Conception, spécification et preuve formelle d'un algorithme de calcul d'enveloppe convexe avec des hypercartes en Coq

Christophe Brun, Jean-François Dufourd, Nicolas Magaud

LSIIT UMR CNRS-ULP 7005
Université Louis Pasteur de Strasbourg
Pôle API, Bd Sébastien Brant, BP10413, 67412 Illkirch Cedex

Projet ANR : Galapagos : Géométrie, algorithmes et preuves

Présentation

- Objectif : Etude formelle dans le domaine de la modélisation et de la géométrie algorithmique
  - Améliorer les techniques de programmation
  - Assurer la correction des algorithmes
- Etude de cas : Algorithme incrémental de calcul de l'enveloppe convexe d'un ensemble fini de points du plan
- Originalité des moyens utilisés :
  - Spécifications et preuves formelles de programmes : calcul des constructions inductives et système d'aide à la preuve Coq
  - Modélisation géométrique à base topologique : hypercartes

Présentation

- Etapes de développement :
  1. Conception d'un algorithme fonctionnel
  2. Extraction automatique d'un programme en Ocaml
  3. Preuve formelle de sa correction : mise en évidence de nombreuses propriétés topologiques et géométriques
- Motivation : travail de description et de démonstration formelle pas pour prouver simplement un algorithme de plus mais pour poser les fondements afin d'obtenir un cadre commun de raisonnement et d'acquérir une certaine pratique de développement
- Algorithme naif : certification > complexité
  (stratégie de calcul très simple mais pas forcément efficace)

Travaux connexes

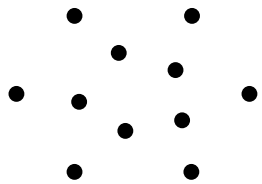
- Algorithmes de calcul d'enveloppes convexes
  - David PICHARDIE et Yves BERTOT, « Formalizing Convex Hull Algorithms ». *TPHOLs'01*, p.346-361. LNCS 2152, Springer Verlag, 2001.
  - Laura MEIKLE et Jacques FLEURIOT, « Mechanical Theorem Proving in Computational Geometry ». *Automated Deduction in Geometry*, p.1-18. LNCS 3763, Springer Verlag, 2004.
- Hypercartes par induction structurelle en Coq
  - Jean-François DUFOURD, « Design and formal proof of a new optimal image segmentation program with hypermaps ». *Pattern Recognition*, p.2974-2993. Elsevier, 2007.
  - Jean-François DUFOURD, « Polyhedra genus theorem and Euler Formula : A hypermap-formalized intuitionistic proof ». *Theoretical Computer Science*. Elsevier, 2008.

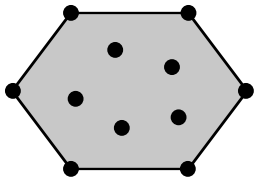
Enveloppe convexe

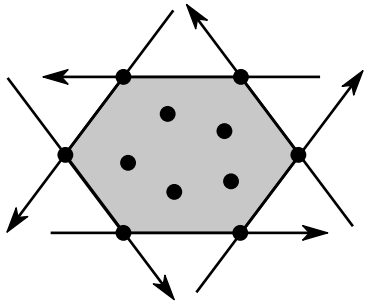
Soit S un ensemble fini de n points du plan euclidien E.

Définition 2.1 (Enveloppe convexe)

L'enveloppe convexe de S est le plus petit polygone convexe T contenant tous les éléments de S et de telle sorte que l'intérieur de T se situe toujours à gauche des droites engendrées par ses arêtes orientées pour un parcours de T dans le sens trigonométrique.

