

Linear Inequalities*

Ralph Bottesch¹, Alban Reynaud², and René Thiemann¹

¹University of Innsbruck

²ENS Lyon

March 8, 2026

Abstract

We formalize results about linear inequalities, mainly from Schrijver's book [3]. The main results are the proof of the fundamental theorem on linear inequalities, Farkas' lemma, Carathéodory's theorem, the Farkas-Minkowsky-Weyl theorem, the decomposition theorem of polyhedra, and Meyer's result that the integer hull of a polyhedron is a polyhedron itself. Several theorems include bounds on the appearing numbers, and in particular we provide an a-priori bound on mixed-integer solutions of linear inequalities.

Contents

1	Introduction	2
2	Missing Lemmas on Vectors and Matrices	3
3	Missing Lemmas on Vector Spaces	13
4	Basis Extension	17
5	Sum of Vector Sets	20
6	Integral and Bounded Matrices and Vectors	21
7	Cones	33
8	Convex Hulls	46
9	Normal Vectors	60

*Supported by FWF (Austrian Science Fund) project Y757.

10 Dimension of Spans	70
11 The Fundamental Theorem of Linear Inequalities	73
12 Farkas' Lemma	90
13 The Theorem of Farkas, Minkowsky and Weyl	94
14 The Decomposition Theorem	102
15 Mixed Integer Solutions	119
16 Integer Hull	129

1 Introduction

The motivation for this formalization is the aim of developing a verified theory solver for linear integer arithmetic. Such a solver can be a combination of a simplex-implementation within a branch-and-bound approach, that might also utilize Gomory cuts [1, Section 4 of the extended version]. However, the branch-and-bound algorithm does not terminate in general, since the search space is infinite. To solve this latter problem, one can use results of Papadimitriou: he showed that whenever a set of linear inequalities has an integer solution, then it also has a small solution, where the bound on such a solution can be computed easily from the input [2].

In this entry, we therefore formalize several results on linear inequalities which are required to obtain the desired bound, by following the proofs of Schrijver's textbook [3, Sections 7 and 16].

We start with basic definitions and results on cones, convex hulls, and polyhedra. Next, we verify the fundamental theorem of linear inequalities, which in our formalization shows the equivalence of four statements to describe a cone. From this theorem, one easily derives Farkas' Lemma and Carathéodory's theorem. Moreover we verify the Farkas-Minkowsky-Weyl theorem, that a convex cone is polyhedral if and only if it is finitely generated, and use this result to obtain the decomposition theorem for polyhedra, i.e., that a polyhedron can always be decomposed into a polytope and a finitely generated cone. For most of the previously mentioned results, we include bounds, so that in particular we have a quantitative version of the decomposition theorem, which provides bounds on the vectors that construct the polytope and the cone, and where these bounds are computed directly from the input polyhedron that should be decomposed.

We further prove the decomposition theorem also for the integer hull of a polyhedron, using the same bounds, which gives rise to small integer solutions for linear inequalities. We finally formalize a direct proof for the

more general case of mixed integer solutions, where we also permit both strict and non-strict linear inequalities.

Theorem 1. Consider $A_1 \in \mathbb{Z}^{m_1 \times n}$, $b_1 \in \mathbb{Z}^{m_1}$, $A_2 \in \mathbb{Z}^{m_2 \times n}$, $b_2 \in \mathbb{Z}^{m_2}$. Let β be a bound on A_1, b_1, A_2, b_2 , i.e., $\beta \geq |z|$ for all numbers z that occur within A_1, b_1, A_2, b_2 . Let $n = n_1 + n_2$. Then if $x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} \subseteq \mathbb{R}^n$ is a mixed integer solution of the linear inequalities, i.e., $A_1 x \leq b_1$ and $A_2 x < b_2$, then there also exists a mixed integer solution $y \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$ where $|y_i| \leq (n+1) \cdot \sqrt{n^n} \cdot \beta^n$ for each entry y_i of y .

The verified bound in Theorem 1 in particular implies that integer-satisfiability of linear-inequalities with integer coefficients is in NP.

2 Missing Lemmas on Vectors and Matrices

We provide some results on vector spaces which should be merged into Jordan-Normal-Form/Matrix.

theory *Missing-Matrix*

imports *Jordan-Normal-Form.Matrix*

begin

lemma *orthogonalD'*: **assumes** *orthogonal vs*

and $v \in \text{set } vs$ **and** $w \in \text{set } vs$

shows $(v \cdot w = 0) = (v \neq w)$

proof –

from *assms(2)* **obtain** i **where** $v: v = vs ! i$ **and** $i: i < \text{length } vs$ **by** (*auto simp: set-conv-nth*)

from *assms(3)* **obtain** j **where** $w: w = vs ! j$ **and** $j: j < \text{length } vs$ **by** (*auto simp: set-conv-nth*)

from *orthogonalD[OF assms(1) i j, folded v w]* *orthogonalD[OF assms(1) i i, folded v v]*

show *?thesis using v w by auto*

qed

lemma *zero-mat-mult-vector[simp]*: $x \in \text{carrier-vec } nc \implies 0_m \text{ nr } nc *_v x = 0_v$
nr

by (*intro eq-vecI, auto*)

lemma *add-diff-cancel-right-vec*:

$a \in \text{carrier-vec } n \implies (b :: 'a :: \text{cancel-ab-semigroup-add vec}) \in \text{carrier-vec } n \implies (a + b) - b = a$

by (*intro eq-vecI, auto*)

lemma *elements-four-block-mat-id*:

assumes $c: A \in \text{carrier-mat } nr1 \ nc1$ $B \in \text{carrier-mat } nr1 \ nc2$

$C \in \text{carrier-mat } nr2 \ nc1$ $D \in \text{carrier-mat } nr2 \ nc2$

shows

elements-mat (four-block-mat A B C D) =

```

elements-mat A ∪ elements-mat B ∪ elements-mat C ∪ elements-mat D
(is elements-mat ?four = ?X)
proof
show elements-mat ?four ⊆ ?X
  by (rule elements-four-block-mat[OF c])
have 4: ?four ∈ carrier-mat (nr1 + nr2) (nc1 + nc2) using c by auto
{
  fix x
  assume x ∈ ?X
  then consider (A) x ∈ elements-mat A
    | (B) x ∈ elements-mat B
    | (C) x ∈ elements-mat C
    | (D) x ∈ elements-mat D by auto
  hence x ∈ elements-mat ?four
  proof (cases)
    case A
    from elements-matD[OF this] obtain i j
      where *: i < nr1 j < nc1 and x: x = A $$ (i,j)
      using c by auto
    from elements-matI[OF 4, of i j x] * c
    show ?thesis unfolding x by auto
  next
    case B
    from elements-matD[OF this] obtain i j
      where *: i < nr1 j < nc2 and x: x = B $$ (i,j)
      using c by auto
    from elements-matI[OF 4, of i nc1 + j x] * c
    show ?thesis unfolding x by auto
  next
    case C
    from elements-matD[OF this] obtain i j
      where *: i < nr2 j < nc1 and x: x = C $$ (i,j)
      using c by auto
    from elements-matI[OF 4, of nr1 + i j x] * c
    show ?thesis unfolding x by auto
  next
    case D
    from elements-matD[OF this] obtain i j
      where *: i < nr2 j < nc2 and x: x = D $$ (i,j)
      using c by auto
    from elements-matI[OF 4, of nr1 + i nc1 + j x] * c
    show ?thesis unfolding x by auto
  qed
}
thus elements-mat ?four ⊇ ?X by blast
qed

```

lemma elements-mat-append-rows: $A \in \text{carrier-mat } nr \ n \implies B \in \text{carrier-mat } nr2$

$n \implies$

$elements\text{-}mat (A @_r B) = elements\text{-}mat A \cup elements\text{-}mat B$

unfolding $append\text{-}rows\text{-}def$

by ($subst\ elements\text{-}four\text{-}block\text{-}mat\text{-}id, auto$)

lemma $elements\text{-}mat\text{-}uminus[simp]$: $elements\text{-}mat (-A) = uminus \text{ ` } elements\text{-}mat A$

unfolding $elements\text{-}mat\text{-}def$ **by** $auto$

lemma $vec\text{-}set\text{-}uminus[simp]$: $vec\text{-}set (-A) = uminus \text{ ` } vec\text{-}set A$

unfolding $vec\text{-}set\text{-}def$ **by** $auto$

definition $append\text{-}cols :: 'a :: zero\ mat \implies 'a\ mat \implies 'a\ mat$ (**infixr** $\langle @_c \rangle$ 65)
where

$A @_c B = (A^T @_r B^T)^T$

lemma $carrier\text{-}append\text{-}cols[simp, intro]$:

$A \in carrier\text{-}mat\ nr\ nc1 \implies$

$B \in carrier\text{-}mat\ nr\ nc2 \implies (A @_c B) \in carrier\text{-}mat\ nr\ (nc1 + nc2)$

unfolding $append\text{-}cols\text{-}def$ **by** $auto$

lemma $elements\text{-}mat\text{-}transpose\text{-}mat[simp]$: $elements\text{-}mat (A^T) = elements\text{-}mat A$

unfolding $elements\text{-}mat\text{-}def$ **by** $auto$

lemma $elements\text{-}mat\text{-}append\text{-}cols$: $A \in carrier\text{-}mat\ n\ nc \implies B \in carrier\text{-}mat\ n\ nc1$

$\implies elements\text{-}mat (A @_c B) = elements\text{-}mat A \cup elements\text{-}mat B$

unfolding $append\text{-}cols\text{-}def\ elements\text{-}mat\text{-}transpose\text{-}mat$

by ($subst\ elements\text{-}mat\text{-}append\text{-}rows, auto$)

lemma $vec\text{-}first\text{-}index$:

assumes $v: dim\text{-}vec\ v \geq n$

and $i: i < n$

shows $(vec\text{-}first\ v\ n) \$ i = v \$ i$

unfolding $vec\text{-}first\text{-}def$ **using** $assms$ **by** $simp$

lemma $vec\text{-}last\text{-}index$:

assumes $v: v \in carrier\text{-}vec\ (n + m)$

and $i: i < m$

shows $(vec\text{-}last\ v\ m) \$ i = v \$ (n + i)$

unfolding $vec\text{-}last\text{-}def$ **using** $assms$ **by** $simp$

lemma $vec\text{-}first\text{-}add$:

assumes $dim\text{-}vec\ x \geq n$

and $dim\text{-}vec\ y \geq n$

shows $vec\text{-}first\ (x + y)\ n = vec\text{-}first\ x\ n + vec\text{-}first\ y\ n$

unfolding $vec\text{-}first\text{-}def$ **using** $assms$ **by** $auto$

lemma $vec\text{-}first\text{-}zero[simp]$: $m \leq n \implies vec\text{-}first\ (0_v\ n)\ m = 0_v\ m$

unfolding *vec-first-def* **by** *auto*

lemma *vec-first-smult*:

$\llbracket m \leq n; x \in \text{carrier-vec } n \rrbracket \implies \text{vec-first } (c \cdot_v x) m = c \cdot_v \text{vec-first } x m$

unfolding *vec-first-def* **by** *auto*

lemma *elements-mat-mat-of-row[simp]*: $\text{elements-mat } (\text{mat-of-row } v) = \text{vec-set } v$

by (*auto simp: mat-of-row-def elements-mat-def vec-set-def*)

lemma *vec-set-append-vec[simp]*: $\text{vec-set } (v @_v w) = \text{vec-set } v \cup \text{vec-set } w$

by (*metis list-of-vec-append set-append set-list-of-vec*)

lemma *vec-set-vNil[simp]*: $\text{set}_v v\text{Nil} = \{\}$ **using** *set-list-of-vec* **by** *force*

lemma *diff-smult-distrib-vec*: $((x :: 'a::\text{ring}) - y) \cdot_v v = x \cdot_v v - y \cdot_v v$

unfolding *smult-vec-def minus-vec-def*

by (*rule eq-vecI, auto simp: left-diff-distrib*)

lemma *add-diff-eq-vec*: **fixes** $y :: 'a :: \text{group-add } \text{vec}$

shows $y \in \text{carrier-vec } n \implies x \in \text{carrier-vec } n \implies z \in \text{carrier-vec } n \implies y + (x - z) = y + x - z$

by (*intro eq-vecI, auto simp: add-diff-eq*)

definition *mat-of-col* $v = (\text{mat-of-row } v)^T$

lemma *elements-mat-mat-of-col[simp]*: $\text{elements-mat } (\text{mat-of-col } v) = \text{vec-set } v$

unfolding *mat-of-col-def* **by** *auto*

lemma *mat-of-col-dim[simp]*: $\text{dim-row } (\text{mat-of-col } v) = \text{dim-vec } v$

$\text{dim-col } (\text{mat-of-col } v) = 1$

$\text{mat-of-col } v \in \text{carrier-mat } (\text{dim-vec } v) 1$

unfolding *mat-of-col-def* **by** *auto*

lemma *col-mat-of-col[simp]*: $\text{col } (\text{mat-of-col } v) 0 = v$

unfolding *mat-of-col-def* **by** *auto*

lemma *mult-mat-of-col*: $A \in \text{carrier-mat } nr \ nc \implies v \in \text{carrier-vec } nc \implies$

$A * \text{mat-of-col } v = \text{mat-of-col } (A *_v v)$

by (*intro mat-col-eqI, auto*)

lemma *mat-mult-append-cols*: **fixes** $A :: 'a :: \text{comm-semiring-0 } \text{mat}$

assumes $A: A \in \text{carrier-mat } nr \ nc1$

and $B: B \in \text{carrier-mat } nr \ nc2$

and $v1: v1 \in \text{carrier-vec } nc1$

and $v2: v2 \in \text{carrier-vec } nc2$

shows $(A @_c B) *_v (v1 @_v v2) = A *_v v1 + B *_v v2$

proof –

have $(A @_c B) *_v (v1 @_v v2) = (A @_c B) *_v \text{col } (\text{mat-of-col } (v1 @_v v2)) 0$ **by**

auto

also have $\dots = \text{col } ((A @_c B) * \text{mat-of-col } (v1 @_v v2)) \ 0$ **by** *auto*

also have $(A @_c B) * \text{mat-of-col } (v1 @_v v2) = ((A @_c B) * \text{mat-of-col } (v1 @_v v2))^{TT}$

by *auto*

also have $((A @_c B) * \text{mat-of-col } (v1 @_v v2))^T =$
 $(\text{mat-of-row } (v1 @_v v2))^{TT} * (A^T @_r B^T)^{TT}$

unfolding *append-cols-def mat-of-col-def*

proof (*rule transpose-mult, force, unfold transpose-carrier-mat, rule mat-of-row-carrier*)

have $A^T \in \text{carrier-mat } nc1 \ nr$ **using** *A* **by** *auto*

moreover have $B^T \in \text{carrier-mat } nc2 \ nr$ **using** *B* **by** *auto*

ultimately have $A^T @_r B^T \in \text{carrier-mat } (nc1 + nc2) \ nr$ **by** *auto*

hence $\text{dim-row } (A^T @_r B^T) = nc1 + nc2$ **by** *auto*

thus $v1 @_v v2 \in \text{carrier-vec } (\text{dim-row } (A^T @_r B^T))$ **using** *v1 v2* **by** *auto*

qed

also have $\dots = (\text{mat-of-row } (v1 @_v v2)) * (A^T @_r B^T)$ **by** *auto*

also have $\dots = \text{mat-of-row } v1 * A^T + \text{mat-of-row } v2 * B^T$

using *mat-of-row-mult-append-rows[OF v1 v2] A B* **by** *auto*

also have $\dots^T = (\text{mat-of-row } v1 * A^T)^T + (\text{mat-of-row } v2 * B^T)^T$

using *transpose-add A B* **by** *auto*

also have $(\text{mat-of-row } v1 * A^T)^T = A^{TT} * ((\text{mat-of-row } v1)^T)$

using *transpose-mult A v1 transpose-carrier-mat mat-of-row-carrier(1)*

by *metis*

also have $(\text{mat-of-row } v2 * B^T)^T = B^{TT} * ((\text{mat-of-row } v2)^T)$

using *transpose-mult B v2 transpose-carrier-mat mat-of-row-carrier(1)*

by *metis*

also have $A^{TT} * ((\text{mat-of-row } v1)^T) + B^{TT} * ((\text{mat-of-row } v2)^T) =$
 $A * \text{mat-of-col } v1 + B * \text{mat-of-col } v2$

unfolding *mat-of-col-def* **by** *auto*

also have $\text{col } \dots \ 0 = \text{col } (A * \text{mat-of-col } v1) \ 0 + \text{col } (B * \text{mat-of-col } v2) \ 0$

using *assms* **by** *auto*

also have $\dots = \text{col } (\text{mat-of-col } (A *_v v1)) \ 0 + \text{col } (\text{mat-of-col } (B *_v v2)) \ 0$

using *mult-mat-of-col assms* **by** *auto*

also have $\dots = A *_v v1 + B *_v v2$ **by** *auto*

finally show *?thesis* **by** *auto*

qed

lemma *vec-first-append*:

assumes $v \in \text{carrier-vec } n$

shows $\text{vec-first } (v @_v w) \ n = v$

proof –

have $v @_v w = \text{vec-first } (v @_v w) \ n @_v \text{vec-last } (v @_v w) \ (\text{dim-vec } w)$

using *vec-first-last-append assms* **by** *simp*

thus *?thesis* **using** *append-vec-eq[OF assms]* **by** *simp*

qed

lemma *vec-le-iff-diff-le-0*: **fixes** $a :: 'a :: \text{ordered-ab-group-add vec}$

shows $(a \leq b) = (a - b \leq 0_v \ (\text{dim-vec } a))$

unfolding *less-eq-vec-def* **by** *auto*

definition *mat-row-first* $A\ n \equiv \text{mat } n\ (\text{dim-col } A)\ (\lambda\ (i, j).\ A\ \$\$ (i, j))$

definition *mat-row-last* $A\ n \equiv \text{mat } n\ (\text{dim-col } A)\ (\lambda\ (i, j).\ A\ \$\$ (\text{dim-row } A - n + i, j))$

lemma *mat-row-first-carrier*[simp]: $\text{mat-row-first } A\ n \in \text{carrier-mat } n\ (\text{dim-col } A)$
unfolding *mat-row-first-def* **by** *simp*

lemma *mat-row-first-dim*[simp]:
 $\text{dim-row } (\text{mat-row-first } A\ n) = n$
 $\text{dim-col } (\text{mat-row-first } A\ n) = \text{dim-col } A$
unfolding *mat-row-first-def* **by** *simp-all*

lemma *mat-row-last-carrier*[simp]: $\text{mat-row-last } A\ n \in \text{carrier-mat } n\ (\text{dim-col } A)$
unfolding *mat-row-last-def* **by** *simp*

lemma *mat-row-last-dim*[simp]:
 $\text{dim-row } (\text{mat-row-last } A\ n) = n$
 $\text{dim-col } (\text{mat-row-last } A\ n) = \text{dim-col } A$
unfolding *mat-row-last-def* **by** *simp-all*

lemma *mat-row-first-nth*[simp]: $i < n \implies \text{row } (\text{mat-row-first } A\ n)\ i = \text{row } A\ i$
unfolding *mat-row-first-def* *row-def* **by** *fastforce*

lemma *append-rows-nth*:
assumes $A \in \text{carrier-mat } nr1\ nc$
and $B \in \text{carrier-mat } nr2\ nc$
shows $i < nr1 \implies \text{row } (A\ @_r\ B)\ i = \text{row } A\ i$
and $[i \geq nr1; i < nr1 + nr2] \implies \text{row } (A\ @_r\ B)\ i = \text{row } B\ (i - nr1)$
unfolding *append-rows-def* **using** *row-four-block-mat* *assms* **by** *auto*

lemma *mat-of-row-last-nth*[simp]:
 $i < n \implies \text{row } (\text{mat-row-last } A\ n)\ i = \text{row } A\ (\text{dim-row } A - n + i)$
unfolding *mat-row-last-def* *row-def* **by** *auto*

lemma *mat-row-first-last-append*:
assumes $\text{dim-row } A = m + n$
shows $(\text{mat-row-first } A\ m)\ @_r\ (\text{mat-row-last } A\ n) = A$
proof (*rule* *eq-rowI*)
show $\text{dim-row } (\text{mat-row-first } A\ m\ @_r\ \text{mat-row-last } A\ n) = \text{dim-row } A$
unfolding *append-rows-def* **using** *assms* **by** *fastforce*
show $\text{dim-col } (\text{mat-row-first } A\ m\ @_r\ \text{mat-row-last } A\ n) = \text{dim-col } A$
unfolding *append-rows-def* **by** *fastforce*
fix i
assume $i: i < \text{dim-row } A$
show $\text{row } (\text{mat-row-first } A\ m\ @_r\ \text{mat-row-last } A\ n)\ i = \text{row } A\ i$
proof *cases*
assume $i: i < m$

thus *?thesis* **using** *append-rows-nth(1)*[*OF mat-row-first-carrier*[*of A m*]
mat-row-last-carrier[*of A n*] *i*] **by** *simp*

next

assume *i'*: $\neg i < m$

thus *?thesis* **using** *append-rows-nth(2)*[*OF mat-row-first-carrier*[*of A m*]
mat-row-last-carrier[*of A n*]] *i* *assms* **by** *simp*

qed

qed

definition *mat-col-first* $A\ n \equiv (\text{mat-row-first } A^T\ n)^T$

definition *mat-col-last* $A\ n \equiv (\text{mat-row-last } A^T\ n)^T$

lemma *mat-col-first-carrier*[*simp*]: *mat-col-first* $A\ n \in \text{carrier-mat } (\text{dim-row } A)\ n$
unfolding *mat-col-first-def* **by** *fastforce*

lemma *mat-col-first-dim*[*simp*]:
dim-row (*mat-col-first* $A\ n$) = *dim-row* A
dim-col (*mat-col-first* $A\ n$) = n
unfolding *mat-col-first-def* **by** *simp-all*

lemma *mat-col-last-carrier*[*simp*]: *mat-col-last* $A\ n \in \text{carrier-mat } (\text{dim-row } A)\ n$
unfolding *mat-col-last-def* **by** *fastforce*

lemma *mat-col-last-dim*[*simp*]:
dim-row (*mat-col-last* $A\ n$) = *dim-row* A
dim-col (*mat-col-last* $A\ n$) = n
unfolding *mat-col-last-def* **by** *simp-all*

lemma *mat-col-first-nth*[*simp*]:
 $\llbracket i < n; i < \text{dim-col } A \rrbracket \implies \text{col } (\text{mat-col-first } A\ n)\ i = \text{col } A\ i$
unfolding *mat-col-first-def* **by** *force*

lemma *append-cols-nth*:
assumes $A \in \text{carrier-mat } nr\ nc1$
and $B \in \text{carrier-mat } nr\ nc2$
shows $i < nc1 \implies \text{col } (A @_c B)\ i = \text{col } A\ i$
and $\llbracket i \geq nc1; i < nc1 + nc2 \rrbracket \implies \text{col } (A @_c B)\ i = \text{col } B\ (i - nc1)$
unfolding *append-cols-def* *append-rows-def* **using** *row-four-block-mat* *assms*
by *auto*

lemma *mat-of-col-last-nth*[*simp*]:
 $\llbracket i < n; i < \text{dim-col } A \rrbracket \implies \text{col } (\text{mat-col-last } A\ n)\ i = \text{col } A\ (\text{dim-col } A - n + i)$
unfolding *mat-col-last-def* **by** *auto*

lemma *mat-col-first-last-append*:
assumes $\text{dim-col } A = m + n$
shows $(\text{mat-col-first } A\ m) @_c (\text{mat-col-last } A\ n) = A$

unfolding *append-cols-def mat-col-first-def mat-col-last-def*
using *mat-row-first-last-append[of A^T] assms* **by** *simp*

lemma *mat-of-row-dim-row-1*: $(\dim\text{-row } A = 1) = (A = \text{mat-of-row } (\text{row } A \ 0))$

proof

show $\dim\text{-row } A = 1 \implies A = \text{mat-of-row } (\text{row } A \ 0)$ **by** *force*

show $A = \text{mat-of-row } (\text{row } A \ 0) \implies \dim\text{-row } A = 1$ **using** *mat-of-row-dim(1)*

by *metis*

qed

lemma *mat-of-col-dim-col-1*: $(\dim\text{-col } A = 1) = (A = \text{mat-of-col } (\text{col } A \ 0))$

proof

show $\dim\text{-col } A = 1 \implies A = \text{mat-of-col } (\text{col } A \ 0)$

unfolding *mat-of-col-def* **by** *auto*

show $A = \text{mat-of-col } (\text{col } A \ 0) \implies \dim\text{-col } A = 1$ **by** (*metis mat-of-col-dim(2)*)

qed

definition *vec-of-scal* :: $'a \Rightarrow 'a \text{ vec}$ **where** *vec-of-scal* $x \equiv \text{vec } 1 \ (\lambda \ i. \ x)$

lemma *vec-of-scal-dim[simp]*:

dim-vec $(\text{vec-of-scal } x) = 1$

vec-of-scal $x \in \text{carrier-vec } 1$

unfolding *vec-of-scal-def* **by** *auto*

lemma *index-vec-of-scal[simp]*: $(\text{vec-of-scal } x) \ \$ \ 0 = x$

unfolding *vec-of-scal-def* **by** *auto*

lemma *row-mat-of-col[simp]*: $i < \dim\text{-vec } v \implies \text{row } (\text{mat-of-col } v) \ i = \text{vec-of-scal } (v \ \$ \ i)$

unfolding *mat-of-col-def* **by** *auto*

lemma *vec-of-scal-dim-1*: $(v \in \text{carrier-vec } 1) = (v = \text{vec-of-scal } (v \ \$ \ 0))$

by (*standard, auto simp del: One-nat-def, metis vec-of-scal-dim(2)*)

lemma *mult-mat-of-row-vec-of-scal*: **fixes** $x :: 'a :: \text{comm-ring-1}$

shows $\text{mat-of-col } v \ *_v \ \text{vec-of-scal } x = x \ \cdot_v \ v$

by (*auto simp add: scalar-prod-def*)

lemma *smult-pos-vec[simp]*: **fixes** $l :: 'a :: \text{linordered-ring-strict}$

assumes $l: l > 0$

shows $(l \ \cdot_v \ v \ \leq \ 0_v \ n) = (v \ \leq \ 0_v \ n)$

proof (*cases dim-vec v = n*)

case *True*

have $i < n \implies ((l \ \cdot_v \ v) \ \$ \ i \leq 0) \longleftrightarrow v \ \$ \ i \leq 0$ **for** i **using** *True*

mult-le-cancel-left-pos[OF l, of - 0] **by** *simp*

thus *?thesis* **using** *True* **unfolding** *less-eq-vec-def* **by** *auto*

qed (*auto simp: less-eq-vec-def*)

lemma *finite-elements-mat[simp]*: *finite* $(\text{elements-mat } A)$

unfolding *elements-mat-def* **by** (*rule finite-set*)

lemma *finite-vec-set[simp]*: *finite (vec-set A)*
unfolding *vec-set-def* **by** *auto*

lemma *lesseq-vecI*: **assumes** $v \in \text{carrier-vec } n$ $w \in \text{carrier-vec } n$
 $\bigwedge i. i < n \implies v \$ i \leq w \$ i$
shows $v \leq w$
using *assms* **unfolding** *less-eq-vec-def* **by** *auto*

lemma *lesseq-vecD*: **assumes** $w \in \text{carrier-vec } n$
and $v \leq w$
and $i < n$
shows $v \$ i \leq w \$ i$
using *assms* **unfolding** *less-eq-vec-def* **by** *auto*

lemma *vec-add-mono*: **fixes** $a :: 'a :: \text{ordered-ab-semigroup-add vec}$
assumes *dim*: $\text{dim-vec } b = \text{dim-vec } d$
and *ab*: $a \leq b$
and *cd*: $c \leq d$
shows $a + c \leq b + d$
proof –
have $\bigwedge i. i < \text{dim-vec } d \implies (a + c) \$ i \leq (b + d) \$ i$
proof –
fix i
assume *id*: $i < \text{dim-vec } d$
have *ic*: $i < \text{dim-vec } c$ **using** *id cd* **unfolding** *less-eq-vec-def* **by** *auto*
have *ib*: $i < \text{dim-vec } b$ **using** *id dim* **by** *auto*
have *ia*: $i < \text{dim-vec } a$ **using** *ib ab* **unfolding** *less-eq-vec-def* **by** *auto*
have $a \$ i \leq b \$ i$ **using** *ab ia ib* **unfolding** *less-eq-vec-def* **by** *auto*
moreover **have** $c \$ i \leq d \$ i$ **using** *cd ic id* **unfolding** *less-eq-vec-def* **by** *auto*
ultimately **have** *abcdi*: $a \$ i + c \$ i \leq b \$ i + d \$ i$ **using** *add-mono* **by** *auto*
have $(a + c) \$ i = a \$ i + c \$ i$ **using** *index-add-vec(1)* *ic* **by** *auto*
also **have** $\dots \leq b \$ i + d \$ i$ **using** *abcdi* **by** *auto*
also **have** $b \$ i + d \$ i = (b + d) \$ i$ **using** *index-add-vec(1)* *id* **by** *auto*
finally **show** $(a + c) \$ i \leq (b + d) \$ i$ **by** *auto*
qed
then **show** $a + c \leq b + d$ **unfolding** *less-eq-vec-def*
using *dim index-add-vec(2)* *cd less-eq-vec-def* **by** *auto*
qed

lemma *smult-nneg-npos-vec*: **fixes** $l :: 'a :: \text{ordered-semiring-0}$
assumes *l*: $l \geq 0$
and *v*: $v \leq 0_v n$
shows $l \cdot_v v \leq 0_v n$
proof –
{
fix i

assume $i: i < n$
then have $vi: v \$ i \leq 0$ **using** v **unfolding** $less\text{-}eq\text{-}vec\text{-}def$ **by** $simp$
then have $(l \cdot_v v) \$ i = l * v \$ i$ **using** v i **unfolding** $less\text{-}eq\text{-}vec\text{-}def$ **by** $auto$
also have $l * v \$ i \leq 0$ **by** $(rule\ mult\text{-}nonneg\text{-}nonpos[OF\ l\ vi])$
finally have $(l \cdot_v v) \$ i \leq 0$ **by** $auto$
}
then show $?thesis$ **using** v **unfolding** $less\text{-}eq\text{-}vec\text{-}def$ **by** $auto$
qed

lemma $smult\text{-}vec\text{-}nonneg\text{-}eq$: **fixes** $c :: 'a :: field$
shows $c \neq 0 \implies (c \cdot_v x = c \cdot_v y) = (x = y)$
proof $-$
have $c \neq 0 \implies c \cdot_v x = c \cdot_v y \implies x = y$
by $(metis\ smult\text{-}smult\text{-}assoc[of\ 1 / c\ c]\ nonzero\text{-}divide\text{-}eq\text{-}eq\ one\text{-}smult\text{-}vec)$
thus $c \neq 0 \implies ?thesis$ **by** $auto$
qed

lemma $distinct\text{-}smult\text{-}nonneg$: **fixes** $c :: 'a :: field$
assumes $c: c \neq 0$
shows $distinct\ lC \implies distinct\ (map\ ((\cdot_v)\ c)\ lC)$
proof $(induction\ lC)$
case $(Cons\ v\ lC)$
from $Cons.prem$ s **have** $v \notin set\ lC$ **by** $fastforce$
hence $c \cdot_v v \notin set\ (map\ ((\cdot_v)\ c)\ lC)$ **using** $smult\text{-}vec\text{-}nonneg\text{-}eq[OF\ c]$ **by** $fastforce$
moreover have $map\ ((\cdot_v)\ c)\ (v \# lC) = c \cdot_v v \# map\ ((\cdot_v)\ c)\ lC$ **by** $simp$
ultimately show $?case$ **using** $Cons.IH\ Cons.prem$ s **by** $simp$
qed $auto$

lemma $exists\text{-}vec\text{-}append$: $(\exists x \in carrier\text{-}vec\ (n + m). P\ x) \longleftrightarrow (\exists x1 \in carrier\text{-}vec\ n. \exists x2 \in carrier\text{-}vec\ m. P\ (x1 @_v x2))$
proof
assume $\exists x \in carrier\text{-}vec\ (n + m). P\ x$
from $this$ **obtain** x **where** $xcarr: x \in carrier\text{-}vec\ (n+m)$ **and** $Px: P\ x$ **by** $auto$
have $x = vec\ n\ (\lambda i. x \$ i) @_v vec\ m\ (\lambda i. x \$ (n + i))$
by $(rule\ eq\text{-}vecI, insert\ xcarr, auto)$
hence $P\ x = P\ (vec\ n\ (\lambda i. x \$ i) @_v vec\ m\ (\lambda i. x \$ (n + i)))$ **by** $simp$
also have $1: \dots$ **using** $xcarr\ Px$ **calculation** **by** $blast$
finally show $\exists x1 \in carrier\text{-}vec\ n. \exists x2 \in carrier\text{-}vec\ m. P\ (x1 @_v x2)$ **using** 1
 $vec\text{-}carrier$ **by** $blast$
next
assume $(\exists x1 \in carrier\text{-}vec\ n. \exists x2 \in carrier\text{-}vec\ m. P\ (x1 @_v x2))$
from $this$ **obtain** $x1\ x2$ **where** $x1: x1 \in carrier\text{-}vec\ n$
and $x2: x2 \in carrier\text{-}vec\ m$ **and** $P12: P\ (x1 @_v x2)$ **by** $auto$
define x **where** $x = x1 @_v x2$
have $xcarr: x \in carrier\text{-}vec\ (n+m)$ **using** $x1\ x2$ **by** $(simp\ add: x\text{-}def)$
have $P\ x$ **using** $P12\ xcarr$ **using** $x\text{-}def$ **by** $blast$
then show $(\exists x \in carrier\text{-}vec\ (n + m). P\ x)$ **using** $xcarr$ **by** $auto$
qed

end

3 Missing Lemmas on Vector Spaces

We provide some results on vector spaces which should be merged into other AFP entries.

theory *Missing-VS-Connect*

imports

Jordan-Normal-Form.VS-Connect

Missing-Matrix

Polynomial-Factorization.Missing-List

begin

context *vec-space*

begin

lemma *span-diff*: **assumes** $A: A \subseteq \text{carrier-vec } n$

and $a: a \in \text{span } A$ **and** $b: b \in \text{span } A$

shows $a - b \in \text{span } A$

proof –

from A a **have** $an: a \in \text{carrier-vec } n$ **by** *auto*

from A b **have** $bn: b \in \text{carrier-vec } n$ **by** *auto*

have $a + (-1 \cdot_v b) \in \text{span } A$

by (*rule span-add1*[*OF* A a], *insert* b A , *auto*)

also have $a + (-1 \cdot_v b) = a - b$ **using** an bn **by** *auto*

finally show *?thesis* **by** *auto*

qed

lemma *finsum-scalar-prod-sum'*:

assumes $f: f \in U \rightarrow \text{carrier-vec } n$

and $w: w \in \text{carrier-vec } n$

shows $w \cdot \text{finsum } V f U = \text{sum } (\lambda u. w \cdot f u) U$

by (*subst comm-scalar-prod*[*OF* w], (*insert* f , *auto*)[1],

subst finsum-scalar-prod-sum[*OF* f w],

insert f , *intro sum.cong*[*OF refl*] *comm-scalar-prod*[*OF* $-$ w], *auto*)

lemma *lincomb-scalar-prod-left*: **assumes** $W \subseteq \text{carrier-vec } n$ $v \in \text{carrier-vec } n$

shows $\text{lincomb } a W \cdot v = (\sum_{w \in W}. a w * (w \cdot v))$

unfolding *lincomb-def*

by (*subst finsum-scalar-prod-sum*, *insert assms*, *auto intro!*: *sum.cong*)

lemma *lincomb-scalar-prod-right*: **assumes** $W \subseteq \text{carrier-vec } n$ $v \in \text{carrier-vec } n$

shows $v \cdot \text{lincomb } a W = (\sum_{w \in W}. a w * (v \cdot w))$

unfolding *lincomb-def*

by (*subst finsum-scalar-prod-sum'*, *insert assms*, *auto intro!*: *sum.cong*)

lemma *lin-indpt-empty[simp]*: *lin-indpt* {}

using *lin-dep-def* **by** *auto*

lemma *span-carrier-lin-indpt-card-n*:
assumes $W \subseteq \text{carrier-vec } n$ $\text{card } W = n$ *lin-indpt* W
shows $\text{span } W = \text{carrier-vec } n$
using *assms basis-def dim-is-n dim-li-is-basis fin-dim-li-fin* **by** *simp*

lemma *ortho-span*: **assumes** $W: W \subseteq \text{carrier-vec } n$
and $X: X \subseteq \text{carrier-vec } n$
and *ortho*: $\bigwedge w x. w \in W \implies x \in X \implies w \cdot x = 0$
and $w: w \in \text{span } W$ **and** $x: x \in X$
shows $w \cdot x = 0$
proof –
from $w \in W$ **obtain** $c \in V$ **where** *finite* V **and** $VW: V \subseteq W$ **and** $w: w = \text{lincomb } c \ V$
by (*meson in-spanE*)
show *?thesis unfolding* w
by (*subst lincomb-scalar-prod-left, insert W VW X x ortho, auto intro!: sum.neutral*)
qed

lemma *ortho-span'*: **assumes** $W: W \subseteq \text{carrier-vec } n$
and $X: X \subseteq \text{carrier-vec } n$
and *ortho*: $\bigwedge w x. w \in W \implies x \in X \implies x \cdot w = 0$
and $w: w \in \text{span } W$ **and** $x: x \in X$
shows $x \cdot w = 0$
proof –
from $w \in W$ **obtain** $c \in V$ **where** *finite* V **and** $VW: V \subseteq W$ **and** $w: w = \text{lincomb } c \ V$
by (*meson in-spanE*)
show *?thesis unfolding* w
by (*subst lincomb-scalar-prod-right, insert W VW X x ortho, auto intro!: sum.neutral*)
qed

lemma *ortho-span-span*: **assumes** $W: W \subseteq \text{carrier-vec } n$
and $X: X \subseteq \text{carrier-vec } n$
and *ortho*: $\bigwedge w x. w \in W \implies x \in X \implies w \cdot x = 0$
and $w: w \in \text{span } W$ **and** $x: x \in \text{span } X$
shows $w \cdot x = 0$
by (*rule ortho-span[OF W - ortho-span'[OF X W -] w x], insert W X ortho, auto*)

lemma *lincomb-in-span[intro]*:
assumes $X: X \subseteq \text{carrier-vec } n$
shows $\text{lincomb } a \ X \in \text{span } X$
proof(*cases finite X*)
case *False* **hence** $\text{lincomb } a \ X = 0_v \ n$ **using** X
by (*simp add: lincomb-def*)
thus *?thesis using X by force*
qed (*insert X, auto*)

lemma *generating-card-n-basis*: **assumes** $X: X \subseteq \text{carrier-vec } n$
and *span*: $\text{carrier-vec } n \subseteq \text{span } X$
and *card*: $\text{card } X = n$
shows *basis* X
proof –
have *fin*: *finite* X
proof (*cases* $n = 0$)
case *False*
with *card* **show** *finite* X **by** (*meson card.infinite*)
next
case *True*
with X **have** $X \subseteq \text{carrier-vec } 0$ **by** *auto*
also **have** $\dots = \{0_v\}$ **by** *auto*
finally **have** $X \subseteq \{0_v\}$.
from *finite-subset[OF this]* **show** *finite* X **by** *auto*
qed
from X **have** $\text{span } X \subseteq \text{carrier-vec } n$ **by** *auto*
with *span* **have** *span*: $\text{span } X = \text{carrier-vec } n$ **by** *auto*
from *dim-is-n card* **have** *card*: $\text{card } X \leq \text{dim}$ **by** *auto*
from *dim-gen-is-basis[OF fin X span card]* **show** *basis* X .
qed

lemma *lincomb-list-append*:
assumes *Ws*: $\text{set } Ws \subseteq \text{carrier-vec } n$
shows $\text{set } Vs \subseteq \text{carrier-vec } n \implies \text{lincomb-list } f (Vs @ Ws) =$
 $\text{lincomb-list } f Vs + \text{lincomb-list } (\lambda i. f (i + \text{length } Vs)) Ws$
proof (*induction* Vs *arbitrary*: f)
case *Nil* **show** *?case* **by** (*simp add: lincomb-list-carrier[OF Ws]*)
next
case (*Cons* $x Vs$)
have $\text{lincomb-list } f (x \# (Vs @ Ws)) = f 0 \cdot_v x + \text{lincomb-list } (f \circ \text{Suc}) (Vs @$
 $Ws)$
by (*rule lincomb-list-Cons*)
also **have** $\text{lincomb-list } (f \circ \text{Suc}) (Vs @ Ws) =$
 $\text{lincomb-list } (f \circ \text{Suc}) Vs + \text{lincomb-list } (\lambda i. (f \circ \text{Suc}) (i + \text{length } Vs))$
 Ws
using *Cons* **by** *auto*
also **have** $(\lambda i. (f \circ \text{Suc}) (i + \text{length } Vs)) = (\lambda i. f (i + \text{length } (x \# Vs)))$ **by**
simp
also **have** $f 0 \cdot_v x + ((\text{lincomb-list } (f \circ \text{Suc}) Vs) + \text{lincomb-list } \dots Ws) =$
 $(f 0 \cdot_v x + (\text{lincomb-list } (f \circ \text{Suc}) Vs)) + \text{lincomb-list } \dots Ws$
using *assoc-add-vec Cons.prem1 Ws lincomb-list-carrier* **by** *auto*
finally **show** *?case* **using** *lincomb-list-Cons* **by** *auto*
qed

lemma *lincomb-list-snoc[simp]*:
shows $\text{set } Vs \subseteq \text{carrier-vec } n \implies x \in \text{carrier-vec } n \implies$
 $\text{lincomb-list } f (Vs @ [x]) = \text{lincomb-list } f Vs + f (\text{length } Vs) \cdot_v x$
using *lincomb-list-append* **by** *auto*

lemma *lincomb-list-smult*:

set $Vs \subseteq \text{carrier-vec } n \implies \text{lincomb-list } (\lambda i. a * c i) Vs = a \cdot_v \text{lincomb-list } c Vs$

proof (*induction* Vs *rule*: *rev-induct*)

case (*snoc* $x Vs$)

have $x: x \in \text{carrier-vec } n$ **and** $Vs: \text{set } Vs \subseteq \text{carrier-vec } n$ **using** *snoc.prem*s **by** *auto*

have $\text{lincomb-list } (\lambda i. a * c i) (Vs @ [x]) =$

$\text{lincomb-list } (\lambda i. a * c i) Vs + (a * c (\text{length } Vs)) \cdot_v x$

using $x Vs$ **by** *auto*

also have $\text{lincomb-list } (\lambda i. a * c i) Vs = a \cdot_v \text{lincomb-list } c Vs$

by (*rule* *snoc.IH*[*OF* Vs])

also have $(a * c (\text{length } Vs)) \cdot_v x = a \cdot_v (c (\text{length } Vs) \cdot_v x)$

using *smult-smult-assoc* x **by** *auto*

also have $a \cdot_v \text{lincomb-list } c Vs + \dots = a \cdot_v (\text{lincomb-list } c Vs + c (\text{length } Vs) \cdot_v x)$

using *smult-add-distrib-vec*[*of* $- n - a$] *lincomb-list-carrier*[*OF* Vs] x **by** *simp*

also have $\text{lincomb-list } c Vs + c (\text{length } Vs) \cdot_v x = \text{lincomb-list } c (Vs @ [x])$

using $Vs x$ **by** *auto*

finally show *?case* **by** *auto*

qed *simp*

lemma *lincomb-list-index*:

assumes $i: i < n$

shows $\text{set } Xs \subseteq \text{carrier-vec } n \implies$

$\text{lincomb-list } c Xs \$ i = \text{sum } (\lambda j. c j * (Xs ! j) \$ i) \{0..<\text{length } Xs\}$

proof (*induction* Xs *rule*: *rev-induct*)

case (*snoc* $x Xs$)

hence $x: x \in \text{carrier-vec } n$ **and** $Xs: \text{set } Xs \subseteq \text{carrier-vec } n$ **by** *auto*

hence $\text{lincomb-list } c (Xs @ [x]) = \text{lincomb-list } c Xs + c (\text{length } Xs) \cdot_v x$ **by** *auto*

also have $\dots \$ i = \text{lincomb-list } c Xs \$ i + (c (\text{length } Xs) \cdot_v x) \$ i$

using *i index-add-vec*(1) x **by** *simp*

also have $(c (\text{length } Xs) \cdot_v x) \$ i = c (\text{length } Xs) * x \$ i$ **using** $i x$ **by** *simp*

also have $x \$ i = (Xs @ [x]) ! (\text{length } Xs) \$ i$ **by** *simp*

also have $\text{lincomb-list } c Xs \$ i = (\sum j = 0..<\text{length } Xs. c j * Xs ! j \$ i)$

by (*rule* *snoc.IH*[*OF* Xs])

also have $\dots = (\sum j = 0..<\text{length } Xs. c j * (Xs @ [x]) ! j \$ i)$

by (*rule* *R.finsum-restrict*, *force*, *rule restrict-ext*, *auto simp: append-Cons-nth-left*)

finally show *?case*

using *sum.atLeast0-lessThan-Suc*[*of* $\lambda j. c j * (Xs @ [x]) ! j \$ i$ *length* Xs]

by *fastforce*

qed (*simp add: i*)

end

end

4 Basis Extension

We prove that every linear independent set/list of vectors can be extended into a basis. Similarly, from every set of vectors one can extract a linear independent set of vectors that spans the same space.

theory *Basis-Extension*

imports

LLL-Basis-Reduction.Gram-Schmidt-2

begin

context *cof-vec-space*

begin

lemma *lin-indpt-list-length-le-n*: **assumes** *lin-indpt-list xs*

shows $\text{length } xs \leq n$

proof –

from *assms[unfolded lin-indpt-list-def]*

have *xs*: $\text{set } xs \subseteq \text{carrier-vec } n$ **and** *dist*: *distinct xs* **and** *lin*: *lin-indpt (set xs)*

by *auto*

from *dist* **have** $\text{card } (\text{set } xs) = \text{length } xs$ **by** (*rule distinct-card*)

moreover **have** $\text{card } (\text{set } xs) \leq n$

using *lin xs dim-is-n li-le-dim(2)* **by** *auto*

ultimately show *?thesis* **by** *auto*

qed

lemma *lin-indpt-list-length-eq-n*: **assumes** *lin-indpt-list xs*

and $\text{length } xs = n$

shows $\text{span } (\text{set } xs) = \text{carrier-vec } n \text{ basis } (\text{set } xs)$

proof –

from *assms[unfolded lin-indpt-list-def]*

have *xs*: $\text{set } xs \subseteq \text{carrier-vec } n$ **and** *dist*: *distinct xs* **and** *lin*: *lin-indpt (set xs)*

by *auto*

from *dist* **have** $\text{card } (\text{set } xs) = \text{length } xs$ **by** (*rule distinct-card*)

with *assms* **have** $\text{card } (\text{set } xs) = n$ **by** *auto*

with *lin xs* **show** $\text{span } (\text{set } xs) = \text{carrier-vec } n \text{ basis } (\text{set } xs)$ **using** *dim-is-n*

by (*metis basis-def dim-basis dim-li-is-basis fin-dim finite-basis-exists gen-ge-dim li-le-dim(1)*)**+**

qed

lemma *expand-to-basis*: **assumes** *lin: lin-indpt-list xs*

shows $\exists ys. \text{set } ys \subseteq \text{set } (\text{unit-vecs } n) \wedge \text{lin-indpt-list } (xs @ ys) \wedge \text{length } (xs @ ys) = n$

proof –

define *y* **where** $y = n - \text{length } xs$

from *lin* **have** $\text{length } xs \leq n$ **by** (*rule lin-indpt-list-length-le-n*)

hence $\text{length } xs + y = n$ **unfolding** *y-def* **by** *auto*

thus $\exists ys. \text{set } ys \subseteq \text{set } (\text{unit-vecs } n) \wedge \text{lin-indpt-list } (xs @ ys) \wedge \text{length } (xs @ ys) = n$

```

    using lin
  proof (induct y arbitrary: xs)
    case (0 xs)
    thus ?case by (intro exI[of - Nil], auto)
  next
    case (Suc y xs)
    hence length xs < n by auto
    from Suc(3)[unfolded lin-indpt-list-def]
    have xs: set xs  $\subseteq$  carrier-vec n and dist: distinct xs and lin: lin-indpt (set xs)
  by auto
    from distinct-card[OF dist] Suc(2) have card: card (set xs) < n by auto
    have span (set xs)  $\neq$  carrier-vec n using card dim-is-n xs basis-def dim-basis
  lin by auto
    with span-closed[OF xs] have span (set xs)  $\subset$  carrier-vec n by auto
    also have carrier-vec n = span (set (unit-vecs n))
      unfolding span-unit-vecs-is-carrier ..
    finally have sub: span (set xs)  $\subset$  span (set (unit-vecs n)) .
    have  $\exists$  u. u  $\in$  set (unit-vecs n)  $\wedge$  u  $\notin$  span (set xs)
      using span-subsetI[OF xs, of set (unit-vecs n)] sub by force
    then obtain u where uu: u  $\in$  set (unit-vecs n) and usxs: u  $\notin$  span (set xs)
  by auto
    then have u: u  $\in$  carrier-vec n unfolding unit-vecs-def by auto
    let ?xs = xs @ [u]
    from span-mem[OF xs, of u] usxs have usxs: u  $\notin$  set xs by auto
    with dist have dist: distinct ?xs by auto
    have lin: lin-indpt (set ?xs) using lin-dep-iff-in-span[OF xs lin u usxs] usxs by
  auto
    from lin dist u xs have lin: lin-indpt-list ?xs unfolding lin-indpt-list-def by
  auto
    from Suc(2) have length ?xs + y = n by auto
    from Suc(1)[OF this lin] obtain ys where
      set ys  $\subseteq$  set (unit-vecs n) lin-indpt-list (?xs @ ys) length (?xs @ ys) = n by
  auto
    thus ?case using uu
      by (intro exI[of - u # ys], auto)
  qed
qed

```

definition *basis-extension* xs = (SOME ys.
 set ys \subseteq set (unit-vecs n) \wedge lin-indpt-list (xs @ ys) \wedge length (xs @ ys) = n)

lemma *basis-extension*: **assumes** lin-indpt-list xs
shows set (basis-extension xs) \subseteq set (unit-vecs n)
 lin-indpt-list (xs @ basis-extension xs)
 length (xs @ basis-extension xs) = n
using someI-ex[OF expand-to-basis[OF assms], folded basis-extension-def] **by**
 auto

lemma *exists-lin-indpt-sublist*: **assumes** X: X \subseteq carrier-vec n

```

shows  $\exists Ls. \text{lin-indpt-list } Ls \wedge \text{span } (\text{set } Ls) = \text{span } X \wedge \text{set } Ls \subseteq X$ 
proof –
  let  $?T = ?thesis$ 
  have  $(\exists Ls. \text{lin-indpt-list } Ls \wedge \text{span } (\text{set } Ls) \subseteq \text{span } X \wedge \text{set } Ls \subseteq X \wedge \text{length } Ls = k) \vee ?T$  for  $k$ 
  proof (induct  $k$ )
    case  $0$ 
    have  $\text{lin-indpt } \{\}$  by (simp add: lindep-span)
    thus  $?case$  using span-is-monotone by (auto simp: lin-indpt-list-def)
  next
    case (Suc  $k$ )
    show  $?case$ 
    proof (cases  $?T$ )
      case False
      with Suc obtain  $Ls$  where  $\text{lin}: \text{lin-indpt-list } Ls$ 
      and  $\text{span}: \text{span } (\text{set } Ls) \subseteq \text{span } X$  and  $Ls: \text{set } Ls \subseteq X$  and  $\text{len}: \text{length } Ls = k$  by auto
      from  $Ls$   $X$  have  $LsC: \text{set } Ls \subseteq \text{carrier-vec } n$  by auto
      show  $?thesis$ 
      proof (cases  $X \subseteq \text{span } (\text{set } Ls)$ )
        case True
        hence  $\text{span } X \subseteq \text{span } (\text{set } Ls)$  using  $LsC$   $X$  by (metis span-subsetI)
        with  $\text{span}$  have  $\text{span } (\text{set } Ls) = \text{span } X$  by auto
        hence  $?T$  by (intro exI[of -  $Ls$ ] conjI True lin Ls)
        thus  $?thesis$  by auto
      next
        case False
        with  $\text{span}$  obtain  $x$  where  $xX: x \in X$  and  $xSLs: x \notin \text{span } (\text{set } Ls)$  by auto
        from  $Ls$   $X$  have  $LsC: \text{set } Ls \subseteq \text{carrier-vec } n$  by auto
        from  $\text{span-mem}[OF \text{ this, of } x]$   $xSLs$  have  $xLs: x \notin \text{set } Ls$  by auto
        let  $?Ls = x \# Ls$ 
        show  $?thesis$ 
        proof (intro disjI1 exI[of -  $?Ls$ ] conjI)
          show  $\text{length } ?Ls = \text{Suc } k$  using  $\text{len}$  by auto
          show  $\text{lin-indpt-list } ?Ls$  using  $\text{lin } xSLs$   $xLs$  unfolding lin-indpt-list-def
            using lin-dep-iff-in-span[OF  $LsC$  - -  $xLs$ ] xX X by auto
          show  $\text{set } ?Ls \subseteq X$  using  $xX$   $Ls$  by auto
          from  $\text{span-is-monotone}[OF \text{ this}]$ 
          show  $\text{span } (\text{set } ?Ls) \subseteq \text{span } X$  .
        qed
      qed
    qed auto
  qed
from  $\text{this}[of \text{ } n + 1]$  lin-indpt-list-length-le-n show  $?thesis$  by fastforce
qed

```

```

lemma exists-lin-indpt-subset: assumes  $X \subseteq \text{carrier-vec } n$ 
shows  $\exists Ls. \text{lin-indpt } Ls \wedge \text{span } (Ls) = \text{span } X \wedge Ls \subseteq X$ 

```

```

proof –
  from exists-lin-indpt-sublist[OF assms]
  obtain Ls where lin-indpt-list Ls  $\wedge$  span (set Ls) = span X  $\wedge$  set Ls  $\subseteq$  X by
auto
  thus ?thesis by (intro exI[of - set Ls], auto simp: lin-indpt-list-def)
qed
end

end

```

5 Sum of Vector Sets

We use Isabelle's Set-Algebra theory to be able to write $V + W$ for sets of vectors V and W , and prove some obvious properties about them.

```

theory Sum-Vec-Set
  imports
    Missing-Matrix
    HOL-Library.Set-Algebras
begin

