Extending a Verified SMT Solver for Mixed-Integer Linear Arithmetic

Alban Reynaud

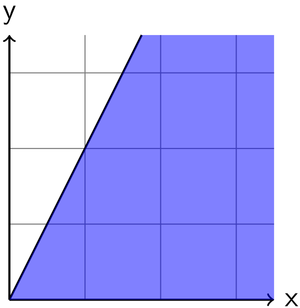
4 septembre 2019

Arithmétique Linéaire

$$\left\{\begin{array}{rcrl} 2x & - & y & > 0 \\ & & 2y & \geqslant 1 \\ 2x & + & 2y & \leqslant 7 \end{array}\right.$$

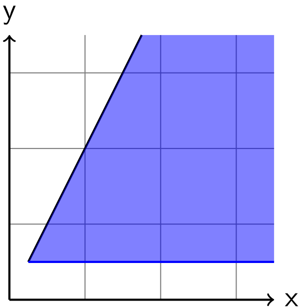
Arithmétique Linéaire

$$\begin{cases} 2x - y > 0 \\ \end{cases}$$



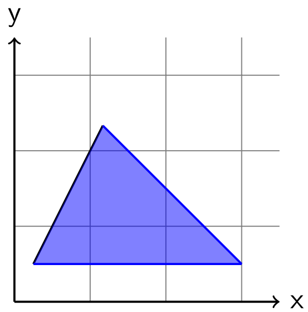
Arithmétique Linéaire

$$\begin{cases} 2x - y > 0 \\ 2y \geqslant 1 \end{cases}$$



Arithmétique Linéaire

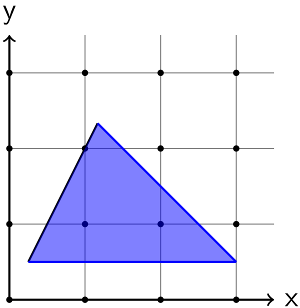
$$\begin{cases} 2x - y > 0 \\ 2y \geqslant 1 \\ 2x + 2y \leqslant 7 \end{cases}$$



Arithmétique Linéaire

$$\begin{cases} 2x - y > 0 \\ 2y \geqslant 1 \\ 2x + 2y \leqslant 7 \end{cases}$$

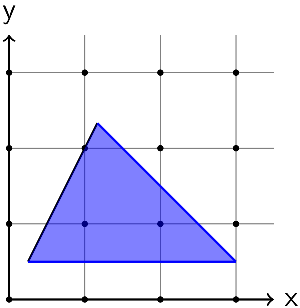
$$x, y \in \mathbb{Z}$$



Arithmétique Linéaire

$$\begin{cases} 2x - y > 0 \\ 2y \geqslant 1 \\ 2x + 2y \leqslant 7 \end{cases}$$

$$x \in \mathbb{Z}$$



Définitions

- Trouver une affectation des variables satisfiant une conjonction de contraintes linéaires : *Problème Linéaire*.

Définitions

- Trouver une affectation des variables satisfiant une conjonction de contraintes linéaires : *Problème Linéaire*.
- Trouver une affectation des variables vers des entiers satisfiant une conjoction de contraintes linéaires : *Integer Linear Problem (ILP)*.

Définitions

- Trouver une affectation des variables satisfiant une conjonction de contraintes linéaires : *Problème Linéaire*.
- Trouver une affectation des variables vers des entiers satisfiant une conjoction de contraintes linéaires : *Integer Linear Problem (ILP)*.
- Trouver une affectation des variables dont certaines doivent être entières satisfiant conjoction de contraintes linéaires : *Mixed-Integer Linear Problem (MILP)*.

Isabelle

Quelques caractéristiques de cet assistant :

- Automatisation des preuves
- Utilisation de symboles mathématiques

Travaux Précédents

- Dutertre et de Moura (2006) : algorithme incrémental pour résoudre les problèmes linéaires [2]
- Thiemann, Bottesch et Haslbeck (2018) : formalisation de cet algorithme avec Isabelle [1].

L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

- Chercher une solution réelle v. S'il n'en existe pas, terminer.