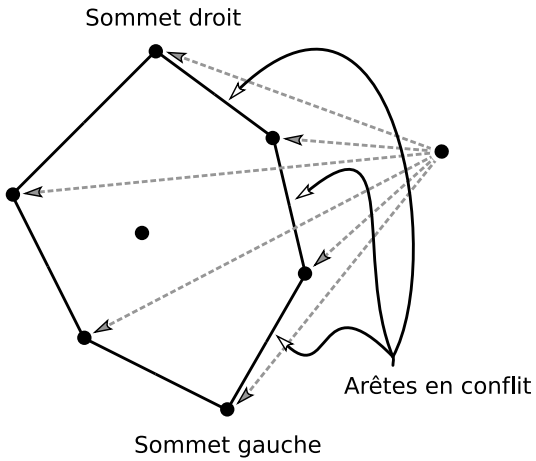


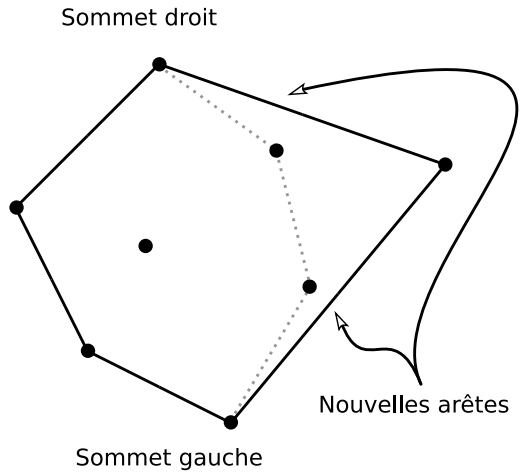




Algorithme incrémental

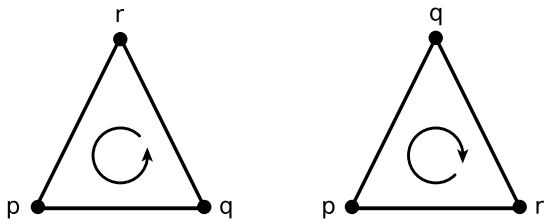
- Construction progressive d'une nouvelle enveloppe convexe par l'insertion des points les uns après les autres
- Concepts de visibilité, arête en conflit, sommets gauche et droit
- A chaque itération : si nouveau point à l'intérieur alors étape suivante sinon suppression des arêtes en conflit et ajout de deux arêtes le reliant au sommet gauche et au sommet droit





Contexte d'étude

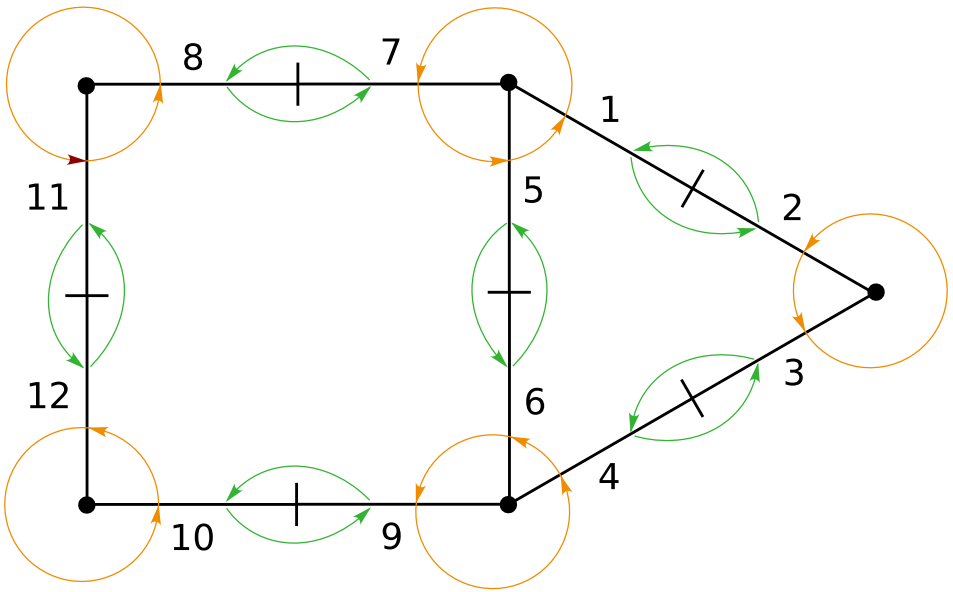
- Prédicat géométrique d'orientation :
  orientation d'un triplet de points dans le plan



- Axiomatisation de Knuth :
  distinction entre prédicats et calculs numériques
  - Résoudre les problèmes de robustesse
  - Isoler les calculs arithmétiques
  - Se focaliser principalement sur les aspects algorithmiques
- Points en position générale :
  tous distincts et jamais 3 points alignés

Présentation

- Structure algébrique très simple pour modéliser les subdivisions du plan (notions de brins et de permutations)
- Séparation des aspects topologiques et géométriques
- Description aisée par induction structurelle



