Van der Waerden's Theorem

Katharina Kreuzer, Manuel Eberl

March 17, 2025

Abstract

This article formalises the proof of Van der Waerden's Theorem from Ramsey theory.

Van der Waerden's Theorem states that for integers k and l there exists a number N which guarantees that if an integer interval of length at least N is coloured with k colours, there will always be an arithmetic progression of length l of the same colour in said interval. The proof goes along the lines of Swan [1].

The smallest number $N_{k,l}$ fulfilling Van der Waerden's Theorem is then called the Van der Waerden Number. Finding the Van der Waerden Number is still an open problem for most values of k and l.

Contents

1 Representation of integers in different bases **3**

2 Van der Waerden's Theorem **5**

2.1 Arithmetic progressions 5

2.2 Van der Waerden's Theorem 7

```
theory Digits
  imports Complex_Main
begin
```

1 Representation of integers in different bases

First, we look at some useful lemmas for splitting sums.

```
lemma split_sum_first_elt_less: assumes "n<m"
  shows "(∑ i∈{n..<m}. f i) = f n + (∑ i∈{Suc n ..<m}. f i)"
  ⟨proof⟩

lemma split_sum_mid_less: assumes "i<(n::nat)"
  shows "(∑ j<n. f j) = (∑ j<i. f j) + (∑ j=i..<n. f j)"
⟨proof⟩
```

In order to use representation of numbers in a basis *base* and to calculate the conversion to and from integers, we introduce the following locale.

```
locale digits =
  fixes base :: nat
  assumes base_pos: "base > 0"
begin
```

Conversion from basis base to integers: *from_digits n d*

n:	*nat*	length of representation in basis base
d:	*nat ⇒ nat*	function of digits in basis base where *d i* is the i-th digit in basis base
output:	*nat*	natural number corresponding to $d(n-1) \ldots d(0)$ as integer

```
fun from_digits :: "nat ⇒ (nat ⇒ nat) ⇒ nat" where
  "from_digits 0 d = 0"
| "from_digits (Suc n) d = d 0 + base * from_digits n (d ∘ Suc)"
```

Alternative definition using sum:

```
lemma from_digits_altdef: "from_digits n d = (∑ i<n. d i * base ^ i)"
  ⟨proof⟩
```

Digit in basis base of some integer number: *digit x i*

x:	*nat*	integer
i:	*nat*	index
output:	*nat*	i-th digit of representation in basis base of x

```
fun digit :: "nat ⇒ nat ⇒ nat" where
  "digit x 0 = x mod base"
| "digit x (Suc i) = digit (x div base) i"
```

Alternative definition using divisor and modulo:

lemma *digit_altdef: "digit x i = (x div (base ^ i)) mod base"*
⟨*proof*⟩

Every digit must be smaller that the base.

lemma *digit_less_base: "digit x i < base"*
⟨*proof*⟩

A representation in basis `base` of length n must be less than `base`^{n}.

lemma *from_digits_less:*
assumes *"∀ i<n. d i < base"*
shows *"from_digits n d < base ^ n"*
⟨*proof*⟩

Lemmas for `mod` and `div` in number systems of basis `base`:

lemma *mod_base:* **assumes** *"⋀i. i<n ⟹ d i < base" "n>0"*
shows *"from_digits n d mod base = d 0 "*
⟨*proof*⟩

lemma *mod_base_i:*
assumes *"⋀i. i<n ⟹ d i < base" "n>0" "i<n"*
shows *"(∑ j=i..<n. d j * base ^ (j-i)) mod base = d i "*
⟨*proof*⟩

lemma *div_base_i:*
assumes *"⋀i. i<n ⟹ d i < base" "n>0" "i<n"*
shows *"from_digits n d div (base ^i) = (∑ j=i..<n. d j * base ^ (j-i))"*
⟨*proof*⟩

Conversions are inverse to each other.