```

```

lemma add-0-right-vecset:
  assumes (A :: 'a :: monoid-add vec set')  $\subseteq$  carrier-vec n
  shows A + {0v n} = A
  unfolding set-plus-def using assms by force

```

```

lemma add-0-left-vecset:
  assumes (A :: 'a :: monoid-add vec set')  $\subseteq$  carrier-vec n
  shows {0v n} + A = A
  unfolding set-plus-def using assms by force

```

```

lemma assoc-add-vecset:
  assumes (A :: 'a :: semigroup-add vec set')  $\subseteq$  carrier-vec n
  and B  $\subseteq$  carrier-vec n
  and C  $\subseteq$  carrier-vec n
  shows A + (B + C) = (A + B) + C

```

```

proof –
  {
    fix x
    assume x  $\in$  A + (B + C)
    then obtain a b c where x = a + (b + c) and *: a  $\in$  A b  $\in$  B c  $\in$  C
    unfolding set-plus-def by auto
    with assms have x = (a + b) + c using assoc-add-vec[of a n b c] by force
    with * have x  $\in$  (A + B) + C by auto
  }
moreover
  {
    fix x

```

```

assume  $x \in (A + B) + C$ 
then obtain  $a\ b\ c$  where  $x = (a + b) + c$  and  $*$ :  $a \in A\ b \in B\ c \in C$ 
  unfolding set-plus-def by auto
  with assms have  $x = a + (b + c)$  using assoc-add-vec[of a n b c] by force
  with  $*$  have  $x \in A + (B + C)$  by auto
}
ultimately show ?thesis by blast
qed

```

```

lemma sum-carrier-vec[intro]:  $A \subseteq \text{carrier-vec } n \implies B \subseteq \text{carrier-vec } n \implies A + B \subseteq \text{carrier-vec } n$ 
  unfolding set-plus-def by force

```

```

lemma comm-add-vecset:
  assumes  $(A :: 'a :: \text{ab-semigroup-add vec set}) \subseteq \text{carrier-vec } n$ 
  and  $B \subseteq \text{carrier-vec } n$ 
  shows  $A + B = B + A$ 
  unfolding set-plus-def using comm-add-vec assms by blast

```

end

6 Integral and Bounded Matrices and Vectors

We define notions of integral vectors and matrices and bounded vectors and matrices and prove some preservation lemmas. Moreover, we prove two bounds on determinants.

```

theory Integral-Bounded-Vectors
  imports
    Missing-VS-Connect
    Sum-Vec-Set
    LLL-Basis-Reduction.Gram-Schmidt-2
begin

```

```

lemma sq-norm-unit-vec[simp]: assumes  $i: i < n$ 
  shows  $\| \text{unit-vec } n\ i \|^2 = (1 :: 'a :: \{ \text{comm-ring-1, conjugatable-ring} \})$ 
proof -
  from  $i$  have  $\text{id}: [0..<n] = [0..<i] @ [i] @ [\text{Suc } i ..<n]$ 
  by (metis append-Cons append-Nil diff-zero length-upt list-trisect)
  show ?thesis unfolding sq-norm-vec-def unit-vec-def
  by (auto simp: o-def id, subst (1 2) sum-list-0, auto)
qed

```

```

definition Ints-vec ( $\langle \mathbb{Z}_v \rangle$ ) where
   $\mathbb{Z}_v = \{x. \forall i < \text{dim-vec } x. x\ \$\ i \in \mathbb{Z}\}$ 

```

```

definition indexed-Ints-vec where

```

$indexed\text{-Ints-vec } I = \{x. \forall i < dim\text{-vec } x. i \in I \longrightarrow x \$ i \in \mathbb{Z}\}$

lemma $indexed\text{-Ints-vec-UNIV}$: $\mathbb{Z}_v = indexed\text{-Ints-vec } UNIV$
unfolding $Ints\text{-vec-def } indexed\text{-Ints-vec-def}$ **by** $auto$

lemma $indexed\text{-Ints-vec-subset}$: $\mathbb{Z}_v \subseteq indexed\text{-Ints-vec } I$
unfolding $Ints\text{-vec-def } indexed\text{-Ints-vec-def}$ **by** $auto$

lemma $Ints\text{-vec-vec-set}$: $v \in \mathbb{Z}_v = (vec\text{-set } v \subseteq \mathbb{Z})$
unfolding $Ints\text{-vec-def } vec\text{-set-def}$ **by** $auto$

definition $Ints\text{-mat } (\langle \mathbb{Z}_m \rangle)$ **where**
 $\mathbb{Z}_m = \{A. \forall i < dim\text{-row } A. \forall j < dim\text{-col } A. A \$\$ (i,j) \in \mathbb{Z}\}$

lemma $Ints\text{-mat-elements-mat}$: $A \in \mathbb{Z}_m = (elements\text{-mat } A \subseteq \mathbb{Z})$
unfolding $Ints\text{-mat-def } elements\text{-mat-def}$ **by** $force$

lemma $minus\text{-in-Ints-vec-iff}[simp]$: $(-x) \in \mathbb{Z}_v \longleftrightarrow (x :: 'a :: ring\text{-1 } vec) \in \mathbb{Z}_v$
unfolding $Ints\text{-vec-vec-set}$ **by** $(auto simp: minus\text{-in-Ints-iff})$

lemma $minus\text{-in-Ints-mat-iff}[simp]$: $(-A) \in \mathbb{Z}_m \longleftrightarrow (A :: 'a :: ring\text{-1 } mat) \in \mathbb{Z}_m$
unfolding $Ints\text{-mat-elements-mat}$ **by** $(auto simp: minus\text{-in-Ints-iff})$

lemma $Ints\text{-vec-rows-Ints-mat}[simp]$: $set (rows A) \subseteq \mathbb{Z}_v \longleftrightarrow A \in \mathbb{Z}_m$
unfolding $rows\text{-def } Ints\text{-vec-def } Ints\text{-mat-def}$ **by** $force$

lemma $unit\text{-vec-integral}[simp,intro]$: $unit\text{-vec } n \ i \in \mathbb{Z}_v$
unfolding $Ints\text{-vec-def}$ **by** $(auto simp: unit\text{-vec-def})$

lemma $diff\text{-indexed-Ints-vec}$:
 $x \in carrier\text{-vec } n \Longrightarrow y \in carrier\text{-vec } n \Longrightarrow x \in indexed\text{-Ints-vec } I \Longrightarrow y \in indexed\text{-Ints-vec } I$
 $\Longrightarrow x - y \in indexed\text{-Ints-vec } I$
unfolding $indexed\text{-Ints-vec-def}$ **by** $auto$

lemma $smult\text{-indexed-Ints-vec}$: $x \in \mathbb{Z} \Longrightarrow v \in indexed\text{-Ints-vec } I \Longrightarrow x \cdot_v v \in indexed\text{-Ints-vec } I$
unfolding $indexed\text{-Ints-vec-def } smult\text{-vec-def}$ **by** $simp$

lemma $add\text{-indexed-Ints-vec}$:
 $x \in carrier\text{-vec } n \Longrightarrow y \in carrier\text{-vec } n \Longrightarrow x \in indexed\text{-Ints-vec } I \Longrightarrow y \in indexed\text{-Ints-vec } I$
 $\Longrightarrow x + y \in indexed\text{-Ints-vec } I$
unfolding $indexed\text{-Ints-vec-def}$ **by** $auto$

lemma **(in** $vec\text{-space}$) $lincomb\text{-indexed-Ints-vec}$: **assumes** $cI: \bigwedge x. x \in C \Longrightarrow c x \in \mathbb{Z}$
and $C: C \subseteq carrier\text{-vec } n$
and $CI: C \subseteq indexed\text{-Ints-vec } I$

shows $\text{lincomb } c \ C \in \text{indexed-Ints-vec } I$
proof –
from C **have** $\text{id}: \text{dim-vec } (\text{lincomb } c \ C) = n$ **by** *auto*
show *?thesis unfolding indexed-Ints-vec-def mem-Collect-eq id*
proof (*intro allI impI*)
fix i
assume $i: i < n$ **and** $iI: i \in I$
have $\text{lincomb } c \ C \ \$ \ i = (\sum x \in C. c \ x * x \ \$ \ i)$
by (*rule lincomb-index[OF i C]*)
also have $\dots \in \mathbb{Z}$
by (*intro Ints-sum Ints-mult cI, insert i iI CI[unfolded indexed-Ints-vec-def]*
 $C, \text{force+}$)
finally show $\text{lincomb } c \ C \ \$ \ i \in \mathbb{Z}$.
qed
qed

definition $\text{Bounded-vec } (b :: 'a :: \text{linordered-idom}) = \{x . \forall i < \text{dim-vec } x . \text{abs } (x \ \$ \ i) \leq b\}$

lemma $\text{Bounded-vec-vec-set}: v \in \text{Bounded-vec } b \longleftrightarrow (\forall x \in \text{vec-set } v. \text{abs } x \leq b)$
unfolding $\text{Bounded-vec-def vec-set-def}$ **by** *auto*

definition $\text{Bounded-mat } (b :: 'a :: \text{linordered-idom}) = \{A . (\forall i < \text{dim-row } A . \forall j < \text{dim-col } A . \text{abs } (A \ \$ \ \$ \ (i,j)) \leq b)\}$

lemma $\text{Bounded-mat-elements-mat}: A \in \text{Bounded-mat } b \longleftrightarrow (\forall x \in \text{elements-mat } A. \text{abs } x \leq b)$
unfolding $\text{Bounded-mat-def elements-mat-def}$ **by** *auto*

lemma $\text{Bounded-vec-rows-Bounded-mat[simp]}: \text{set } (\text{rows } A) \subseteq \text{Bounded-vec } B \longleftrightarrow A \in \text{Bounded-mat } B$
unfolding $\text{rows-def Bounded-vec-def Bounded-mat-def}$ **by** *force*

lemma $\text{unit-vec-Bounded-vec[simp,intro]}: \text{unit-vec } n \ i \in \text{Bounded-vec } (\text{max } 1 \ \text{Bnd})$
unfolding $\text{Bounded-vec-def unit-vec-def}$ **by** *auto*

lemma $\text{unit-vec-int-bounds}: \text{set } (\text{unit-vecs } n) \subseteq \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } (\text{max } 1 \ \text{Bnd}))$
unfolding unit-vecs-def **by** (*auto simp: Bounded-vec-def*)

lemma $\text{Bounded-matD}: \text{assumes } A \in \text{Bounded-mat } b$
 $A \in \text{carrier-mat } nr \ nc$
shows $i < nr \implies j < nc \implies \text{abs } (A \ \$ \ \$ \ (i,j)) \leq b$
using *assms* **unfolding** Bounded-mat-def **by** *auto*

lemma $\text{Bounded-vec-mono}: b \leq B \implies \text{Bounded-vec } b \subseteq \text{Bounded-vec } B$
unfolding Bounded-vec-def **by** *auto*

lemma $\text{Bounded-mat-mono}: b \leq B \implies \text{Bounded-mat } b \subseteq \text{Bounded-mat } B$

unfolding *Bounded-mat-def* **by** *force*

lemma *finite-Bounded-vec-Max*:

assumes $A: A \subseteq \text{carrier-vec } n$

and $\text{fin}: \text{finite } A$

shows $A \subseteq \text{Bounded-vec } (\text{Max } \{ \text{abs } (a \ \$ \ i) \mid a \ i. a \in A \wedge i < n \})$

proof

let $?B = \{ \text{abs } (a \ \$ \ i) \mid a \ i. a \in A \wedge i < n \}$

have $\text{fin}: \text{finite } ?B$

by (*rule finite-subset[of - ($\lambda (a,i). \text{abs } (a \ \$ \ i)$) ' ($A \times \{0 \ ..< n\}$)], *insert fin, auto*)*

fix a

assume $a: a \in A$

show $a \in \text{Bounded-vec } (\text{Max } ?B)$

unfolding *Bounded-vec-def*

by (*standard, intro allI impI Max-ge[OF fin], insert a A, force*)

qed

definition *is-det-bound* $:: (\text{nat} \Rightarrow 'a :: \text{linordered-idom} \Rightarrow 'a) \Rightarrow \text{bool}$ **where**

$\text{is-det-bound } f = (\forall A \ n \ x. A \in \text{carrier-mat } n \ n \longrightarrow A \in \text{Bounded-mat } x \longrightarrow \text{abs } (\text{det } A) \leq f \ n \ x)$

lemma *is-det-bound-ge-zero*: **assumes** *is-det-bound* f

and $x \geq 0$

shows $f \ n \ x \geq 0$

using *assms(1)[unfolding is-det-bound-def, rule-format, of $0_m \ n \ n \ n \ x$]*

using *assms(2)* **unfolding** *Bounded-mat-def* **by** *auto*

definition *det-bound-fact* $:: \text{nat} \Rightarrow 'a :: \text{linordered-idom} \Rightarrow 'a$ **where**

$\text{det-bound-fact } n \ x = \text{fact } n * (x \hat{=} n)$

lemma *det-bound-fact: is-det-bound det-bound-fact*

unfolding *is-det-bound-def*

proof (*intro allI impI*)

fix $A :: 'a :: \text{linordered-idom mat}$ **and** $n \ x$

assume $A: A \in \text{carrier-mat } n \ n$

and $x: A \in \text{Bounded-mat } x$

show $\text{abs } (\text{det } A) \leq \text{det-bound-fact } n \ x$

proof –

have $\text{abs } (\text{det } A) = \text{abs } (\sum p \mid p \text{ permutes } \{0..<n\}. \text{signof } p * (\prod i = 0..<n. A \ \$\$ (i, p \ i)))$

unfolding *det-def'[OF A]* ..

also have $\dots \leq (\sum p \mid p \text{ permutes } \{0..<n\}. \text{abs } (\text{signof } p * (\prod i = 0..<n. A \ \$\$ (i, p \ i))))$

by (*rule sum-abs*)

also have $\dots = (\sum p \mid p \text{ permutes } \{0..<n\}. (\prod i = 0..<n. \text{abs } (A \ \$\$ (i, p \ i))))$

by (*rule sum.cong[OF refl], auto simp: abs-mult abs-prod sign-def simp flip: of-int-abs*)

also have $\dots \leq (\sum p \mid p \text{ permutes } \{0..<n\}). (\prod i = 0..<n. x)$
by (*intro sum-mono prod-mono conjI Bounded-matD[OF x A], auto*)
also have $\dots = \text{fact } n * x^{\wedge} n$ **by** (*auto simp add: card-permutations*)
finally show $\text{abs } (\det A) \leq \text{det-bound-fact } n \ x$ **unfolding** *det-bound-fact-def*
by *auto*
qed
qed

lemma (*in gram-schmidt-fs*) *Gramian-determinant-det*: **assumes** *A*: $A \in \text{carrier-mat } n \ n$
shows *Gramian-determinant* (*rows A*) $n = \det A * \det A$
proof –
have [*simp*]: *mat-of-rows* n (*rows A*) = *A* **using** *A*
by (*intro eq-matI, auto*)
show ?*thesis* **using** *A*
unfolding *Gramian-determinant-def*
by (*subst Gramian-matrix-alt-def, force, simp add: Let-def, subst det-mult[of -*
n],
auto simp: det-transpose)
qed

lemma (*in gram-schmidt-fs-lin-indpt*) *det-bound-main*: **assumes** *rows*: *rows A* =
fs
and *A*: $A \in \text{carrier-mat } n \ n$
and *n0*: $n > 0$
and *Bnd*: $A \in \text{Bounded-mat } c$
shows
 $(\text{abs } (\det A))^{\wedge} 2 \leq \text{of-nat } n \ ^{\wedge} n * c \ ^{\wedge} (2 * n)$
proof –
from *n0 A Bnd* **have** $\text{abs } (A \ \$\$ (0,0)) \leq c$ **by** (*auto simp: Bounded-mat-def*)
hence *c0*: $c \geq 0$ **by** *auto*
from *n0 A rows* **have** *fs*: $\text{set } fs \neq \{\}$ **by** (*auto simp: rows-def*)
from *rows A* **have** *len*: $\text{length } fs = n$ **by** *auto*
have $(\text{abs } (\det A))^{\wedge} 2 = \det A * \det A$ **unfolding** *power2-eq-square* **by** *simp*
also have $\dots = d \ n$ **using** *Gramian-determinant-det[OF A]* **unfolding** *rows* **by**
simp
also have $\dots = (\prod j < n. \|gso \ j\|^2)$
by (*rule Gramian-determinant(1), auto simp: len*)
also have $\dots \leq (\prod j < n. N)$
by (*rule prod-mono, insert N-gso, auto simp: len*)
also have $\dots = N^{\wedge} n$ **by** *simp*
also have $\dots \leq (\text{of-nat } n * c^{\wedge} 2)^{\wedge} n$
proof (*rule power-mono*)
show $0 \leq N$ **using** *N-ge-0 len n0* **by** *auto*
show $N \leq \text{of-nat } n * c^{\wedge} 2$ **unfolding** *N-def*
proof (*intro Max.boundedI, force, use fs in force, clarify*)
fix *f*
assume $f \in \text{set } fs$

```

from this[folded rows] obtain i where i: i < n and f: f = row A i
  using A unfolding rows-def by auto
have  $\|f\|^2 = (\sum x \leftarrow \text{list-of-vec } (\text{row } A \ i). \ x^{\wedge}2)$ 
  unfolding sq-norm-vec-def power2-eq-square by simp
also have  $\text{list-of-vec } (\text{row } A \ i) = \text{map } (\lambda j. \ A \ \$\$ \ (i, j)) \ [0..<n]$ 
  using i A by (intro nth-equalityI, auto)
also have  $\text{sum-list } (\text{map } \text{power2 } (\text{map } (\lambda j. \ A \ \$\$ \ (i, j)) \ [0..<n])) \leq$ 
   $\text{sum-list } (\text{map } (\lambda j. \ c^{\wedge}2) \ [0..<n])$  unfolding map-map o-def
proof (intro sum-list-mono)
  fix j
  assume j ∈ set  $[0 ..< n]$ 
  hence j: j < n by auto
  from Bnd i j A have  $|A \ \$\$ \ (i, j)| \leq c$  by (auto simp: Bounded-mat-def)
  thus  $(A \ \$\$ \ (i, j))^2 \leq c^2$ 
    by (meson abs-ge-zero order-trans power2-le-iff-abs-le)
qed
also have  $\dots = (\sum j < n. \ c^2)$ 
  unfolding interv-sum-list-conv-sum-set-nat by auto
also have  $\dots = \text{of-nat } n * c^2$  by auto
finally show  $\|f\|^2 \leq \text{of-nat } n * c^2$  .
qed
qed
also have  $\dots = (\text{of-nat } n)^{\wedge}n * (c^2)^{\wedge}n$  by (auto simp: algebra-simps)
also have  $\dots = \text{of-nat } n^{\wedge}n * c^{\wedge}(2 * n)$  unfolding power-mult[symmetric]
  by (simp add: ac-simps)
finally show ?thesis .
qed

```

lemma *det-bound-hadamard-squared*: **fixes** *A::'a :: trivial-conjugatable-linordered-field*
mat

```

assumes A: A ∈ carrier-mat n n
  and Bnd: A ∈ Bounded-mat c
shows  $(\text{abs } (\text{det } A))^{\wedge}2 \leq \text{of-nat } n^{\wedge}n * c^{\wedge}(2 * n)$ 
proof (cases n > 0)
  case n: True
    from n A Bnd have  $\text{abs } (A \ \$\$ \ (0,0)) \leq c$  by (auto simp: Bounded-mat-def)
    hence c0: c ≥ 0 by auto
    let ?us = map (row A)  $[0 ..< n]$ 
    interpret gso: gram-schmidt-fs n ?us .
    have len: length ?us = n by simp
    have us: set ?us ⊆ carrier-vec n using A by auto
    let ?vs = map gso.gso  $[0..<n]$ 
    show ?thesis
    proof (cases carrier-vec n ⊆ gso.span (set ?us))
      case False
        from mat-of-rows-rows[unfolded rows-def, of A] A gram-schmidt.non-span-det-zero[OF  

len False us]
        have zero: det A = 0 by auto

```

```

  show ?thesis unfolding zero using c0 by simp
next
case True
with us len have basis: gso.basis-list ?us unfolding gso.basis-list-def by auto
note in-dep = gso.basis-list-imp-lin-indpt-list[OF basis]
interpret gso: gram-schmidt-fs-lin-indpt n ?us
  by (standard) (use in-dep gso.lin-indpt-list-def in auto)
from gso.det-bound-main[OF - A n Bnd]
show ?thesis using A by (auto simp: rows-def)
qed
next
case False
with A show ?thesis by auto
qed

```

definition *det-bound-hadamard* :: $\text{nat} \Rightarrow \text{int} \Rightarrow \text{int}$ **where**
det-bound-hadamard $n\ c = (\text{sqrt-int-floor } ((\text{int } n * c^2)^{\wedge} n))$

lemma *det-bound-hadamard-altdef*[code]:
det-bound-hadamard $n\ c = (\text{if } n = 1 \vee \text{even } n \text{ then } \text{int } n^{\wedge} (n \text{ div } 2) * (\text{abs } c)^{\wedge} n$
 else $\text{sqrt-int-floor } ((\text{int } n * c^2)^{\wedge} n))$

```

proof (cases  $n = 1 \vee \text{even } n$ )
  case False
  thus ?thesis unfolding det-bound-hadamard-def by auto
next
case True
define thesis where  $\text{thesis} = ?thesis$ 
have  $\text{thesis} \iff \text{sqrt-int-floor } ((\text{int } n * c^2)^{\wedge} n) = \text{int } n^{\wedge} (n \text{ div } 2) * \text{abs } c^{\wedge} n$ 
  using True unfolding thesis-def det-bound-hadamard-def by auto
also have  $(\text{int } n * c^2)^{\wedge} n = \text{int } n^{\wedge} n * c^{\wedge} (2 * n)$ 
  unfolding power-mult[symmetric] power-mult-distrib by (simp add: ac-simps)
also have  $\text{int } n^{\wedge} n = \text{int } n^{\wedge} (2 * (n \text{ div } 2))$  using True by auto
also have  $\dots * c^{\wedge} (2 * n) = (\text{int } n^{\wedge} (n \text{ div } 2) * c^{\wedge} n)^2$ 
  unfolding power-mult-distrib power-mult[symmetric] by (simp add: ac-simps)
also have  $\text{sqrt-int-floor } \dots = \text{int } n^{\wedge} (n \text{ div } 2) * |c|^{\wedge} n$ 
  unfolding sqrt-int-floor of-int-power real-sqrt-abs of-int-abs[symmetric] floor-of-int  

abs-mult power-abs by simp
finally have thesis by auto
thus ?thesis unfolding thesis-def by auto
qed

```

lemma *det-bound-hadamard: is-det-bound det-bound-hadamard*
unfolding *is-det-bound-def*

```

proof (intro allI impI)
  fix  $A :: \text{int mat}$  and  $n\ c$ 
  assume  $A: A \in \text{carrier-mat } n\ n$  and  $\text{Bnd}A: A \in \text{Bounded-mat } c$ 
  let ?h = rat-of-int
  let ?hA = map-mat ?h A
  let ?hc = ?h  $c$ 

```

```

from  $A$  have  $hA$ :  $?hA \in \text{carrier-mat } n \ n$  by auto
from  $BndA$  have  $Bnd$ :  $?hA \in \text{Bounded-mat } ?hc$ 
  unfolding Bounded-mat-def
  by (auto, unfold of-int-abs[symmetric] of-int-le-iff, auto)
have  $\text{sqrt}$ :  $\text{sqrt} ((\text{real } n * (\text{real-of-int } c)^2) \wedge n) \geq 0$ 
  by simp
from det-bound-hadamard-squared[OF hA Bnd, unfolded of-int-hom.hom-det of-int-abs[symmetric]]
have  $?h (|\det A| \wedge 2) \leq ?h (int \ n \wedge n * c \wedge (2 * n))$  by simp
from this[unfolded of-int-le-iff]
have  $|\det A| \wedge 2 \leq int \ n \wedge n * c \wedge (2 * n)$  .
also have  $\dots = (int \ n * c \wedge 2) \wedge n$  unfolding power-mult power-mult-distrib by
simp
  finally have  $|\det A|^2 \leq (int \ n * c^2) \wedge n$  by simp
  hence  $\text{sqrt-int-floor} (|\det A|^2) \leq \text{sqrt-int-floor} ((int \ n * c^2) \wedge n)$ 
    unfolding sqrt-int-floor by (intro floor-mono real-sqrt-le-mono, linarith)
  also have  $\text{sqrt-int-floor} (|\det A|^2) = |\det A|$  by (simp del: of-int-abs add: of-int-abs[symmetric])
  finally show  $|\det A| \leq \text{det-bound-hadamard } n \ c$  unfolding det-bound-hadamard-def
by simp
qed

lemma n-pow-n-le-fact-square:  $n \wedge n \leq (\text{fact } n) \wedge 2$ 
proof –
  define  $ii$  where  $ii (i :: nat) = (n + 1 - i)$  for  $i$ 
  have  $id$ :  $ii \text{ ' } \{1..n\} = \{1..n\}$  unfolding ii-def
  proof (auto, goal-cases)
    case (1  $i$ )
    hence  $i$ :  $i = (-) (\text{Suc } n) (ii \ i)$  unfolding ii-def by auto
    show  $?case$  by (subst i, rule imageI, insert 1, auto simp: ii-def)
  qed
  have  $(\text{fact } n) = (\prod \{1..n\})$ 
    by (simp add: fact-prod)
  hence  $(\text{fact } n) \wedge 2 = ((\prod \{1..n\}) * (\prod \{1..n\}))$  by (auto simp: power2-eq-square)
  also have  $\dots = ((\prod \{1..n\}) * \text{prod } (\lambda \ i. \ i) (ii \ \{1..n\}))$ 
    by (rule arg-cong[of - - \lambda x. (- * x), rule prod.cong[OF id[symmetric]], auto)
  also have  $\dots = ((\prod \{1..n\}) * \text{prod } ii \ \{1..n\})$ 
    by (subst prod.reindex, auto simp: ii-def inj-on-def)
  also have  $\dots = (\text{prod } (\lambda \ i. \ i * ii \ i) \ \{1..n\})$ 
    by (subst prod.distrib, auto)
  also have  $\dots \geq (\text{prod } (\lambda \ i. \ n) \ \{1..n\})$ 
  proof (intro prod-mono conjI, simp)
    fix  $i$ 
    assume  $i$ :  $i \in \{1 .. n\}$ 
    let  $?j = ii \ i$ 
    show  $n \leq i * ?j$ 
    proof (cases i = 1 \vee i = n)
      case True
      thus  $?thesis$  unfolding ii-def by auto
    next
    case False

```

hence *min*: $\min i \ ?j \geq 2$ **using** *i* **by** (*auto simp: ii-def*)
 have *max*: $n \leq 2 * \max i \ ?j$ **using** *i* **by** (*auto simp: ii-def*)
 also have $\dots \leq \min i \ ?j * \max i \ ?j$ **using** *min*
 by (*intro mult-mono, auto*)
 also have $\dots = i * \ ?j$ **by** (*cases i < ?j, auto simp: ac-simps*)
 finally show *?thesis* .
 qed
 qed
 finally show *?thesis* **by** *simp*
 qed

lemma *sqrt-int-floor-bound*: $0 \leq x \implies (\text{sqrt-int-floor } x)^2 \leq x$
unfolding *sqrt-int-floor-def*
using *root-int-floor-def root-int-floor-pos-lower* **by** *auto*

lemma *det-bound-hadamard-improves-det-bound-fact*: **assumes** *c*: $c \geq 0$
shows *det-bound-hadamard n c* \leq *det-bound-fact n c*
proof –
 have $(\text{det-bound-hadamard } n \ c)^2 \leq (\text{int } n * c^2)^n$ **unfolding** *det-bound-hadamard-def*
 by (*rule sqrt-int-floor-bound, auto*)
 also have $\dots = \text{int } (n^n) * c^{(2 * n)}$ **by** (*simp add: power-mult power-mult-distrib*)
 also have $\dots \leq \text{int } ((\text{fact } n)^2) * c^{(2 * n)}$
 by (*intro mult-right-mono, unfold of-nat-le-iff, rule n-pow-n-le-fact-square, auto*)
 also have $\dots = (\text{det-bound-fact } n \ c)^2$ **unfolding** *det-bound-fact-def*
 by (*simp add: power-mult-distrib power-mult[symmetric] ac-simps*)
 finally have $\text{abs } (\text{det-bound-hadamard } n \ c) \leq \text{abs } (\text{det-bound-fact } n \ c)$
 unfolding *abs-le-square-iff* .
 hence *det-bound-hadamard n c* \leq *abs (det-bound-fact n c)* **by** *simp*
 also have $\dots = \text{det-bound-fact } n \ c$ **unfolding** *det-bound-fact-def* **using** *c* **by**
auto
 finally show *?thesis* .
 qed

context
begin
private fun *syl* :: *int* \Rightarrow *nat* \Rightarrow *int mat* **where**
 syl c 0 = *mat 1 1* (λ -. *c*)
 | *syl c (Suc n)* = (*let A = syl c n in*
 four-block-mat A A (-A) A)

private lemma *syl*: **assumes** *c*: $c \geq 0$
shows *syl c n* \in *Bounded-mat c* \wedge *syl c n* \in *carrier-mat* (2^n) (2^n)
 \wedge *det (syl c n)* = *det-bound-hadamard* (2^n) *c*
proof (*cases n = 0*)
case *True*
thus *?thesis* **using** *c*
 unfolding *det-bound-hadamard-altdef*
 by (*auto simp: Bounded-mat-def det-single*)
next

```

case False
then obtain m where n: n = Suc m by (cases n, auto)
show ?thesis unfolding n
proof (induct m)
  case 0
    show ?case unfolding syl.simps Let-def using c
      apply (subst det-four-block-mat[of - 1]; force?)
      apply (subst det-single,
        auto simp: Bounded-mat-def scalar-prod-def det-bound-hadamard-altdef
power2-eq-square)
      done
    next
      case (Suc m)
        define A where A = syl c (Suc m)
        let ?FB = four-block-mat A A (- A) A
        define n :: nat where n =  $2 \wedge \text{Suc } m$ 
        from Suc[folded A-def n-def]
        have Bnd: A  $\in$  Bounded-mat c
          and A: A  $\in$  carrier-mat n n
          and detA: det A = det-bound-hadamard n c
          by auto
        have n2:  $2 \wedge \text{Suc } (\text{Suc } m) = 2 * n$  unfolding n-def by auto
        show ?case unfolding syl.simps(2)[of - Suc m] A-def[symmetric] Let-def n2
proof (intro conjI)
      show ?FB  $\in$  carrier-mat (2 * n) (2 * n) using A by auto
      show ?FB  $\in$  Bounded-mat c using Bnd A unfolding Bounded-mat-elements-mat
        by (subst elements-four-block-mat-id, auto)
      have ev: even n and sum:  $n \text{ div } 2 + n \text{ div } 2 = n$  unfolding n-def by auto
      have n2:  $n * 2 = n + n$  by simp
      have det ?FB = det (A * A - A * - A)
        by (rule det-four-block-mat[OF A A - A], insert A, auto)
      also have  $A * A - A * - A = A * A + A * A$  using A by auto
      also have  $\dots = 2 \cdot_m (A * A)$  using A by auto
      also have  $\text{det } \dots = 2 \wedge n * \text{det } (A * A)$ 
        by (subst det-smult, insert A, auto)
      also have  $\text{det } (A * A) = \text{det } A * \text{det } A$  by (rule det-mult[OF A A])
      also have  $2 \wedge n * \dots = \text{det-bound-hadamard } (2 * n) c$  unfolding detA
unfolding det-bound-hadamard-altdef by (simp add: ev ac-simps power-add[symmetric]
sum n2)
      finally show  $\text{det } ?FB = \text{det-bound-hadamard } (2 * n) c .$ 
    qed
  qed
qed

lemma det-bound-hadamard-tight:
  assumes c:  $c \geq 0$ 
  and n =  $2 \wedge m$ 
  shows  $\exists A. A \in \text{carrier-mat } n n \wedge A \in \text{Bounded-mat } c \wedge \text{det } A = \text{det-bound-hadamard}$ 
n c

```

by (rule exI[of - syl c m], insert syl[OF c, of m, folded assms(2)], auto)
end

lemma *Ints-matE*: **assumes** $A \in \mathbb{Z}_m$

shows $\exists B. A = \text{map-mat of-int } B$

proof –

have $\forall ij. \exists x. \text{fst } ij < \text{dim-row } A \longrightarrow \text{snd } ij < \text{dim-col } A \longrightarrow A \ \$\$ \ ij = \text{of-int } x$

using *assms unfolding Ints-mat-def Ints-def* **by** *auto*

from *choice[OF this]* **obtain** f **where**

$f: \forall i j. i < \text{dim-row } A \longrightarrow j < \text{dim-col } A \longrightarrow A \ \$\$ \ (i,j) = \text{of-int } (f \ (i,j))$

by *auto*

show *?thesis*

by (*intro exI[of - mat (dim-row A) (dim-col A) f] eq-matI, insert f, auto*)

qed

lemma *is-det-bound-of-int*: **fixes** $A :: 'a :: \text{linordered-idom mat}$

assumes $db: \text{is-det-bound } db$

and $A: A \in \text{carrier-mat } n \ n$

and $A \in \mathbb{Z}_m \cap \text{Bounded-mat (of-int bnd)}$

shows $abs \ (det \ A) \leq \text{of-int } (db \ n \ bnd)$

proof –

from *assms* **have** $A \in \mathbb{Z}_m$ **by** *auto*

from *Ints-matE[OF this]* **obtain** B **where**

$AB: A = \text{map-mat of-int } B$ **by** *auto*

from *assms* **have** $A \in \text{Bounded-mat (of-int bnd)}$ **by** *auto*

hence $B \in \text{Bounded-mat bnd}$ **unfolding** AB *Bounded-mat-elements-mat*

by (*auto simp flip: of-int-abs*)

from $db[\text{unfolded is-det-bound-def, rule-format, OF - this, of } n]$ $AB \ A$

have $|det \ B| \leq db \ n \ bnd$ **by** *auto*

thus *?thesis* **unfolding** AB *of-int-hom.hom-det*

by (*simp flip: of-int-abs*)

qed

lemma *minus-in-Bounded-vec[simp]*:

$(-x) \in \text{Bounded-vec } b \longleftrightarrow x \in \text{Bounded-vec } b$

unfolding *Bounded-vec-def* **by** *auto*

lemma *sum-in-Bounded-vecI[intro]*: **assumes**

$xB: x \in \text{Bounded-vec } B1$ **and**

$yB: y \in \text{Bounded-vec } B2$ **and**

$x: x \in \text{carrier-vec } n$ **and**

$y: y \in \text{carrier-vec } n$

shows $x + y \in \text{Bounded-vec } (B1 + B2)$

proof –

from $x \ y$ **have** $id: \text{dim-vec } (x + y) = n$ **by** *auto*

show *?thesis* **unfolding** *Bounded-vec-def mem-Collect-eq id*

```

proof (intro allI impI)
  fix i
  assume i: i < n
  with x y xB yB have *: abs (x $ i) ≤ B1 abs (y $ i) ≤ B2
    unfolding Bounded-vec-def by auto
  thus |(x + y) $ i| ≤ B1 + B2 using i x y by simp
qed
qed

```

lemma (in gram-schmidt) lincomb-card-bound: **assumes** XBnd: $X \subseteq \text{Bounded-vec } Bnd$

```

and X:  $X \subseteq \text{carrier-vec } n$ 
and Bnd:  $Bnd \geq 0$ 
and c:  $\bigwedge x. x \in X \implies \text{abs } (c x) \leq 1$ 
and card:  $\text{card } X \leq k$ 
shows lincomb c X ∈ Bounded-vec (of-nat k * Bnd)
proof –
from X have dim:  $\text{dim-vec } (\text{lincomb } c X) = n$  by auto
show ?thesis unfolding Bounded-vec-def mem-Collect-eq dim
proof (intro allI impI)

```

```

  fix i
  assume i: i < n
  have abs (lincomb c X $ i) = abs ( $\sum x \in X. c x * x $ i$ )
    by (subst lincomb-index[OF i X], auto)
  also have ... ≤ ( $\sum x \in X. \text{abs } (c x * x $ i)$ ) by auto
  also have ... = ( $\sum x \in X. \text{abs } (c x) * \text{abs } (x $ i)$ ) by (auto simp: abs-mult)
  also have ... ≤ ( $\sum x \in X. 1 * \text{abs } (x $ i)$ )
    by (rule sum-mono[OF mult-right-mono], insert c, auto)
  also have ... = ( $\sum x \in X. \text{abs } (x $ i)$ ) by simp
  also have ... ≤ ( $\sum x \in X. Bnd$ )
    by (rule sum-mono, insert i XBnd[unfolded Bounded-vec-def] X, force)
  also have ... = of-nat (card X) * Bnd by simp
  also have ... ≤ of-nat k * Bnd
    by (rule mult-right-mono[OF - Bnd], insert card, auto)
  finally show abs (lincomb c X $ i) ≤ of-nat k * Bnd by auto
qed
qed

```

lemma bounded-vecset-sum:

```

assumes Acarr:  $A \subseteq \text{carrier-vec } n$ 
and Bcarr:  $B \subseteq \text{carrier-vec } n$ 
and sum:  $C = A + B$ 
and Cbnd:  $\exists \text{ bndC}. C \subseteq \text{Bounded-vec } \text{bndC}$ 
shows  $A \neq \{\}$   $\implies (\exists \text{ bndB}. B \subseteq \text{Bounded-vec } \text{bndB})$ 
and  $B \neq \{\}$   $\implies (\exists \text{ bndA}. A \subseteq \text{Bounded-vec } \text{bndA})$ 
proof –
  {
    fix A B :: 'a vec set
    assume Acarr:  $A \subseteq \text{carrier-vec } n$ 

```

```

assume  $Bcarr: B \subseteq \text{carrier-vec } n$ 
assume  $sum: C = A + B$ 
assume  $Ane: A \neq \{\}$ 
have  $\exists \text{ bnd}B. B \subseteq \text{Bounded-vec bnd}B$ 
proof(cases  $B = \{\}$ )
  case  $Bne: \text{False}$ 
  from  $Cbnd$  obtain  $bndC$  where  $bndC: C \subseteq \text{Bounded-vec bnd}C$  by auto
  from  $Ane$  obtain  $a$  where  $aA: a \in A$  and  $acarr: a \in \text{carrier-vec } n$  using
Acarr by auto
  let  $?M = \{\text{abs } (a \ \$ \ i) \mid i. i < n\}$ 
  have  $finM: \text{finite } ?M$  by simp
  define  $nb$  where  $nb = \text{abs } bndC + \text{Max } ?M$ 
  {
    fix  $b$ 
    assume  $bB: b \in B$  and  $bcarr: b \in \text{carrier-vec } n$ 
    have  $ab: a + b \in \text{Bounded-vec bnd}C$  using  $aA \ bB \ bndC \ \text{sum}$  by auto
    {
      fix  $i$ 
      assume  $i\text{-lt-}n: i < n$ 
      hence  $ai\text{-le-max}: \text{abs}(a \ \$ \ i) \leq \text{Max } ?M$  using  $acarr \ finM \ \text{Max-ge}$  by blast
      hence  $\text{abs}(a \ \$ \ i + b \ \$ \ i) \leq \text{abs } bndC$ 
      using  $ab \ bcarr \ acarr \ \text{index-add-vec}(1) \ i\text{-lt-}n$  unfolding Bounded-vec-def
by auto
      hence  $\text{abs}(b \ \$ \ i) \leq \text{abs } bndC + \text{abs}(a \ \$ \ i)$  by simp
      hence  $\text{abs}(b \ \$ \ i) \leq nb$  using  $i\text{-lt-}n \ bcarr \ ai\text{-le-max}$  unfolding  $nb\text{-def}$  by
simp
    }
    hence  $b \in \text{Bounded-vec } nb$  unfolding Bounded-vec-def using  $bcarr \ \text{carrier-vec}D$  by blast
  }
  hence  $B \subseteq \text{Bounded-vec } nb$  unfolding Bounded-vec-def using  $Bcarr$  by auto
  thus  $?thesis$  by auto
  qed auto
} note  $theor = \text{this}$ 
show  $A \neq \{\} \implies (\exists \text{ bnd}B. B \subseteq \text{Bounded-vec bnd}B)$  using  $theor[OF \ Acarr \ Bcarr \ \text{sum}]$  by simp
have  $CBA: C = B + A$  unfolding  $sum$  by (rule comm-add-vecset[OF Acarr Bcarr])
show  $B \neq \{\} \implies \exists \text{ bnd}A. A \subseteq \text{Bounded-vec bnd}A$  using  $theor[OF \ Bcarr \ Acarr \ CBA]$  by simp
qed

```

end

7 Cones

We define the notions like cone, polyhedral cone, etc. and prove some basic facts about them.

```

theory Cone
  imports
    Basis-Extension
    Missing-VS-Connect
    Integral-Bounded-Vectors
begin

context gram-schmidt
begin

definition nonneg-lincomb  $c\ Vs\ b = (\text{lincomb } c\ Vs = b \wedge c \text{ ' } Vs \subseteq \{x. x \geq 0\})$ 
definition nonneg-lincomb-list  $c\ Vs\ b = (\text{lincomb-list } c\ Vs = b \wedge (\forall i < \text{length } Vs. c\ i \geq 0))$ 

definition finite-cone :: 'a vec set  $\Rightarrow$  'a vec set where
  finite-cone  $Vs = (\{ b. \exists c. \text{nonneg-lincomb } c\ (\text{if finite } Vs \text{ then } Vs \text{ else } \{\})\} b)$ 

definition cone :: 'a vec set  $\Rightarrow$  'a vec set where
  cone  $Vs = (\{ x. \exists Ws. \text{finite } Ws \wedge Ws \subseteq Vs \wedge x \in \text{finite-cone } Ws\})$ 

definition cone-list :: 'a vec list  $\Rightarrow$  'a vec set where
  cone-list  $Vs = \{b. \exists c. \text{nonneg-lincomb-list } c\ Vs\ b\}$ 

lemma finite-cone-iff-cone-list: assumes  $Vs: Vs \subseteq \text{carrier-vec } n$ 
  and  $id: Vs = \text{set } Vsl$ 
shows finite-cone  $Vs = \text{cone-list } Vsl$ 
proof –
  have  $fin: \text{finite } Vs$  unfolding  $id$  by auto
  from  $Vs\ id$  have  $Vsl: \text{set } Vsl \subseteq \text{carrier-vec } n$  by auto
  {
    fix  $c\ b$ 
    assume  $b: \text{lincomb } c\ Vs = b$  and  $c: c \text{ ' } Vs \subseteq \{x. x \geq 0\}$ 
    from  $\text{lincomb-as-lincomb-list}[OF\ Vsl, \text{of } c]$ 
    have  $b: \text{lincomb-list } (\lambda i. \text{if } \exists j < i. Vsl\ !\ i = Vsl\ !\ j \text{ then } 0 \text{ else } c\ (Vsl\ !\ i))\ Vsl$ 
    =  $b$ 
    unfolding  $b[\text{symmetric}]\ id$  by simp
    have  $\exists c. \text{nonneg-lincomb-list } c\ Vsl\ b$ 
    unfolding  $\text{nonneg-lincomb-list-def}$ 
    apply  $(\text{intro } exI\ conjI, \text{rule } b)$ 
    by  $(\text{insert } c, \text{auto simp: set-conv-nth } id)$ 
  }
moreover
  {
    fix  $c\ b$ 
    assume  $b: \text{lincomb-list } c\ Vsl = b$  and  $c: (\forall i < \text{length } Vsl. c\ i \geq 0)$ 
    have  $\text{nonneg-lincomb } (\text{mk-coeff } Vsl\ c)\ Vs\ b$ 
    unfolding  $b[\text{symmetric}]\ \text{nonneg-lincomb-def}$ 
    apply  $(\text{subst } \text{lincomb-list-as-lincomb}[OF\ Vsl])$ 
    by  $(\text{insert } c, \text{auto simp: id mk-coeff-def intro!: sum-list-nonneg})$ 
  }

```

hence $\exists c. \text{nonneg-lincomb } c \text{ } Vs \text{ } b$ **by blast**
 }
 ultimately show *?thesis* **unfolding** *finite-cone-def cone-list-def*
nonneg-lincomb-def nonneg-lincomb-list-def **using** *fin* **by auto**
qed

lemma *cone-alt-def*: **assumes** $Vs: Vs \subseteq \text{carrier-vec } n$
shows $\text{cone } Vs = (\{ x. \exists Ws. \text{set } Ws \subseteq Vs \wedge x \in \text{cone-list } Ws \})$
unfolding *cone-def*
proof (*intro Collect-cong iffI*)
fix x
assume $\exists Ws. \text{finite } Ws \wedge Ws \subseteq Vs \wedge x \in \text{finite-cone } Ws$
then obtain Ws **where** $*$: $\text{finite } Ws \text{ } Ws \subseteq Vs \text{ } x \in \text{finite-cone } Ws$ **by auto**
from *finite-list[OF *(1)]* **obtain** Wsl **where** $\text{id}: Ws = \text{set } Wsl$ **by auto**
from *finite-cone-iff-cone-list[OF - this] *(2-3)* Vs
have $x \in \text{cone-list } Wsl$ **by auto**
with $*(2)$ id **show** $\exists Wsl. \text{set } Wsl \subseteq Vs \wedge x \in \text{cone-list } Wsl$ **by blast**
next
fix x
assume $\exists Wsl. \text{set } Wsl \subseteq Vs \wedge x \in \text{cone-list } Wsl$
then obtain Wsl **where** $\text{set } Wsl \subseteq Vs \text{ } x \in \text{cone-list } Wsl$ **by auto**
thus $\exists Ws. \text{finite } Ws \wedge Ws \subseteq Vs \wedge x \in \text{finite-cone } Ws$ **using** Vs
by (*intro exI[of - set Wsl], subst finite-cone-iff-cone-list, auto*)
qed

lemma *cone-mono*: $Vs \subseteq Ws \implies \text{cone } Vs \subseteq \text{cone } Ws$
unfolding *cone-def* **by blast**

lemma *finite-cone-mono*: **assumes** $\text{fin}: \text{finite } Ws$
and $Ws: Ws \subseteq \text{carrier-vec } n$
and $\text{sub}: Vs \subseteq Ws$
shows $\text{finite-cone } Vs \subseteq \text{finite-cone } Ws$
proof
fix b
assume $b \in \text{finite-cone } Vs$
then obtain c **where** $b: b = \text{lincomb } c \text{ } Vs$ **and** $c: c \text{ ' } Vs \subseteq \{x. x \geq 0\}$
unfolding *finite-cone-def nonneg-lincomb-def* **using** *finite-subset[OF sub fin]*
by auto
define d **where** $d = (\lambda v. \text{if } v \in Vs \text{ then } c \text{ } v \text{ else } 0)$
from c **have** $d: d \text{ ' } Ws \subseteq \{x. x \geq 0\}$ **unfolding** *d-def* **by auto**
have $\text{lincomb } d \text{ } Ws = \text{lincomb } d \text{ } (Ws - Vs) + \text{lincomb } d \text{ } Vs$
by (*rule lincomb-vec-diff-add[OF Ws sub fin], auto*)
also have $\text{lincomb } d \text{ } Vs = \text{lincomb } c \text{ } Vs$
by (*rule lincomb-cong, insert Ws sub, auto simp: d-def*)
also have $\text{lincomb } d \text{ } (Ws - Vs) = 0_v \text{ } n$
by (*rule lincomb-zero, insert Ws sub, auto simp: d-def*)
also have $0_v \text{ } n + \text{lincomb } c \text{ } Vs = \text{lincomb } c \text{ } Vs$ **using** $Ws \text{ } sub$ **by auto**
also have $\dots = b$ **unfolding** b **by simp**
finally

have $b = \text{lincomb } d \text{ } Ws$ **by auto**
then show $b \in \text{finite-cone } Ws$ **using** $d \text{ } fin$
unfolding $\text{finite-cone-def nonneg-lincomb-def}$ **by auto**
qed

lemma $\text{finite-cone-carrier}: A \subseteq \text{carrier-vec } n \implies \text{finite-cone } A \subseteq \text{carrier-vec } n$
unfolding $\text{finite-cone-def nonneg-lincomb-def}$ **by auto**

lemma $\text{cone-carrier}: A \subseteq \text{carrier-vec } n \implies \text{cone } A \subseteq \text{carrier-vec } n$
using $\text{finite-cone-carrier}$ **unfolding** cone-def **by blast**

lemma $\text{cone-iff-finite-cone}: \text{assumes } A: A \subseteq \text{carrier-vec } n$
and $fin: \text{finite } A$

shows $\text{cone } A = \text{finite-cone } A$

proof

show $\text{finite-cone } A \subseteq \text{cone } A$ **unfolding** cone-def **using** fin **by auto**

show $\text{cone } A \subseteq \text{finite-cone } A$ **unfolding** cone-def **using** $fin \text{ } \text{finite-cone-mono}[OF \text{ } fin \text{ } A]$ **by auto**

qed

lemma $\text{set-in-finite-cone}:$

assumes $Vs: Vs \subseteq \text{carrier-vec } n$

and $fin: \text{finite } Vs$

shows $Vs \subseteq \text{finite-cone } Vs$

proof

fix x

assume $x: x \in Vs$

show $x \in \text{finite-cone } Vs$ **unfolding** finite-cone-def

proof

let $?c = \lambda y. \text{if } x = y \text{ then } 1 \text{ else } 0 :: 'a$

have $Vsx: Vs - \{x\} \subseteq \text{carrier-vec } n$ **using** Vs **by auto**

have $\text{lincomb } ?c \text{ } Vs = x + \text{lincomb } ?c \text{ } (Vs - \{x\})$

using $\text{lincomb-del2 } x \text{ } Vs \text{ } fin$ **by auto**

also have $\text{lincomb } ?c \text{ } (Vs - \{x\}) = 0_v \text{ } n$ **using** $\text{lincomb-zero } Vsx$ **by auto**

also have $x + 0_v \text{ } n = x$ **using** $M.r\text{-zero } Vs \text{ } x$ **by auto**

finally have $\text{lincomb } ?c \text{ } Vs = x$ **by auto**

moreover have $?c \text{ } ' Vs \subseteq \{z. z \geq 0\}$ **by auto**

ultimately show $\exists c. \text{nonneg-lincomb } c \text{ } (\text{if } \text{finite } Vs \text{ then } Vs \text{ else } \{\}) x$

unfolding $\text{nonneg-lincomb-def}$

using fin **by auto**

qed

qed

lemma $\text{set-in-cone}:$

assumes $Vs: Vs \subseteq \text{carrier-vec } n$

shows $Vs \subseteq \text{cone } Vs$

proof

fix x

assume $x: x \in Vs$

show $x \in \text{cone } Vs$ **unfolding** *cone-def*
proof (*intro CollectI exI*)
 have $x \in \text{carrier-vec } n$ **using** Vs x **by** *auto*
 then have $x \in \text{finite-cone } \{x\}$ **using** *set-in-finite-cone* **by** *auto*
 then show $\text{finite } \{x\} \wedge \{x\} \subseteq Vs \wedge x \in \text{finite-cone } \{x\}$ **using** x **by** *auto*
qed
qed

lemma *zero-in-finite-cone*:
 assumes $Vs: Vs \subseteq \text{carrier-vec } n$
 shows $0_v \ n \in \text{finite-cone } Vs$
proof –
 let $?Vs = (\text{if finite } Vs \text{ then } Vs \text{ else } \{\})$
 have $\text{lincomb } (\lambda x. 0 :: 'a) \ ?Vs = 0_v \ n$ **using** *lincomb-zero* Vs **by** *auto*
 moreover have $(\lambda x. 0 :: 'a) \ ' ?Vs \subseteq \{y. y \geq 0\}$ **by** *auto*
 ultimately show *?thesis* **unfolding** *finite-cone-def nonneg-lincomb-def* **by** *blast*
qed

lemma *lincomb-in-finite-cone*:
 assumes $x = \text{lincomb } l \ W$
 and *finite* W
 and $\forall i \in W. l \ i \geq 0$
 and $W \subseteq \text{carrier-vec } n$
 shows $x \in \text{finite-cone } W$
 using *cone-iff-finite-cone* *assms* **unfolding** *finite-cone-def nonneg-lincomb-def*
by *auto*

lemma *lincomb-in-cone*:
 assumes $x = \text{lincomb } l \ W$
 and *finite* W
 and $\forall i \in W. l \ i \geq 0$
 and $W \subseteq \text{carrier-vec } n$
 shows $x \in \text{cone } W$
 using *cone-iff-finite-cone* *assms* **unfolding** *finite-cone-def nonneg-lincomb-def*
by *auto*

lemma *zero-in-cone*: $0_v \ n \in \text{cone } Vs$
proof –
 have *finite* $\{\}$ **by** *auto*
 moreover have $\{\} \subseteq \text{cone } Vs$ **by** *auto*
 moreover have $0_v \ n \in \text{finite-cone } \{\}$ **using** *zero-in-finite-cone* **by** *auto*
 ultimately show *?thesis* **unfolding** *cone-def* **by** *blast*
qed

lemma *cone-smult*:
 assumes $a: a \geq 0$
 and $Vs: Vs \subseteq \text{carrier-vec } n$
 and $x: x \in \text{cone } Vs$
 shows $a \cdot_v x \in \text{cone } Vs$

proof –
from x Vs **obtain** Ws c **where** $Ws: Ws \subseteq Vs$ **and** $fin: finite$ Ws **and**
nonneg-lincomb c Ws x
unfolding *cone-def* *finite-cone-def* **by** *auto*
then have *nonneg-lincomb* $(\lambda w. a * c w)$ Ws $(a \cdot_v x)$
unfolding *nonneg-lincomb-def* **using** *a lincomb-distrib* Vs **by** *auto*
then show *?thesis* **using** Ws fin **unfolding** *cone-def* *finite-cone-def* **by** *auto*
qed

lemma *finite-cone-empty[simp]*: *finite-cone* $\{\}$ = $\{0_v\ n\}$
by (*auto simp: finite-cone-def nonneg-lincomb-def*)

lemma *cone-empty[simp]*: *cone* $\{\}$ = $\{0_v\ n\}$
unfolding *cone-def* **by** *simp*

lemma *cone-elem-sum*:
assumes $Vs: Vs \subseteq carrier-vec\ n$
and $x: x \in cone\ Vs$
and $y: y \in cone\ Vs$
shows $x + y \in cone\ Vs$

