Duality theory in linear optimization and its extensions — formally verified

Abstract: Farkas established that a system of linear inequalities has a solution if and only if we cannot obtain a contradiction by taking a linear combination of the inequalities. We state and formally prove several Farkas-like theorems in Lean 4. Furthermore, we consider a linearly ordered field extended with two special elements denoted by $\bot$ and $\top$ where $\bot$ is below every element and $\top$ is above every element. We define $\bot + a = \bot = a + \bot$ of all a and we define $\top + b = \top = b + \top$ for all $b \neq \bot$. For multiplication, we define $\bot \cdot c = \bot = c \cdot \bot$ for every $c \geq 0$ but $\top \cdot d = \top = d \cdot \top$ only for $d > 0$ because $\top \cdot 0 = 0 = 0 \cdot \top$. We extend certain Farkas-like theorems to a setting where coefficients are from an extended linearly ordered field.

1 Introduction

A basic knowledge from linear algebra is that a system of linear equalities has a solution if and only if we cannot obtain a contradiction by taking a linear combination of the equalities. We state this theorem as follows.

Theorem (equalityFredholm): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F$. Let b be a vector of type $I \to F$. Exactly one of the following exists:

- vector $x : J \to F$ such that $A \cdot x = b$
- vector $y : I \to F$ such that $A^T \cdot y = 0$ and $b \cdot y \neq 0$

Geometric interpretation of equalityFredholm is straightforward. The column vectors of A generate a hyperplane in the $|I|$-dimensional Euclidean space that contains the origin. The point b either lies in this hyperplane (in this case, the entries of x give coefficients which, when applied to the column vectors of A, give a vector from the origin to the point b), or there exists a line through the origin that is orthogonal to all the column vectors of A (i.e., orthogonal to the entire hyperplane) such that b projected onto this line falls outside of the origin (in this case, y gives a direction of this line), i.e., to a different point from where all column vectors of A get projected.

This theorem can be given in much more general settings. In our paper, however, this is the only version we provide. This staple of linear algebra is not our main focus but a byproduct of the other theorems we prove. In particular, we obtain equalityFredholm as an immediate corollary of the following theorem.

Theorem (equalityFredholm_lt): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F$. Let b be a vector of type $I \to F$. Exactly one of the following exists:

- vector $x : J \to F$ such that $A \cdot x = b$
- vector $y : I \to F$ such that $A^T \cdot y = 0$ and $b \cdot y < 0$

The way we state the theorem exemplifies certain patterns that permeate through our work. Our results are phrased as "there are two systems of (in)equalities; exactly one of them has a solution". This goes hand-in-hand with our decision to focus mostly on "symmetric" Farkas-like theorems. Note that it must be impossible to satisfy the second system by $y = 0$. Had it been allowed, we would have said nothing about the first system as it would have to lead to a contradiction every time. The constraint $b \cdot y < 0$ disqualifies the zero solution in most of our theorems. Intuitively, it should make sense that one of the systems is always "strict" (to easily see why, consider I and J singletons), which also means that our two systems will never be "fully symmetric".

Farkas (TODO citation) gave a similar characterization for systems of linear equalities with nonnegative variables. We state his theorem as follows.

Theorem (equalityFarkas): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F$. Let b be a vector of type $I \to F$. Exactly one of the following exists:

- nonnegative vector $x : J \to F$ such that $A \cdot x = b$
- vector $y : I \to F$ such that $A^T \cdot y \geq 0$ and $b \cdot y < 0$

Geometric interpretation of equalityFarkas is easy. The column vectors of A generate a cone in the $|I|$-dimensional Euclidean space from the origin towards some infinity. The point b either lies inside this cone (in this case, the entries of x give nonnegative coefficients which, when applied to the column vectors of A, give a vector from the origin to the point b), or there exists a hyperplane that contains the origin and that strictly separates b from given cone (in this case, y gives a normal vector of this hyperplane).

We prove equalityFredholm_lt by applying equalityFarkas to the matrix $(A \mid -A)$. However, equalityFarkas will be proved later, from a more general theorem.

Minkowski (TODO citation) similarly established that a system of linear inequalities has a nonnegative solution if and only if we cannot obtain a contradiction by taking a nonnegative linear combination of the inequalities. For reasons that will be apparent later, we give two versions of this theorem.

Theorem (inequalityFarkas): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F$. Let b be a vector of type $I \to F$. Exactly one of the following exists:

- nonnegative vector $x : J \to F$ such that $A \cdot x \le b$
- nonnegative vector $y : I \to F$ such that $A^T \cdot y \ge 0$ and $b \cdot y < 0$

Geometric interpretation of inequalityFarkas is a bit harder. The column vectors of A generate a cone in the $|I|$-dimensional Euclidean space from the origin to some infinity. The point b determines an orthogonal cone that starts in b and goes to negative infinity in the direction of all coordinate axes. Either these two cones intersect (in this case, the entries of x give nonnegative coefficients which, when applied to the column vectors of A, give a vector from the origin to a point in the intersection), or there exists a hyperplane that contains the origin and that strictly separates b from the cone generated by A but does not cut through the positive orthant, i.e., the origin is the only nonnegative point contained in the hyperplane (in this case, y gives a normal vector of this hyperplane).

