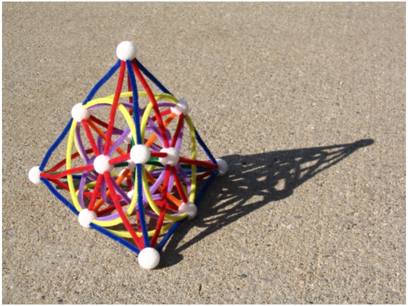
Spreads and Packings of PG(3,2), Formally!



Picture taken from David A. Richter
http://homepages.wmich.edu/~drichter/projectivespace.htm

Nicolas Magaud - Université de Strasbourg, France

ADG 2021 : International Workshop on Automated Deduction in Geometry
September 15-17, 2021

Outline

1. Projective Geometry, esp. in 3D
2. Data Structures for Finite Projective Geometry
3. Spreads and Packings
4. Formal Proofs
5. Conclusions and Future Work

Projective Space Geometry

- Context
  - Incidence Geometry
    - points, lines and an incidence relation
  - Projective Incidence Geometry
    - in 2D : 2 lines always intersect
    - in 3D : Pasch's axiom
  - Simple description : only 6 axioms
- Goal :
  - Specifying some finite models of projective geometry
  - Formally checking the axioms
  - Computing spreads and packings
  - Proving some of their properties
  - Taking Coq to its limits (w.r.t. specification and w.r.t. proof)

Objects and Operations

- Objects : Point, Line

```
Parameter Point, Line : Type.
```

- Incidence relation : incid_lp

```
Parameter incid_lp : Point -> Line -> bool.
```

- Boolean equalities on points and lines : eqP, eqL

```
Parameter eqP : Point -> Point -> bool.
Parameter eqL : Line -> Line -> bool.
```

- All distinct points and/or lines : dist_3p, dist_4p, . . . , dist_5l

```
Definition dist_3p (A B C :Point) : bool :=
(negb (eqP A B)) && (negb (eqP A C)) && (negb (eqP B C)).

Definition dist_4p (A B C D:Point) : bool := ...
Definition dist_5l (l1 l2 l3:Line) : bool := ...
Definition dist_5l (l1 l2 l3 l4 l5:Line) : bool := ...
```

- Intersection of 2 lines : Intersect_In

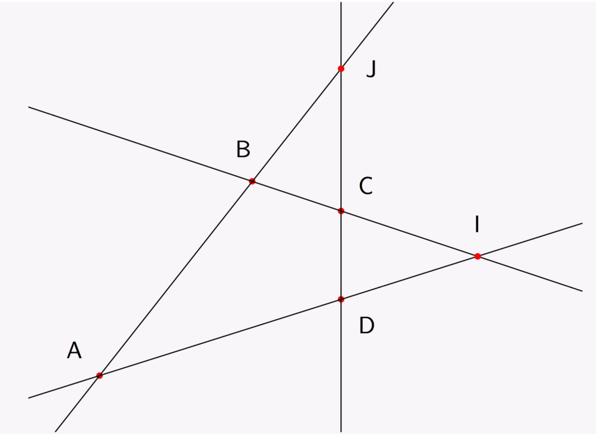
```
Definition Intersect_In (l1 l2 :Line) (P:Point) :=
incid_lp P l1 && incid_lp P l2.
```

Axioms for Projective Space Geometry : from a geometry point of view

- a1 : throught 2 points, there is one line.
- uniqueness : Given 2 points and 2 lines, if the 2 points are both on both lines, either the points are equal, or the lines.
- a2 : Pasch's axiom (if 2 lines intersect. . .).
- a3_1 : Each line has at least 3 points.
- a3_2 : There exists 2 lines which do not intersect (dim>2).
- a3_3 : Given 3 distinct lines, there exists a fourth one which intersects with all three (dim<=3).

