Calculating Convex Hulls with a Linear Solver

With thanks to Mats Carlsson, Bart Demoen, Raimund Seidel and Axel Simon

Florence Benoy†, Andy King† and Fred Mesnard‡

†Computing Laboratory,
University of Kent,
UK

‡Iremia,
Université de La Réunion,
France

Calculating convex hulls

- The problem of calculating the convex hull of a set of points $\{x_1, \ldots, x_n\}$ is almost trivial
- Consider, however, the problem of calculating the smallest polyhedron P that includes $P_1 \cup P_2$ where P_1 and P_2 are represented in standard form:

$$P_1 = \{\vec{x} \in \mathbb{R}^n \mid A_1\vec{x} \leq \vec{B}_1\} \qquad P_2 = \{\vec{x} \in \mathbb{R}^n \mid A_2\vec{x} \leq \vec{B}_2\}$$

Recall that $P \subseteq \mathbb{R}^n$ is polyhedral iff it can be represented as the intersection of a finite number of half-spaces, that is, as $P = \{\vec{x} \in \mathbb{R}^n \mid A\vec{x} \leq \vec{B}\}$.

Convex hull of $P_1 \cup P_2$ is not closed

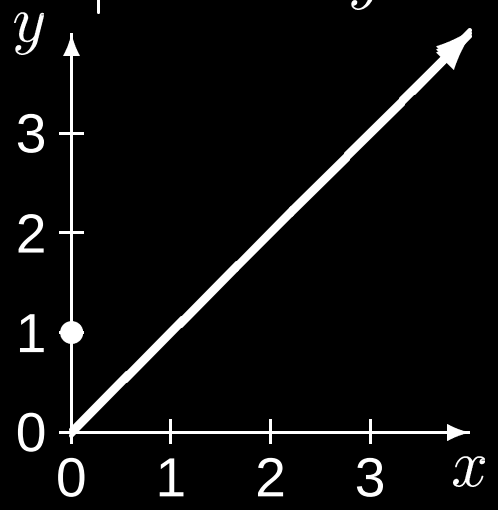
Consider P_1 and P_2 where

$$P_1 = \left\{ \vec{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \vec{x} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \right\}$$

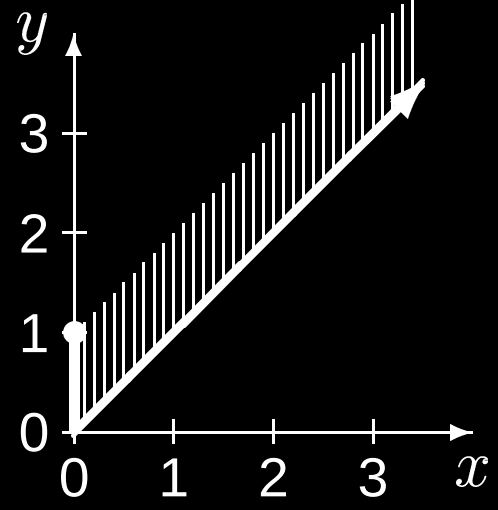
$$P_2 = \left\{ \vec{x} \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ -1 & 0 \end{bmatrix} \vec{x} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Convex hull of $P_1 \cup P_2$ is not closed

The convex hull of $P_1 \cup P_2$ excludes the points $\{\langle x, y\rangle \in \mathbb{R}^2 \mid x > 0 \wedge y = x + 1\}$ and hence is not closed.



P_1 and P_2



convex hull of $P_1 \cup P_2$

Since the convex hull of $P_1 \cup P_2$ is not necessarily closed, the convex hull cannot always be represented by a system of non-strict linear inequalities. Thus apply closure.

Classic approach to computing the closure of the hull

- The classic approach to computing the closure of the convex hull is to employ two representations: (i) points and rays and (ii) systems of non-strict inequalities;
- The Chernikova algorithm [Le Verge, 1992] can convert between (i) and (ii) in both directions;
- Thus convert from (ii) to (i), then calculate closure of the convex hull, then convert from (i) to (ii).