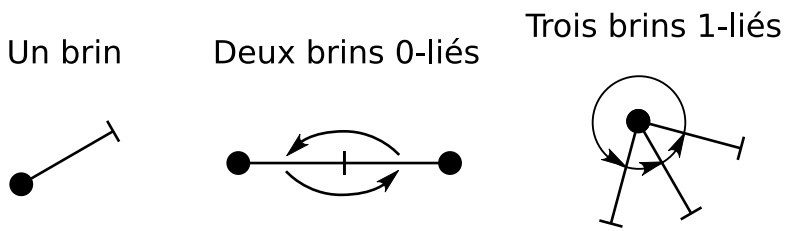
Aspects mathématiques

Définition 3.1 (Hypercarte)

Une hypercarte est une structure algébrique $M = (D, \alpha_0, \alpha_1)$, où D est un ensemble fini dont les éléments sont appelés des brins et α_0, α_1 sont des permutations dans D.

Définition 3.2 (Carte combinatoire)

Une carte combinatoire est une hypercarte où α_0 est une involution dans D, c'est-à-dire $\forall x \in D$, $\alpha_0(\alpha_0(x)) = x$.



Aspects mathématiques

Définition 3.3 (Orbite)

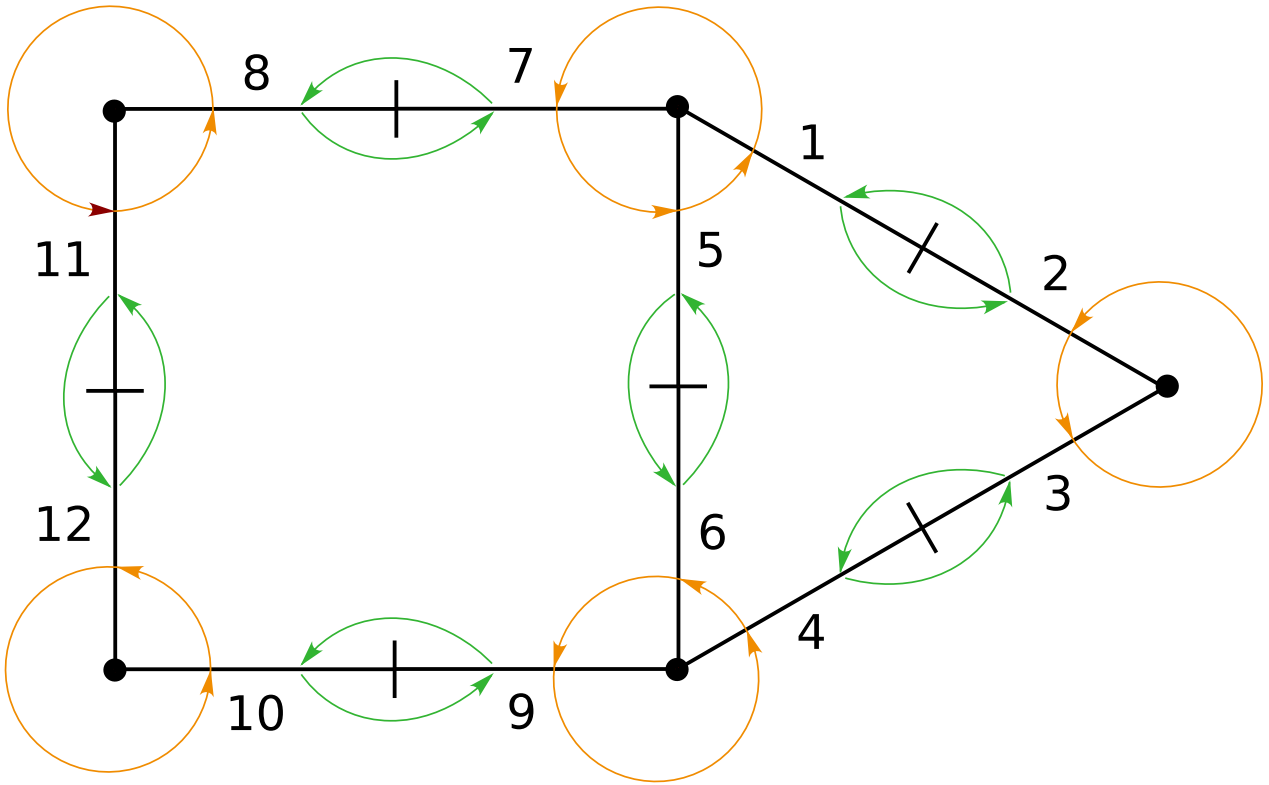
Soient D un ensemble et $f_1, \ldots, f_n$ des fonctions dans D. L'orbite de $x \in D$ pour ces fonctions est le sous-ensemble de D noté $\langle f_1, \ldots, f_n \rangle(x)$ des éléments accessibles depuis x par n'importe quelle composition des fonctions $f_1, \ldots, f_n$.

