of a deduction. While a deduction of the former type is simply a sequence *D* of clauses where each element of *D* is either from the input set or derived from earlier elements of D, the deduction process consists of a sequence of increasing clause sets. If the deduction process is based on unrestricted resolution—which is free of reduction rules like *subsumption deletion*—, then any state of the deduction process is just a deduction of the declarative type. This property holds for all calculi which are *accumulative*[6]

[5]The transparency properties of other calculi are studied in [Letz, 1993].

[6]In general, however, the states of a deduction process need not represent declarative deduction objects, even if no reduction rules, like subsumption deletion, are applied. This example also exhibits a certain disadvantage of measuring the size of a deduction as the sum of the sizes of the states in the deduction process, since *untouched* parts of the states are counted multiply. A finer model would count only the touched parts of the non-initial states.

3.1 Resolution and Polynomial Transparency

While polynomial size-transparency can be guaranteed for resolution—provided polynomial unification algorithms are used—, it is evident that resolution is not polynomially transparent.

Proposition 2 *Resolution for first-order logic is not polynomially transparent.*

Proof Consider the following formula *F* consisting of three clauses[7] of the shape

Example 1 $\neg P(x)$
$P(s(x)) \vee \neg P(x)$
$P(0)$

where 0 denotes a constant. By performing self-resolution on the second clause *co* of *F* and then repeatedly applying self-resolution to the deduced resolvents, in *k* steps one can generate a clause ck of size $> 2^k$. From c_k the empty clause can be deduced in two further resolution steps. Clearly for any polynomial *p* there exists a proof $D = D_1,\ldots,D_m$ of this type such that size$(D) > p(\text{size}(F), m)$, that is, the size of *D* cannot be bounded by any polynomial on the size of the input formula and the number of resolution steps. D

Consequently, in contrast to propositional logic, for first-order logic the number of resolution steps is not an adequate measure for the complexities of resolution derivations and proofs. The apparent reason is the following. Due to the renaming of derived clauses, resolution violates the logp size step-reliability.[8]

But one may object that a resolution proof of the specified type is not an optimal one, and that there exists a shorter resolution proof for *F* which immediately derives the empty clause, by simply resolving the two facts $\neg P(x)$ and $P(0)$. For this short proof the relation between the proof size and the proof steps is polynomial modulo the input size.

3.2 Resolution and Weak Polynomial Transparency

The question is now whether for *shortest* resolution proofs the sizes and the inference steps are always polynomially related, or in our terminology, whether *weak* polynomial transparency can be guaranteed for resolution. Unfortunately, the answer to this question is no, too. There is an infinite formula class for which *every* resolution proof is exponential in size with respect to the input formula, whereas there are proofs that get by on polynomially many resolution steps.

The next example specifies a formula class with this property. Assume in the following that, for any 1 < * <

[7] For reasons of readability, we prefer to write clauses as disjunctions of literals; furthermore, we use the term 'formula' for sets of clauses.

[8]It should be emphasized that the reason is indeed the *renaming* of derived clauses and not their *multiple use* as parent clauses.

n, $\mathfrak{P}_i$ is the value of the i-th prime number, and that $s^k(x)$ abbreviates a term of the structure $\underbrace{s(\cdots s(}_{k-\text{times}} x)\cdots)$.

Example 2 For any positive integer n, let Fn denote a Horn formula consisting of the following structure:

$$\begin{array}{l} \neg P_1(s(x)) \vee \cdots \vee \neg P_n(s(x)) \\ P_1(s^{\mathfrak{P}_1}(x)) \vee \neg P_1(x) \\ \quad\cdots \\ P_n(s^{\mathfrak{P}_n}(x)) \vee \neg P_n(x) \\ P_1(0) \\ \quad\cdots \\ P_n(0). \end{array}$$

If in this class of Horn formulae the function symbol 8 is interpreted as the successor function, and if the denotation of a predicate P, is the set of natural numbers divisible by the i-th prime number, then such a formula can be used to compute common multiples of primes. Apparently, from these considerations we can derive the following lemma.

