

Duality of Linear Programming

René Thiemann

February 6, 2026

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

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

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)

```

For natural numbers, having a maximum is the same as being bounded from above and non-empty, or being finite and non-empty.

```

lemma has-Maximum-nat-iff-bdd-above: has-Maximum (A :: nat set) ⟷ bdd-above
A ∧ A ≠ {}
  unfolding has-Maximum-def
  by (metis bdd-above.I bdd-above-nat emptyE finite-has-maximal nat-le-linear)

```

```

lemma has-Maximum-nat-iff-finite: has-Maximum (A :: nat set) ⟷ finite A ∧
A ≠ {}
  unfolding has-Maximum-nat-iff-bdd-above bdd-above-nat ..

```

```

lemma bdd-above-Maximum-nat:  $(x :: nat) \in A \implies \text{bdd-above } A \implies x \leq \text{Maximum } A$ 
  by (rule has-MaximumD, auto simp: has-Maximum-nat-iff-bdd-above)

```

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 $\neg (\exists y. y \geq 0_v nr \wedge A^T *_v y = c)$

proof

assume $(\exists y. y \geq 0_v nr \wedge A^T *_v y = c)$

then obtain y **where** $y: y \geq 0_v nr$ **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*: $\forall v. \exists y. y \geq 0_v nr \wedge A^T *_v y = c \wedge b \cdot y \leq v$

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  $\text{unbounded}[rule-format, \text{of } c \cdot x - 1]$ 
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  $\text{weak-duality-theorem}[OF A b c x Axb y Ayc]$ 
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 $A: A \in \text{carrier-mat } nr \text{ } nc$

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 $\exists x y.$

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

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

$c \cdot x = b \cdot y$

proof –

define $M\text{-up}$ **where** $M\text{-up} = \text{four-block-mat } A (0_m \text{ nr } nr) (\text{mat-of-row } (- c))$
 $(\text{mat-of-row } b)$

define $M\text{-low}$ **where** $M\text{-low} = \text{four-block-mat } (0_m \text{ nc } nc) (A^T) (0_m \text{ nc } nc) (-$
 $(A^T))$

define $M\text{-last}$ **where** $M\text{-last} = \text{append-cols } (0_m \text{ nr } nc) (- 1_m \text{ nr} :: 'a \text{ mat})$

define M **where** $M = (M\text{-up} \ @_r \ M\text{-low}) \ @_r \ M\text{-last}$

define bc **where** $bc = ((b \ @_v \ 0_v \ 1) \ @_v \ (c \ @_v \ -c)) \ @_v \ (0_v \ nr)$

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 auto**

have $M\text{-low}: M\text{-low} \in \text{carrier-mat } (nc + nc) \ ?nc$

unfolding $M\text{-low-def}$ **using** A **by auto**

have $M\text{-last}: M\text{-last} \in \text{carrier-mat } nr \ ?nc$

unfolding $M\text{-last-def}$ **by 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 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 $(\exists xy. xy \in \text{carrier-vec } ?nc \wedge M *_v xy \leq bc)$

proof $(\text{subst } \text{gram-schmidt.Farkas-Lemma}'[OF \ M \ bc], \text{intro } \text{allI } \text{impI}, \text{elim } \text{conjE})$

fix ulv

```

assume  $ulv0: 0_v \text{ ?nr} \leq ulv$  and  $Mulv: M^T *_v ulv = 0_v \text{ ?nc}$ 
from  $ulv0$  [unfolded less-eq-vec-def]
have  $ulv: ulv \in \text{carrier-vec ?nr}$  by 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  $ulvid: ulv = u1 @_v t$  using  $ulv$ 
  unfolding  $u1\text{-def } t\text{-def}$  by auto
have  $t: t \in \text{carrier-vec } nr$  unfolding  $t\text{-def}$  by auto
have  $u1: u1 \in \text{carrier-vec } ((nr + 1) + (nc + nc))$ 
  unfolding  $u1\text{-def}$  by auto
have  $u1id: u1 = u2 @_v u3$  using  $u1$ 
  unfolding  $u2\text{-def } u3\text{-def}$  by auto
have  $u2: u2 \in \text{carrier-vec } (nr + 1)$  unfolding  $u2\text{-def}$  by auto
have  $u3: u3 \in \text{carrier-vec } (nc + nc)$  unfolding  $u3\text{-def}$  by auto
define  $v$  where  $v = \text{vec-first } u3 nc$ 
define  $w$  where  $w = \text{vec-last } u3 nc$ 
have  $u3id: u3 = v @_v w$  using  $u3$ 
  unfolding  $v\text{-def } w\text{-def}$  by auto
have  $v: v \in \text{carrier-vec } nc$  unfolding  $v\text{-def}$  by auto
have  $w: w \in \text{carrier-vec } nc$  unfolding  $w\text{-def}$  by auto

