

# Duality of Linear Programming

René Thiemann

March 24, 2023

## Abstract

We formalize the weak and strong duality theorems of linear programming. For the strong duality theorem we provide three sufficient preconditions: both the primal problem and the dual problem are satisfiable, the primal problem is satisfiable and bounded, or the dual problem is satisfiable and bounded. The proofs are based on an existing formalization of Farkas' Lemma.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Minimum and Maximum of Potentially Infinite Sets</b>	<b>2</b>
<b>3</b>	<b>Weak and Strong Duality of Linear Programming</b>	<b>3</b>

## 1 Introduction

The proofs are taken from a textbook on linear programming [3]. There clearly is already an related AFP entry on linear programming [2] and we briefly explain the relationship between that entry and this one.

- The other AFP entry provides an algorithm for solving linear programs based on an existing simplex implementation. Since the simplex implementation is formulated only for rational numbers, several results are only available for rational numbers. Moreover, the simplex algorithm internally works on sets of inequalities that are represented by linear polynomials, and there are conversions between matrix-vector inequalities and linear polynomial inequalities. Finally, that AFP entry does not contain the strong duality theorem, which is the essential result in this AFP entry.
- This AFP entry has completely been formalized in the matrix-vector representation. It mainly consists of the strong duality theorems without any algorithms. The proof of these theorems are based on Farkas'

Lemma which is provided in [1] for arbitrary linearly ordered fields. Therefore, also the duality theorems are proven in that generality without the restriction to rational numbers.

## 2 Minimum and Maximum of Potentially Infinite Sets

```
theory Minimum-Maximum
  imports Main
begin
```

We define minima and maxima of sets. In contrast to the existing *Min* and *Max* operators, these operators are not restricted to finite sets

```
definition Maximum :: 'a :: linorder set  $\Rightarrow$  'a where
  Maximum S = (THE x. x  $\in$  S  $\wedge$  ( $\forall$  y  $\in$  S. y  $\leq$  x))
```

```
definition Minimum :: 'a :: linorder set  $\Rightarrow$  'a where
  Minimum S = (THE x. x  $\in$  S  $\wedge$  ( $\forall$  y  $\in$  S. x  $\leq$  y))
```

```
definition has-Maximum where has-Maximum S = ( $\exists$  x. x  $\in$  S  $\wedge$  ( $\forall$  y  $\in$  S. y  $\leq$  x))
```

```
definition has-Minimum where has-Minimum S = ( $\exists$  x. x  $\in$  S  $\wedge$  ( $\forall$  y  $\in$  S. x  $\leq$  y))
```

```
lemma eqMaximumI:
  assumes x  $\in$  S
  and  $\bigwedge$  y. y  $\in$  S  $\implies$  y  $\leq$  x
shows Maximum S = x
  unfolding Maximum-def
  by (standard, insert assms, auto, fastforce)
```

```
lemma eqMinimumI:
  assumes x  $\in$  S
  and  $\bigwedge$  y. y  $\in$  S  $\implies$  x  $\leq$  y
shows Minimum S = x
  unfolding Minimum-def
  by (standard, insert assms, auto, fastforce)
```

```
lemma has-MaximumD:
  assumes has-Maximum S
  shows Maximum S  $\in$  S
  x  $\in$  S  $\implies$  x  $\leq$  Maximum S
proof –
  from assms[unfolded has-Maximum-def]
  obtain m where *: m  $\in$  S  $\wedge$   $\forall$  y. y  $\in$  S  $\implies$  y  $\leq$  m by auto
  have id: Maximum S = m
  by (rule eqMaximumI, insert *, auto)
  from * id show Maximum S  $\in$  S x  $\in$  S  $\implies$  x  $\leq$  Maximum S by auto
qed
```

```

lemma has-MinimumD:
  assumes has-Minimum S
  shows Minimum S ∈ S
     $x \in S \implies \text{Minimum } S \leq x$ 
proof –
  from assms[unfolded has-Minimum-def]
  obtain m where  $*$ :  $m \in S \wedge y. y \in S \implies m \leq y$  by auto
  have id: Minimum S = m
    by (rule eqMinimumI, insert *, auto)
  from  $*$  id show Minimum S ∈ S  $x \in S \implies \text{Minimum } S \leq x$  by auto
qed

```

On non-empty finite sets, *Minimum* and *Min* coincide, and similarly *Maximum* and *Max*.

```

lemma Minimum-Min: assumes finite S S ≠ {}
  shows Minimum S = Min S
  by (rule eqMinimumI, insert assms, auto)

```

```

lemma Maximum-Max: assumes finite S S ≠ {}
  shows Maximum S = Max S
  by (rule eqMaximumI, insert assms, auto)

```

**end**

### 3 Weak and Strong Duality of Linear Programming

```

theory LP-Duality
  imports
    Linear-Inequalities.Farkas-Lemma
    Minimum-Maximum
begin

```

```

lemma weak-duality-theorem:
  fixes  $A :: 'a :: \text{linordered-comm-semiring-strict mat}$ 
  assumes  $A: A \in \text{carrier-mat } nr \ nc$ 
    and  $b: b \in \text{carrier-vec } nr$ 
    and  $c: c \in \text{carrier-vec } nc$ 
    and  $x: x \in \text{carrier-vec } nc$ 
    and  $Axb: A *_v x \leq b$ 
    and  $y0: y \geq 0_v \ nr$ 
    and  $yA: A^T *_v y = c$ 
  shows  $c \cdot x \leq b \cdot y$ 
proof –
  from  $y0$  have  $y: y \in \text{carrier-vec } nr$  unfolding less-eq-vec-def by auto
  have  $c \cdot x = (A^T *_v y) \cdot x$  unfolding  $yA$  by simp
  also have  $\dots = y \cdot (A *_v x)$  using  $x \ y \ A$  by (metis transpose-vec-mult-scalar)