Theorem (inequalityFarkas_neg): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F$. Let b be a vector of type $I \to F$. Exactly one of the following exists:

- nonnegative vector $x : J \to F$ such that $A \cdot x \le b$
- nonnegative vector $y : I \to F$ such that $(-A^T) \cdot y \le 0$ and $b \cdot y < 0$

Obviously, inequalityFarkas_neg is an immediate corollary of inequalityFarkas. We prove inequalityFarkas by applying equalityFarkas to the matrix $(1 \mid A)$ where 1 is the identity matrix of type $(I \times I) \to F$.

The next theorem generalizes equalityFarkas to structures where multiplication does not have to be commutative. Furthermore, it supports infinitely many equations.

Theorem (coordinateFarkas): Let I be any type. Let J be a finite type. Let R be a linearly ordered division ring. Let A be an R-linear map from from $(I \to R)$ to $(J \to R)$. Let b be an R-linear map from from $(I \to R)$ to R. Exactly one of the following exists:

- nonnegative vector $x : J \to R$ such that, for all $w : I \to R$, we have $\sum_{j:J} (A\ w)_j \bullet x_j = b\ w$
- vector $y : I \to R$ such that $A\ y \ge 0$ and $b\ y < 0$

In the next generalization, we replace the partially ordered module $I \to R$ by a general R-module W.

Theorem (scalarFarkas): Let J be a finite type. Let R be a linearly ordered division ring. Let W be an R-module. Let A be an R-linear map from from W to $(J \to R)$. Let b be an R-linear map from from W to R. Exactly one of the following exists:

- nonnegative vector $x : J \to R$ such that, for all $w : W$, we have $\sum_{j:J} (A\ w)_j \bullet x_j = b\ w$
- vector $y : W$ such that $A\ y \ge 0$ and $b\ y < 0$

In the most general theorem, stated below, we replace certain occurrences of R by a linearly ordered R-module V whose order respects order on R. This result origins from TODO.

Theorem (fintypeFarkasBartl): Let J be a finite type. Let R be a linearly ordered division ring. Let W be an R-module. Let V be a linearly ordered R-module[1]. Let A be an R-linear map from W to $(J \to R)$. Let b be an R-linear map from W to V. Exactly one of the following exists:

- nonnegative vector family $x : J \to V$ such that, for all $w : W$, we have $\sum_{j:J} (A\ w)_j \bullet x_j = b\ w$
- vector $y : W$ such that $A\ y \ge 0$ and $b\ y < 0$

In the last branch, $A\ y \ge 0$ uses the partial order[2] on $(J \to R)$ whereäs $b\ y < 0$ uses the linear order[3] on V. Note that fintypeFarkasBartl subsumes scalarFarkas (as well as the other versions based on equality), since R can be viewed as a linearly ordered module over itself. We prove fintypeFarkasBartl in Section TODO, which is where the heavy lifting comes.

Until now, we have talked about known results. What follows is a new extension of the theory.

Definition: Let F be a linearly ordered field. We define an **extended** linearly ordered field F_∞ as $F \cup \{\bot, \top\}$ with the following properties. Let p and q be numbers from F. We have $\bot < p < \top$. We define addition, scalar action, and negation on F_∞ as follows:

$+$	$\bot$	q	$\top$
$\bot$	$\bot$	$\bot$	$\bot$
p	$\bot$	$p+q$	$\top$
$\top$	$\bot$	$\top$	$\top$

$\bullet$	$\bot$	q	$\top$
0	$\bot$	0	0
$p > 0$	$\bot$	$p \cdot q$	$\top$

$-$	$\bot$	q	$\top$
	$\top$	$-q$	$\bot$

[1] We furthermore require monotonicity of scalar multiplication by nonnegative elements on the left. This assumption will be implicit in later occurrences.
[2] The order on $(J \to R)$ is always the coordinate-wise application of R's linear order.
[3] In case V has finite dimension, you can choose an arbitrary direction and project vectors from V onto it, or you can order elements of V lexicographically.

When we talk about elements of F_∞, we say that values from F are **finite**.

Informally speaking, $\top$ represents the positive infinity, $\bot$ represents the negative infinity, and we say that $\bot$ is "stronger" than $\top$ in all arithmetic operations. The surprising parts are $\bot + \top = \bot$ and $0 \bullet \bot = \bot$. Because of them, F_∞ is not a field. In fact, F_∞ is not even a group. However, F_∞ is still a densely linearly ordered abelian monoid with characteristic zero.

Theorem (extendedFarkas): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F_\infty$. Let b be a vector of type $I \to F_\infty$. Assume that A does not have $\bot$ and $\top$ in the same row. Assume that A does not have $\bot$ and $\top$ in the same column. Assume that A does not have $\top$ in any row where b has $\top$. Assume that A does not have $\bot$ in any row where b has $\bot$. Exactly one of the following exists:

- nonnegative vector $x : J \to F$ such that $A \cdot x \le b$
- nonnegative vector $y : I \to F$ such that $(-A^T) \cdot y \le 0$ and $b \cdot y < 0$

Note that extendedFarkas looks pretty much like equalityFarkas_neg and, in certain sense, generalizes it. Indeed, in Section TODO, we prove extendedFarkas using equalityFarkas_neg and some additional machinery.

