A BOUND FOR DICKSON'S LEMMA

JOSEF BERGER AND HELMUT SCHWICHTENBERG

Mathematisches Institut der LMU München, Theresienstraße 39, 80333 München
e-mail address: {jberger,schwicht}@math.lmu.de

ABSTRACT. We consider a special case of Dickson's lemma: for any two functions f, g on the natural numbers there are two numbers $i < j$ such that both f and g weakly increase on them, i.e., $f_i \leq f_j$ and $g_i \leq g_j$. By a combinatorial argument (due to the first author) a simple bound for such i, j is constructed. The combinatorics is based on the finite pigeon hole principle and results in a descent lemma. From the descent lemma one can prove Dickson's lemma, then guess what the bound might be, and verify it by an appropriate proof. We also extract (via realizability) a bound from (a formalization of) our proof of the descent lemma.

1. INTRODUCTION

Consider the following special case of Dickson's lemma: for any two functions f, g on the natural numbers there are two numbers $i < j$ such that both f and g weakly increase on them, i.e., $f_i \leq f_j$ and $g_i \leq g_j$. By a combinatorial argument (due to the first author) a simple bound for such i, j is constructed. The combinatorics is based on the finite pigeon hole principle and results in a certain descent lemma. From the descent lemma one can prove Dickson's lemma, then directly guess what the bound might be, and finally verify it by an appropriate proof. We also extract (via realizability) a bound from (a formalization of) our proof of the descent lemma.

In its usual formulation, Dickson's lemma (for fixed functions) is a Σ_1^0-formula. In contrast, we shall prove a quantifier-free statement which implies Dickson's lemma in its usual form, but not vice versa. Our proof can be carried out in the formal system of Elementary Analysis [16, p.144], a conservative extension of Heyting arithmetic with variables and quantifiers for number-theoretic functions. In fact, we don't make use of the axiom of choice at all. Furthermore, we can restrict induction to quantifier-free formulas.

Dickson's lemma has many applications. For instance, it is used to prove termination of Buchberger's algorithm for computing Gröbner bases [4], and to prove Hilbert's basis theorem [14].

There are many other proofs of Dickson's lemma in the literature, both with and without usage of non-constructive (or "classical") arguments. The original proof of Dickson [5] and the particularly nice one by Nash-Williams [11] (using minimal bad sequences) are non-constructive, and hence do not immediately provide a bound. But it is well known

---

that by using some logical machinery one can still read off bounds, using either Gödel's [8] Dialectica translation as in Hertz [9] or Friedman's [7] A-translation as in [3]. However, these bounds – even for the case of just two functions considered here – heavily use higher type (primitive recursive) functionals and are less perspicious than the one obtained below.

The first constructive proof of Dickson's lemma has been given by Schütte and Simpson [12, 14], using ordinal numbers and transfinite induction up to ϵ_0. Similar methods have been used by Sustik [15] and Martín-Mateos et al. [10]. Since initial segments of transfinite induction are used, these proofs when written in arithmetical systems require ordinary induction on quantified formulas. A different constructive proof has been given by Veldman [17]. It uses dependent choice for Σ_1-formulas (with parameters), and induction on Π_2-formulas. This proof also provides the basis of Fridlender's [6] formalization in Agda. The computational content of these proofs has not been studied; the bound involved will be very different from the present one.

2. A COMBINATORIAL PROOF OF DICKSON'S LEMMA

We start with a finite pigeonhole principle, in two disjunctive forms. The (rather trivial) proofs are carried out because they have computational content which will influence the term extracted from a formalization of our proofs in Section 3.

Lemma 2.1 (FPHDisj). $\forall_{m,f}(\exists_{i<j\leq m} f_i = f_j \vee \exists_{j\leq m} m \leq f_j)$.

