Gyula 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:

```
theorem equalityFarkas [Fintype I] [Fintype J] [LinearOrderedField F] (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)
```

Geometric interpretation of `equalityFarkas` is as follows. The column vectors of `A` generate a cone in the `|I|`-dimensional Euclidean space from the origin. 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).

The next theorem generalizes `equalityFarkas` to structures where multiplication does not have to be commutative; furthermore, it supports infinitely many equations:

```
theorem coordinateFarkasBartl {I : 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)
```

In the next generalization, the partially ordered module `I → R` is replaced by a general `R`-module `W`:

```
theorem almostFarkasBartl [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)
```

In the most general theorem, stated below, certain occurrences of `R` are replaced by a linearly ordered `R`-module `V` whose order respects the order on `R`:

```
theorem fintypeFarkasBartl [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)
```

Note that `fintypeFarkasBartl` subsumes `scalarFarkas` as well as the other versions, since `R` can be viewed as a linearly ordered module over itself.

We have hereby stated a three-fold generalization of the original Farkas' result. Let's prove it! Our proof, starting on the next page, is based on a modern algebraic proof by David Bartl. We first prove a tiny-bit-less-general version `finFarkasBartl` which uses `Fin n` (i.e., indexing by natural numbers between `0` inclusive and `n` exclusive) instead of an arbitrary (unordered) finite type `J`. In the end, we obtain `fintypeFarkasBartl` from `finFarkasBartl` using some boring mechanisms regarding equivalence between finite types.

Figure 1: AI-generated image fills the rest of the first page.

Proving finFarkasBartl

In this section, we prove:

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

We first rephrase the goal to:

```
(∃ x : Fin n → V, 0 ≤ x ∧ ∀ w : W, ∑ j : Fin n, A w j • x j = b w) ↔ (∀ y : W, 0 ≤ A y → 0 ≤ b y)
```

Implication from left to right is immediately satisfied by the following term:

```
fun ⟨x, hx, hb⟩ y hy => hb y ▸ Finset.sum_nonneg (fun i _ => smul_nonneg (hy i) (hx i))
```

Implication from right to left will be proved by induction on `n` with generalized `A` and `b`. In case `n = 0` we immediately have:

```
A_tauto (w : W) : 0 ≤ A w
```

We have an assumption:

```
hAb : ∀ y : W, 0 ≤ A y → 0 ≤ b y
```

We set `x` to be the empty vector family. Now, for every `w : W`, we must prove:

```
∑ j : Fin 0, A w j • (0 : Fin 0 → V) j = b w
```

We simplify the goal to:

```
0 = b w
```

We utilize that `V` is ordered and prove the equality as two inequalities. Inequality `0 ≤ b w` is directly satisfied by:

```
hAb w (A_tauto w)
```

Inequality `b w ≤ 0` is easily reduced to:

```
hAb (-w) (A_tauto (-w))
```

The induction step is stated as a lemma:

```
lemma industepFarkasBartl {m : ℕ} [LinearOrderedDivisionRing R]
    [LinearOrderedAddCommGroup V] [Module R V] [PosSMulMono R V] [AddCommGroup W] [Module R W]
    (ih : ∀ A₀ : W →ₗ[R] Fin m → R, ∀ b₀ : W →ₗ[R] V,
      (∀ y₀ : W, 0 ≤ A₀ y₀ → 0 ≤ b₀ y₀) →
        (∃ x₀ : Fin m → V, 0 ≤ x₀ ∧ ∀ w₀ : W, ∑ i₀ : Fin m, A₀ w₀ i₀ • x₀ i₀ = b₀ w₀))
    {A : W →ₗ[R] Fin m.succ → R} {b : W →ₗ[R] V} (hAb : ∀ y : W, 0 ≤ A y → 0 ≤ b y) :
    ∃ x : Fin m.succ → V, 0 ≤ x ∧ ∀ w : W, ∑ i : Fin m.succ, A w i • x i = b w
```

We define

```
a : W →ₗ[R] Fin m → R
```

as the first `m` rows of `A` (i.e., `A` without the last row):

```
a := (fun w : W => fun i : Fin m => A w i)
```