Lemma 3 Given a formula Fn of the type specified in Example 2, let $c\theta$ be any ground instance of the first clause c $\in$ Fn such that $(F_n \setminus \{c\}) \cup \{c\theta\}$ is unsatisfiable. Then the largest occurring term in c0 must denote a common multiple of the first n prime numbers.

Since the least common multiple of a sequence $\mathfrak{P}_1, \ldots, \mathfrak{P}_n$ of primes equals $\prod_{i=1}^n \mathfrak{P}_i$, the following obvious result gains importance.

Lemma 4 There is no polynomial p such that for every positive integer n:

$$p(\sum_{i=1}^n \mathfrak{P}_i) > \prod_{i=1}^n \mathfrak{P}_i.$$

An immediate consequence of this result is that $\prod_{i=1}^n \mathfrak{P}_i$ cannot be polynomially bounded by the size .of the input formula Fn.

Lemma 5 There is no polynomial p such that for every positive integer n: p(size$(F_n)) > \prod_{i=1}^n \mathfrak{P}_i$, where Fn is a formula of the type specified in Example 2.

The formula class described in Example 2 is intractable for resolution.

Proposition 6 There is no polynomial p such that for every positive integer n: p(size(Fn)) 25 greater than the size of any resolution refutation for Fn.

In the proof of this proposition we shall exploit the fact that the formula class in question consists of Horn clause formulae, for which the following lemma holds.

Lemma 7 // / is a resolution refutation dag9 for a Horn clause formula, then t contains one branch b — called the negative branch—on which exactly the negative clauses of the refutation lie, i.e., those clauses which are void of positive literals.

9A resolution dag (directed acyclic graph) is a rooted dag whose nodes are labelled with clauses such that every non-leaf node N has two outgoing edges to nodes N\, N2, and the clause at N is a resolvent of the clauses at N1 and N2 A resolution refutation dag is a resolution dag whose root is labelled with the empty clause.

Proof of Lemma 7 It suffices to notice that, on the one hand, in such a dag no non-negative clause can dominate a negative clause, and, on the other hand, every negative clause must be derived from a negative and a non-negative clause.

Proof of Proposition 6 Let t be an arbitrary resolution refutation dag for a formula Fn, and let b be the negative branch of t, which exists by Lemma 7. Clearly, each occurrence of a negative clause on 6 is used only once as a parent clause in t. Consequently, replacing all clauses on the branch 6 by appropriate ground instances does not alter the length of the branch, while the resulting dag remains a refutation—of resolution with free, i.e., not necessarily most general, unification rule. If this partial instantiation is performed on t, the negative branch b' of the resulting refutation dag t' must contain ground instances

$$\neg P_1(s^\xi(0)) \vee \cdots \vee \neg P_n(s^\xi(0))$$

of the first clause $c \in F_n$. Let $c_0, \ldots, c_k$ be the clauses on the initial segment of the branch b' from the root labelled with c_0 (the empty clause) up to the first instance c_k of c. Obtain t'' by making c_k a leaf of t' (it may already be one) plus removing the nodes and edges which are no more accessible from the root. Apparently, t'' still remains a refutation dag. Since c_k is the only instance of c in t'', $(F_n \setminus \{c\}) \cup \{c_k\}$ must be unsatisfiable. From Lemma 3 follows that in c_k the maximal term depth $\xi \geq \prod_{i=1}^n \mathfrak{P}_i$. Consider now the non-negative clauses $s_1, \ldots, s_k$ in the refutation t'' which are resolution partners of the clauses $c_1, \ldots, c_k$ respectively—let us call those non-negative clauses the *side* clauses. The structure of F_n guarantees that each side clause either has the form