Next we define an extended notion of linear program, i.e., linear programming over extended linearly ordered fields. The implicit intention is that the linear program is to be minimized.

Definition: Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F_\infty$, let b be a vector of type $I \to F_\infty$, and c be a vector of type $J \to F_\infty$ such that the following six conditions hold:

- A does not have $\bot$ and $\top$ in the same row
- A does not have $\bot$ and $\top$ in the same column
- b does not contain $\bot$
- c does not contain $\bot$
- A does not have $\top$ in any row where b has $\top$
- A does not have $\bot$ in any column where c has $\top$

We say that $P = (A, b, c)$ is a **linear program** over F_∞ whose constraints are indexed by I and variables are indexed by J. We say that a nonnegative vector $x : J \to R$ is a **solution** to P if and only if $A \cdot x \le b$. We say that P **reaches** an objective value r if and only if there exists x such that x is a solution to P and $c \cdot x = r$. We say that P is **feasible** if and only if P reaches a finite[4] value. We say that P is **bounded by** a finite value r if and only if, for every value p reached by P, we have $r \le p$. We say that P is **unbounded** if and only if there is no finite value r such that P is bounded by r. We say that the linear program $(-A^T, c, b)$ is the **dual** of P.

Theorem (weakDuality): Let F be a linearly ordered field. Let P be a linear program over F_∞. If P reaches p and the dual of P reaches q, then $p + q \ge 0$.

Definition: Let F be a linearly ordered field. Let P be a linear program over F_∞. We define the **optimum** of P as follows. If P is feasible and unbounded, its optimum is $\bot$. If P is not feasible, its optimum is $\top$. In all other cases, we ask whether P reaches a finite value r such that P is bounded by r. If so, its optimum is r. Otherwise, P does not have optimum.[5]

Theorem (strongDuality):[6] Let F be a linearly ordered field. Let P be a linear program over F_∞. If P or its dual is feasible (at least one of them), then there exists p in F_∞ such that P has optimum p and the dual of P has optimum $-p$.

2 Formalization

2.1 We start with a review of algebraic typeclasses that our project depends on

Additive semigroup is a structure on any type with addition where the addition is associative:

```
class AddSemigroup (G : Type u) extends Add G where
  add_assoc : ∀ a b c : G, (a + b) + c = a + (b + c)
```

Additive monoid is an additive semigroup with the zero element, thanks to which we can define a scalar multiplication by the natural numbers (TODO AddZeroClass):

```
class AddMonoid (M : Type u) extends AddSemigroup M, AddZeroClass M where
  nsmul : ℕ → M → M
  nsmul_zero : ∀ x : M, nsmul 0 x = 0
  nsmul_succ : ∀ (n : ℕ) (x : M), nsmul (n + 1) x = nsmul n x + x
```

[4] It would be perhaps more natural to say that P reaches a value different from $\top$. However, since $\bot$ cannot be reached because of the way linear programming is defined, it is equivalent to our definition by reaching a finite value.

[5] By the end of the paper, we will have proved that optimum always exists, i.e., it cannot happen that the set of objective values reached by P has a finite infimum that is not attained. However, because we do not have the theorem now, the optimum is defined as a partial function from linear programs to F_∞.

[6] For simplicity, we rephrased the theorem without mentioning partial functions. Also note that it would be incorrect to say the following: P has optimum p, the dual of P has optimum q, and $p + q = 0$. It would fail for unbounded linear programs because the arithmetics of F_∞ defines $\top + \bot = \bot$.

Subtractive monoid is an additive monoid that adds two more operations (unary and binary minus) that satisfy some basic properties:

```
class SubNegMonoid (G : Type u) extends AddMonoid G, Neg G, Sub G where
  sub := SubNegMonoid.sub'
  sub_eq_add_neg : ∀ a b : G, a - b = a + -b
  zsmul : ℤ → G → G
  zsmul_zero' : ∀ a : G, zsmul 0 a = 0
  zsmul_succ' (n : ℕ) (a : G) : zsmul (Int.ofNat n.succ) a = zsmul (Int.ofNat n) a + a
  zsmul_neg' (n : ℕ) (a : G) : zsmul (Int.negSucc n) a = -(zsmul n.succ a)
```

Additive group is a subtractive monoid in which the unary minus acts as an inverse with respect to addition:

```
class AddGroup (A : Type u) extends SubNegMonoid A where
  add_left_neg : ∀ a : A, -a + a = 0
```

Abelian group is defined as an additive group that is a commutative additive monoid at the same time (TODO AddCommMonoid):

```
class AddCommGroup (G : Type u) extends AddGroup G, AddCommMonoid G
```

Ring is defined as a semiring that is an abelian group at the same time and has 1 that behaves well (TODO Semiring):

```
class Ring (R : Type u) extends Semiring R, AddCommGroup R, AddGroupWithOne R
```

Division ring is a ring with a lot of extra requirements (TODOs DivInvMonoid, Nontrivial, NNRatCast, RatCast):

```
class DivisionRing (α : Type*) extends Ring α, DivInvMonoid α, Nontrivial α, NNRatCast α, RatCast α where
  mul_inv_cancel : ∀ (a : α), a ≠ 0 → a * a⁻¹ = 1
  inv_zero : (0 : α)⁻¹ = 0
  nnratCast := NNRat.castRec
```