```

**also have**  $\dots \leq y \cdot b$   
**unfolding** *scalar-prod-def* **using**  $A \ b \ Axb \ y0$   
**by** (*auto intro! sum-mono mult-left-mono simp less-eq-vec-def*)  
**also have**  $\dots = b \cdot y$  **using**  $y \ b$  **by** (*metis comm-scalar-prod*)  
**finally show** *?thesis* .  
**qed**

**corollary** *unbounded-primal-solutions:*

**fixes**  $A :: 'a :: \text{linordered-idom mat}$   
**assumes**  $A : A \in \text{carrier-mat nr nc}$   
**and**  $b : b \in \text{carrier-vec nr}$   
**and**  $c : c \in \text{carrier-vec nc}$   
**and** *unbounded*:  $\forall v. \exists x \in \text{carrier-vec nc}. A *_v x \leq b \wedge c \cdot x \geq v$   
**shows**  $\neg (\exists y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c)$

**proof**

**assume**  $(\exists y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c)$   
**then obtain**  $y$  **where**  $y : y \geq 0_v \text{ nr}$  **and**  $Ayc : A^T *_v y = c$   
**by** *auto*  
**from** *unbounded*[*rule-format, of*  $b \cdot y + 1$ ]  
**obtain**  $x$  **where**  $x : x \in \text{carrier-vec nc}$  **and**  $Axb : A *_v x \leq b$   
**and**  $le : b \cdot y + 1 \leq c \cdot x$  **by** *auto*  
**from** *weak-duality-theorem*[*OF*  $A \ b \ c \ x \ Axb \ y \ Ayc$ ]  
**have**  $c \cdot x \leq b \cdot y$  **by** *auto*  
**with**  $le$  **show** *False* **by** *auto*

**qed**

**corollary** *unbounded-dual-solutions:*

**fixes**  $A :: 'a :: \text{linordered-idom mat}$   
**assumes**  $A : A \in \text{carrier-mat nr nc}$   
**and**  $b : b \in \text{carrier-vec nr}$   
**and**  $c : c \in \text{carrier-vec nc}$   
**and** *unbounded*:  $\forall v. \exists y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c \wedge b \cdot y \leq v$   
**shows**  $\neg (\exists x \in \text{carrier-vec nc}. A *_v x \leq b)$

**proof**

**assume**  $\exists x \in \text{carrier-vec nc}. A *_v x \leq b$   
**then obtain**  $x$  **where**  $x : x \in \text{carrier-vec nc}$  **and**  $Axb : A *_v x \leq b$  **by** *auto*  
**from** *unbounded*[*rule-format, of*  $c \cdot x - 1$ ]  
**obtain**  $y$  **where**  $y : y \geq 0_v \text{ nr}$  **and**  $Ayc : A^T *_v y = c$  **and**  $le : b \cdot y \leq c \cdot x - 1$   
**by** *auto*  
**from** *weak-duality-theorem*[*OF*  $A \ b \ c \ x \ Axb \ y \ Ayc$ ]  
**have**  $c \cdot x \leq b \cdot y$  **by** *auto*  
**with**  $le$  **show** *False* **by** *auto*

**qed**

A version of the strong duality theorem which demands that both primal and dual problem are solvable. At this point we do not use min- or max-operations

**theorem** *strong-duality-theorem-both-sat:*

**fixes**  $A :: 'a :: \text{trivial-conjugatable-linordered-field mat}$

```

assumes  $A: A \in \text{carrier-mat } nr \ nc$ 
and  $b: b \in \text{carrier-vec } nr$ 
and  $c: c \in \text{carrier-vec } nc$ 
and  $\text{primal}: \exists x \in \text{carrier-vec } nc. A *_{\mathbb{V}} x \leq b$ 
and  $\text{dual}: \exists y. y \geq 0_{\mathbb{V}} \ nr \wedge A^T *_{\mathbb{V}} y = c$ 
shows  $\exists x \ y.$ 
   $x \in \text{carrier-vec } nc \wedge A *_{\mathbb{V}} x \leq b \wedge$ 
   $y \geq 0_{\mathbb{V}} \ nr \wedge A^T *_{\mathbb{V}} y = c \wedge$ 
   $c \cdot x = b \cdot y$ 
proof –
  define  $M\text{-up}$  where  $M\text{-up} = \text{four-block-mat } A \ (0_m \ nr \ nr) \ (\text{mat-of-row } (- \ c))$ 
   $(\text{mat-of-row } b)$ 
  define  $M\text{-low}$  where  $M\text{-low} = \text{four-block-mat } (0_m \ nc \ nc) \ (A^T) \ (0_m \ nc \ nc) \ (-$ 
   $(A^T))$ 
  define  $M\text{-last}$  where  $M\text{-last} = \text{append-cols } (0_m \ nr \ nc) \ (- \ 1_m \ nr \ :: \ 'a \ \text{mat})$ 
  define  $M$  where  $M = (M\text{-up} \ @_r \ M\text{-low}) \ @_r \ M\text{-last}$ 
  define  $bc$  where  $bc = ((b \ @_{\mathbb{V}} \ 0_{\mathbb{V}} \ 1) \ @_{\mathbb{V}} \ (c \ @_{\mathbb{V}} \ -c)) \ @_{\mathbb{V}} \ (0_{\mathbb{V}} \ nr)$ 

  let  $?nr = ((nr + 1) + (nc + nc)) + nr$ 
  let  $?nc = nc + nr$ 
  have  $M\text{-up}: M\text{-up} \in \text{carrier-mat } (nr + 1) \ ?nc$ 
  unfolding  $M\text{-up-def}$  using  $A \ b \ c$  by  $\text{auto}$ 
  have  $M\text{-low}: M\text{-low} \in \text{carrier-mat } (nc + nc) \ ?nc$ 
  unfolding  $M\text{-low-def}$  using  $A$  by  $\text{auto}$ 
  have  $M\text{-last}: M\text{-last} \in \text{carrier-mat } nr \ ?nc$ 
  unfolding  $M\text{-last-def}$  by  $\text{auto}$ 
  have  $M: M \in \text{carrier-mat } ?nr \ ?nc$ 
  using  $\text{carrier-append-rows}[OF \ \text{carrier-append-rows}[OF \ M\text{-up} \ M\text{-low}] \ M\text{-last}]$ 
  unfolding  $M\text{-def}$  by  $\text{auto}$ 
  have  $bc: bc \in \text{carrier-vec } ?nr$  unfolding  $bc\text{-def}$ 
  by  $(\text{intro} \ \text{append-carrier-vec}, \ \text{insert} \ b \ c, \ \text{auto})$ 
  have  $(\exists xy. xy \in \text{carrier-vec } ?nc \wedge M *_{\mathbb{V}} xy \leq bc)$ 
proof  $(\text{subst} \ \text{gram-schmidt.Farkas-Lemma}'[OF \ M \ bc], \ \text{intro} \ \text{allI} \ \text{impI}, \ \text{elim} \ \text{conjE})$ 
  fix  $ulv$ 
  assume  $ulv0: 0_{\mathbb{V}} \ ?nr \leq ulv$  and  $Mulv: M^T *_{\mathbb{V}} ulv = 0_{\mathbb{V}} \ ?nc$ 
  from  $ulv0[\text{unfolded less-eq-vec-def}]$ 
  have  $ulv: ulv \in \text{carrier-vec } ?nr$  by  $\text{auto}$ 
  define  $u1$  where  $u1 = \text{vec-first } ulv \ ((nr + 1) + (nc + nc))$ 
  define  $u2$  where  $u2 = \text{vec-first } u1 \ (nr + 1)$ 
  define  $u3$  where  $u3 = \text{vec-last } u1 \ (nc + nc)$ 
  define  $t$  where  $t = \text{vec-last } ulv \ nr$ 
  have  $ulvid: ulv = u1 \ @_{\mathbb{V}} \ t$  using  $ulv$ 
  unfolding  $u1\text{-def} \ t\text{-def}$  by  $\text{auto}$ 
  have  $t: t \in \text{carrier-vec } nr$  unfolding  $t\text{-def}$  by  $\text{auto}$ 
  have  $u1: u1 \in \text{carrier-vec } ((nr + 1) + (nc + nc))$ 
  unfolding  $u1\text{-def}$  by  $\text{auto}$ 
  have  $u1id: u1 = u2 \ @_{\mathbb{V}} \ u3$  using  $u1$ 
  unfolding  $u2\text{-def} \ u3\text{-def}$  by  $\text{auto}$ 
  have  $u2: u2 \in \text{carrier-vec } (nr + 1)$  unfolding  $u2\text{-def}$  by  $\text{auto}$ 