define  $u$  where  $u = \text{vec-first } u2 nr$ 
define  $L$  where  $L = \text{vec-last } u2 1$ 
have  $u2id: u2 = u @_v L$  using  $u2$ 
  unfolding  $u\text{-def } L\text{-def}$  by auto
have  $u: u \in \text{carrier-vec } nr$  unfolding  $u\text{-def}$  by auto
have  $L: L \in \text{carrier-vec } 1$  unfolding  $L\text{-def}$  by auto
define  $vec1$  where  $vec1 = A^T *_v u + \text{mat-of-col } (-c) *_v L$ 
have  $vec1: vec1 \in \text{carrier-vec } nc$ 
  unfolding  $vec1\text{-def } \text{mat-of-col-def}$  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  $vec2: vec2 \in \text{carrier-vec } nr$ 
  unfolding  $vec2\text{-def}$  using  $A v w$  by auto
define  $vec3$  where  $vec3 = \text{mat-of-col } b *_v L$ 
have  $vec3: vec3 \in \text{carrier-vec } nr$ 
  using  $A b L$  unfolding  $\text{mat-of-col-def } vec3\text{-def}$ 
  by (meson add-carrier-vec mat-of-row-carrier(1) mult-mat-vec-carrier trans-
pose-carrier-mat uminus-carrier-vec)
have  $Mt: M^T = (M\text{-up}^T @_c M\text{-low}^T) @_c M\text{-last}^T$ 
  unfolding  $M\text{-def } \text{append-cols-def}$  by simp
have  $M^T *_v ulv = (M\text{-up}^T @_c M\text{-low}^T) *_v u1 + M\text{-last}^T *_v t$ 
  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  $M\text{-last}^T = 0_m nc nr @_r - 1_m nr$  unfolding  $M\text{-last-def}$ 

```

unfolding *append-cols-def* **by** (*simp*, *subst transpose-uminus*, *auto*)
also have ... $*_v t = 0_v \text{ nc } @_v - t$
by (*subst mat-mult-append*[*OF - - t*], *insert t*, *auto*)
also have $(M\text{-up}^T @_c M\text{-low}^T) *_v u1 = (M\text{-up}^T *_v u2) + (M\text{-low}^T *_v u3)$
unfolding *u1id*
by (*rule mat-mult-append-cols*[*OF - - u2 u3*], *insert M-up M-low*, *auto*)
also have $M\text{-low}^T = \text{four-block-mat } (0_m \text{ nc } \text{ nc}) (0_m \text{ nc } \text{ nc}) A (- A)$
unfolding *M-low-def*
by (*subst transpose-four-block-mat*, *insert A*, *auto*)
also have ... $*_v u3 = (0_m \text{ nc } \text{ nc } *_v v + 0_m \text{ nc } \text{ nc } *_v w) @_v (A *_v v + - A *_v w)$ **unfolding** *u3id*
by (*subst four-block-mat-mult-vec*[*OF - - A - v w*], *insert A*, *auto*)
also have $0_m \text{ nc } \text{ nc } *_v v + 0_m \text{ nc } \text{ nc } *_v w = 0_v \text{ nc}$
using *v w by auto*
also have $A *_v v + - A *_v w = \text{vec2}$ **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 $M\text{-up}^T = \text{four-block-mat } A^T (\text{mat-of-col } (- c)) (0_m \text{ nr } \text{ nr}) (\text{mat-of-col } b)$
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 $(\text{vec1 } @_v \text{ vec3}) + (0_v \text{ nc } @_v \text{ vec2}) + (0_v \text{ nc } @_v - t) = (\text{vec1 } @_v (\text{vec3} + \text{vec2} - t))$
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 $\text{vec1 } @_v (\text{vec3} + \text{vec2} - t) = 0_v \text{ ?nc}$
unfolding *Mulv by simp*
also have ... $= 0_v \text{ nc } @_v 0_v \text{ nr}$ **by auto**
finally have $\text{vec1} = 0_v \text{ nc } \wedge \text{vec3} + \text{vec2} - t = 0_v \text{ nr}$
by (*subst (asm) append-vec-eq*[*OF vec1*], *auto*)
hence *01*: $\text{vec1} = 0_v \text{ nc}$ **and** *02*: $\text{vec3} + \text{vec2} - t = 0_v \text{ nr}$ **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  $(vec3 + vec2 - t) + t = 0_v nr + t$ 