proof –
obtain Xs **where** $Xs: Xs \subseteq Vs$ **and** $fin-Xs: finite$ Xs
and $Xs-cone: x \in finite-cone$ Xs
using Vs x **unfolding** *cone-def* **by** *auto*
obtain Ys **where** $Ys: Ys \subseteq Vs$ **and** $fin-Ys: finite$ Ys
and $Ys-cone: y \in finite-cone$ Ys
using Vs y **unfolding** *cone-def*
by *auto*
have $x \in finite-cone$ $(Xs \cup Ys)$ **and** $y \in finite-cone$ $(Xs \cup Ys)$
using *finite-cone-mono* $fin-Xs$ $fin-Ys$ Xs Ys Vs $Xs-cone$ $Ys-cone$
by (*blast, blast*)
then obtain cx cy **where** *nonneg-lincomb* cx $(Xs \cup Ys)$ x
and *nonneg-lincomb* cy $(Xs \cup Ys)$ y
unfolding *finite-cone-def* **using** $fin-Xs$ $fin-Ys$ **by** *auto*
hence *nonneg-lincomb* $(\lambda v. cx\ v + cy\ v)$ $(Xs \cup Ys)$ $(x + y)$
unfolding *nonneg-lincomb-def*
using *lincomb-sum[of* $Xs \cup Ys$ cx $cy]$ $fin-Xs$ $fin-Ys$ Xs Ys Vs
by *fastforce*
hence $x + y \in finite-cone$ $(Xs \cup Ys)$
unfolding *finite-cone-def* **using** $fin-Xs$ $fin-Ys$ **by** *auto*
thus *?thesis* **unfolding** *cone-def* **using** $fin-Xs$ $fin-Ys$ Xs Ys **by** *auto*
qed

lemma *cone-cone*:
assumes $Vs: Vs \subseteq carrier-vec\ n$
shows *cone* $(cone\ Vs)$ = *cone* Vs
proof
show *cone* $Vs \subseteq cone$ $(cone\ Vs)$

by (rule set-in-cone[OF cone-carrier[OF Vs]])
 next
 show cone (cone Vs) \subseteq cone Vs
 proof
 fix x
 assume x: x \in cone (cone Vs)
 then obtain Ws c where Ws: set Ws \subseteq cone Vs
 and c: nonneg-lincomb-list c Ws x
 using cone-alt-def Vs cone-carrier unfolding cone-list-def by auto

 have set Ws \subseteq cone Vs \implies nonneg-lincomb-list c Ws x \implies x \in cone Vs
 proof (induction Ws arbitrary: x c)
 case Nil
 hence x = 0_v n unfolding nonneg-lincomb-list-def by auto
 thus x \in cone Vs using zero-in-cone by auto
 next
 case (Cons a Ws)
 have a \in cone Vs using Cons.prem1 by auto
 moreover have c 0 \geq 0
 using Cons.prem2 unfolding nonneg-lincomb-list-def by fastforce
 ultimately have c 0 \cdot_v a \in cone Vs using cone-smult Vs by auto
 moreover have lincomb-list (c \circ Suc) Ws \in cone Vs
 using Cons unfolding nonneg-lincomb-list-def by fastforce
 moreover have x = c 0 \cdot_v a + lincomb-list (c \circ Suc) Ws
 using Cons.prem2 unfolding nonneg-lincomb-list-def
 by auto
 ultimately show x \in cone Vs using cone-elem-sum Vs by auto
 qed

 thus x \in cone Vs using Ws c by auto
 qed
 qed

 lemma cone-smult-basis:
 assumes Vs: Vs \subseteq carrier-vec n
 and l: l ' Vs \subseteq {x. x > 0}
 shows cone {l v \cdot_v v | v . v \in Vs} = cone Vs
 proof
 have {l v \cdot_v v | v . v \in Vs} \subseteq cone Vs
 proof
 fix x
 assume x \in {l v \cdot_v v | v . v \in Vs}
 then obtain v where v \in Vs and x = l v \cdot_v v by auto
 thus x \in cone Vs using
 set-in-cone[OF Vs] cone-smult[OF - Vs, of l v v] l by fastforce
 qed
 thus cone {l v \cdot_v v | v . v \in Vs} \subseteq cone Vs
 using cone-mono cone-cone[OF Vs] by blast
 next

have $lVs: \{l v \cdot_v v \mid v. v \in Vs\} \subseteq \text{carrier-vec } n$ **using** Vs **by** *auto*
have $Vs \subseteq \text{cone } \{l v \cdot_v v \mid v. v \in Vs\}$
proof
fix v **assume** $v: v \in Vs$
hence $l v \cdot_v v \in \text{cone } \{l v \cdot_v v \mid v. v \in Vs\}$ **using** *set-in-cone[OF lVs]* **by** *auto*
moreover **have** $1 / l v > 0$ **using** $l v$ **by** *auto*
ultimately **have** $(1 / l v) \cdot_v (l v \cdot_v v) \in \text{cone } \{l v \cdot_v v \mid v. v \in Vs\}$
using *cone-smult[OF - lVs]* **by** *auto*
also **have** $(1 / l v) \cdot_v (l v \cdot_v v) = v$ **using** $l v$
by(*auto simp add: smult-smult-assoc*)
finally **show** $v \in \text{cone } \{l v \cdot_v v \mid v. v \in Vs\}$ **by** *auto*
qed
thus $\text{cone } Vs \subseteq \text{cone } \{l v \cdot_v v \mid v. v \in Vs\}$
using *cone-mono cone-cone[OF lVs]* **by** *blast*
qed

lemma *cone-add-cone*:

assumes $C: C \subseteq \text{carrier-vec } n$
shows $\text{cone } C + \text{cone } C = \text{cone } C$
proof
note $CC = \text{cone-carrier}[OF C]$
have $\text{cone } C = \text{cone } C + \{0_v n\}$ **by** (*subst add-0-right-vecset[OF CC], simp*)
also **have** $\dots \subseteq \text{cone } C + \text{cone } C$
by (*rule set-plus-mono2, insert zero-in-cone, auto*)
finally **show** $\text{cone } C \subseteq \text{cone } C + \text{cone } C$ **by** *auto*
from *cone-elem-sum[OF C]*
show $\text{cone } C + \text{cone } C \subseteq \text{cone } C$
by (*auto elim!: set-plus-elim*)
qed

lemma *orthogonal-cone*:

assumes $X: X \subseteq \text{carrier-vec } n$
and $W: W \subseteq \text{carrier-vec } n$
and $\text{fin}X: \text{finite } X$
and $\text{span}LW: \text{span } (\text{set } Ls \cup W) = \text{carrier-vec } n$
and $\text{ortho}: \bigwedge w x. w \in W \implies x \in \text{set } Ls \implies w \cdot x = 0$
and $WWs: W = \text{set } Ws$
and $\text{span}L: \text{span } (\text{set } Ls) = \text{span } X$
and $LX: \text{set } Ls \subseteq X$
and $\text{lin-Ls-Bs}: \text{lin-indpt-list } (Ls @ Bs)$
and $\text{len-Ls-Bs}: \text{length } (Ls @ Bs) = n$
shows $\text{cone } (X \cup \text{set } Bs) \cap \{x \in \text{carrier-vec } n. \forall w \in W. w \cdot x = 0\} = \text{cone } X$
 $\bigwedge x. \forall w \in W. w \cdot x = 0 \implies Z \subseteq X \implies B \subseteq \text{set } Bs \implies x = \text{lincomb } c (Z \cup B)$
 $\implies x = \text{lincomb } c (Z - B)$

proof –

from WWs **have** $\text{fin}W: \text{finite } W$ **by** *auto*
define Y **where** $Y = X \cup \text{set } Bs$
from $\text{lin-Ls-Bs}[\text{unfolded lin-indpt-list-def}]$ **have**

```

Ls: set  $Ls \subseteq \text{carrier-vec } n$  and
Bs: set  $Bs \subseteq \text{carrier-vec } n$  and
distLsBs: distinct ( $Ls @ Bs$ ) and
lin: lin-indpt (set ( $Ls @ Bs$ )) by auto
have LW: set  $Ls \cap W = \{\}$ 
proof (rule ccontr)
  assume  $\neg ?thesis$ 
  then obtain x where xX:  $x \in \text{set } Ls$  and xW:  $x \in W$  by auto
  from ortho[OF xW xX] have  $x \cdot x = 0$  by auto
  hence sq-norm  $x = 0$  by (auto simp: sq-norm-vec-as-cscalar-prod)
  with vs-zero-lin-dep[OF - lin] xX Ls Bs show False by auto
qed
have Y:  $Y \subseteq \text{carrier-vec } n$  using X Bs unfolding Y-def by auto
have CLB:  $\text{carrier-vec } n = \text{span } (\text{set } (Ls @ Bs))$ 
  using lin-Ls-Bs len-Ls-Bs lin-indpt-list-length-eq-n by blast
also have  $\dots \subseteq \text{span } Y$ 
  by (rule span-is-monotone, insert LX, auto simp: Y-def)
finally have span:  $\text{span } Y = \text{carrier-vec } n$  using Y by auto
have finY: finite Y using finX finW unfolding Y-def by auto
{
  fix x Z B d
  assume xX:  $\forall w \in W. w \cdot x = 0$  and ZX:  $Z \subseteq X$  and B:  $B \subseteq \text{set } Bs$  and
  xd:  $x = \text{lincomb } d (Z \cup B)$ 
  from ZX B X Bs have ZB:  $Z \cup B \subseteq \text{carrier-vec } n$  by auto
  with xd have x:  $x \in \text{carrier-vec } n$  by auto
  from xX W have w0:  $w \in W \implies w \cdot x = 0$  for w by auto
  from finite-in-span[OF - - x[folded spanLW]] Ls X W finW finX
  obtain c where xc:  $x = \text{lincomb } c (\text{set } Ls \cup W)$  by auto
  have  $x = \text{lincomb } c (\text{set } Ls \cup W)$  unfolding xc by auto
  also have  $\dots = \text{lincomb } c (\text{set } Ls) + \text{lincomb } c W$ 
  by (rule lincomb-union, insert X LX W LW finW, auto)
  finally have xsum:  $x = \text{lincomb } c (\text{set } Ls) + \text{lincomb } c W$  .
  {
    fix w
    assume wW:  $w \in W$ 
    with W have w:  $w \in \text{carrier-vec } n$  by auto
    from w0[OF wW, unfolded xsum]
    have  $0 = w \cdot (\text{lincomb } c (\text{set } Ls) + \text{lincomb } c W)$  by simp
    also have  $\dots = w \cdot \text{lincomb } c (\text{set } Ls) + w \cdot \text{lincomb } c W$ 
    by (rule scalar-prod-add-distrib[OF w], insert Ls W, auto)
    also have  $w \cdot \text{lincomb } c (\text{set } Ls) = 0$  using ortho[OF wW]
    by (subst lincomb-scalar-prod-right[OF Ls w], auto)
    finally have  $w \cdot \text{lincomb } c W = 0$  by simp
  }
}
hence  $\text{lincomb } c W \cdot \text{lincomb } c W = 0$  using W
  by (subst lincomb-scalar-prod-left, auto)
hence sq-norm ( $\text{lincomb } c W$ ) = 0
  by (auto simp: sq-norm-vec-as-cscalar-prod)
hence 0:  $\text{lincomb } c W = 0_v n$  using lincomb-closed[OF W, of c] by simp

```

```

have xc: x = lincomb c (set Ls) unfolding xsum 0 using Ls by auto
hence xL: x ∈ span (set Ls) by auto
let ?X = Z - B
have lincomb d ?X ∈ span X using finite-subset[OF - finX, of ?X] X ZX by
auto
from finite-in-span[OF finite-set Ls this[folded spanL]]
obtain e where ed: lincomb e (set Ls) = lincomb d ?X by auto
from B finite-subset[OF B] have finB: finite B by auto
from B Bs have BC: B ⊆ carrier-vec n by auto
define f where f =
  (λ x. if x ∈ set Bs then if x ∈ B then d x else 0 else if x ∈ set Ls then e x else
undefined)
have x = lincomb d (?X ∪ B) unfolding xd by auto
also have ... = lincomb d ?X + lincomb d B
  by (rule lincomb-union[OF - - - finite-subset[OF - finX]], insert ZX X finB B
Bs, auto)
finally have xd: x = lincomb d ?X + lincomb d B .
also have ... = lincomb e (set Ls) + lincomb d B unfolding ed by auto
also have lincomb e (set Ls) = lincomb f (set Ls)
  by (rule lincomb-cong[OF - Ls], insert distLsBs, auto simp: f-def)
also have lincomb d B = lincomb f B
  by (rule lincomb-cong[OF - BC], insert B, auto simp: f-def)
also have lincomb f B = lincomb f (B ∪ (set Bs - B))
  by (subst lincomb-clean, insert finB Bs B, auto simp: f-def)
also have B ∪ (set Bs - B) = set Bs using B by auto
finally have x = lincomb f (set Ls) + lincomb f (set Bs) by auto
also have lincomb f (set Ls) + lincomb f (set Bs) = lincomb f (set (Ls @ Bs))
  by (subst lincomb-union[symmetric], insert Ls distLsBs Bs, auto)
finally have x = lincomb f (set (Ls @ Bs)) .
hence f: f ∈ set (Ls @ Bs) →E UNIV ∧ lincomb f (set (Ls @ Bs)) = x
  by (auto simp: f-def split: if-splits)
from finite-in-span[OF finite-set Ls xL] obtain g where
  xg: x = lincomb g (set Ls) by auto
define h where h = (λ x. if x ∈ set Bs then 0 else if x ∈ set Ls then g x else
undefined)
have x = lincomb h (set Ls) unfolding xg
  by (rule lincomb-cong[OF - Ls], insert distLsBs, auto simp: h-def)
also have ... = lincomb h (set Ls) + 0v n using Ls by auto
also have 0v n = lincomb h (set Bs)
  by (rule lincomb-zero[symmetric, OF Bs], auto simp: h-def)
also have lincomb h (set Ls) + lincomb h (set Bs) = lincomb h (set (Ls @ Bs))
  by (subst lincomb-union[symmetric], insert Ls Bs distLsBs, auto)
finally have x = lincomb h (set (Ls @ Bs)) .
hence h: h ∈ set (Ls @ Bs) →E UNIV ∧ lincomb h (set (Ls @ Bs)) = x
  by (auto simp: h-def split: if-splits)
have basis: basis (set (Ls @ Bs)) using lin-Ls-Bs[unfolding lin-indpt-list-def]
len-Ls-Bs
  using CLB basis-def by blast
from Ls Bs have set (Ls @ Bs) ⊆ carrier-vec n by auto

```

```

from basis[unfolding basis-criterion[OF finite-set this], rule-format, OF x] f h
have fh: f = h by auto
hence  $\bigwedge x. x \in \text{set } Bs \implies f x = 0$  unfolding h-def by auto
hence  $\bigwedge x. x \in B \implies d x = 0$  unfolding f-def using B by force
thus  $x = \text{lincomb } d \ ?X$  unfolding xd
  by (subst (2) lincomb-zero, insert BC ZB X, auto intro!: M.r-zero)
} note main = this
have cone Y  $\cap \{x \in \text{carrier-vec } n. \forall w \in W. w \cdot x = 0\} = \text{cone } X$  (is ?I = -)
proof
{
  fix x
  assume xX:  $x \in \text{cone } X$ 
  with cone-carrier[OF X] have x:  $x \in \text{carrier-vec } n$  by auto
  have  $X \subseteq Y$  unfolding Y-def by auto
  from cone-mono[OF this] xX have xY:  $x \in \text{cone } Y$  by auto
  from cone-iff-finite-cone[OF X finX] xX have  $x \in \text{finite-cone } X$  by auto
  from this[unfolding finite-cone-def nonneg-lincomb-def] finX obtain c
    where  $x = \text{lincomb } c \ X$  by auto
  with finX X have  $x \in \text{span } X$  by auto
  with spanL have  $x \in \text{span } (\text{set } Ls)$  by auto
  from finite-in-span[OF - Ls this] obtain c where
     $xc: x = \text{lincomb } c \ (\text{set } Ls)$  by auto
  {
    fix w
    assume wW:  $w \in W$ 
    hence  $w: w \in \text{carrier-vec } n$  using W by auto
    have  $w \cdot x = 0$  unfolding xc using ortho[OF wW]
      by (subst lincomb-scalar-prod-right[OF Ls w], auto)
  }
  with xY x have  $x \in ?I$  by blast
}
thus  $\text{cone } X \subseteq ?I$  by blast
{
  fix x
  let ?X = X - set Bs
  assume  $x \in ?I$ 
  with cone-carrier[OF Y] cone-iff-finite-cone[OF Y finY]
  have xY:  $x \in \text{finite-cone } Y$  and  $x: x \in \text{carrier-vec } n$ 
    and  $w0: \bigwedge w. w \in W \implies w \cdot x = 0$  by auto
  from xY[unfolding finite-cone-def nonneg-lincomb-def] finY obtain d
    where  $xd: x = \text{lincomb } d \ Y$  and  $\text{nonneg: } d \ ' Y \subseteq \text{Collect } ((\leq) 0)$  by auto
  from main[OF - - xd[unfolding Y-def]] w0
  have  $x = \text{lincomb } d \ ?X$  by auto
  hence  $\text{nonneg-lincomb } d \ ?X \ x$  unfolding nonneg-lincomb-def
    using nonneg[unfolding Y-def] by auto
  hence  $x \in \text{finite-cone } ?X$  using finX
    unfolding finite-cone-def by auto
  hence  $x \in \text{cone } X$  using finite-subset[OF - finX, of ?X] unfolding cone-def
by blast
}

```

}
then show $?I \subseteq \text{cone } X$ **by auto**
qed
thus $\text{cone } (X \cup \text{set } Bs) \cap \{x \in \text{carrier-vec } n. \forall w \in W. w \cdot x = 0\} = \text{cone } X$
unfolding $Y\text{-def}$.
qed

definition $\text{polyhedral-cone } (A :: 'a \text{ mat}) = \{x . x \in \text{carrier-vec } n \wedge A *_{\mathbf{v}} x \leq 0_{\mathbf{v}} (\text{dim-row } A)\}$

lemma $\text{polyhedral-cone-carrier}$: **assumes** $A \in \text{carrier-mat } nr \ n$
shows $\text{polyhedral-cone } A \subseteq \text{carrier-vec } n$
using assms **unfolding** $\text{polyhedral-cone-def}$ **by auto**

lemma $\text{cone-in-polyhedral-cone}$:
assumes $CA: C \subseteq \text{polyhedral-cone } A$
and $A: A \in \text{carrier-mat } nr \ n$
shows $\text{cone } C \subseteq \text{polyhedral-cone } A$
proof

interpret $nr: \text{gram-schmidt } nr \ \text{TYPE } ('a)$.
from $\text{polyhedral-cone-carrier}[OF \ A] \ \text{assms}(1)$
have $C: C \subseteq \text{carrier-vec } n$ **by auto**
fix x

assume $x: x \in \text{cone } C$

then have $xn: x \in \text{carrier-vec } n$

using $\text{cone-carrier}[OF \ C]$ **by auto**

from $x[\text{unfolded cone-alt-def}[OF \ C] \ \text{cone-list-def} \ \text{nonneg-lincomb-list-def}]$

obtain $ll \ Ds$ **where** $l0: \text{lincomb-list } ll \ Ds = x$ **and** $l1: \forall i < \text{length } Ds. 0 \leq ll \ i$

and $DsC: \text{set } Ds \subseteq C$

by auto

from $DsC \ C$ **have** $Ds: \text{set } Ds \subseteq \text{carrier-vec } n$ **by auto**

have $A *_{\mathbf{v}} x = A *_{\mathbf{v}} (\text{lincomb-list } ll \ Ds)$ **using** $l0$ **by auto**

also have $\dots = nr.\text{lincomb-list } ll \ (\text{map } (\lambda d. A *_{\mathbf{v}} d) \ Ds)$

proof –

have $one: \forall w \in \text{set } Ds. \text{dim-vec } w = n$ **using** $DsC \ C$ **by auto**

have $two: \forall w \in \text{set } (\text{map } ((*_{\mathbf{v}}) \ A) \ Ds). \text{dim-vec } w = nr$ **using** $A \ DsC \ C$ **by auto**

show $A *_{\mathbf{v}} \text{lincomb-list } ll \ Ds = nr.\text{lincomb-list } ll \ (\text{map } ((*_{\mathbf{v}}) \ A) \ Ds)$

unfolding $\text{lincomb-list-as-mat-mult}[OF \ one] \ nr.\text{lincomb-list-as-mat-mult}[OF \ two] \ \text{length-map}$

proof $(\text{subst } \text{assoc-mult-mat-vec}[\text{symmetric}, \ OF \ A], \ \text{force+}, \ \text{rule } \text{arg-cong}[\text{of } - \ \lambda \ x. \ x *_{\mathbf{v}} \ -])$

show $A * \text{mat-of-cols } n \ Ds = \text{mat-of-cols } nr \ (\text{map } ((*_{\mathbf{v}}) \ A) \ Ds)$

unfolding mat-of-cols-def

by $(\text{intro } \text{eq-matI}, \ \text{insert } A \ Ds[\text{unfolded set-conv-nth}],$

$(\text{force } \text{intro!}: \ \text{arg-cong}[\text{of } - \ \lambda \ x. \ \text{row } A \ - \cdot \ x])+$

qed

qed

```

also have ...  $\leq 0_v$  nr
proof (intro lesseq-vecI[of - nr])
  have *: set (map ((*v) A) Ds)  $\subseteq$  carrier-vec nr using A Ds by auto
  show Carr: nr.lincomb-list ll (map ((*v) A) Ds)  $\in$  carrier-vec nr
    by (intro nr.lincomb-list-carrier[OF *])
  fix i
  assume i:  $i < nr$ 
  from CA[unfolded polyhedral-cone-def] A
  have l2:  $x \in C \implies A *_{v} x \leq 0_v$  nr for x by auto
  show nr.lincomb-list ll (map ((*v) A) Ds)  $\$ i \leq 0_v$  nr  $\$ i$ 
  unfolding subst nr.lincomb-list-index[OF i *] length-map index-zero-vec(1)[OF
i]
proof (intro sum-nonpos mult-nonneg-nonpos)
  fix j
  assume j  $\in \{0..<length Ds\}$ 
  hence j:  $j < length Ds$  by auto
  from j show  $0 \leq ll j$  using l1 by auto
  from j have Ds ! j  $\in C$  using DsC by auto
  from l2[OF this] have l2:  $A *_{v} Ds ! j \leq 0_v$  nr by auto
  from lesseq-vecD[OF - this i] i have (A *v Ds ! j)  $\$ i \leq 0$  by auto
  thus map ((*v) A) Ds ! j  $\$ i \leq 0$  using j i by auto
qed
qed auto
finally show  $x \in polyhedral-cone A$ 
  unfolding polyhedral-cone-def using A xn by auto
qed

lemma bounded-cone-is-zero:
  assumes Ccarr:  $C \subseteq carrier-vec n$  and bnd: cone C  $\subseteq Bounded-vec bnd$ 
  shows cone C =  $\{0_v n\}$ 
proof(rule ccontr)
  assume  $\neg ?thesis$ 
  then obtain v where vC:  $v \in cone C$  and vnz:  $v \neq 0_v n$ 
  using zero-in-cone assms by auto
  have vcarr:  $v \in carrier-vec n$  using vC Ccarr cone-carrier by blast
  from vnz vcarr obtain i where i-le-n:  $i < dim-vec v$  and vnz:  $v \$ i \neq 0$  by
force
  define M where  $M = (1 / (v \$ i) * (bnd + 1))$ 
  have abs-ge-bnd:  $abs (M * (v \$ i)) > bnd$  unfolding M-def by (simp add: vnz)
  have aMvC:  $(abs M) \cdot_v v \in cone C$  using cone-smult[OF - Ccarr vC] abs-ge-bnd
by simp
  have  $\neg (abs (abs M * (v \$ i)) \leq bnd)$  using abs-ge-bnd by simp
  hence (abs M)  $\cdot_v v \notin Bounded-vec bnd$  unfolding Bounded-vec-def using i-le-n
aMvC by auto
  thus False using aMvC bnd by auto
qed

lemma cone-of-cols: fixes A :: 'a mat and b :: 'a vec
  assumes A:  $A \in carrier-mat n nr$  and b:  $b \in carrier-vec n$ 

```

```

shows  $b \in \text{cone} (\text{set} (\text{cols } A)) \longleftrightarrow (\exists x. x \geq 0_v \text{ nr} \wedge A *_v x = b)$ 
proof -
  let  $?C = \text{set} (\text{cols } A)$ 
  from  $A$  have  $C: ?C \subseteq \text{carrier-vec } n$  and  $C': \forall w \in \text{set} (\text{cols } A). \text{dim-vec } w = n$ 
    unfolding  $\text{cols-def}$  by  $\text{auto}$ 
  have  $\text{id}: \text{finite } ?C = \text{True}$   $\text{length} (\text{cols } A) = \text{nr}$  using  $A$  by  $\text{auto}$ 
  have  $\text{Aid}: \text{mat-of-cols } n (\text{cols } A) = A$  using  $A$  unfolding  $\text{mat-of-cols-def}$ 
    by  $(\text{intro } \text{eq-matI}, \text{auto})$ 
  show  $?thesis$ 
    unfolding  $\text{cone-iff-finite-cone}[OF C \text{ finite-set}] \text{ finite-cone-iff-cone-list}[OF C$ 
 $\text{refl}]$ 
    unfolding  $\text{cone-list-def} \text{ nonneg-lincomb-list-def} \text{ mem-Collect-eq id}$ 
    unfolding  $\text{lincomb-list-as-mat-mult}[OF C \uparrow \text{id Aid}]$ 
  proof -
    {
      fix  $x$ 
      assume  $x \geq 0_v \text{ nr} \wedge A *_v x = b$ 
      hence  $\exists c. A *_v \text{vec } nr c = b \wedge (\forall i < nr. 0 \leq c i)$  using  $A b$ 
      by  $(\text{intro } \text{exI}[\text{of } - \lambda i. x \$ i], \text{auto } \text{simp: less-eq-vec-def } \text{intro!}: \text{arg-cong}[\text{of } -$ 
 $(*_v) A])$ 
    }
    moreover
    {
      fix  $c$ 
      assume  $A *_v \text{vec } nr c = b (\forall i < nr. 0 \leq c i)$ 
      hence  $\exists x. x \geq 0_v \text{ nr} \wedge A *_v x = b$ 
      by  $(\text{intro } \text{exI}[\text{of } - \text{vec } nr c], \text{auto } \text{simp: less-eq-vec-def})$ 
    }
    ultimately show  $(\exists c. A *_v \text{vec } nr c = b \wedge (\forall i < nr. 0 \leq c i)) = (\exists x \geq 0_v \text{ nr.}$ 
 $A *_v x = b)$  by  $\text{blast}$ 
  qed
qed

end
end

```

8 Convex Hulls

We define the notion of convex hull of a set or list of vectors and derive basic properties thereof.

```

theory Convex-Hull
  imports Cone
begin

```

```

context gram-schmidt
begin

```

```

definition convex-lincomb  $c \text{ Vs } b = (\text{nonneg-lincomb } c \text{ Vs } b \wedge \text{sum } c \text{ Vs } = 1)$ 

```

definition *convex-lincomb-list* c Vs $b = (\text{nonneg-lincomb-list } c \text{ } Vs \text{ } b \wedge \text{sum } c \{0..<\text{length } Vs\} = 1)$

definition *convex-hull* $Vs = \{x. \exists \text{ } Ws \text{ } c. \text{finite } Ws \wedge Ws \subseteq Vs \wedge \text{convex-lincomb } c \text{ } Ws \text{ } x\}$

lemma *convex-hull-carrier[intro]*: $Vs \subseteq \text{carrier-vec } n \implies \text{convex-hull } Vs \subseteq \text{carrier-vec } n$

unfolding *convex-hull-def convex-lincomb-def nonneg-lincomb-def* **by** *auto*

lemma *convex-hull-mono*: $Vs \subseteq Ws \implies \text{convex-hull } Vs \subseteq \text{convex-hull } Ws$

unfolding *convex-hull-def* **by** *auto*

lemma *convex-lincomb-empty[simp]*: $\neg (\text{convex-lincomb } c \{ \} x)$

unfolding *convex-lincomb-def* **by** *simp*

lemma *set-in-convex-hull*:

assumes $A \subseteq \text{carrier-vec } n$

shows $A \subseteq \text{convex-hull } A$

proof

fix a

assume $a \in A$

hence $a \in \text{carrier-vec } n$ **using** *assms* **by** *auto*

hence $\text{convex-lincomb } (\lambda x. 1) \{a\} a$ **unfolding** *convex-lincomb-def*
by (*auto simp: nonneg-lincomb-def lincomb-def*)

then show $a \in \text{convex-hull } A$ **using** $\langle a \in A \rangle$ **unfolding** *convex-hull-def* **by** *auto*

qed

lemma *convex-hull-empty[simp]*:

$\text{convex-hull } \{ \} = \{ \}$

$A \subseteq \text{carrier-vec } n \implies \text{convex-hull } A = \{ \} \longleftrightarrow A = \{ \}$

proof –

show $\text{convex-hull } \{ \} = \{ \}$ **unfolding** *convex-hull-def* **by** *auto*

then show $A \subseteq \text{carrier-vec } n \implies \text{convex-hull } A = \{ \} \longleftrightarrow A = \{ \}$

using *set-in-convex-hull[of A]* **by** *auto*

qed

lemma *convex-hull-bound*: **assumes** $XBnd: X \subseteq \text{Bounded-vec } Bnd$

and $X: X \subseteq \text{carrier-vec } n$

shows $\text{convex-hull } X \subseteq \text{Bounded-vec } Bnd$

proof

fix x

assume $x \in \text{convex-hull } X$

from *this[unfolded convex-hull-def]*

obtain $Y \text{ } c$ **where** $\text{fin: finite } Y$ **and** $YX: Y \subseteq X$ **and** $cx: \text{convex-lincomb } c \text{ } Y$

by *auto*

from cx [*unfolded convex-lincomb-def nonneg-lincomb-def*]

have $x: x = \text{lincomb } c \text{ } Y$ **and** $\text{sum: sum } c \text{ } Y = 1$ **and** $c0: \bigwedge y. y \in Y \implies c \text{ } y$

≥ 0 **by** *auto*
from $YX\ X\ XBnd$ **have** $Y: Y \subseteq \text{carrier-vec } n$ **and** $YBnd: Y \subseteq \text{Bounded-vec } Bnd$ **by** *auto*
from $x\ Y$ **have** $\text{dim}: \text{dim-vec } x = n$ **by** *auto*
show $x \in \text{Bounded-vec } Bnd$ **unfolding** *Bounded-vec-def mem-Collect-eq dim*
proof (*intro allI impI*)
 fix i
 assume $i: i < n$
 have $\text{abs } (x\ \$\ i) = \text{abs } (\sum x \in Y. c\ x * x\ \$\ i)$ **unfolding** x
 by (*subst lincomb-index[OF i Y], auto*)
 also have $\dots \leq (\sum x \in Y. \text{abs } (c\ x * x\ \$\ i))$ **by** *auto*
 also have $\dots = (\sum x \in Y. \text{abs } (c\ x) * \text{abs } (x\ \$\ i))$ **by** (*simp add: abs-mult*)
 also have $\dots \leq (\sum x \in Y. \text{abs } (c\ x) * Bnd)$
 by (*intro sum-mono mult-left-mono, insert YBnd[unfolding Bounded-vec-def]*
 $i\ Y, \text{force+}$)
 also have $\dots = (\sum x \in Y. \text{abs } (c\ x)) * Bnd$
 by (*simp add: sum-distrib-right*)
 also have $(\sum x \in Y. \text{abs } (c\ x)) = (\sum x \in Y. c\ x)$
 by (*rule sum.cong, insert c0, auto*)
 also have $\dots = 1$ **by** *fact*
 finally show $|x\ \$\ i| \leq Bnd$ **by** *auto*
qed
qed

definition *convex-hull-list* $Vs = \{x. \exists c. \text{convex-lincomb-list } c\ Vs\ x\}$

lemma *lincomb-list-elem*:

set $Vs \subseteq \text{carrier-vec } n \implies$

lincomb-list $(\lambda j. \text{if } i=j \text{ then } 1 \text{ else } 0)\ Vs = (\text{if } i < \text{length } Vs \text{ then } Vs\ !\ i \text{ else } 0_v\ n)$

proof (*induction Vs rule: rev-induct*)

case (*snoc x Vs*)

have $x: x \in \text{carrier-vec } n$ **and** $Vs: \text{set } Vs \subseteq \text{carrier-vec } n$ **using** *snoc.premis* **by** *auto*

let $?f = \lambda j. \text{if } i = j \text{ then } 1 \text{ else } 0$

have *lincomb-list* $?f\ (Vs\ @\ [x]) = \text{lincomb-list } ?f\ Vs + ?f\ (\text{length } Vs) \cdot_v\ x$

using $x\ Vs$ **by** *simp*

also have $\dots = (\text{if } i < \text{length } (Vs\ @\ [x]) \text{ then } (Vs\ @\ [x])\ !\ i \text{ else } 0_v\ n)$ (**is** *?goal*)

using *less-linear[of i length Vs]*

proof (*elim disjE*)

assume $i: i < \text{length } Vs$

have *lincomb-list* $(\lambda j. \text{if } i = j \text{ then } 1 \text{ else } 0)\ Vs = Vs\ !\ i$

using *snoc.IH[OF Vs] i* **by** *auto*

moreover have $(\text{if } i = \text{length } Vs \text{ then } 1 \text{ else } 0) \cdot_v\ x = 0_v\ n$ **using** $i\ x$ **by** *auto*

moreover have $(\text{if } i < \text{length } (Vs\ @\ [x]) \text{ then } (Vs\ @\ [x])\ !\ i \text{ else } 0_v\ n) = Vs\ !\ i$

using *i append-Cons-nth-left* **by** *fastforce*

ultimately show *?goal* **using** $Vs\ i$ *lincomb-list-carrier M.r-zero* **by** *metis*

next

assume $i: i = \text{length } Vs$

```

have lincomb-list ( $\lambda j$ . if  $i = j$  then 1 else 0)  $Vs = 0_v n$ 
  using snoc.IH[OF  $Vs$ ] by auto
moreover have (if  $i = \text{length } Vs$  then 1 else 0)  $\cdot_v x = x$  using  $i x$  by auto
moreover have (if  $i < \text{length } (Vs @ [x])$  then  $(Vs @ [x]) ! i$  else  $0_v n$ ) =  $x$ 
  using  $i$  append-Cons-nth-left by simp
ultimately show ?goal using  $x$  by simp
next
assume  $i: i > \text{length } Vs$ 
have lincomb-list ( $\lambda j$ . if  $i = j$  then 1 else 0)  $Vs = 0_v n$ 
  using snoc.IH[OF  $Vs$ ] by auto
moreover have (if  $i = \text{length } Vs$  then 1 else 0)  $\cdot_v x = 0_v n$  using  $i x$  by auto
moreover have (if  $i < \text{length } (Vs @ [x])$  then  $(Vs @ [x]) ! i$  else  $0_v n$ ) =  $0_v n$ 
  using  $i$  by simp
ultimately show ?goal by simp
qed
finally show ?case by auto
qed simp

lemma set-in-convex-hull-list: fixes  $Vs :: 'a \text{ vec list}$ 
  assumes set  $Vs \subseteq \text{carrier-vec } n$ 
  shows set  $Vs \subseteq \text{convex-hull-list } Vs$ 
proof
fix  $x$  assume  $x \in \text{set } Vs$ 
then obtain  $i$  where  $i: i < \text{length } Vs$ 
  and  $x: x = Vs ! i$  using set-conv-nth[of  $Vs$ ] by auto
let ?f =  $\lambda j$ . if  $i = j$  then 1 else 0  $:: 'a$ 
have lincomb-list ?f  $Vs = x$  using  $i x$  lincomb-list-elem[OF assms] by auto
moreover have  $\forall j < \text{length } Vs. ?f j \geq 0$  by auto
moreover have sum ?f  $\{0..<\text{length } Vs\} = 1$  using  $i$  by simp
ultimately show  $x \in \text{convex-hull-list } Vs$ 
  unfolding convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def
  by auto
qed

lemma convex-hull-list-combination:
  assumes  $Vs: \text{set } Vs \subseteq \text{carrier-vec } n$ 
  and  $x: x \in \text{convex-hull-list } Vs$ 
  and  $y: y \in \text{convex-hull-list } Vs$ 
  and  $l0: 0 \leq l$  and  $l1: l \leq 1$ 
  shows  $l \cdot_v x + (1 - l) \cdot_v y \in \text{convex-hull-list } Vs$ 
proof -
from  $x$  obtain  $cx$  where  $x: \text{lincomb-list } cx \ Vs = x$  and  $cx0: \forall i < \text{length } Vs. cx i \geq 0$ 
  and  $cx1: \text{sum } cx \ \{0..<\text{length } Vs\} = 1$ 
  unfolding convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def
  by auto
from  $y$  obtain  $cy$  where  $y: \text{lincomb-list } cy \ Vs = y$  and  $cy0: \forall i < \text{length } Vs. cy i \geq 0$ 
  and  $cy1: \text{sum } cy \ \{0..<\text{length } Vs\} = 1$ 

```

unfolding *convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def*
by auto
let $?c = \lambda i. l * cx\ i + (1 - l) * cy\ i$
have $set\ Vs \subseteq carrier\text{-}vec\ n \implies$
 $lincomb\text{-}list\ ?c\ Vs = l \cdot_v lincomb\text{-}list\ cx\ Vs + (1 - l) \cdot_v lincomb\text{-}list\ cy\ Vs$
proof (*induction Vs rule: rev-induct*)
case (*snoc v Vs*)
have $v \in carrier\text{-}vec\ n$ **and** $Vs: set\ Vs \subseteq carrier\text{-}vec\ n$
using *snoc.prem* **by auto**
have $lincomb\text{-}list\ ?c\ (Vs\ @\ [v]) = lincomb\text{-}list\ ?c\ Vs + ?c\ (length\ Vs) \cdot_v v$
using *snoc.prem* **by auto**
also have $lincomb\text{-}list\ ?c\ Vs =$
 $l \cdot_v lincomb\text{-}list\ cx\ Vs + (1 - l) \cdot_v lincomb\text{-}list\ cy\ Vs$
by (*rule snoc.IH[OF Vs]*)
also have $?c\ (length\ Vs) \cdot_v v =$
 $l \cdot_v (cx\ (length\ Vs) \cdot_v v) + (1 - l) \cdot_v (cy\ (length\ Vs) \cdot_v v)$
using *add-smult-distrib-vec smult-smult-assoc* **by metis**
also have $l \cdot_v lincomb\text{-}list\ cx\ Vs + (1 - l) \cdot_v lincomb\text{-}list\ cy\ Vs + \dots =$
 $l \cdot_v (lincomb\text{-}list\ cx\ Vs + cx\ (length\ Vs) \cdot_v v) +$
 $(1 - l) \cdot_v (lincomb\text{-}list\ cy\ Vs + cy\ (length\ Vs) \cdot_v v)$
using *lincomb-list-carrier[OF Vs] v*
by (*simp add: M.add.m-assoc M.add.m-lcomm smult-r-distr*)
finally show $?case$ **using** $Vs\ v$ **by simp**
qed simp
hence $lincomb\text{-}list\ ?c\ Vs = l \cdot_v x + (1 - l) \cdot_v y$ **using** $Vs\ x\ y$ **by simp**
moreover have $\forall i < length\ Vs. ?c\ i \geq 0$ **using** $cx0\ cy0\ l0\ l1$ **by simp**
moreover have $sum\ ?c\ \{0..<length\ Vs\} = 1$
proof (*simp add: sum.distrib*)
have $(\sum i = 0..<length\ Vs. (1 - l) * cy\ i) = (1 - l) * sum\ cy\ \{0..<length\ Vs\}$
using *sum-distrib-left* **by metis**
moreover have $(\sum i = 0..<length\ Vs. l * cx\ i) = l * sum\ cx\ \{0..<length\ Vs\}$
using *sum-distrib-left* **by metis**
ultimately show $(\sum i = 0..<length\ Vs. l * cx\ i) + (\sum i = 0..<length\ Vs. (1 - l) * cy\ i) = 1$
using $cx1\ cy1$ **by simp**
qed
ultimately show $?thesis$
unfolding *convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def*
by auto
qed

lemma *convex-hull-list-mono*:
assumes $set\ Vs \subseteq carrier\text{-}vec\ n$
shows $set\ Vs \subseteq set\ Ws \implies convex\text{-}hull\text{-}list\ Vs \subseteq convex\text{-}hull\text{-}list\ Ws$
proof (*standard, induction Vs*)
case *Nil*
from *Nil(2)* **show** $?case$ **unfolding** *convex-hull-list-def convex-lincomb-list-def*
by auto

next
case (*Cons v Vs*)
have $v: v \in \text{set } Ws$ **and** $Vs: \text{set } Vs \subseteq \text{set } Ws$ **using** *Cons.prem1* **by** *auto*
hence $v1: v \in \text{convex-hull-list } Ws$ **using** *set-in-convex-hull-list[OF assms]* **by**
auto
from *Cons.prem2* **obtain** c
where $x: \text{lincomb-list } c (v \# Vs) = x$ **and** $c0: \forall i < \text{length } Vs + 1. c\ i \geq 0$
and $c1: \text{sum } c \{0..<\text{length } Vs + 1\} = 1$
unfolding *convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def*
by *auto*
have $x: x = c\ 0 \cdot_v v + \text{lincomb-list } (c \circ \text{Suc})\ Vs$ **using** $Vs\ v\ \text{assms } x$ **by** *auto*

show *?case proof (cases)*
assume $P: c\ 0 = 1$
hence $\text{sum } (c \circ \text{Suc})\ \{0..<\text{length } Vs\} = 0$
using *sum.atLeast0-lessThan-Suc-shift c1*
by (*metis One-nat-def R.show-r-zero add.right-neutral add-Suc-right*)
moreover **have** $\bigwedge i. i \in \{0..<\text{length } Vs\} \implies (c \circ \text{Suc})\ i \geq 0$
using $c0$ **by** *simp*
ultimately **have** $\forall i \in \{0..<\text{length } Vs\}. (c \circ \text{Suc})\ i = 0$
using *sum-nonneg-eq-0-iff* **by** *blast*
hence $\bigwedge i. i < \text{length } Vs \implies (c \circ \text{Suc})\ i \cdot_v Vs\ !\ i = 0_v\ n$
using $Vs\ \text{assms}$ **by** (*simp add: subset-code(1)*)
hence $\text{lincomb-list } (c \circ \text{Suc})\ Vs = 0_v\ n$
using *lincomb-list-eq-0* **by** *simp*
hence $x = v$ **using** $P\ x\ v\ \text{assms}$ **by** *auto*
thus *?case* **using** $v1$ **by** *auto*

next

assume $P: c\ 0 \neq 1$
have $c1: c\ 0 + \text{sum } (c \circ \text{Suc})\ \{0..<\text{length } Vs\} = 1$
using *sum.atLeast0-lessThan-Suc-shift[of c] c1* **by** *simp*
have $\text{sum } (c \circ \text{Suc})\ \{0..<\text{length } Vs\} \geq 0$ **by** (*rule sum-nonneg, insert c0, simp*)
hence $c\ 0 < 1$ **using** $P\ c1$ **by** *auto*
let $?c' = \lambda i. 1 / (1 - c\ 0) * (c \circ \text{Suc})\ i$
have $\text{sum } ?c'\ \{0..<\text{length } Vs\} = 1 / (1 - c\ 0) * \text{sum } (c \circ \text{Suc})\ \{0..<\text{length } Vs\}$
using $c1\ P\ \text{sum-distrib-left}$ **by** *metis*
hence $\text{sum } ?c'\ \{0..<\text{length } Vs\} = 1$ **using** $P\ c1$ **by** *simp*
moreover **have** $\forall i < \text{length } Vs. ?c'\ i \geq 0$ **using** $c0\ \langle c\ 0 < 1 \rangle$ **by** *simp*
ultimately **have** $c': \text{lincomb-list } ?c'\ Vs \in \text{convex-hull-list } Ws$
using *Cons.IH[OF Vs]*
convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def
by *blast*
have $\text{lincomb-list } ?c'\ Vs = 1 / (1 - c\ 0) \cdot_v \text{lincomb-list } (c \circ \text{Suc})\ Vs$
by (*rule lincomb-list-smult, insert Vs assms, auto*)
hence $(1 - c\ 0) \cdot_v \text{lincomb-list } ?c'\ Vs = \text{lincomb-list } (c \circ \text{Suc})\ Vs$
using P **by** *auto*

hence $x = c \cdot_0 v + (1 - c) \cdot_0 \text{lincomb-list } ?c' \text{ } Vs$ **using** x **by** *auto*
thus $x \in \text{convex-hull-list } Vs$
using *convex-hull-list-combination*[*OF assms v1 c*] $c \cdot_0 \langle c \cdot_0 < 1 \rangle$
by *simp*
qed
qed

lemma *convex-hull-list-eq-set*:
 $\text{set } Vs \subseteq \text{carrier-vec } n \implies \text{set } Vs = \text{set } Vs \implies \text{convex-hull-list } Vs = \text{convex-hull-list } Vs$
using *convex-hull-list-mono* **by** *blast*

lemma *find-indices-empty*: $(\text{find-indices } x \text{ } Vs = []) = (x \notin \text{set } Vs)$
proof (*induction* Vs *rule*: *rev-induct*)
case (*snoc* $v \text{ } Vs$)
show *?case*
proof
assume $\text{find-indices } x \text{ } (Vs @ [v]) = []$
hence $x \neq v \wedge \text{find-indices } x \text{ } Vs = []$ **by** *auto*
thus $x \notin \text{set } (Vs @ [v])$ **using** *snoc* **by** *simp*
next
assume $x \notin \text{set } (Vs @ [v])$
hence $x \neq v \wedge \text{find-indices } x \text{ } Vs = []$ **using** *snoc* **by** *auto*
thus $\text{find-indices } x \text{ } (Vs @ [v]) = []$ **by** *simp*
qed
qed *simp*

lemma *distinct-list-find-indices*:
 $\llbracket i < \text{length } Vs; Vs ! i = x; \text{distinct } Vs \rrbracket \implies \text{find-indices } x \text{ } Vs = [i]$
proof (*induction* Vs *rule*: *rev-induct*)
case (*snoc* $v \text{ } Vs$)
have *dist*: $\text{distinct } Vs$ **and** $xVs: v \notin \text{set } Vs$ **using** *snoc.prem*(3) **by** (*simp-all*)
show *?case*
proof (*cases*)
assume $i: i = \text{length } Vs$
hence $x = v$ **using** *snoc.prem*(2) **by** *auto*
thus *?case* **using** xVs *find-indices-empty* i
by *fastforce*
next
assume $i \neq \text{length } Vs$
hence $i: i < \text{length } Vs$ **using** *snoc.prem*(1) **by** *simp*
hence $Vs ! i: Vs ! i = x$ **using** *snoc.prem*(2) *append-Cons-nth-left* **by** *fastforce*
hence $x \neq v$ **using** *snoc.prem*(3) i **by** *auto*
thus *?case* **using** *snoc.IH*[*OF* $i \text{ } Vs ! i \text{ } \text{dist}$] **by** *simp*
qed
qed *auto*

lemma *finite-convex-hull-iff-convex-hull-list*: **assumes** $Vs: Vs \subseteq \text{carrier-vec } n$
and $id': Vs = \text{set } Vsl'$

```

shows convex-hull Vs = convex-hull-list Vsl'
proof -
  have fin: finite Vs unfolding id' by auto
  from finite-distinct-list fin obtain Vsl
    where id: Vs = set Vsl and dist: distinct Vsl by auto
  from Vs id have Vsl: set Vsl  $\subseteq$  carrier-vec n by auto
  {
    fix c :: nat  $\Rightarrow$  'a
    have distinct Vsl  $\implies$  ( $\sum x \in \text{set } Vsl. \text{sum-list } (\text{map } c \text{ (find-indices } x \text{ Vsl)})$ ) =
      sum c {0.. $\text{length } Vsl$ }
    proof (induction Vsl rule: rev-induct)
      case (snoc v Vsl)
        let ?coef =  $\lambda x. \text{sum-list } (\text{map } c \text{ (find-indices } x \text{ (Vsl @ [v])})$ )
        let ?coef' =  $\lambda x. \text{sum-list } (\text{map } c \text{ (find-indices } x \text{ Vsl)})$ 
        have dist: distinct Vsl using snoc.premis by simp
        have sum ?coef (set (Vsl @ [v])) = sum-list (map ?coef (Vsl @ [v]))
          by (rule sum.distinct-set-conv-list[OF snoc.premis, of ?coef])
        also have ... = sum-list (map ?coef Vsl) + ?coef v by simp
        also have sum-list (map ?coef Vsl) = sum ?coef (set Vsl)
          using sum.distinct-set-conv-list[OF dist, of ?coef] by auto
        also have ... = sum ?coef' (set Vsl)
        proof (intro R.finsum-restrict[of ?coef] restrict-ext, standard)
          fix x
          assume x  $\in$  set Vsl
          then obtain i where i: i < length Vsl and x: x = Vsl ! i
            using in-set-conv-nth[of x Vsl] by blast
          hence (Vsl @ [v]) ! i = x by (simp add: append-Cons-nth-left)
          hence ?coef x = c i
            using distinct-list-find-indices[OF - - snoc.premis] i by fastforce
          also have c i = ?coef' x
            using distinct-list-find-indices[OF i - dist] x by simp
          finally show ?coef x = ?coef' x by auto
        qed
        also have ... = sum c {0.. $\text{length } Vsl$ } by (rule snoc.IH[OF dist])
        also have ?coef v = c (length Vsl)
          using distinct-list-find-indices[OF - - snoc.premis, of length Vsl v]
            nth-append-length by simp
        finally show ?case using sum.atLeast0-lessThan-Suc by simp
      qed simp
    }
  } note sum-sumlist = this
  {
    fix b
    assume b  $\in$  convex-hull-list Vsl
    then obtain c where b: lincomb-list c Vsl = b and c: ( $\forall i < \text{length } Vsl. c \ i$ 
 $\geq 0$ )
      and c1: sum c {0.. $\text{length } Vsl$ } = 1
    unfolding convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def
      by auto
    have convex-lincomb (mk-coeff Vsl c) Vs b
  }

```

```

    unfolding b[symmetric] convex-lincomb-def nonneg-lincomb-def
    apply (subst lincomb-list-as-lincomb[OF Vsl])
  by (insert c c1, auto simp: id mk-coeff-def dist sum-sumlist intro!: sum-list-nonneg)
  hence b ∈ convex-hull Vs
    unfolding convex-hull-def convex-lincomb-def using fin by blast
}
moreover
{
  fix b
  assume b ∈ convex-hull Vs
  then obtain c Ws where Ws: Ws ⊆ Vs and b: lincomb c Ws = b
    and c: c ' Ws ⊆ {x. x ≥ 0} and c1: sum c Ws = 1
    unfolding convex-hull-def convex-lincomb-def nonneg-lincomb-def by auto
  let ?d = λ x. if x ∈ Ws then c x else 0
  have lincomb ?d Vs = lincomb c Ws + lincomb (λ x. 0) (Vs - Ws)
    using lincomb-union2[OF - - Diff-disjoint[of Ws Vs], of c λ x. 0]
    fin Vs Diff-partition[OF Ws] by metis
  also have lincomb (λ x. 0) (Vs - Ws) = 0_v n
    using lincomb-zero[of Vs - Ws λ x. 0] Vs by auto
  finally have lincomb ?d Vs = b using b lincomb-closed Vs Ws by auto
  moreover have ?d ' Vs ⊆ {t. t ≥ 0} using c by auto
  moreover have sum ?d Vs = 1 using c1 R.extend-sum[OF fin Ws] by auto
  ultimately have ∃ c. convex-lincomb c Vs b
    unfolding convex-lincomb-def nonneg-lincomb-def by blast
}
moreover
{
  fix b
  assume ∃ c. convex-lincomb c Vs b
  then obtain c where b: lincomb c Vs = b and c: c ' Vs ⊆ {x. x ≥ 0}
    and c1: sum c Vs = 1
    unfolding convex-lincomb-def nonneg-lincomb-def by auto
  from lincomb-as-lincomb-list-distinct[OF Vsl dist, of c]
  have b: lincomb-list (λ i. c (Vsl ! i)) Vsl = b
    unfolding b[symmetric] id by simp
  have 1 = sum c (set Vsl) using c1 id by auto
  also have ... = sum-list (map c Vsl) by(rule sum.distinct-set-conv-list[OF
dist])
  also have ... = sum (!) (map c Vsl) {0..<length Vsl}
    using sum-list-sum-nth length-map by metis
  also have ... = sum (λ i. c (Vsl ! i)) {0..<length Vsl} by simp
  finally have sum-1: (∑ i = 0..<length Vsl. c (Vsl ! i)) = 1 by simp

  have ∃ c. convex-lincomb-list c Vsl b
    unfolding convex-lincomb-list-def nonneg-lincomb-list-def
    by (intro exI[of - λ i. c (Vsl ! i)] conjI b sum-1)
    (insert c, force simp: set-conv-nth id)
  hence b ∈ convex-hull-list Vsl unfolding convex-hull-list-def by auto
}