```

```

have u3: u3 ∈ carrier-vec (nc + nc) unfolding u3-def by auto
define v where v = vec-first u3 nc
define w where w = vec-last u3 nc
have u3id: u3 = v @v w using u3
  unfolding v-def w-def by auto
have v: v ∈ carrier-vec nc unfolding v-def by auto
have w: w ∈ carrier-vec nc unfolding w-def by auto

define u where u = vec-first u2 nr
define L where L = vec-last u2 1
have u2id: u2 = u @v L using u2
  unfolding u-def L-def by auto
have u: u ∈ carrier-vec nr unfolding u-def by auto
have L: L ∈ carrier-vec 1 unfolding L-def by auto
define vec1 where vec1 = AT *v u + mat-of-col (- c) *v L
have vec1: vec1 ∈ carrier-vec nc
  unfolding vec1-def mat-of-col-def using A u c L
  by (meson add-carrier-vec mat-of-row-carrier(1) mult-mat-vec-carrier trans-
pose-carrier-mat uminus-carrier-vec)
define vec2 where vec2 = A *v (v - w)
have vec2: vec2 ∈ carrier-vec nr
  unfolding vec2-def using A v w by auto
define vec3 where vec3 = mat-of-col b *v L
have vec3: vec3 ∈ carrier-vec nr
  using A b L unfolding mat-of-col-def vec3-def
  by (meson add-carrier-vec mat-of-row-carrier(1) mult-mat-vec-carrier trans-
pose-carrier-mat uminus-carrier-vec)
have Mt: MT = (M-upT @c M-lowT) @c M-lastT
  unfolding M-def append-cols-def by simp
have MT *v ulv = (M-upT @c M-lowT) *v u1 + M-lastT *v t
  unfolding Mt ulvid
  by (subst mat-mult-append-cols[OF carrier-append-cols - u1 t],
insert M-up M-low M-last, auto)
also have M-lastT = 0m nc nr @r - 1m nr unfolding M-last-def
  unfolding append-cols-def by (simp, subst transpose-uminus, auto)
also have ... *v t = 0v nc @v - t
  by (subst mat-mult-append[OF - - t], insert t, auto)
also have (M-upT @c M-lowT) *v u1 = (M-upT *v u2) + (M-lowT *v u3)
  unfolding u1id
  by (rule mat-mult-append-cols[OF - - u2 u3], insert M-up M-low, auto)
also have M-lowT = four-block-mat (0m nc nc) (0m nc nc) A (- A)
  unfolding M-low-def
  by (subst transpose-four-block-mat, insert A, auto)
also have ... *v u3 = (0m nc nc *v v + 0m nc nc *v w) @v (A *v v + - A
*v w) unfolding u3id
  by (subst four-block-mat-mult-vec[OF - - A - v w], insert A, auto)
also have 0m nc nc *v v + 0m nc nc *v w = 0v nc
  using v w by auto
also have A *v v + - A *v w = vec2 unfolding vec2-def using A v w