Proof. By induction on m. For $m = 0$ the second alternative holds. In case $m + 1$ let f_j be maximal among $f_0, \ldots, f_{m+1}$. If $m + 1 \leq f_j$ we are done. Else we have $f_j \leq m$. Now we apply the induction hypothesis to $f' := f_0, \ldots, f_{j-1}, f_{j+1}, \ldots, f_{m+1}$. If two of them are equal we are done. Else $m \leq f_k$ for some $k \neq j$ and hence $f_j \leq f_k$. If $f_j = f_k$ we are done. Else we have $f_j < f_k$, contradicting the choice of j. $\square$

Note that quantifier-free induction suffices here, since we only prove a property of finite lists of natural numbers.

In the key lemma 2.3 below we will need a somewhat stronger disjunctive version of the pigeonhole principle. To this end we need an injective coding $\langle n, m \rangle$ of natural numbers which is "square-filling", i.e. with the property

$$k^2 \leq \langle n, m \rangle \to k \leq n \vee k \leq m. \tag{2.1}$$

This can be achieved by

$$
\begin{array}{ccccc}
\ldots & & & & \\
12 & 13 & 14 & 15 & \ldots \\
6 & 7 & 8 & 11 & \ldots \\
2 & 3 & 5 & 10 & \ldots \\
0 & 1 & 4 & 9 & \ldots
\end{array}
$$

or explicitly

$$\langle n, m \rangle := \begin{cases} n^2 + m & \text{if } m < n, \\ m^2 + m + n & \text{otherwise.} \end{cases}$$

Lemma 2.2 (FPHDisj2).

$$\forall_{f,g,k}(\exists_{i<j\leq k^2}(f_i = f_j \wedge g_i = g_j) \vee \exists_{j\leq k^2} k \leq f_j \vee \exists_{j\leq k^2} k \leq g_j).$$

Proof. Fix f, g, k. Use Lemma 2.1 with $s_i := \langle f_i, g_i \rangle$ and $m := k^2$. In the first case from $s_i = s_j$ we obtain $f_i = f_j$ and $g_i = g_j$ by the injectivity of the coding. In the second case we have some $j \leq k^2$ with $k^2 \leq s_j$. From the square-filling property (2.1) of the coding we obtain $k \leq f_j$ or $k \leq g_j$. $\qquad\square$

As an immediate consequence we have

Lemma 2.3 (Key).
$$\forall_{f,g,n,k}(\exists_{n<i<j\leq n+k^2+1}(f_i = f_j \wedge g_i = g_j) \vee$$
$$\exists_{n<j\leq n+k^2+1} k \leq f_j \vee \exists_{n<j\leq n+k^2+1} k \leq g_j).$$

Proof. Use Lemma 2.2 for $\lambda_i f_{n+1+i}$, $\lambda_i g_{n+1+i}$ and k. $\qquad\square$

Now we introduce some notation. $\mathrm{Mini}(f, n)$ is the first argument where f is minimal on $\{0, \ldots, n\}$:
$$\mathrm{Mini}(f, 0) := 0,$$
$$\mathrm{Mini}(f, n+1) := \begin{cases} \mathrm{Mini}(f, n) & \text{if } f_{\mathrm{Mini}(f,n)} \leq f_{n+1}, \\ n+1 & \text{otherwise.} \end{cases}$$

We define functions Ψ, Φ, I and a formula D with arguments f, g, n. For readability f, g are omitted.
$$\begin{aligned} \Psi(n) &:= \max\{f_{\mathrm{Mini}(g,n)}, g_{\mathrm{Mini}(f,n)}\}, \\ \Phi(n) &:= f_{\mathrm{Mini}(f,n)} + g_{\mathrm{Mini}(g,n)}, \\ I(n) &:= n + \Psi(n)^2 + 1, \\ D(n) &:= \exists_{i<j\leq n}(f_i \leq f_j \wedge g_i \leq g_j). \end{aligned} \qquad (2.2)$$

$D(n)$ expresses that n is a bound for Dickson's lemma.

The next lemma states a crucial property of the function I: either $I(n)$ already is a bound for Dickson's lemma, or else Φ decreases properly when going from n to $I(n)$. Since this cannot happen infinitely often, iteration of I will finally give us the desired bound.

Lemma 2.4 (Descent). $D(I(n)) \vee \Phi(I(n)) < \Phi(n)$.