```

ultimately have $\text{convex-hull } Vs = \text{convex-hull-list } Vsl$ by auto
also have $\text{convex-hull-list } Vsl = \text{convex-hull-list } Vsl'$
using $\text{convex-hull-list-eq-set}[OF\ Vsl, \text{ of } Vsl']\ id\ id'$ by simp
finally show $?thesis$ by simp
qed

definition $\text{convex } S = (\text{convex-hull } S = S)$

lemma $\text{convex-convex-hull}$: $\text{convex } S \implies \text{convex-hull } S = S$
unfolding convex-def by auto

lemma $\text{convex-hull-convex-hull-listD}$: **assumes** $A: A \subseteq \text{carrier-vec } n$
and $x: x \in \text{convex-hull } A$
shows $\exists\ as.\ \text{set } as \subseteq A \wedge x \in \text{convex-hull-list } as$
proof –
from $x[\text{unfolded convex-hull-def}]$
obtain $X\ c$ **where** $\text{fin}X$: $\text{finite } X$ **and** XA : $X \subseteq A$ **and** $\text{convex-lincomb } c\ X\ x$
by auto
hence $x: x \in \text{convex-hull } X$ **unfolding** convex-hull-def **by** auto
from $\text{finite-list}[OF\ \text{fin}X]$ **obtain** xs **where** $X: X = \text{set } xs$ **by** auto
from $\text{finite-convex-hull-iff-convex-hull-list}[OF\ -\ \text{this}]\ x\ XA\ A$ **have** $x: x \in \text{convex-hull-list } xs$ **by** auto
thus $?thesis$ **using** XA **unfolding** X **by** auto
qed

lemma $\text{convex-hull-convex-sum}$: **assumes** $A: A \subseteq \text{carrier-vec } n$
and $x: x \in \text{convex-hull } A$
and $y: y \in \text{convex-hull } A$
and $a: 0 \leq a \wedge a \leq 1$
shows $a \cdot_v x + (1 - a) \cdot_v y \in \text{convex-hull } A$
proof –
from $\text{convex-hull-convex-hull-listD}[OF\ A\ x]$ **obtain** xs **where** $xs: \text{set } xs \subseteq A$
and $x: x \in \text{convex-hull-list } xs$ **by** auto
from $\text{convex-hull-convex-hull-listD}[OF\ A\ y]$ **obtain** ys **where** $ys: \text{set } ys \subseteq A$
and $y: y \in \text{convex-hull-list } ys$ **by** auto
have $\text{fin}: \text{finite } (\text{set } (xs @ ys))$ **by** auto
have $\text{sub}: \text{set } (xs @ ys) \subseteq A$ **using** $xs\ ys$ **by** auto
from $\text{convex-hull-list-mono}[of\ xs\ @\ ys\ xs]\ x\ \text{sub } A$ **have** $x: x \in \text{convex-hull-list } (xs @ ys)$ **by** auto
from $\text{convex-hull-list-mono}[of\ xs\ @\ ys\ ys]\ y\ \text{sub } A$ **have** $y: y \in \text{convex-hull-list } (xs @ ys)$ **by** auto
from $\text{convex-hull-list-combination}[OF\ -\ x\ y\ a]$
have $a \cdot_v x + (1 - a) \cdot_v y \in \text{convex-hull-list } (xs @ ys)$ **using** $\text{sub } A$ **by** auto
from $\text{finite-convex-hull-iff-convex-hull-list}[of\ -\ xs\ @\ ys]\ \text{this}\ \text{sub } A$
have $a \cdot_v x + (1 - a) \cdot_v y \in \text{convex-hull } (\text{set } (xs @ ys))$ **by** auto
with $\text{convex-hull-mono}[OF\ \text{sub}]$
show $a \cdot_v x + (1 - a) \cdot_v y \in \text{convex-hull } A$ **by** auto
qed

```

lemma convexI: assumes S: S ⊆ carrier-vec n
  and step: ∧ a x y. x ∈ S ⇒ y ∈ S ⇒ 0 ≤ a ⇒ a ≤ 1 ⇒ a ·v x + (1 -
a) ·v y ∈ S
shows convex S
  unfolding convex-def
proof (standard, standard)
  fix z
  assume z ∈ convex-hull S
  from this[unfolded convex-hull-def] obtain W c where finite W and WS: W ⊆
S
  and convex-lincomb c W z by auto
then show z ∈ S
proof (induct W arbitrary: c z)
  case empty
  thus ?case unfolding convex-lincomb-def by auto
next
  case (insert w W c z)
  have convex-lincomb c (insert w W) z by fact
  hence zl: z = lincomb c (insert w W) and nonneg: ∧ w. w ∈ W ⇒ 0 ≤ c w
  and cw: c w ≥ 0
  and sum: sum c (insert w W) = 1
  unfolding convex-lincomb-def nonneg-lincomb-def by auto
  have zl: z = c w ·v w + lincomb c W unfolding zl
  by (rule lincomb-insert2, insert insert S, auto)
  have sum: c w + sum c W = 1 unfolding sum[symmetric]
  by (subst sum.insert, insert insert, auto)
  have W: W ⊆ carrier-vec n and w: w ∈ carrier-vec n using S insert by auto
  show ?case
  proof (cases sum c W = 0)
  case True
  with nonneg have c0: ∧ w. w ∈ W ⇒ c w = 0
  using insert(1) sum-nonneg-eq-0-iff by auto
  with sum have cw: c w = 1 by auto
  have lin0: lincomb c W = 0v n
  by (intro lincomb-zero W, insert c0, auto)
  have z = w unfolding zl cw lin0 using w by simp
  with insert(4) show ?thesis by simp
  next
  case False
  have sum c W ≥ 0 using nonneg by (metis sum-nonneg)
  with False have pos: sum c W > 0 by auto
  define b where b = (λ w. inverse (sum c W) * c w)
  have convex-lincomb b W (lincomb b W)
  unfolding convex-lincomb-def nonneg-lincomb-def b-def
  proof (intro conjI refl)
  show (λw. inverse (sum c W) * c w) ‘ W ⊆ Collect ((≤) 0) using nonneg
pos by auto
  show (∑ w∈W. inverse (sum c W) * c w) = 1 unfolding sum-distrib-left[symmetric]
using False by auto

```

```

qed
from insert(3)[OF - this] insert
have IH: lincomb b W ∈ S by auto
have lin: lincomb c W = sum c W ·v lincomb b W
  unfolding b-def
  by (subst lincomb-smult[symmetric, OF W], rule lincomb-cong[OF - W],
insert False, auto)
from sum cw pos have sum: sum c W = 1 - c w and cw1: c w ≤ 1 by auto
show ?thesis unfolding zl lin unfolding sum
  by (rule step[OF - IH cw cw1], insert insert, auto)
qed
qed
next
show S ⊆ convex-hull S using S by (rule set-in-convex-hull)
qed

```

```

lemma convex-hulls-are-convex: assumes A ⊆ carrier-vec n
shows convex (convex-hull A)
by (intro convexI convex-hull-convex-sum convex-hull-carrier assms)

```

```

lemma convex-hull-sum: assumes A: A ⊆ carrier-vec n and B: B ⊆ carrier-vec
n
shows convex-hull (A + B) = convex-hull A + convex-hull B
proof
note cA = convex-hull-carrier[OF A]
note cB = convex-hull-carrier[OF B]
have convex (convex-hull A + convex-hull B)
proof (intro convexI sum-carrier-vec convex-hull-carrier A B)
fix a :: 'a and x1 x2
assume x1 ∈ convex-hull A + convex-hull B x2 ∈ convex-hull A + convex-hull
B

```

```

then obtain y1 y2 z1 z2 where
x12: x1 = y1 + z1 x2 = y2 + z2 and
y12: y1 ∈ convex-hull A y2 ∈ convex-hull A and
z12: z1 ∈ convex-hull B z2 ∈ convex-hull B
unfolding set-plus-def by auto
from y12 z12 cA cB have carr:
y1 ∈ carrier-vec n y2 ∈ carrier-vec n
z1 ∈ carrier-vec n z2 ∈ carrier-vec n
by auto
assume a: 0 ≤ a a ≤ 1
have A: a ·v y1 + (1 - a) ·v y2 ∈ convex-hull A using y12 a A by (metis
convex-hull-convex-sum)
have B: a ·v z1 + (1 - a) ·v z2 ∈ convex-hull B using z12 a B by (metis
convex-hull-convex-sum)
have a ·v x1 + (1 - a) ·v x2 = (a ·v y1 + a ·v z1) + ((1 - a) ·v y2 + (1 -
a) ·v z2) unfolding x12
using carr by (auto simp: smult-add-distrib-vec)
also have ... = (a ·v y1 + (1 - a) ·v y2) + (a ·v z1 + (1 - a) ·v z2) using

```

```

carr
  by (intro eq-vecI, auto)
  finally show  $a \cdot_v x1 + (1 - a) \cdot_v x2 \in \text{convex-hull } A + \text{convex-hull } B$ 
  using A B by auto
qed
from convex-convex-hull[OF this]
have id:  $\text{convex-hull } (\text{convex-hull } A + \text{convex-hull } B) = \text{convex-hull } A + \text{convex-hull } B$  .
show  $\text{convex-hull } (A + B) \subseteq \text{convex-hull } A + \text{convex-hull } B$ 
  by (subst id[symmetric], rule convex-hull-mono[OF set-plus-mono2]; intro set-in-convex-hull A B)
show  $\text{convex-hull } A + \text{convex-hull } B \subseteq \text{convex-hull } (A + B)$ 
proof
  fix x
  assume  $x \in \text{convex-hull } A + \text{convex-hull } B$ 
  then obtain y z where  $x = y + z$  and  $y \in \text{convex-hull } A$  and  $z \in \text{convex-hull } B$ 
  by (auto simp: set-plus-def)
  from convex-hull-convex-hull-listD[OF A y] obtain ys where  $ysA: \text{set } ys \subseteq A$ 
  and
   $y \in \text{convex-hull-list } ys$  by auto
  from convex-hull-convex-hull-listD[OF B z] obtain zs where  $zsB: \text{set } zs \subseteq B$ 
  and
   $z \in \text{convex-hull-list } zs$  by auto
  from  $y \in \text{convex-hull-list } ys$  [unfolded convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def]
  obtain c where  $yid: y = \text{lincomb-list } c \ ys$ 
  and  $\text{conv-c}: (\forall i < \text{length } ys. 0 \leq c \ i) \wedge \text{sum } c \ \{0..<\text{length } ys\} = 1$ 
  by auto
  from  $z \in \text{convex-hull-list } zs$  [unfolded convex-hull-list-def convex-lincomb-list-def nonneg-lincomb-list-def]
  obtain d where  $zid: z = \text{lincomb-list } d \ zs$ 
  and  $\text{conv-d}: (\forall i < \text{length } zs. 0 \leq d \ i) \wedge \text{sum } d \ \{0..<\text{length } zs\} = 1$ 
  by auto
  from  $ysA \ A$  have  $ys: \text{set } ys \subseteq \text{carrier-vec } n$  by auto
  from  $zsB \ B$  have  $zs: \text{set } zs \subseteq \text{carrier-vec } n$  by auto
  have [intro, simp]:  $\text{lincomb-list } x \ ys \in \text{carrier-vec } n$  for x using lincomb-list-carrier[OF ys] .
  have [intro, simp]:  $\text{lincomb-list } x \ zs \in \text{carrier-vec } n$  for x using lincomb-list-carrier[OF zs] .
  have [dim][simp]:  $\text{dim-vec } (\text{lincomb-list } d \ zs) = n$  by auto
  from yid have  $y: y \in \text{carrier-vec } n$  by auto
  from zid have  $z: z \in \text{carrier-vec } n$  by auto
  {
  fix x
  assume  $x \in \text{set } (\text{map } ((+) \ y) \ zs)$ 
  then obtain z where  $x = y + z$  and  $z \in \text{set } zs$  by auto
  then obtain j where  $j: j < \text{length } zs$  and  $x: x = y + zs \ ! \ j$  unfolding
set-conv-nth by auto
  hence mem:  $zs \ ! \ j \in \text{set } zs$  by auto
  hence zsj:  $zs \ ! \ j \in \text{carrier-vec } n$  using zs by auto

```

```

let ?list = (map (λ y. y + zs ! j) ys)
let ?set = set ?list
have set: ?set ⊆ carrier-vec n using ys A zsj by auto
have lin-map: lincomb-list c ?list ∈ carrier-vec n
  by (intro lincomb-list-carrier[OF set])
have y + (zs ! j) = lincomb-list c ?list
unfolding yid using zsj lin-map lincomb-list-index[OF - set] lincomb-list-index[OF
- ys]
  by (intro eq-vecI, auto simp: field-simps sum-distrib-right[symmetric] conv-c)
hence convex-lincomb-list c ?list (y + (zs ! j))
  unfolding convex-lincomb-list-def nonneg-lincomb-list-def using conv-c by
auto
hence y + (zs ! j) ∈ convex-hull-list ?list unfolding convex-hull-list-def by
auto
with finite-convex-hull-iff-convex-hull-list[OF set refl]
have (y + zs ! j) ∈ convex-hull ?set by auto
also have ... ⊆ convex-hull (A + B)
  by (rule convex-hull-mono, insert mem ys ysA zsj, force simp: set-plus-def)
finally have x ∈ convex-hull (A + B) unfolding x .
} note step1 = this
{
let ?list = map ((+) y) zs
let ?set = set ?list
have set: ?set ⊆ carrier-vec n using zs B y by auto
have lin-map: lincomb-list d ?list ∈ carrier-vec n
  by (intro lincomb-list-carrier[OF set])
have [simp]: i < n ⇒ (∑ j = 0..<length zs. d j * (y + zs ! j) $ i) =
  (∑ j = 0..<length zs. d j * (y $ i + zs ! j $ i)) for i
  by (rule sum.cong, insert zs[unfolded set-conv-nth] y, auto)
have y + z = lincomb-list d ?list
unfolding zid using y zs lin-map lincomb-list-index[OF - set] lincomb-list-index[OF
- zs]
  set lincomb-list-carrier[OF zs, of d] zs[unfolded set-conv-nth]
by (intro eq-vecI, auto simp: field-simps sum-distrib-right[symmetric] conv-d)
hence convex-lincomb-list d ?list x unfolding x
  unfolding convex-lincomb-list-def nonneg-lincomb-list-def using conv-d by
auto
hence x ∈ convex-hull-list ?list unfolding convex-hull-list-def by auto
with finite-convex-hull-iff-convex-hull-list[OF set refl]
have x ∈ convex-hull ?set by auto
also have ... ⊆ convex-hull (convex-hull (A + B))
  by (rule convex-hull-mono, insert step1, auto)
also have ... = convex-hull (A + B)
by (rule convex-convex-hull[OF convex-hulls-are-convex], intro sum-carrier-vec
A B)
finally show x ∈ convex-hull (A + B) .
}
qed
qed

```

```

lemma convex-hull-in-cone:
  convex-hull  $C \subseteq \text{cone } C$ 
  unfolding convex-hull-def cone-def convex-lincomb-def finite-cone-def by auto

lemma convex-cone:
  assumes  $C: C \subseteq \text{carrier-vec } n$ 
  shows convex (cone  $C$ )
  unfolding convex-def
  using convex-hull-in-cone set-in-convex-hull[OF cone-carrier[OF C]] cone-cone[OF C]
  by blast

end
end

```

9 Normal Vectors

We provide a function for the normal vector of a half-space (given as $n-1$ linearly independent vectors). We further provide a function that returns a list of normal vectors that span the orthogonal complement of some subspace of R^n . Bounds for all normal vectors are provided.

```

theory Normal-Vector
  imports
    Integral-Bounded-Vectors
    Basis-Extension
begin

context gram-schmidt
begin

lemma ortho-sum-in-span:
  assumes  $W: W \subseteq \text{carrier-vec } n$ 
  and  $X: X \subseteq \text{carrier-vec } n$ 
  and ortho:  $\bigwedge w x. w \in W \implies x \in X \implies x \cdot w = 0$ 
  and inspan:  $\text{lincomb } l1 X + \text{lincomb } l2 W \in \text{span } X$ 
  shows  $\text{lincomb } l2 W = 0_v n$ 
proof (rule ccontr)
  let  $?v = \text{lincomb } l2 W$ 
  have  $vcarr: ?v \in \text{carrier-vec } n$  using  $W$  by auto
  have  $vspan: ?v \in \text{span } W$  using  $W$  by auto
  assume  $\neg ?thesis$ 
  from this have  $vnz: ?v \neq 0_v n$  by auto
  let  $?x = \text{lincomb } l1 X$ 
  have  $xcarr: ?x \in \text{carrier-vec } n$  using  $X$  by auto
  have  $xspan: ?x \in \text{span } X$  using  $X$   $xcarr$  by auto
  have  $0 \neq \text{sq-norm } ?v$  using  $vnz$   $vcarr$  by simp
  also have  $\text{sq-norm } ?v = 0 + ?v \cdot ?v$  by (simp add: sq-norm-vec-as-cscalar-prod)

```

also have $\dots = ?x \cdot ?v + ?v \cdot ?v$
by (*subst* (2) *ortho-span-span*[*OF* *X* *W* *ortho*], *insert* *X* *W*, *auto*)
also have $\dots = (?x + ?v) \cdot ?v$ **using** *xcarr* *vcarr*
using *add-scalar-prod-distrib* **by** *force*
also have $\dots = 0$
by (*rule* *ortho-span-span*[*OF* *X* *W* *ortho* *in**span* *vspan*])
finally show *False* **by** *simp*
qed

lemma *ortho-lin-indpt*: **assumes** *W*: $W \subseteq \text{carrier-vec } n$
and *X*: $X \subseteq \text{carrier-vec } n$
and *ortho*: $\bigwedge w x. w \in W \implies x \in X \implies x \cdot w = 0$
and *linW*: *lin-indpt* *W*
and *linX*: *lin-indpt* *X*
shows *lin-indpt* ($W \cup X$)
proof (*rule* *ccontr*)
assume $\neg ?thesis$
from *this* **obtain** *c* **where** *zerocomb*: *lincomb* *c* ($W \cup X$) = 0_v *n*
and *notallz*: $\exists v \in (W \cup X). c v \neq 0$
using *assms* *fin-dim* *fin-dim-li-fin* *finite-lin-indpt2* *infinite-Un* *le-sup-iff*
by *metis*
have *zero-nin-W*: 0_v *n* $\notin W$ **using** *assms* **by** (*metis* *vs-zero-lin-dep*)
have *WXinters*: $W \cap X = \{\}$
proof (*rule* *ccontr*)
assume $\neg ?thesis$
from *this* **obtain** *v* **where** *v*: $v \in W \cap X$ **by** *auto*
hence $v \cdot v = 0$ **using** *ortho* **by** *auto*
moreover have $v \in \text{carrier-vec } n$ **using** *assms* *v* **by** *auto*
ultimately have $v = 0_v$ *n* **using** *sq-norm-vec-as-cscalar-prod*[*of* *v*] **by** *auto*
then show *False* **using** *zero-nin-W* *v* **by** *auto*
qed
have *finX*: *finite* *X* **using** *X* *linX* **by** (*simp* *add*: *fin-dim-li-fin*)
have *finW*: *finite* *W* **using** *W* *linW* **by** (*simp* *add*: *fin-dim-li-fin*)
have *split*: *lincomb* *c* ($W \cup X$) = *lincomb* *c* *X* + *lincomb* *c* *W*
using *lincomb-union*[*OF* *W* *X* *WXinters* *finW* *finX*]
by (*simp* *add*: *M.add.m-comm* *W* *X*)
hence *lincomb* *c* *X* + *lincomb* *c* *W* $\in \text{span } X$ **using** *zerocomb*
using *local.span-zero* **by** *auto*
hence *z1*: *lincomb* *c* *W* = 0_v *n*
using *ortho-sum-in-span*[*OF* *W* *X* *ortho*] **by** *simp*
hence *z2*: *lincomb* *c* *X* = 0_v *n* **using** *split* *zerocomb* *X* **by** *simp*
have *or*: $(\exists v \in W. c v \neq 0) \vee (\exists v \in X. c v \neq 0)$ **using** *notallz* **by** *auto*
have *ex1*: $\exists v \in W. c v \neq 0 \implies \text{False}$ **using** *linW*
using *finW* *lin-dep-def* *z1* **by** *blast*
have *ex2*: $\exists v \in X. c v \neq 0 \implies \text{False}$ **using** *linX*
using *finX* *lin-dep-def* *z2* **by** *blast*
show *False* **using** *ex1* *ex2* *or* **by** *auto*
qed

definition *normal-vector* :: 'a vec set \Rightarrow 'a vec **where**
normal-vector $W = (\text{let } ws = (\text{SOME } ws. \text{set } ws = W \wedge \text{distinct } ws);$
 $m = \text{length } ws;$
 $B = (\lambda j. \text{mat } m m (\lambda(i, j'). ws ! i \$ (\text{if } j' < j \text{ then } j' \text{ else } \text{Suc } j')))$
 $\text{in } \text{vec } n (\lambda j. (-1)^{\wedge(m+j)} * \text{det } (B j))$)

lemma *normal-vector*: **assumes** *fin*: *finite* W
and *card*: $\text{Suc } (\text{card } W) = n$
and *lin*: *lin-indpt* W
and $W: W \subseteq \text{carrier-vec } n$
shows *normal-vector* $W \in \text{carrier-vec } n$
normal-vector $W \neq 0_v n$
 $w \in W \Longrightarrow w \cdot \text{normal-vector } W = 0$
 $w \in W \Longrightarrow \text{normal-vector } W \cdot w = 0$
lin-indpt (*insert* (*normal-vector* W) W)
normal-vector $W \notin W$
is-det-bound $db \Longrightarrow W \subseteq \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } Bnd) \Longrightarrow \text{normal-vector } W \in \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } (db (n-1) Bnd))$

proof –
define ws **where** $ws = (\text{SOME } ws. \text{set } ws = W \wedge \text{distinct } ws)$
from *finite-distinct-list*[*OF* *fin*]
have $\exists ws. \text{set } ws = W \wedge \text{distinct } ws$ **by** *auto*
from *someI-ex*[*OF* *this*, *folded* *ws-def*] **have** *id*: $\text{set } ws = W$ **and** *dist*: *distinct* ws **by** *auto*
have *len*: $\text{length } ws = \text{card } W$ **using** *distinct-card*[*OF* *dist*] *id* **by** *auto*
let $?n = \text{length } ws$
define B **where** $B = (\lambda j. \text{mat } ?n ?n (\lambda(i, j'). ws ! i \$ (\text{if } j' < j \text{ then } j' \text{ else } \text{Suc } j')))$
define nv **where** $nv = \text{vec } n (\lambda j. (-1)^{\wedge(?n+j)} * \text{det } (B j))$
have $nv2$: *normal-vector* $W = nv$ **unfolding** *normal-vector-def* *Let-def*
 $ws\text{-def}$ [*symmetric*] *B-def* *nv-def* ..
define A **where** $A = (\lambda w. \text{mat-of-rows } n (ws @ [w]))$
from *len id card* **have** *len*: $\text{Suc } ?n = n$ **by** *auto*
have $A: A w \in \text{carrier-mat } n n$ **for** w **using** *id* W *len* **unfolding** *A-def* **by** *auto*
{
fix $w :: 'a \text{ vec}$
assume $w: w \in \text{carrier-vec } n$
from *len* **have** $n1$ [*simp*]: $n - \text{Suc } 0 = ?n$ **by** *auto*
{
fix j
assume $j: j < n$
have *mat-delete* ($A w$) $?n j = B j$
unfolding *mat-delete-def* *A-def* *mat-of-rows-def* *B-def*
by (*rule* *eq-matI*, *insert* j *len*, *auto* *simp*: *nth-append*)
} **note** $B = \text{this}$
have $\text{det } (A w) = (\sum j < n. (A w) \$ \$ (\text{length } ws, j) * \text{cofactor } (A w) ?n j)$
by (*subst* *laplace-expansion-row*[*OF* A , *of* $?n$], *insert* *len*, *auto*)
also **have** $\dots = (\sum j < n. w \$ j * (-1)^{\wedge(?n+j)} * \text{det } (\text{mat-delete } (A w) ?n j))$

```

    by (rule sum.cong, auto simp: A-def mat-of-rows-def cofactor-def)
  also have ... = (∑ j<n. w $ j * (-1)^(?n+j) * det (B j))
    by (rule sum.cong[OF refl], subst B, auto)
  also have ... = (∑ j<n. w $ j * nv $ j)
    by (rule sum.cong[OF refl], auto simp: nv-def)
  also have ... = w · nv unfolding scalar-prod-def unfolding nv-def
    by (rule sum.cong, auto)
  finally have det (A w) = w · nv .
} note det-scalar = this
have nv: nv ∈ carrier-vec n unfolding nv-def by auto
{
  fix w
  assume wW: w ∈ W
  with W have w: w ∈ carrier-vec n by auto
  from wW id obtain i where i: i < ?n and ws: ws ! i = w unfolding
set-conv-nth by auto
  from det-scalar[OF w] have det (A w) = w · nv .
  also have det (A w) = 0
    by (subst det-identical-rows[OF A, of i ?n], insert i ws len, auto simp: A-def
mat-of-rows-def nth-append)
  finally have w · nv = 0 ..
  note this this[unfolded comm-scalar-prod[OF w nv]]
} note ortho = this
have nv0: nv ≠ 0v n
proof
  assume nv: nv = 0v n
  define bs where bs = basis-extension ws
  define w where w = hd bs
  have lin-indpt-list ws using dist lin W unfolding lin-indpt-list-def id by auto
  from basis-extension[OF this, folded bs-def] len
  have lin: lin-indpt-list (ws @ bs) and length bs = 1 and bsc: set bs ⊆ carrier-vec
n
    by (auto simp: unit-vecs-def)
  hence bs: bs = [w] unfolding w-def by (cases bs, auto)
  with bsc have w: w ∈ carrier-vec n by auto
  note lin = lin[unfolded bs]
  from lin-indpt-list-length-eq-n[OF lin] len
  have basis: basis (set (ws @ [w])) by auto
  from w det-scalar nv have det0: det (A w) = 0 by auto
  with basis-det-nonzero[OF basis] len show False
    unfolding A-def by auto
qed
let ?nv = normal-vector W
from ortho nv nv0
show nv: ?nv ∈ carrier-vec n
  and ortho: ∧ w. w ∈ W ⇒ w · ?nv = 0
  ∧ w. w ∈ W ⇒ ?nv · w = 0
  and n0: ?nv ≠ 0v n unfolding nv2 by auto
from n0 nv have sq-norm ?nv ≠ 0 by auto

```

```

hence  $nnv: ?nv \cdot ?nv \neq 0$  by (auto simp: sq-norm-vec-as-cscalar-prod)
show  $nvW: ?nv \notin W$  using  $nnv$  ortho by blast
have  $?nv \notin \text{span } W$  using  $W$  ortho  $nnv$   $nv$ 
  using orthocompl-span by blast
with lin-dep-iff-in-span[OF  $W$  lin  $nv$   $nvW$ ]
show lin-indpt (insert  $?nv$   $W$ ) by auto
{
  assume  $db: \text{is-det-bound } db$ 
  assume  $W \subseteq \mathbb{Z}_v \cap \text{Bounded-vec}$  (of-int  $Bnd$ )
  hence  $wsI: \text{set } ws \subseteq \mathbb{Z}_v \cap \text{Bounded-vec}$  (of-int  $Bnd$ ) unfolding  $id$  by auto
  have  $ws: \text{set } ws \subseteq \text{carrier-vec } n$  using  $W$  unfolding  $id$  by auto
  from  $wsI$   $ws$  have  $wsI: i < ?n \implies ws ! i \in \mathbb{Z}_v \cap \text{Bounded-vec}$  (of-int  $Bnd$ )  $\cap$ 
  carrier-vec  $n$  for  $i$ 
    using len  $wsI$  unfolding set-conv-nth by auto
  have  $ints: i < ?n \implies j < n \implies ws ! i \ \$ j \in \mathbb{Z}$  for  $i$   $j$ 
    using  $wsI$ [of  $i$ , unfolded Ints-vec-def] by force
  have  $bnd: i < ?n \implies j < n \implies \text{abs } (ws ! i \ \$ j) \leq \text{of-int } Bnd$  for  $i$   $j$ 
    using  $wsI$ [unfolded Bounded-vec-def, of  $i$ ] by auto
  {
    fix  $i$ 
    assume  $i: i < n$ 
    have  $ints-nv: nv \ \$ i \in \mathbb{Z}$  unfolding  $nv$ -def using  $wsI$  len  $ws$ 
      by (auto simp:  $i$   $B$ -def set-conv-nth intro!: Ints-mult Ints-det ints)
    have  $B$   $i \in \mathbb{Z}_m \cap \text{Bounded-mat}$  (of-int  $Bnd$ )
      unfolding  $B$ -def using len  $ws$   $i$   $bnd$   $ints-nv$ 
    apply (simp add: Ints-mat-def Ints-vec-def Bounded-mat-def, intro allI impI)
    subgoal for  $ii$   $j$  using  $ints$ [of  $ii$   $j$ ]  $ints$ [of  $ii$   $Suc$   $j$ ]
      by auto
    done
    from is-det-bound-of-int[OF  $db$  - this, of  $?n$ ]
    have  $|nv \ \$ i| \leq \text{of-int } (db (n - 1) Bnd)$ 
      unfolding  $nv$ -def using  $wsI$  len  $ws$   $i$ 
      by (auto simp:  $B$ -def abs-mult  $bnd$ )
    note  $ints-nv$  this
  }
  with  $nv$   $nv2$  show  $?nv \in \mathbb{Z}_v \cap \text{Bounded-vec}$  (of-int (db (n - 1) Bnd))
    unfolding Ints-vec-def Bounded-vec-def by auto
}
}
qed

```

lemma normal-vector-span:

```

assumes  $card: \text{Suc } (card D) = n$ 
  and  $D: D \subseteq \text{carrier-vec } n$  and  $fin: \text{finite } D$  and  $lin: \text{lin-indpt } D$ 
shows  $\text{span } D = \{ x. x \in \text{carrier-vec } n \wedge x \cdot \text{normal-vector } D = 0 \}$ 
proof -
  note  $nv = \text{normal-vector}$ [OF  $fin$   $card$   $lin$   $D$ ]
  {
    fix  $x$ 
    assume  $xspan: x \in \text{span } D$ 

```

```

from finite-in-span[OF fin D xspan] obtain c where
  x · normal-vector D = lincomb c D · normal-vector D by auto
also have ... = ( $\sum_{w \in D}. c w * (w \cdot \text{normal-vector } D)$ )
  by (rule lincomb-scalar-prod-left, insert D nv, auto)
also have ... = 0
apply (rule sum.neutral) using nv(1,2,3) D comm-scalar-prod[of normal-vector
D] by fastforce
  finally have  $x \in \text{carrier-vec } n$   $x \cdot \text{normal-vector } D = 0$  using xspan D by
auto
}
moreover
{
  let ?n = normal-vector D
  fix x
  assume  $x \in \text{carrier-vec } n$  and xscal:  $x \cdot \text{normal-vector } D = 0$ 
  let ?B = (insert (normal-vector D) D)
  have card ?B = n using card card-insert-disjoint[OF fin nv(6)] by auto
  moreover have B:  $?B \subseteq \text{carrier-vec } n$  using D nv by auto
  ultimately have span ?B = carrier-vec n
    by (intro span-carrier-lin-indpt-card-n, insert nv(5), auto)
  hence xspan:  $x \in \text{span } ?B$  using x by auto
  obtain c where  $x = \text{lincomb } c ?B$  using finite-in-span[OF - B xspan] fin by
auto
  hence  $0 = \text{lincomb } c ?B \cdot \text{normal-vector } D$  using xscal by auto
  also have ... = ( $\sum_{w \in ?B}. c w * (w \cdot \text{normal-vector } D)$ )
    by (subst lincomb-scalar-prod-left, insert B, auto)
  also have ... = ( $\sum_{w \in D}. c w * (w \cdot \text{normal-vector } D)$ ) +  $c ?n * (?n \cdot ?n)$ 
    by (subst sum.insert[OF fin nv(6)], auto)
  also have ( $\sum_{w \in D}. c w * (w \cdot \text{normal-vector } D)$ ) = 0
    apply(rule sum.neutral) using nv(1,3) comm-scalar-prod[OF nv(1)] D by
fastforce
  also have  $?n \cdot ?n = \text{sq-norm } ?n$  using sq-norm-vec-as-cscalar-prod[of ?n] by
simp
  finally have  $c ?n * \text{sq-norm } ?n = 0$  by simp
  hence ncoord:  $c ?n = 0$  using nv(1-5) by auto
  have  $x = \text{lincomb } c ?B$  by fact
  also have ... = lincomb c D
    apply (subst lincomb-insert2[OF fin D - nv(6,1)]) using ncoord nv(1) D by
auto
  finally have  $x \in \text{span } D$  using fin by auto
}
ultimately show ?thesis by auto
qed

```

definition *normal-vectors* :: 'a *vec list* \Rightarrow 'a *vec list* **where**
normal-vectors ws = (*let us* = *basis-extension ws*
in map ($\lambda i. \text{normal-vector (set (ws @ us) - \{us ! i\})}$) [$0..<\text{length } us$])

lemma *normal-vectors*:

```

assumes lin: lin-indpt-list ws
shows set (normal-vectors ws) ⊆ carrier-vec n
  w ∈ set ws ⇒ nv ∈ set (normal-vectors ws) ⇒ nv · w = 0
  w ∈ set ws ⇒ nv ∈ set (normal-vectors ws) ⇒ w · nv = 0
  lin-indpt-list (ws @ normal-vectors ws)
  length ws + length (normal-vectors ws) = n
  set ws ∩ set (normal-vectors ws) = {}
  is-det-bound db ⇒ set ws ⊆ Zv ∩ Bounded-vec (of-int Bnd) ⇒
    set (normal-vectors ws) ⊆ Zv ∩ Bounded-vec (of-int (db (n-1) (max 1 Bnd)))
proof -
  define us where us = basis-extension ws
  from basis-extension[OF assms, folded us-def]
  have units: set us ⊆ set (unit-vecs n)
    and lin: lin-indpt-list (ws @ us)
    and len: length (ws @ us) = n
    by auto
  from lin-indpt-list-length-eq-n[OF lin len]
  have span: span (set (ws @ us)) = carrier-vec n by auto
  from lin[unfolded lin-indpt-list-def]
  have wsus: set (ws @ us) ⊆ carrier-vec n
    and dist: distinct (ws @ us)
    and lin': lin-indpt (set (ws @ us)) by auto
  let ?nv = normal-vectors ws
  note nv-def = normal-vectors-def[of ws, unfolded Let-def, folded us-def]
  let ?m = length ws
  let ?n = length us
  have lnv[simp]: length ?nv = length us unfolding nv-def by auto
  {
    fix i
    let ?V = set (ws @ us) - {us ! i}
    assume i: i < ?n
    hence nvi: ?nv ! i = normal-vector ?V unfolding nv-def by auto
    from i have us ! i ∈ set us by auto
    with wsus have u: us ! i ∈ carrier-vec n by auto
    have id: ?V ∪ {us ! i} = set (ws @ us) using i by auto
    have V: ?V ⊆ carrier-vec n using wsus by auto
    have finV: finite ?V by auto
    have Suc (card ?V) = card (insert (us ! i) ?V)
      by (subst card-insert-disjoint[OF finV], auto)
    also have insert (us ! i) ?V = set (ws @ us) using i by auto
    finally have cardV: Suc (card ?V) = n
      using len distinct-card[OF dist] by auto
    from subset-li-is-li[OF lin'] have linV: lin-indpt ?V by auto
    from lin-dep-iff-in-span[OF - linV u, unfolded id] wsus lin'
    have usV: us ! i ∉ span ?V by auto
    note nv = normal-vector[OF finV cardV linV V, folded nvi]
    from normal-vector-span[OF cardV V - linV, folded nvi] comm-scalar-prod[OF
    - nv(1)]
    have span: span ?V = {x ∈ carrier-vec n. ?nv ! i · x = 0}
  }

```

```

    by auto
  from  $nv(1,2)$  have  $sq\text{-norm } (?nv ! i) \neq 0$  by auto
  hence  $nvi: ?nv ! i \cdot ?nv ! i \neq 0$ 
    by (auto simp:  $sq\text{-norm-vec-as-cscalar-prod}$ )
  from  $span\ nvi$  have  $nvspan: ?nv ! i \notin span\ ?V$  by auto
  from  $u\ usV[un\ folded\ span]$  have  $?nv ! i \cdot us ! i \neq 0$  by blast
  note  $nv\ nvi\ this\ span\ usV\ nvspan$ 
} note  $nvi = this$ 
show  $nv: set\ ?nv \subseteq carrier\text{-vec } n$ 
  unfolding  $set\text{-conv-nth}$  using  $nvi(1)$  by auto
{
  fix  $w\ nv$ 
  assume  $w: w \in set\ ws$ 
  with  $dist$  have  $wus: w \notin set\ us$  by auto
  assume  $n: nv \in set\ ?nv$ 
  with  $w\ wus$  show  $nv \cdot w = 0$ 
    unfolding  $set\text{-conv-nth}[of\ normal\text{-vectors } -]$  by (auto intro!:  $nvi(4)[of\ -\ w]$ )
    thus  $w \cdot nv = 0$  using  $comm\text{-scalar-prod}[of\ w\ n\ nv]$   $w\ nv\ n\ wsus$  by auto
} note  $scalar\text{-}0 = this$ 
show  $length\ ws + length\ ?nv = n$  using  $len$  by simp
{
  let  $?oi = of\text{-int} :: int \Rightarrow 'a$ 
  assume  $wsI: set\ ws \subseteq \mathbb{Z}_v \cap Bounded\text{-vec } (?oi\ Bnd)$  and  $db: is\text{-det-bound } db$ 
  {
    fix  $nv$ 
    assume  $nv \in set\ ?nv$ 
    then obtain  $i$  where  $nv: nv = ?nv ! i$  and  $i: i < ?n$  unfolding  $set\text{-conv-nth}$ 
  }
}
by auto
  from  $order.trans[OF\ units\ unit\text{-vec-int-bounds}]$ 
   $wsI$  have  $set\ (ws @ us) - \{us ! i\} \subseteq \mathbb{Z}_v \cap Bounded\text{-vec } (?oi\ (max\ 1\ Bnd))$ 
using
   $Bounded\text{-vec-mono}[of\ ?oi\ Bnd\ ?oi\ (max\ 1\ Bnd),\ un\ folded\ of\text{-int-le-iff}]$ 
  by auto
  from  $nvi(7)[OF\ i\ db\ this]$   $nv$ 
  have  $nv \in \mathbb{Z}_v \cap Bounded\text{-vec } (?oi\ (db\ (n - 1)\ (max\ 1\ Bnd)))$ 
  by auto
}
}
thus  $set\ ?nv \subseteq \mathbb{Z}_v \cap Bounded\text{-vec } (?oi\ (db\ (n - 1)\ (max\ 1\ Bnd)))$  by auto
}
have  $dist\text{-}nv: distinct\ ?nv$  unfolding  $distinct\text{-conv-nth}\ lnv$ 
proof (intro allI impI)
  fix  $i\ j$ 
  assume  $i: i < ?n$  and  $j: j < ?n$  and  $ij: i \neq j$ 
  with  $dist$  have  $usj: us ! j \in set\ (ws @ us) - \{us ! i\}$ 
  by (simp, auto simp:  $distinct\text{-conv-nth}\ set\text{-conv-nth}$ )
  from  $nvi(4)[OF\ i\ this]$   $nvi(9)[OF\ j]$ 
  show  $?nv ! i \neq ?nv ! j$  by auto
qed
show  $disj: set\ ws \cap set\ ?nv = \{\}$ 

```

```

proof (rule ccontr)
  assume  $\neg$  ?thesis
  then obtain  $w$  where  $w \in \text{set } ws \ w \in \text{set } ?nv$  by auto
  from scalar-0[OF this] this(1) have sq-norm  $w = 0$ 
    by (auto simp: sq-norm-vec-as-cscalar-prod)
  with  $w$   $wsus$  have  $w = 0_v \ n$  by auto
  with vs-zero-lin-dep[OF  $wsus \ lin^\wedge$ ]  $w(1)$  show False by auto
qed
have dist': distinct ( $ws \ @ \ ?nv$ ) using dist disj dist-nv by auto
show lin-indpt-list ( $ws \ @ \ ?nv$ ) unfolding lin-indpt-list-def
proof (intro conjI dist')
  show set: set ( $ws \ @ \ ?nv$ )  $\subseteq$  carrier-vec  $n$  using  $nv \ wsus$  by auto
  hence  $ws$ : set  $ws \subseteq$  carrier-vec  $n$  by auto
  have lin-nv: lin-indpt (set ?nv)
proof
  assume lin-dep (set ?nv)
  from finite-lin-dep[OF finite-set this  $nv$ ]
  obtain  $a \ v$  where comb: lincomb  $a$  (set ?nv) =  $0_v \ n$  and  $vnv$ :  $v \in \text{set } ?nv$ 
and  $av0$ :  $a \ v \neq 0$  by auto
  from  $vnv$ [unfolded set-conv-nth] obtain  $i$  where  $i < ?n$  and  $vi$ :  $v = ?nv$ 
!  $i$  by auto
  define  $b$  where  $b = (\lambda \ w. \ a \ w / a \ v)$ 
  define  $c$  where  $c = (\lambda \ w. \ -1 * b \ w)$ 
  define  $x$  where  $x = \text{lincomb } b \ (\text{set } ?nv - \{v\})$ 
  define  $w$  where  $w = \text{lincomb } c \ (\text{set } ?nv - \{v\})$ 
  have  $w$ :  $w \in \text{carrier-vec } n$  unfolding  $w$ -def using  $nv$  by auto
  have  $x$ :  $x \in \text{carrier-vec } n$  unfolding  $x$ -def using  $nv$  by auto
  from arg-cong[OF comb, of  $\lambda \ x. \ (1 / a \ v) \cdot_v \ x$ ]
  have  $0_v \ n = 1 / a \ v \cdot_v \ \text{lincomb } a \ (\text{set } ?nv)$  by auto
  also have  $\dots = \text{lincomb } b \ (\text{set } ?nv)$ 
    by (subst lincomb-smult[symmetric, OF  $nv$ ], auto simp:  $b$ -def)
  also have  $\dots = b \ v \cdot_v \ v + \text{lincomb } b \ (\text{set } ?nv - \{v\})$ 
    by (subst lincomb-del2[OF -  $nv - vnv$ ], auto)
  also have  $b \ v \cdot_v \ v = v$  using  $av0$  unfolding  $b$ -def by auto
  finally have  $v + \text{lincomb } b \ (\text{set } ?nv - \{v\}) - \text{lincomb } b \ (\text{set } ?nv - \{v\}) =$ 
 $0_v \ n - \text{lincomb } b \ (\text{set } ?nv - \{v\})$  (is ?l = ?r) by simp
  also have ?l =  $v$ 
    by (rule add-diff-cancel-right-vec, insert  $vnv \ nv$ , auto)
  also have ?r =  $w$  unfolding  $w$ -def  $c$ -def
    by (subst lincomb-smult, unfold  $x$ -def[symmetric], insert  $nv \ x$ , auto)
  finally have  $vw$ :  $v = w$  .
  have  $u$ :  $us ! i \in \text{carrier-vec } n$  using  $i \ wsus$  by auto
  have  $nv'$ : set ?nv - {?nv !  $i$ }  $\subseteq$  carrier-vec  $n$  using  $nv$  by auto
  have ?nv !  $i \cdot us ! i = 0$ 
    unfolding  $vi$ [symmetric]  $vw$  unfolding  $w$ -def  $vi$ 
    unfolding lincomb-scalar-prod-left[OF  $nv' \ u$ ]
proof (rule sum.neutral, intro ballI)
  fix  $x$ 
  assume  $x \in \text{set } ?nv - \{?nv ! i\}$ 

```

then obtain j where $j: j < ?n$ and $x: x = ?nv ! j$ and $ij: i \neq j$ unfolding
set-conv-nth **by auto**
from $dist[simplified] ij i j$ have $us ! i \neq us ! j$ unfolding $distinct-conv-nth$
by auto
with i have $us ! i \in set (ws @ us) - \{us ! j\}$ by auto
from $nvi(3-4)[OF j this]$
show $c x * (x \cdot us ! i) = 0$ unfolding x by auto
qed
with $nvi(9)[OF i]$ show $False ..$
qed
from $subset-li-is-li[OF lin^]$ have $lin-indpt (set ws)$ by auto
from $ortho-lin-indpt[OF nv ws scalar-0 lin-nv this]$
have $lin-indpt (set ?nv \cup set ws)$.
also have $set ?nv \cup set ws = set (ws @ ?nv)$ by auto
finally show $lin-indpt (set (ws @ ?nv))$.
qed
qed

definition $pos-norm-vec :: 'a vec set \Rightarrow 'a vec \Rightarrow 'a vec$ where
 $pos-norm-vec D x = (let c' = normal-vector D;$
 $c = (if c' \cdot x > 0 then c' else -c') in c)$

lemma $pos-norm-vec$:

assumes $D: D \subseteq carrier-vec n$ and $fin: finite D$ and $lin: lin-indpt D$
and $card: Suc (card D) = n$
and $c-def: c = pos-norm-vec D x$
shows $c \in carrier-vec n$ $span D = \{ x. x \in carrier-vec n \wedge x \cdot c = 0 \}$
 $x \notin span D \Longrightarrow x \in carrier-vec n \Longrightarrow c \cdot x > 0$
 $c \in \{normal-vector D, -normal-vector D\}$
proof -
have $n: normal-vector D \in carrier-vec n$ using $normal-vector$ $assms$ by auto
show $cnorm: c \in \{normal-vector D, -normal-vector D\}$ unfolding $c-def$ $pos-norm-vec-def$
Let-def **by auto**
then show $c: c \in carrier-vec n$ using $assms$ $normal-vector$ by auto
have $span D = \{ x. x \in carrier-vec n \wedge x \cdot normal-vector D = 0 \}$
using $normal-vector-span[OF card D fin lin]$ by auto
also have $\dots = \{ x. x \in carrier-vec n \wedge x \cdot c = 0 \}$ using $cnorm c$ by auto
finally show $span-char: span D = \{ x. x \in carrier-vec n \wedge x \cdot c = 0 \}$ by auto
{
assume $x: x \notin span D$ $x \in carrier-vec n$
hence $c \cdot x \neq 0$ using $comm-scalar-prod[OF c]$ unfolding $span-char$ by auto
hence $normal-vector D \cdot x \neq 0$ using $cnorm n x$ by auto
with x have $b: \neg (normal-vector D \cdot x > 0) \Longrightarrow (-normal-vector D) \cdot x > 0$
using $assms n$ by auto
then show $c \cdot x > 0$ unfolding $c-def$ $pos-norm-vec-def$ $Let-def$
by ($auto$ $split: if-splits$)
}
qed

end

end

10 Dimension of Spans

We define the notion of dimension of a span of vectors and prove some natural results about them. The definition is made as a function, so that no interpretation of locales like subspace is required.

theory *Dim-Span*

imports *Missing-VS-Connect*

begin

context *vec-space*

begin

definition *dim-span* $W = \text{Max} (\text{card} \{ V. V \subseteq \text{carrier-vec } n \wedge V \subseteq \text{span } W \wedge \text{lin-indpt } V \})$

lemma fixes $V W :: 'a \text{ vec set}$

shows

card-le-dim-span:

$V \subseteq \text{carrier-vec } n \implies V \subseteq \text{span } W \implies \text{lin-indpt } V \implies \text{card } V \leq \text{dim-span } W$

and

card-eq-dim-span-imp-same-span:

$V \subseteq \text{carrier-vec } n \implies V \subseteq \text{span } W \implies \text{lin-indpt } V \implies \text{card } V = \text{dim-span } W$

$W \implies \text{span } V = \text{span } W$ **and**

same-span-imp-card-eq-dim-span:

$V \subseteq \text{carrier-vec } n \implies W \subseteq \text{carrier-vec } n \implies \text{span } V = \text{span } W \implies \text{lin-indpt } V \implies \text{card } V = \text{dim-span } W$ **and**

dim-span-cong:

$\text{span } V = \text{span } W \implies \text{dim-span } V = \text{dim-span } W$ **and**

ex-basis-span:

$V \subseteq \text{carrier-vec } n \implies \exists W. W \subseteq \text{carrier-vec } n \wedge \text{lin-indpt } W \wedge \text{span } V = \text{span } W \wedge \text{dim-span } V = \text{card } W$

proof –

show *cong*: $\bigwedge V W. \text{span } V = \text{span } W \implies \text{dim-span } V = \text{dim-span } W$ **unfolding** *dim-span-def* **by** *auto*

{

fix $W :: 'a \text{ vec set}$

let $?M = \{ V. V \subseteq \text{carrier-vec } n \wedge V \subseteq \text{span } W \wedge \text{lin-indpt } V \}$

have $\text{card} \{ ?M \subseteq \{ 0 .. n \}$

proof

fix k

assume $k \in \text{card} \{ ?M$

then obtain V **where** $V: V \subseteq \text{carrier-vec } n \wedge V \subseteq \text{span } W \wedge \text{lin-indpt } V$

and $k: k = \text{card } V$

by *auto*

from V **have** $\text{card } V \leq n$ **using** *dim-is-n li-le-dim* **by** *auto*

```

    with k show k ∈ {0 .. n} by auto
  qed
  from finite-subset[OF this]
  have fin: finite (card ' ?M) by auto
  have {} ∈ ?M by (auto simp: span-empty span-zero)
  from imageI[OF this, of card]
  have 0 ∈ card ' ?M by auto
  hence Mempty: card ' ?M ≠ {} by auto
  from Max-ge[OF fin, folded dim-span-def]
  show  $\bigwedge V :: 'a \text{ vec set. } V \subseteq \text{carrier-vec } n \implies V \subseteq \text{span } W \implies \text{lin-indpt } V$ 
 $\implies \text{card } V \leq \text{dim-span } W$ 
    by auto
  note this fin Mempty
} note part1 = this
{
  fix V W :: 'a vec set
  assume W: W ⊆ carrier-vec n
  and VsW: V ⊆ span W and linV: lin-indpt V and card: card V = dim-span
W
  from W VsW have V: V ⊆ carrier-vec n using span-mem[OF W] by auto
  from Max-in[OF part1(2,3), folded dim-span-def, of W]
  obtain WW where WW: WW ⊆ carrier-vec n WW ⊆ span W lin-indpt WW
    and id: dim-span W = card WW by auto
  show span V = span W
  proof (rule ccontr)
    from VsW V W have sub: span V ⊆ span W using span-subsetI by metis
    assume span V ≠ span W
    with sub obtain w where wW: w ∈ span W and wsV: w ∉ span V by auto
    from wW W have w: w ∈ carrier-vec n by auto
    from linV V have finV: finite V using fin-dim fin-dim-li-fin by blast
    from wsV span-mem[OF V, of w] have wV: w ∉ V by auto
    let ?X = insert w V
    have card ?X = Suc (card V) using wV finV by simp
    hence gt: card ?X > dim-span W unfolding card by simp
    have linX: lin-indpt ?X using lin-dep-iff-in-span[OF V linV w wV] wsV by
auto
    have XW: ?X ⊆ span W using wW VsW by auto
    from part1(1)[OF - XW linX] w V have card ?X ≤ dim-span W by auto
    with gt show False by auto
  qed
} note card-dim-span = this
{
  fix V :: 'a vec set
  assume V: V ⊆ carrier-vec n
  from Max-in[OF part1(2,3), folded dim-span-def, of V]
  obtain W where W: W ⊆ carrier-vec n W ⊆ span V lin-indpt W
    and idW: card W = dim-span V by auto
  show  $\exists W. W \subseteq \text{carrier-vec } n \wedge \text{lin-indpt } W \wedge \text{span } V = \text{span } W \wedge \text{dim-span}$ 
 $V = \text{card } W$ 

```

```

proof (intro exI[of - W] conjI W idW[symmetric])
  from card-dim-span[OF V(1) W(2-3) idW] show span V = span W by
auto
  qed
}
{
  fix V W
  assume V: V  $\subseteq$  carrier-vec n
  and W: W  $\subseteq$  carrier-vec n
  and span: span V = span W
  and lin: lin-indpt V
  from Max-in[OF part1(2,3), folded dim-span-def, of W]
  obtain WW where WW: WW  $\subseteq$  carrier-vec n WW  $\subseteq$  span W lin-indpt WW
  and idWW: card WW = dim-span W by auto
  from card-dim-span[OF W WW(2-3) idWW] span
  have spanWW: span WW = span V by auto
  from span have V  $\subseteq$  span W using span-mem[OF V] by auto
  from part1(1)[OF V this lin] have VW: card V  $\leq$  dim-span W .
  have finWW: finite WW using WW by (simp add: fin-dim-li-fin)
  have finV: finite V using lin V by (simp add: fin-dim-li-fin)
  from replacement[OF finWW finV V WW(3) WW(2)[folded span], unfolded
idWW]
  obtain C :: 'a vec set
  where le: int (card C)  $\leq$  int (card V) - int (dim-span W) by auto
  from le have int (dim-span W) + int (card C)  $\leq$  int (card V) by linarith
  hence dim-span W + card C  $\leq$  card V by linarith
  with VW show card V = dim-span W by auto
}
qed

```

lemma dim-span-le-n: **assumes** W: W \subseteq carrier-vec n **shows** dim-span W \leq n
proof –
from ex-basis-span[OF W] **obtain** V **where**
 V: V \subseteq carrier-vec n
and lin: lin-indpt V
and dim: dim-span W = card V
by auto
show ?thesis **unfolding** dim **using** lin V
using dim-is-n li-le-dim **by** auto
qed

lemma dim-span-insert: **assumes** W: W \subseteq carrier-vec n
and v: v \in carrier-vec n **and** vs: v \notin span W
shows dim-span (insert v W) = Suc (dim-span W)
proof –
from ex-basis-span[OF W] **obtain** V **where**
 V: V \subseteq carrier-vec n
and lin: lin-indpt V
and span: span W = span V

```

    and dim: dim-span W = card V
    by auto
  from V vs[unfolded span] have vV: v ∉ V using span-mem[OF V] by blast
  from lin-dep-iff-in-span[OF V lin v vV] vs span
  have lin': lin-indpt (insert v V) by auto
  have finV: finite V using lin V using fin-dim fin-dim-li-fin by blast
  have card (insert v V) = Suc (card V) using finV vV by auto
  hence cvV: card (insert v V) = Suc (dim-span W) using dim by auto
  have span (insert v V) = span (insert v W)
    using span V W v by (metis bot-least insert-subset insert-union span-union-is-sum)
  from same-span-imp-card-eq-dim-span[OF - - this lin'] cvV v V W
  show ?thesis by auto
qed
end
end

```

11 The Fundamental Theorem of Linear Inequalities

The theorem states that for a given set of vectors A and vector b , either b is in the cone of a linear independent subset of A , or there is a hyperplane that contains $\text{span}(A, b) - 1$ linearly independent vectors of A that separates A from b . We prove this theorem and derive some consequences, e.g., Caratheodory's theorem that b is the cone of A iff b is in the cone of a linear independent subset of A .

theory *Fundamental-Theorem-Linear-Inequalities*

imports

Cone

Normal-Vector

Dim-Span

begin

context *gram-schmidt*

begin

The mentions equivances A-D are:

- A: b is in the cone of vectors A ,
- B: b is in the cone of a subset of linear independent of vectors A ,
- C: there is no separating hyperplane of b and the vectors A , which contains \dim many linear independent vectors of A
- D: there is no separating hyperplane of b and the vectors A

lemma *fundamental-theorem-of-linear-inequalities-A-imp-D*:

assumes $A: A \subseteq \text{carrier-vec } n$
and $\text{fin}: \text{finite } A$
and $b: b \in \text{cone } A$
shows $\nexists c. c \in \text{carrier-vec } n \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0$
proof
assume $\exists c. c \in \text{carrier-vec } n \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0$
then obtain c **where** $c: c \in \text{carrier-vec } n$
and $\text{ai}: \bigwedge ai. ai \in A \implies c \cdot ai \geq 0$
and $\text{cb}: c \cdot b < 0$ **by** *auto*
from $b[\text{unfolded cone-def nonneg-lincomb-def finite-cone-def}]$
obtain $l \text{ AA}$ **where** $\text{bc}: b = \text{lincomb } l \text{ AA}$ **and** $l: l \text{ ' AA} \subseteq \{x. x \geq 0\}$ **and** $\text{AA}: \text{AA} \subseteq A$ **by** *auto*
from $\text{cone-carrier}[OF A] b$ **have** $b: b \in \text{carrier-vec } n$ **by** *auto*
have $0 \leq (\sum ai \in \text{AA}. l \text{ ai} * (c \cdot ai))$
by (*intro sum-nonneg mult-nonneg-nonneg, insert l ai AA, auto*)
also have $\dots = (\sum ai \in \text{AA}. l \text{ ai} * (ai \cdot c))$
by (*rule sum.cong, insert c A AA comm-scalar-prod, force+*)
also have $\dots = (\sum ai \in \text{AA}. ((l \text{ ai} \cdot_v ai) \cdot c))$
by (*rule sum.cong, insert smult-scalar-prod-distrib c A AA, auto*)
also have $\dots = b \cdot c$ **unfolding** bc *lincomb-def*
by (*subst finsum-scalar-prod-sum[symmetric], insert c A AA, auto*)
also have $\dots = c \cdot b$ **using** *comm-scalar-prod b c* **by** *auto*
also have $\dots < 0$ **by** *fact*
finally show *False* **by** *simp*
qed

The difficult direction is that C implies B. To this end we follow the proof that at least one of B and the negation of C is satisfied.

context

fixes $a :: \text{nat} \Rightarrow 'a \text{ vec}$
and $b :: 'a \text{ vec}$
and $m :: \text{nat}$
assumes $a: a \text{ ' } \{0 ..< m\} \subseteq \text{carrier-vec } n$
and $\text{inj-a}: \text{inj-on } a \text{ } \{0 ..< m\}$
and $b: b \in \text{carrier-vec } n$
and $\text{full-span}: \text{span } (a \text{ ' } \{0 ..< m\}) = \text{carrier-vec } n$
begin

private definition $\text{goal} = ((\exists I. I \subseteq \{0 ..< m\} \wedge \text{card } (a \text{ ' } I) = n \wedge \text{lin-indpt } (a \text{ ' } I) \wedge b \in \text{finite-cone } (a \text{ ' } I))$
 $\vee (\exists c \text{ I}. I \subseteq \{0 ..< m\} \wedge c \in \{\text{normal-vector } (a \text{ ' } I), - \text{normal-vector } (a \text{ ' } I)\}$
 $\wedge \text{Suc } (\text{card } (a \text{ ' } I)) = n$
 $\wedge \text{lin-indpt } (a \text{ ' } I) \wedge (\forall i < m. c \cdot a \text{ } i \geq 0) \wedge c \cdot b < 0))$

private lemma $\text{card-a-I}[simp]: I \subseteq \{0 ..< m\} \implies \text{card } (a \text{ ' } I) = \text{card } I$
by (*smt inj-a card-image inj-on-image-eq-iff subset-image-inj subset-refl subset-trans*)

private lemma $\text{in-a-I}[simp]: I \subseteq \{0 ..< m\} \implies i < m \implies (a \text{ } i \in a \text{ ' } I) = (i \in$

I)

using *inj-a*

by (*meson atLeastLessThan-iff image-eqI inj-on-image-mem-iff zero-le*)

private definition *valid-I* = { *I*. *card I* = *n* ∧ *lin-indpt* (*a* ' *I*) ∧ *I* ⊆ {0 ..< *m*}}

private definition *cond* where *cond I I' l c h k* ≡

b = *lincomb l* (*a* ' *I*) ∧

h ∈ *I* ∧ *l* (*a h*) < 0 ∧ (∀ *h'*. *h'* ∈ *I* → *h'* < *h* → *l* (*a h'*) ≥ 0) ∧

c ∈ *carrier-vec n* ∧ *span* (*a* ' (*I* - {*h*})) = { *x*. *x* ∈ *carrier-vec n* ∧ *c* · *x* = 0 }

∧ *c* · *b* < 0 ∧

k < *m* ∧ *c* · *a k* < 0 ∧ (∀ *k'*. *k'* < *k* → *c* · *a k'* ≥ 0) ∧

I' = *insert k* (*I* - {*h*})

private definition *step-rel* = *Restr* { (*I''*, *I*). ∃ *l c h k*. *cond I I'' l c h k* } *valid-I*

private lemma *finite-step-rel*: *finite step-rel*

proof (*rule finite-subset*)

show *step-rel* ⊆ (*Pow* {0 ..< *m*} × *Pow* {0 ..< *m*}) **unfolding** *step-rel-def*

valid-I-def **by** *auto*

qed *auto*

private lemma *acyclic-imp-goal*: *acyclic step-rel* ⇒ *goal*

proof (*rule ccontr*)

assume *ngoal*: ¬ *goal*

{

fix *I*

assume *I*: *I* ∈ *valid-I*

hence *Im*: *I* ⊆ {0..<*m*} **and**

lin: *lin-indpt* (*a* ' *I*) **and**

cardI: *card I* = *n*

by (*auto simp: valid-I-def*)

let *?D* = (*a* ' *I*)

have *finD*: *finite ?D* **using** *Im infinite-super* **by** *blast*

have *carrD*: *?D* ⊆ *carrier-vec n* **using** *a Im* **by** *auto*

have *cardD*: *card ?D* = *n* **using** *cardI Im* **by** *simp*

have *spanD*: *span ?D* = *carrier-vec n*

by (*intro span-carrier-lin-indpt-card-n lin cardD carrD*)

obtain *lamb* where *b-is-lincomb*: *b* = *lincomb lamb* (*a* ' *I*)

using *finite-in-span[OF fin carrD, of b]* **using** *spanD b carrD fin-dim lin* **by**

auto

define *h* where *h* = (*LEAST h*. *h* ∈ *I* ∧ *lamb* (*a h*) < 0)

have ∃ *I'*. (*I'*, *I*) ∈ *step-rel*

proof (*cases* ∀ *i* ∈ *I* . *lamb* (*a i*) ≥ 0)

case *cond1-T*: *True*

have *goal* **unfolding** *goal-def*

by (*intro disjI1 exI[of - I] conjI lin cardI*

lincomb-in-finite-cone[OF b-is-lincomb finD - carrD], insert cardI Im

```

cond1-T, auto)
  with ngoal show ?thesis by auto
next
  case cond1-F: False
  hence  $\exists h. h \in I \wedge \text{lamb } (a h) < 0$  by fastforce
  from LeastI-ex[OF this, folded h-def] have  $h: h \in I \text{ lamb } (a h) < 0$  by auto
  from not-less-Least[of -  $\lambda h. h \in I \wedge \text{lamb } (a h) < 0$ , folded h-def]
  have h-least:  $\forall k. k \in I \longrightarrow k < h \longrightarrow \text{lamb } (a k) \geq 0$  by fastforce
  obtain I' where I'-def:  $I' = I - \{h\}$  by auto
  obtain c where c-def:  $c = \text{pos-norm-vec } (a \text{ ' } I') (a h)$  by auto
  let ?D' = a ' I'
  have I'm:  $I' \subseteq \{0..<m\}$  using Im I'-def by auto
  have carrD':  $?D' \subseteq \text{carrier-vec } n$  using a Im I'-def by auto
  have finD': finite (?D') using Im I'-def subset-eq-atLeast0-lessThan-finite by
auto
  have D'subs:  $?D' \subseteq ?D$  using I'-def by auto
  have linD': lin-indpt (?D') using lin I'-def Im D'subs subset-li-is-li by auto
  have D'strictsubs:  $?D = ?D' \cup \{a h\}$  using h I'-def by auto
  have h-nin-I:  $h \notin I'$  using h I'-def by auto
  have ah-nin-D':  $a h \notin ?D'$  using h inj-a Im h-nin-I by (subst in-a-I, auto
simp: I'-def)
  have cardD':  $\text{Suc } (\text{card } (?D')) = n$  using cardD ah-nin-D' D'strictsubs finD'
by simp
  have ah-carr:  $a h \in \text{carrier-vec } n$  using h a Im by auto
  note pnv = pos-norm-vec[OF carrD' finD' linD' cardD' c-def]
  have ah-nin-span:  $a h \notin \text{span } ?D'$ 
  using D'strictsubs lin-dep-iff-in-span[OF carrD' linD' ah-carr ah-nin-D'] lin
by auto
  have cah-ge-zero:  $c \cdot a h > 0$  and  $c \in \text{carrier-vec } n$ 
  and cnorm:  $\text{span } ?D' = \{x \in \text{carrier-vec } n. x \cdot c = 0\}$ 
  using ah-carr ah-nin-span pnv by auto
  have ccarr:  $c \in \text{carrier-vec } n$  by fact
  have  $b \cdot c = \text{lincomb } \text{lamb } (a \text{ ' } I) \cdot c$  using b-is-lincomb by auto
  also have  $\dots = (\sum w \in ?D. \text{lamb } w * (w \cdot c))$ 
  using lincomb-scalar-prod-left[OF carrD, of c lamb] pos-norm-vec ccarr by
blast
  also have  $\dots = \text{lamb } (a h) * (a h \cdot c) + (\sum w \in ?D'. \text{lamb } w * (w \cdot c))$ 
  using sum.insert[OF finD' ah-nin-D', of lamb] D'strictsubs ah-nin-D' finD'
by auto
  also have  $(\sum w \in ?D'. \text{lamb } w * (w \cdot c)) = 0$ 
  apply (rule sum.neutral)
  using span-mem[OF carrD', unfolded cnorm] by simp
  also have  $\text{lamb } (a h) * (a h \cdot c) + 0 < 0$ 
  using cah-ge-zero h(2) comm-scalar-prod[OF ah-carr ccarr]
  by (auto intro: mult-neg-pos)
  finally have cb-le-zero:  $c \cdot b < 0$  using comm-scalar-prod[OF b ccarr] by
auto

show ?thesis

```

```

proof (cases  $\forall i < m . c \cdot a i \geq 0$ )
  case cond2-T: True
  have goal
    unfolding goal-def
    by (intro disjI2 exI[of - c] exI[of - I'] conjI cb-le-zero linD' cond2-T cardD'
I'm pnv(4))
    with ngoal show ?thesis by auto
  next
  case cond2-F: False
  define k where k = (LEAST k. k < m  $\wedge$  c  $\cdot$  a k < 0)
  let ?I'' = insert k I'
  show ?thesis unfolding step-rel-def
  proof (intro exI[of - ?I''], standard, unfold mem-Collect-eq split, intro exI)
    from LeastI-ex[OF ]
    have  $\exists k. k < m \wedge c \cdot a k < 0$  using cond2-F by fastforce
    from LeastI-ex[OF this, folded k-def] have k: k < m c  $\cdot$  a k < 0 by auto
    show cond I ?I'' lamb c h k unfolding cond-def I'-def[symmetric] cnorm
    proof(intro conjI cb-le-zero b-is-lincomb h ccarr h-least refl k)
      show  $\{x \in \text{carrier-vec } n. x \cdot c = 0\} = \{x \in \text{carrier-vec } n. c \cdot x = 0\}$ 
        using comm-scalar-prod[OF ccarr] by auto
      from not-less-Least[of -  $\lambda k. k < m \wedge c \cdot a k < 0$ , folded k-def]
      have  $\forall k' < k . k' > m \vee c \cdot a k' \geq 0$  using k(1) less-trans not-less by
blast
        then show  $\forall k' < k . c \cdot a k' \geq 0$  using k(1) by auto
    qed

    have ?I''  $\in$  valid-I unfolding valid-I-def
    proof(standard, intro conjI)
      from k a have ak-carr: a k  $\in$  carrier-vec n by auto
      have ak-nin-span: a k  $\notin$  span ?D' using k(2) cnorm comm-scalar-prod[OF
ak-carr ccarr] by auto
      hence ak-nin-D': a k  $\notin$  ?D' using span-mem[OF carrD'] by auto
      from lin-dep-iff-in-span[OF carrD' linD' ak-carr ak-nin-D']
      show lin-indpt (a ' ?I'') using ak-nin-span by auto
      show ?I''  $\subseteq$  {0..m} using I'm k by auto
      show card (insert k I') = n using cardD' ak-nin-D' finD'
      by (metis <insert k I'  $\subseteq$  {0..m> card-a-I card-insert-disjoint
image-insert)
    qed
    then show (?I'', I)  $\in$  valid-I  $\times$  valid-I using I by auto

    qed
  qed
  qed
} note step = this
{
from exists-lin-indpt-subset[OF a, unfolded full-span]
obtain A where lin: lin-indpt A and span: span A = carrier-vec n and Am:
A  $\subseteq$  a ' {0 ..<i>m} by auto

```

```

from  $Am$  have  $A: A \subseteq \text{carrier-vec } n$  by auto
from  $\text{lin span } A$  have  $\text{card}: \text{card } A = n$ 
  using basis-def dim-basis dim-is-n fin-dim-li-fin by auto
from  $A$   $Am$  obtain  $I$  where  $A: A = a \text{ ' } I$  and  $I: I \subseteq \{0 \dots m\}$  by (metis
subset-imageE)
  have  $I \in \text{valid-}I$  using  $I$   $\text{card}$   $\text{lin}$  unfolding  $\text{valid-}I\text{-def } A$  by auto
  hence  $\exists I. I \in \text{valid-}I$  ..
}
note  $\text{init} = \text{this}$ 
have  $\text{step-valid}: (I', I) \in \text{step-rel} \implies I' \in \text{valid-}I$  for  $I I'$  unfolding  $\text{step-rel-def}$ 
by auto
have  $\neg (\text{wf } \text{step-rel})$ 
proof
  from  $\text{init}$  obtain  $I$  where  $I: I \in \text{valid-}I$  by auto
  assume  $\text{wf } \text{step-rel}$ 
  from  $\text{this}[\text{unfolded } \text{wf-eq-minimal}, \text{rule-format}, \text{OF } I]$   $\text{step } \text{step-valid}$  show False
by blast
qed
with  $\text{wf-iff-acyclic-if-finite}[\text{OF } \text{finite-step-rel}]$ 
have  $\neg \text{acyclic } \text{step-rel}$  by auto
thus  $\text{acyclic } \text{step-rel} \implies \text{False}$  by auto
qed

```

```

private lemma  $\text{acyclic-step-rel}: \text{acyclic } \text{step-rel}$ 
proof (rule ccontr)
  assume  $\neg ?\text{thesis}$ 
  hence  $\neg \text{acyclic } (\text{step-rel}^{-1})$  by auto

```

```

then obtain  $I$  where  $(I, I) \in (\text{step-rel}^{-1})^+ \text{ unfolding } \text{acyclic-def}$  by blast
from  $\text{this}[\text{unfolded } \text{trancl-power}]$ 
obtain  $\text{len}$  where  $(I, I) \in (\text{step-rel}^{-1})^{\sim \text{len}}$  and  $\text{len0}: \text{len} > 0$  by blast

```

```

from  $\text{this}[\text{unfolded } \text{relpow-fun-conv}]$  obtain  $Is$  where
   $\text{stepsIs}: \bigwedge i. i < \text{len} \implies (Is (\text{Suc } i), Is i) \in \text{step-rel}$ 
  and  $IsI: Is 0 = I$   $Is \text{len} = I$  by auto
{
  fix  $i$ 
  assume  $i \leq \text{len}$  hence  $i - 1 < \text{len}$  using  $\text{len0}$  by auto
  from  $\text{stepsIs}[\text{unfolded } \text{step-rel-def}, \text{OF } \text{this}]$ 
  have  $Is i \in \text{valid-}I$  by (cases i, auto)
} note  $Is\text{-valid} = \text{this}$ 
from  $\text{stepsIs}[\text{unfolded } \text{step-rel-def}]$ 
have  $\forall i. \exists l c h k. i < \text{len} \longrightarrow \text{cond } (Is i) (Is (\text{Suc } i)) l c h k$  by auto

```

```

from  $\text{choice}[\text{OF } \text{this}]$  obtain  $ls$  where  $\forall i. \exists c h k. i < \text{len} \longrightarrow \text{cond } (Is i) (Is$ 
 $(\text{Suc } i)) (ls i) c h k$  by auto
from  $\text{choice}[\text{OF } \text{this}]$  obtain  $cs$  where  $\forall i. \exists h k. i < \text{len} \longrightarrow \text{cond } (Is i) (Is$ 
 $(\text{Suc } i)) (ls i) (cs i) h k$  by auto

```

from *choice*[*OF this*] **obtain** *hs* **where** $\forall i. \exists k. i < \text{len} \implies \text{cond } (Is\ i) (Is\ (\text{Suc } i)) (ls\ i) (cs\ i) (hs\ i) k$ **by** *auto*
from *choice*[*OF this*] **obtain** *ks* **where**
cond: $\bigwedge i. i < \text{len} \implies \text{cond } (Is\ i) (Is\ (\text{Suc } i)) (ls\ i) (cs\ i) (hs\ i) (ks\ i)$ **by** *auto*

let $?R = \{hs\ i \mid i. i < \text{len}\}$
define *r* **where** $r = \text{Max } ?R$
from *cond*[*OF len0*] **have** $hs\ 0 \in I$ **using** *IsI* **unfolding** *cond-def* **by** *auto*
hence $R0: hs\ 0 \in ?R$ **using** *len0* **by** *auto*
have *finR*: *finite* $?R$ **by** *auto*
from *Max-in*[*OF finR*] $R0$
have *rR*: $r \in ?R$ **unfolding** *r-def*[*symmetric*] **by** *auto*
then obtain *p* **where** $rp: r = hs\ p$ **and** $p: p < \text{len}$ **by** *auto*
from *Max-ge*[*OF finR, folded r-def*]
have *rLarge*: $i < \text{len} \implies hs\ i \leq r$ **for** *i* **by** *auto*
have *exq*: $\exists q. ks\ q = r \wedge q < \text{len}$
proof (*rule ccontr*)
 assume *neg*: $\neg ?thesis$
 show *False*
 proof(*cases r* $\in I$)
 case *True*
 have $1: j \in \{\text{Suc } p.. \text{len}\} \implies r \notin Is\ j$ **for** *j*
 proof(*induction j* *rule: less-induct*)
 case (*less j*)
 from *less*(2) **have** *j-bounds*: $j = \text{Suc } p \vee j > \text{Suc } p$ **by** *auto*
 from *less*(2) **have** *j-len*: $j \leq \text{len}$ **by** *auto*
 have *pj-cond*: $j = \text{Suc } p \implies \text{cond } (Is\ p) (Is\ j) (ls\ p) (cs\ p) (hs\ p) (ks\ p)$
using *cond p* **by** *blast*
 have *r-neq-ksp*: $r \neq ks\ p$ **using** *p neg* **by** *auto*
 have $j = \text{Suc } p \implies Is\ j = \text{insert } (ks\ p) (Is\ p - \{r\})$
 using *rp cond pj-cond cond-def*[*of Is p Is j - - r*] **by** *blast*
 hence *c1*: $j = \text{Suc } p \implies r \notin Is\ j$ **using** *r-neq-ksp* **by** *simp*
 have *IH*: $\bigwedge t. t < j \implies t \in \{\text{Suc } p.. \text{len}\} \implies r \notin Is\ t$ **by** *fact*
 have *r-neq-kspj*: $j > \text{Suc } p \wedge j \leq \text{len} \implies r \neq ks\ (j-1)$ **using** *j-len neg IH*
by *auto*
 have *jsucj-cond*: $j > \text{Suc } p \wedge j \leq \text{len} \implies Is\ j = \text{insert } (ks\ (j-1)) (Is\ (j-1) - \{hs\ (j-1)\})$
 using *cond-def*[*of Is (j-1) Is j*] *cond*
 by (*metis (no-types, lifting) Suc-less-eq2 diff-Suc-1 le-simps*(3))
 hence $j > \text{Suc } p \wedge j \leq \text{len} \implies r \notin \text{insert } (ks\ (j-1)) (Is\ (j-1))$
 using *IH r-neq-kspj* **by** *auto*
 hence $j > \text{Suc } p \wedge j \leq \text{len} \implies r \notin Is\ j$ **using** *jsucj-cond* **by** *simp*
 then show *?case* **using** *j-bounds j-len c1* **by** *blast*
 qed
 then show *?thesis* **using** *neg IsI*(2) *True p* **by** *auto*
 next
 case *False*
 have $2: j \in \{0..p\} \implies r \notin Is\ j$ **for** *j*
 proof(*induction j* *rule: less-induct*)

```

    case(less j)
    from less(2) have j-bound: j ≤ p by auto
    have r-nin-Is0: r ∉ Is 0 using IsI(1) False by simp
    have IH: ∧t. t < j ∧ t ∈ {0..p} ⇒ r ∉ Is t using less.IH by blast
    have j-neg-ksjpred: j > 0 ⇒ r ≠ ks (j - 1) using neg j-bound p by auto
    have Is-jpredj: j > 0 ⇒ Is j = insert (ks (j-1)) (Is (j-1) - {hs (j-1)})
      using cond-def[of Is (j-1) Is j - - hs (j-1) ks (j-1)] cond
      by (metis (full-types) One-nat-def Suc-pred diff-le-self j-bound le-less-trans
    p)
    have j > 0 ⇒ r ∉ insert (ks (j-1)) (Is (j-1))
      using j-neg-ksjpred IH j-bound by fastforce
    hence j > 0 ⇒ r ∉ Is j using Is-jpredj by blast
    then show ?case using j-bound r-nin-Is0 by blast
  qed
  have 3: r ∈ Is p using rp cond cond-def p by blast
  then show ?thesis using 2 3 by auto
  qed
  then obtain q where q1: ks q = r and q-len: q < len by blast

  {
    fix t i1 i2
    assume i1 < len i2 < len t < m
    assume t ∈ Is i1 t ∉ Is i2
    have ∃j < len. t = hs j
    proof (rule ccontr)
      assume ¬ ?thesis
      hence hst: ∧j. j < len ⇒ hs j ≠ t by auto
      have main: t ∉ Is (i + k) ⇒ i + k ≤ len ⇒ t ∉ Is k for i k
      proof (induct i)
        case (Suc i)
          hence i: i + k < len by auto
          from cond[OF this, unfolded cond-def]
          have Is (Suc i + k) = insert (ks (i + k)) (Is (i + k) - {hs (i + k)}) by
    auto
          from Suc(2)[unfolded this] hst[OF i] have t ∉ Is (i + k) by auto
          from Suc(1)[OF this] i show ?case by auto
      qed auto
      from main[of i2 0] ⟨i2 < len⟩ ⟨t ∉ Is i2⟩ have t ∉ Is 0 by auto
      with IsI have t ∉ Is len by auto
      with main[of len - i1 i1] ⟨i1 < len⟩ have t ∉ Is i1 by auto
      with ⟨t ∈ Is i1⟩ show False by blast
    qed
  } note innotin = this

  {
    fix i
    assume i: i ∈ {Suc r..<m}
    {

```

```

assume  $i$ -in- $Isp$ :  $i \in Is\ p$ 
have  $i \in Is\ q$ 
proof (rule ccontr)
  have  $i$ -range:  $i < m$  using  $i$  by simp
  assume  $\neg$  ?thesis
  then have  $ex$ :  $\exists j < len. i = hs\ j$ 
    using innotin[OF p q-len i-range i-in-Isp] by simp
  then obtain  $j$  where  $j$ - $hs$ :  $i = hs\ j$  by blast
  have  $i > r$  using  $i$  by simp
  then show False using  $j$ - $hs\ p\ rLarge\ ex$  by force
qed
}
hence  $(i \in Is\ p) \implies (i \in Is\ q)$  by blast
} note  $bla = this$ 
have  $blin$ :  $b = lincomb\ (ls\ p)\ (a\ ' (Is\ p))$  using cond-def p cond by blast
have  $carrDp$ :  $(a\ ' (Is\ p)) \subseteq carrier\ vec\ n$  using Is-valid valid-I-def a p
  by (smt image-subset-iff less-imp-le-nat mem-Collect-eq subsetD)
have  $carrcq$ :  $cs\ q \in carrier\ vec\ n$  using cond cond-def q-len by simp
have  $ineq1$ :  $(cs\ q) \cdot b < 0$  using cond-def q-len cond by blast
let  $?Isp\ lt\ r = \{x \in Is\ p . x < r\}$ 
let  $?Isp\ gt\ r = \{x \in Is\ p . x > r\}$ 
have  $Is\ disj$ :  $?Isp\ lt\ r \cap ?Isp\ gt\ r = \{\}$  using Is-valid by auto
have  $?Isp\ lt\ r \subseteq Is\ p$  by simp
hence  $Isp\ lt\ 0m$ :  $?Isp\ lt\ r \subseteq \{0..<m\}$  using valid-I-def Is-valid p less-imp-le-nat
by blast
  have  $?Isp\ gt\ r \subseteq Is\ p$  by simp
hence  $Isp\ gt\ 0m$ :  $?Isp\ gt\ r \subseteq \{0..<m\}$  using valid-I-def Is-valid p less-imp-le-nat
by blast
let  $?Dp\ lt = a\ ' ?Isp\ lt\ r$ 
let  $?Dp\ ge = a\ ' ?Isp\ gt\ r$ 
{
  fix  $A\ B$ 
  assume  $Asub$ :  $A \subseteq \{0..<m\} \cup \{0..<Suc\ r\}$ 
  assume  $Bsub$ :  $B \subseteq \{0..<m\} \cup \{0..<Suc\ r\}$ 
  assume  $ABinters$ :  $A \cap B = \{\}$ 
  have  $r \in Is\ p$  using rp p cond unfolding cond-def by simp
  hence  $r\ lt\ m$ :  $r < m$  using p Is-valid[of p] unfolding valid-I-def by auto
  hence  $1$ :  $A \subseteq \{0..<m\}$  using  $Asub$  by auto
  have  $2$ :  $B \subseteq \{0..<m\}$  using  $r\ lt\ m\ Bsub$  by auto
  have  $a\ ' A \cap a\ ' B = \{\}$ 
    using inj-on-image-Int[OF inj-a 1 2] ABinters by auto
} note  $inja = this$ 

have  $(Is\ p \cap \{0..<r\}) \cap (Is\ p \cap \{r\}) = \{\}$  by auto
hence  $a\ ' (Is\ p \cap \{0..<r\} \cup Is\ p \cap \{r\}) = a\ ' (Is\ p \cap \{0..<r\}) \cup a\ ' (Is\ p \cap \{r\})$ 
  using inj-a by auto
moreover have  $Is\ p \cap \{0..<r\} \cup Is\ p \cap \{r\} \subseteq \{0..<m\} \cup \{0..<Suc\ r\}$  by auto
moreover have  $Is\ p \cap \{Suc\ r..<m\} \subseteq \{0..<m\} \cup \{0..<Suc\ r\}$  by auto

```

moreover have $(Is\ p \cap \{0..<r\} \cup Is\ p \cap \{r\}) \cap (Is\ p \cap \{Suc\ r..<m\}) = \{\}$ **by**
auto
ultimately have one: $(a \text{ ' } (Is\ p \cap \{0..<r\}) \cup a \text{ ' } (Is\ p \cap \{r\})) \cap a \text{ ' } (Is\ p \cap \{Suc\ r..<m\}) = \{\}$
using *inja*[of $Is\ p \cap \{0..<r\} \cup Is\ p \cap \{r\}$ $Is\ p \cap \{Suc\ r..<m\}$] **by** *auto*
have *split*: $Is\ p = Is\ p \cap \{0..<r\} \cup Is\ p \cap \{r\} \cup Is\ p \cap \{Suc\ r..<m\}$
using *rp p Is-valid*[of *p*] **unfolding** *valid-I-def* **by** *auto*
have *gtr*: $(\sum w \in (a \text{ ' } (Is\ p \cap \{Suc\ r..<m\})). ((ls\ p)\ w) * (cs\ q \cdot w)) = 0$
proof (*rule sum.neutral, clarify*)
fix *w*
assume *w1*: $w \in Is\ p$ **and** *w2*: $w \in \{Suc\ r..<m\}$
have *w-in-q*: $w \in Is\ q$ **using** *bla*[OF *w2*] *w1* **by** *blast*
moreover have $hs\ q \leq r$ **using** *rR rLarge* **using** *q-len* **by** *blast*
ultimately have $w \neq hs\ q$ **using** *w2* **by** *simp*
hence $w \in Is\ q - \{hs\ q\}$ **using** *w1 w-in-q* **by** *auto*
moreover have $Is\ q - \{hs\ q\} \subseteq \{0..<m\}$
using *q-len Is-valid*[of *q*] **unfolding** *valid-I-def* **by** *auto*
ultimately have $a\ w \in span\ (a \text{ ' } (Is\ q - \{hs\ q\}))$ **using** *a* **by** (*intro span-mem, auto*)
moreover have $cs\ q \in carrier\ vec\ n \wedge span\ (a \text{ ' } (Is\ q - \{hs\ q\})) = \{x. x \in carrier\ vec\ n \wedge cs\ q \cdot x = 0\}$
using *cond*[of *q*] *q-len* **unfolding** *cond-def* **by** *auto*
ultimately have $(cs\ q) \cdot (a\ w) = 0$ **using** *a w2* **by** *simp*
then show $ls\ p\ (a\ w) * (cs\ q \cdot a\ w) = 0$ **by** *simp*
qed
note *pp* = *cond*[OF *p, unfolded cond-def rp*[*symmetric*]]
note *qq* = *cond*[OF *q-len, unfolded cond-def q1*]
have $(cs\ q) \cdot b = (cs\ q) \cdot lincomb\ (ls\ p)\ (a \text{ ' } (Is\ p))$ **using** *blin* **by** *auto*
also have $\dots = (\sum w \in (a \text{ ' } (Is\ p)). ((ls\ p)\ w) * (cs\ q \cdot w))$
by (*subst lincomb-scalar-prod-right*[OF *carrDp carrcq*], *simp*)
also have $\dots = (\sum w \in (a \text{ ' } (Is\ p \cap \{0..<r\}) \cup a \text{ ' } (Is\ p \cap \{r\}) \cup a \text{ ' } (Is\ p \cap \{Suc\ r..<m\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
by (*subst (1) split, rule sum.cong, auto*)
also have $\dots = (\sum w \in (a \text{ ' } (Is\ p \cap \{0..<r\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
 $+ (\sum w \in (a \text{ ' } (Is\ p \cap \{r\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
 $+ (\sum w \in (a \text{ ' } (Is\ p \cap \{Suc\ r..<m\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
apply (*subst sum.union-disjoint*[OF *- - one*])
apply (*force+*)[2]
apply (*subst sum.union-disjoint*)
apply (*force+*)[2]
apply (*rule injA*)
by *auto*
also have $\dots = (\sum w \in (a \text{ ' } (Is\ p \cap \{0..<r\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
 $+ (\sum w \in (a \text{ ' } (Is\ p \cap \{r\})). ((ls\ p)\ w) * (cs\ q \cdot w))$
using *sum.neutral gtr* **by** *simp*
also have $\dots > 0 + 0$
proof (*intro add-le-less-mono sum-nonneg mult-nonneg-nonneg*)
 $\{$

```

fix x
assume x: x ∈ a ‘ (Is p ∩ {0..<r})
show 0 ≤ ls p x using pp x by auto
show 0 ≤ cs q · x using qq x by auto
}
have r ∈ Is p using pp by blast
hence a ‘ (Is p ∩ {r}) = {a r} by auto
hence id: (∑ w∈a ‘ (Is p ∩ {r}). ls p w * (cs q · w)) = ls p (a r) * (cs q · a r)
by simp
show 0 < (∑ w∈a ‘ (Is p ∩ {r}). ls p w * (cs q · w))
unfolding id
proof (rule mult-neg-neg)
show ls p (a r) < 0 using pp by auto
show cs q · a r < 0 using qq by auto
qed
qed
finally have cs q · b > 0 by simp
moreover have cs q · b < 0 using qq by blast
ultimately show False by auto
qed

```

lemma *fundamental-theorem-neg-C-or-B-in-context:*

```

assumes W: W = a ‘ {0 ..<m}
shows (∃ U. U ⊆ W ∧ card U = n ∧ lin-indpt U ∧ b ∈ finite-cone U) ∨
(∃ c U. U ⊆ W ∧
c ∈ {normal-vector U, - normal-vector U} ∧
Suc (card U) = n ∧ lin-indpt U ∧ (∀ w ∈ W. 0 ≤ c · w) ∧ c · b < 0)
using acyclic-imp-goal[unfolded goal-def, OF acyclic-step-rel]
proof
assume ∃ I. I ⊆ {0..<m} ∧ card (a ‘ I) = n ∧ lin-indpt (a ‘ I) ∧ b ∈ finite-cone
(a ‘ I)
thus ?thesis unfolding W by (intro disjI1, blast)
next
assume ∃ c I. I ⊆ {0..<m} ∧
c ∈ {normal-vector (a ‘ I), - normal-vector (a ‘ I)} ∧
Suc (card (a ‘ I)) = n ∧ lin-indpt (a ‘ I) ∧ (∀ i<m. 0 ≤ c · a i) ∧ c · b
< 0
then obtain c I where I ⊆ {0..<m} ∧
c ∈ {normal-vector (a ‘ I), - normal-vector (a ‘ I)} ∧
Suc (card (a ‘ I)) = n ∧ lin-indpt (a ‘ I) ∧ (∀ i<m. 0 ≤ c · a i) ∧ c · b
< 0 by auto
thus ?thesis unfolding W
by (intro disjI2 exI[of - c] exI[of - a ‘ I], auto)
qed

```

end

lemma *fundamental-theorem-of-linear-inequalities-C-imp-B-full-dim:*

```

assumes A: A ⊆ carrier-vec n

```

and *fin*: *finite A*
and *span*: *span A = carrier-vec n*
and *b*: *b ∈ carrier-vec n*
and *C*: $\nexists c \in B. B \subseteq A \wedge c \in \{\text{normal-vector } B, - \text{normal-vector } B\} \wedge \text{Suc}(\text{card } B) = n$
 $\wedge \text{lin-indpt } B \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0$
shows $\exists B \subseteq A. \text{lin-indpt } B \wedge \text{card } B = n \wedge b \in \text{finite-cone } B$
proof –
from *finite-distinct-list*[*OF fin*] **obtain** *as* **where** *Aas*: *A = set as* **and** *dist*:
distinct as **by** *auto*
define *m* **where** *m = length as*
define *a* **where** *a = (λ i. as ! i)*
have *inj*: *inj-on a {0..< (m :: nat)}*
and *id*: *A = a ' {0..< m}*
unfolding *m-def a-def Aas* **using** *inj-on-nth*[*OF dist*] **unfolding** *set-conv-nth*
by *auto*
from *fundamental-theorem-neg-C-or-B-in-context*[*OF - inj b, folded id, OF A span refl*] *C*
show *?thesis* **by** *blast*
qed

lemma *fundamental-theorem-of-linear-inequalities-full-dim*: **fixes** *A* :: '*a vec set*
defines *HyperN* $\equiv \{b. b \in \text{carrier-vec } n \wedge (\nexists B \subseteq A \wedge c \in \{\text{normal-vector } B, - \text{normal-vector } B\} \wedge \text{Suc}(\text{card } B) = n \wedge \text{lin-indpt } B \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0)\}$
defines *HyperA* $\equiv \{b. b \in \text{carrier-vec } n \wedge (\nexists c. c \in \text{carrier-vec } n \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0)\}$
defines *lin-indpt-cone* $\equiv \bigcup \{\text{finite-cone } B \mid B. B \subseteq A \wedge \text{card } B = n \wedge \text{lin-indpt } B\}$
assumes *A*: *A ⊆ carrier-vec n*
and *fin*: *finite A*
and *span*: *span A = carrier-vec n*
shows
cone A = lin-indpt-cone
cone A = HyperN
cone A = HyperA
proof –
have *lin-indpt-cone* \subseteq *cone A* **unfolding** *lin-indpt-cone-def cone-def* **using** *fn finite-cone-mono A*
by *auto*
moreover **have** *cone A* \subseteq *HyperA*
proof
fix *c*
assume *cA*: *c ∈ cone A*
from *fundamental-theorem-of-linear-inequalities-A-imp-D*[*OF A fin this*] *cone-carrier*[*OF A*] *cA*
show *c ∈ HyperA* **unfolding** *HyperA-def* **by** *auto*
qed

moreover have $\text{Hyper}A \subseteq \text{Hyper}N$
proof
 fix c
 assume $c \in \text{Hyper}A$
 hence $\text{False}: \bigwedge v. v \in \text{carrier-vec } n \implies (\forall a_i \in A. 0 \leq v \cdot a_i) \implies v \cdot c < 0$
 $\implies \text{False}$
 and $c: c \in \text{carrier-vec } n$ **unfolding** $\text{Hyper}A\text{-def}$ **by** auto
 show $c \in \text{Hyper}N$
 unfolding $\text{Hyper}N\text{-def}$
 proof (standard , $\text{intro conjI } c \text{ notI}$, clarify , goal-cases)
 case ($1 \ W \ nv$)
 with $A \ \text{fin}$ **have** $\text{fin}: \text{finite } W$ **and** $W: W \subseteq \text{carrier-vec } n$ **by** (auto intro:
 finite-subset)
 show $?case$ **using** $\text{False}[of \ nv] \ 1 \ \text{normal-vector}[OF \ \text{fin} \ - \ W]$ **by** auto
 qed
 qed
moreover have $\text{Hyper}N \subseteq \text{lin-indpt-cone}$
proof
 fix b
 assume $b \in \text{Hyper}N$
 from $\text{this}[\text{unfolded } \text{Hyper}N\text{-def}]$
 $\text{fundamental-theorem-of-linear-inequalities-C-imp-B-full-dim}[OF \ A \ \text{fin} \ \text{span},$
 $\text{of } b]$
 show $b \in \text{lin-indpt-cone}$ **unfolding** $\text{lin-indpt-cone-def}$ **by** auto
 qed
 ultimately show
 $\text{cone } A = \text{lin-indpt-cone}$
 $\text{cone } A = \text{Hyper}N$
 $\text{cone } A = \text{Hyper}A$
 by auto
qed

lemma $\text{fundamental-theorem-of-linear-inequalities-C-imp-B}$:
assumes $A: A \subseteq \text{carrier-vec } n$
and $\text{fin}: \text{finite } A$
and $b: b \in \text{carrier-vec } n$
and $C: \nexists c \ A'. c \in \text{carrier-vec } n$
 $\wedge A' \subseteq A \wedge \text{Suc } (\text{card } A') = \text{dim-span } (\text{insert } b \ A)$
 $\wedge (\forall a \in A'. c \cdot a = 0)$
 $\wedge \text{lin-indpt } A' \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0$
shows $\exists B \subseteq A. \text{lin-indpt } B \wedge \text{card } B = \text{dim-span } A \wedge b \in \text{finite-cone } B$
proof –
 from $\text{exists-lin-indpt-sublist}[OF \ A]$ **obtain** A' **where**
 $\text{lin}: \text{lin-indpt-list } A'$ **and** $\text{span}: \text{span } (\text{set } A') = \text{span } A$ **and** $A'A: \text{set } A' \subseteq A$
by auto
 hence $\text{lin}A': \text{lin-indpt } (\text{set } A')$ **unfolding** $\text{lin-indpt-list-def}$ **by** auto
 from $A'A \ A$ **have** $A': \text{set } A' \subseteq \text{carrier-vec } n$ **by** auto
 have $\text{dim-span}A: \text{dim-span } A = \text{card } (\text{set } A')$
 by (rule sym , $\text{rule same-span-imp-card-eq-dim-span}[OF \ A' \ A \ \text{span } \text{lin}A']$)

```

show ?thesis
proof (cases b ∈ span A)
  case False
  with span have b ∉ span (set A') by auto
  with lin have linAb: lin-indpt-list (A' @ [b]) unfolding lin-indpt-list-def
    using lin-dep-iff-in-span[OF A' - b] span-mem[OF A', of b] b by auto
  interpret gso: gram-schmidt-fs-lin-indpt n A' @ [b]
    by (standard, insert linAb[unfolded lin-indpt-list-def], auto)
  let ?m = length A'
  define c where c = - gso.gso ?m
  have c: c ∈ carrier-vec n using gso.gso-carrier[of ?m] unfolding c-def by
auto
  from gso.gso-times-self-is-norm[of ?m]
  have b · gso.gso ?m = sq-norm (gso.gso ?m) unfolding c-def using b c by
auto
  also have ... > 0 using gso.sq-norm-pos[of ?m] by auto
  finally have cb: c · b < 0 using b c comm-scalar-prod[OF b c] unfolding
c-def by auto
  {
  fix a
  assume a ∈ A
  hence a ∈ span (set A') unfolding span using span-mem[OF A] by auto
  from finite-in-span[OF - A' this]
  obtain l where a = lincomb l (set A') by auto
  hence c · a = c · lincomb l (set A') by simp
  also have ... = 0
  by (subst lincomb-scalar-prod-right[OF A' c], rule sum.neutral, insert A',
unfold set-conv-nth,
insert gso.gso-scalar-zero[of ?m] c, auto simp: c-def nth-append )
  finally have c · a = 0 .
  } note cA = this
  have ∃ c A'. c ∈ carrier-vec n ∧ A' ⊆ A ∧ Suc (card A') = dim-span (insert
b A)
  ∧ (∀ a ∈ A'. c · a = 0) ∧ lin-indpt A' ∧ (∀ ai ∈ A. c · ai ≥ 0) ∧ c · b < 0
  proof (intro exI[of - c] exI[of - set A'] conjI A'A linA' cb c)
  show ∀ a ∈ set A'. c · a = 0 ∀ ai ∈ A. 0 ≤ c · ai using cA A'A by auto
  have dim-span (insert b A) = Suc (dim-span A)
  by (rule dim-span-insert[OF A b False])
  also have ... = Suc (card (set A')) unfolding dim-spanA ..
  finally show Suc (card (set A')) = dim-span (insert b A) ..
  qed
  with C have False by blast
  thus ?thesis ..
next
case bspan: True
define N where N = normal-vectors A'
from normal-vectors[OF lin, folded N-def]
have N: set N ⊆ carrier-vec n and
orthA'N: ⋀ w nv. w ∈ set A' ⇒ nv ∈ set N ⇒ nv · w = 0 and

```

$linAN$: $lin\text{-indpt-list } (A' @ N)$ **and**
 $lenAN$: $length (A' @ N) = n$ **and**
 $disj$: $set A' \cap set N = \{\}$ **by auto**
from $linAN$ $lenAN$ **have** $full\text{-span}'$: $span (set (A' @ N)) = carrier\text{-vec } n$
using $lin\text{-indpt-list-length-eq-n}$ **by blast**
hence $full\text{-span}''$: $span (set A' \cup set N) = carrier\text{-vec } n$ **by auto**
from A N A' **have** AN : $A \cup set N \subseteq carrier\text{-vec } n$ **and** $A'N$: $set (A' @ N) \subseteq$
 $carrier\text{-vec } n$ **by auto**
hence $span (A \cup set N) \subseteq carrier\text{-vec } n$ **by** ($simp$ add : $span\text{-is-subset2}$)
with $A'A$ $span\text{-is-monotone}$ [of $set (A' @ N)$ $A \cup set N$, $unfolded$ $full\text{-span}'$]
have $full\text{-span}$: $span (A \cup set N) = carrier\text{-vec } n$ **unfolding** $set\text{-append}$ **by fast**
from fin **have** $finAN$: $finite (A \cup set N)$ **by auto**
note $fundamental = fundamental\text{-theorem-of-linear-inequalities-full-dim}$ [OF AN
 $finAN$ $full\text{-span}$]
show $?thesis$
proof ($cases$ $b \in cone (A \cup set N)$)
case $True$
from $this$ [$unfolded$ $fundamental(1)$] **obtain** C **where** CAN : $C \subseteq A \cup set N$
and $cardC$: $card C = n$
and $linC$: $lin\text{-indpt } C$
and bC : $b \in finite\text{-cone } C$ **by auto**
have $finC$: $finite C$ **using** $finite\text{-subset}$ [OF CAN] fin **by auto**
from CAN A N **have** C : $C \subseteq carrier\text{-vec } n$ **by auto**
from bC [$unfolded$ $finite\text{-cone-def}$ $nonneg\text{-lincomb-def}$] $finC$ **obtain** c
where bC : $b = lincomb c C$ **and** $nonneg$: $\bigwedge b. b \in C \implies c b \geq 0$ **by auto**
let $?C = C - set N$
show $?thesis$
proof ($intro$ exI [of - $?C$] $conjI$)
from $subset\text{-li-is-li}$ [OF $linC$] **show** $lin\text{-indpt } ?C$ **by auto**
show CA : $?C \subseteq A$ **using** CAN **by auto**
have bc : $b = lincomb c (?C \cup (C \cap set N))$ **unfolding** bC
by ($rule$ $arg\text{-cong}$ [of - - $lincomb$ -], $auto$)
have $b = lincomb c (?C - C \cap set N)$
proof ($rule$ $orthogonal\text{-cone}(2)$ [OF A N fin $full\text{-span}''$ $orthA'N$ $refl$ $span$
 $A'A$ $linAN$ $lenAN$ - CA - bc])
show $\forall w \in set N. w \cdot b = 0$
using $ortho\text{-span}'$ [OF $A' N$ - $bspan$ [$folded$ $span$]] $orthA'N$ **by auto**
qed auto
also have $?C - C \cap set N = ?C$ **by auto**
finally have $b = lincomb c ?C$.
with $nonneg$ **have** $nonneg\text{-lincomb } c ?C b$ **unfolding** $nonneg\text{-lincomb-def}$
by auto
thus $b \in finite\text{-cone } ?C$ **unfolding** $finite\text{-cone-def}$ **using** $finite\text{-subset}$ [OF
 CA fin] **by auto**
have Cid : $C \cap set N \cup ?C = C$ **by auto**
have $length A' + length N = n$ **by fact**
also have $\dots = card (C \cap set N \cup ?C)$ **using** Cid $cardC$ **by auto**
also have $\dots = card (C \cap set N) + card ?C$
by ($subst$ $card\text{-Un-disjoint}$, $insert$ $finC$, $auto$)

```

also have ... ≤ length  $N$  + card  $?C$ 
  by (rule add-right-mono, rule order.trans, rule card-mono[OF finite-set[of
 $N$ ]],
    auto intro: card-length)
also have length  $A' = \text{card } (\text{set } A')$  using lin[unfolded lin-indpt-list-def]
  distinct-card[of  $A'$ ] by auto
finally have le: dim-span  $A \leq \text{card } ?C$  using dim-spanA by auto
have CA:  $?C \subseteq \text{span } A$  using CA C in-own-span[OF  $A$ ] by auto
have linC: lin-indpt  $?C$  using subset-li-is-li[OF linC] by auto
show card  $?C = \text{dim-span } A$ 
  using card-le-dim-span[OF - CA linC] le C by force
qed
next
case False
from False[unfolded fundamental(2)] b
obtain C c where
  CAN:  $C \subseteq A \cup \text{set } N$  and
  cardC: Suc (card C) = n and
  linC: lin-indpt C and
  contains: ( $\forall a_i \in A \cup \text{set } N. 0 \leq c \cdot a_i$ ) and
  cb:  $c \cdot b < 0$  and
  nv:  $c \in \{\text{normal-vector } C, - \text{normal-vector } C\}$ 
  by auto
from CAN A N have C:  $C \subseteq \text{carrier-vec } n$  by auto
from cardC have cardCn: card C < n by auto
from finite-subset[OF CAN] fin have finC: finite C by auto
let  $?C = C - \text{set } N$ 
note nv' = normal-vector(1-4)[OF finC cardC linC C]
from nv' nv have c:  $c \in \text{carrier-vec } n$  by auto
have  $\exists c A'. c \in \text{carrier-vec } n \wedge A' \subseteq A \wedge \text{Suc } (\text{card } A') = \text{dim-span } (\text{insert } b A)$ 
   $\wedge (\forall a \in A'. c \cdot a = 0) \wedge \text{lin-indpt } A' \wedge (\forall a_i \in A. c \cdot a_i \geq 0) \wedge c \cdot b < 0$ 
proof (intro exI[of - c] exI[of - ?C] conjI cb c)
  show CA:  $?C \subseteq A$  using CAN by auto
  show  $\forall a_i \in A. 0 \leq c \cdot a_i$  using contains by auto
  show lin': lin-indpt  $?C$  using subset-li-is-li[OF linC] by auto
  show sC0:  $\forall a \in ?C. c \cdot a = 0$  using nv' nv C by auto
  have Cid:  $C \cap \text{set } N \cup ?C = C$  by auto
  have dim-span (set  $A'$ ) = card (set  $A'$ )
    by (rule sym, rule same-span-imp-card-eq-dim-span[OF  $A' A'$  refl linA'])
  also have ... = length  $A'$ 
    using lin[unfolded lin-indpt-list-def] distinct-card[of  $A'$ ] by auto
  finally have dimA': dim-span (set  $A'$ ) = length  $A'$ .
from bspan have span (insert b  $A$ ) = span  $A$  using b A using already-in-span
by auto
from dim-span-cong[OF this[folded span]] dimA'
have dimbA: dim-span (insert b  $A$ ) = length  $A'$  by simp
also have ... = Suc (card  $?C$ )

```

```

proof (rule ccontr)
  assume neq: length A' ≠ Suc (card ?C)
  have length A' + length N = n by fact
  also have ... = Suc (card (C ∩ set N ∪ ?C)) using Cid cardC by auto
  also have ... = Suc (card (C ∩ set N) + card ?C)
    by (subst card-Un-disjoint, insert finC, auto)
  finally have id: length A' + length N = Suc (card (C ∩ set N) + card
?C) .
  have le1: card (C ∩ set N) ≤ length N
    by (metis Int-lower2 List.finite-set card-length card-mono inf.absorb-iff2
le-inf-iff)
  from CA C A have CsA: ?C ⊆ span (set A') unfolding span by (meson
in-own-span order.trans)
  from card-le-dim-span[OF - this lin'] C
  have le2: card ?C ≤ length A' unfolding dimA' by auto
  from id le1 le2 neq
  have id2: card ?C = length A' by linarith+
  from card-eq-dim-span-imp-same-span[OF A' CsA lin' id2[folded dimA']]
  have span ?C = span A unfolding span by auto
  with bspan have b ∈ span ?C by auto
  from orthocompl-span[OF - - c this] C sC0
  have c · b = 0 by auto
  with cb show False by simp
qed
finally show Suc (card ?C) = dim-span (insert b A) by simp
qed
with assms(4) have False by blast
thus ?thesis ..
qed
qed
qed

```

```

lemma fundamental-theorem-of-linear-inequalities: fixes A :: 'a vec set
defines HyperN ≡ {b. b ∈ carrier-vec n ∧ (∃ c B. c ∈ carrier-vec n ∧ B ⊆ A
  ∧ Suc (card B) = dim-span (insert b A) ∧ lin-indpt B
  ∧ (∀ a ∈ B. c · a = 0)
  ∧ (∀ ai ∈ A. c · ai ≥ 0) ∧ c · b < 0)}
defines HyperA ≡ {b. b ∈ carrier-vec n ∧ (∃ c. c ∈ carrier-vec n ∧ (∀ ai ∈ A.
c · ai ≥ 0) ∧ c · b < 0)}
defines lin-indpt-cone ≡ ∪ {finite-cone B | B. B ⊆ A ∧ card B = dim-span A
  ∧ lin-indpt B}
assumes A: A ⊆ carrier-vec n
and fin: finite A
shows
  cone A = lin-indpt-cone
  cone A = HyperN
  cone A = HyperA
proof -
have lin-indpt-cone ⊆ cone A

```

```

    unfolding lin-indpt-cone-def cone-def using fin finite-cone-mono A by auto
  moreover have cone A  $\subseteq$  HyperA
  using fundamental-theorem-of-linear-inequalities-A-imp-D[OF A fin] cone-carrier[OF
A]
  unfolding HyperA-def by blast
  moreover have HyperA  $\subseteq$  HyperN unfolding HyperA-def HyperN-def by blast
  moreover have HyperN  $\subseteq$  lin-indpt-cone
  proof
    fix b
    assume b  $\in$  HyperN
    from this[unfolded HyperN-def]
      fundamental-theorem-of-linear-inequalities-C-imp-B[OF A fin, of b]
    show b  $\in$  lin-indpt-cone unfolding lin-indpt-cone-def by blast
  qed
  ultimately show
    cone A = lin-indpt-cone
    cone A = HyperN
    cone A = HyperA
  by auto
  qed

corollary Caratheodory-theorem: assumes A: A  $\subseteq$  carrier-vec n
  shows cone A =  $\bigcup$  {finite-cone B | B. B  $\subseteq$  A  $\wedge$  lin-indpt B}
  proof
    show  $\bigcup$  {finite-cone B | B. B  $\subseteq$  A  $\wedge$  lin-indpt B}  $\subseteq$  cone A unfolding cone-def
      using fin[OF fin-dim - subset-trans[OF - A]] by auto
    {
      fix a
      assume a  $\in$  cone A
      from this[unfolded cone-def] obtain B where
        finB: finite B and BA: B  $\subseteq$  A and a: a  $\in$  finite-cone B by auto
      from BA A have B: B  $\subseteq$  carrier-vec n by auto
      hence a  $\in$  cone B using finB a by (simp add: cone-iff-finite-cone)
      with fundamental-theorem-of-linear-inequalities(1)[OF B finB]
      obtain C where CB: C  $\subseteq$  B and a: a  $\in$  finite-cone C and lin-indpt C by
      auto
      with BA have a  $\in$   $\bigcup$  {finite-cone B | B. B  $\subseteq$  A  $\wedge$  lin-indpt B} by auto
    }
    thus  $\bigcup$  {finite-cone B | B. B  $\subseteq$  A  $\wedge$  lin-indpt B}  $\supseteq$  cone A by blast
  qed
end
end

```

12 Farkas' Lemma

We prove two variants of Farkas' lemma. Note that type here is more general than in the versions of Farkas' Lemma which are in the AFP-entry Farkas-Lemma, which is restricted to rational matrices. However, there δ -rationals

are supported, which are not present here.