L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

- Chercher une solution réelle v. S'il n'en existe pas, terminer.
- Si pour tout $i \in I$, $v(x_i) \in \mathbb{Z}$ renvoyer v

L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

- Chercher une solution réelle v. S'il n'en existe pas, terminer.
- Si pour tout $i \in I$, $v(x_i) \in \mathbb{Z}$ renvoyer v
- S'il existe $i \in$ tel que $v(x_i) \notin \mathbb{Z}$:

L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

- Chercher une solution réelle v. S'il n'en existe pas, terminer.
- Si pour tout $i \in I$, $v(x_i) \in \mathbb{Z}$ renvoyer v
- S'il existe $i \in$ tel que $v(x_i) \notin \mathbb{Z}$:
  - Essayer de résoudre récursivement le problème avec l'ensemble de contraintes $S \cup \{x_i \leqslant \lfloor v(x_i) \rfloor\}$

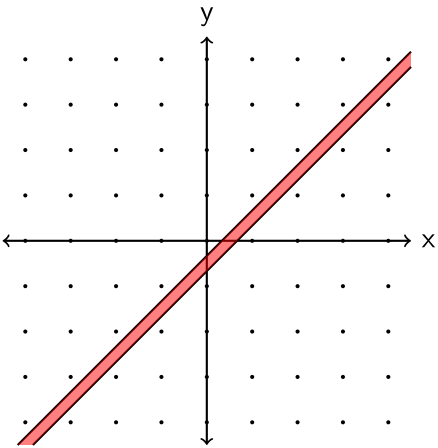
L'algorithme Branch-and-Bound

Variables : $x_0, ..., x_{n-1}$
Ensemble des variables entières : I
Ensemble des contraintes : S

- Chercher une solution réelle v. S'il n'en existe pas, terminer.
- Si pour tout $i \in I$, $v(x_i) \in \mathbb{Z}$ renvoyer v
- S'il existe $i \in$ tel que $v(x_i) \notin \mathbb{Z}$:
  - Essayer de résoudre récursivement le problème avec l'ensemble de contraintes $S \cup \{x_i \leqslant \lfloor v(x_i) \rfloor\}$
  - Essayer de résoudre récursivement le problème avec l'ensemble de contraintes $S \cup \{x_i \geqslant \lceil v(x_i) \rceil\}$

Terminaison de l'algorithme

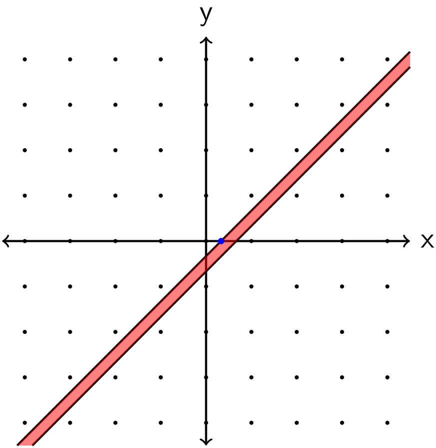
$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \end{array}$$



Terminaison de l'algorithme

$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \end{array}$$

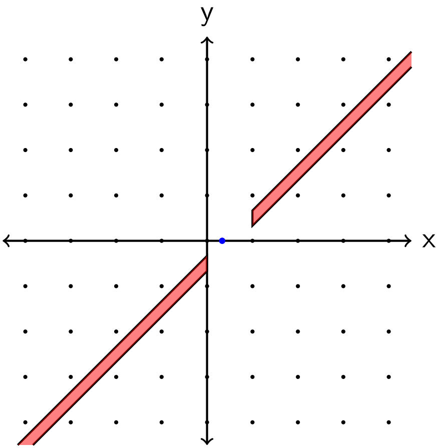
$$(x, y) = \left(\frac{1}{3}, 0\right)$$



Terminaison de l'algorithme

$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \\ \color{red}{x} & & & \color{red}{\geqslant} & \color{red}{1} \end{array}$$