We define a linearly ordered division ring as a division ring that is a linearly ordered ring at the same time (TODO all about order):

```
class LinearOrderedDivisionRing (R : Type*) extends LinearOrderedRing R, DivisionRing R
```

Linearly ordered field is defined as a linearly ordered commutative ring that is a field at the same time (TODO Field):

```
class LinearOrderedField (α : Type*) extends LinearOrderedCommRing α, Field α
```

Note that LinearOrderedDivisionRing is not a part of the algebraic hierarchy provided by Mathlib, hence LinearOrderedField does not inherit LinearOrderedDivisionRing, thus we provide a custom instance that converts LinearOrderedField to LinearOrderedDivisionRing:

```
instance LinearOrderedField.toLinearOrderedDivisionRing {F : Type*} [instF : LinearOrderedField F] :
  LinearOrderedDivisionRing F := { instF with }
```

This instance is needed for the step from coordinateFarkas to equalityFarkas.

2.2 Extended linearly ordered fields

Given any type F, we construct $F \cup \{\bot, \top\}$ as follows:

```
def Extend (F : Type*) := WithBot (WithTop F)
```

From now on we assume that F is a linearly ordered field:

```
variable {F : Type*} [LinearOrderedField F]
```

The following instance defines how addition and comparison behaves on F_∞ and automatically generates a proof that F_∞ forms a linearly ordered abelian monoid:

```
instance : LinearOrderedAddCommMonoid (Extend F) :=
  inferInstanceAs (LinearOrderedAddCommMonoid (WithBot (WithTop F)))
```

The following instance provides a proof that the constant 1 behaves with respect to addition the way it should:

```
instance : AddCommMonoidWithOne (Extend F) :=
  inferInstanceAs (AddCommMonoidWithOne (WithBot (WithTop F)))
```

The following instance provides a proof that $0 \leq 1$ holds:

```
instance : ZeroLEOneClass (Extend F) := inferInstanceAs (ZeroLEOneClass (WithBot (WithTop F)))
```

The following instance provides a proof F_∞ as a monoid has characteristic zero, i.e., you cannot obtain 0 by summing up 1 repeatedly:

```
instance : CharZero (Extend F) := inferInstanceAs (CharZero (WithBot (WithTop F)))
```

The following instance provides a proof F_∞ has a bounded order:

```
instance : BoundedOrder (Extend F) := inferInstanceAs (BoundedOrder (WithBot (WithTop F)))
```

The following instance provides a proof F_∞ is densely ordered:

```
instance : DenselyOrdered (Extend F) := inferInstanceAs (DenselyOrdered (WithBot (WithTop F)))
```

The following instance provides a proof that $<$ is decidable on F_∞ if it is decidable on F:

```
instance : DecidableRel ((· < ·) : Extend F → (Extend F) → Prop) := WithBot.decidableLT
```

The following definition embeds F in F_∞ and registers this canonical embedding as a type coercion:

```
@[coe] def toE : F → (Extend F) := some ∘ some
instance : Coe F (Extend F) := ⟨toE⟩
```

In the file `FarkasSpecial.lean` and everything downstream, we have the notation

```
F∞
```

for F_∞ and also the notation

```
F≥0
```

for all nonnegative elements of F. TODO define scalar action and explain SMulZeroClass.

2.3 Vectors and stuff

We distinguish two types of vectors; implicit vectors and explicit vectors. Implicit vectors are members of a vector space; they don't have any internal structure. Explicit vectors are functions from coordinates to values. The set of coordinates needn't be ordered. Matrices live next to explicit vectors. They are also functions; they take a row index and a column index and they output a value at the given spot. Neither the row indices nor the column vertices are required to form an ordered set. That's why multiplication between matrices and vectors is defined only in structures where addition forms a commutative semigroup. Consider the following example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} ? \\ \text{-} \end{pmatrix}$$

We don't know whether the value at the question mark is equal to $(1 \cdot 7 + 2 \cdot 8) + 3 \cdot 9$ or to $(2 \cdot 8 + 1 \cdot 7) + 3 \cdot 9$ or to any other ordering of summands. This is why commutativity of addition is necessary for the definition to be valid. On the other hand, we don't assume any property of multiplication in the definition of multiplication between matrices and vectors; they don't even have to be of the same type; we only require the elements of the vector to have an action on the elements of the matrix (this is not a typo – normally, we would want matrices to have an action on vectors – not in our work).

TODO formal definitions.