theory *Farkas-Lemma*

imports *Fundamental-Theorem-Linear-Inequalities*

begin

context *gram-schmidt*

begin

lemma *Farkas-Lemma*: **fixes** $A :: 'a \text{ mat}$ **and** $b :: 'a \text{ vec}$

assumes $A: A \in \text{carrier-mat } n \text{ nr}$ **and** $b: b \in \text{carrier-vec } n$

shows $(\exists x. x \geq 0_v \text{ nr} \wedge A *_v x = b) \longleftrightarrow (\forall y. y \in \text{carrier-vec } n \longrightarrow A^T *_v y \geq 0_v \text{ nr} \longrightarrow y \cdot b \geq 0)$

proof –

let $?C = \text{set } (\text{cols } A)$

from A **have** $C: ?C \subseteq \text{carrier-vec } n$ **and** $C': \forall w \in \text{set } (\text{cols } A). \text{dim-vec } w = n$

unfolding *cols-def* **by** *auto*

have $(\exists x. x \geq 0_v \text{ nr} \wedge A *_v x = b) = (b \in \text{cone } ?C)$

using *cone-of-cols[OF A b]* **by** *simp*

also have $\dots = (\exists y. y \in \text{carrier-vec } n \wedge (\forall a_i \in ?C. 0 \leq y \cdot a_i) \wedge y \cdot b < 0)$

unfolding *fundamental-theorem-of-linear-inequalities(3)[OF C finite-set]* *mem-Collect-eq*

using b **by** *auto*

also have $\dots = (\forall y. y \in \text{carrier-vec } n \longrightarrow (\forall a_i \in ?C. 0 \leq y \cdot a_i) \longrightarrow y \cdot b \geq 0)$

by *auto*

also have $\dots = (\forall y. y \in \text{carrier-vec } n \longrightarrow A^T *_v y \geq 0_v \text{ nr} \longrightarrow y \cdot b \geq 0)$

proof (*intro all-cong imp-cong refl*)

fix $y :: 'a \text{ vec}$

assume $y: y \in \text{carrier-vec } n$

have $(\forall a_i \in ?C. 0 \leq y \cdot a_i) = (\forall a_i \in ?C. 0 \leq a_i \cdot y)$

by (*intro ball-cong[OF refl], subst comm-scalar-prod[OF y], insert C, auto*)

also have $\dots = (0_v \text{ nr} \leq A^T *_v y)$

unfolding *less-eq-vec-def* **using** $C A y$ **by** (*auto simp: cols-def*)

finally show $(\forall a_i \in \text{set } (\text{cols } A). 0 \leq y \cdot a_i) = (0_v \text{ nr} \leq A^T *_v y) .$

qed

finally show *?thesis* .

qed

lemma *Farkas-Lemma'*:

fixes $A :: 'a \text{ mat}$ **and** $b :: 'a \text{ vec}$

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

shows $(\exists x. x \in \text{carrier-vec } nc \wedge A *_v x \leq b)$

$\longleftrightarrow (\forall y. y \geq 0_v \text{ nr} \wedge A^T *_v y = 0_v \ nc \longrightarrow y \cdot b \geq 0)$

proof –

define B **where** $B = (1_m \ nr) @_c (A @_c -A)$

define b' **where** $b' = 0_v \ nc @_v (b @_v -b)$

define n **where** $n = nr + (nc + nc)$

have $id0: 0_v (nr + (nc + nc)) = 0_v \ nr @_v (0_v \ nc @_v 0_v \ nc)$ **by** (*intro eq-vecI, auto*)

have $idcarr: (1_m \ nr) \in \text{carrier-mat } nr \ nr$ **by** *auto*

have $B: B \in \text{carrier-mat } nr \ n$ **unfolding** $B\text{-def } n\text{-def}$ **using** A **by** auto
have $(\exists x \in \text{carrier-vec } nc. A *_{\nu} x \leq b) =$
 $(\exists x1 \in \text{carrier-vec } nr. \exists x2 \in \text{carrier-vec } nc. \exists x3 \in \text{carrier-vec } nc.$
 $x1 \geq 0_{\nu} nr \wedge x2 \geq 0_{\nu} nc \wedge x3 \geq 0_{\nu} nc \wedge B *_{\nu} (x1 @_{\nu} (x2 @_{\nu} x3)) = b)$
proof
assume $\exists x \in \text{carrier-vec } nc. A *_{\nu} x \leq b$
from this obtain x **where** $Axb: A *_{\nu} x \leq b$ **and** $xcarr: x \in \text{carrier-vec } nc$ **by**
 auto
have $bmAx: b - A *_{\nu} x \in \text{carrier-vec } nr$ **using** $A \ b \ xcarr$ **by** simp
define $x1$ **where** $x1 = b - A *_{\nu} x$
have $x1: x1 \in \text{carrier-vec } nr$ **using** $bmAx$ **unfolding** $x1\text{-def}$ **by** $(\text{simp add:}$
 $xcarr)$
define $x2$ **where** $x2 = \text{vec } (\text{dim-vec } x) (\lambda i. \text{if } x \ \$ \ i \geq 0 \ \text{then } x \ \$ \ i \ \text{else } 0)$
have $x2: x2 \in \text{carrier-vec } nc$ **using** $xcarr$ **unfolding** $x2\text{-def}$ **by** simp
define $x3$ **where** $x3 = \text{vec } (\text{dim-vec } x) (\lambda i. \text{if } x \ \$ \ i < 0 \ \text{then } -x \ \$ \ i \ \text{else } 0)$
have $x3: x3 \in \text{carrier-vec } nc$ **using** $xcarr$ **unfolding** $x3\text{-def}$ **by** simp
have $x2x3carr: x2 @_{\nu} x3 \in \text{carrier-vec } (nc + nc)$ **using** $x2 \ x3$ **by** simp
have $x2x3x: x2 - x3 = x$ **unfolding** $x2\text{-def } x3\text{-def}$ **by** auto
have $A *_{\nu} x - b \leq 0_{\nu} nr$ **using** $\text{vec-le-iff-diff-le-0 } b$
by $(\text{metis } A \ Axb \ \text{carrier-matD}(1) \ \text{dim-mult-mat-vec})$
hence $x1lez: x1 \geq 0_{\nu} nr$ **using** $x1$ **unfolding** $x1\text{-def}$
by $(\text{smt } A \ Axb \ \text{carrier-matD}(1) \ \text{carrier-vecD } \text{diff-ge-0-iff-ge } \text{dim-mult-mat-vec}$
 $\text{index-minus-vec}(1) \ \text{index-zero-vec}(1) \ \text{index-zero-vec}(2) \ \text{less-eq-vec-def})$
have $x2lez: x2 \geq 0_{\nu} nc$ **using** $x2$ less-eq-vec-def **unfolding** $x2\text{-def}$ **by** fastforce
have $x3lez: x3 \geq 0_{\nu} nc$ **using** $x3$ less-eq-vec-def **unfolding** $x3\text{-def}$ **by** fastforce
have $B1: (1_m \ nr) *_{\nu} x1 = b - A *_{\nu} x$ **using** $xcarr \ x1$ **unfolding** $x1\text{-def}$ **by**
 simp
have $A *_{\nu} x2 + (-A) *_{\nu} x3 = A *_{\nu} x2 + A *_{\nu} (-x3)$ **using** $x2 \ x3 \ A$ **by** auto
also have $\dots = A *_{\nu} (x2 + (-x3))$ **using** $A \ x2 \ x3$
by $(\text{metis } \text{mult-add-distrib-mat-vec } \text{uminus-carrier-vec})$
also have $\dots = A *_{\nu} x$ **using** $x2x3x$ $\text{minus-add-uminus-vec}$ $x2 \ x3$ **by** fastforce
finally have $B2: A *_{\nu} x2 + (-A) *_{\nu} x3 = A *_{\nu} x$ **by** auto
have $B *_{\nu} (x1 @_{\nu} (x2 @_{\nu} x3)) = (1_m \ nr) *_{\nu} x1 + (A *_{\nu} x2 + (-A) *_{\nu} x3)$
(is $\dots = ?p1 + ?p2)$
using $x1 \ x2 \ x3 \ A$ $\text{mat-mult-append-cols}$ **unfolding** $B\text{-def}$
by $(\text{subst } \text{mat-mult-append-cols}[OF - - \ x1 \ x2x3carr], \ \text{auto } \text{simp add: } \text{mat-mult-append-cols})$
also have $?p1 = b - A *_{\nu} x$ **using** $B1$ **unfolding** $x1\text{-def}$ **by** auto
also have $?p2 = A *_{\nu} x$ **using** $B2$ **by** simp
finally have $\text{res: } B *_{\nu} (x1 @_{\nu} (x2 @_{\nu} x3)) = b$ **using** $A \ xcarr \ b$ **by** auto
show $\exists x \in \text{carrier-vec } nc. A *_{\nu} x \leq b \implies \exists x1 \in \text{carrier-vec } nr. \exists x2 \in \text{carrier-vec}$
 $nc. \exists x3 \in \text{carrier-vec } nc.$
 $0_{\nu} nr \leq x1 \wedge 0_{\nu} nc \leq x2 \wedge 0_{\nu} nc \leq x3 \wedge B *_{\nu} (x1 @_{\nu} x2 @_{\nu} x3) = b$
using $x1 \ x2 \ x3 \ x1lez \ x2lez \ x3lez \ \text{res}$ **by** auto
next
assume $\exists x1 \in \text{carrier-vec } nr. \exists x2 \in \text{carrier-vec } nc. \exists x3 \in \text{carrier-vec } nc.$
 $x1 \geq 0_{\nu} nr \wedge x2 \geq 0_{\nu} nc \wedge x3 \geq 0_{\nu} nc \wedge B *_{\nu} (x1 @_{\nu} (x2 @_{\nu} x3)) = b$
from this obtain $x1 \ x2 \ x3$ **where** $x1: x1 \in \text{carrier-vec } nr$ **and** $x1lez: x1 \geq 0_{\nu}$
 nr

and $x2$: $x2 \in \text{carrier-vec } nc$ **and** $x2lez$: $x2 \geq 0_v \ nc$
and $x3$: $x3 \in \text{carrier-vec } nc$ **and** $x3lez$: $x3 \geq 0_v \ nc$
and clc : $B *_v (x1 @_v (x2 @_v x3)) = b$ **by** *auto*
have $x2x3carr$: $x2 @_v x3 \in \text{carrier-vec } (nc + nc)$ **using** $x2 \ x3$ **by** *simp*
define x **where** $x = x2 - x3$
have $xcarr$: $x \in \text{carrier-vec } nc$ **using** $x2 \ x3$ **unfolding** $x\text{-def}$ **by** *simp*
have $A *_v x2 + (-A) *_v x3 = A *_v x2 + A *_v (-x3)$ **using** $x2 \ x3 \ A$ **by** *auto*
also have $\dots = A *_v (x2 + (-x3))$ **using** $A \ x2 \ x3$
by (*metis mult-add-distrib-mat-vec uminus-carrier-vec*)
also have $\dots = A *_v x$ **using** *minus-add-uminus-vec* $x2 \ x3$ **unfolding** $x\text{-def}$
by *fastforce*
finally have $B2:A *_v x2 + (-A) *_v x3 = A *_v x$ **by** *auto*
have $Axcarr$: $A *_v x \in \text{carrier-vec } nr$ **using** $A \ xcarr$ **by** *auto*
have $b = B *_v (x1 @_v (x2 @_v x3))$ **using** clc **by** *auto*
also have $\dots = (1_m \ nr) *_v x1 + (A *_v x2 + (-A) *_v x3)$ (**is** $\dots = ?p1 +$
 $?p2$)
using $x1 \ x2 \ x3 \ A \ \text{mat-mult-append-cols}$ **unfolding** $B\text{-def}$
by (*subst mat-mult-append-cols[OF - - x1 x2x3carr]*, *auto simp add: mat-mult-append-cols*)
also have $?p2 = A *_v x$ **using** $B2$ **by** *simp*
finally have res : $b = (1_m \ nr) *_v x1 + A *_v x$ **using** $A \ xcarr \ b$ **by** *auto*
hence $b = x1 + A *_v x$ **using** $x1 \ A \ b$ **by** *simp*
hence $b - A *_v x = x1$ **using** $x1 \ A \ b$ **by** *auto*
hence $b - A *_v x \geq 0_v \ nr$ **using** $x1lez$ **by** *auto*
hence $A *_v x \leq b$ **using** $Axcarr$
by (*smt* $\langle b - A *_v x = x1 \rangle \langle b = x1 + A *_v x \rangle$ *carrier-vecD comm-add-vec*
index-zero-vec(2)
minus-add-minus-vec minus-cancel-vec vec-le-iff-diff-le-0 x1)
then show $\exists x1 \in \text{carrier-vec } nr. \exists x2 \in \text{carrier-vec } nc. \exists x3 \in \text{carrier-vec } nc.$
 $0_v \ nr \leq x1 \wedge 0_v \ nc \leq x2 \wedge 0_v \ nc \leq x3 \wedge B *_v (x1 @_v x2 @_v x3) = b$
 \implies
 $\exists x \in \text{carrier-vec } nc. A *_v x \leq b$ **using** $xcarr$ **by** *blast*
qed
also have $\dots = (\exists x1 \in \text{carrier-vec } nr. \exists x2 \in \text{carrier-vec } nc. \exists x3 \in \text{carrier-vec } nc.$
 $nc.$
 $(x1 @_v (x2 @_v x3)) \geq 0_v \ n \wedge B *_v (x1 @_v (x2 @_v x3)) = b)$
by (*metis append-vec-le id0 n-def zero-carrier-vec*)
also have $\dots = (\exists x \in \text{carrier-vec } n. x \geq 0_v \ n \wedge B *_v x = b)$
unfolding $n\text{-def exists-vec-append}$ **by** *auto*
also have $\dots = (\exists x \geq 0_v \ n. B *_v x = b)$ **unfolding** *less-eq-vec-def* **by** *fastforce*
also have $\dots = (\forall y. y \in \text{carrier-vec } nr \longrightarrow B^T *_v y \geq 0_v \ n \longrightarrow y \cdot b \geq 0)$
by (*rule gram-schmidt.Farkas-Lemma[OF B b]*)
also have $\dots = (\forall y. y \in \text{carrier-vec } nr \longrightarrow (y \geq 0_v \ nr \wedge A^T *_v y = 0_v \ nc)$
 $\longrightarrow y \cdot b \geq 0)$
proof (*intro all-cong imp-cong refl*)
fix $y :: 'a \ \text{vec}$
assume y : $y \in \text{carrier-vec } nr$
have $idtcarr$: $(1_m \ nr)^T \in \text{carrier-mat } nr \ nr$ **by** *auto*
have $Atcarr$: $A^T \in \text{carrier-mat } nc \ nr$ **using** A **by** *auto*
have $mAtcarr$: $(-A)^T \in \text{carrier-mat } nc \ nr$ **using** A **by** *auto*

```

have AtAtcarr:  $A^T @_r (-A)^T \in \text{carrier-mat } (nc + nc) \text{ nr}$  using A by auto
have  $B^T *_v y = ((1_m \text{ nr})^T @_r A^T @_r (-A)^T) *_v y$  unfolding B-def
  by (simp add: append-cols-def)
also have ... =  $((1_m \text{ nr})^T *_v y) @_v (A^T *_v y) @_v ((-A)^T *_v y)$ 
  using mat-mult-append[OF Atcarr mAtcarr y] mat-mult-append y Atcarr
idtcarr mAtcarr
  by (metis AtAtcarr)
finally have eq:  $B^T *_v y = ((1_m \text{ nr})^T *_v y) @_v (A^T *_v y) @_v ((-A)^T *_v y)$ 
by auto
have  $(B^T *_v y \geq 0_v n) = (0_v n \leq (1_m \text{ nr})^T *_v y @_v A^T *_v y @_v (-A)^T *_v y)$ 
unfolding eq by simp
also have ... =  $((1_m \text{ nr})^T *_v y) @_v (A^T *_v y) @_v ((-A)^T *_v y) \geq 0_v \text{ nr}$ 
@_v  $0_v \text{ nc} @_v 0_v \text{ nc}$ 
  using id0 by (metis eq n-def)
also have ... =  $(y \geq 0_v \text{ nr} \wedge A^T *_v y \geq 0_v \text{ nc} \wedge ((-A)^T *_v y) \geq 0_v \text{ nc})$ 
  by (metis Atcarr append-vec-le mult-mat-vec-carrier one-mult-mat-vec transpose-one y zero-carrier-vec)
also have ... =  $(y \geq 0_v \text{ nr} \wedge A^T *_v y \geq 0_v \text{ nc} \wedge -(A^T *_v y) \geq 0_v \text{ nc})$ 
  by (metis A Atcarr carrier-matD(2) carrier-vecD transpose-uminus uminus-mult-mat-vec y)
also have ... =  $(y \geq 0_v \text{ nr} \wedge A^T *_v y \geq 0_v \text{ nc} \wedge (A^T *_v y) \leq 0_v \text{ nc})$ 
  by (metis (mono-tags, lifting) A Atcarr carrier-matD(2) carrier-vecD index-zero-vec(2)
mAtcarr mult-mat-vec-carrier transpose-uminus uminus-mult-mat-vec uminus-uminus-vec
vec-le-iff-diff-le-0 y zero-minus-vec)
also have ... =  $(y \geq 0_v \text{ nr} \wedge A^T *_v y = 0_v \text{ nc})$  by auto
finally show  $(B^T *_v y \geq 0_v n) = (y \geq 0_v \text{ nr} \wedge A^T *_v y = 0_v \text{ nc})$  .
qed
finally show ?thesis by (auto simp: less-eq-vec-def)
qed
end
end

```

13 The Theorem of Farkas, Minkowsky and Weyl

We prove the theorem of Farkas, Minkowsky and Weyl that a cone is finitely generated iff it is polyhedral. Moreover, we provide quantitative bounds via determinant bounds.

```

theory Farkas-Minkowsky-Weyl
  imports Fundamental-Theorem-Linear-Inequalities
begin

context gram-schmidt
begin

```

We first prove the one direction of the theorem for the case that the span

of the vectors is the full n -dimensional space.

lemma *farkas-minkowsky-weyl-theorem-1-full-dim*:

assumes $X: X \subseteq \text{carrier-vec } n$
and $\text{fin}: \text{finite } X$
and $\text{span}: \text{span } X = \text{carrier-vec } n$
shows $\exists \text{ nr } A. A \in \text{carrier-mat nr } n \wedge \text{cone } X = \text{polyhedral-cone } A$
 $\wedge (\text{is-det-bound } db \longrightarrow X \subseteq \mathbf{Z}_v \cap \text{Bounded-vec (of-int Bnd)} \longrightarrow A \in \mathbf{Z}_m \cap \text{Bounded-mat (of-int (db (n-1) Bnd))})$
proof –
define cond **where** $\text{cond} = (\lambda W. \text{Suc (card } W) = n \wedge \text{lin-indpt } W \wedge W \subseteq X)$
let $?oi = \text{of-int} :: \text{int} \Rightarrow 'a$
 $\{$
fix W
assume $\text{cond } W$
hence $*$: $\text{finite } W \text{ Suc (card } W) = n \text{ lin-indpt } W \text{ } W \subseteq \text{carrier-vec } n$ **and** $WX: W \subseteq X$ **unfolding** cond-def
using $\text{finite-subset}[OF - \text{fin}] X$ **by** auto
note $\text{nv} = \text{normal-vector}[OF *]$
hence $\text{normal-vector } W \in \text{carrier-vec } n \wedge w. w \in W \implies \text{normal-vector } W \cdot w = 0$
 $\text{normal-vector } W \neq 0_v \text{ } n \text{ is-det-bound } db \implies X \subseteq \mathbf{Z}_v \cap \text{Bounded-vec (?oi Bnd)} \implies \text{normal-vector } W \in \mathbf{Z}_v \cap \text{Bounded-vec (?oi (db (n - 1) Bnd))}$
using WX **by** blast+
 $\}$ **note** $\text{condD} = \text{this}$
define Ns **where** $Ns = \{ \text{normal-vector } W \mid W. \text{cond } W \wedge (\forall w \in X. \text{normal-vector } W \cdot w \geq 0) \}$
 $\cup \{ - \text{normal-vector } W \mid W. \text{cond } W \wedge (\forall w \in X. (- \text{normal-vector } W) \cdot w \geq 0) \}$
have $Ns \subseteq \text{normal-vector } \{ \{ W . W \subseteq X \} \cup (\lambda W. - \text{normal-vector } W) \{ \{ W . W \subseteq X \} \}$ **unfolding** $Ns\text{-def cond-def}$ **by** blast
moreover **have** $\text{finite} \dots$ **using** $\langle \text{finite } X \rangle$ **by** auto
ultimately **have** $\text{finite } Ns$ **by** $(\text{metis finite-subset})$
from $\text{finite-list}[OF \text{ this}]$ **obtain** ns **where** $ns: \text{set } ns = Ns$ **by** auto
have $Ns: Ns \subseteq \text{carrier-vec } n$ **unfolding** $Ns\text{-def}$ **using** condD **by** auto
define A **where** $A = \text{mat-of-rows } n \text{ } ns$
define nr **where** $nr = \text{length } ns$
have $A: - A \in \text{carrier-mat nr } n$ **unfolding** $A\text{-def nr-def}$ **by** auto
show $?thesis$
proof $(\text{intro exI conjI impI, rule } A)$
have $\text{not-conj}: \neg (a \wedge b) \longleftrightarrow (a \longrightarrow \neg b)$ **for** $a \ b$ **by** auto
have $\text{id}: Ns = \{ \text{nv} . \exists W. W \subseteq X \wedge \text{nv} \in \{ \text{normal-vector } W, - \text{normal-vector } W \} \wedge \text{Suc (card } W) = n \wedge \text{lin-indpt } W \wedge (\forall a_i \in X. 0 \leq \text{nv} \cdot a_i) \}$
unfolding $Ns\text{-def cond-def}$ **by** auto
have $\text{polyhedral-cone } (- A) = \{ b. b \in \text{carrier-vec } n \wedge (- A) * _v b \leq 0_v \text{ nr} \}$
unfolding $\text{polyhedral-cone-def}$
using A **by** auto
also **have** $\dots = \{ b. b \in \text{carrier-vec } n \wedge (\forall i < nr. \text{row } (- A) \ i \cdot b \leq 0) \}$
unfolding less-eq-vec-def **using** A **by** auto

```

also have ... = {b. b ∈ carrier-vec n ∧ (∀ i < nr. - (ns ! i) · b ≤ 0)} using
A Ns[folded ns]
  by (intro Collect-cong conj-cong refl all-cong arg-cong[of - - λ x. x · - ≤ -],
    force simp: A-def mat-of-rows-def nr-def set-conv-nth)
also have ... = {b. b ∈ carrier-vec n ∧ (∀ n ∈ Ns. - n · b ≤ 0)}
  unfolding ns[symmetric] nr-def by (auto simp: set-conv-nth)
also have ... = {b. b ∈ carrier-vec n ∧ (∀ n ∈ Ns. n · b ≥ 0)}
  by (intro Collect-cong conj-cong refl ball-cong, insert Ns, auto)
also have ... = cone X
  unfolding fundamental-theorem-of-linear-inequalities-full-dim(2)[OF X fin
span]
  by (intro Collect-cong conj-cong refl, unfold not-le[symmetric] not-ex not-conj
not-not id, blast)
finally show cone X = polyhedral-cone (- A) ..
{
  assume XI: X ⊆  $\mathbb{Z}_v \cap \text{Bounded-vec}$  (?oi Bnd) and db: is-det-bound db
  {
    fix v
    assume v ∈ set (rows (- A))
    hence -v ∈ set (rows A) unfolding rows-def by auto
    hence -v ∈ Ns unfolding A-def using ns Ns by auto
    from this[unfolded Ns-def] obtain W where cW: cond W
      and v: -v = normal-vector W ∨ v = normal-vector W by auto
    from cW[unfolded cond-def] have WX: W ⊆ X by auto
    from v have v: v = normal-vector W ∨ v = - normal-vector W
      by (metis uminus-uminus-vec)
    from condD(4)[OF cW db XI]
    have normal-vector W ∈  $\mathbb{Z}_v \cap \text{Bounded-vec}$  (?oi (db (n - 1) Bnd)) by
auto
    hence v ∈  $\mathbb{Z}_v \cap \text{Bounded-vec}$  (?oi (db (n - 1) Bnd)) using v by auto
  }
  hence set (rows (- A)) ⊆  $\mathbb{Z}_v \cap \text{Bounded-vec}$  (?oi (db (n - 1) Bnd)) by blast
  thus - A ∈  $\mathbb{Z}_m \cap \text{Bounded-mat}$  (?oi (db (n - 1) Bnd)) by simp
}
qed
qed

```

We next generalize the theorem to the case where X does not span the full space. To this end, we extend X by unit-vectors until the full space is spanned, and then add the normal-vectors of these unit-vectors which are orthogonal to span X as additional constraints to the resulting matrix.

lemma *farkas-minkowsky-weyl-theorem-1*:

```

assumes X: X ⊆ carrier-vec n
and finX: finite X
shows ∃ nr A. A ∈ carrier-mat nr n ∧ cone X = polyhedral-cone A ∧
  (is-det-bound db → X ⊆  $\mathbb{Z}_v \cap \text{Bounded-vec}$  (of-int Bnd) → A ∈  $\mathbb{Z}_m \cap$ 
Bounded-mat (of-int (db (n-1) (max 1 Bnd))))
proof -
  let ?oi = of-int :: int ⇒ 'a

```

from *exists-lin-indpt-sublist*[*OF X*]
obtain *Ls* **where** *lin-Ls*: *lin-indpt-list Ls* **and**
spanL: *span (set Ls) = span X* **and** *LX*: *set Ls \subseteq X* **by** *auto*
define *Ns* **where** *Ns = normal-vectors Ls*
define *Bs* **where** *Bs = basis-extension Ls*
from *basis-extension*[*OF lin-Ls, folded Bs-def*]
have *BU*: *set Bs \subseteq set (unit-vecs n)*
and *lin-Ls-Bs*: *lin-indpt-list (Ls @ Bs)*
and *len-Ls-Bs*: *length (Ls @ Bs) = n*
by *auto*
note *nv = normal-vectors*[*OF lin-Ls, folded Ns-def*]
from *nv(1-6) nv(7)*[*of db Bnd*]
have *N*: *set Ns \subseteq carrier-vec n*
and *LN'*: *lin-indpt-list (Ls @ Ns) length (Ls @ Ns) = n*
and *ortho*: $\bigwedge l w. l \in \text{set } Ls \implies w \in \text{set } Ns \implies w \cdot l = 0$
and *Ns-bnd*: *is-det-bound db \implies set Ls \subseteq $\mathbb{Z}_v \cap \text{Bounded-vec} (?oi Bnd)$*
 $\implies \text{set } Ns \subseteq \mathbb{Z}_v \cap \text{Bounded-vec} (?oi (db (n-1) (max 1 Bnd)))$
by *auto*
from *lin-indpt-list-length-eq-n*[*OF LN'*]
have *spanLN*: *span (set Ls \cup set Ns) = carrier-vec n* **by** *auto*
let *?Bnd = Bounded-vec (?oi (db (n-1) (max 1 Bnd)))*
let *?Bndm = Bounded-mat (?oi (db (n-1) (max 1 Bnd)))*
define *Y* **where** *Y = X \cup set Bs*
from *lin-Ls-Bs*[*unfolded lin-indpt-list-def*] **have**
Ls: *set Ls \subseteq carrier-vec n* **and**
Bs: *set Bs \subseteq carrier-vec n* **and**
distLsBs: *distinct (Ls @ Bs)* **and**
lin': *lin-indpt (set (Ls @ Bs))* **by** *auto*
have *LN*: *set Ls \cap set Ns = {}*
proof (*rule ccontr*)
assume $\neg ?thesis$
then obtain *x* **where** *xX*: *x \in set Ls* **and** *xW*: *x \in set Ns* **by** *auto*
from *ortho*[*OF xX xW*] **have** *x \cdot x = 0* **by** *auto*
hence *sq-norm x = 0* **by** (*auto simp: sq-norm-vec-as-cscalar-prod*)
with *xX LX X* **have** *x = 0_v n* **by** *auto*
with *vs-zero-lin-dep*[*OF - lin'*] *Ls Bs xX* **show** *False* **by** *auto*
qed
have *Y*: *Y \subseteq carrier-vec n* **using** *X Bs unfolding Y-def* **by** *auto*
have *CLB*: *carrier-vec n = span (set (Ls @ Bs))*
using *lin-Ls-Bs len-Ls-Bs lin-indpt-list-length-eq-n* **by** *blast*
also have $\dots \subseteq \text{span } Y$
by (*rule span-is-monotone, insert LX, auto simp: Y-def*)
finally have *span*: *span Y = carrier-vec n* **using** *Y* **by** *auto*
have *finY*: *finite Y* **using** *finX unfolding Y-def* **by** *auto*
from *farkas-minkowsky-weyl-theorem-1-full-dim*[*OF Y finY span*]
obtain *A nr* **where** *A*: *A \in carrier-mat nr n* **and** *YA*: *cone Y = polyhedral-cone*
A
and *Y-Ints*: *is-det-bound db \implies Y \subseteq $\mathbb{Z}_v \cap \text{Bounded-vec} (?oi (max 1 Bnd))$*
 $\implies A \in \mathbb{Z}_m \cap ?Bndm$ **by** *blast*

have fin : $finite (\{row\ A\ i\ |\ i.\ i < nr\} \cup set\ Ns \cup uminus\ 'set\ Ns)$ **by** *auto*
from $finite-list[OF\ this]$ **obtain** rs **where** $rs-def$: $set\ rs = \{row\ A\ i\ |\ i.\ i < nr\}$
 $\cup\ set\ Ns \cup uminus\ 'set\ Ns$ **by** *auto*
from $A\ N$ **have** rs : $set\ rs \subseteq carrier-vec\ n$ **unfolding** $rs-def$ **by** *auto*
let $?m = length\ rs$
define B **where** $B = mat-of-rows\ n\ rs$
have B : $B \in carrier-mat\ ?m\ n$ **unfolding** $B-def$ **by** *auto*
show $?thesis$
proof (*intro exI conjI impI, rule B*)
have id : $(\forall r \in \{rs\ !\ i\ |\ i.\ i < ?m\}.\ P\ r) = (\forall r < ?m.\ P\ (rs\ !\ r))$ **for** P **by** *auto*
have $polyhedral-cone\ B = \{x \in carrier-vec\ n.\ B\ *_v\ x \leq 0_v\ ?m\}$ **unfolding**
 $polyhedral-cone-def$
using B **by** *auto*
also **have** $\dots = \{x \in carrier-vec\ n.\ \forall i < ?m.\ row\ B\ i \cdot x \leq 0\}$
unfolding $less-eq-vec-def$ **using** B **by** *auto*
also **have** $\dots = \{x \in carrier-vec\ n.\ \forall r \in set\ rs.\ r \cdot x \leq 0\}$ **using** rs
unfolding $set-conv-nth\ id$
by (*intro Collect-cong conj-cong refl all-cong arg-cong[of - - $\lambda x.\ x \cdot - \leq 0$],*
auto simp: B-def)
also **have** $\dots = \{x \in carrier-vec\ n.\ \forall i < nr.\ row\ A\ i \cdot x \leq 0\}$
 $\cap \{x \in carrier-vec\ n.\ \forall w \in set\ Ns \cup uminus\ 'set\ Ns.\ w \cdot x \leq 0\}$
unfolding $rs-def$ **by** *blast*
also **have** $\{x \in carrier-vec\ n.\ \forall i < nr.\ row\ A\ i \cdot x \leq 0\} = polyhedral-cone\ A$
unfolding $polyhedral-cone-def$ **using** A **by** (*auto simp: less-eq-vec-def*)
also **have** $\dots = cone\ Y$ **unfolding** $YA\ ..$
also **have** $\{x \in carrier-vec\ n.\ \forall w \in set\ Ns \cup uminus\ 'set\ Ns.\ w \cdot x \leq 0\}$
 $= \{x \in carrier-vec\ n.\ \forall w \in set\ Ns.\ w \cdot x = 0\}$
(is ?l = ?r)
proof
show $?r \subseteq ?l$ **using** N **by** *auto*
{
fix $x\ w$
assume $x \in ?l\ w \in set\ Ns$
with N **have** x : $x \in carrier-vec\ n$ **and** w : $w \in carrier-vec\ n$
and one : $w \cdot x \leq 0$ **and** two : $(-w) \cdot x \leq 0$ **by** *auto*
from two **have** $w \cdot x \geq 0$
by (*subst (asm) scalar-prod-uminus-left, insert w x, auto*)
with one **have** $w \cdot x = 0$ **by** *auto*
}
thus $?l \subseteq ?r$ **by** *blast*
qed
finally **have** $polyhedral-cone\ B = cone\ Y \cap \{x \in carrier-vec\ n.\ \forall w \in set\ Ns.\ w$
 $\cdot x = 0\}$.
also **have** $\dots = cone\ X$ **unfolding** $Y-def$
by (*rule orthogonal-cone(1)[OF X N finX spanLN ortho refl spanL LX lin-Ls-Bs*
len-Ls-Bs])
finally **show** $cone\ X = polyhedral-cone\ B$..
assume $X-I$: $X \subseteq \mathbb{Z}_v \cap Bounded-vec\ (?oi\ Bnd)$ **and** db : *is-det-bound db*
with LX **have** $set\ Ls \subseteq \mathbb{Z}_v \cap Bounded-vec\ (?oi\ Bnd)$ **by** *auto*

```

from Ns-bnd[OF db this] have N-I-Bnd: set Ns  $\subseteq \mathbb{Z}_v \cap ?Bnd$  by auto
from lin-Ls-Bs have linLs: lin-indpt-list Ls unfolding lin-indpt-list-def
using subset-li-is-li[of - set Ls] by auto
from X-I LX have L-I: set Ls  $\subseteq \mathbb{Z}_v$  by auto
have Y-I: Y  $\subseteq \mathbb{Z}_v \cap \text{Bounded-vec } (?oi (max 1 Bnd))$  unfolding Y-def using
X-I order.trans[OF BU unit-vec-int-bounds, of Bnd]
  Bounded-vec-mono[of ?oi Bnd ?oi (max 1 Bnd)] by auto
from Y-Ints[OF db Y-I]
have A-I-Bnd: set (rows A)  $\subseteq \mathbb{Z}_v \cap ?Bnd$  by auto
have set (rows B) = set (rows (mat-of-rows n rs)) unfolding B-def by auto
also have  $\dots = \text{set } rs$  using rs by auto
also have  $\dots = \text{set (rows A)} \cup \text{set } Ns \cup \text{uminus ' set } Ns$  unfolding rs-def
rows-def using A by auto
also have  $\dots \subseteq \mathbb{Z}_v \cap ?Bnd$  using A-I-Bnd N-I-Bnd by auto
finally show  $B \in \mathbb{Z}_m \cap ?Bndm$  by simp
qed
qed

```

Now for the other direction.

lemma *farkas-minkowsky-weyl-theorem-2*:

```

assumes A: A  $\in \text{carrier-mat } nr$  n
shows  $\exists X. X \subseteq \text{carrier-vec } n \wedge \text{finite } X \wedge \text{polyhedral-cone } A = \text{cone } X$ 
   $\wedge (\text{is-det-bound } db \longrightarrow A \in \mathbb{Z}_m \cap \text{Bounded-mat } (\text{of-int } Bnd) \longrightarrow X \subseteq \mathbb{Z}_v \cap$ 
Bounded-vec (of-int (db (n-1) (max 1 Bnd)))))
proof -
let ?oi = of-int :: int  $\Rightarrow$  'a
let ?rows-A = {row A i | i. i < nr}
let ?Bnd = Bounded-vec (?oi (db (n-1) (max 1 Bnd)))
have rows-A-n: ?rows-A  $\subseteq \text{carrier-vec } n$  using row-carrier-vec A by auto
hence  $\exists mr B. B \in \text{carrier-mat } mr$  n  $\wedge \text{cone } ?rows-A = \text{polyhedral-cone } B$ 
   $\wedge (\text{is-det-bound } db \longrightarrow ?rows-A \subseteq \mathbb{Z}_v \cap \text{Bounded-vec } (?oi Bnd) \longrightarrow \text{set (rows
B) \subseteq \mathbb{Z}_v \cap ?Bnd)
using farkas-minkowsky-weyl-theorem-1[of ?rows-A] by auto
then obtain mr B
where mr: B  $\in \text{carrier-mat } mr$  n and B: cone ?rows-A = polyhedral-cone B
and Bnd: is-det-bound db  $\implies ?rows-A \subseteq \mathbb{Z}_v \cap \text{Bounded-vec } (?oi Bnd) \implies$ 
set (rows B) \subseteq \mathbb{Z}_v \cap ?Bnd
by blast
let ?rows-B = {row B i | i. i < mr}
have rows-B: ?rows-B  $\subseteq \text{carrier-vec } n$  using mr by auto
have cone ?rows-B = polyhedral-cone A
proof
have ?rows-B  $\subseteq \text{polyhedral-cone } A$ 
proof
fix r
assume  $r \in ?rows-B$ 
then obtain j where  $r = \text{row } B j$  and  $j: j < mr$  by auto
then have rn:  $r \in \text{carrier-vec } n$  using mr row-carrier by auto
moreover have  $A *_v r \leq 0_v nr$  unfolding less-eq-vec-def$ 
```

```

proof (standard, unfold index-zero-vec)
  show dim-vec (A *_v r) = nr using A by auto
next
  show  $\forall i < nr. (A *_v r) \$ i \leq 0_v nr \$ i$ 
  proof (standard, rule impI)
    fix i
    assume i: i < nr
    then have row A i  $\in$  ?rows-A by auto
    then have row A i  $\in$  cone ?rows-A
      using set-in-cone rows-A-n by blast
    then have row A i  $\in$  polyhedral-cone B using B by auto
    then have Br: B *_v (row A i)  $\leq 0_v mr$ 
      unfolding polyhedral-cone-def using rows-A-n mr by auto

    then have (A *_v r) $ i = (row A i) . r using A i index-mult-mat-vec by
auto
    also have ... = r . (row A i)
      using comm-scalar-prod[OF - rn] row-carrier A by auto
    also have ... = (row B j) . (row A i) using r by auto
    also have ... = (B *_v (row A i)) $ j using index-mult-mat-vec mr j by
auto
    also have ...  $\leq 0$  using Br j unfolding less-eq-vec-def by auto
    also have ... = 0_v nr $ i using i by auto
    finally show (A *_v r) $ i  $\leq 0_v nr \$ i$  by auto
  qed
qed
then show r  $\in$  polyhedral-cone A
  unfolding polyhedral-cone-def
  using A rn by auto
qed
then show cone ?rows-B  $\subseteq$  polyhedral-cone A
  using cone-in-polyhedral-cone A by auto

next

show polyhedral-cone A  $\subseteq$  cone ?rows-B
proof (rule ccontr)
  assume  $\neg$  polyhedral-cone A  $\subseteq$  cone ?rows-B
  then obtain y where yA: y  $\in$  polyhedral-cone A
    and yB: y  $\notin$  cone ?rows-B by auto
  then have yn: y  $\in$  carrier-vec n unfolding polyhedral-cone-def by auto
  have finRB: finite ?rows-B by auto
  from farkas-minkowsky-weyl-theorem-1[OF rows-B finRB]
  obtain nr' A' where A': A'  $\in$  carrier-mat nr' n and cone: cone ?rows-B =
polyhedral-cone A'
    by blast
  from yB[unfolded cone polyhedral-cone-def] yn A'
  have  $\neg$  (A' *_v y  $\leq 0_v nr'$ ) by auto
  then obtain i where i: i < nr' and row A' i . y > 0

```

```

    unfolding less-eq-vec-def using A' yn by auto
  define w where w = row A' i
  have w: w ∈ carrier-vec n using i A' yn unfolding w-def by auto
  from ⟨row A' i · y > 0⟩ comm-scalar-prod[OF w yn] have wy: w · y > 0 y ·
w > 0 unfolding w-def by auto
  {
    fix b
    assume b ∈ ?rows-B
    hence b ∈ cone ?rows-B using set-in-cone[OF rows-B] by auto
    from this[unfolded cone polyhedral-cone-def] A'
    have b: b ∈ carrier-vec n and A' *v b ≤ 0v nr' by auto
    from this(2)[unfolded less-eq-vec-def, THEN conjunct2, rule-format, of i]
    have w · b ≤ 0 unfolding w-def using i A' by auto
    hence b · w ≤ 0 using comm-scalar-prod[OF b w] by auto
  }
  hence wA: w ∈ cone ?rows-A unfolding B polyhedral-cone-def using mr w
  by (auto simp: less-eq-vec-def)
  from wy have yw: -y · w < 0
  by (subst scalar-prod-uminus-left, insert yn w, auto)
  have ?rows-A ⊆ carrier-vec n finite ?rows-A using assms by auto
  from fundamental-theorem-of-linear-inequalities-A-imp-D[OF this wA, un-
folded not-ex,
  rule-format, of -y ] yn yw
  obtain i where i: i < nr and - y · row A i < 0 by auto
  hence y · row A i > 0 by (subst (asm) scalar-prod-uminus-left, insert i assms
yn, auto)
  hence row A i · y > 0 using comm-scalar-prod[OF - yn, of row A i] i assms
yn by auto
  with yA show False unfolding polyhedral-cone-def less-eq-vec-def using i
assms by auto
qed
qed
moreover have ?rows-B ⊆ carrier-vec n
  using row-carrier-vec mr by auto
moreover have finite ?rows-B by auto
moreover {
  have rA: set (rows A) = ?rows-A using A unfolding rows-def by auto
  have rB: set (rows B) = ?rows-B using mr unfolding rows-def by auto
  assume A ∈ ℤm ∩ Bounded-mat (?oi Bnd) and db: is-det-bound db
  hence set (rows A) ⊆ ℤv ∩ Bounded-vec (?oi Bnd) by simp
  from Bnd[OF db this[unfolded rA]]
  have ?rows-B ⊆ ℤv ∩ ?Bnd unfolding rA rB .
}
ultimately show ?thesis by blast
qed

```

lemma farkas-minkowsky-weyl-theorem:

$(\exists X. X \subseteq \text{carrier-vec } n \wedge \text{finite } X \wedge P = \text{cone } X)$
 $\longleftrightarrow (\exists A \text{ nr}. A \in \text{carrier-mat } \text{nr } n \wedge P = \text{polyhedral-cone } A)$

```

using farkas-minkowsky-weyl-theorem-1 farkas-minkowsky-weyl-theorem-2 by metis
end
end

```

14 The Decomposition Theorem

This theory contains a proof of the fact, that every polyhedron can be decomposed into a convex hull of a finite set of points + a finitely generated cone, including bounds on the numbers that are required in the decomposition. We further prove the inverse direction of this theorem (without bounds) and as a corollary, we derive that a polyhedron is bounded iff it is the convex hull of finitely many points, i.e., a polytope.

```

theory Decomposition-Theorem

```

```

imports

```

```

  Farkas-Minkowsky-Weyl

```

```

  Convex-Hull

```

```

begin

```

```

context gram-schmidt

```

```

begin

```

```

definition polytope  $P = (\exists V. V \subseteq \text{carrier-vec } n \wedge \text{finite } V \wedge P = \text{convex-hull } V)$ 

```

```

definition polyhedron  $A \ b = \{x \in \text{carrier-vec } n. A *_v x \leq b\}$ 

```

```

lemma polyhedra-are-convex:

```

```

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

```

```

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

```

```

  and  $P: P = \text{polyhedron } A \ b$ 

```

```

  shows  $\text{convex } P$ 

```

```

proof (intro convexI)

```

```

  show  $P_{\text{carr}}: P \subseteq \text{carrier-vec } n$  using  $\text{assms}$  unfolding  $\text{polyhedron-def}$  by auto

```

```

  fix  $a :: 'a$  and  $x \ y$ 

```

```

  assume  $xy: x \in P \ y \in P$  and  $a: 0 \leq a \ a \leq 1$ 

```

```

  from  $xy[\text{unfolded } P \ \text{polyhedron-def}]$ 

```

```

  have  $x: x \in \text{carrier-vec } n$  and  $y: y \in \text{carrier-vec } n$  and  $le: A *_v x \leq b \ A *_v y \leq b$  by auto

```

```

  show  $a *_v x + (1 - a) *_v y \in P$  unfolding  $P \ \text{polyhedron-def}$ 

```

```

  proof (intro CollectI conjI)

```

```

    from  $x$  have  $ax: a *_v x \in \text{carrier-vec } n$  by auto

```

```

    from  $y$  have  $ay: (1 - a) *_v y \in \text{carrier-vec } n$  by auto

```

```

    show  $a *_v x + (1 - a) *_v y \in \text{carrier-vec } n$  using  $ax \ ay$  by auto

```

```

    show  $A *_v (a *_v x + (1 - a) *_v y) \leq b$ 

```

```

    proof (intro lesseq-vecI[OF - b])

```

```

      show  $A *_v (a *_v x + (1 - a) *_v y) \in \text{carrier-vec } nr$  using  $A \ x \ y$  by auto

```

```

      fix  $i$ 

```

```

      assume  $i: i < nr$ 

```

```

from lesseq-vecD[OF b le(1) i] lesseq-vecD[OF b le(2) i]
have le: (A *v x) $ i ≤ b $ i (A *v y) $ i ≤ b $ i by auto
have (A *v (a ·v x + (1 - a) ·v y)) $ i = a * (A *v x) $ i + (1 - a) * (A
*v y) $ i
  using A x y i by (auto simp: scalar-prod-add-distrib[of - n])
also have ... ≤ a * b $ i + (1 - a) * b $ i
  by (rule add-mono; rule mult-left-mono, insert le a, auto)
also have ... = b $ i by (auto simp: field-simps)
finally show (A *v (a ·v x + (1 - a) ·v y)) $ i ≤ b $ i .
qed
qed
qed
end

```

```

locale gram-schmidt-m = n: gram-schmidt n f-ty + m: gram-schmidt m f-ty
for n m :: nat and f-ty
begin

```

```

lemma vec-first-lincomb-list:

```

```

  assumes Xs: set Xs ⊆ carrier-vec n
  and nm: m ≤ n
  shows vec-first (n.lincomb-list c Xs) m =
    m.lincomb-list c (map (λ v. vec-first v m) Xs)
  using Xs
proof (induction Xs arbitrary: c)
  case Nil
  show ?case by (simp add: nm)
next
  case (Cons x Xs)
  from Cons.prem1 have x: x ∈ carrier-vec n and Xs: set Xs ⊆ carrier-vec n by
  auto

  have vec-first (n.lincomb-list c (x # Xs)) m =
    vec-first (c 0 ·v x + n.lincomb-list (c ∘ Suc) Xs) m by auto
  also have ... = vec-first (c 0 ·v x) m + vec-first (n.lincomb-list (c ∘ Suc) Xs)
  m
  using vec-first-add[of m c 0 ·v x] x n.lincomb-list-carrier[OF Xs, of c ∘ Suc]
  nm
  by simp
  also have vec-first (c 0 ·v x) m = c 0 ·v vec-first x m
  using vec-first-smult[OF nm, of x c 0] Cons.prem1 by auto
  also have vec-first (n.lincomb-list (c ∘ Suc) Xs) m =
    m.lincomb-list (c ∘ Suc) (map (λ v. vec-first v m) Xs)
  using Cons by simp
  also have c 0 ·v vec-first x m + ... =
    m.lincomb-list c (map (λ v. vec-first v m) (x # Xs))

```

by *simp*
 finally show *?case* by *auto*
 qed

lemma *convex-hull-next-dim*:

assumes $n = m + 1$

and $X: X \subseteq \text{carrier-vec } n$

and *finite* X

and $Xm1: \forall y \in X. y \$ m = 1$

and $y\text{-dim}: y \in \text{carrier-vec } n$

and $y: y \$ m = 1$

shows $(\text{vec-first } y \ m \in m.\text{convex-hull } \{\text{vec-first } y \ m \mid y. y \in X\}) = (y \in n.\text{cone } X)$

proof –

from $\langle \text{finite } X \rangle$ obtain Xs where $Xs: X = \text{set } Xs$ using *finite-list* by *auto*

let $?Y = \{\text{vec-first } y \ m \mid y. y \in X\}$

let $?Ys = \text{map } (\lambda y. \text{vec-first } y \ m) \ Xs$

have $Ys: ?Y = \text{set } ?Ys$ using Xs by *auto*

define x where $x = \text{vec-first } y \ m$

{
 have $y = \text{vec-first } y \ m @_v \text{vec-last } y \ 1$
 using $\langle n = m + 1 \rangle$ *vec-first-last-append* $y\text{-dim}$ by *auto*
 also have $\text{vec-last } y \ 1 = \text{vec-of-scal } (\text{vec-last } y \ 1 \ \$ \ 0)$
 using *vec-of-scal-dim-1* [of $\text{vec-last } y \ 1$] by *simp*
 also have $\text{vec-last } y \ 1 \ \$ \ 0 = y \ \$ \ m$
 using $y\text{-dim}$ $\langle n = m + 1 \rangle$ *vec-last-index* [of $y \ m \ 1 \ 0$] by *auto*
 finally have $y = x @_v \text{vec-of-scal } 1$ unfolding $x\text{-def}$ using y by *simp*
 } note $xy = \text{this}$
 {
 assume $y \in n.\text{cone } X$
 then obtain c where $x: n.\text{nonneg-lincomb } c \ X \ y$
 using *n.cone-iff-finite-cone* [OF X] $\langle \text{finite } X \rangle$
 unfolding *n.finite-cone-def* by *auto*

have $1 = y \ \$ \ m$ by (*simp add: y*)

also have $y = n.\text{lincomb } c \ X$

using x unfolding *n.nonneg-lincomb-def* by *simp*

also have $\dots \ \$ \ m = (\sum x \in X. c \ x * x \ \$ \ m)$

using *n.lincomb-index* [OF - X] $\langle n = m + 1 \rangle$ by *simp*

also have $\dots = \text{sum } c \ X$

by (rule *n.R.finsum-restrict*, *auto*, rule *restrict-ext*, *simp add: Xm1*)

finally have $y \in n.\text{convex-hull } X$

unfolding *n.convex-hull-def* *n.convex-lincomb-def*

using $\langle \text{finite } X \rangle$ x by *auto*

}

moreover have $n.\text{convex-hull } X \subseteq n.\text{cone } X$

unfolding *n.convex-hull-def* *n.convex-lincomb-def* *n.finite-cone-def* *n.cone-def*

using $\langle \text{finite } X \rangle$ by *auto*

moreover have $n.\text{convex-hull } X = n.\text{convex-hull-list } Xs$
by (rule $n.\text{finite-convex-hull-iff-convex-hull-list}[OF \ X \ Xs]$)

moreover {
assume $y \in n.\text{convex-hull-list } Xs$
then obtain c **where** $c: n.\text{lincomb-list } c \ Xs = y$
and $c0: \forall i < \text{length } Xs. c \ i \geq 0$ **and** $c1: \text{sum } c \ \{0..<\text{length } Xs\} = 1$
unfolding $n.\text{convex-hull-list-def } n.\text{convex-lincomb-list-def}$
 $n.\text{nonneg-lincomb-list-def}$ **by** *fast*
have $m.\text{lincomb-list } c \ ?Ys = \text{vec-first } y \ m$
using $c \ \text{vec-first-lincomb-list}[of \ Xs \ c] \ X \ Xs \ \langle n = m + 1 \rangle$ **by** *simp*
hence $x \in m.\text{convex-hull-list } ?Ys$
unfolding $m.\text{convex-hull-list-def } m.\text{convex-lincomb-list-def}$
 $m.\text{nonneg-lincomb-list-def}$
using $x\text{-def } c0 \ c1 \ x\text{-def}$ **by** *auto*
} moreover {
assume $x \in m.\text{convex-hull-list } ?Ys$
then obtain c **where** $x: m.\text{lincomb-list } c \ ?Ys = x$
and $c0: \forall i < \text{length } Xs. c \ i \geq 0$
and $c1: \text{sum } c \ \{0..<\text{length } Xs\} = 1$
unfolding $m.\text{convex-hull-list-def } m.\text{convex-lincomb-list-def}$
 $m.\text{nonneg-lincomb-list-def}$ **by** *auto*

have $n.\text{lincomb-list } c \ Xs \ \$ \ m = (\sum j = 0..<\text{length } Xs. c \ j * Xs \ ! \ j \ \$ \ m)$
using $n.\text{lincomb-list-index}[of \ m \ Xs \ c] \ \langle n = m + 1 \rangle \ Xs \ X$ **by** *fastforce*
also have $\dots = \text{sum } c \ \{0..<\text{length } Xs\}$
apply(rule $n.R.\text{finsum-restrict}$, *auto*, rule *restrict-ext*)
by (*simp add: Xm1 Xs*)
also have $\dots = 1$ **by** (rule $c1$)
finally have $\text{vec-last } (n.\text{lincomb-list } c \ Xs) \ 1 \ \$ \ 0 = 1$
using $\text{vec-of-scal-dim-1 } \text{vec-last-index}[of \ n.\text{lincomb-list } c \ Xs \ m \ 1 \ 0]$
 $n.\text{lincomb-list-carrier } Xs \ X \ \langle n = m + 1 \rangle$ **by** *simp*
hence $\text{vec-last } (n.\text{lincomb-list } c \ Xs) \ 1 = \text{vec-of-scal } 1$
using vec-of-scal-dim-1 **by** *auto*

moreover have $\text{vec-first } (n.\text{lincomb-list } c \ Xs) \ m = x$
using $\text{vec-first-lincomb-list} \ \langle n = m + 1 \rangle \ Xs \ X \ x$ **by** *auto*

moreover have $n.\text{lincomb-list } c \ Xs =$
 $\text{vec-first } (n.\text{lincomb-list } c \ Xs) \ m \ @_v \ \text{vec-last } (n.\text{lincomb-list } c \ Xs) \ 1$
using $\text{vec-first-last-append } Xs \ X \ n.\text{lincomb-list-carrier} \ \langle n = m + 1 \rangle$ **by** *auto*

ultimately have $n.\text{lincomb-list } c \ Xs = y$ **using** xy **by** *simp*

hence $y \in n.\text{convex-hull-list } Xs$
unfolding $n.\text{convex-hull-list-def } n.\text{convex-lincomb-list-def}$
 $n.\text{nonneg-lincomb-list-def}$ **using** $c0 \ c1$ **by** *blast*
}

moreover have $m.\text{convex-hull } ?Y = m.\text{convex-hull-list } ?Ys$
using $m.\text{finite-convex-hull-iff-convex-hull-list}[OF \ - \ Ys]$ **by** *fastforce*

ultimately show *?thesis unfolding x-def by blast*
qed

lemma *cone-next-dim*:

assumes $n = m + 1$

and $X: X \subseteq \text{carrier-vec } n$

and *finite X*

and $Xm0: \forall y \in X. y \$ m = 0$

and $y\text{-dim}: y \in \text{carrier-vec } n$

and $y: y \$ m = 0$

shows $(\text{vec-first } y \ m \in m.\text{cone } \{\text{vec-first } y \ m \mid y. y \in X\}) = (y \in n.\text{cone } X)$

proof –

from $\langle \text{finite } X \rangle$ obtain Xs where $Xs: X = \text{set } Xs$ using *finite-list by auto*

let $?Y = \{\text{vec-first } y \ m \mid y. y \in X\}$

let $?Ys = \text{map } (\lambda y. \text{vec-first } y \ m) \ Xs$

have $Ys: ?Y = \text{set } ?Ys$ using Xs by *auto*

define x where $x = \text{vec-first } y \ m$

{

have $y = \text{vec-first } y \ m @_v \text{vec-last } y \ 1$

using $\langle n = m + 1 \rangle \text{vec-first-last-append } y\text{-dim}$ by *auto*

also have $\text{vec-last } y \ 1 = \text{vec-of-scal } (\text{vec-last } y \ 1 \ \$ \ 0)$

using $\text{vec-of-scal-dim-1}[\text{of } \text{vec-last } y \ 1]$ by *simp*

also have $\text{vec-last } y \ 1 \ \$ \ 0 = y \ \$ \ m$

using $y\text{-dim } \langle n = m + 1 \rangle \text{vec-last-index}[\text{of } y \ m \ 1 \ 0]$ by *auto*

finally have $y = x @_v \text{vec-of-scal } 0$ unfolding $x\text{-def}$ using y by *simp*