  by simp
also have  $(vec3 + vec2 - t) + t = vec2 + vec3$ 
  using  $vec3\ vec2\ t$  by auto
finally have  $t23: t = vec2 + vec3$  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  $ineqs: 0_v nr \leq u\ 0_v 1 \leq L\ 0_v nc \leq v\ 0_v nc \leq w\ 0_v nr \leq t$ 
  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  $As: A^T *_v u = lam \cdot_v c$  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 ·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 ≤ u · b + c · (v - w)
proof (cases lam > 0)
  case True
  hence u · b = inverse lam * (lam * (b · u))
    using comm-scalar-prod[OF b u] by simp
  also have ... = inverse lam * ((lam ·v b) · u)
    using b u by simp
  also have ... ≥ inverse lam * (-(A *v (v - w)) · u)
proof (intro mult-left-mono)
  show 0 ≤ inverse lam using preconds by auto
  show -(A *v (v - w)) · u ≤ (lam ·v b) · 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)) · u) =
  - (inverse lam * ((A *v (v - w)) · u))
  by (subst scalar-prod-uminus-left, insert A u v w, auto)
also have (A *v (v - w)) · u = (AT *v u) · (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 · (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 ∈ carrier-vec nc
    and Ax0b: A *v x0 ≤ b by auto
  from dual obtain y0 where y00: y0 ≥ 0v nr
    and Ay0c: AT *v y0 = c by auto
  from y00 have y0: y0 ∈ carrier-vec nr
    unfolding less-eq-vec-def by auto
  have Au: AT *v u = 0v nc
    unfolding preconds lam using c by auto
  have 0 = (AT *v u) · x0 unfolding Au using x0 by auto
  also have ... = u · (A *v x0)
    by (rule transpose-vec-mult-scalar[OF A x0 u])
  also have ... ≤ u · 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  $\dots = y0 \cdot (A *_v (v - w))$ 
  by (subst transpose-vec-mult-scalar[OF A - y0], insert v w, auto)
also have  $\dots \geq 0$ 
  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)[unfolding
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  $0 \leq ulv \cdot bc$  unfolding ulvbc .
qed
then obtain xy where xy:  $xy \in \text{carrier-vec } ?nc$  and le:  $M *_v xy \leq bc$  by auto
define x where  $x = \text{vec-first } xy \ nc$ 
define y where  $y = \text{vec-last } xy \ nr$ 
have xyid:  $xy = x @_v y$  using xy
  unfolding x-def y-def by auto
have x:  $x \in \text{carrier-vec } nc$  unfolding x-def by auto
have y:  $y \in \text{carrier-vec } nr$  unfolding y-def by auto
have At:  $A^T \in \text{carrier-mat } nc \ nr$  using A by auto
have Ax1:  $A *_v x @_v \text{vec } 1 \ (\lambda-. b \cdot y - c \cdot x) \in \text{carrier-vec } (nr + 1)$ 
  using A x by fastforce
have b0cc:  $(b @_v 0_v \ 1) @_v c @_v - c \in \text{carrier-vec } ((nr + 1) + (nc + nc))$ 
  using b c
  by (intro append-carrier-vec, auto)
have  $M *_v xy = (M\text{-up } *_v xy @_v M\text{-low } *_v xy) @_v (M\text{-last } *_v xy)$ 
  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  $M\text{-low } *_v xy = (0_m \ nc \ nc *_v x + A^T *_v y) @_v (0_m \ nc \ nc *_v x + -$ 
 $A^T *_v y)$ 
  unfolding M-low-def xyid
  by (rule four-block-mat-mult-vec[OF - At - - x y], insert A, auto)
also have  $0_m \ nc \ nc *_v x + A^T *_v y = A^T *_v y$  using A x y by auto
also have  $0_m \ nc \ nc *_v x + - A^T *_v y = - A^T *_v y$  using A x y by auto