Proof. Use Lemma 2.3 with f, g, n and $\Psi(n)$. In the first case we have $D(I(n))$. In the second case we have $n < j \leq I(n)$ with $\Psi(n) \leq f_j$; the third case is symmetric. Let $i := \mathrm{Mini}(g, n)$. Then $f_i \leq \Psi(n)$. In case $g_i \leq g_j$ we have $D(I(n))$ and are done. Therefore assume $g_j < g_i$. We show (i) $\Phi(I(n)) \leq \Phi(j)$ and (ii) $\Phi(j) < \Phi(n)$. From $j \leq I(n)$ we obtain (i). For (ii) we show $f_{\mathrm{Mini}(f,j)} + g_{\mathrm{Mini}(g,j)} < f_{\mathrm{Mini}(f,n)} + g_i$. Now $n < j$ implies $f_{\mathrm{Mini}(f,j)} \leq f_{\mathrm{Mini}(f,n)}$, and $g_{\mathrm{Mini}(g,j)} \leq g_j < g_i$. $\qquad\square$

From Lemma 2.4 we construct a bound for Dickson's lemma. Let
$$I^0(n) := n, \quad I^{m+1}(n) := I(I^m(n)).$$

Lemma 2.5. $D(I^n(0)) \vee \Phi(I^n(0)) + n \leq \Phi(0)$.

Proof. Induction on n. Step $n \mapsto n+1$. Applying Lemma 2.4 to $I^n(0)$ gives $D(I^{n+1}(0)) \vee \Phi(I^{n+1}(0)) < \Phi(I^n(0))$. In the second case we have
$$\Phi(I^{n+1}(0)) + n + 1 < \Phi(I^n(0)) + n + 1 \leq \Phi(0) + 1$$
The latter inequality follows from the induction hypothesis, since $D(I^n(0))$ implies $D(I^{n+1}(0))$. $\qquad\square$

Proposition 2.6. $D(I^{f_0+g_0+1}(0))$.

Proof. Apply Lemma 2.5 to $\Phi(0) + 1$.   □

This bound is far from optimal: already for

$$f_n := \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{else} \end{cases} \qquad g_n := 0$$

with optimal bound 2 we have

$$I^{f_0+g_0+1}(0) = I^2(0) = I(I(0)) > I(0) = \Psi(0)^2 + 1 = 2.$$

Can we extend this proof to show Dickson's lemma for finitely many functions? For instance for three functions a corresponding version of the key lemma holds:

$$\forall_{f,g,h,n,k}(\exists_{n<i<j\leq n+k^4+1}(f_i = f_j \wedge g_i = g_j \wedge h_i = h_j) \vee$$
$$\exists_{n<j\leq n+k^4+1}k \leq f_j \vee \exists_{n<j\leq n+k^4+1}k \leq g_j \vee \exists_{n<j\leq n+k^4+1}k \leq h_j)$$

(Proof. Apply the original key lemma to $\langle f, g\rangle, h, n$ and k^2). We can also define a measure function $\Phi(n) := f_{\mathrm{Mini}(f,n)} + g_{\mathrm{Mini}(g,n)} + h_{\mathrm{Mini}(h,n)}$. A natural candidate for Ψ is

$$\Psi(n) := \max\{f_{\mathrm{Mini}(g,n)}, f_{\mathrm{Mini}(h,n)}, g_{\mathrm{Mini}(f,n)}, g_{\mathrm{Mini}(h,n)}, h_{\mathrm{Mini}(f,n)}, h_{\mathrm{Mini}(g,n)}\}$$

and a natural candidate for I is $I(n) := n + \Psi(n)^4 + 1$. But the corresponding version of the descent lemma is false: let $n := 2$ and

$$f := (0, 1, 1, 1, 0, f_5, \dots),$$
$$g := (1, 0, 1, 0, 1, g_5, \dots),$$
$$h := (1, 1, 0, 0, 0, h_5, \dots).$$

Then $\Phi(n) = 0$, $\Psi(n) = 1$, $I(n) = 4$, and we neither have $D(I(n))$ nor $\Phi(I(n)) < \Phi(n)$. – However, it may well be that a more refined form of the present approach works. We leave this for future research.