To prove `industepFarkasBartl` we first consider the easy case:

```
is_easy : ∀ y : W, 0 ≤ a y → 0 ≤ b y
```

From `ih a b is_easy` we obtain:

```
x : Fin m → V
hx : 0 ≤ x
hxb : ∀ w₀ : W, ∑ i₀ : Fin m, a w₀ i₀ • x i₀ = b w₀
```

The lemma is satisfied by this vector family:

```
(fun i : Fin m.succ => if hi : i < m then x i else 0)
```

Easy case analysis shows that the vector family is nonnegative. Now we need to prove:

```
∀ w : W, ∑ i : Fin m.succ, A w i • (fun i : Fin m.succ => if hi : i < m then x i else 0) i = b w
```

We simplify the goal to:

```
∀ w : W, ∑ i : Fin m, A w i • x i = b w
```

This is exactly `hxb`.

Now for the hard case; negation of `is_easy` gives us:

```
y' : W
hay' : 0 ≤ a y'
hby' : b y' < 0
```

Let `y` be flipped and rescaled `y'` as follows:

$\texttt{y : W := (A y' m)}^{-1} \bullet \texttt{ y'}$

From `hAb` we get:

```
hAy' : A y' m < 0
```

Therefore `hAy'.ne : A y' m ≠ 0` implies that `y` has the property that motivated the rescaling:

```
hAy : A y m = 1
```

From `hAy` we have:

```
hAA : ∀ w : W, A (w - (A w m • y)) m = 0
```

Using `hAA` and `hAb` we prove:

```
hbA : ∀ w : W, 0 ≤ a (w - (A w m • y)) → 0 ≤ b (w - (A w m • y))
```

From `hbA` we have:

```
hbAb : ∀ w : W, 0 ≤ (a - (A · m • a y)) w → 0 ≤ (b - (A · m • b y)) w
```

We observe that these two terms (appearing in `hbAb` we just proved) are linear maps:

```
(a - (A · m • a y))
(b - (A · m • b y))
```

Therefore, we can plug them into `ih` and provide `hbAb` as the last argument. We obtain:

```
x' : Fin m → V
hx' : 0 ≤ x'
hxb' : ∀ w_0 : W, ∑ i_0 : Fin m, (a - (A · m • a y)) w_0 i_0 • x' i_0 = (b - (A · m • b y)) w_0
```

We claim that our lemma is satisfied by this vector family:

```
(fun i : Fin m.succ => if hi : i < m then x' i else b y - ∑ j : Fin m, a y i • x' j)
```

Let us show the nonnegativity first. Nonnegativity of everything except of the last vector follows from `hx'`. From `hAy'` we have:

$\texttt{hAy'' : (A y' m)}^{-1} \le \texttt{0}$

From `hAy''` with `hay'` we have:

```
hay : a y ≤ 0
```

From `hAy''` with `hby'` converted to nonstrict inequality we have:

```
hby : 0 ≤ b y
```

For the nonnegativity of the last vector, we need to prove:

```
∑ j : Fin m, a y j • x' j ≤ b y
```

It follows from `hay j` with `hx' j` and `hby` using basic properties of inequalities. The only remaining task is to show:

```
∀ w : W,
  ∑ i : Fin m.succ, (A w i • (if hi : i < m then x' i else b y - ∑ j : Fin m, a y j • x' j)) = b w
```

Given general `w : W` we make a key observation (using `hxb' w`):

```
haAa : ∑ i : Fin m, (a w i - A w m * a y i) • x' i = b w - A w m • b y
```

With the help of `haAa` we transform the goal to:

```
∑ i : Fin m.succ, (A w i • (if hi : i < m then x' i else b y - ∑ j : Fin m, a y j • x' j)) =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

We distribute • over `if` so that the goal becomes:

```
∑ i : Fin m.succ, (if hi : i < m then A w i • x' i else A w i • (b y - ∑ j : Fin m, a y j • x' j)) =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

We split the left-hand side into two parts:

```
∑ i : Fin m, (a w i • x' i) + A w M • (b y - ∑ j : Fin m, a y j • x' j) =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

The rest is a simple manipulation with sums.