Définition 3.4 (Cellules topologiques)

Dans l'hypercarte $M = (D, \alpha_0, \alpha_1)$, $\langle \alpha_0 \rangle(x)$ représente l'arête du brin x, $\langle \alpha_1 \rangle(x)$ son sommet, $\langle \alpha_1^{-1} \circ \alpha_0^{-1} \rangle(x)$ sa face, et $\langle \alpha_0, \alpha_1 \rangle(x)$ sa composante connexe.

Aspects mathématiques

arête $\rightsquigarrow \langle \alpha_0 \rangle(x)$; sommet $\rightsquigarrow \langle \alpha_1 \rangle(x)$; face $\rightsquigarrow \langle \alpha_1^{-1} \circ \alpha_0^{-1} \rangle(x)$;
composante connexe $\rightsquigarrow \langle \alpha_0, \alpha_1 \rangle(x)$



Spécifications de base

- Définition des types

```
Definition dart := nat.
Definition nil := 0%nat.

Definition point := (R*R)%type.

Inductive dim : Set := zero : dim | one : dim.
```

- Décidabilité de l'égalité

```
Lemma eq_dim_dec :
  forall (i:dim)(j:dim), {i=j} + {~i=j}.

Definition eq_dart_dec := eq_nat_dec.
```

Spécifications des hypercartes libres

- Définition du type `fmap` de hypercarte libre

```
Inductive fmap : Set :=
    V : fmap
  | I : fmap -> dart -> point -> fmap
  | L : fmap -> dim -> dart -> dart -> fmap.
```

- Définition de prédicats et de fonctions (observateurs)

```
Fixpoint exd (m:fmap)(x:dart) {struct m} : Prop :=
  match m with
    V => False
  | I m0 x0 _ => x = x0 \/ exd m0 x
  | L m0 _ _ _  => exd m0 x
  end.
```

Spécifications des hypercartes

- Définition des préconditions `prec_I` et `prec_L`

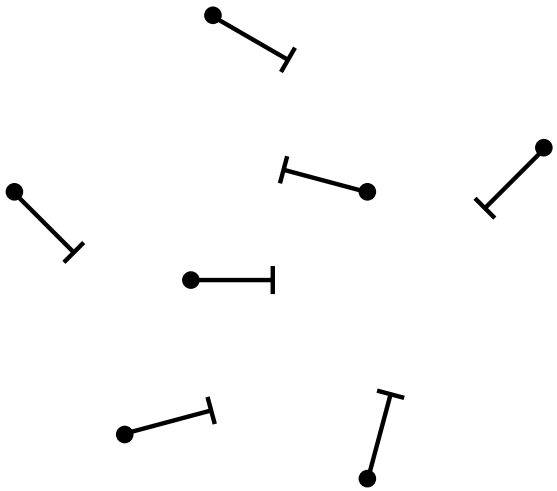
```
Definition prec_I (m:fmap)(x:dart) : Prop :=
  x <> nil /\ ~ exd m x .

Definition prec_L (m:fmap)(k:dim)(x y:dart) :=
  exd m x /\ exd m y /\ ~ succ m k x /\
  ~ pred m k y /\ cA m k x <> y.
```