3. Extracting computational content

In the following, we demonstrate how a bound for Dickson's lemma can be extracted from a proof of the existence of such a bound. The proof we will use is essentially the one presented in Section 2, i.e., it is based on the descent lemma 2.4. We will then apply the realizability interpretation to obtain the bound. In fact, the bound will be *machine* extracted from a formalization of the existence proof.

In more detail, we shall use that I is increasing (i.e., $n < I(n)$) and that from $D(n)$ and $n < m$ we can infer $D(m)$. Then we prove the existence of a bound by general induction with measure Φ.

3.1. **General induction and recursion.** We first explain general induction w.r.t. a measure, and the corresponding definition principle of general recursion.

General induction allows recurrence to all points "strictly below" the present one. In applications it is best to make the necessary comparisons w.r.t. a measure function μ; for simplicity we restrict ourselves to the case where μ has values in the natural numbers, and the ordering we refer to is the standard $<$-relation. The principle of general induction then is

$$\forall_{\mu,x}(\text{Prog}_x^\mu Px \to Px),$$

where $\text{Prog}_x^\mu Px$ expresses "progressiveness" w.r.t. μ and $<$, i.e.,

$$\text{Prog}_x^\mu Px := \forall_x(\forall_y(\mu y < \mu x \to Py) \to Px).$$

It is easy to see that in our special case of the $<$-relation we can prove general induction from structural induction. However, it will be convenient to use general induction as a primitive axiom, for then the more efficient general recursion constant $\mathcal{F}$ will be extracted. It is defined by

$$\mathcal{F}\mu xG = Gx(\lambda_y[\textbf{if } \mu y < \mu x \textbf{ then } \mathcal{F}\mu yG \textbf{ else } \varepsilon]),$$

where ε denotes a canonical inhabitant of the range. It is easy to prove that $\mathcal{F}$ is definable from an appropriate structural recursion operator.

3.2. **Non-computational quantifiers.** We now use general induction in our constructive proof of Dickson's lemma. However, we have to be careful with the precise formulation of what we want to prove. We are not interested in the pair i, j of numbers where both f and g increase, but only in a bound telling us when at the latest this must have happened. Therefore the existential quantifiers $\exists_{i,j}$ must be made "uniform" (i.e., non-computational); it will be disregarded in the realizability interpretation. Such non-computational quantifiers have first been introduced in [1, 2]; in [13] this concept is extended to all connectives and discussed in detail. Let

$$D'(n) := \exists_{i<j\le n}^{\text{u}}(f_i \le f_j \wedge g_i \le g_j).$$

Using this non-computational form of $D(n)$ we modify Lemma 2.4 to

Lemma 3.1 (Descent$^{\text{nc}}$)**.** $D'(I(n)) \vee \Phi(I(n)) < \Phi(n)$.

Note that the computational content of a proof of this lemma is that of a functional mapping two unary functions and a number into a boolean. From Lemma 3.1 we obtain as before a modification of Proposition 2.6 to

Proposition 3.2 (Bound for Dickson's lemma)**.**

$$\forall_{f,g,n}\exists_k(I(n) \le k \wedge D'(k)).$$

Proof. By general induction with measure function Φ. We fix f, g and prove progressiveness of the remaining $\forall_n\exists_k$-formula. Therefore we can assume as induction hypothesis that for all m with $\Phi(m) < \Phi(n)$ we have

$$\exists_k(I(m) \le k \wedge D'(k)).$$

We must show

$$\exists_k(I(n) \le k \wedge D'(k)).$$

By Lemma 3.1 we know $D'(I(n)) \vee \Phi(I(n)) < \Phi(n)$. In the first case we have $D'(I(n))$ and can take $k := I(n)$. In the second case we apply the induction hypothesis to $I(n)$. It provides a k with $I(I(n)) \le k$ and $D'(k)$. But $I(n) \le I(I(n))$ since $n < I(n)$. $\qquad\square$