Computing the closure of the hull

The convex hull of $P_1 \cup P_2$ is given by:

$$P_H = \left\{ \vec{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \vec{x} = \sigma_1 \vec{x}_1 + \sigma_2 \vec{x}_2 \wedge \sigma_1 + \sigma_2 = 1 \wedge 0 \leq \sigma_1 \wedge \\ A_1 \vec{x}_1 \leq \vec{B}_1 \qquad\quad \wedge A_2 \vec{x}_2 \leq \vec{B}_2 \;\; \wedge 0 \leq \sigma_2 \end{array} \right\}$$

This system can be relaxed by putting $\vec{y}_1 = \sigma_1 \vec{x}_1$ and $\vec{y}_2 = \sigma_2 \vec{x}_2$ so that $\vec{x} = \vec{y}_1 + \vec{y}_2$ and $A_i \vec{y}_i \leq \sigma_i \vec{B}_i$ to define:

$$P_{CH} = \left\{ \vec{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \vec{x} = \vec{y}_1 + \vec{y}_2 \;\; \wedge \sigma_1 + \sigma_2 = 1 \;\; \wedge 0 \leq \sigma_1 \wedge \\ A_1 \vec{y}_1 \leq \sigma_1 \vec{B}_1 \wedge A_2 \vec{y}_2 \leq \sigma_2 \vec{B}_2 \wedge 0 \leq \sigma_2 \end{array} \right\}$$

Note that P_{CH} is polyhedral since it is the projection of a system of non-strict linear inequalities.

P_{CH} is the closure of the hull of $P_1 \cup P_2$

To see this, consider the recession cone of a polyhedron P, that is, $0^+P = \{\vec{y} \in \mathbb{R}^n \mid \forall \lambda \geq 0 \,.\, \forall \vec{x} \in P \,.\, \vec{x} + \lambda \vec{y} \in P\}$.

Theorem 19.6 of Rockafellar states that the closure of the convex hull of $P_1 \cup P_2$ is the set

$$(0^+P_1 + P_2) \cup (P_1 + 0^+P_2) \cup$$
$$(\cup\{\sigma_1 P_1 + \sigma_2 P_2 \mid \sigma_1 + \sigma_2 = 1 \wedge 0 < \sigma_1, \sigma_2\})$$

Intuitively, $0^+P_1 + P_2$ is P_2 extended in the directions of half-lines contained within P_1.

P_{CH} is the closure of the hull of $P_1 \cup P_2$

Let $\vec{x} \in P_i$. Now $\vec{y} \in 0^+P_i$ iff $A_i(\vec{x} + \lambda\vec{y}) \leq \vec{B}_i$ for all $\lambda \geq 0$ which holds iff $A_i\vec{y} \leq \vec{0}$.

Therefore

$0^+P_1 + P_2 = \{\vec{x} \in \mathbb{R}^n \,|\, \vec{x} = \vec{y}_1 + \vec{y}_2 \wedge A_1\vec{y}_1 \leq \vec{0} \wedge A_2\vec{y}_2 \leq \vec{B}_2\}$.

Observe that

$\{\vec{x} \in \mathbb{R}^n \mid \vec{x} = \vec{y}_1 + \vec{y}_2 \wedge A_1\vec{y}_1 \leq \sigma_1\vec{B}_1 \wedge A_2\vec{y}_2 \leq \sigma_2\vec{B}_2\}$

coincides with the sets (i) $0^+P_1 + P_2$, (ii) $P_1 + 0^+P_2$ and (iii) $\cup\{\sigma_1P_1 + \sigma_2P_2 \mid \sigma_1 + \sigma_2 = 1 \wedge 0 < \sigma_1, \sigma_2\}$ when (i) $\sigma_1 = 0$ and $\sigma_2 = 1$, (ii) $\sigma_1 = 1$ and $\sigma_2 = 0$ and (iii) $\sigma_1 + \sigma_2 = 1$ and $0 < \sigma_1, \sigma_2$ respectively.

Therefore P_{CH} is the closure of the convex hull.