```

by (*metis* (*full-types*) *carrier-matD*(2) *carrier-vecD* *minus-add-uminus-vec*  
*mult-mat-vec-carrier* *mult-minus-distrib-mat-vec* *uminus-mult-mat-vec*)  
 also have  $M\text{-up}^T = \text{four-block-mat } A^T (\text{mat-of-col } (-\ c)) (0_m\ nr\ nr) (\text{mat-of-col } b)$   
 unfolding *M-up-def* *mat-of-col-def*  
 by (*subst* *transpose-four-block-mat*[*OF A*], *insert* *b c*, *auto*)  
 also have ...  $*_v\ u2 = \text{vec1 } @_v\ \text{vec3}$   
 unfolding *u2id* *vec1-def* *vec3-def*  
 by (*subst* *four-block-mat-mult-vec*[*OF - - - u L*], *insert* *A b c u*, *auto*)  
 also have ( $\text{vec1 } @_v\ \text{vec3}$ )  
 $+ (0_v\ nc\ @_v\ \text{vec2}) + (0_v\ nc\ @_v\ -\ t) =$   
 $(\text{vec1 } @_v\ (\text{vec3} + \text{vec2} - t))$   
 apply (*subst* *append-vec-add*[*of - nc - - nr*, *OF* *vec1 - vec3* *vec2*])  
 subgoal by *force*  
 apply (*subst* *append-vec-add*[*of - nc - - nr*])  
 subgoal using *vec1* by *auto*  
 subgoal by *auto*  
 subgoal using *vec2* *vec3* by *auto*  
 subgoal using *t* by *auto*  
 subgoal using *vec1* by *auto*  
 done  
 finally have  $\text{vec1 } @_v\ (\text{vec3} + \text{vec2} - t) = 0_v\ ?nc$   
 unfolding *Mulv* by *simp*  
 also have ...  $= 0_v\ nc\ @_v\ 0_v\ nr$  by *auto*  
 finally have  $\text{vec1} = 0_v\ nc \wedge \text{vec3} + \text{vec2} - t = 0_v\ nr$   
 by (*subst* (*asm*) *append-vec-eq*[*OF* *vec1*], *auto*)  
 hence *01*:  $\text{vec1} = 0_v\ nc$  and *02*:  $\text{vec3} + \text{vec2} - t = 0_v\ nr$  by *auto*  
 from *01* have  $\text{vec1} + \text{mat-of-col } c *_v\ L = \text{mat-of-col } c *_v\ L$   
 using *c L* *vec1* unfolding *mat-of-col-def* by *auto*  
 also have  $\text{vec1} + \text{mat-of-col } c *_v\ L = A^T *_v\ u$   
 unfolding *vec1-def*  
 using *A u c L* unfolding *mat-of-col-def* *mat-of-row-uminus* *transpose-uminus*  
 by (*subst* *uminus-mult-mat-vec*, *auto*)  
 finally have *As*:  $A^T *_v\ u = \text{mat-of-col } c *_v\ L$  .  
 from *02* have  $(\text{vec3} + \text{vec2} - t) + t = 0_v\ nr + t$   
 by *simp*  
 also have  $(\text{vec3} + \text{vec2} - t) + t = \text{vec2} + \text{vec3}$   
 using *vec3* *vec2* *t* by *auto*  
 finally have *t23*:  $t = \text{vec2} + \text{vec3}$  using *t* by *auto*  
 have *id0*:  $0_v\ ?nr = ((0_v\ nr\ @_v\ 0_v\ 1) @_v\ (0_v\ nc\ @_v\ 0_v\ nc)) @_v\ 0_v\ nr$   
 by *auto*  
 from *ulv0*[*unfolded* *id0* *ulvid* *u1id* *u2id* *u3id*]  
 have  $0_v\ nr \leq u \wedge 0_v\ 1 \leq L \wedge 0_v\ nc \leq v \wedge 0_v\ nc \leq w \wedge 0_v\ nr \leq t$   
 apply (*subst* (*asm*) *append-vec-le*[*of - (nr + 1) + (nc + nc)*])  
 subgoal by (*intro* *append-carrier-vec*, *auto*)  
 subgoal by (*intro* *append-carrier-vec* *u L v w*)  
 apply (*subst* (*asm*) *append-vec-le*[*of - (nr + 1)*])  
 subgoal by (*intro* *append-carrier-vec*, *auto*)  
 subgoal by (*intro* *append-carrier-vec* *u L v w*)

```

apply (subst (asm) append-vec-le[OF - u], force)
apply (subst (asm) append-vec-le[OF - v], force)
by auto
hence ineqs:  $0_v \text{ nr} \leq u \ 0_v \ 1 \leq L \ 0_v \ \text{nc} \leq v \ 0_v \ \text{nc} \leq w \ 0_v \ \text{nr} \leq t$ 
by auto
have  $ulv \cdot bc = u \cdot b + (v \cdot c + w \cdot (-c))$ 
unfolding ulvid u1id u2id u3id bc-def
apply (subst scalar-prod-append[OF - t])
apply (rule append-carrier-vec[OF append-carrier-vec[OF u L] append-carrier-vec[OF
v w]])
apply (rule append-carrier-vec[OF append-carrier-vec[OF b] append-carrier-vec];
use c in force)
apply force
apply (subst scalar-prod-append)
apply (rule append-carrier-vec[OF u L])
apply (rule append-carrier-vec[OF v w])
subgoal by (rule append-carrier-vec, insert b, auto)
subgoal by (rule append-carrier-vec, insert c, auto)
apply (subst scalar-prod-append[OF u L b], force)
apply (subst scalar-prod-append[OF v w c], use c in force)
apply (insert L t, auto)
done
also have  $v \cdot c + w \cdot (-c) = c \cdot v + (-c) \cdot w$ 
by (subst (1 2) comm-scalar-prod, insert w c v, auto)
also have  $\dots = c \cdot v - (c \cdot w)$  using c w by simp
also have  $\dots = c \cdot (v - w)$  using c v w
by (simp add: scalar-prod-minus-distrib)
finally have ulvbc:  $ulv \cdot bc = u \cdot b + c \cdot (v - w)$  .
define lam where lam = L $ 0
from ineqs(2) L have lam0: lam  $\geq 0$  unfolding less-eq-vec-def lam-def by
auto
have As:  $A^T \cdot_v u = lam \cdot_v c$  unfolding As using c L
unfolding lam-def mat-of-col-def
by (intro eq-vecI, auto simp: scalar-prod-def)
have vec3:  $vec3 = lam \cdot_v b$  unfolding vec3-def using b L
unfolding lam-def mat-of-col-def
by (intro eq-vecI, auto simp: scalar-prod-def)
note preconds = lam0 ineqs(1,3-)[unfolded t23[unfolded vec2-def vec3]] As
have  $0 \leq u \cdot b + c \cdot (v - w)$ 
proof (cases lam > 0)
case True
hence  $u \cdot b = inverse \ lam \cdot (lam \cdot (b \cdot u))$ 
using comm-scalar-prod[OF b u] by simp
also have  $\dots = inverse \ lam \cdot ((lam \cdot_v b) \cdot u)$ 
using b u by simp
also have  $\dots \geq inverse \ lam \cdot (-(A \cdot_v (v - w)) \cdot u)$ 
proof (intro mult-left-mono)
show  $0 \leq inverse \ lam$  using preconds by auto
show  $-(A \cdot_v (v - w)) \cdot u \leq (lam \cdot_v b) \cdot u$ 