$$P_i(s^l(x)) \vee \neg P_i(x)$$

or the form

$$P_i(s^l(0)).$$

Consequently, if ascending the branch b' by one step towards the root from c_i to c_{i-1}, $1 \leq i \leq k$, the clause size can only decrease by at most the size of the respective side clause s_i (disregarding the logical symbols):

$$\text{size}(c_{i-1}) \geq \text{size}(c_i) - \text{size}(s_i).$$

Therefore,

$$\text{size}(c_0) \geq \text{size}(c_k) - \sum_{i=1}^k \text{size}(s_i).$$

Because $\text{size}(c_0) = 1$, and since the side clauses have not been modified by the partial instantiation operation, we get that

$$\text{size}(t) > \sum_{i=1}^k \text{size}(s_i) \geq \text{size}(c_k) - 1 > \prod_{i=1}^n \mathfrak{P}_i.$$

An application of Lemma 5 completes the proof. □

The existence of intractable formula classes for resolution is nothing exceptional, even for the propositional

case (at least since Haken's work [Haken, 1985]). The special property of the formula class considered here concerns the relation between the proof sizes and the numbers of derivation steps. Although all resolution proofs for the formula class are superpolynomial, there are short proofs in terms of *inference steps*.

Proposition 8 *There is a polynomial p such that for every formula F_n from the class specified in Example 2 there exists a resolution refutation $D_1, \ldots, D_m$ of F_n such that $m < p(\text{size}(F_n))$.*

Proof Let $\xi = \prod_{i=1}^{n} \mathfrak{P}_i$, i.e., the least common multiple of the primes $\mathfrak{P}_1, \ldots, \mathfrak{P}_n$. Then a polynomial-step proof can be constructed as follows. For every clause of the type

$$P_i(s^{\mathfrak{P}_i}(x)) \vee \neg P_i(x)$$

perform self-resolution and repeatedly apply self-resolution to the respective resolvents. Within k steps this operation deduces clauses in which the number of occurrences of the function symbol s in the positive literals successively takes the values $\mathfrak{P}_i\,2^1, \mathfrak{P}_i\,2^2, \ldots, \mathfrak{P}_i\,2^k$. This is done as long as $\mathfrak{P}_i\,2^k \leq \xi$. Then, after at most k further resolution steps which use clauses from this derivation, each clause at most once, a formula of the structure

$$P_i(s^{\xi}(x)) \vee \neg P_i(x)$$

can be deduced. Accordingly, for any $1 \leq i \leq n$, we need at most $2\log_2 \frac{\xi}{\mathfrak{P}_i}$ steps, which is less than $2\log_2 \xi$, hence for all i: less than $2n\log_2 \xi$. Lastly, in further $2n$ resolution steps the empty clause can be derived by resolving these clauses with the facts and the resulting n facts $P_i(s^{\xi}(0))$, $1 \leq i \leq n$, with the first clause. The whole refutation takes less than $2n + (2n\log_2 \xi) \leq 4n\log_2 \xi$ steps. It remains to be shown that this value is polynomially bounded by the size of F_n. For this purpose we may just use $\zeta = \sum_{i=1}^{n} \mathfrak{P}_i$ as a lower bound for the size of F_n and consider the chain

$$4n\log_2 \prod_{i=1}^{n} \mathfrak{P}_i \leq 4n\log_2 \left(\frac{\sum_{i=1}^{n} \mathfrak{P}_i}{n}\right)^n = 4n^2 \log_2 \frac{\zeta}{n} < 4\zeta^3.$$

The first inequality holds because of properties of the arithmetical mean, while the others are trivial. □

The Propositions 6 and 8 have as an immediate consequence that, even if only step-minimal proofs are considered, the number of steps of a resolution proof may not be a representative measure for the complexity of the proof.

Theorem 9 *Resolution for first-order logic is not weakly polynomially transparent.*

The violation of the logp size step-reliability turns out to be fatal, even if only short proofs are counted.