lemma *digit_from_digits:*
assumes *"⋀j. j<n ⟹ d j < base" "n>0" "i<n"*
shows *"digit (from_digits n d) i = d i"*
⟨*proof*⟩

lemma *div_distrib:* **assumes** *"i<n"*
shows *"(a*base^n + b) div base^i mod base = b div base^i mod base"*
⟨*proof*⟩

lemma *from_digits_digit:*
assumes *"x < base ^ n"*
shows *"from_digits n (digit x) = x"*
⟨*proof*⟩

Stronger formulation of above lemma.

lemma *from_digits_digit':*
"from_digits n (digit x) = x mod (base ^ n)"
⟨*proof*⟩

```
end
end
theory Van_der_Waerden
  imports Main "HOL-Library.FuncSet" Digits
begin
```

2 Van der Waerden's Theorem

In combinatorics, Van der Waerden's Theorem is about arithmetic progressions of a certain length of the same colour in a colouring of an interval. In order to state the theorem and to prove it, we need to formally introduce arithmetic progressions. We will express k-colourings as functions mapping an integer interval to the set $\{0, \ldots, k-1\}$ of colours.

2.1 Arithmetic progressions

A sequence of integer numbers with the same step size is called an arithmetic progression. We say an m-fold arithmetic progression is an arithmetic progression with multiple step lengths.

Arithmetic progressions are defined in the following using the variables:

start:	`int`	starting value
step:	`nat`	positive integer for step length
i:	`nat`	*i*-th value in the arithmetic progression

```
definition arith_prog :: "int ⇒ nat ⇒ nat ⇒ int"
  where "arith_prog start step i = start + int (i * step)"
```

An m-fold arithmetic progression (which we will also call a multi-arithmetic progression) is defined in the following using the variables:

dims:	`nat`	number of dimensions/step directions of m-fold arithmetic progression
start:	`int`	starting value
steps:	`nat ⇒ nat`	function of steps, returns step in i-th dimension for $i \in [0..< dims]$
c:	`nat ⇒ nat`	function of coefficients, returns coefficient in i-th dimension for $i \in [0..< dims]$

```
definition multi_arith_prog ::
    "nat ⇒ int ⇒ (nat ⇒ nat) ⇒ (nat ⇒ nat) ⇒ int"
  where "multi_arith_prog dims start steps c =
          start + int (∑i<dims. c i * steps i)"
```

An m-fold arithmetic progression of dimension 1 is also an arithmetic progression and vice versa. This is shown in the following lemmas.

```
lemma multi_to_arith_prog:
```

```
  "multi_arith_prog 1 start steps c =
    arith_prog start (steps 0) (c 0)"
  ⟨proof⟩

lemma arith_prog_to_multi:
  "arith_prog start step c =
    multi_arith_prog 1 start (λ_. step) (λ_. c)"
  ⟨proof⟩
```

To show that an arithmetic progression is well-defined, we introduce the following predicate. It assures that `arith_prog start step ‘ [0..<l]` is contained in the integer interval $[a..b]$.

```
definition is_arith_prog_on ::
    "nat ⇒ int ⇒ nat ⇒ int ⇒ int ⇒ bool"
  where "is_arith_prog_on l start step a b ⟷
    (start ≥ a ∧ arith_prog start step (l-1) ≤ b)"
```

Furthermore, we have monotonicity for arithmetic progressions.

```
lemma arith_prog_mono:
  assumes "c ≤ c’"
  shows   "arith_prog start step c ≤ arith_prog start step c’"
  ⟨proof⟩
```

Now, we state the well-definedness of an arithmetic progression of length l in an integer interval $[a..b]$. Indeed, `is_arith_prog_on` guarantees that every element of `arith_prog start step` of length l lies in $[a..b]$.

```
lemma is_arith_prog_onD:
  assumes "is_arith_prog_on l start step a b"
  assumes "c ∈ {0..<l}"
  shows   "arith_prog start step c ∈ {a..b}"
⟨proof⟩
```

We also need a predicate for an m-fold arithmetic progression to be well-defined. It assures that `multi_arith_prog start step ‘ [0..<l]^m` is contained in $[a..b]$.

```
definition is_multi_arith_prog_on ::
    "nat ⇒ nat ⇒ int ⇒ (nat ⇒ nat) ⇒ int ⇒ int ⇒ bool"
  where "is_multi_arith_prog_on l m start steps a b ⟷
     (start ≥ a ∧ multi_arith_prog m start steps (λ_. l-1) ≤ b)"
```

