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 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
```

First we introduce an auxiliary definition. 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.castSucc)
```

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.val < m then x ⟨i.val, hi⟩ else 0)
```

Easy case analysis shows that the vector family is nonnegative. In order to prove that, given `w : W` in the context,

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

holds, we first transform the goal to:

```
∑ i ∈ (Finset.univ.filter (fun k : Fin m.succ => k.val < m)).attach, A w i.val • x ⟨i.val.val, _⟩ = b w
```

We compare it with `hxb w` which says:

```
∑ i₀ : Fin m, a w i₀ • x i₀ = b w
```

We finish the proof for the easy case using the following technical lemma (which will also be used in one more place):

```
private lemma finishing_piece {m : ℕ} [Semiring R]
    [AddCommMonoid V] [Module R V] [AddCommMonoid W] [Module R W]
    {A : W →ₗ[R] Fin m.succ → R} {w : W} {x : Fin m → V} :
    ∑ i : Fin m, a w i • x i =
    ∑ i : { j : Fin m.succ // j ∈ Finset.univ.filter (·.val < m) }, A w i.val • x ⟨i.val.val, by aesop⟩
```

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

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

Let us make an alias for the last (new) index, i.e., the term `M` is just the number `m` converted to the type `Fin (m+1)`:

```
M : Fin m.succ := ⟨m, lt_add_one m⟩
```

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

```
y : W := (A y' M)⁻¹ • 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' : ∀ w₀ : W, ∑ i₀ : Fin m, (a - (A · M • a y)) w₀ i₀ • x' i₀ = (b - (A · M • b y)) w₀
```

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

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

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

```
hAy'' : (A y' M)⁻¹ ≤ 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:

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

It follows from `hay i` with `hx' i` 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.val < m then x' ⟨i.val, hi⟩ else b y - ∑ i : Fin m, a y i • x' i)
  ) =
  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.val < m then x' ⟨i.val, hi⟩ else b y - ∑ i' : Fin m, a y i' • x' i')) =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

From here, the direction should be clear; the rest of the proof is just a manipulation with the goal without any additional hypotheses. We distribute • over `if` so that the goal becomes:

```
∑ i : Fin m.succ,
  (if hi : i.val < m then A w i • x' ⟨i.val, hi⟩ else A w i • (b y - ∑ i' : Fin m, a y i' • x' i')) =
∑ 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 ∈ (Finset.univ.filter (fun i : Fin m.succ => i.val < m)).attach,
  A w i.val • x' ⟨i.val.val, _⟩ +
∑ i ∈ (Finset.univ.filter (fun i : Fin m.succ => ¬(i.val < m))).attach,
  A w i.val • (b y - ∑ i' : Fin m, a y i' • x' i') =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

Since the second sum is singleton, it simplifies to:

```
∑ i ∈ (Finset.univ.filter (fun i : Fin m.succ => i.val < m)).attach, A w i.val • x' ⟨i.val.val, _⟩ +
A w M • (b y - ∑ i : Fin m, a y i • x' i) =
∑ i : Fin m, (a w i - A w M * a y i) • x' i + A w M • b y
```

After simplifying the right-hand side:

```
∑ i ∈ (Finset.univ.filter (fun i : Fin m.succ => i.val < m)).attach, A w i.val • x' ⟨i.val.val, _⟩ +
A w M • (b y - ∑ i : Fin m, a y i • x' i) =
∑ i : Fin m, (a w i • x' i) - A w M • ∑ i : Fin m, (a y i • x' i) + A w M • b y
```

We distribute • over `-` on the left-hand side:

```
∑ i ∈ (Finset.univ.filter (fun i : Fin m.succ => i.val < m)).attach, A w i.val • x' ⟨i.val.val, _⟩ +
A w M • b y - A w M • (∑ i : Fin m, a y i • x' i) =
∑ i : Fin m, (a w i • x' i) - A w M • ∑ i : Fin m, (a y i • x' i) + A w M • b y
```

Exploiting `finishing_piece` again, it is easy to finish the proof.