Towards a linear solver-based implementation

This result leads to an algorithm for computing the closure of the convex hull:

- construct the systems $A_i\vec{y}_i \leq \sigma_i\vec{B}_i$ by scaling the constant vectors $\vec{B}_i$ by σ_i,
- add the constraints $\vec{x} = \vec{y}_1 + \vec{y}_2$, $\sigma_1 + \sigma_2 = 1$ and $0 \leq \sigma_i$,
- eliminate variables other than $\vec{x}$ using projection to obtain P_{CH} in terms of $\vec{x}$.

Hence the closure of the convex hull can be computed without recourse to another representation.

Non-ground representation

- Closed polyhedra will be represented by lists (conjunctions) of linear constraints.
- A linear constraint c and an expression e takes the form

$$c ::= e \leq e | e = e | e \geq e$$

$$e ::= x | n | n * x | -e | e + e | e - e$$

  where n is a rational number and x is a variable.
- Adding constraints to a system then amounts to list concatenation.

Projection

We need to construct a predicate `project(+Xs,+Cxs,-ProjectCxs)` that is true when for a given list of dimensions `Xs` and a given list of constraints `Cxs`, `ProjectCxs` is the projection of `Cxs` onto `Xs`.

Fortunately, the CLP($\mathbb{R}$) library provides a projection facility, `dump`, that can be used as in the query `{Y > X}, dump([X, Y], NewVars, Answer)` that binds `NewVars` to `[A, B]` and `Answer` to `[A < B]`.

Projection code

```
project(Xs, Cxs, ProjectCxs) :-
  tell_cs(Cxs),
  dump(Xs, Vs, ProjectCxs), Xs = Vs.

tell_cs([]).
tell_cs([C|Cs]) :- {C}, tell_cs(Cs).
```

For example, the query `project([X, Z], [X < Y, Y < Z], ProjectCs)` will bind `Cs` to `[X-Z<0]`.

However, the compound query `{X = Z + 1}, project([X, Z], [X < Y, Y < Z], ProjectCs)` will fail because of store interaction.

Projection code

To insulate the constraints posted in `tell_cs`, both the variables `Xs` and the constraints `Cxs` need to be renamed.

```
project(Xs, Cxs, ProjectCxs) :-
  copy_term(Xs-Cxs, CpyXs-CpyCxs),
  tell_cs(CpyCxs),
  dump(CpyXs, Vs, ProjectCxs), Xs = Vs.
```

Note that care must be taken to ensure that `Xs` and `Cxs` are renamed consistently.

Code for closure of hull code

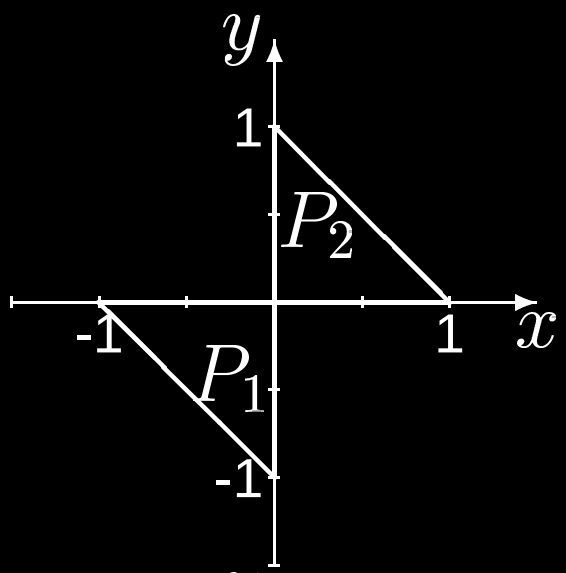
```
convex_hull(Xs, Cxs, Ys, Cys, Zs, Czs) :-
  scale(Cxs, Sig1, [], C1s),
  scale(Cys, Sig2, C1s, C2s),
  add_vect(Xs, Ys, Zs, C2s, C3s),
  project(Zs, [Sig1 >= 0, Sig2 >= 0, Sig1+Sig2 = 1|C3s],

add_vect([], [], [], Cs, Cs).
add_vect([U|Us], [V|Vs], [W|Ws], C1s, C2s) :-
    add_vect(Us, Vs, Ws, [W = U+V|C1s], C2s).
```