```



```

    unfolding scalar-prod-def
    apply (rule sum-mono)
    subgoal for i
    using lesseq-vecD[OF - preconds(2), of nr i] lesseq-vecD[OF - preconds(5),
of nr i] u v w b A
      by (intro mult-right-mono, auto)
    done
  qed
  also have inverse lam *  $-(A *_v (v - w)) \cdot u =$ 
     $- (inverse lam * ((A *_v (v - w)) \cdot u))$ 
    by (subst scalar-prod-uminus-left, insert A u v w, auto)
  also have  $(A *_v (v - w)) \cdot u = (A^T *_v u) \cdot (v - w)$ 
    apply (subst transpose-vec-mult-scalar[OF A - u])
    subgoal using v w by force
    by (rule comm-scalar-prod[OF - u], insert A v w, auto)
  also have inverse lam * ... =  $c \cdot (v - w)$  unfolding preconds(6)
    using True
    by (subst scalar-prod-smult-left, insert c v w, auto)
  finally show ?thesis by simp
next
case False
with preconds have lam: lam = 0 by auto
from primal obtain x0 where x0:  $x0 \in carrier-vec\ nc$ 
  and Ax0b:  $A *_v x0 \leq b$  by auto
from dual obtain y0 where y00:  $y0 \geq 0_v\ nr$ 
  and Ay0c:  $A^T *_v y0 = c$  by auto
from y00 have y0:  $y0 \in carrier-vec\ nr$ 
  unfolding less-eq-vec-def by auto
have Au:  $A^T *_v u = 0_v\ nc$ 
  unfolding preconds lam using c by auto
have 0 =  $(A^T *_v u) \cdot x0$  unfolding Au using x0 by auto
also have ... =  $u \cdot (A *_v x0)$ 
  by (rule transpose-vec-mult-scalar[OF A x0 u])
also have ...  $\leq u \cdot b$ 
  unfolding scalar-prod-def
  apply (use A x0 b in simp)
  apply (intro sum-mono)
  subgoal for i
    using lesseq-vecD[OF - preconds(2), of nr i] lesseq-vecD[OF - Ax0b, of nr
i] u v w b A x0
      by (intro mult-left-mono, auto)
    done
  finally have ub:  $0 \leq u \cdot b$  .
  have  $c \cdot (v - w) = (A^T *_v y0) \cdot (v - w)$  unfolding Ay0c by simp
  also have ... =  $y0 \cdot (A *_v (v - w))$ 
    by (subst transpose-vec-mult-scalar[OF A - y0], insert v w, auto)
  also have ...  $\geq 0$ 
    unfolding scalar-prod-def
    apply (use A v w in simp)

```

```

    apply (intro sum-nonneg)
    subgoal for i
      using lesseq-vecD[OF - y00, of nr i] lesseq-vecD[OF - preconds(5)[unfolded
lam], of nr i] A y0 v w b
      by (intro mult-nonneg-nonneg, auto)
    done
    finally show ?thesis using ub by auto
  qed
  thus 0 ≤ ulv · bc unfolding ulvbc .
qed
then obtain xy where xy: xy ∈ carrier-vec ?nc and le: M *v xy ≤ bc by auto
define x where x = vec-first xy nc
define y where y = vec-last xy nr
have xyid: xy = x @v y using xy
  unfolding x-def y-def by auto
have x: x ∈ carrier-vec nc unfolding x-def by auto
have y: y ∈ carrier-vec nr unfolding y-def by auto
have At: AT ∈ carrier-mat nc nr using A by auto
have Ax1: A *v x @v vec 1 (λ-. b · y - c · x) ∈ carrier-vec (nr + 1)
  using A x by fastforce
have b0cc: (b @v 0v 1) @v c @v - c ∈ carrier-vec ((nr + 1) + (nc + nc))
  using b c
  by (intro append-carrier-vec, auto)
have M *v xy = (M-up *v xy @v M-low *v xy) @v (M-last *v xy)
  unfolding M-def
  unfolding mat-mult-append[OF carrier-append-rows[OF M-up M-low] M-last
xy]
  by (simp add: mat-mult-append[OF M-up M-low xy])
also have M-low *v xy = (0m nc nc *v x + AT *v y) @v (0m nc nc *v x + -
AT *v y)
  unfolding M-low-def xyid
  by (rule four-block-mat-mult-vec[OF - At - - x y], insert A, auto)
also have 0m nc nc *v x + AT *v y = AT *v y using A x y by auto
also have 0m nc nc *v x + - AT *v y = - AT *v y using A x y by auto
also have M-up *v xy = (A *v x + 0m nr nr *v y) @v
  (mat-of-row (- c) *v x + mat-of-row b *v y)
  unfolding M-up-def xyid
  by (rule four-block-mat-mult-vec[OF A - - - x y], insert b c, auto)
also have A *v x + 0m nr nr *v y = A *v x using A x y by auto
also have mat-of-row (- c) *v x + mat-of-row b *v y =
  vec 1 (λ -. b · y - c · x)
  unfolding mult-mat-vec-def using c x by (intro eq-vecI, auto)
also have M-last *v xy = - y
  unfolding M-last-def xyid using x y
  by (subst mat-mult-append-cols[OF - - x y], auto)
finally have ((A *v x @v vec 1 (λ-. b · y - c · x)) @v (AT *v y @v - AT *v
y)) @v -y
  = M *v xy ..
also have ... ≤ bc by fact