2.4 Linear programming

Extended linear programs are defined as follows:

```
structure ExtendedLP (I J F : Type*) [LinearOrderedField F] where
  /-- The left-hand-side matrix. -/
  A : Matrix I J F∞
  /-- The right-hand-side vector. -/
  b : I → F∞
  /-- The objective function coefficients. -/
  c : J → F∞
  /-- No `⊥` and `⊤` in the same row. -/
  hAi : ¬∃ i : I, (∃ j : J, A i j = ⊥) ∧ (∃ j : J, A i j = ⊤)
  /-- No `⊥` and `⊤` in the same column. -/
  hAj : ¬∃ j : J, (∃ i : I, A i j = ⊥) ∧ (∃ i : I, A i j = ⊤)
  /-- No `⊥` in the right-hand-side vector. -/
  hbi : ¬∃ i : I, b i = ⊥
  /-- No `⊥` in the objective function coefficients. -/
  hcj : ¬∃ j : J, c j = ⊥
  /-- No `⊤` in the row where the right-hand-side vector has `⊤`. -/
  hAb : ¬∃ i : I, (∃ j : J, A i j = ⊤) ∧ b i = ⊤
  /-- No `⊥` in the column where the objective function has `⊤`. -/
  hAc : ¬∃ j : J, (∃ i : I, A i j = ⊥) ∧ c j = ⊤
```

Solution is defined as follows:

```
def ExtendedLP.IsSolution (P : ExtendedLP I J F) (x : J → F≥0) : Prop :=
  P.A ₘ* x ≤ P.b
```

Reaching a value is defined as follows:

```
def ExtendedLP.Reaches (P : ExtendedLP I J F) (r : F∞) : Prop :=
  ∃ x : J → F≥0, P.IsSolution x ∧ P.c ᵥ⬝ x = r
```

Feasibility is defined as follows:

```
def ExtendedLP.IsFeasible (P : ExtendedLP I J F) : Prop :=
  ∃ p : F, P.Reaches (toE p)
```

Being bounded by a value (from below – we always minimize) is defined as follows:

```
def ExtendedLP.IsBoundedBy (P : ExtendedLP I J F) (r : F) : Prop :=
  ∀ p : F∞, P.Reaches p → r ≤ p
```

Being unbounded is defined as follows:

```
def ExtendedLP.IsUnbounded (P : ExtendedLP I J F) : Prop :=
  ¬∃ r : F, P.IsBoundedBy r
```

The following definition says how linear programs are dualized:

```
def ExtendedLP.dualize (P : ExtendedLP I J F) : ExtendedLP J I F :=
  ⟨-P.Aᵀ, P.c, P.b, by aeply P.hAj, by aeply P.hAi, P.hcj, P.hbi, by aeply P.hAc, by aeply P.hAb⟩
```

The definition of optimum is, sadly, very complicated:

```
noncomputable def ExtendedLP.optimum (P : ExtendedLP I J F) : Option F∞ :=
  if P.IsFeasible then
    if P.IsUnbounded then
      some none --some ⊥ -- unbounded means that the minimum is ‘⊥‘
    else
      if hf : ∃ r : F, P.Reaches (toE r) ∧ P.IsBoundedBy r then
        some (toE hf.choose) -- the minimum is finite
      else
        none -- invalid finite value (infimum is not attained)
  else
    some ⊤ -- infeasible means that the minimum is ‘⊤‘
```

Finally, we define what opposite values are:

```
def OppositesOpt : Option F∞ → Option F∞ → Prop
| (p : F∞), (q : F∞) => p = -q  -- opposite values; includes ‘⊥ = -⊤‘ and ‘⊤ = -⊥‘
| _       , _        => False   -- namely ‘OppositesOpt none none = False‘
```

2.5 Results

Theorem equalityFredholm is stated as follows:

```
theorem equalityFredholm (A : Matrix I J F) (b : I → F) :
    (∃ x : J → F, A *ᵥ x = b) ≠ (∃ y : I → F, Aᵀ *ᵥ y = 0 ∧ b ⬝ᵥ y ≠ 0)
```

Theorem equalityFredholm_lt is stated as follows:

```
theorem equalityFredholm_lt (A : Matrix I J F) (b : I → F) :
    (∃ x : J → F, A *ᵥ x = b) ≠ (∃ y : I → F, Aᵀ *ᵥ y = 0 ∧ b ⬝ᵥ y < 0)
```

Theorem equalityFarkas is stated as follows:

```
theorem equalityFarkas (A : Matrix I J F) (b : I → F) :
    (∃ x : J → F, 0 ≤ x ∧ A *ᵥ x = b) ≠ (∃ y : I → F, 0 ≤ Aᵀ *ᵥ y ∧ b ⬝ᵥ y < 0)
```

Theorem inequalityFarkas is stated as follows:

```
theorem inequalityFarkas [DecidableEq I] (A : Matrix I J F) (b : I → F) :
    (∃ x : J → F, 0 ≤ x ∧ A *ᵥ x ≤ b) ≠ (∃ y : I → F, 0 ≤ y ∧ 0 ≤ Aᵀ *ᵥ y ∧ b ⬝ᵥ y < 0)
```

Theorem inequalityFarkas_neg is stated as follows:

```
theorem inequalityFarkas_neg [DecidableEq I] (A : Matrix I J F) (b : I → F) :
    (∃ x : J → F, 0 ≤ x ∧ A *ᵥ x ≤ b) ≠ (∃ y : I → F, 0 ≤ y ∧ -Aᵀ *ᵥ y ≤ 0 ∧ b ⬝ᵥ y < 0)
```

Theorem coordinateFarkas is stated as follows:

```
theorem coordinateFarkas {I J : Type*} [Fintype J] [LinearOrderedDivisionRing R]
    (A : (I → R) →ₗ[R] J → R) (b : (I → R) →ₗ[R] R) :
    (∃ x : J → R, 0 ≤ x ∧ ∀ w : I → R, ∑ j : J, A w j • x j = b w) ≠ (∃ y : I → R, 0 ≤ A y ∧ b y < 0)
```