Axioms for Projective Space Geometry : from a geometry point of view

- Pasch's axiom



Axioms for Projective Space Geometry : from a geometry point of view

```
Axiom a1_exists : forall A B : Point, { l : Line| incid_lp A l && incid_lp B l}.

Axiom uniqueness : forall (A B :Point)(l1 l2:Line),
incid_lp A l1 -> incid_lp B l1 -> incid_lp A l2 -> incid_lp B l2 -> A = B \/ l1 = l2.

Axiom a2 : forall A B C D:Point, forall lAB lCD lAC lBD :Line, dist_4p A B C D ->
incid_lp A lAB && incid_lp B lAB -> incid_lp C lCD && incid_lp D lCD ->
incid_lp A lAC && incid_lp C lAC -> incid_lp B lBD && incid_lp D lBD ->
(exists I:Point, incid_lp I lAB && incid_lp I lCD) ->
exists J:Point, incid_lp J lAC && incid_lp J lBD.

Axiom a3_1 : forall l:Line,
{A:Point & {B:Point & {C:Point |
(dist_3p A B C) && (incid_lp A l && incid_lp B l && incid_lp C l)}}}.

Axiom a3_2 : exists l1:Line, exists l2:Line,
forall p:Point,  (incid_lp p l1 && incid_lp p l2).

Axiom a3_3 : forall l1 l2 l3:Line, dist_3l l1 l2 l3 ->
exists l4 :Line, exists J1:Point, exists J2:Point, exists J3:Point,
Intersect_In l1 l4 J1 && Intersect_In l2 l4 J2 && Intersect_In l3 l4 J3.
```

Axioms for Projective Space Geometry : from a logic point of view

```
Axiom a1_exists : forall A B : Point, { l : Line| incid_lp A l && incid_lp B l}.

Axiom uniqueness : forall (A B :Point)(l1 l2:Line),
incid_lp A l1 -> incid_lp B l1 -> incid_lp A l2 -> incid_lp B l2 -> A = B \/ l1 = l2.

Axiom a2 : forall A B C D:Point, forall lAB lCD lAC lBD :Line, dist_4p A B C D ->
incid_lp A lAB && incid_lp B lAB -> incid_lp C lCD && incid_lp D lCD ->
incid_lp A lAC && incid_lp C lAC -> incid_lp B lBD && incid_lp D lBD ->
(exists I:Point, incid_lp I lAB && incid_lp I lCD) ->
exists J:Point, incid_lp J lAC && incid_lp J lBD.

Axiom a3_1 : forall l:Line,
{A:Point & {B:Point & {C:Point |
(dist_3p A B C) && (incid_lp A l && incid_lp B l && incid_lp C l)}}}.

Axiom a3_2 : exists l1:Line, exists l2:Line,
forall p:Point,  (incid_lp p l1 && incid_lp p l2).

Axiom a3_3 : forall l1 l2 l3:Line, dist_3l l1 l2 l3 ->
exists l4 :Line, exists J1:Point, exists J2:Point, exists J3:Point,
Intersect_In l1 l4 J1 && Intersect_In l2 l4 J2 && Intersect_In l3 l4 J3.
```

Finite Projective Spaces PG(3,q)

	# points	# lines	# points per line
$PG(3,2)$	15	35	3
$PG(3,3)$	40	130	4
$PG(3,4)$	85	357	5
$PG(3,q)$	$(q^2+1)(q+1)$	$(q^2+q+1)(q^2+1)$	$q+1$

- By duality : # planes = # points.
- Describing the incidence relation of PG(3, q) :
  for each line, we provide the q+1 points which belong to it.

Outline

Coq specifications

- Point and Line as simple inductive types.
  - Case analysis is easy.
  - Finding a witness can be challenging.
    = trying each possible value and running the tactics.
  - Writing the specification is a bit boring (even worse with higher orders).