```

**also have**  $\dots = ((b @_v 0_v 1) @_v (c @_v -c)) @_v 0_v nr$  **unfolding** *bc-def* **by**  
*auto*  
**finally have** *ineqs*:  $A *_v x \leq b \wedge \text{vec } 1 (\lambda-. b \cdot y - c \cdot x) \leq 0_v 1$   
 $\wedge A^T *_v y \leq c \wedge -A^T *_v y \leq -c \wedge -y \leq 0_v nr$   
**apply** (*subst (asm) append-vec-le[OF - b0cc]*)  
**subgoal using** *A x y* **by** (*intro append-carrier-vec, auto*)  
**apply** (*subst (asm) append-vec-le[OF Ax1], use b in fastforce*)  
**apply** (*subst (asm) append-vec-le[OF - b], use A x in force*)  
**apply** (*subst (asm) append-vec-le[OF - c], use A y in force*)  
**by** *auto*  
**show** *?thesis*  
**proof** (*intro exI conjI*)  
**from** *ineqs* **show** *Axb*:  $A *_v x \leq b$  **by** *auto*  
**from** *ineqs* **have**  $-A^T *_v y \leq -c \wedge A^T *_v y \leq c$  **by** *auto*  
**hence**  $A^T *_v y \geq c \wedge A^T *_v y \leq c$  **unfolding** *less-eq-vec-def* **using** *A y* **by** *auto*  
**then show** *Aty*:  $A^T *_v y = c$  **by** *simp*  
**from** *ineqs* **have**  $-y \leq 0_v nr$  **by** *simp*  
**then show** *y0*:  $0_v nr \leq y$  **unfolding** *less-eq-vec-def* **by** *auto*  
**from** *ineqs* **have**  $b \cdot y \leq c \cdot x$  **unfolding** *less-eq-vec-def* **by** *auto*  
**with** *weak-duality-theorem[OF A b c x Axb y0 Aty]*  
**show**  $c \cdot x = b \cdot y$  **by** *auto*  
**qed** (*insert x*)  
**qed**

A version of the strong duality theorem which demands that the primal problem is solvable and the objective function is bounded.

**theorem** *strong-duality-theorem-primal-sat-bounded*:  
**fixes** *bound* :: '*a* :: *trivial-conjugatable-linordered-field*  
**assumes** *A*:  $A \in \text{carrier-mat } nr \ nc$   
**and** *b*:  $b \in \text{carrier-vec } nr$   
**and** *c*:  $c \in \text{carrier-vec } nc$   
**and** *sat*:  $\exists x \in \text{carrier-vec } nc. A *_v x \leq b$   
**and** *bounded*:  $\forall x \in \text{carrier-vec } nc. A *_v x \leq b \longrightarrow c \cdot x \leq \text{bound}$   
**shows**  $\exists x y.$   
 $x \in \text{carrier-vec } nc \wedge A *_v x \leq b \wedge$   
 $y \geq 0_v nr \wedge A^T *_v y = c \wedge$   
 $c \cdot x = b \cdot y$   
**proof** (*rule strong-duality-theorem-both-sat[OF A b c sat]*)  
**show**  $\exists y \geq 0_v nr. A^T *_v y = c$   
**proof** (*rule ccontr*)  
**assume**  $\neg ?thesis$   
**hence**  $\exists y. y \in \text{carrier-vec } nc \wedge 0_v nr \leq A *_v y \wedge 0 > y \cdot c$   
**by** (*subst (asm) gram-schmidt.Farkas-Lemma[OF - c], insert A, auto*)  
**then obtain** *y* **where**  $y: y \in \text{carrier-vec } nc$   
**and** *Ay0*:  $A *_v y \geq 0_v nr$  **and** *yc0*:  $y \cdot c < 0$  **by** *auto*  
**from** *sat* **obtain** *x* **where**  $x: x \in \text{carrier-vec } nc$   
**and** *Axb*:  $A *_v x \leq b$  **by** *auto*  
**define** *diff* **where**  $\text{diff} = \text{bound} + 1 - c \cdot x$   
**from** *x Axb bounded* **have**  $c \cdot x < \text{bound} + 1$  **by** *auto*

**hence**  $diff: diff > 0$  **unfolding**  $diff-def$  **by**  $auto$   
**from**  $yc0$  **have**  $inv: inverse (- (y \cdot c)) > 0$  **by**  $auto$   
**define**  $fact$  **where**  $fact = diff * (inverse (- (y \cdot c)))$   
**have**  $fact: fact > 0$  **unfolding**  $fact-def$  **using**  $diff inv$  **by**  $(metis mult-pos-pos)$   
**define**  $z$  **where**  $z = x - fact \cdot_v y$   
**have**  $A *_v z = A *_v x - A *_v (fact \cdot_v y)$   
**unfolding**  $z-def$  **using**  $A x y$  **by**  $(meson mult-minus-distrib-mat-vec smult-carrier-vec)$   
**also have**  $\dots = A *_v x - fact \cdot_v (A *_v y)$  **using**  $A y$  **by**  $auto$   
**also have**  $\dots \leq b$   
**proof**  $(intro lesseq-vecI[OF - b])$   
**show**  $A *_v x - fact \cdot_v (A *_v y) \in carrier-vec nr$  **using**  $A x y$  **by**  $auto$   
**fix**  $i$   
**assume**  $i: i < nr$   
**have**  $(A *_v x - fact \cdot_v (A *_v y)) \$ i$   
 $= (A *_v x) \$ i - fact * (A *_v y) \$ i$   
**using**  $i A x y$  **by**  $auto$   
**also have**  $\dots \leq b \$ i - fact * (A *_v y) \$ i$   
**using**  $lesseq-vecD[OF b Axb i]$  **by**  $auto$   
**also have**  $\dots \leq b \$ i - 0 * 0$  **using**  $lesseq-vecD[OF - Ay0 i]$   $fact A y i$   
**by**  $(intro diff-left-mono mult-monom, auto)$   
**finally show**  $(A *_v x - fact \cdot_v (A *_v y)) \$ i \leq b \$ i$  **by**  $simp$   
**qed**  
**finally have**  $Azb: A *_v z \leq b$  .  
**have**  $z: z \in carrier-vec nc$  **using**  $x y$  **unfolding**  $z-def$  **by**  $auto$   
**have**  $c \cdot z = c \cdot x - fact * (c \cdot y)$  **unfolding**  $z-def$   
**using**  $c x y$  **by**  $(simp add: scalar-prod-minus-distrib)$   
**also have**  $\dots = c \cdot x + diff$   
**unfolding**  $comm-scalar-prod[OF c y]$   $fact-def$  **using**  $yc0$  **by**  $simp$   
**also have**  $\dots = bound + 1$  **unfolding**  $diff-def$  **by**  $simp$   
**also have**  $\dots > c \cdot z$  **using**  $bounded Azb z$  **by**  $auto$   
**finally show**  $False$  **by**  $simp$   
**qed**  
**qed**