Theorem scalarFarkas is stated as follows:

```
theorem scalarFarkas {J : Type*} [Fintype J] [LinearOrderedDivisionRing R] [AddCommGroup W] [Module R W]
    (A : W →ₗ[R] J → R) (b : W →ₗ[R] R) :
    (∃ x : J → R, 0 ≤ x ∧ ∀ w : W, ∑ j : J, A w j • x j = b w) ≠ (∃ y : W, 0 ≤ A y ∧ b y < 0)
```

Theorem fintypeFarkasBartl is stated as follows:

```
theorem fintypeFarkasBartl {J : Type*} [Fintype J] [LinearOrderedDivisionRing R]
    [LinearOrderedAddCommGroup V] [Module R V] [PosSMulMono R V] [AddCommGroup W] [Module R W]
    (A : W →ₗ[R] J → R) (b : W →ₗ[R] V) :
    (∃ x : J → V, 0 ≤ x ∧ ∀ w : W, ∑ j : J, A w j • x j = b w) ≠ (∃ y : W, 0 ≤ A y ∧ b y < 0)
```

The existence of optimum (minimum) for every linear program is stated as follows:

```
theorem ExtendedLP.optimum_neq_none (P : ExtendedLP I J F) : P.optimum ≠ none
```

The weak duality theorem is stated as follows:

```
theorem ExtendedLP.weakDuality [DecidableEq I] [DecidableEq J] {P : ExtendedLP I J F}
  {p : F∞} (hP : P.Reaches p) {q : F∞} (hQ : P.dualize.Reaches q) :
  0 ≤ p + q
```

The strong duality theorem is stated as follows:

```
theorem ExtendedLP.strongDuality {P : ExtendedLP I J F} (hP : P.IsFeasible ∨ P.dualize.IsFeasible) :
  OppositesOpt P.optimum P.dualize.optimum
```

TODO somehow politely say that the Mathlib's API for block matrices is a mess and needs an overhaul.

3 Proving the Farkas-Bartl theorem

We prove finFarkasBartl and, in the end, we obtain fintypeFarkasBartl as corollary.

Theorem (finFarkasBartl): Let n be a natural number. Let R be a linearly ordered division ring. Let W be an R-module. Let V be a linearly ordered R-module. Let A be an R-linear map from W to $([n] \to R)$. Let b be an R-linear map from W to V. Exactly one of the following exists:

- nonnegative vector family $x : [n] \to V$ such that, for all $w : W$, we have $\sum_{j:[n]} (A\ w)_j \bullet x_j = b\ w$
- vector $y : W$ such that $A\ y \geq 0$ and $b\ y < 0$

The only difference is that finFarkasBartl uses $[n] = \{0, \ldots, n-1\}$ instead of an arbitrary (unordered) finite type J.

Proof idea: We first prove that both cannot exist at the same time. Assume we have x and y of said properties. We plug y for w and obtain $\sum_{j:[n]} (A\ y)_j \bullet x_j = b\ y$. On the left-hand side, we have a sum of nonnegative vectors, which contradicts $b\ y < 0$.

We prove "at least one exists" by induction on n. If $n = 0$ then $A\ y \geq 0$ is a tautology. We consider b. Either b maps everything to the zero vector, which allows x to be the empty vector family, or some w gets mapped to a nonzero vector, where we choose y to be either w or $(-w)$. Since V is linearly ordered, one of them satisfies $b\ y < 0$. Now we precisely state how the induction step will be.

Lemma (industepFarkasBartl): Let m be a natural number. Let R be a linearly ordered division ring. Let W be an R-module. Let V be a linearly ordered R-module. Assume (induction hypothesis) that for all R-linear maps $A_0 : W \to ([m] \to R)$ and $b_0 : W \to V$, the formula "$\forall y_0 : W,\ A_0\ y_0 \geq 0 \implies b_0\ y_0 \geq 0$" implies existence of a nonnegative vector family $x_0 : [m] \to V$ such that, for all $w_0 : W$, $\sum_{i:[m]} (A_0\ w_0)_i \bullet (x_0)_i = b_0\ w_0$. Let A be an R-linear map from W to $([m+1] \to R)$. Let b be an R-linear map from W to V. Assume that, for all $y : W$, $A\ y \geq 0$ implies $b\ y \geq 0$. We claim there exists a nonnegative vector family $x : [m+1] \to V$ such that, for all $w : W$, we have $\sum_{i:[m+1]} (A\ w)_i \bullet x_i = b\ w$. TODO names like A_0 don't work well with the "subscript notation" on paper.

Proof idea: Let $A_{<m}$ roughly mean $A \restriction [m]$. To be more precise, $A_{<m}$ is a function that maps $(w : W)$ to $(A\ w) \restriction [m]$, i.e., $A_{<m}$ is an R-linear map from W to $([m] \to R)$ that behaves exactly like A where it is defined. We distinguish two cases. If, for all $y : W$, the inequality $A_{<m}\ y \geq 0$ implies $b\ y \geq 0$, then plug $A_{<m}$ for A_0, obtain x_0, and construct a vector family x such that $x_m = 0$ and otherwise x copies x_0. We easily check that x is nonnegative and that $\sum_{i:[m+1]} (A\ w)_i \bullet x_i = b\ w$ holds.