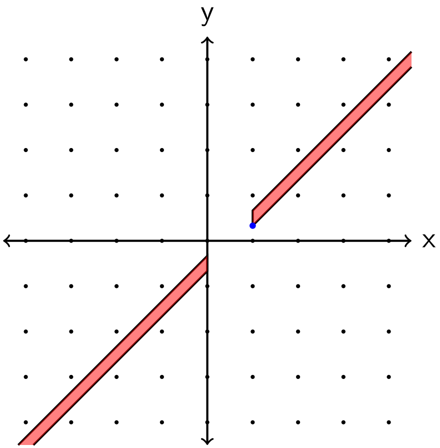
$$(x, y) = \left(\frac{1}{3}, 0\right)$$



Terminaison de l'algorithme

$$
\begin{array}{rcrcl}
3x & - & 3y & \geqslant & 1 \\
3x & - & 3y & \leqslant & 2 \\
x & & & \geqslant & 1
\end{array}
$$

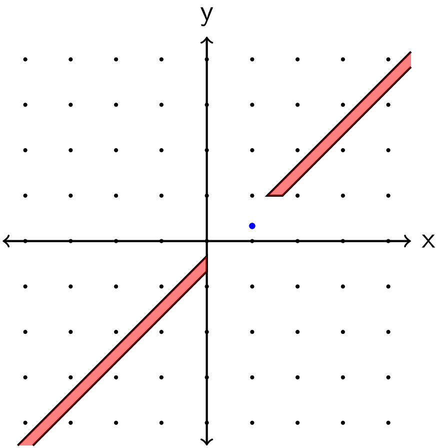
$$(x, y) = \left(1, \frac{1}{3}\right)$$



Terminaison de l'algorithme

$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \\ \color{red}{x} & & & \color{red}{\geqslant} & \color{red}{1} \\ & & \color{red}{y} & \color{red}{\geqslant} & \color{red}{1} \end{array}$$

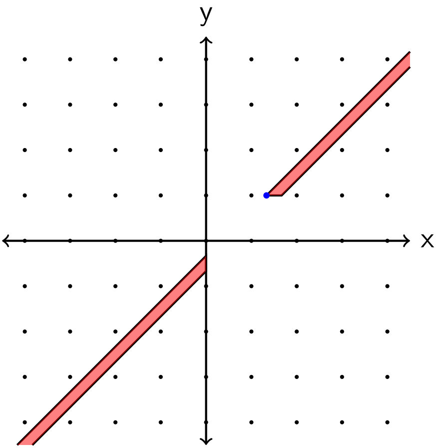
$$(x, y) = \left(1, \frac{1}{3}\right)$$



Terminaison de l'algorithme

$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \\ \color{red}{x} & & & \color{red}{\geqslant} & \color{red}{2} \\ & & \color{red}{y} & \color{red}{\geqslant} & \color{red}{1} \end{array}$$

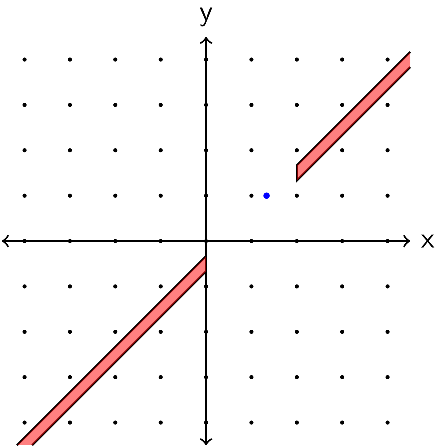
$$(x, y) = \left(1 + \frac{1}{3}, 1\right)$$



Terminaison de l'algorithme

$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \\ x & & & \geqslant & 2 \\ & & y & \geqslant & 1 \end{array}$$

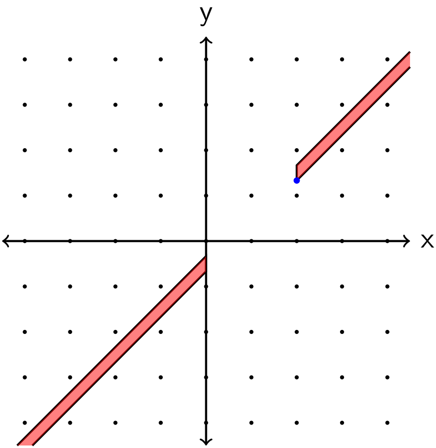
$$(x, y) = \left(1 + \frac{1}{3}, 1\right)$$