```

also have $M\text{-up } *_v xy = (A *_v x + 0_m nr nr *_v y) @_v$
 $(mat\text{-of-row } (- c) *_v x + mat\text{-of-row } b *_v y)$
unfolding $M\text{-up-def } xyid$
by (*rule four-block-mat-mult-vec*[$OF A - - - x y$], *insert b c*, *auto*)
also have $A *_v x + 0_m nr nr *_v y = A *_v x$ **using** $A x y$ **by** *auto*
also have $mat\text{-of-row } (- c) *_v x + mat\text{-of-row } b *_v y =$
 $vec\ 1 (\lambda -. b \cdot y - c \cdot x)$
unfolding $mult\text{-mat-vec-def}$ **using** $c x$ **by** (*intro eq-vecI*, *auto*)
also have $M\text{-last } *_v xy = - y$
unfolding $M\text{-last-def } xyid$ **using** $x y$
by (*subst mat-mult-append-cols*[$OF - - x y$], *auto*)
finally have $((A *_v x @_v vec\ 1 (\lambda -. b \cdot y - c \cdot x)) @_v (A^T *_v y @_v - A^T *_v$
 $y)) @_v -y$
 $= M *_v xy ..$
also have $\dots \leq bc$ **by** *fact*
also have $\dots = ((b @_v 0_v 1) @_v (c @_v -c)) @_v 0_v nr$ **unfolding** $bc\text{-def}$ **by**
auto
finally have $ineqs: A *_v x \leq b \wedge vec\ 1 (\lambda -. b \cdot y - c \cdot x) \leq 0_v 1$
 $\wedge A^T *_v y \leq c \wedge - A^T *_v y \leq -c \wedge -y \leq 0_v nr$
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*[$OF AxI$], *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** $less\text{-eq-vec-def}$ **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** $less\text{-eq-vec-def}$ **by** *auto*
from $ineqs$ **have** $b \cdot y \leq c \cdot x$ **unfolding** $less\text{-eq-vec-def}$ **by** *auto*
with $weak\text{-duality-theorem}$ [$OF A b c x Axb y0 Aty$]
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 $bound :: 'a :: trivial\text{-conjugatable-linordered-field}$

assumes $A: A \in carrier\text{-mat } nr\ nc$

and $b: b \in carrier\text{-vec } nr$

and $c: c \in carrier\text{-vec } nc$

and $sat: \exists x \in carrier\text{-vec } nc. A *_v x \leq b$

and $bounded: \forall x \in carrier\text{-vec } nc. A *_v x \leq b \longrightarrow c \cdot x \leq bound$

shows $\exists x y.$

$x \in \text{carrier-vec } nc \wedge A *_{\nu} x \leq b \wedge$
 $y \geq 0_{\nu} nr \wedge A^T *_{\nu} y = c \wedge$
 $c \cdot x = b \cdot y$

proof (*rule strong-duality-theorem-both-sat*[*OF A b c sat*])
show $\exists y \geq 0_{\nu} nr. A^T *_{\nu} y = c$
proof (*rule ccontr*)
assume $\neg ?thesis$
hence $\exists y. y \in \text{carrier-vec } nc \wedge 0_{\nu} nr \leq A *_{\nu} y \wedge 0 > y \cdot c$
by (*subst (asm) gram-schmidt.Farkas-Lemma*[*OF - c*], *insert A, auto*)
then obtain y **where** $y: y \in \text{carrier-vec } nc$
and $Ay0: A *_{\nu} y \geq 0_{\nu} nr$ **and** $yc0: y \cdot c < 0$ **by** *auto*
from sat obtain x **where** $x: x \in \text{carrier-vec } nc$
and $Axb: A *_{\nu} x \leq b$ **by** *auto*
define $diff$ **where** $diff = bound + 1 - c \cdot x$
from $x Axb$ **bounded** **have** $c \cdot x < bound + 1$ **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** $z = x - fact \cdot_{\nu} y$
have $A *_{\nu} z = A *_{\nu} x - A *_{\nu} (fact \cdot_{\nu} y)$
unfolding $z-def$ **using** $A x y$ **by** (*meson mult-minus-distrib-mat-vec smult-carrier-vec*)
also have $\dots = A *_{\nu} x - fact \cdot_{\nu} (A *_{\nu} y)$ **using** $A y$ **by** *auto*
also have $\dots \leq b$
proof (*intro lesseq-vecI*[*OF - b*])
show $A *_{\nu} x - fact \cdot_{\nu} (A *_{\nu} y) \in \text{carrier-vec } nr$ **using** $A x y$ **by** *auto*
fix i
assume $i: i < nr$
have $(A *_{\nu} x - fact \cdot_{\nu} (A *_{\nu} y)) \$ i$
 $= (A *_{\nu} x) \$ i - fact * (A *_{\nu} y) \$ i$
using $i A x y$ **by** *auto*
also have $\dots \leq b \$ i - fact * (A *_{\nu} y) \$ i$
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 $(A *_{\nu} x - fact \cdot_{\nu} (A *_{\nu} y)) \$ i \leq b \$ i$ **by** *simp*

qed
finally have $Azb: A *_{\nu} z \leq b$.
have $z: z \in \text{carrier-vec } nc$ **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 prob-

lem is solvable and the objective function is bounded.