} note $xy = \text{this}$

have $n.\text{cone } X = n.\text{cone-list } Xs$

using $n.\text{cone-iff-finite-cone}[\text{OF } X \ \langle \text{finite } X \rangle] \ n.\text{finite-cone-iff-cone-list}[\text{OF } X \ Xs]$

by *simp*

moreover {

assume $y \in n.\text{cone-list } Xs$

then obtain c where $y: n.\text{lincomb-list } c \ Xs = y$ and $c: \forall i < \text{length } Xs. c \ i \geq 0$

unfolding $n.\text{cone-list-def } n.\text{nonneg-lincomb-list-def}$ by *blast*

from y have $m.\text{lincomb-list } c \ ?Ys = x$

unfolding $x\text{-def}$

using $\text{vec-first-lincomb-list } Xs \ X \ \langle n = m + 1 \rangle$ by *auto*

hence $x \in m.\text{cone-list } ?Ys$ using c

unfolding $m.\text{cone-list-def } m.\text{nonneg-lincomb-list-def}$ by *auto*

} moreover {

assume $x \in m.\text{cone-list } ?Ys$

then obtain c where $x: m.\text{lincomb-list } c \ ?Ys = x$ and $c: \forall i < \text{length } Xs. c \ i \geq 0$

unfolding $m.\text{cone-list-def } m.\text{nonneg-lincomb-list-def}$ by *auto*

have $\text{vec-last } (n.\text{lincomb-list } c \ Xs) \ 1 \ \$ \ 0 = n.\text{lincomb-list } c \ Xs \ \$ \ m$

```

    using ⟨n = m + 1⟩ n.lincomb-list-carrier X Xs vec-last-index[of - m 1 0]
    by auto
  also have ... = 0
    using n.lincomb-list-index[of m Xs c] Xs X ⟨n = m + 1⟩ Xm0 by simp
  also have ... = vec-last y 1 $ 0
    using y y-dim ⟨n = m + 1⟩ vec-last-index[of y m 1 0] by auto
  finally have vec-last (n.lincomb-list c Xs) 1 = vec-last y 1 by fastforce

  moreover have vec-first (n.lincomb-list c Xs) m = x
    using vec-first-lincomb-list[of Xs c] x X Xs ⟨n = m + 1⟩
    unfolding x-def by simp

  ultimately have n.lincomb-list c Xs = y unfolding x-def
    using vec-first-last-append[of - m 1] ⟨n = m + 1⟩ y-dim
    n.lincomb-list-carrier[of Xs c] Xs X
    by metis
  hence y ∈ n.cone-list Xs
    unfolding n.cone-list-def n.nonneg-lincomb-list-def using c by blast
}
moreover have m.cone-list ?Ys = m.cone ?Y
  using m.finite-cone-iff-cone-list[OF - Ys] m.cone-iff-finite-cone[of ?Y]
  ⟨finite X⟩ by force
ultimately show ?thesis unfolding x-def by blast
qed

end

context gram-schmidt
begin

lemma decomposition-theorem-polyhedra-1:
  assumes A: A ∈ carrier-mat nr n
    and b: b ∈ carrier-vec nr
    and P: P = polyhedron A b
  shows ∃ Q X. X ⊆ carrier-vec n ∧ finite X ∧
    Q ⊆ carrier-vec n ∧ finite Q ∧
    P = convex-hull Q + cone X ∧
    (is-det-bound db ⟶ A ∈ ℤm ∩ Bounded-mat (of-int Bnd) ⟶ b ∈ ℤv ∩
    Bounded-vec (of-int Bnd) ⟶
    X ⊆ ℤv ∩ Bounded-vec (of-int (db n (max 1 Bnd))))
    ∧ Q ⊆ Bounded-vec (of-int (db n (max 1 Bnd))))
proof -
  let ?oi = of-int :: int ⇒ 'a

  interpret next-dim: gram-schmidt n + 1 TYPE ('a).
  interpret gram-schmidt-m n + 1 n TYPE ('a).

  from P[unfolded polyhedron-def] have P ⊆ carrier-vec n by auto

```

```

have mcb: mat-of-col  $(-b) \in \text{carrier-mat } nr \ 1$  using b by auto
define M where  $M = (A \ @_c \ \text{mat-of-col} \ (-b)) \ @_r \ (0_m \ 1 \ n \ @_c \ -1_m \ 1)$ 
have M-top:  $A \ @_c \ \text{mat-of-col} \ (-b) \in \text{carrier-mat } nr \ (n + 1)$ 
  by (rule carrier-append-cols[OF A mcb])
have M-bottom:  $(0_m \ 1 \ n \ @_c \ -1_m \ 1) \in \text{carrier-mat } 1 \ (n + 1)$ 
  by (rule carrier-append-cols, auto)
have M-dim:  $M \in \text{carrier-mat } (nr + 1) \ (n + 1)$ 
  unfolding M-def
  by (rule carrier-append-rows[OF M-top M-bottom])

{
  fix x :: 'a vec fix t assume x:  $x \in \text{carrier-vec } n$ 
  have  $x \ @_v \ \text{vec-of-scal } t \in \text{next-dim.polyhedral-cone } M =$ 
     $(A \ *_v \ x - t \ \cdot_v \ b \leq 0_v \ nr \wedge t \geq 0)$ 
  proof -
    let ?y =  $x \ @_v \ \text{vec-of-scal } t$ 
    have y:  $?y \in \text{carrier-vec } (n + 1)$  using x by(simp del: One-nat-def)
    have  $?y \in \text{next-dim.polyhedral-cone } M =$ 
       $(M \ *_v \ ?y \leq 0_v \ (nr + 1))$ 
      unfolding next-dim.polyhedral-cone-def using y M-dim by auto
    also have  $0_v \ (nr + 1) = 0_v \ nr \ @_v \ 0_v \ 1$  by auto
    also have  $M \ *_v \ ?y \leq 0_v \ nr \ @_v \ 0_v \ 1 =$ 
       $((A \ @_c \ \text{mat-of-col} \ (-b)) \ *_v \ ?y \leq 0_v \ nr \wedge$ 
       $(0_m \ 1 \ n \ @_c \ -1_m \ 1) \ *_v \ ?y \leq 0_v \ 1)$ 
      unfolding M-def
      by (intro append-rows-le[OF M-top M-bottom - y], auto)
    also have  $(A \ @_c \ \text{mat-of-col} \ (-b)) \ *_v \ ?y =$ 
       $A \ *_v \ x + \text{mat-of-col} \ (-b) \ *_v \ \text{vec-of-scal } t$ 
      by (rule mat-mult-append-cols[OF A - x],
      auto simp add: b simp del: One-nat-def)
    also have  $\text{mat-of-col} \ (-b) \ *_v \ \text{vec-of-scal } t = t \ \cdot_v \ (-b)$ 
      by(rule mult-mat-of-row-vec-of-scal)
    also have  $A \ *_v \ x + t \ \cdot_v \ (-b) = A \ *_v \ x - t \ \cdot_v \ b$  by auto
    also have  $(0_m \ 1 \ n \ @_c \ -1_m \ 1) \ *_v \ (x \ @_v \ \text{vec-of-scal } t) =$ 
       $0_m \ 1 \ n \ *_v \ x + -1_m \ 1 \ *_v \ \text{vec-of-scal } t$ 
      by(rule mat-mult-append-cols, auto simp add: x simp del: One-nat-def)
    also have  $\dots = - \ \text{vec-of-scal } t$  using x by (auto simp del: One-nat-def)
    also have  $(\dots \leq 0_v \ 1) = (t \geq 0)$  unfolding less-eq-vec-def by auto
    finally show  $(?y \in \text{next-dim.polyhedral-cone } M) =$ 
       $(A \ *_v \ x - t \ \cdot_v \ b \leq 0_v \ nr \wedge t \geq 0)$  by auto

  qed
} note M-cone-car = this
from next-dim.farkas-minkowsky-weyl-theorem-2[OF M-dim, of db max 1 Bnd]
obtain X where  $X: \text{next-dim.polyhedral-cone } M = \text{next-dim.cone } X$  and
  fin-X: finite X and X-carrier:  $X \subseteq \text{carrier-vec } (n+1)$ 
and Bnd: is-det-bound db  $\implies M \in \mathbf{Z}_m \cap \text{Bounded-mat } (?oi \ (\text{max } 1 \ \text{Bnd})) \implies$ 
   $X \subseteq \mathbf{Z}_v \cap \text{Bounded-vec } (?oi \ (\text{db } n \ (\text{max } 1 \ \text{Bnd})))$ 
by auto
let ?f =  $\lambda x. \text{if } x \ \$ \ n = 0 \ \text{then } 1 \ \text{else } 1 / (x \ \$ \ n)$ 

```

```

define  $Y$  where  $Y = \{?f\ x \cdot_v\ x \mid x. x \in X\}$ 
have finite  $Y$  unfolding  $Y$ -def using  $fin$ - $X$  by auto
have  $Y$ -carrier:  $Y \subseteq carrier$ -vec  $(n+1)$  unfolding  $Y$ -def using  $X$ -carrier by
auto
have  $?f\ ' X \subseteq \{y. y > 0\}$ 
proof
  fix  $y$ 
  assume  $y \in ?f\ ' X$ 
  then obtain  $x$  where  $x: x \in X$  and  $y: y = ?f\ x$  by auto
  show  $y \in \{y. y > 0\}$ 
  proof cases
    assume  $x\ \$\ n = 0$ 
    thus  $y \in \{y. y > 0\}$  using  $y$  by auto
  next
    assume  $P: x\ \$\ n \neq 0$ 
    have  $x = vec$ -first  $x\ n\ @_v\ vec$ -last  $x\ 1$ 
      using  $X$ -carrier  $vec$ -first-last-append by auto
    also have  $vec$ -last  $x\ 1 = vec$ -of-scal  $(vec$ -last  $x\ 1\ \$\ 0)$  by auto
    also have  $vec$ -last  $x\ 1\ \$\ 0 = x\ \$\ n$ 
      using  $X$ -carrier unfolding  $vec$ -last-def by auto
    finally have  $x = vec$ -first  $x\ n\ @_v\ vec$ -of-scal  $(x\ \$\ n)$  by auto
    moreover have  $x \in next$ -dim.polyhedral-cone  $M$ 
      using  $X$   $X$ -carrier  $next$ -dim.set-in-cone by auto
    ultimately have  $x\ \$\ n \geq 0$  using  $M$ -cone-car  $vec$ -first-carrier by metis
    hence  $x\ \$\ n > 0$  using  $P$  by auto
    thus  $y \in \{y. y > 0\}$  using  $y$  by auto
  qed
qed
hence  $Y: next$ -dim.cone  $Y = next$ -dim.polyhedral-cone  $M$  unfolding  $Y$ -def
  using  $next$ -dim.cone-smult-basis[ $OF\ X$ -carrier]  $X$  by auto
define  $Y0$  where  $Y0 = \{v \in Y. v\ \$\ n = 0\}$ 
define  $Y1$  where  $Y1 = Y - Y0$ 
have  $Y0$ -carrier:  $Y0 \subseteq carrier$ -vec  $(n + 1)$  and  $Y1$ -carrier:  $Y1 \subseteq carrier$ -vec
 $(n + 1)$ 
  unfolding  $Y0$ -def  $Y1$ -def using  $Y$ -carrier by auto
have finite  $Y0$  and finite  $Y1$ 
  unfolding  $Y0$ -def  $Y1$ -def using  $\langle finite\ Y \rangle$  by auto

have  $Y1: \bigwedge y. y \in Y1 \implies y\ \$\ n = 1$ 
proof -
  fix  $y$  assume  $y: y \in Y1$ 
  hence  $y \in Y$  unfolding  $Y1$ -def by auto
  then obtain  $x$  where  $x \in X$  and  $x: y = ?f\ x \cdot_v\ x$  unfolding  $Y$ -def by auto
  then have  $x\ \$\ n \neq 0$  using  $x\ y\ Y1$ -def  $Y0$ -def by auto
  then have  $y = 1 / (x\ \$\ n) \cdot_v\ x$  using  $x$  by auto
  then have  $y\ \$\ n = 1 / (x\ \$\ n) * x\ \$\ n$  using  $X$ -carrier  $\langle x \in X \rangle$  by auto
  thus  $y\ \$\ n = 1$  using  $\langle x\ \$\ n \neq 0 \rangle$  by auto
qed

```

```

let ?Z0 = {vec-first y n | y. y ∈ Y0}
let ?Z1 = {vec-first y n | y. y ∈ Y1}
show ?thesis
proof (intro exI conjI impI)
  show ?Z0 ⊆ carrier-vec n by auto
  show ?Z1 ⊆ carrier-vec n by auto
  show finite ?Z0 using ⟨finite Y0⟩ by auto
  show finite ?Z1 using ⟨finite Y1⟩ by auto
  show P = convex-hull ?Z1 + cone ?Z0
proof -
  {
    fix x
    assume x ∈ P
    hence xn: x ∈ carrier-vec n and A *v x ≤ b
      using P unfolding polyhedron-def by auto
    hence A *v x - 1 ·v b ≤ 0v nr
    using vec-le-iff-diff-le-0 A b carrier-vecD mult-mat-vec-carrier one-smult-vec
      by metis
    hence x @v vec-of-scal 1 ∈ next-dim.polyhedral-cone M
      using M-cone-car[OF xn] by auto
    hence x @v vec-of-scal 1 ∈ next-dim.cone Y using Y by auto
    hence x @v vec-of-scal 1 ∈ next-dim.finite-cone Y
      using next-dim.cone-iff-finite-cone[OF Y-carrier ⟨finite Y⟩] by auto
    then obtain c where c: next-dim.nonneg-lincomb c Y (x @v vec-of-scal 1)
      unfolding next-dim.finite-cone-def using ⟨finite Y⟩ by auto
    let ?y = next-dim.lincomb c Y1
    let ?z = next-dim.lincomb c Y0
    have y-dim: ?y ∈ carrier-vec (n + 1) and z-dim: ?z ∈ carrier-vec (n + 1)
      unfolding next-dim.nonneg-lincomb-def
      using Y0-carrier Y1-carrier next-dim.lincomb-closed by simp-all
    hence yz-dim: ?y + ?z ∈ carrier-vec (n + 1) by auto
    have x @v vec-of-scal 1 = next-dim.lincomb c Y
      using c unfolding next-dim.nonneg-lincomb-def by auto
    also have Y = Y1 ∪ Y0 unfolding Y1-def using Y0-def by blast
    also have next-dim.lincomb c (Y1 ∪ Y0) = ?y + ?z
      using next-dim.lincomb-union2[of Y1 Y0]
      ⟨finite Y0⟩ ⟨finite Y⟩ Y0-carrier Y-carrier
      unfolding Y1-def by fastforce
    also have ?y + ?z = vec-first (?y + ?z) n @v vec-last (?y + ?z) 1
      using vec-first-last-append[of ?y + ?z n 1] add-carrier-vec yz-dim
      by simp
    also have vec-last (?y + ?z) 1 = vec-of-scal ((?y + ?z) $ n)
      using vec-of-scal-dim-1 vec-last-index[OF yz-dim, of 0] by auto
    finally have x @v vec-of-scal 1 =
      vec-first (?y + ?z) n @v vec-of-scal ((?y + ?z) $ n) by auto
    hence x = vec-first (?y + ?z) n and
      yz-last: vec-of-scal 1 = vec-of-scal ((?y + ?z) $ n)
      using append-vec-eq yz-dim xn by auto
    hence xyz: x = vec-first ?y n + vec-first ?z n
  }

```

using *vec-first-add*[of n $?y$ $?z$] *y-dim z-dim* **by** *simp*

have $1 = ((?y + ?z) \$ n)$ **using** *yz-last index-vec-of-scal*
by (*metis (no-types, lifting)*)

hence $1 = ?y \$ n + ?z \$ n$ **using** *y-dim z-dim* **by** *auto*

moreover have $zn0: ?z \$ n = 0$
using *next-dim.lincomb-index*[*OF - Y0-carrier*] *Y0-def* **by** *auto*

ultimately have $yn1: 1 = ?y \$ n$ **by** *auto*

have *next-dim.nonneg-lincomb* c $Y1$ $?y$
using c *Y1-def*
unfolding *next-dim.nonneg-lincomb-def* **by** *auto*

hence $?y \in \text{next-dim.cone } Y1$
using *next-dim.cone-iff-finite-cone*[*OF Y1-carrier*] $\langle \text{finite } Y1 \rangle$
unfolding *next-dim.finite-cone-def* **by** *auto*

hence $y: \text{vec-first } ?y$ $n \in \text{convex-hull } ?Z1$
using *convex-hull-next-dim*[*OF - Y1-carrier* $\langle \text{finite } Y1 \rangle$ - *y-dim*] $Y1$ $yn1$
by *simp*

have *next-dim.nonneg-lincomb* c $Y0$ $?z$ **using** c *Y0-def*
unfolding *next-dim.nonneg-lincomb-def* **by** *blast*

hence $?z \in \text{next-dim.cone } Y0$
using $\langle \text{finite } Y0 \rangle$ *next-dim.cone-iff-finite-cone*[*OF Y0-carrier* $\langle \text{finite } Y0 \rangle$]
unfolding *next-dim.finite-cone-def*
by *fastforce*

hence $z: \text{vec-first } ?z$ $n \in \text{cone } ?Z0$
using *cone-next-dim*[*OF - Y0-carrier* $\langle \text{finite } Y0 \rangle$ - - $zn0$] *Y0-def*
next-dim.lincomb-closed[*OF Y0-carrier*] **by** *blast*

from xyz y z **have** $x \in \text{convex-hull } ?Z1 + \text{cone } ?Z0$ **by** *blast*

} moreover {
fix x
assume $x \in \text{convex-hull } ?Z1 + \text{cone } ?Z0$
then obtain y z **where** $x = y + z$ **and** $y: y \in \text{convex-hull } ?Z1$
and $z: z \in \text{cone } ?Z0$ **by** (*auto elim: set-plus-elim*)

have $yn: y \in \text{carrier-vec } n$
using y *convex-hull-carrier*[*OF* $\langle ?Z1 \subseteq \text{carrier-vec } n \rangle$] **by** *blast*

hence $y @_v \text{vec-of-scal } 1 \in \text{carrier-vec } (n + 1)$
using *vec-of-scal-dim*(2) **by** *fast*

moreover have *vec-first* ($y @_v \text{vec-of-scal } 1$) $n \in \text{convex-hull } ?Z1$
using *vec-first-append*[*OF yn*] y **by** *auto*

moreover have ($y @_v \text{vec-of-scal } 1$) $\$ n = 1$ **using** yn **by** *simp*

ultimately have $y @_v \text{vec-of-scal } 1 \in \text{next-dim.cone } Y1$
using *convex-hull-next-dim*[*OF - Y1-carrier* $\langle \text{finite } Y1 \rangle$] $Y1$ **by** *blast*

hence $y\text{-cone}: y @_v \text{vec-of-scal } 1 \in \text{next-dim.cone } Y$
using *next-dim.cone-mono*[of $Y1$ Y] *Y1-def* **by** *blast*

have $zn: z \in \text{carrier-vec } n$ **using** z *cone-carrier*[of $?Z0$] **by** *fastforce*

hence $z @_v \text{vec-of-scal } 0 \in \text{carrier-vec } (n + 1)$

```

    using vec-of-scal-dim(2) by fast
  moreover have vec-first (z @v vec-of-scal 0) n ∈ cone ?Z0
    using vec-first-append[OF zn] z by auto
  moreover have (z @v vec-of-scal 0) $ n = 0 using zn by simp
  ultimately have z @v vec-of-scal 0 ∈ next-dim.cone Y0
    using cone-next-dim[OF - Y0-carrier ⟨finite Y0⟩] Y0-def by blast
  hence z-cone: z @v vec-of-scal 0 ∈ next-dim.cone Y
    using Y0-def next-dim.cone-mono[of Y0 Y] by blast

  have xn: x ∈ carrier-vec n using ⟨x = y + z⟩ yn zn by blast
  have x @v vec-of-scal 1 = (y @v vec-of-scal 1) + (z @v vec-of-scal 0)
    using ⟨x = y + z⟩ append-vec-add[OF yn zn]
  unfolding vec-of-scal-def by auto
  hence x @v vec-of-scal 1 ∈ next-dim.cone Y
    using next-dim.cone-elem-sum[OF Y-carrier y-cone z-cone] by simp
  hence A *v x - b ≤ 0v nr using M-cone-car[OF xn] Y by simp
  hence A *v x ≤ b using vec-le-iff-diff-le-0[of A *v x b]
    dim-mult-mat-vec[of A x] A by simp
  hence x ∈ P using P xn unfolding polyhedron-def by blast
}
ultimately show P = convex-hull ?Z1 + cone ?Z0 by blast
qed

let ?Bnd = db n (max 1 Bnd)
assume A ∈ ℤm ∩ Bounded-mat (?oi Bnd)
  b ∈ ℤv ∩ Bounded-vec (?oi Bnd)
  and db: is-det-bound db
  hence *: A ∈ ℤm A ∈ Bounded-mat (?oi Bnd) b ∈ ℤv b ∈ Bounded-vec (?oi
Bnd) by auto
  have elements-mat M ⊆ elements-mat A ∪ vec-set (-b) ∪ {0, -1}
    unfolding M-def
  unfolding elements-mat-append-rows[OF M-top M-bottom]
  unfolding elements-mat-append-cols[OF A mcb]
  by (subst elements-mat-append-cols, auto)
  also have ... ⊆ ℤ ∩ ({x. abs x ≤ ?oi Bnd} ∪ {0, -1})
    using *[unfolded Bounded-mat-elements-mat Ints-mat-elements-mat
Bounded-vec-vec-set Ints-vec-vec-set] by auto
  also have ... ⊆ ℤ ∩ ({x. abs x ≤ ?oi (max 1 Bnd)}) by (auto simp: of-int-max)
  finally have M ∈ ℤm M ∈ Bounded-mat (?oi (max 1 Bnd))
    unfolding Bounded-mat-elements-mat Ints-mat-elements-mat by auto
  hence M ∈ ℤm ∩ Bounded-mat (?oi (max 1 Bnd)) by blast
  from Bnd[OF db this]
  have XBnd: X ⊆ ℤv ∩ Bounded-vec (?oi ?Bnd) .
  {
  fix y
  assume y: y ∈ Y
  then obtain x where y: y = ?f x ·v x and xX: x ∈ X unfolding Y-def by
auto
  with ⟨X ⊆ carrier-vec (n+1)⟩ have x: x ∈ carrier-vec (n+1) by auto

```

```

from  $XBnd$   $xX$  have  $xI: x \in \mathbb{Z}_v$  and  $xB: x \in Bounded\text{-}vec$  ( $?oi$   $?Bnd$ ) by
auto
{
  assume  $y \$ n = 0$ 
  hence  $y = x$  unfolding  $y$  using  $x$  by auto
  hence  $y \in \mathbb{Z}_v \cap Bounded\text{-}vec$  ( $?oi$   $?Bnd$ ) using  $xI$   $xB$  by auto
} note  $y0 = this$ 
{
  assume  $y \$ n \neq 0$ 
  hence  $x0: x \$ n \neq 0$  using  $x$  unfolding  $y$  by auto
  from  $x$   $xI$  have  $x \$ n \in \mathbb{Z}$  unfolding Ints-vec-def by auto
  with  $x0$  have  $abs(x \$ n) \geq 1$  by (meson Ints-nonzero-abs-ge1)
  hence  $abs: abs(1 / (x \$ n)) \leq 1$  by simp
  {
    fix  $a$ 
    have  $abs((1 / (x \$ n)) * a) = abs(1 / (x \$ n)) * abs a$ 
      by simp
    also have  $\dots \leq 1 * abs a$ 
      by (rule mult-right-mono[OF abs], auto)
    finally have  $abs((1 / (x \$ n)) * a) \leq abs a$  by auto
  } note  $abs = this$ 
  from  $x0$  have  $y: y = (1 / (x \$ n)) \cdot_v x$  unfolding  $y$  by auto
  have  $vy: vec\text{-}set y = (\lambda a. (1 / (x \$ n)) * a) \text{ ` } vec\text{-}set x$ 
    unfolding  $y$  by (auto simp: vec-set-def)
  have  $y \in Bounded\text{-}vec$  ( $?oi$   $?Bnd$ ) using  $xB$   $abs$ 
    unfolding Bounded-vec-vec-set  $vy$ 
    by (smt imageE max.absorb2 max.bounded-iff)
  } note  $yn0 = this$ 
note  $y0 yn0$ 
} note  $BndY = this$ 
from  $\langle Y \subseteq carrier\text{-}vec (n+1) \rangle$ 
have  $setvY: y \in Y \implies set_v (vec\text{-}first y n) \subseteq set_v y$  for  $y$ 
  unfolding vec-first-def vec-set-def by auto
from  $BndY(1)$   $setvY$ 
show  $?Z0 \subseteq \mathbb{Z}_v \cap Bounded\text{-}vec$  ( $?oi$  ( $db\ n$  ( $max\ 1\ Bnd$ )))
  by (force simp: Bounded-vec-vec-set Ints-vec-vec-set Y0-def)
from  $BndY(2)$   $setvY$ 
show  $?Z1 \subseteq Bounded\text{-}vec$  ( $?oi$  ( $db\ n$  ( $max\ 1\ Bnd$ )))
  by (force simp: Bounded-vec-vec-set Ints-vec-vec-set Y0-def Y1-def)
qed
qed

```

lemma *decomposition-theorem-polyhedra-2:*

assumes $Q: Q \subseteq carrier\text{-}vec\ n$ **and** $fin\text{-}Q: finite\ Q$

and $X: X \subseteq carrier\text{-}vec\ n$ **and** $fin\text{-}X: finite\ X$

and $P: P = convex\text{-}hull\ Q + cone\ X$

shows $\exists A\ b\ nr. A \in carrier\text{-}mat\ nr\ n \wedge b \in carrier\text{-}vec\ nr \wedge P = polyhedron\ A\ b$

proof –

interpret *next-dim*: *gram-schmidt* $n + 1$ *TYPE* ('a).
interpret *gram-schmidt-m* $n + 1$ n *TYPE*('a).

from *fin-Q* **obtain** *Qs* **where** *Qs*: $Q = \text{set } Qs$ **using** *finite-list* **by** *auto*
from *fin-X* **obtain** *Xs* **where** *Xs*: $X = \text{set } Xs$ **using** *finite-list* **by** *auto*
define *Y* **where** $Y = \{x @_v \text{vec-of-scal } 1 \mid x. x \in Q\}$
define *Z* **where** $Z = \{x @_v \text{vec-of-scal } 0 \mid x. x \in X\}$
have *fin-Y*: *finite* *Y* **unfolding** *Y-def* **using** *fin-Q* **by** *simp*
have *fin-Z*: *finite* *Z* **unfolding** *Z-def* **using** *fin-X* **by** *simp*
have *Y-dim*: $Y \subseteq \text{carrier-vec } (n + 1)$
 unfolding *Y-def* **using** *Q* *append-carrier-vec*[*OF - vec-of-scal-dim*(2)] [*of 1*]
 by *blast*
have *Z-dim*: $Z \subseteq \text{carrier-vec } (n + 1)$
 unfolding *Z-def* **using** *X* *append-carrier-vec*[*OF - vec-of-scal-dim*(2)] [*of 0*]
 by *blast*
have *Y-car*: $Q = \{\text{vec-first } x \ n \mid x. x \in Y\}$
proof (*intro equalityI subsetI*)
 fix *x* **assume** $x: x \in Q$
 hence $x @_v \text{vec-of-scal } 1 \in Y$ **unfolding** *Y-def* **by** *blast*
 thus $x \in \{\text{vec-first } x \ n \mid x. x \in Y\}$
 using *Q* *vec-first-append*[*of x n vec-of-scal 1*] *x* **by** *force*
next
 fix *x* **assume** $x \in \{\text{vec-first } x \ n \mid x. x \in Y\}$
 then obtain *y* **where** $y \in Q$ **and** $x = \text{vec-first } (y @_v \text{vec-of-scal } 1) \ n$
 unfolding *Y-def* **by** *blast*
 thus $x \in Q$ **using** *Q* *vec-first-append*[*of y*] **by** *auto*
qed
have *Z-car*: $X = \{\text{vec-first } x \ n \mid x. x \in Z\}$
proof (*intro equalityI subsetI*)
 fix *x* **assume** $x: x \in X$
 hence $x @_v \text{vec-of-scal } 0 \in Z$ **unfolding** *Z-def* **by** *blast*
 thus $x \in \{\text{vec-first } x \ n \mid x. x \in Z\}$
 using *X* *vec-first-append*[*of x n vec-of-scal 0*] *x* **by** *force*
next
 fix *x* **assume** $x \in \{\text{vec-first } x \ n \mid x. x \in Z\}$
 then obtain *y* **where** $y \in X$ **and** $x = \text{vec-first } (y @_v \text{vec-of-scal } 0) \ n$
 unfolding *Z-def* **by** *blast*
 thus $x \in X$ **using** *X* *vec-first-append*[*of y*] **by** *auto*
qed
have *Y-last*: $\forall x \in Y. x \$ n = 1$ **unfolding** *Y-def* **using** *Q* **by** *auto*
have *Z-last*: $\forall x \in Z. x \$ n = 0$ **unfolding** *Z-def* **using** *X* **by** *auto*

have *finite* ($Y \cup Z$) **using** *fin-Y* *fin-Z* **by** *blast*
moreover have $Y \cup Z \subseteq \text{carrier-vec } (n + 1)$ **using** *Y-dim* *Z-dim* **by** *blast*
ultimately obtain *B nr*
 where *B*: *next-dim.cone* ($Y \cup Z$) = *next-dim.polyhedral-cone* *B*
 and *B-carrier*: $B \in \text{carrier-mat } nr \ (n + 1)$
 using *next-dim.farkas-minkowsky-weyl-theorem*[*of next-dim.cone* ($Y \cup Z$)]
 by *blast*

```

define A where  $A = \text{mat-col-first } B \ n$ 
define b where  $b = \text{col } B \ n$ 
have B-blocks:  $B = A \ @_c \ \text{mat-of-col } b$ 
  unfolding A-def b-def
  using mat-col-first-last-append[of  $B \ n \ 1$ ] B-carrier
    mat-of-col-dim-col-1[of mat-col-last  $B \ 1$ ] by auto
have A-carrier:  $A \in \text{carrier-mat } nr \ n$  unfolding A-def using B-carrier by force
have b-carrier:  $b \in \text{carrier-vec } nr$  unfolding b-def using B-carrier by force

{
  fix x assume  $x \in P$ 
  then obtain y z where  $x: x = y + z$  and  $y: y \in \text{convex-hull } Q$  and  $z: z \in$ 
cone X
  using P by (auto elim: set-plus-elim)

  have yn:  $y \in \text{carrier-vec } n$  using y convex-hull-carrier[OF Q] by blast
  moreover have zn:  $z \in \text{carrier-vec } n$  using z cone-carrier[OF X] by blast
  ultimately have xn:  $x \in \text{carrier-vec } n$  using x by blast

  have yn1:  $y \ @_v \ \text{vec-of-scal } 1 \in \text{carrier-vec } (n + 1)$ 
    using append-carrier-vec[OF yn] vec-of-scal-dim by fast
  have y-last:  $(y \ @_v \ \text{vec-of-scal } 1) \ \$ \ n = 1$  using yn by force
  have vec-first  $(y \ @_v \ \text{vec-of-scal } 1) \ n = y$ 
    using vec-first-append[OF yn] by simp
  hence  $y \ @_v \ \text{vec-of-scal } 1 \in \text{next-dim.cone } Y$ 
    using convex-hull-next-dim[OF - Y-dim fin-Y Y-last yn1 y-last] Y-car y by
argo
  hence y-cone:  $y \ @_v \ \text{vec-of-scal } 1 \in \text{next-dim.cone } (Y \cup Z)$ 
    using next-dim.cone-mono[of Y Y \cup Z] by blast

  have zn1:  $z \ @_v \ \text{vec-of-scal } 0 \in \text{carrier-vec } (n + 1)$ 
    using append-carrier-vec[OF zn] vec-of-scal-dim by fast
  have z-last:  $(z \ @_v \ \text{vec-of-scal } 0) \ \$ \ n = 0$  using zn by force
  have vec-first  $(z \ @_v \ \text{vec-of-scal } 0) \ n = z$ 
    using vec-first-append[OF zn] by simp
  hence  $z \ @_v \ \text{vec-of-scal } 0 \in \text{next-dim.cone } Z$ 
    using cone-next-dim[OF - Z-dim fin-Z Z-last zn1 z-last] Z-car z by argo
  hence z-cone:  $z \ @_v \ \text{vec-of-scal } 0 \in \text{next-dim.cone } (Y \cup Z)$ 
    using next-dim.cone-mono[of Z Y \cup Z] by blast

  from  $\langle x = y + z \rangle$ 
  have  $x \ @_v \ \text{vec-of-scal } 1 = (y \ @_v \ \text{vec-of-scal } 1) + (z \ @_v \ \text{vec-of-scal } 0)$ 
    using append-vec-add[OF yn zn] vec-of-scal-dim-1
    unfolding vec-of-scal-def by auto
  hence  $x \ @_v \ \text{vec-of-scal } 1 \in \text{next-dim.cone } (Y \cup Z) \wedge x \in \text{carrier-vec } n$ 
    using next-dim.cone-elem-sum[OF - y-cone z-cone] Y-dim Z-dim xn by auto
} moreover {
  fix x assume  $x \ @_v \ \text{vec-of-scal } 1 \in \text{next-dim.cone } (Y \cup Z)$ 
  then obtain c where  $x: \text{next-dim.lincomb } c \ (Y \cup Z) = x \ @_v \ \text{vec-of-scal } 1$ 

```

```

and c: c ' (Y ∪ Z) ⊆ {t. t ≥ 0}
using next-dim.cone-iff-finite-cone Y-dim Z-dim fin-Y fin-Z
unfolding next-dim.finite-cone-def next-dim.nonneg-lincomb-def by auto

let ?y = next-dim.lincomb c Y
let ?z = next-dim.lincomb c Z
have xyz: x @v vec-of-scal 1 = ?y + ?z
  using x next-dim.lincomb-union[OF Y-dim Z-dim - fin-Y fin-Z] Y-last Z-last
  by fastforce

have y-dim: ?y ∈ carrier-vec (n + 1) using next-dim.lincomb-closed[OF Y-dim]
  by blast
have z-dim: ?z ∈ carrier-vec (n + 1) using next-dim.lincomb-closed[OF Z-dim]
  by blast
have x @v vec-of-scal 1 ∈ carrier-vec (n + 1)
  using xyz add-carrier-vec[OF y-dim z-dim] by argo
hence x-dim: x ∈ carrier-vec n
  using carrier-dim-vec[of x n] carrier-dim-vec[of - n + 1]
  by force

have z-last: ?z $ n = 0 using Z-last next-dim.lincomb-index[OF - Z-dim, of n]
  by force
have ?y $ n + ?z $ n = (x @v vec-of-scal 1) $ n
  using xyz index-add-vec(1) z-dim by simp
also have ... = 1 using x-dim by auto
finally have y-last: ?y $ n = 1 using z-last by algebra

have ?y ∈ next-dim.cone Y
  using next-dim.cone-iff-finite-cone[OF Y-dim] fin-Y c
  unfolding next-dim.finite-cone-def next-dim.nonneg-lincomb-def by auto
hence y-cone: vec-first ?y n ∈ convex-hull Q
  using convex-hull-next-dim[OF - Y-dim fin-Y Y-last y-dim y-last] Y-car
  by blast

have ?z ∈ next-dim.cone Z
  using next-dim.cone-iff-finite-cone[OF Z-dim] fin-Z c
  unfolding next-dim.finite-cone-def next-dim.nonneg-lincomb-def by auto
hence z-cone: vec-first ?z n ∈ cone X
  using cone-next-dim[OF - Z-dim fin-Z Z-last z-dim z-last] Z-car
  by blast

have x = vec-first (x @v vec-of-scal 1) n using vec-first-append[OF x-dim] by
simp
also have ... = vec-first ?y n + vec-first ?z n
  using xyz vec-first-add[of n ?y ?z] y-dim z-dim carrier-dim-vec by auto
finally have x ∈ P
  using y-cone z-cone P by blast
} moreover {
  fix x :: 'a vec

```

assume xn : $x \in \text{carrier-vec } n$
hence $(x @_v \text{vec-of-scal } 1 \in \text{next-dim.polyhedral-cone } B) =$
 $(B *_v (x @_v \text{vec-of-scal } 1) \leq 0_v \text{ nr})$
unfolding $\text{next-dim.polyhedral-cone-def}$ **using** $B\text{-carrier}$
using $\text{append-carrier-vec}[OF - \text{vec-of-scal-dim}(2)[\text{of } 1]]$ **by** auto
also have $\dots = ((A @_c \text{mat-of-col } b) *_v (x @_v \text{vec-of-scal } 1) \leq 0_v \text{ nr})$
using $B\text{-blocks}$ **by** blast
also have $(A @_c \text{mat-of-col } b) *_v (x @_v \text{vec-of-scal } 1) =$
 $A *_v x + \text{mat-of-col } b *_v \text{vec-of-scal } 1$
by $(\text{rule mat-mult-append-cols, insert } A\text{-carrier } b\text{-carrier } xn, \text{ auto simp del: One-nat-def})$
also have $\text{mat-of-col } b *_v \text{vec-of-scal } 1 = b$
using $\text{mult-mat-of-row-vec-of-scal}[\text{of } b \ 1]$ **by** simp
also have $A *_v x + b = A *_v x - -b$ **by** auto
finally have $(x @_v \text{vec-of-scal } 1 \in \text{next-dim.polyhedral-cone } B) = (A *_v x \leq -b)$
using $\text{vec-le-iff-diff-le-0}[\text{of } A *_v x - b]$ $A\text{-carrier}$ **by** simp
}
ultimately have $P = \text{polyhedron } A (-b)$
unfolding polyhedron-def **using** B **by** blast
moreover have $-b \in \text{carrier-vec } nr$ **using** $b\text{-carrier}$ **by** simp
ultimately show $?thesis$ **using** $A\text{-carrier}$ **by** blast
qed

lemma $\text{decomposition-theorem-polyhedra}$:

$(\exists A \ b \ nr. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b)$
 \longleftrightarrow

$(\exists Q \ X. Q \cup X \subseteq \text{carrier-vec } n \wedge \text{finite } (Q \cup X) \wedge P = \text{convex-hull } Q + \text{cone } X)$ **(is ?l = ?r)**

proof

assume $?l$

then obtain $A \ b \ nr$ **where** $A: A \in \text{carrier-mat } nr \ n$

and $b: b \in \text{carrier-vec } nr$ **and** $P: P = \text{polyhedron } A \ b$ **by** auto

from $\text{decomposition-theorem-polyhedra-1}[OF \ \text{this}]$ **obtain** $Q \ X$

where $*$: $X \subseteq \text{carrier-vec } n$ $\text{finite } X$ $Q \subseteq \text{carrier-vec } n$ $\text{finite } Q$ $P = \text{convex-hull } Q + \text{cone } X$

by meson

show $?r$

by $(\text{rule } exI[\text{of } - \ Q], \text{ rule } exI[\text{of } - \ X], \text{ insert } *, \text{ auto simp: polytope-def})$

next

assume $?r$

then obtain $Q \ X$ **where** $QX\text{-carrier}: Q \cup X \subseteq \text{carrier-vec } n$

and $QX\text{-fin}: \text{finite } (Q \cup X)$

and $P: P = \text{convex-hull } Q + \text{cone } X$ **by** blast

from $QX\text{-carrier}$ **have** $Q: Q \subseteq \text{carrier-vec } n$ **and** $X: X \subseteq \text{carrier-vec } n$ **by** simp-all

from $QX\text{-fin}$ **have** $\text{fin-}Q: \text{finite } Q$ **and** $\text{fin-}X: \text{finite } X$ **by** simp-all

show $?l$ **using** $\text{decomposition-theorem-polyhedra-2}[OF \ Q \ \text{fin-}Q \ X \ \text{fin-}X \ P]$ **by** blast

qed

lemma *polytope-equiv-bounded-polyhedron*:

polytope $P \longleftrightarrow$

$(\exists A \ b \ nr \ bnd. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b$
 $\wedge P \subseteq \text{Bounded-vec } bnd)$

proof

assume *polyP*: *polytope* P

from this obtain Q **where** $Qcarr$: $Q \subseteq \text{carrier-vec } n$ **and** $finQ$: *finite* Q

and $PconvhQ$: $P = \text{convex-hull } Q$ **unfolding** *polytope-def* **by** *auto*

let $?X = \{\}$

have $\text{convex-hull } Q + \{0_v \ n\} = \text{convex-hull } Q$ **using** $Qcarr$ *add-0-right-vecset* [of *convex-hull* Q]

by (*simp add: convex-hull-carrier*)

hence $P = \text{convex-hull } Q + \text{cone } ?X$ **using** $PconvhQ$ **by** *simp*

hence $Q \cup ?X \subseteq \text{carrier-vec } n \wedge \text{finite } (Q \cup ?X) \wedge P = \text{convex-hull } Q + \text{cone } ?X$

using $Qcarr$ $finQ$ $PconvhQ$ **by** *simp*

hence $\exists A \ b \ nr. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b$

using *decomposition-theorem-polyhedra* **by** *blast*

hence $Ppolyh$: $\exists A \ b \ nr. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b$ **by** *blast*

from *finite-Bounded-vec-Max* [*OF* $Qcarr$ $finQ$] **obtain** bnd **where** $Q \subseteq \text{Bounded-vec } bnd$ **by** *auto*

hence $Pbnd$: $P \subseteq \text{Bounded-vec } bnd$ **using** *convex-hull-bound* $PconvhQ$ $Qcarr$ **by** *auto*

from $Ppolyh$ $Pbnd$ **show** $\exists A \ b \ nr \ bnd. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr$

$\wedge P = \text{polyhedron } A \ b \wedge P \subseteq \text{Bounded-vec } bnd$ **by** *auto*

next

assume $\exists A \ b \ nr \ bnd. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b$

$\wedge P \subseteq \text{Bounded-vec } bnd$

from this obtain $A \ b \ nr \ bnd$ **where** $Adim$: $A \in \text{carrier-mat } nr \ n$ **and** $bdim$: $b \in \text{carrier-vec } nr$

and $Ppolyh$: $P = \text{polyhedron } A \ b$ **and** $Pbnd$: $P \subseteq \text{Bounded-vec } bnd$ **by** *auto*

have $\exists A \ b \ nr. A \in \text{carrier-mat } nr \ n \wedge b \in \text{carrier-vec } nr \wedge P = \text{polyhedron } A \ b$

using $Adim$ $bdim$ $Ppolyh$ **by** *blast*

hence $\exists Q \ X. Q \cup X \subseteq \text{carrier-vec } n \wedge \text{finite } (Q \cup X) \wedge P = \text{convex-hull } Q + \text{cone } X$

using *decomposition-theorem-polyhedra* **by** *simp*

from this obtain $Q \ X$ **where** $QXcarr$: $Q \cup X \subseteq \text{carrier-vec } n$

and $finQX$: *finite* $(Q \cup X)$ **and** $Psum$: $P = \text{convex-hull } Q + \text{cone } X$ **by** *auto*

from $QXcarr$ **have** $Qcarr$: $\text{convex-hull } Q \subseteq \text{carrier-vec } n$ **by** (*simp add: convex-hull-carrier*)

from $QXcarr$ **have** $Xcarr$: $\text{cone } X \subseteq \text{carrier-vec } n$ **by** (*simp add: gram-schmidt.cone-carrier*)

from $Pbnd$ **have** $Pcarr$: $P \subseteq \text{carrier-vec } n$ **using** $Ppolyh$ **unfolding** *polyhe-*

```

dron-def by simp
have P = convex-hull Q
proof(cases Q = {})
  case True
  then show P = convex-hull Q unfolding Psum by (auto simp: set-plus-def)
next
  case False
  hence convnotempty: convex-hull Q ≠ {} using QXcarr by simp
  have Pbound: ∃ bnd. P ⊆ Bounded-vec bnd using Pbound
  using QXcarr by auto
  from False have (∃ bndc. cone X ⊆ Bounded-vec bndc)
  using bounded-vecset-sum[OF Qcarr Xcarr Psum Pbound] False convnotempty
by blast
  hence cone X = {0_v n} using bounded-cone-is-zero QXcarr by auto
  thus ?thesis unfolding Psum using Qcarr by (auto simp: add-0-right-vecset)
qed
thus polytope P using finQX QXcarr unfolding polytope-def by auto
qed
end

end

```

15 Mixed Integer Solutions

We prove that if an integral system of linear inequalities $Ax \leq b \wedge A'x < b'$ has a (mixed)integer solution, then there is also a small (mixed)integer solution, where the numbers are bounded by $(n + 1) * dbm n$ where n is the number of variables, m is a bound on the absolute values of numbers occurring in A, A', b, b' , and $dbm n$ is a bound on determinants for matrices of size n with values of at most m .

```

theory Mixed-Integer-Solutions
  imports Decomposition-Theorem
begin

```

```

definition less-vec :: 'a vec ⇒ ('a :: ord) vec ⇒ bool (infix '<_v' 50) where
  v <_v w = (dim-vec v = dim-vec w ∧ (∀ i < dim-vec w. v $ i < w $ i))

```

```

lemma less-vecD: assumes v <_v w and w ∈ carrier-vec n
  shows i < n ⇒ v $ i < w $ i
  using assms unfolding less-vec-def by auto

```

```

lemma less-vecI: assumes v ∈ carrier-vec n w ∈ carrier-vec n
  ∧ i. i < n ⇒ v $ i < w $ i
shows v <_v w
  using assms unfolding less-vec-def by auto

```

```

lemma less-vec-lesseq-vec: v <_v (w :: 'a :: preorder vec) ⇒ v ≤ w

```

```

unfolding less-vec-def less-eq-vec-def
by (auto simp: less-le-not-le)

lemma floor-less:  $x \notin \mathbf{Z} \implies \text{of-int } \lfloor x \rfloor < x$ 
using le-less by fastforce

lemma floor-of-int-eq[simp]:  $x \in \mathbf{Z} \implies \text{of-int } \lfloor x \rfloor = x$ 
by (metis Ints-cases of-int-floor-cancel)

locale gram-schmidt-floor = gram-schmidt n f-ty
  for n :: nat and f-ty :: 'a :: {floor-ceiling,
    trivial-conjugatable-linordered-field} itself
begin

lemma small-mixed-integer-solution-main: fixes A1 :: 'a mat
assumes db: is-det-bound db
  and A1: A1 ∈ carrier-mat nr1 n
  and A2: A2 ∈ carrier-mat nr2 n
  and b1: b1 ∈ carrier-vec nr1
  and b2: b2 ∈ carrier-vec nr2
  and A1Bnd: A1 ∈  $\mathbf{Z}_m \cap \text{Bounded-mat (of-int Bnd)}$ 
  and b1Bnd: b1 ∈  $\mathbf{Z}_v \cap \text{Bounded-vec (of-int Bnd)}$ 
  and A2Bnd: A2 ∈  $\mathbf{Z}_m \cap \text{Bounded-mat (of-int Bnd)}$ 
  and b2Bnd: b2 ∈  $\mathbf{Z}_v \cap \text{Bounded-vec (of-int Bnd)}$ 
  and x: x ∈ carrier-vec n
  and xI: x ∈ indexed-Ints-vec I
  and sol-nonstrict: A1 *v x ≤ b1
  and sol-strict: A2 *v x <v b2
shows ∃ x.
  x ∈ carrier-vec n ∧
  x ∈ indexed-Ints-vec I ∧
  A1 *v x ≤ b1 ∧
  A2 *v x <v b2 ∧
  x ∈ Bounded-vec (of-int (of-nat (n + 1) * db n (max 1 Bnd)))
proof –
  let ?oi = of-int :: int ⇒ 'a
  let ?Bnd = ?oi Bnd
  define B where B = ?oi (db n (max 1 Bnd))
  define A where A = A1 @r A2
  define b where b = b1 @v b2
  define nr where nr = nr1 + nr2
  have B0: B ≥ 0 unfolding B-def of-int-0-le-iff
  by (rule is-det-bound-ge-zero[OF db], auto)
  note defs = A-def b-def nr-def
  from A1 A2 have A: A ∈ carrier-mat nr n unfolding defs by auto
  from b1 b2 have b: b ∈ carrier-vec nr unfolding defs by auto
  from A1Bnd A2Bnd A1 A2 have ABnd: A ∈  $\mathbf{Z}_m \cap \text{Bounded-mat ?Bnd}$  un-
folding defs

```

by (*auto simp: Ints-mat-elements-mat Bounded-mat-elements-mat elements-mat-append-rows*)
from $b1Bnd\ b2Bnd\ b1\ b2$ **have** $bBnd: b \in \mathbb{Z}_v \cap \text{Bounded-vec } ?Bnd$ **unfolding**
defs
by (*auto simp: Ints-vec-vec-set Bounded-vec-vec-set*)
from *decomposition-theorem-polyhedra-1*[*OF A b refl, of db Bnd*] $ABnd\ bBnd\ db$
obtain $Y\ Z$ **where** $Z: Z \subseteq \text{carrier-vec } n$
and $finX: \text{finite } Z$
and $Y: Y \subseteq \text{carrier-vec } n$
and $finY: \text{finite } Y$
and $poly: \text{polyhedron } A\ b = \text{convex-hull } Y + \text{cone } Z$
and $ZBnd: Z \subseteq \mathbb{Z}_v \cap \text{Bounded-vec } B$
and $YBnd: Y \subseteq \text{Bounded-vec } B$ **unfolding** $B\text{-def}$ **by** *blast*
let $?P = \{x \in \text{carrier-vec } n. A_1 *_{\nu} x \leq b_1 \wedge A_2 *_{\nu} x \leq b_2\}$
let $?L = ?P \cap \{x. A_2 *_{\nu} x <_{\nu} b_2\} \cap \text{indexed-Ints-vec } I$
have $\text{polyhedron } A\ b = \{x \in \text{carrier-vec } n. A *_{\nu} x \leq b\}$ **unfolding** *polyhedron-def*
by *auto*
also **have** $\dots = ?P$ **unfolding** *defs*
by (*intro Collect-cong conj-cong refl append-rows-le*[*OF A1 A2 b1*])
finally **have** $poly: ?P = \text{convex-hull } Y + \text{cone } Z$ **unfolding** *poly ..*
have $x \in ?P$ **using** x *sol-nonstrict less-vec-lesseq-vec*[*OF sol-strict*] **by** *blast*
note $sol = \text{this}$ [*unfolded poly*]
from *set-plus-elim*[*OF sol*] **obtain** $y\ z$ **where** $xyz: x = y + z$ **and**
 $yY: y \in \text{convex-hull } Y$ **and** $zZ: z \in \text{cone } Z$ **by** *auto*
from *convex-hull-carrier*[*OF Y*] yY **have** $y: y \in \text{carrier-vec } n$ **by** *auto*
from *Caratheodory-theorem*[*OF Z*] zZ
obtain C **where** $zC: z \in \text{finite-cone } C$ **and** $CZ: C \subseteq Z$ **and** $lin: \text{lin-indpt } C$
by *auto*
from *subset-trans*[*OF CZ Z*] lin **have** $card: \text{card } C \leq n$
using *dim-is-n li-le-dim*(2) **by** *auto*
from *finite-subset*[*OF CZ finX*] **have** $finC: \text{finite } C$.
from zC [*unfolded finite-cone-def nonneg-lincomb-def*] $finC$ **obtain** a
where $za: z = \text{lincomb } a\ C$ **and** $nonneg: \bigwedge u. u \in C \implies a\ u \geq 0$ **by** *auto*
from $CZ\ Z$ **have** $C: C \subseteq \text{carrier-vec } n$ **by** *auto*
have $z: z \in \text{carrier-vec } n$ **using** C **unfolding** za **by** *auto*
have $yB: y \in \text{Bounded-vec } B$ **using** yY *convex-hull-bound*[*OF YBnd Y*] **by** *auto*
{
fix D
assume $DC: D \subseteq C$
from *finite-subset*[*OF this finC*] **have** $\text{finite } D$.
hence $\exists a. y + \text{lincomb } a\ C \in ?L \wedge (\forall c \in C. a\ c \geq 0) \wedge (\forall c \in D. a\ c \leq 1)$
using DC
proof (*induct D*)
case *empty*
show *?case* **by** (*intro exI*[*of - a*], *fold za xyz, insert sol-strict x xI nonneg* $\langle x \in ?P \rangle$, *auto*)
next
case (*insert c D*)
then **obtain** a **where** $sol: y + \text{lincomb } a\ C \in ?L$
and $a: (\forall c \in C. a\ c \geq 0)$ **and** $D: (\forall c \in D. a\ c \leq 1)$ **by** *auto*
}

```

from insert(4) C have c: c ∈ carrier-vec n and cC: c ∈ C by auto
show ?case
proof (cases a c > 1)
  case False
  thus ?thesis by (intro exI[of - a], insert sol a D, auto)
next
case True
let ?z = λ d. lincomb a C - d ·v c
let ?x = λ d. y + ?z d
{
  fix d
  have lin: lincomb a (C - {c}) ∈ carrier-vec n using C by auto
  have id: ?z d = lincomb (λ e. if e = c then (a c - d) else a e) C
  unfolding lincomb-del2[OF finC C TrueI cC]
  by (subst (2) lincomb-cong[OF refl, of - - a], insert C c lin, auto simp:
diff-smult-distrib-vec)
  {
    assume le: d ≤ a c
    have ?z d ∈ finite-cone C
    proof -
      have ∀f∈C. 0 ≤ (λe. if e = c then a c - d else a e) f using le a finC
    by simp
    then show ?thesis unfolding id using le a finC
    by (simp add: C lincomb-in-finite-cone)
  qed
  hence ?z d ∈ cone Z using CZ
  using finC local.cone-def by blast
  hence ?x d ∈ ?P unfolding poly
  by (intro set-plus-intro[OF yY], auto)
} note sol = this
{
  fix w :: 'a vec
  assume w: w ∈ carrier-vec n
  have w · (?x d) = w · y + w · lincomb a C - d * (w · c)
  by (subst scalar-prod-add-distrib[OF w y], (insert C c, force),
subst scalar-prod-minus-distrib[OF w], insert w c C, auto)
} note scalar = this
note id sol scalar
} note generic = this
let ?fl = (of-int (floor (a c)) :: 'a)
define p where p = (if ?fl = a c then a c - 1 else ?fl)
have p-lt-ac: p < a c unfolding p-def
using floor-less floor-of-int-eq by auto
have p1-ge-ac: p + 1 ≥ a c unfolding p-def
using floor-correct le-less by auto
have p1: p ≥ 1 using True unfolding p-def by auto
define a' where a' = (λe. if e = c then a c - p else a e)
have lin-id: lincomb a' C = lincomb a C - p ·v c unfolding a'-def using
id