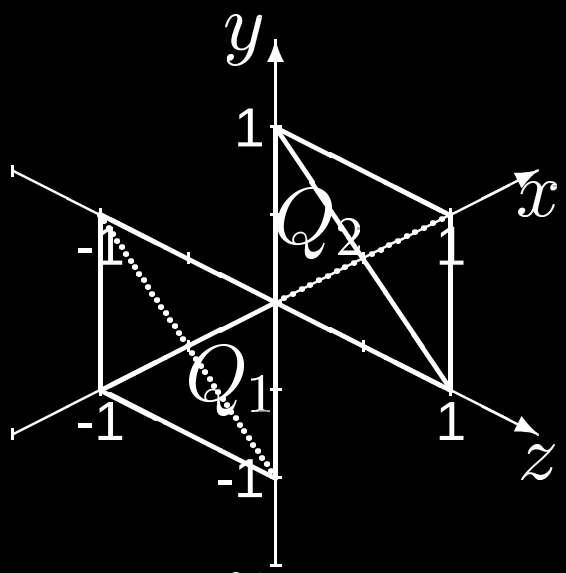
Running this code on the original example yields the query `convex_hull([X1,Y1],[X1=0,Y1=1],[X2,Y2], [X2>=0,Y2=X2],[A,B],S)` which gives the answer `S = [A>=0,A-B>=-1,A-B=<0]`.

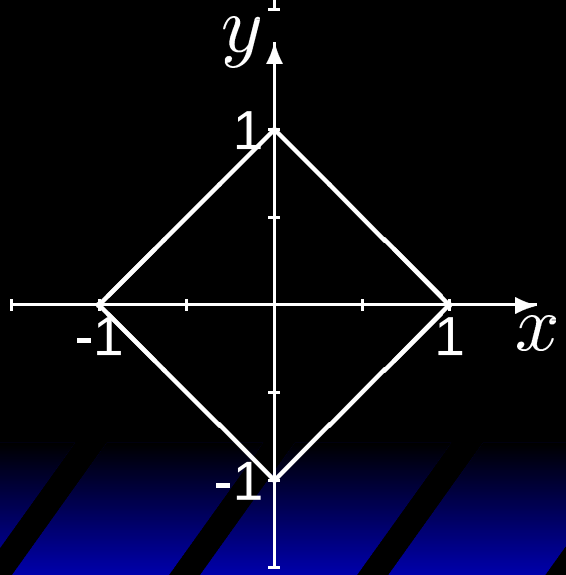
Complexity of closure of the hull

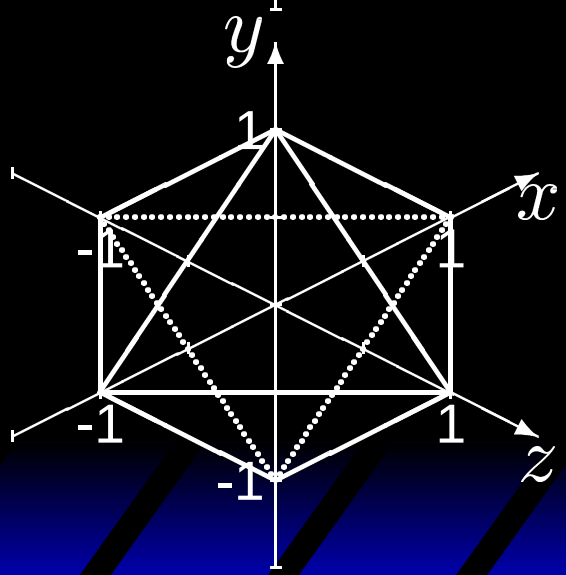
- The algorithm relies on project – which is implemented Fourier-Motzkin style – which is known to be exponential;
- It was recently shown [Chandru, 2000] that linear projection is fundamentally exponential;
- The problem of converting between points and rays and systems of non-strict inequalities is also intrinsically exponential.

Complexity of closure of the hull









Conclusions

- Despite the scaling problems, the technique has been widely applied in logic programming, mostly to satisfaction.
- In the context of termination inference, this method is feasible since it accounts for 42% of the first pass of the analysis and the first pass itself accounts for only 23% of the total analysis time.
- In principle the tactic is very simple since it exploits built in machinery; in fact it will appear as a pearl in TPLP early in 2005.