3.3 Methods for Obtaining Polynomial Transparency

The situation is quite instructive, because we can illustrate at the example of resolution the three principal solution methodologies when facing the polynomial intransparency of a transition relation K

The first approach is to weaken the transition relation ⊢ and to define a transition relation I-', for example, by taking out each pair (S, *S')* which violates the logp size step-reliability, since this may be the problematic property, like in the case of resolution. The most radical method to perform this modification on the resolution calculus is to forbid the renaming or even the multiple use of lemmata. The latter results in the calculus of *tree resolution.* Tree resolution is polynomially transparent, even in the strict sense (a proof can be found, e.g., in [Eder, 1992] or in [Letz, 1993]). But unfortunately, this has the unacceptable consequence that many proofs are thrown out which are short in steps *and* small in size.

Also, eliminating problematic pairs from a transition relation does not work for arbitrary transition relations. This leads to the second alternative. In order to preserve the problem solving functionality of the relation, that is, to guarantee that the transitive closures—or at least the provable states—of both transition relations remain identical, in the general case, each problematic step must be replaced by a series of computationally innocuous steps. For logic calculi, this amounts to a redefinition of the notion of an inference step.

Both methods are relatively unappealing for the practical working with logic calculi, since in no case the w-*determintstic power* of a calculus is increased, either it is weakened or it remains unchanged, and only the presentation structure of the calculus is modified.

The *real* importance of the notion of polynomial transparency for the advance of science is that it can motivate research following the third approach. The third approach is to let the general structure of a transition relation as it is, and to try to remedy the polynomial intransparency of the transition relation. Since the typical stumble-block for attaining polynomial transparency is the violation of the logp size step-reliability, a promising research direction consists of improving the *data structures* of the elements in the transition relation in such a way that they can be represented with less space than in the original relation, with the hope to gain polynomial transparency this way. The advantage of such an attempt, if it succeeds, is that the distances between the elements in the transition relation can be preserved while the real computing cost and sizes properly decrease.

The difference between the solution methodologies is that the second approach always succeeds, whereas the third one may fail in principle.[10]

3.4 Improvements of the Representation of Formulae

Similar to the case of the unification operation, which, in order to attain the polynomial time step-reliability of an inference system, has enforced the necessity to represent logical terms as dags, one should think about the development of more sophisticated mechanisms which would admit a notation for resolvents polynomially bounded in

[10] Such an impossibility result for resolution is proven in [Letz, 1993]. There it is shown that there can be no data structure for achieving the strong polynomial transparency of resolution, if not the *factoring rule* is modified. This result indeed enforces a redefinition of the resolution inference step.

size by the number of their derivation steps, with respect to the input formula.

An obvious improvement is to integrate into the object language the same vocabulary of upper indices we already used in our meta-language for the purpose of polynomially specifying terms of exponential depth. It is apparent that with the use of such number terms the transparency problems of the Examples 1 and 2 can be solved, even polynomial transparency in the strong sense can be achieved for these examples. One can predict that number terms will play an important role in future automated deduction systems.[11]

We shall not pursue further the attempt of extending the representation of logical formulae, instead we want to present a critical example class which may turn out to be a hard problem for the efforts to achieve polynomial transparency- These new formulae are obtained from the previous class of Example 2 by augmenting the arity of the function symbol s by 1. This means that the previous formula class is just an abstraction of the new class.