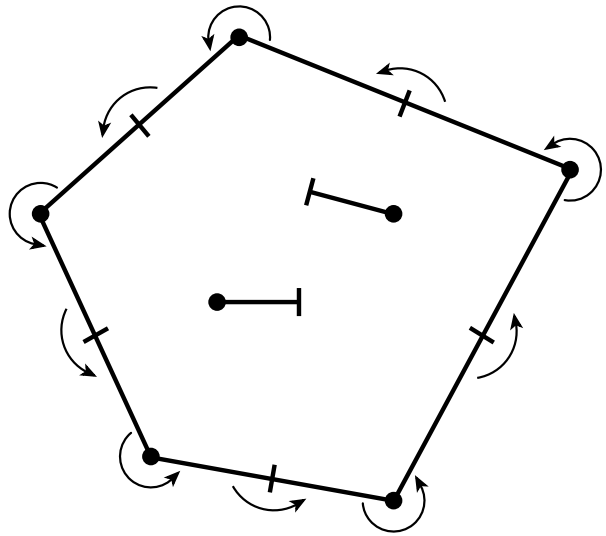
- Définition de l'invariant `inv_hmap`

```
Fixpoint inv_hmap (m:fmap) {struct m} : Prop :=
  match m with
    V => True
  | I m0 x0 _ => inv_hmap m0 /\ prec_I m0 x0
  | L m0 k0 x0 y0 => inv_hmap m0 /\
    prec_L m0 k0 x0 y0  end.
```

Représentation des données

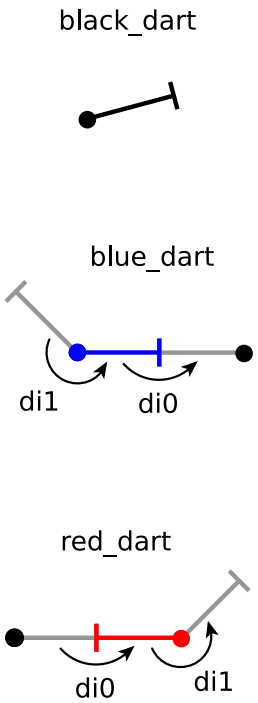
- Représentation sous forme d'hypercarte
  - L'ensemble fini de points du plan :
    un point = un brin isolé sans couture
  - L'enveloppe convexe : polygone topologiquement orienté
    (orbites non closes)

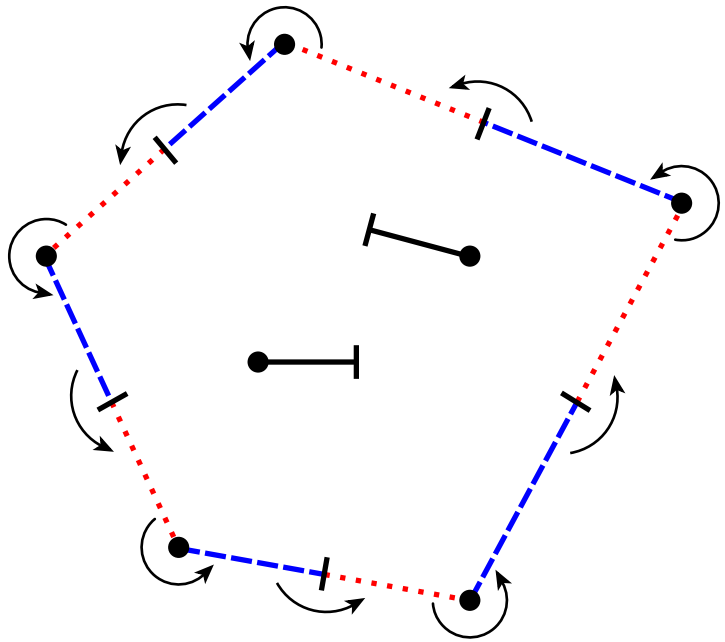




Classification des brins

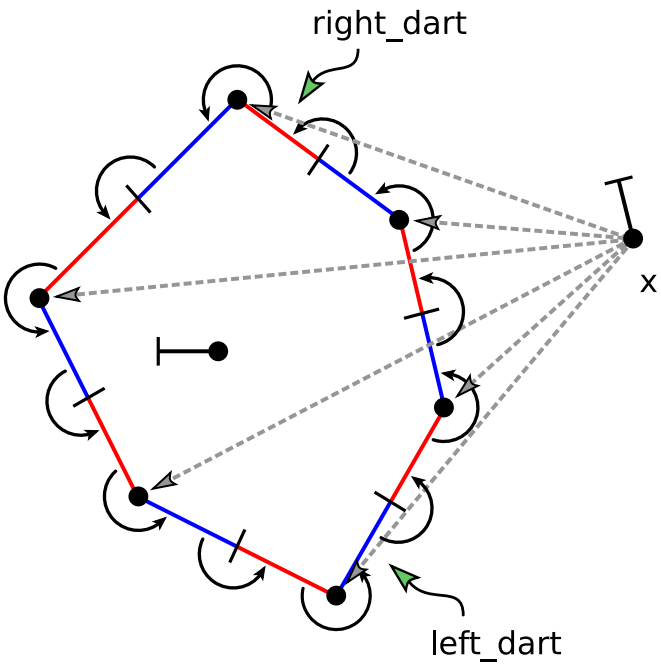
- Classification très précise des brins en 3 catégories :
  - **brins noirs** = brins isolés sans couture
  - brins bleus = brins avec un 1-prédécesseur et un 0-successeur
  - brins rouges = brins avec un 0-prédécesseur et un 1-successeur