3.3. **Formalization and extraction.** The formalization[1] (in Minlog[2]) of the proof above is now routine. The term extracted from it is

```
[f,g,n](GRecGuard nat nat)(Phi f g)n
([n0,f1][if (cDesc f g n0) (I f g n0) (f1(I f g n0))])
True
```

To explain this term we rewrite it in the notation above

$$\lambda_{f,g,n}\mathcal{F}\mu n G$$

with measure μ and step function G defined by

$$\mu := \Phi,$$

$$G(n,h) := \begin{cases} I(n) & \text{if } \texttt{cDesc}(n), \text{ i.e., } D'(I(n)), \\ h(I(n)) & \text{otherwise, i.e., } \Phi(I(n)) < I(n), \end{cases}$$

where for readability we again omit the arguments f, g from $\Phi, I, \texttt{cDesc}$. The functions Φ, I are defined as in (2.2), and $\texttt{cDesc}$ is the computational content of Lemma 3.1:

```
[f,g,n][case (cKey f g n(f(Mini g n)max g(Mini f n)))
  ((DummyL nat ysum nat) -> True)
  (Inr nn ->
  [case nn
    ((InL nat nat)n0 ->
     (cNatLeLtCases boole)(g(Mini g n))(g n0)True False)
    ((InR nat nat)n0 ->
     (cNatLeLtCases boole)(f(Mini f n))(f n0)True False)])]
```

Here **nn** is a variable of type $\mathbf{N}+\mathbf{N}$ with $\mathbf{N}$ the type of natural numbers, and `cNatLeLtCases`:

```
(Rec nat=>nat=>alpha=>alpha=>alpha)n
([n0,x,x0][case n0 (0 -> x0) (Succ n1 -> x)])
([n0,h,n1,x,x0][case n1 (0 -> x) (Succ n2 -> h n2 x x0)])
```

is the computational content of the (simple) proof of

$$\forall_{n,m}((n \le m \to P) \to (m < n \to P) \to P)$$

expressing case distinction w.r.t. $\le$ and $<$.

cKey is the computational content of Lemma 2.3:

```
[f,g,n,n0]
[case (cFPHDisjTwo([n1]f(Succ(n+n1)))([n1]g(Succ(n+n1)))n0)
  ((DummyL nat ysum nat) -> (DummyL nat ysum nat))
  (Inr nn ->
  Inr[case nn
      ((InL nat nat)n1 -> (InL nat nat)(Succ(n+n1)))
      ((InR nat nat)n1 -> (InR nat nat)(Succ(n+n1)))])]
```

which uses `cFPHDisjTwo`:

```
[f,g,n][if (cFPHDisj(n*n)
      ([n0][if (g n0<f n0)
              (f n0*f n0+g n0)
```

---

[1]Available at `git/minlog/examples/arith/dickson.scm`

[2]See `http://www.minlog-system.de`

```
                  (g n0*g n0+g n0+f n0)])))
  ([ij](DummyL nat ysum nat))
  ([n0]
   Inr[if (cCodeSqFill(f n0)(g n0)n)
          ((InL nat nat)n0)
          ((InR nat nat)n0)])]
```

which in turn depends on cCodeSqFill:

```
[n,n0,n1](Rec nat=>nat=>boole)n([n2]False)
([n2,(nat=>boole),n3]
  [case n3 (0 -> True) (Succ n -> (nat=>boole)n)])
n0
```

and cFPHDisj:

```
[n](Rec nat=>(nat=>nat)=>nat@@nat ysum nat)n
([f](InR nat nat@@nat)0)
([n0,d,f]
  [let n1
    [if (f(Succ n0)<=f(Maxi f n0)) (Maxi f n0) (Succ n0)]
    [if (Succ n0<=f n1)
     ((InR nat nat@@nat)n1)
     [if (d([n2][if (n2<n1) (f n2) (f(Succ n2))]))
      ([ij]
        (InL nat@@nat nat)
        [if (right ij<n1)
          ij
          ([if (left ij<n1)
                (left ij)
                (Succ left ij)]@Succ right ij)])
      ([n2]
        [if (n2<n1)
          ((cNatLeCases nat@@nat ysum nat)(f n2)(f n1)
          ((InL nat@@nat nat)(0@0))
          ((InL nat@@nat nat)(n2@n1)))
          ((cNatLeCases nat@@nat ysum nat)(f(Succ n2))(f n1)
          ((InL nat@@nat nat)(0@0))
          ((InL nat@@nat nat)(n1@Succ n2)))])]]]])
```