Moreover, we have monotonicity for m-fold arithmetic progressions as well.

```
lemma multi_arith_prog_mono:
  assumes "⋀i. i < m ⟹ c i ≤ c’ i"
  shows   "multi_arith_prog m start steps c ≤
             multi_arith_prog m start steps c’"
  ⟨proof⟩
```

Finally, we get the well-definedness for m-fold arithmetic progressions of length l. Here, *is_multi_arith_prog_on* guarantees that every element of *multi_arith_prog start step* of length l lies in $[a..b]$.

```
lemma is_multi_arith_prog_onD:
  assumes "is_multi_arith_prog_on l m start steps a b"
  assumes "c ∈ {0..<m} → {0..<l}"
  shows   "multi_arith_prog m start steps c ∈ {a..b}"
⟨proof⟩
```

2.2 Van der Waerden's Theorem

The property for a number n to fulfill Van der Waerden's theorem is the following:
For a k-colouring col of $[a..b]$ there exist

- *start*: starting value of an arithmetic progression
- *step*: step length of an arithmetic progression
- j: colour

such that *arith_prog start step* is a valid arithmetic progression of length l lying in $[a..b]$ of the same colour j.

The following variables will be used:

k:	nat	number of colours in segment colouring on $[a..b]$
l:	nat	length of arithmetic progression
n:	nat	number fulfilling Van der Waerden's Theorem

```
definition vdw ::
    "nat ⇒ nat ⇒ nat ⇒ bool"
  where "vdw k l n ⟷
     (∀a b col. b + 1 ≥ a + int n ∧ col ∈ {a..b} → {..<k} ⟶
       (∃j start step. j < k ∧ step > 0 ∧
        is_arith_prog_on l start step a b ∧
        arith_prog start step ‘ {..<l} ⊆ col -‘ {j} ∩ {a..b}))"
```

To better work with the property of Van der Waerden's theorem, we introduce an elimination rule.

```
lemma vdwE:
  assumes "vdw k l n"
          "b + 1 ≥ a + int n"
          "col ∈ {a..b} → {..<k}"
  obtains j start step where
    "j < k" "step > 0"
    "is_arith_prog_on l start step a b"
    "arith_prog start step ‘ {..<l} ⊆ col -‘ {j} ∩ {a..b}"
  ⟨proof⟩
```

Van der Waerden's theorem implies that the number fulfilling it is positive. This is show in the following lemma.

```
lemma vdw_imp_pos:
  assumes "vdw k l n"
          "l > 0"
  shows "n > 0"
⟨proof⟩
```

Van der Waerden's Theorem is trivial for a non-existent colouring. It also makes no sense for arithmetic progressions of length 0.

```
lemma vdw_0_left [simp, intro]: "n>0 ⟹ vdw 0 l n"
  ⟨proof⟩
```

In the case of $k = 1$, Van der Waerden's Theorem holds. Then every number has the same colour, hence also the arithmetic progression. A possible choice for the number fulfilling Van der Waerden Theorem is l.

```
lemma vdw_1_left:
  assumes "l>0"
  shows "vdw 1 l l"
⟨proof⟩
```

In the case $l = 1$, Van der Waerden's Theorem holds. As the length of the arithmetic progression is 1, it consists of just one element. Thus every nonempty integer interval fulfills the Van der Waerden property. We can prove $N_{k,1}$ to be 1.

```
lemma vdw_1_right: "vdw k 1 1"
⟨proof⟩
```

In the case $l = 2$, Van der Waerden's Theorem holds as well. Here, any two distinct numbers form an arithmetic progression of length 2. Thus we only have to find two numbers with the same colour. Using the pigeonhole principle on $k + 1$ values, we can find two integers with the same colour.

```
lemma vdw_2_right: "vdw k 2 (k+1)"
⟨proof⟩
```

In order to prove Van der Waerden's Theorem, we first prove a slightly different lemma. The statement goes as follows:
For a k-colouring *col* on $[a..b]$ there exist

- *start*: starting value of an arithmetic progression
- *steps*: step length of an arithmetic progression

such that `f = multi_arith_prog m start step` is a valid m-fold arithmetic progression of length l lying in $[a..b]$ such that for every $s < m$ have: if $cj < l$ for all $j \leq s$ then $f(c_0, c_1, \ldots, c_{m-1})$ and $f(0, \ldots, 0, c_{s+1}, \ldots, c_{m-1})$ have the same colour.