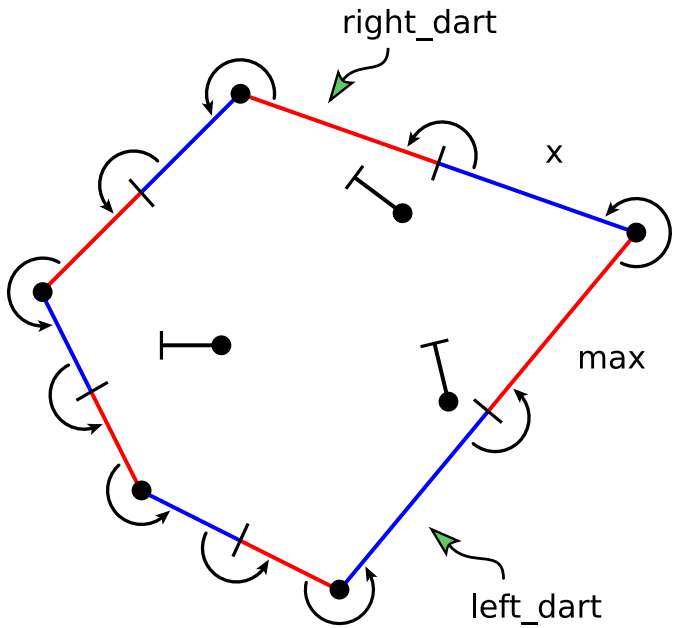




Implémentation

- Définition de la fonction d'insertion





Implémentation

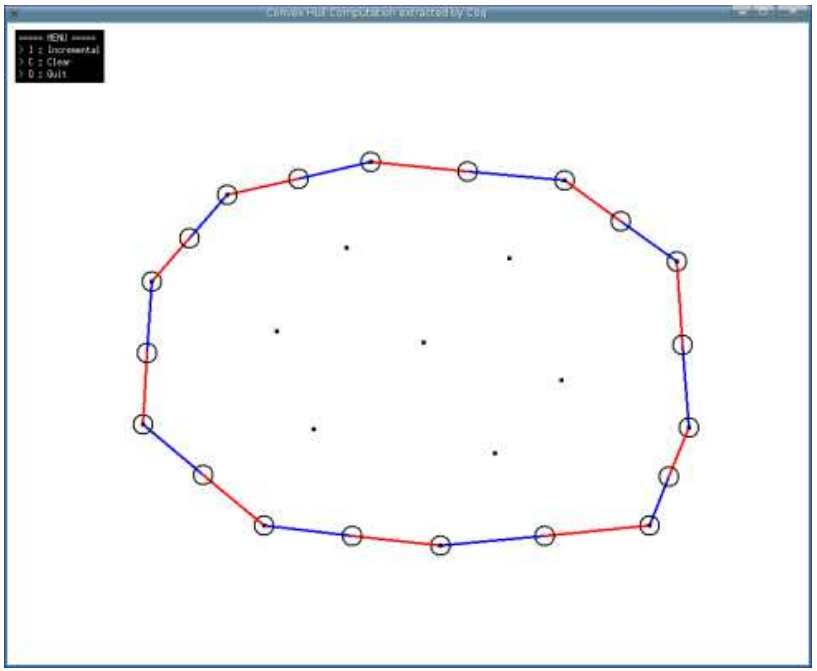
```
Fixpoint CHID (m:fmap)(mr:fmap)(x:dart)(p:point)(max:dart)
  {struct m} : fmap :=
  match m with
    V => I V x p
  | I m0 x0 p0 =>
    if (blue_dart_dec mr x0) then
      if (invisible_dec mr x0 p) then
        (I (CHID m0 mr x p max) x0 p0)
      else if (left_dart_dec mr p x0) then
(L (L (I (I (CHID m0 mr x p max) x0 p0) max p) one max x) zero x0 max)
          else (I (CHID m0 mr x p max) x0 p0)
    else if (red_dart_dec mr x0) then [...]
         else (I (CHID m0 mr x p max) x0 p0)
  | L m0 zero x0 y0 =>
    if (ccw_dec (fpoint mr x0) (fpoint mr y0) p) then
      (L (CHID m0 mr x p max) di0 x0 y0)
    else (CHID m0 mr x p max)
  | L m0 one x0 y0 => [...]
  end.
```