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

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

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

and *b*: $b \in \text{carrier-vec } nr$

and *c*: $c \in \text{carrier-vec } nc$

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

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

shows $\exists x y.$

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

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

$c \cdot x = b \cdot y$

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

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

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*: $\exists y. y \geq 0_v nr \wedge A^T *_v y = c$
shows $\text{Maximum } \{c \cdot x \mid x. x \in \text{carrier-vec nc} \wedge A *_v x \leq b\}$
 $= \text{Minimum } \{b \cdot y \mid y. y \geq 0_v nr \wedge A^T *_v y = c\}$
and *has-Maximum* $\{c \cdot x \mid x. x \in \text{carrier-vec nc} \wedge A *_v x \leq b\}$
and *has-Minimum* $\{b \cdot y \mid y. y \geq 0_v nr \wedge A^T *_v y = c\}$

proof –

let $?Prim = \{c \cdot x \mid x. x \in \text{carrier-vec nc} \wedge A *_v x \leq b\}$
let $?Dual = \{b \cdot y \mid y. y \geq 0_v nr \wedge A^T *_v y = c\}$
define $Prim$ **where** $Prim = ?Prim$
define $Dual$ **where** $Dual = ?Dual$
from *strong-duality-theorem-both-sat*[*OF assms*]
obtain $x y$ **where** $x: x \in \text{carrier-vec nc}$ **and** $Axb: A *_v x \leq b$
and $y: y \geq 0_v nr$ **and** $Ayc: A^T *_v y = c$
and $eq: c \cdot x = b \cdot y$ **by** *auto*
have $cxP: c \cdot x \in Prim$ **unfolding** *Prim-def* **using** $x Axb$ **by** *auto*
have $cxD: c \cdot x \in Dual$ **unfolding** $eq Dual-def$ **using** $y Ayc$ **by** *auto*
{
fix z
assume $z \in Prim$
from $this[unfolding Prim-def]$ **obtain** x' **where** $x': x' \in \text{carrier-vec nc}$
and $Axb': A *_v x' \leq b$ **and** $z: z = c \cdot x'$ **by** *auto*
from *weak-duality-theorem*[*OF A b c x' Axb' y Ayc, folded eq*]
have $z \leq c \cdot x$ **unfolding** z .
} note $cxMax = this$
have $max: \text{Maximum } Prim = c \cdot x$
by (*intro eqMaximumI cxP cxMax*)
show *has-Maximum* $?Prim$
unfolding *Prim-def*[*symmetric*] *has-Maximum-def* **using** $cxP cxMax$ **by** *auto*
{
fix z
assume $z \in Dual$
from $this[unfolding Dual-def]$ **obtain** y' **where** $y': y' \geq 0_v nr$
and $Ayc': A^T *_v y' = c$ **and** $z: z = b \cdot y'$ **by** *auto*
from *weak-duality-theorem*[*OF A b c x Axb y' Ayc', folded z*]
have $c \cdot x \leq z$.
} note $cxMin = this$
show *has-Minimum* $?Dual$
unfolding *Dual-def*[*symmetric*] *has-Minimum-def* **using** $cxD cxMin$ **by** *auto*
have $min: \text{Minimum } Dual = c \cdot x$
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 · y | y. y ≥ 0_v nr ∧ 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}*
from *strong-duality-theorem-primal-sat-bounded[OF assms]*
have *∃ y ≥ 0_v nr. A^T *_v y = c* **by** *blast*
from *strong-duality-theorem-min-max[OF A b c sat this]*
show *Maximum ?Prim = 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 ∈ carrier-mat nr nc*
and *b: b ∈ carrier-vec nr*
and *c: c ∈ carrier-vec nc*
and *sat: ∃ y. y ≥ 0_v nr ∧ A^T *_v y = c*
and *bounded: ∀ y. y ≥ 0_v nr ∧ A^T *_v y = c ⟶ bound ≤ b · y*
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 · y | y. y ≥ 0_v nr ∧ 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}*
from *strong-duality-theorem-dual-sat-bounded[OF assms]*
have *∃ x ∈ carrier-vec nc. A *_v x ≤ b* **by** *blast*
from *strong-duality-theorem-min-max[OF A b c this sat]*
show *Maximum ?Prim = 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, 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, Formal proof development.
- [3] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.