```

```

    by (simp add: generic(1))
  hence 1:  $y + \text{lincomb } a' C \in \{x \in \text{carrier-vec } n. A_1 *_v x \leq b_1 \wedge A_2 *_v x \leq b_2\}$ 
    using p-lt-ac generic(2)[of p] by auto
  have pInt:  $p \in \mathbb{Z}$  unfolding p-def using sol by auto
  have C  $\subseteq$  indexed-Ints-vec I using CZ ZBnd
    using indexed-Ints-vec-subset by force
  hence c  $\in$  indexed-Ints-vec I using cC by auto
  hence pvindInts:  $p \cdot_v c \in \text{indexed-Ints-vec } I$  unfolding indexed-Ints-vec-def
using pInt by simp
  have prod:  $A_2 *_v (?x b) \in \text{carrier-vec } nr_2$  for b using A2 C c y by auto
  have 2:  $y + \text{lincomb } a' C \in \{x. A_2 *_v x <_v b_2\}$  unfolding lin-id
  proof (intro less-vecI[OF prod b2] CollectI)
    fix i
    assume i:  $i < nr_2$ 
    from sol have  $A_2 *_v (?x 0) <_v b_2$  using y C c by auto
    from less-vecD[OF this b2 i]
    have lt:  $\text{row } A_2 i \cdot ?x 0 < b_2$   $\$ i$  using A2 i by auto
    from generic(2)[of a c] i A2
    have le:  $\text{row } A_2 i \cdot ?x (a c) \leq b_2$   $\$ i$ 
      unfolding less-eq-vec-def by auto
    from A2 i have A2icarr:  $\text{row } A_2 i \in \text{carrier-vec } n$  by auto
    have row  $A_2 i \cdot ?x p < b_2$   $\$ i$ 
    proof -
      define lhs where  $\text{lhs} = \text{row } A_2 i \cdot y + \text{row } A_2 i \cdot \text{lincomb } a C - b_2$   $\$ i$ 
      define mult where  $\text{mult} = \text{row } A_2 i \cdot c$ 
      have le2:  $\text{lhs} \leq a c * \text{mult}$  using le unfolding generic(3)[OF A2icarr]
lhs-def mult-def by auto
      have lt2:  $\text{lhs} < 0 * \text{mult}$  using lt unfolding generic(3)[OF A2icarr]
lhs-def by auto
      from le2 lt2 have  $\text{lhs} < p * \text{mult}$  using p-lt-ac p1 True
      by (smt dual-order.strict-trans linorder-neqE-linordered-idom
        mult-less-cancel-right not-less zero-less-one-class.zero-less-one)
      then show ?thesis unfolding generic(3)[OF A2icarr] lhs-def mult-def
    by auto
  qed
  thus  $(A_2 *_v ?x p) \text{ } \$ i < b_2$   $\$ i$  using i A2 by auto
  qed
  have  $y + \text{lincomb } a' C = y + \text{lincomb } a C - p \cdot_v c$ 
    by (subst lin-id, insert y C c, auto simp: add-diff-eq-vec)
  also have  $\dots \in \text{indexed-Ints-vec } I$  using sol
    by (intro diff-indexed-Ints-vec[OF - - - pvindInts, of - n ], insert c C, auto)
  finally have 3:  $y + \text{lincomb } a' C \in \text{indexed-Ints-vec } I$  by auto
  have 4:  $\forall c \in C. 0 \leq a' c$  unfolding a'-def p-def using p-lt-ac a by auto
  have 5:  $\forall c \in \text{insert } c D. a' c \leq 1$  unfolding a'-def using p1-ge-ac D p-def
by auto
  show ?thesis
    by (intro exI[of - a'], intro conjI IntI 1 2 3 4 5)
  qed

```

```

    qed
  }
  from this[of C] obtain a where
    sol:  $y + \text{lincomb } a \ C \in ?L$  and bnds:  $(\forall c \in C. a \ c \geq 0) (\forall c \in C. a \ c \leq 1)$ 
  by auto
  show ?thesis
  proof (intro exI[of -  $y + \text{lincomb } a \ C$ ] conjI)
    from ZBnd CZ have BndC:  $C \subseteq \text{Bounded-vec } B$  and IntC:  $C \subseteq \mathbb{Z}_v$  by auto
    have  $\text{lincomb } a \ C \in \text{Bounded-vec } (\text{of-nat } n * B)$ 
      using  $\text{lincomb-card-bound}[OF \ BndC \ C \ B0 - \text{card}]$  bnds by auto
    from  $\text{sum-in-Bounded-vecI}[OF \ yB \ \text{this } y] \ C$ 
    have  $y + \text{lincomb } a \ C \in \text{Bounded-vec } (B + \text{of-nat } n * B)$  by auto
    also have  $B + \text{of-nat } n * B = \text{of-nat } (n+1) * B$  by (auto simp: field-simps)
    finally show  $y + \text{lincomb } a \ C \in \text{Bounded-vec } (\text{of-int } (\text{of-nat } (n + 1)) * db \ n$ 
      ( $\text{max } 1 \ Bnd$ ))
      unfolding B-def by auto
    qed (insert sol, auto)
  qed

```

We get rid of the max-1 operation, by showing that a smaller value of Bnd can only occur in very special cases where the theorem is trivially satisfied.

```

lemma small-mixed-integer-solution: fixes  $A_1 :: 'a \ \text{mat}$ 
  assumes  $db: \text{is-det-bound } db$ 
  and  $A1: A_1 \in \text{carrier-mat } nr_1 \ n$ 
  and  $A2: A_2 \in \text{carrier-mat } nr_2 \ n$ 
  and  $b1: b_1 \in \text{carrier-vec } nr_1$ 
  and  $b2: b_2 \in \text{carrier-vec } nr_2$ 
  and  $A1Bnd: A_1 \in \mathbb{Z}_m \cap \text{Bounded-mat } (\text{of-int } Bnd)$ 
  and  $b1Bnd: b_1 \in \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } Bnd)$ 
  and  $A2Bnd: A_2 \in \mathbb{Z}_m \cap \text{Bounded-mat } (\text{of-int } Bnd)$ 
  and  $b2Bnd: b_2 \in \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } Bnd)$ 
  and  $x: x \in \text{carrier-vec } n$ 
  and  $xI: x \in \text{indexed-Ints-vec } I$ 
  and  $\text{sol-nonstrict}: A_1 *_{\mathbb{Z}_v} x \leq b_1$ 
  and  $\text{sol-strict}: A_2 *_{\mathbb{Z}_v} x <_{\mathbb{Z}_v} b_2$ 
  and  $\text{non-degenerate}: nr_1 \neq 0 \vee nr_2 \neq 0 \vee Bnd \geq 0$ 
  shows  $\exists x.$ 
   $x \in \text{carrier-vec } n \wedge$ 
   $x \in \text{indexed-Ints-vec } I \wedge$ 
   $A_1 *_{\mathbb{Z}_v} x \leq b_1 \wedge$ 
   $A_2 *_{\mathbb{Z}_v} x <_{\mathbb{Z}_v} b_2 \wedge$ 
   $x \in \text{Bounded-vec } (\text{of-int } (\text{int } (n+1)) * db \ n \ Bnd)$ 
proof (cases  $Bnd \geq 1$ )
  case True
  hence  $\text{max } 1 \ Bnd = Bnd$  by auto
  with  $\text{small-mixed-integer-solution-main}[OF \ \text{assms}(1-13)]$  True show ?thesis by
  auto
next

```

```

case trivial: False
let ?oi = of-int :: int  $\Rightarrow$  'a
show ?thesis
proof (cases n = 0)
  case True
    with x have x  $\in$  Bounded-vec b for b unfolding Bounded-vec-def by auto
    with xI x sol-nonstrict sol-strict show ?thesis by blast
  next
    case n: False
      {
        fix A nr
        assume A: A  $\in$  carrier-mat nr n and Bnd: A  $\in$   $\mathbb{Z}_m \cap$  Bounded-mat (?oi
Bnd)
        {
          fix i j
          assume i < nr j < n
          with Bnd A have *: A  $\$ \$$  (i,j)  $\in$   $\mathbb{Z}$  abs (A  $\$ \$$  (i,j))  $\leq$  ?oi Bnd
            unfolding Bounded-mat-def Ints-mat-def by auto
            from Ints-nonzero-abs-less1 [OF *(1)] *(2) trivial
            have A  $\$ \$$  (i,j) = 0
            by (meson add-le-less-mono int-one-le-iff-zero-less less-add-same-cancel2
of-int-0-less-iff zero-less-abs-iff)
            with *(2) have Bnd  $\geq$  0 A  $\$ \$$  (i,j) = 0 by auto
          } note main = this
          have A0: A = 0m nr n
            by (intro eq-matI, insert main A, auto)
            have nr  $\neq$  0  $\implies$  Bnd  $\geq$  0 using main[of 0 0] n by auto
            note A0 this
          } note main = this
          from main[OF A1 A1Bnd] have A1: A1 = 0m nr1 n and nr1: nr1  $\neq$  0  $\implies$ 
Bnd  $\geq$  0
            by auto
          from main[OF A2 A2Bnd] have A2: A2 = 0m nr2 n and nr2: nr2  $\neq$  0  $\implies$ 
Bnd  $\geq$  0
            by auto
          let ?x = 0v n
          show ?thesis
          proof (intro exI[of - ?x] conjI)
            show A1 *v ?x  $\leq$  b1 using sol-nonstrict x unfolding A1 by auto
            show A2 *v ?x <v b2 using sol-strict x unfolding A2 by auto
            show ?x  $\in$  carrier-vec n by auto
            show ?x  $\in$  indexed-Ints-vec I unfolding indexed-Ints-vec-def by auto
            from nr1 nr2 non-degenerate have Bnd: Bnd  $\geq$  0 by auto
            from is-det-bound-ge-zero [OF db Bnd] have db n Bnd  $\geq$  0 .
            hence ?oi (of-nat (n + 1) * db n Bnd)  $\geq$  0 by simp
            thus ?x  $\in$  Bounded-vec (?oi (of-nat (n + 1) * db n Bnd)) by (auto simp:
Bounded-vec-def)
          } qed
        } qed
      }

```

qed

lemmas *small-mixed-integer-solution-hadamard* =
small-mixed-integer-solution[*OF det-bound-hadamard, unfolded det-bound-hadamard-def*
of-int-mult of-int-of-nat-eq]

lemma *Bounded-vec-of-int*: **assumes** $v \in \text{Bounded-vec } \text{bnd}$
shows $(\text{map-vec of-int } v :: 'a \text{ vec}) \in \mathbb{Z}_v \cap \text{Bounded-vec } (\text{of-int } \text{bnd})$
using *assms*
apply (*simp add: Ints-vec-vec-set Bounded-vec-vec-set Ints-def*)
apply (*intro conjI, force*)
apply (*clarsimp*)
subgoal for x **apply** (*elim ballE[of - - x], auto*)
by (*metis of-int-abs of-int-le-iff*)
done

lemma *Bounded-mat-of-int*: **assumes** $A \in \text{Bounded-mat } \text{bnd}$
shows $(\text{map-mat of-int } A :: 'a \text{ mat}) \in \mathbb{Z}_m \cap \text{Bounded-mat } (\text{of-int } \text{bnd})$
using *assms*
apply (*simp add: Ints-mat-elements-mat Bounded-mat-elements-mat Ints-def*)
apply (*intro conjI, force*)
apply (*clarsimp*)
subgoal for x **apply** (*elim ballE[of - - x], auto*)
by (*metis of-int-abs of-int-le-iff*)
done

lemma *small-mixed-integer-solution-int-mat*: **fixes** $x :: 'a \text{ vec}$
assumes *db: is-det-bound db*
and $A1: A_1 \in \text{carrier-mat } nr_1 \ n$
and $A2: A_2 \in \text{carrier-mat } nr_2 \ n$
and $b1: b_1 \in \text{carrier-vec } nr_1$
and $b2: b_2 \in \text{carrier-vec } nr_2$
and $A1\text{Bnd}: A_1 \in \text{Bounded-mat } \text{Bnd}$
and $b1\text{Bnd}: b_1 \in \text{Bounded-vec } \text{Bnd}$
and $A2\text{Bnd}: A_2 \in \text{Bounded-mat } \text{Bnd}$
and $b2\text{Bnd}: b_2 \in \text{Bounded-vec } \text{Bnd}$
and $x: x \in \text{carrier-vec } n$
and $xI: x \in \text{indexed-Ints-vec } I$
and *sol-nonstrict: map-mat of-int* $A_1 *_{\mathbb{V}} x \leq \text{map-vec of-int } b_1$
and *sol-strict: map-mat of-int* $A_2 *_{\mathbb{V}} x <_{\mathbb{V}} \text{map-vec of-int } b_2$
and *non-degenerate: nr_1* $\neq 0 \vee nr_2 \neq 0 \vee \text{Bnd} \geq 0$
shows $\exists x :: 'a \text{ vec.}$
 $x \in \text{carrier-vec } n \wedge$
 $x \in \text{indexed-Ints-vec } I \wedge$
 $\text{map-mat of-int } A_1 *_{\mathbb{V}} x \leq \text{map-vec of-int } b_1 \wedge$
 $\text{map-mat of-int } A_2 *_{\mathbb{V}} x <_{\mathbb{V}} \text{map-vec of-int } b_2 \wedge$
 $x \in \text{Bounded-vec } (\text{of-int } (\text{of-nat } (n+1)) * \text{db } n \ \text{Bnd}))$
proof –
let $?oi = \text{of-int} :: \text{int} \Rightarrow 'a$

```

let ?A1 = map-mat ?oi A1
let ?A2 = map-mat ?oi A2
let ?b1 = map-vec ?oi b1
let ?b2 = map-vec ?oi b2
let ?Bnd = ?oi Bnd
from A1 have A1': ?A1 ∈ carrier-mat nr1 n by auto
from A2 have A2': ?A2 ∈ carrier-mat nr2 n by auto
from b1 have b1': ?b1 ∈ carrier-vec nr1 by auto
from b2 have b2': ?b2 ∈ carrier-vec nr2 by auto
from small-mixed-integer-solution[OF db A1' A2' b1' b2'
  Bounded-mat-of-int[OF A1Bnd] Bounded-vec-of-int[OF b1Bnd]
  Bounded-mat-of-int[OF A2Bnd] Bounded-vec-of-int[OF b2Bnd]
  x xI sol-nonstrict sol-strict non-degenerate]
show ?thesis .
qed

```

```

lemmas small-mixed-integer-solution-int-mat-hadamard =
  small-mixed-integer-solution-int-mat[OF det-bound-hadamard, unfolded det-bound-hadamard-def
  of-int-mult of-int-of-nat-eq]

```

end

```

lemma of-int-hom-le: (of-int-hom.vec-hom v :: 'a :: linordered-field vec) ≤ of-int-hom.vec-hom
w ↔ v ≤ w
unfolding less-eq-vec-def by auto

```

```

lemma of-int-hom-less: (of-int-hom.vec-hom v :: 'a :: linordered-field vec) <v of-int-hom.vec-hom
w ↔ v <v w
unfolding less-vec-def by auto

```

```

lemma Ints-vec-to-int-vec: assumes v ∈ ℤv
shows ∃ w. v = map-vec of-int w
proof –
have ∀ i. ∃ x. i < dim-vec v → v $ i = of-int x
using assms unfolding Ints-vec-def Ints-def by auto
from choice[OF this] obtain x where ∧ i. i < dim-vec v ⇒ v $ i = of-int (x
i)
by auto
thus ?thesis
by (intro exI[of - vec (dim-vec v) x], auto)
qed

```

```

lemma small-integer-solution: fixes A1 :: int mat
assumes db: is-det-bound db
and A1: A1 ∈ carrier-mat nr1 n
and A2: A2 ∈ carrier-mat nr2 n
and b1: b1 ∈ carrier-vec nr1
and b2: b2 ∈ carrier-vec nr2
and A1Bnd: A1 ∈ Bounded-mat Bnd

```

and $b1Bnd$: $b_1 \in \text{Bounded-vec } Bnd$
and $A2Bnd$: $A_2 \in \text{Bounded-mat } Bnd$
and $b2Bnd$: $b_2 \in \text{Bounded-vec } Bnd$
and x : $x \in \text{carrier-vec } n$
and $sol\text{-nonstrict}$: $A_1 *_v x \leq b_1$
and $sol\text{-strict}$: $A_2 *_v x <_v b_2$
and $non\text{-degenerate}$: $nr_1 \neq 0 \vee nr_2 \neq 0 \vee Bnd \geq 0$
shows $\exists x$.
 $x \in \text{carrier-vec } n \wedge$
 $A_1 *_v x \leq b_1 \wedge$
 $A_2 *_v x <_v b_2 \wedge$
 $x \in \text{Bounded-vec } (of\text{-nat } (n+1) * db \ n \ Bnd)$
proof –
let $?oi = \text{rat-of-int}$
let $?x = \text{map-vec } ?oi \ x$
let $?oiM = \text{map-mat } ?oi$
let $?oiv = \text{map-vec } ?oi$
from x **have** xx : $?x \in \text{carrier-vec } n$ **by** *auto*
have Int : $?x \in \text{indexed-Ints-vec } UNIV$ **unfolding** $\text{indexed-Ints-vec-def } Ints\text{-def}$
by *auto*
interpret $\text{gram-schmidt-floor } n \ TYPE(rat)$.
from
 $\text{small-mixed-integer-solution-int-mat}[OF \ db \ A1 \ A2 \ b1 \ b2 \ A1Bnd \ b1Bnd \ A2Bnd$
 $b2Bnd \ xx \ Int$
– – $non\text{-degenerate}$,
 $\text{folded of-int-hom.mult-mat-vec-hom}[OF \ A1 \ x] \ \text{of-int-hom.mult-mat-vec-hom}[OF$
 $A2 \ x]$,
 $\text{unfolded of-int-hom-less of-int-hom-le, } OF \ sol\text{-nonstrict } sol\text{-strict, folded in-}$
 $\text{dexed-Ints-vec-UNIV}]$
obtain x **where**
 x : $x \in \text{carrier-vec } n$ **and**
 xI : $x \in \mathbb{Z}_v$ **and**
 le : $?oiM \ A_1 *_v x \leq ?oiv \ b_1$ **and**
 $less$: $?oiM \ A_2 *_v x <_v ?oiv \ b_2$ **and**
 Bnd : $x \in \text{Bounded-vec } (?oi \ (int \ (n + 1) * db \ n \ Bnd))$
by *blast*
from $\text{Ints-vec-to-int-vec}[OF \ xI]$ **obtain** xI **where** xI : $x = ?oiv \ xI$ **by** *auto*
from $x[\text{unfolded } xI]$ **have** x : $xI \in \text{carrier-vec } n$ **by** *auto*
from $le[\text{unfolded } xI, \ \text{folded of-int-hom.mult-mat-vec-hom}[OF \ A1 \ x], \ \text{unfolded}$
 $\text{of-int-hom-le}]$
have le : $A_1 *_v xI \leq b_1$.
from $less[\text{unfolded } xI, \ \text{folded of-int-hom.mult-mat-vec-hom}[OF \ A2 \ x], \ \text{unfolded}$
 $\text{of-int-hom-less}]$
have $less$: $A_2 *_v xI <_v b_2$.
show $?thesis$
proof ($\text{intro } exI[\text{of } - \ xI] \ conjI \ x \ le \ less$)
show $xI \in \text{Bounded-vec } (int \ (n + 1) * db \ n \ Bnd)$
unfolding Bounded-vec-def
proof *clarsimp*

```

fix  $i$ 
assume  $i: i < \dim\text{-vec } xI$ 
with  $Bnd$ [unfolded Bounded-vec-def]
have  $|x \$ i| \leq ?oi (int (n + 1) * db n Bnd)$  by (auto simp: xI)
also have  $|x \$ i| = ?oi (|xI \$ i|)$  unfolding  $xI$  using  $i$  by simp
finally show  $|xI \$ i| \leq (1 + int n) * db n Bnd$  unfolding of-int-le-iff by
auto
qed
qed
qed

```

corollary *small-integer-solution-nonstrict*: **fixes** $A :: int\ mat$

```

assumes  $db: is\text{-det-bound } db$ 
and  $A: A \in carrier\text{-mat } nr\ n$ 
and  $b: b \in carrier\text{-vec } nr$ 
and  $ABnd: A \in Bounded\text{-mat } Bnd$ 
and  $bBnd: b \in Bounded\text{-vec } Bnd$ 
and  $x: x \in carrier\text{-vec } n$ 
and  $sol: A *_v x \leq b$ 
and non-degenerate:  $nr \neq 0 \vee Bnd \geq 0$ 
shows  $\exists y.$ 
 $y \in carrier\text{-vec } n \wedge$ 
 $A *_v y \leq b \wedge$ 
 $y \in Bounded\text{-vec } (of\text{-nat } (n+1) * db n Bnd)$ 

```

proof –

```

let  $?A2 = 0_m\ 0\ n :: int\ mat$ 
let  $?b2 = 0_v\ 0 :: int\ vec$ 
from non-degenerate have degen:  $nr \neq 0 \vee (0 :: nat) \neq 0 \vee Bnd \geq 0$  by auto
have  $\exists y. y \in carrier\text{-vec } n \wedge A *_v y \leq b \wedge ?A2 *_v y <_v ?b2$ 
 $\wedge y \in Bounded\text{-vec } (of\text{-nat } (n+1) * db n Bnd)$ 
apply (rule small-integer-solution[OF db A - b - ABnd bBnd - - x sol - degen])
by (auto simp: Bounded-mat-def Bounded-vec-def less-vec-def)
thus thesis by blast
qed

```

lemmas *small-integer-solution-nonstrict-hadamard* =

small-integer-solution-nonstrict[*OF det-bound-hadamard, unfolded det-bound-hadamard-def*]

end

16 Integer Hull

We define the integer hull of a polyhedron, i.e., the convex hull of all integer solutions. Moreover, we prove the result of Meyer that the integer hull of a polyhedron defined by an integer matrix is again a polyhedron, and give bounds for a corresponding decomposition theorem.

theory *Integer-Hull*

```

imports
  Decomposition-Theorem
  Mixed-Integer-Solutions
begin

context gram-schmidt
begin
definition integer-hull  $P = \text{convex-hull } (P \cap \mathbb{Z}_v)$ 

lemma integer-hull-mono:  $P \subseteq Q \implies \text{integer-hull } P \subseteq \text{integer-hull } Q$ 
  unfolding integer-hull-def
  by (intro convex-hull-mono, auto)

end

lemma abs-neg-floor:  $|of\text{-int } b| \leq Bnd \implies -(\text{floor } Bnd) \leq b$ 
  using abs-le-D2 floor-mono by fastforce

lemma abs-pos-floor:  $|of\text{-int } b| \leq Bnd \implies b \leq \text{floor } Bnd$ 
  using abs-le-D1 le-floor-iff by auto

context gram-schmidt-floor
begin

lemma integer-hull-integer-cone: assumes  $C: C \subseteq \text{carrier-vec } n$ 
  and  $CI: C \subseteq \mathbb{Z}_v$ 
  shows  $\text{integer-hull } (\text{cone } C) = \text{cone } C$ 
proof
  have  $\text{cone } C \cap \mathbb{Z}_v \subseteq \text{cone } C$  by blast
  thus  $\text{integer-hull } (\text{cone } C) \subseteq \text{cone } C$ 
    using cone-cone[OF C] convex-cone[OF C] convex-hull-mono
    unfolding integer-hull-def convex-def by metis
  {
    fix  $x$ 
    assume  $x \in \text{cone } C$ 
    then obtain  $D$  where  $\text{finD}: \text{finite } D$  and  $DC: D \subseteq C$  and  $x: x \in \text{finite-cone } D$ 
      unfolding cone-def by auto
      from  $DC$   $C$   $CI$  have  $D: D \subseteq \text{carrier-vec } n$  and  $DI: D \subseteq \mathbb{Z}_v$  by auto
      from  $D$   $x$   $\text{finD}$  have  $x \in \text{finite-cone } (D \cup \{0_v\})$  using finite-cone-mono[of
       $D \cup \{0_v\} D]$  by auto
      then obtain  $l$  where  $x: \text{lincomb } l (D \cup \{0_v\}) = x$ 
        and  $l: l \text{ ' } (D \cup \{0_v\}) \subseteq \{t. t \geq 0\}$ 
        using  $\text{finD}$  unfolding finite-cone-def nonneg-lincomb-def by auto
      define  $L$  where  $L = \text{sum } l (D \cup \{0_v\})$ 
      define  $L\text{-sup} :: 'a$  where  $L\text{-sup} = of\text{-int } (\text{floor } L + 1)$ 
      have  $L\text{-sup} \geq L$  using floor-correct[of L] unfolding L-sup-def by linarith
      have  $L \geq 0$  unfolding L-def using sum-nonneg[of - l] l by blast
      hence  $L\text{-sup} \geq 1$  unfolding L-sup-def by simp

```

hence $L\text{-sup} > 0$ by *fastforce*

let $?f = \lambda y. \text{if } y = 0_v \text{ then } L\text{-sup} - L \text{ else } 0$
 have $\text{lincomb } ?f \{0_v \ n\} = 0_v \ n$
 using *already-in-span*[of $\{ \} \ 0_v \ n$] *lincomb-in-span* *local.span-empty*
 by *auto*
 moreover have $\text{lincomb } ?f (D - \{0_v \ n\}) = 0_v \ n$
 by (*rule lincomb-zero, insert D, auto*)
 ultimately have $\text{lincomb } ?f (D \cup \{0_v \ n\}) = 0_v \ n$
 using *lincomb-vec-diff-add*[of $D \cup \{0_v \ n\} \ \{0_v \ n\}$] *D finD* by *simp*
 hence *lcomb-f*: $\text{lincomb } (\lambda y. l \ y + ?f \ y) (D \cup \{0_v \ n\}) = x$
 using *lincomb-sum*[of $D \cup \{0_v \ n\} \ l \ ?f$] *finD D x* by *simp*
 have $\text{sum } ?f (D \cup \{0_v \ n\}) = L\text{-sup} - L$
 by (*simp add: sum.subset-diff*[of $\{0_v \ n\} \ D \cup \{0_v \ n\} \ ?f$] *finD*)
 hence $\text{sum } (\lambda y. l \ y + ?f \ y) (D \cup \{0_v \ n\}) = L\text{-sup}$
 using *l L-def* by *auto*
 moreover have $(\lambda y. l \ y + ?f \ y) \ ' (D \cup \{0_v \ n\}) \subseteq \{t. t \geq 0\}$
 using $\langle L \leq L\text{-sup} \rangle \ l$ by *force*
 ultimately obtain l' where $x: \text{lincomb } l' (D \cup \{0_v \ n\}) = x$
 and $l': l' \ ' (D \cup \{0_v \ n\}) \subseteq \{t. t \geq 0\}$
 and $\text{sum-}l': \text{sum } l' (D \cup \{0_v \ n\}) = L\text{-sup}$
 using *lcomb-f* by *blast*

let $?D' = \{L\text{-sup} \cdot_v \ v \mid v. v \in D \cup \{0_v \ n\}\}$
 have *Did*: $?D' = (\lambda v. L\text{-sup} \cdot_v \ v) \ ' (D \cup \{0_v \ n\})$ by *force*
 define l'' where $l'' = (\lambda y. l' ((1 / L\text{-sup}) \cdot_v \ y) / L\text{-sup})$
 obtain lD where *dist*: *distinct* lD and $lD: D \cup \{0_v \ n\} = \text{set } lD$
 using *finite-distinct-list*[of $D \cup \{0_v \ n\}$] *finD* by *auto*
 let $?lD' = \text{map } ((\cdot_v) \ L\text{-sup}) \ lD$
 have *dist'*: *distinct* $?lD'$
 using *distinct-smult-nonneg*[*OF - dist*] $\langle L\text{-sup} > 0 \rangle$ by *fastforce*

have $x': \text{lincomb } l'' ?D' = x$ **unfolding** $x[\text{symmetric}] \ l''\text{-def}$
unfolding *lincomb-def* *Did*

proof (*subst finsum-reindex*)

from $\langle L\text{-sup} > 0 \rangle$ *smult-vec-nonneg-eq*[of $L\text{-sup}$] **show** *inj-on* $((\cdot_v) \ L\text{-sup}) (D \cup \{0_v \ n\})$

 by (*auto simp: inj-on-def*)

show $(\lambda v. l' (1 / L\text{-sup} \cdot_v \ v) / L\text{-sup} \cdot_v \ v) \in (\cdot_v) \ L\text{-sup} \ ' (D \cup \{0_v \ n\}) \rightarrow$
carrier-vec n

 using D by *auto*

from $\langle L\text{-sup} > 0 \rangle$ **have** $L\text{-sup} \neq 0$ by *auto*

then show $(\bigoplus_{v \in D \cup \{0_v \ n\}} l' (1 / L\text{-sup} \cdot_v \ (L\text{-sup} \cdot_v \ x)) / L\text{-sup} \cdot_v \ (L\text{-sup} \cdot_v \ x)) =$

$(\bigoplus_{v \in D \cup \{0_v \ n\}} l' \ v \cdot_v \ v)$

 by (*intro finsum-cong, insert D, auto simp: smult-smult-assoc*)

qed

have $D \cup \{0_v \ n\} \subseteq \text{cone } C$ **using** *set-in-cone*[*OF C*] *DC zero-in-cone* by *blast*

hence $D': ?D' \subseteq \text{cone } C$ **using** *cone-smult*[of $L\text{-sup}, \text{OF} - C$] $\langle L\text{-sup} > 0 \rangle$ by

auto

have $D \cup \{0_v\} \subseteq \mathbb{Z}_v$ **unfolding** *zero-vec-def* **using** *DI Ints-vec-def* **by** *auto*
moreover have $L\text{-sup} \in \mathbb{Z}$ **unfolding** *L-sup-def* **by** *auto*
ultimately have $D'I: ?D' \subseteq \mathbb{Z}_v$ **unfolding** *Ints-vec-def* **by** *force*

have $1 = \text{sum } l' (D \cup \{0_v\}) * (1 / L\text{-sup})$ **using** *sum-l' <L-sup > 0* **by** *auto*

also have $\text{sum } l' (D \cup \{0_v\}) = \text{sum-list } (\text{map } l' lD)$

using *sum.distinct-set-conv-list[OF dist]* *lD* **by** *auto*

also have $\text{map } l' lD = \text{map } (l' \circ ((\cdot)_v) (1 / L\text{-sup})) ?lD'$

using *smult-smult-assoc[of 1 / L-sup L-sup]* *<L-sup > 0* **by** *simp*

by *(simp add: comp-assoc)*

also have $l' \circ ((\cdot)_v) (1 / L\text{-sup}) = (\lambda x. l' ((1 / L\text{-sup}) \cdot_v x))$ **by** *(rule*

comp-def)

also have $\text{sum-list } (\text{map } \dots ?lD') * (1 / L\text{-sup}) =$

$\text{sum-list } (\text{map } (\lambda y. l' (1 / L\text{-sup}) \cdot_v y) * (1 / L\text{-sup})) ?lD'$

using *sum-list-mult-const[of - 1 / L-sup ?lD']* **by** *presburger*

also have $\dots = \text{sum-list } (\text{map } l'' ?lD')$

unfolding *l''-def* **using** *<L-sup > 0* **by** *simp*

also have $\dots = \text{sum } l'' (\text{set } ?lD')$ **using** *sum.distinct-set-conv-list[OF dist']*

by *metis*

also have $\text{set } ?lD' = ?D'$ **using** *lD* **by** *auto*

finally have $\text{sum-l}': \text{sum } l'' ?D' = 1$ **by** *auto*

moreover have $l'' \text{ ' } ?D' \subseteq \{t. t \geq 0\}$

proof

fix y

assume $y \in l'' \text{ ' } ?D'$

then obtain x **where** $y = l'' x$ **and** $x \in ?D'$ **by** *blast*

then obtain v **where** $v \in D \cup \{0_v\}$ **and** $x = L\text{-sup} \cdot_v v$ **by** *blast*

hence $0 \leq l' v / L\text{-sup}$ **using** *l' <L-sup > 0* **by** *fastforce*

also have $\dots = l'' x$ **unfolding** *x l''-def*

using *smult-smult-assoc[of 1 / L-sup L-sup v]* *<L-sup > 0* **by** *simp*

finally show $y \in \{t. t \geq 0\}$ **using** y **by** *blast*

qed

moreover have *finite* $?D'$ **using** *finD* **by** *simp*

ultimately have $x \in \text{integer-hull } (\text{cone } C)$

unfolding *integer-hull-def* *convex-hull-def*

using $x' D' D'I$ *convex-lincomb-def[of l'' ?D' x]*

nonneg-lincomb-def[of l'' ?D' x] **by** *fast*

}

thus $\text{cone } C \subseteq \text{integer-hull } (\text{cone } C)$ **by** *blast*

qed

theorem *decomposition-theorem-integer-hull-of-polyhedron:*

assumes $db: \text{is-det-bound } db$

and $A: A \in \text{carrier-mat } nr \ n$

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

```

and AI:  $A \in \mathbb{Z}_m$ 
and bI:  $b \in \mathbb{Z}_v$ 
and P:  $P = \text{polyhedron } A \ b$ 
and Bnd:  $\text{of-int } Bnd \geq \text{Max } (\text{abs } ' (\text{elements-mat } A \cup \text{vec-set } b))$ 
shows  $\exists H \ C. H \cup C \subseteq \text{carrier-vec } n \cap \mathbb{Z}_v$ 
 $\wedge H \subseteq \text{Bounded-vec } (\text{of-nat } (n + 1) * \text{of-int } (db \ n \ (\text{max } 1 \ Bnd)))$ 
 $\wedge C \subseteq \text{Bounded-vec } (\text{of-int } (db \ n \ (\text{max } 1 \ Bnd)))$ 
 $\wedge \text{finite } (H \cup C)$ 
 $\wedge \text{integer-hull } P = \text{convex-hull } H + \text{cone } C$ 
proof –
define MBnd where  $MBnd = \text{Max } (\text{abs } ' (\text{elements-mat } A \cup \text{set}_v \ b))$ 
define DBnd :: 'a where  $DBnd = \text{of-int } (db \ n \ (\text{max } 1 \ Bnd))$ 
define nBnd where  $nBnd = \text{of-nat } (n+1) * DBnd$ 
have DBnd0:  $DBnd \geq 0$  unfolding DBnd-def of-int-0-le-iff
  by (rule is-det-bound-ge-zero[OF db], auto)
have Pn:  $P \subseteq \text{carrier-vec } n$  unfolding P polyhedron-def by auto
have  $A \in \text{Bounded-mat } MBnd \wedge b \in \text{Bounded-vec } MBnd$ 
  unfolding MBnd-def Bounded-mat-elements-mat Bounded-vec-vec-set
  by (intro ballI conjI Max-ge finite-imageI imageI finite-UnI, auto)
hence  $A \in \text{Bounded-mat } (\text{of-int } Bnd) \wedge b \in \text{Bounded-vec } (\text{of-int } Bnd)$ 
  using Bounded-mat-mono[OF Bnd] Bounded-vec-mono[OF Bnd] unfolding
  MBnd-def by auto
from decomposition-theorem-polyhedra-1[OF A b P, of db Bnd] db AI bI this
obtain QQ Q C where  $C: C \subseteq \text{carrier-vec } n$  and finC: finite C
  and QQ:  $QQ \subseteq \text{carrier-vec } n$  and finQ: finite QQ and BndQQ:  $QQ \subseteq$ 
  Bounded-vec DBnd
  and P:  $P = Q + \text{cone } C$ 
  and Q-def:  $Q = \text{convex-hull } QQ$ 
  and CI:  $C \subseteq \mathbb{Z}_v$  and BndC:  $C \subseteq \text{Bounded-vec } DBnd$ 
  by (auto simp: DBnd-def)
define Bnd' where  $Bnd' = \text{of-nat } n * DBnd$ 
note coneC = cone-iff-finite-cone[OF C finC]
have Q:  $Q \subseteq \text{carrier-vec } n$  unfolding Q-def using convex-hull-carrier[OF QQ]
  .
define B where  $B = \{x. \exists a \ D. \text{nonneg-lincomb } a \ D \ x \wedge D \subseteq C \wedge \text{lin-indpt } D$ 
 $\wedge (\forall d \in D. a \ d \leq 1)\}$ 
  {
    fix b
    assume  $b \in B$ 
    then obtain a D where  $b = \text{lincomb } a \ D$  and DC:  $D \subseteq C$ 
    and linD: lin-indpt D and bnd-a:  $\forall d \in D. 0 \leq a \ d \wedge a \ d \leq 1$ 
    by (force simp: B-def nonneg-lincomb-def)
    from DC C have D:  $D \subseteq \text{carrier-vec } n$  by auto
    from DC finC have finD: finite D by (metis finite-subset)
    from D linD finD have cardD:  $\text{card } D \leq n$  using dim-is-n li-le-dim(2) by auto
    from BndC DC have BndD:  $D \subseteq \text{Bounded-vec } DBnd$  by auto
    from lincomb-card-bound[OF this D DBnd0 - cardD, of a, folded b] bnd-a
    have  $b \in \text{Bounded-vec } Bnd'$  unfolding Bnd'-def by force
  }

```

```

hence  $BndB$ :  $B \subseteq Bounded\text{-}vec\ Bnd'$  ..
from  $BndQQ$  have  $BndQ$ :  $Q \subseteq Bounded\text{-}vec\ DBnd$  unfolding  $Q\text{-}def$  using  $QQ$ 
by (metis convex-hull-bound)
have  $B$ :  $B \subseteq carrier\text{-}vec\ n$ 
  unfolding  $B\text{-}def\ nonneg\text{-}lincomb\text{-}def$  using  $C$  by auto
from  $Q\ B$  have  $QB$ :  $Q + B \subseteq carrier\text{-}vec\ n$  by (auto elim!: set-plus-elim)
from  $sum\text{-}in\text{-}Bounded\text{-}vecI$ [of - DBnd - Bnd' n]  $BndQ\ BndB\ B\ Q$ 
have  $Q + B \subseteq Bounded\text{-}vec\ (DBnd + Bnd')$  by (auto elim!: set-plus-elim)
also have  $DBnd + Bnd' = nBnd$  unfolding  $nBnd\text{-}def\ Bnd'\text{-}def$  by (simp add:
algebra-simps)
finally have  $QB\text{-}Bnd$ :  $Q + B \subseteq Bounded\text{-}vec\ nBnd$  by blast
have  $finQBZ$ :  $finite\ ((Q + B) \cap \mathbb{Z}_v)$ 
proof (rule finite-subset[OF subsetI])
  define  $ZBnd$  where  $ZBnd = floor\ nBnd$ 
  let  $?vecs = set\ (map\ vec\text{-}of\text{-}list\ (concat\text{-}lists\ (map\ (\lambda\ i.\ map\ (of\text{-}int\ ::\ - \Rightarrow\ 'a)\$ 
 $[-ZBnd..ZBnd])\ [0..<n])))$ 
  have  $id$ :  $?vecs = vec\text{-}of\text{-}list\ '$ 
     $\{as.\ length\ as = n \wedge (\forall\ i < n.\ \exists\ b.\ as\ !\ i = of\text{-}int\ b \wedge b \in \{-ZBnd..ZBnd\})\}$ 
    unfolding set-map by (rule image-cong, auto)
  show  $finite\ ?vecs$  by (rule finite-set)
  fix  $x$ 
  assume  $x \in (Q + B) \cap \mathbb{Z}_v$ 
  hence  $xQB$ :  $x \in Q + B$  and  $xI$ :  $x \in \mathbb{Z}_v$  by auto
  from  $xQB\ QB\text{-}Bnd\ QB$  have  $xBnd$ :  $x \in Bounded\text{-}vec\ nBnd$  and  $x$ :  $x \in carrier\text{-}vec\ n$  by auto
  have  $xid$ :  $x = vec\text{-}of\text{-}list\ (list\text{-}of\text{-}vec\ x)$  by auto
  show  $x \in ?vecs$  unfolding  $id$ 
  proof (subst xid, intro imageI CollectI conjI allI impI)
    show  $length\ (list\text{-}of\text{-}vec\ x) = n$  using  $x$  by auto
    fix  $i$ 
    assume  $i$ :  $i < n$ 
    have  $id$ :  $list\text{-}of\text{-}vec\ x\ !\ i = x\ \$\ i$  using  $i\ x$  by auto
    from  $xBnd$ [unfolded Bounded-vec-def]  $i\ x$  have  $xiBnd$ :  $abs\ (x\ \$\ i) \leq nBnd$ 
by auto
    from  $xI$ [unfolded Ints-vec-def]  $i\ x$  have  $x\ \$\ i \in \mathbb{Z}$  by auto
    then obtain  $b$  where  $b$ :  $x\ \$\ i = of\text{-}int\ b$  unfolding  $Ints\text{-}def$  by blast
    show  $\exists\ b.\ list\text{-}of\text{-}vec\ x\ !\ i = of\text{-}int\ b \wedge b \in \{-ZBnd..ZBnd\}$  unfolding  $id$ 
 $ZBnd\text{-}def$ 
    using  $xiBnd$  unfolding  $b$  by (intro exI[of - b], auto intro!: abs-neg-floor
abs-pos-floor)
  qed
qed
have  $QBZ$ :  $(Q + B) \cap \mathbb{Z}_v \subseteq carrier\text{-}vec\ n$  using  $QB$  by auto
from decomposition-theorem-polyhedra-2[OF QBZ finQBZ, folded integer-hull-def,
OF C finC refl]
obtain  $A'\ b'\ nr'$  where  $A'$ :  $A' \in carrier\text{-}mat\ nr'\ n$  and  $b'$ :  $b' \in carrier\text{-}vec\ nr'$ 
and  $IH$ :  $integer\text{-}hull\ (Q + B) + cone\ C = polyhedron\ A'\ b'$ 
by auto
{

```

```

fix p
assume p ∈ P ∩ Zv
hence pI: p ∈ Zv and p: p ∈ Q + cone C unfolding P by auto
from set-plus-elim[OF p] obtain q c where
  pqc: p = q + c and qQ: q ∈ Q and cC: c ∈ cone C by auto
from qQ Q have q: q ∈ carrier-vec n by auto
from Caratheodory-theorem[OF C] cC
obtain D where cD: c ∈ finite-cone D and DC: D ⊆ C and linD: lin-indpt
D by auto
from DC C have D: D ⊆ carrier-vec n by auto
from DC finC have finD: finite D by (metis finite-subset)
from cD finD
obtain a where nonneg-lincomb a D c unfolding finite-cone-def by auto
hence caD: c = lincomb a D and a0:  $\bigwedge d. d \in D \implies a d \geq 0$ 
  unfolding nonneg-lincomb-def by auto
define a1 where a1 = ( $\lambda c. a c - \text{of-int } (\text{floor } (a c))$ )
define a2 where a2 = ( $\lambda c. \text{of-int } (\text{floor } (a c)) :: 'a$ )
define d where d = lincomb a2 D
define b where b = lincomb a1 D
have b: b ∈ carrier-vec n and d: d ∈ carrier-vec n unfolding d-def b-def using
D by auto
have bB: b ∈ B unfolding B-def b-def nonneg-lincomb-def
proof (intro CollectI exI[of - a1] exI[of - D] conjI ballI refl subsetI linD)
  show x ∈ a1 ' D  $\implies 0 \leq x$  for x using a0 unfolding a1-def by auto
  show a1 c ≤ 1 for c unfolding a1-def by linarith
qed (insert DC, auto)
have cbd: c = b + d unfolding b-def d-def caD lincomb-sum[OF finD D,
symmetric]
  by (rule lincomb-cong[OF refl D], auto simp: a1-def a2-def)
have nonneg-lincomb a2 D d unfolding d-def nonneg-lincomb-def
  by (intro allI conjI refl subsetI, insert a0, auto simp: a2-def)
hence dC: d ∈ cone C unfolding cone-def finite-cone-def using finC finD DC
by auto
have p: p = (q + b) + d unfolding pqc cbd using q b d by auto
have dI: d ∈ Zv using CI DC C unfolding d-def indexed-Ints-vec-UNIV
  by (intro lincomb-indexed-Ints-vec, auto simp: a2-def)
from diff-indexed-Ints-vec[of - - - UNIV, folded indexed-Ints-vec-UNIV, OF -
d pI dI, unfolded p]
  have q + b + d - d ∈ Zv using q b d by auto
  also have q + b + d - d = q + b using q b d by auto
  finally have qbI: q + b ∈ Zv by auto
have p ∈ integer-hull (Q + B) + cone C unfolding p integer-hull-def
  by (intro set-plus-intro dC set-mp[OF set-in-convex-hull] IntI qQ bB qbI,
insert Q B,
auto elim!: set-plus-elim)
}
hence P ∩ Zv ⊆ integer-hull (Q + B) + cone C by auto
hence one-dir: integer-hull P ⊆ integer-hull (Q + B) + cone C unfolding IH
unfolding integer-hull-def using convex-convex-hull[OF polyhedra-are-convex[OF

```

$A' b' \text{ refl}]$
convex-hull-mono **by** *blast*
have $\text{integer-hull } (Q + B) + \text{cone } C \subseteq \text{integer-hull } P + \text{cone } C$ **unfolding** P
proof (*intro set-plus-mono2 subset-refl integer-hull-mono*)
show $B \subseteq \text{cone } C$ **unfolding** $B\text{-def cone-def finite-cone-def}$ **using** $\text{finite-subset}[OF$
 $\text{- fin}C]$ **by** *auto*
qed
also have $\dots = \text{integer-hull } P + \text{integer-hull } (\text{cone } C)$
using $\text{integer-hull-integer-cone}[OF C CI]$ **by** *simp*
also have $\dots = \text{convex-hull } (P \cap \mathbb{Z}_v) + \text{convex-hull } (\text{cone } C \cap \mathbb{Z}_v)$
unfolding integer-hull-def **by** *simp*
also have $\dots = \text{convex-hull } ((P \cap \mathbb{Z}_v) + (\text{cone } C \cap \mathbb{Z}_v))$
by (*rule convex-hull-sum[symmetric], insert Pn cone-carrier[OF C], auto*)
also have $\dots \subseteq \text{convex-hull } ((P + \text{cone } C) \cap \mathbb{Z}_v)$
proof (*rule convex-hull-mono*)
show $P \cap \mathbb{Z}_v + \text{cone } C \cap \mathbb{Z}_v \subseteq (P + \text{cone } C) \cap \mathbb{Z}_v$
using $\text{add-indexed-Ints-vec}[of - n - UNIV, folded \text{indexed-Ints-vec-UNIV}]$
 $\text{cone-carrier}[OF C] Pn$
by (*auto elim!: set-plus-elim*)
qed
also have $\dots = \text{integer-hull } (P + \text{cone } C)$ **unfolding** $\text{integer-hull-def ..}$
also have $P + \text{cone } C = P$
proof –
have $CC: \text{cone } C \subseteq \text{carrier-vec } n$ **using** C **by** (*rule cone-carrier*)
have $P + \text{cone } C = Q + (\text{cone } C + \text{cone } C)$ **unfolding** P
by (*rule assoc-add-vecset[symmetric, OF Q CC CC]*)
also have $\text{cone } C + \text{cone } C = \text{cone } C$ **by** (*rule cone-add-cone[OF C]*)
finally show $?thesis$ **unfolding** P .
qed
finally have $\text{integer-hull } (Q + B) + \text{cone } C \subseteq \text{integer-hull } P$.
with *one-dir* **have** $id: \text{integer-hull } P = \text{integer-hull } (Q + B) + \text{cone } C$ **by** *auto*
show $?thesis$ **unfolding** id **unfolding** $\text{integer-hull-def DBnd-def[symmetric]}$
 $nBnd\text{-def[symmetric]}$
proof (*rule exI[of - (Q + B) \cap \mathbb{Z}_v], intro exI[of - C] conjI refl BndC]*)
from $QB\text{-Bnd}$ **show** $(Q + B) \cap \mathbb{Z}_v \subseteq \text{Bounded-vec } nBnd$ **by** *auto*
show $(Q + B) \cap \mathbb{Z}_v \cup C \subseteq \text{carrier-vec } n \cap \mathbb{Z}_v$
using $QB C CI$ **by** *auto*
show $\text{finite } ((Q + B) \cap \mathbb{Z}_v \cup C)$ **using** $\text{finQBZ fin}C$ **by** *auto*
qed
qed

corollary *integer-hull-of-polyhedron*: **assumes** $A: A \in \text{carrier-mat } nr n$
and $b: b \in \text{carrier-vec } nr$
and $AI: A \in \mathbb{Z}_m$
and $bI: b \in \mathbb{Z}_v$
and $P: P = \text{polyhedron } A b$
shows $\exists A' b' nr'. A' \in \text{carrier-mat } nr' n \wedge b' \in \text{carrier-vec } nr' \wedge$
 $\text{integer-hull } P = \text{polyhedron } A' b'$
proof –

obtain Bnd **where** $Bnd: Max (abs \text{ ' } (elements\text{-}mat\ A \cup set_v\ b)) \leq of\text{-}int\ Bnd$
by (*meson ex-le-of-int*)
from *decomposition-theorem-integer-hull-of-polyhedron*[*OF det-bound-fact A b AI bI P Bnd*]
obtain $H\ C$
where $HC: H \cup C \subseteq carrier\text{-}vec\ n \cap \mathbb{Z}_v\ finite\ (H \cup C)$
and $decomp: integer\text{-}hull\ P = convex\text{-}hull\ H + cone\ C$ **by** *auto*
show *?thesis*
by (*rule decomposition-theorem-polyhedra-2*[*OF - - - decomp*], *insert HC, auto*)
qed

corollary *small-integer-solution-nonstrict-via-decomp*: **fixes** $A :: 'a\ mat$

assumes $db: is\text{-}det\text{-}bound\ db$
and $A: A \in carrier\text{-}mat\ nr\ n$
and $b: b \in carrier\text{-}vec\ nr$
and $AI: A \in \mathbb{Z}_m$
and $bI: b \in \mathbb{Z}_v$
and $Bnd: of\text{-}int\ Bnd \geq Max (abs \text{ ' } (elements\text{-}mat\ A \cup vec\text{-}set\ b))$
and $x: x \in carrier\text{-}vec\ n$
and $xI: x \in \mathbb{Z}_v$
and $sol: A *_v\ x \leq b$

shows $\exists y.$

$y \in carrier\text{-}vec\ n \wedge$

$y \in \mathbb{Z}_v \wedge$

$A *_v\ y \leq b \wedge$

$y \in Bounded\text{-}vec\ (of\text{-}nat\ (n+1) * of\text{-}int\ (db\ n\ (max\ 1\ Bnd)))$

proof –

from $x\ sol$ **have** $x \in polyhedron\ A\ b$ **unfolding** *polyhedron-def* **by** *auto*

with $xI\ x$ **have** $xsol: x \in integer\text{-}hull\ (polyhedron\ A\ b)$ **unfolding** *integer-hull-def*

by (*meson IntI convex-hull-mono in-mono inf-sup-ord(1) inf-sup-ord(2) set-in-convex-hull*)

from *decomposition-theorem-integer-hull-of-polyhedron*[*OF db A b AI bI refl Bnd*]

obtain $H\ C$ **where** $HC: H \cup C \subseteq carrier\text{-}vec\ n \cap \mathbb{Z}_v$

$H \subseteq Bounded\text{-}vec\ (of\text{-}nat\ (n + 1) * of\text{-}int\ (db\ n\ (max\ 1\ Bnd)))$

finite $(H \cup C)$ **and**

id: $integer\text{-}hull\ (polyhedron\ A\ b) = convex\text{-}hull\ H + cone\ C$

by *auto*

from $xsol$ [*unfolded id*] **have** $H \neq \{\}$ **unfolding** *set-plus-def* **by** *auto*

then obtain h **where** $hH: h \in H$ **by** *auto*

with *set-in-convex-hull* **have** $h \in convex\text{-}hull\ H$ **using** HC **by** *auto*

moreover have $0_v\ n \in cone\ C$ **by** (*intro zero-in-cone*)

ultimately have $h + 0_v\ n \in integer\text{-}hull\ (polyhedron\ A\ b)$ **unfolding** *id* **by** *auto*

also have $h + 0_v\ n = h$ **using** $hH\ HC$ **by** *auto*

also have $integer\text{-}hull\ (polyhedron\ A\ b) \subseteq convex\text{-}hull\ (polyhedron\ A\ b)$

unfolding *integer-hull-def* **by** (*rule convex-hull-mono, auto*)

also have $convex\text{-}hull\ (polyhedron\ A\ b) = polyhedron\ A\ b$ **using** $A\ b$

using *convex-convex-hull polyhedra-are-convex* **by** *blast*

finally have $h: h \in carrier\text{-}vec\ n\ A *_v\ h \leq b$ **unfolding** *polyhedron-def* **by** *auto*

show *?thesis*

by (intro exI[of - h] conjI h, insert HC hH, auto)
qed

lemmas small-integer-solution-nonstrict-via-decomp-hadamard =
small-integer-solution-nonstrict-via-decomp[OF det-bound-hadamard, unfolded det-bound-hadamard-def]

end
end

References

- [1] B. Dutertre and L. M. de Moura. A fast linear-arithmetic solver for DPLL(T). In *Proc. Computer Aided Verification*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006. Extended version available as Technical Report, CSL-06-01, SRI International.
- [2] C. H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.
- [3] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.