The property of the lemma uses the following variables:

k:	`nat`	number of colours in segment colouring of $[a..b]$
m:	`nat`	dimension of m-fold arithmetic progression
l:	`nat`	$l+1$ is length of m-fold arithmetic progression
n:	`nat`	number fulfilling *vdw_lemma*

```
definition vdw_lemma :: "nat ⇒ nat ⇒ nat ⇒ nat ⇒ bool" where
  "vdw_lemma k m l n ⟷
     (∀a b col. b + 1 ≥ a + int n ∧ col ∈ {a..b} → {..<k} ⟶
       (∃start steps. (∀i<m. steps i > 0) ∧
        is_multi_arith_prog_on (l+1) m start steps a b ∧ (
           let f = multi_arith_prog m start steps
           in  (∀c ∈ {0..<m} → {0..l}. ∀s<m. (∀ j ≤ s. c j < l) ⟶
                  col (f c) = col (f (λi. if i ≤ s then 0 else c i))))))"
```

To better work with this property, we introduce an elimination rule for *vdw_lemma*.

```
lemma vdw_lemmaE:
  fixes a b :: int
  assumes "vdw_lemma k m l n"
    "b + 1 ≥ a + int n" "col ∈ {a..b} → {..<k}"
  obtains start steps where
    "⋀i. i < m ⟹ steps i > 0"
    "is_multi_arith_prog_on (l+1) m start steps a b"
    "let f = multi_arith_prog m start steps
     in  ∀c ∈ {0..<m} → {0..l}. ∀s<m. (∀ j ≤ s. c j < l) ⟶
            col (f c) = col (f (λi. if i ≤ s then 0 else c i))"
  ⟨proof⟩
```

To simplify the following proof, we show the following formula.

```
lemma sum_mod_poly:
  assumes "(k::nat)>0"
  shows "(k - 1) * (∑ n∈{..<q}. k^n) < k^q "
⟨proof⟩
```

The proof of Van der Waerden's Theorem now proceeds in three steps:

- Firstly, we show that the *vdw* property for all k proves the *vdw_lemma* for fixed l but arbitrary k and m. This is done by induction over m.
- Secondly, we show that *vdw_lemma* implies the induction step of *vdw* using the pigeonhole principle.
- Lastly, we combine the previous steps in an induction over l to show Van der Waerden's Theorem in the general setting.

Firstly, we need to show that *vdw* for arbitrary k implies *vdw_lemma* for fixed l. As mentioned earlier, we use induction over m.

```
lemma vdw_imp_vdw_lemma:
  fixes l
  assumes vdw_assms: "⋀k'. k'>0 ⟹ ∃n_k'. vdw k' l n_k'"
    and "l ≥ 2"
    and "m > 0"
    and "k > 0"
  shows   "∃N. vdw_lemma k m l N"
⟨proof⟩
```

Secondly, we show that *vdw_lemma* implies the induction step of Van der Waerden's Theorem using the pigeonhole principle.

```
lemma vdw_lemma_imp_vdw:
  assumes "vdw_lemma k k l N"
  shows   "vdw k (Suc l) N"
⟨proof⟩
```

Lastly, we assemble all lemmas to finally prove Van der Waerden's Theorem by induction on l. The cases $l = 1$ and the induction start $l = 2$ are treated separately and have been shown earlier.

```
theorem van_der_Waerden: assumes "l>0" "k>0" shows "∃n. vdw k l n"
⟨proof⟩

end
```

References

[1] R. G. Swan. Van der Waerden's theorem on arithmetic progressions. Technical report, Department of Mathematics, University of Chicago. Online at http://www.math.uchicago.edu/~swan/expo/vdW.pdf.