A version of the strong duality theorem which demands that the dual problem is solvable and the objective function is bounded.

**theorem** *strong-duality-theorem-dual-sat-bounded:*

**fixes**  $bound :: 'a :: trivial-conjugatable-linordered-field$

**assumes**  $A: A \in carrier-mat nr nc$

**and**  $b: b \in carrier-vec nr$

**and**  $c: c \in carrier-vec nc$

**and**  $sat: \exists y. y \geq 0_v nr \wedge A^T *_v y = c$

**and**  $bounded: \forall y. y \geq 0_v nr \wedge A^T *_v y = c \longrightarrow bound \leq b \cdot y$

**shows**  $\exists x y.$

$x \in carrier-vec nc \wedge A *_v x \leq b \wedge$

$y \geq 0_v nr \wedge A^T *_v y = c \wedge$

$c \cdot x = b \cdot y$

**proof**  $(rule strong-duality-theorem-both-sat[OF A b c - sat])$

**show**  $\exists x \in carrier-vec nc. A *_v x \leq b$

**proof** (*rule ccontr*)  
**assume**  $\neg ?thesis$   
**hence**  $\neg (\exists x. x \in \text{carrier-vec } nc \wedge A *_v x \leq b)$  **by** *auto*  
**then obtain**  $y$  **where**  $y0: y \geq 0_v \text{ nr}$  **and**  $Ay0: A^T *_v y = 0_v \text{ nc}$  **and**  $yb: y \cdot b < 0$   
**by** (*subst (asm) gram-schmidt.Farkas-Lemma'[OF A b], auto*)  
**from** *sat* **obtain**  $x$  **where**  $x0: x \geq 0_v \text{ nr}$  **and**  $Axc: A^T *_v x = c$  **by** *auto*  
**define** *diff* **where**  $\text{diff} = b \cdot x - (\text{bound} - 1)$   
**from**  $x0$   $Axc$  **bounded** **have**  $\text{bound} \leq b \cdot x$  **by** *auto*  
**hence** *diff*:  $\text{diff} > 0$  **unfolding** *diff-def* **by** *auto*  
**define** *fact* **where**  $\text{fact} = - \text{inverse } (y \cdot b) * \text{diff}$   
**have** *fact*:  $\text{fact} > 0$  **unfolding** *fact-def* **using** *diff yb* **by** (*auto intro: mult-neg-pos*)  
**define**  $z$  **where**  $z = x + \text{fact} \cdot_v y$   
**from**  $x0$  **have**  $x: x \in \text{carrier-vec } nr$   
**unfolding** *less-eq-vec-def* **by** *auto*  
**from**  $y0$  **have**  $y: y \in \text{carrier-vec } nr$   
**unfolding** *less-eq-vec-def* **by** *auto*  
**have**  $A^T *_v z = A^T *_v x + A^T *_v (\text{fact} \cdot_v y)$   
**unfolding** *z-def* **using**  $A \ x \ y$  **by** (*simp add: mult-add-distrib-mat-vec*)  
**also** **have**  $\dots = A^T *_v x + \text{fact} \cdot_v (A^T *_v y)$  **using**  $A \ y$  **by** *auto*  
**also** **have**  $\dots = c$  **unfolding**  $Ay0 \ Axc$  **using**  $c$  **by** *auto*  
**finally** **have**  $Azc: A^T *_v z = c$  .  
**have**  $z0: z \geq 0_v \text{ nr}$  **unfolding** *z-def*  
**by** (*intro lesseq-vecI[of - nr], insert x y lesseq-vecD[OF - x0, of nr] lesseq-vecD[OF - y0, of nr] fact,*  
*auto intro!: add-nonneg-nonneg*)  
**from** *bounded Azc z0* **have**  $bz: \text{bound} \leq b \cdot z$  **by** *auto*  
**also** **have**  $\dots = b \cdot x + \text{fact} * (b \cdot y)$  **unfolding** *z-def* **using**  $b \ x \ y$   
**by** (*simp add: scalar-prod-add-distrib*)  
**also** **have**  $\dots = \text{diff} + (\text{bound} - 1) + \text{fact} * (b \cdot y)$   
**unfolding** *diff-def* **by** *auto*  
**also** **have**  $\text{fact} * (b \cdot y) = - \text{diff}$  **using**  $yb$   
**unfolding** *fact-def comm-scalar-prod[OF y b]* **by** *auto*  
**finally** **show** *False* **by** *simp*  
**qed**  
**qed**

Now the previous three duality theorems are formulated via min/max.