In the second case, we have y' such that $A_{<m}\ y' \geq 0$ holds but $b\ y' < 0$ also holds. We realize that $(A\ y')_m < 0$. We now declare $y := (A\ y')_m \bullet y'$ and observe $(A\ y)_m = 1$. We establish the following facts (proofs are omitted):

- for all $w : W$, we have $A\ (w - ((A\ w)_m \bullet y)) = 0$
- for all $w : W$, the inequality $A_{<m}\ (w - ((A\ w)_m \bullet y)) \geq 0$ implies $b\ (w - ((A\ w)_m \bullet y)) \geq 0$
- for all $w : W$, the inequality $A_{<m}\ w - A_{<m}\ ((A\ w)_m \bullet y) \geq 0$ implies $b\ w - b\ ((A\ w)_m \bullet y) \geq 0$
- for all $w : W$, the inequality $(A_{<m} - (z \mapsto (A\ z)_m \bullet (A_{<m}\ y)))\ w \geq 0$ implies $(b - (z \mapsto (A\ z)_m \bullet (b\ y)))\ w \geq 0$

We observe that $A_0 := A_{<m} - (z \mapsto (A\ z)_m \bullet (A_{<m}\ y))$ and $b_0 := b - (z \mapsto (A\ z)_m \bullet (b\ y))$ are R-linear maps. Thanks to the last fact, we can apply induction hypothesis to A_0 and b_0. We obtain a nonnegative vector family x' such that, for all $w_0 : W$, $\sum_{i:[m]}(A_0\ w_0)_i \bullet x'_i = b_0\ w_0$. It remains to construct a nonnegative vector family $x : [m+1] \to V$ such that, for all $w : W$, we have $\sum_{i:[m+1]} (A\ w)_i \bullet x_i = b\ w$. We choose $x_m = b\ y - \sum_{i:[m]}(A_{<m}\ y)_i \bullet x'_i$ and otherwise x copies x'. We check that our x has the required properties. Qed.

We complete the proof of finFarkasBartl by applying industepFarkasBartl to $A_{\leq n}$ and b. Finally, we obtain fintypeFarkasBartl from finFarkasBartl using some boring mechanisms regarding equivalence between finite types.

4 Extended Farkas theorem

Theorem (extendedFarkas): Let I and J be finite types. Let F be a linearly ordered field. Let A be a matrix of type $(I \times J) \to F_\infty$. Let b be a vector of type $I \to F_\infty$. Assume that A does not have $\perp$ and $\top$ in the same row. Assume that A does not have $\perp$ and $\top$ in the same column. Assume that A does not have $\top$ in any row where b has $\top$. Assume that A does not have $\perp$ in any row where b has $\perp$. Exactly one of the following exists:

- nonnegative vector $x : J \to F$ such that $A \cdot x \le b$
- nonnegative vector $y : I \to F$ such that $(-A^T) \cdot y \le 0$ and $b \cdot y < 0$

(restated)

4.1 Proof idea

We need to do the following steps in given order:

1. Delete all rows of both A and b where A has $\perp$ or b has $\top$ (they are tautologies).
2. Delete all columns of A that contain $\top$ (they force respective variables to be zero).
3. If b contains $\perp$, then $A \cdot x \le b$ cannot be satisfied, but $y = 0$ satisfies $(-A^T) \cdot y \le 0$ and $b \cdot y < 0$. Stop here.
4. Assume there is no $\perp$ in b. Use inequalityFarkas_neg. In either case, extend x or y with zeros on all deleted positions.

4.2 Counterexamples

If A has $\perp$ and $\top$ in the same row, it may happen that both x and y exist:

$$A = \begin{pmatrix} \perp & \top \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

If A has $\perp$ and $\top$ in the same column, it may happen that both x and y exist:

$$A = \begin{pmatrix} \perp \\ \top \end{pmatrix} \qquad b = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \qquad x = \begin{pmatrix} 0 \end{pmatrix} \qquad y = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

If A has $\top$ in a row where b has $\top$, it may happen that both x and y exist:

$$A = \begin{pmatrix} \top \\ -1 \end{pmatrix} \qquad b = \begin{pmatrix} \top \\ -1 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

If A has $\perp$ in a row where b has $\perp$, it may happen that both x and y exist:

$$A = \begin{pmatrix} \perp \end{pmatrix} \qquad b = \begin{pmatrix} \perp \end{pmatrix} \qquad x = \begin{pmatrix} 1 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \end{pmatrix}$$

5 Proving the extended strong LP duality

We start with the weak duality and then move to the strong duality. We will use extendedFarkas in several places.

Theorem (weakDuality): Let F be a linearly ordered field. Let P be a linear program over F_∞. If P reaches p and the dual of P reaches q, then $p + q \ge 0$. (restated)

Proof idea: There is a vector x such that $A \cdot x \le b$ and $c \cdot x = p$. Apply extendedFarkas to the following matrix and vector:

$$\begin{pmatrix} A \\ c \end{pmatrix} \qquad \begin{pmatrix} b \\ c \cdot x \end{pmatrix}$$

Lemma (infeasible_of_unbounded): If a linear program P is unbounded, the dual of P cannot be feasible.