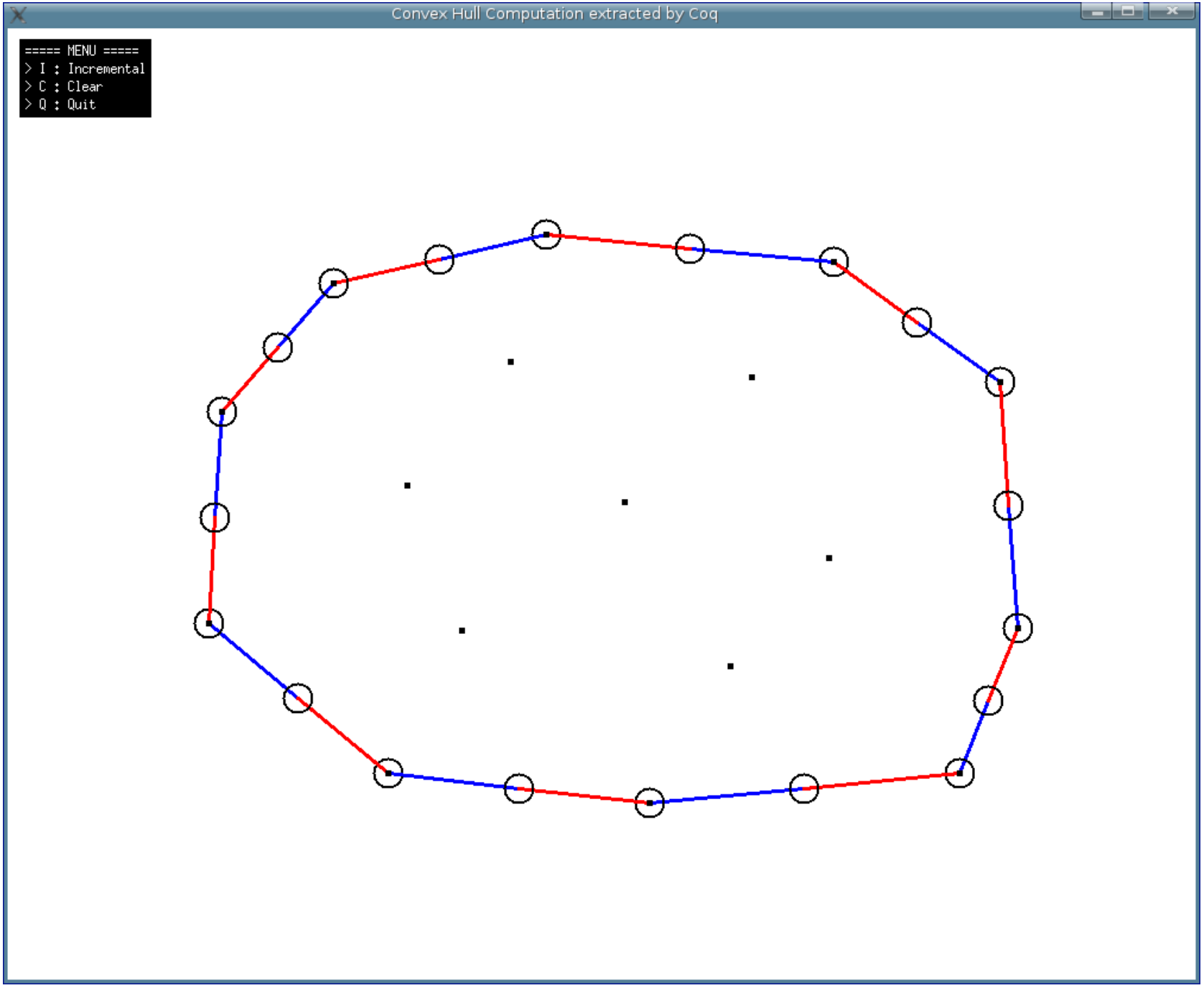
Observations

- Description de l'algorithme par induction structurelle
  - Traitement indéterministe et séparé des brins et des coutures
  - Pas de stratégie de parcours dans l'hypercarte mais analyse complète de l'hypercarte
  - Nécessite une hypercarte de référence pour les tests et une notion de sous-hypercarte
- Hypercarte orientée avec des orbites non closes
  - Moins de liaisons à casser
  - Orientation immédiate