    ```
    Inductive Point := P0 | P1 | P2 | ... | P14.
    ```
- Automation
  - An external program to generate the specification
  - Also useful to generate the witnesses for existential proofs
    - incidence relation as a boolean predicate
    - decidable equality
    - *ad-hoc* order relation on points and lines
    - witnesses are computed in advance

Inductive Definitions and Functions

```
Inductive Point :=
| P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 .

Inductive Line :=
| L0 | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9
| L10 | L11 | L12 | L13 | L14 | L15 | L16 | L17 | L18 | L19
| L20 | L21 | L22 | L23 | L24 | L25 | L26 | L27 | L28 | L29
| L30 | L31 | L32 | L33 | L34 .

    Definition incid_lp (p:Point) (l:Line) : bool :=
match l with
| L0 => match p with P0 | P1 | P2 => true | _ => false end
| L1 => match p with P0 | P3 | P4 => true | _ => false end
| L2 => match p with P0 | P5 | P6 => true | _ => false end
| L3 => match p with P0 | P7 | P8 => true | _ => false end
| L4 => match p with P0 | P10 | P9 => true | _ => false end
| [...]
end.

Definition f_a3_3 (l1:Line) (l2:Line) (l3:Line) :=
match l3 with
 | L0 => match l2 with
       | L0 => match l1 with
                  | L0 => (L0,(P0,P0,P0))
                 | _ => (L0,(P0,P0,P0))
                end
      | _ => (L0, (P0,P0,P0))
       end
 | L1 => [...]
end.
```

Outline

Spreads and Packings of PG(3,q)

- A spread of PG(3,q) is a set of $q^2 + 1$ lines which are pairwise disjoint and thus partitions the set of points.
  - In PG(3,2), it corresponds to some sets of 5 lines.
- A packing of PG(3,q) is a set of $q^2 + q + 1$ spreads which are pairwise disjoint and thus partitions the set of lines.
  - In PG(3,2), it corresponds to some sets of 7 spreads.

Results for PG(3,2)

- There are 56 (isomorphic) spreads in PG(3,2).
- There are 240 packings in PG(3,2), upto isomorphism.
- These 240 packings are divided into 2 distinct equivalence classes (120 packings each).
- See Finite Projective Spaces of Three Dimensions (Hirschfeld,1985) for details

Outline

Spreads

- Formal definition of a spread

```
Definition is_partition (p q r s t: list Point) :bool :=
(forall_Point (fun x => inb x p || inb x q || inb x r || inb x s || inb x t)) &&
(forall_Point (fun x => negb (inb x p && inb x q && inb x r && inb x s && inb x t))).

Definition is_spread5 (l1 l2 l3 l4 l5:Line) : bool :=
disj_5l l1 l2 l3 l4 l5  && is_partition l1 l2 l3 l4.
```

- Spreads are computed externally.
- This exactly computes all the spreads of PG(3,2).

```
Lemma is_spread_descr : forall l1 l2 l3 l4 l5,
  (is_spread5 l1 l2 l3 l4 l5) <-> In [l1;l2;l3;l4;l5] spreads.
```

  - Proceeds by induction of the 5 variables l_1, l_2, l_3, l_4, l_5 (35^5 = 52 521 875 cases)

Spreads

- All these spreads are isomorphic.
  - Collineation : bijection which respects the incidence relation
  - There exists a collineation between each pair of spreads.

```
Lemma all_isomorphic_lemma : forall t1 t2 : list Line,
  In t1 spreads -> In t2 spreads -> are_isomorphic t1 t2.
```

  - Proof achieved using a circular argument
    $S_0 \to S_1 \to S_2 \to \ldots \to S_{55} \to S_0$

Packings

- We build the 240 packings of PG(3,2).
- We still need to show that they are no other packings of PG(3,2).
- We build 2 classes of isomorphim (120 packings each).
- We still need to show that they are actually two distinct classes.

Issues and Solutions

- Issues
  - Finding effective representations of objects for both computation and formal (automated reasoning)
  - Reaching the Frontiers/Limitations of Coq
- Solutions
  - Enhancing Coq abilities
  - Circumventing the limitations
    - Simplifying
    - Decomposing the proof
    - ...

Outline

Conclusions and Future Work

- Achievements
  - Some (Big) Formal Proofs in PG(3,2)
  - Pushing Coq to its limits
- Next steps (examples of state-of-the-art results)
  - Betten. *The packings of PG(3,3)* . 2015
  - Svetlana Topalova and Stela Zhelezova. *On transitive parallelisms of PG(3,4)*. 2017
- Using alternative provers
  - Lean
  - Z3

Thanks ! Questions ?

https://github.com/magaud/PG3q