Proof idea: Assume that P is unbounded, but the dual of P is feasible. Obtain contradiction using weakDuality.

Lemma (unbounded_of_reaches_le): Let F be a linearly ordered field. Let P be a linear program over F_∞. Assume that for each s in F there exists p in F_∞ such that P reaches p and $p \le s$. We conclude that P is unbounded.

Proof idea: It suffices to prove that for each r in F there exists p' in F_∞ such that P reaches p' and $p' < r$. Apply the assumption to $r - 1$.

Lemma (unbounded_of_feasible_of_neg): Let F be a linearly ordered field. Let P be a linear program over F_∞ that is feasible. Let x_0 be a nonnegative vector such that $c \cdot x_0 < 0$ and $A \cdot x_0 + 0 \bullet (-b) \le 0$. We conclude that P is unbounded.

Proof idea: There is a nonnegative vector x_p such that $A \cdot x_p \le b$ and $c \cdot x_p = e$ for some e in F_∞. We apply unbounded_of_feasible_of_neg. In case $e \le s$, we use x_p and we are done. Otherwise, consider what $c \cdot x_0$ equals to. We cannot have $c \cdot x_0 = \bot$ because c does not contain $\bot$. We cannot have $c \cdot x_0 = \top$ because $c \cdot x_0 < 0$. Hence $c \cdot x_0 = d$ for some d in F. Observe that the fraction $\frac{s-e}{d}$ is well defined and it is positive. Use $x_p + \frac{s-e}{d} \bullet x_0$.

Lemma (strongDuality_aux): Let P be a linear program such that P is feasible and the dual of P is also feasible. There is a value p reached by P and a value q reached by the dual of P such that $p + q \le 0$.

Proof idea: TODO.

Lemma (strongDuality_of_both_feasible): Let P be a linear program such that P is feasible and the dual of P is also feasible. There is a finite value r such that P reaches $-r$ and the dual of P reaches r.

Proof idea: From strongDuality_aux we have a value p reached by P and a value q reached by the dual of P such that $p + q \le 0$. We apply weakDuality to p and q to obtain $p + q \ge 0$. We set $r := q$.

Lemma (unbounded_of_feasible_of_infeasible): Let P be a linear program such that P is feasible but the dual of P is not feasible. We conclude that P is unbounded.

Proof idea: TODO.

Lemma (optimum_unique): Let P be a linear program. Let r be a valued reached by P such that P is bounded by r. Let s be a valued reached by P such that P is bounded by s. We conclude $r = s$.

Proof idea: TODO.

Lemma (optimum_eq_of_reaches_bounded): Let P be a linear program. Let r be a valued reached by P such that P is bounded by r. We conclude that the optimum of P is r.

Proof idea: Apply the axiom of choice to the definition of optimum and use optimum_unique.

Lemma (strongDuality_of_prim_feas): Let P be a linear program that is feasible. The strong duality holds.

Proof idea: TODO.

Theorem (optimum_neq_none): Every linear program has optimum.

Proof idea: If a linear program P is feasible, the existence of optimum follows from strongDuality_of_prim_feas. Otherwise, the optimum of P is $\top$ by definition.

Lemma (dualize_dualize): Let P be a linear program. The dual of the dual of P is exactly P.

Proof idea: $-(-A^T)^T = A$

Lemma (strongDuality_of_dual_feas): Let P be a linear program whose dual is feasible. The strong duality holds.

Proof idea: Apply strongDuality_of_prim_feas to the dual of P and use dualize_dualize.

Theorem (strongDuality): Let F be a linearly ordered field. Let P be a linear program over F_∞. If P or its dual is feasible (at least one of them), then there exists p in F_∞ such that P has optimum p and the dual of P has optimum $-p$. (restated)

Proof idea: Use strongDuality_of_prim_feas or strongDuality_of_dual_feas.

6 Related work

TODO presents an overcomplicated proof in Isabelle by analyzing the Simplex algorithm that already had been formally verified. It took them 30 pages to get to the basic Farkas; no generalization was provided.

In Lean, it would be possible to prove Farkas for reals using the Hahn-Banach separation theorem. However, we do not yet know that the set of feasible solutions is closed.

TODO.

7 Conclusion

We formally verified several Farkas-like theorems in Lean 4. We extended the existing theory to a new setting where some coefficient can carry infinite values. We realized that the abstract work with modules over linearly ordered division rings and linear maps between them was fairly easy to carry on in Lean 4 thanks to the library Mathlib that is perfectly suited for such tasks. In contrast, manipulation with matrices got tiresome whenever we needed a not-fully-standard operation. It turns out Lean 4 cannot automate case analyses unless they take place in the "outer layers" of formulas. Summation over subtypes and summation of conditional expression made us developed a lot of ad-hoc machinery which we would have preferred to be handled by existing tactics. Another area where Lean 4 is not yet helpful is the search for counterexamples. Despite these difficulties, we find Lean 4 to be an extremely valuable tool for elegant expressions of mathematical formulas and for proving them formally.