Extraction d'un programme et prototypage

- Mécanisme d'extraction
  - Engendrer automatiquement des programmes en OCaml
- Interface graphique
  - Saisie des données (les points du plan)
  - Visualisation du résultat (l'enveloppe convexe)




Convex Hull Computation extracted by Coq
===== MENU =====
> I : Incremental
> C : Clear
> Q : Quit

Preuve formelle de la correction totale

- Correction totale = terminaison + correction partielle
  - Terminaison : imposée et garantie par Coq
  - Correction partielle
    - Propriétés topologiques
    - Propriétés géométriques
- Preuve par induction structurelle sur les hypercartes
- Préconditions sur l'hypercarte initiale

```
Definition prec_CH (m:fmap) : Prop :=
  inv_hmap m /\ linkless m /\
  well_emb m /\ noalign m.
```

Propriétés topologiques

- Conservation des brins

```
Lemma inv_exd : forall (m:fmap), prec_CH m ->
  forall (d:dart), exd m d -> exd (CH m) d.
```

- Conservation de l'invariant des hypercartes

```
Lemma inv_hmap_CH : forall (m:fmap),
  prec_CH m -> inv_hmap (CH m).
```

- Obtention d'un « polygone topologique »

```
Lemma inv_poly_CH : forall (m:fmap),
  prec_CH m -> inv_poly (CH m).
```

- Une seule composante connexe après suppression des brins intérieurs isolés (relation d'Euler, théorème du genre, planarité)

Propriétés géométriques

- Bon plongement des brins par rapport à leurs coutures

```
Lemma well_emb_CH : forall (m:fmap),
  prec_CH m -> well_emb (CH m).
```

- Caractérisation de la convexité

```
Definition convex (m:fmap) : Prop :=
  forall (x:dart), exd m x -> blue_dart m x ->
  forall (z:dart), exd m z ->
  (fpoint m z) <> (fpoint m x) ->
  (fpoint m z) <> (fpoint m (A m di0 x)) ->
  ccw (fpt m x) (fpt m (A m di0 x)) (fpt m z).

Lemma convex_CH : forall (m:fmap),
  prec_CH m -> convex (CH m).
```

Propriétés des extrémités gauche et droite

- Unicité des extrémités gauche et droite

```
Theorem one_left :
  forall (m:fmap)(p:point)(x:dart)(y:dart),
  inv_hmap m -> inv_poly m ->
  well_emb m -> convex m ->
  left_dart m p x -> left_dart m p y -> x = y.
```

- Equivalence de l'existence des extrémités gauche et droite

```
Theorem exd_left_dart_exd_right_dart :
  forall (m:fmap)(p:point), inv_hmap m -> inv_poly m ->
  (exists da:dart, exd m da /\ left_dart m p da) ->
  (exists db:dart, exd m db /\ right_dart m p db).
```

- Interdépendance forte entre topologie et géométrie

Bilan

- Description formelle d'un algorithme incrémental de calcul d'enveloppe convexe d'un ensemble fini de points du plan
  - basé sur une modélisation à base topologique
  - programmé par induction structurelle dans le langage fonctionnel de Coq
- Extraction d'un prototype de construction exécutable en OCaml
- Certification formelle : terminaison et démonstrations de nombreuses propriétés topologiques et géométriques
- Taille du développement :
  - Définitions : 50
  - Lemmes / théorèmes : 300
  - Lignes de code : 20 000

Perspectives

- Dérivation de programmes efficaces en langage impératif ou orienté objet (C, C++...) et insertion dans la plate-forme de modélisation géométrique de notre équipe
- Étude d'autres algorithmes plus efficaces de calcul d'enveloppe convexe (parcours de Graham ou marche de Jarvis)
- Dimensions supérieures (3D et +)
- Subdivisions et algorithmes plus complexes (triangulation de Delaunay et diagramme de Voronoi)

Fin

Merci de votre attention