To summarize, we have extracted a function which takes two functions f, g (suppressed for readability) and a number n and yields a bound. Notice that already with $n = 0$ we obtain the desired bound for Dickson's lemma. However, the inductive argument requires the general formulation.

Our extracted bound $B(n) := \mathcal{F}\Phi nG$ satisfies

$$B(n) = \mathcal{F}\Phi nG = Gn(\lambda_m[\textbf{if } \Phi m < \Phi n \textbf{ then } \mathcal{F}\Phi mG \textbf{ else } \varepsilon])$$

$$= \begin{cases} I(n) & \text{if } D'(I(n)), \\ B(I(n)) & \text{if } \Phi(I(n)) < I(n). \end{cases}$$

by Lemma 3.1, which also guarantees termination: $B(n)$ will call itself at most $I(n)$ times. As long as the iterations $I(n)$, $I^2(n)$, ..., $I^m(n)$ decrease w.r.t. the measure Φ, the next iteration step is done. However, as soon as Lemma 3.1 goes to its "left" alternative (i.e., $D'(I(n))$ holds), $I(n)$ is returned. Hence this extracted bound differs from the "guessed" one in Proposition 2.6 in that it does not iterate I a prescribed number of times ($f_0 + g_0 + 1$ many) at 0, but stops when allowed to do so by the outcome of Lemma 3.1.

References

[1] U. Berger. Program extraction from normalization proofs. In M. Bezem and J. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 91–106. Springer Verlag, Berlin, Heidelberg, New York, 1993.

[2] U. Berger. Uniform Heyting arithmetic. *Annals of Pure and Applied Logic*, 133:125–148, 2005.

[3] U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.

[4] B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes Mathematicae*, 4:374–383, 1970.

[5] L. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Am. J. Math*, 35:413–422, 1913.

[6] D. Fridlender. *Higman's Lemma in Type Theory*. PhD thesis, Chalmers University of Technology, Göteborg, 1997.

[7] H. Friedman. Classically and intuitionistically provably recursive functions. In D. Scott and G. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978.

[8] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.

[9] A. Hertz. A constructive version of the Hilbert basis theorem. `http://www.andrew.cmu.edu/user/avigad/Students/hertz.pdf`, 2004.

[10] F.-J. Martín-Mateos, J.-L. Ruiz-Reina, J.-A. Alonso, and M.-J. Hidalgo. Proof pearl: A Formal Proof of Higman's Lemma in ACL2. *Journal of Automatic Reasoning*, 47(3):229–250, 2011.

[11] C. Nash-Williams. On well-quasi-ordering finite trees. *Proc. Cambridge Phil. Soc.*, 59:833–835, 1963.

[12] K. Schütte and S. G. Simpson. Ein in der reinen Zahlentheorie unbeweisbarer Satz über endliche Folgen von natürlichen Zahlen. *Archiv für Mathematische Logik und Grundlagenforschung*, 25:75–89, 1985.

[13] H. Schwichtenberg and S. S. Wainer. *Proofs and Computations*. Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.

[14] S. Simpson. Ordinal Numbers and the Hilbert Basis Theorem. *The Journal of Symbolic Logic*, 53:961–974, 1988.

[15] M. Sustik. Proof of Dickson's Lemma using the ACL2 theorem prover via an explicit ordinal mapping. In *Proceedings of the 4th International Workshop on the ACL2 Theorem Prover and its Applications*, 2003.

[16] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction*, volume 121, 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1988.

[17] W. Veldman. An Intuitionistic Proof of Kruskal's Theorem. *Archive for Mathematical Logic*, 43(2):215–264, 2004.