**corollary** *strong-duality-theorem-min-max:*

**fixes**  $A :: 'a :: \text{trivial-conjugatable-linordered-field mat}$   
**assumes**  $A: A \in \text{carrier-mat } nr \ nc$   
**and**  $b: b \in \text{carrier-vec } nr$   
**and**  $c: c \in \text{carrier-vec } nc$   
**and** *primal*:  $\exists x \in \text{carrier-vec } nc. A *_v x \leq b$   
**and** *dual*:  $\exists y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c$   
**shows**  $\text{Maximum } \{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_v x \leq b\}$   
 $= \text{Minimum } \{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c\}$   
**and** *has-Maximum*  $\{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_v x \leq b\}$   
**and** *has-Minimum*  $\{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_v y = c\}$

**proof** –

```

let ?Prim = {c · x | x. x ∈ carrier-vec nc ∧ A *_v x ≤ b}
let ?Dual = {b · y | y. y ≥ 0_v nr ∧ A^T *_v y = c}
define Prim where Prim = ?Prim
define Dual where Dual = ?Dual
from strong-duality-theorem-both-sat[OF assms]
obtain x y where x: x ∈ carrier-vec nc and Axb: A *_v x ≤ b
  and y: y ≥ 0_v nr and Ayc: A^T *_v y = c
  and eq: c · x = b · y by auto
have cxP: c · x ∈ Prim unfolding Prim-def using x Axb by auto
have cxD: c · x ∈ Dual unfolding eq Dual-def using y Ayc by auto
{
  fix z
  assume z ∈ Prim
  from this[unfolded Prim-def] obtain x' where x': x' ∈ carrier-vec nc
    and Axb': A *_v x' ≤ b and z: z = c · x' by auto
  from weak-duality-theorem[OF A b c x' Axb' y Ayc, folded eq]
  have z ≤ c · x unfolding z .
} note cxMax = this
have max: Maximum Prim = c · x
  by (intro eqMaximumI cxP cxMax)
show has-Maximum ?Prim
  unfolding Prim-def[symmetric] has-Maximum-def using cxP cxMax by auto
{
  fix z
  assume z ∈ Dual
  from this[unfolded Dual-def] obtain y' where y': y' ≥ 0_v nr
    and Ayc': A^T *_v y' = c and z: z = b · y' by auto
  from weak-duality-theorem[OF A b c x Axb y' Ayc', folded z]
  have c · x ≤ z .
} note cxMin = this
show has-Minimum ?Dual
  unfolding Dual-def[symmetric] has-Minimum-def using cxD cxMin by auto
have min: Minimum Dual = c · x
  by (intro eqMinimumI cxD cxMin)
from min max show Maximum ?Prim = Minimum ?Dual
  unfolding Dual-def Prim-def by auto

```

**qed**

**corollary** strong-duality-theorem-primal-sat-bounded-min-max:

```

fixes bound :: 'a :: trivial-conjugatable-linordered-field
assumes A: A ∈ carrier-mat nr nc
  and b: b ∈ carrier-vec nr
  and c: c ∈ carrier-vec nc
  and sat: ∃ x ∈ carrier-vec nc. A *_v x ≤ b
  and bounded: ∀ x ∈ carrier-vec nc. A *_v x ≤ b ⟶ c · x ≤ bound
shows Maximum {c · x | x. x ∈ carrier-vec nc ∧ A *_v x ≤ b}
  = Minimum {b · y | y. y ≥ 0_v nr ∧ A^T *_v y = c}
  and has-Maximum {c · x | x. x ∈ carrier-vec nc ∧ A *_v x ≤ b}

```

**and** *has-Minimum*  $\{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c\}$   
**proof** –  
**let**  $?Prim = \{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_{\nu} x \leq b\}$   
**let**  $?Dual = \{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c\}$   
**from** *strong-duality-theorem-primal-sat-bounded*[*OF assms*]  
**have**  $\exists y \geq 0_v \text{ nr}. A^T *_{\nu} y = c$  **by** *blast*  
**from** *strong-duality-theorem-min-max*[*OF A b c sat this*]  
**show** *Maximum*  $?Prim = \text{Minimum } ?Dual$  *has-Maximum*  $?Prim$  *has-Minimum*  $?Dual$   
**by** *blast+*  
**qed**

**corollary** *strong-duality-theorem-dual-sat-bounded-min-max*:  
**fixes** *bound* :: 'a :: *trivial-conjugatable-linordered-field*  
**assumes** *A*:  $A \in \text{carrier-mat } nr \text{ } nc$   
**and** *b*:  $b \in \text{carrier-vec } nr$   
**and** *c*:  $c \in \text{carrier-vec } nc$   
**and** *sat*:  $\exists y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c$   
**and** *bounded*:  $\forall y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c \longrightarrow \text{bound} \leq b \cdot y$   
**shows** *Maximum*  $\{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_{\nu} x \leq b\}$   
 $= \text{Minimum } \{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c\}$   
**and** *has-Maximum*  $\{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_{\nu} x \leq b\}$   
**and** *has-Minimum*  $\{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c\}$   
**proof** –  
**let**  $?Prim = \{c \cdot x \mid x. x \in \text{carrier-vec } nc \wedge A *_{\nu} x \leq b\}$   
**let**  $?Dual = \{b \cdot y \mid y. y \geq 0_v \text{ nr} \wedge A^T *_{\nu} y = c\}$   
**from** *strong-duality-theorem-dual-sat-bounded*[*OF assms*]  
**have**  $\exists x \in \text{carrier-vec } nc. A *_{\nu} x \leq b$  **by** *blast*  
**from** *strong-duality-theorem-min-max*[*OF A b c this sat*]  
**show** *Maximum*  $?Prim = \text{Minimum } ?Dual$  *has-Maximum*  $?Prim$  *has-Minimum*  $?Dual$   
**by** *blast+*  
**qed**

**end**

## References

- [1] R. Bottesch, A. Reynaud, and R. Thiemann. Linear inequalities. *Archive of Formal Proofs*, June 2019. [https://isa-afp.org/entries/Linear\\_Inequalities.html](https://isa-afp.org/entries/Linear_Inequalities.html), Formal proof development.
- [2] J. Parsert and C. Kaliszyk. Linear programming. *Archive of Formal Proofs*, Aug. 2019. [https://isa-afp.org/entries/Linear\\_Programming.html](https://isa-afp.org/entries/Linear_Programming.html), Formal proof development.
- [3] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.