Example 3 For any positive integer n, let F_n denote a Horn formula of the following structure:

$$\begin{array}{l}
\neg P_1(s(x,y)) \vee \cdots \vee \neg P_n(s(x,y)) \\
P_1(s(s(x,y_1),y_2)) \vee \neg P_1(x) \\
P_2(s(s(s(x,y_1),y_2),y_3)) \vee \neg P_2(x) \\
P_3(s(s(s(s(x,y_1),y_2),y_3),y_4),y_5)) \vee \neg P_3(x) \\
\quad \cdots \\
P_n(\underbrace{s(s(\cdots s(s(}_{\mathfrak{p}_n-\text{times}}x,y_1),y_2),\cdots,y_{n-1}),y_n)) \vee \neg P_n(x) \\
\\
P_1(0) \\
\quad \cdots \\
P_n(0).
\end{array}$$

In the new class the second argument of the function symbol s does not play any role at all, the variables at these positions are just dummy variables. Consequently, the results concerning proof steps and proof lengths carry over from Example 2 to this example. But there is a crucial difference between both examples, which becomes apparent when self-resolution is applied to a clause of the mixed type in Example 3. Let us demonstrate this with the input clause corresponding to the prime number 3:

$$P_2(s(s(s(x,y_1),y_2),y_3)) \vee \neg P_2(x).$$

In its self-resolvent

$$P_2(s(s(s(s(s(s(x,y_1),y_2),y_3),y_4),y_5),y_6)) \vee \neg P_2(x)$$

the number of distinct dummy variables has doubled. In general, in any such self-resolution step the resolvent contains 2n — 1 more distinct variables than the original clause. Accordingly, for this class of formulae, in any polynomial-step proof of an instance F_n, clauses are needed in which not only the term depth is exponential (which could be remedied by using number terms in the object language) but also the number of distinct variables. And to this problem no obvious solution is in sight.[12]

Conclusion and Further Research

This paper has illustrated that a formalization of intuitively existing abstraction concepts for logic calculi can be very fruitful. The developed notion of polynomial transparency promises to serve as a useful and research-stimulating property of deduction systems, and of transition relations in general. The main technical result of this paper is the demonstration that the number of inference steps of resolution proofs do not give a representative measure of the actual proof complexities, even if only shortest proofs are considered.

As further obvious research perspectives concerning the transparency problem of resolution and other calculi using the renaming of lemmata, we wish to mention the development of more sophisticated data structures than number terms; the clarification of the relation between strong and weak polynomial transparency; and finally, the study of the difficulties of rendering particular calculi polynomially transparent and the connection of these difficulties with certain problem classes in the polynomial hierarchy.

References

[Cook and Reckhow, 1974] S. A. Cook and R. A. Reckhow. On the Lengths of Proofs in the Propositional Calculus. Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, Seattle, Washington, pp. 135-148, 1974 (corrections are in SIGACT News 6(3):15-22, 1974).

[Eder, 1992] E. Eder. Relative Complexities of First-Order Calculi. Vieweg, 1992.

[Haken, 1985] A. Haken. The Intractability of Resolution. Theoretical Computer Science, 39:297-308, 1985.

[Letz, 1993] R. Letz. First-Order Calculi and Proof Procedures for Automated Deduction. PhD thesis, Technische Hochschule Darmstadt, to appear in Summer 1993.

[Reckhow, 1976] R. A. Reckhow. On the Lenghts of Proofs in the Propositional Calculus. PhD thesis, University of Toronto, 1976.

[Robinson, 1965] J. A. Robinson. A Machine-oriented Logic Based on the Resolution Principle. Journal of the Association for Computing Machinery, 12:23-41, 1965.

[van Emde Boas, 1990] P. van Emde Boas. Machine Models and Simulations. Handbook of Theoretical Computer Science, pages 1-66, Elsevier Science Publishers, 1990.

[11] Much more than polynomial unification algorithms, which have turned out to be relatively unimportant for the practice of deduction systems. This can be verified by observing that the examples (particularly Example 1) for demonstrating the necessity of number terms are much simpler and occur more frequently in practice than the ones which demand polynomial unification techniques.

[12] There seems to be an interesting analogy between decidability and complexity properties with respect to the distinction of clause formulae containing unary function symbols only from those containing binary function symbols. While the former are decidable and permit the successful application of number terms, the latter axe undecidable and polynomial transparency cannot be achieved using number terms.