Terminaison de l'algorithme

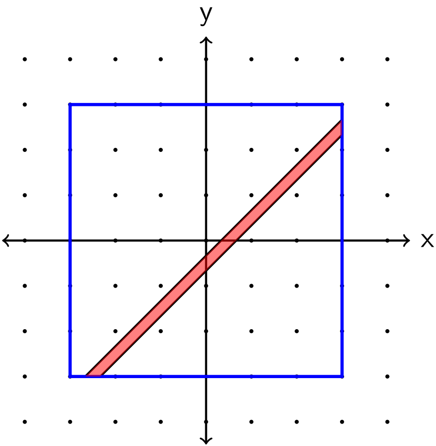
$$\begin{array}{rcrcl} 3x & - & 3y & \geqslant & 1 \\ 3x & - & 3y & \leqslant & 2 \\ x & & & \geqslant & 2 \\ & & y & \geqslant & 1 \end{array}$$

$$(x, y) = \left(2, 1 + \frac{1}{3}\right)$$



Terminaison de l'algorithme

$$
\begin{array}{ccccc}
3x & - & 3y & \geqslant & 1 \\
3x & - & 3y & \leqslant & 2 \\
-B & \leqslant & x & \leqslant & B \\
-B & \leqslant & y & \leqslant & B
\end{array}
$$



Terminaison de l'algorithme

But

Calculer une borne B telle que le problème admet une solution entière si et seulement il admet une solution entière telle que toute variable soit affectée à un nombre dans l'intervalle $[-B, B]$.

Transformation du problèmes

- Affectation $(x_i)_{0 \leqslant i < n} \longrightarrow$ vecteur $x \in \mathbb{R}^n$
- Contraintes linéaires $\{\sum a_{ij} x_j \leqslant b_i\} \longrightarrow Ax \leqslant b$
- Contraintes linéaires $\{\sum a_{ij} x_j < b_i\} \longrightarrow Ax < b$

Définitions

Definition (Polyhèdre)

Ensemble des vecteurs x vérifiant $Ax \leqslant b$

Définitions

Definition (Polyhèdre)

Ensemble des vecteurs x vérifiant $Ax \leqslant b$

Definition (Cône fini)

$$\mathrm{cone}\ \{c_0, \ldots, c_{s-1}\} = \left\{ \sum \lambda_i c_i \mid \lambda_i \geqslant 0 \right\}$$

Définitions

Definition (Polyhèdre)

Ensemble des vecteurs x vérifiant $Ax \leqslant b$

Definition (Cône fini)

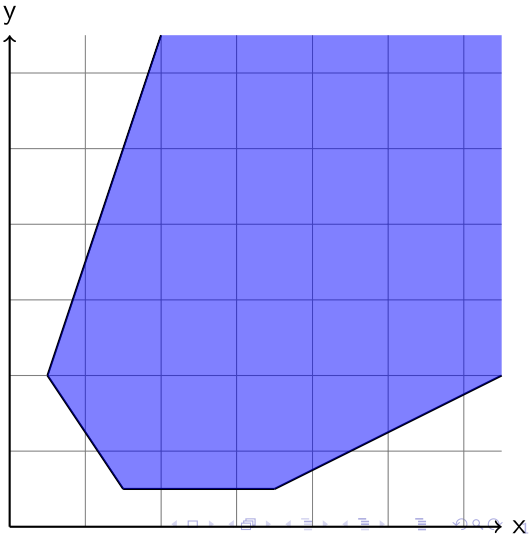
$$\mathrm{cone}\ \{c_0, \ldots, c_{s-1}\} = \left\{ \sum \lambda_i c_i \mid \lambda_i \geqslant 0 \right\}$$

Definition (Polytope)

Enveloppe convexe d'un ensemble fini de points.

Théorème de décomposition

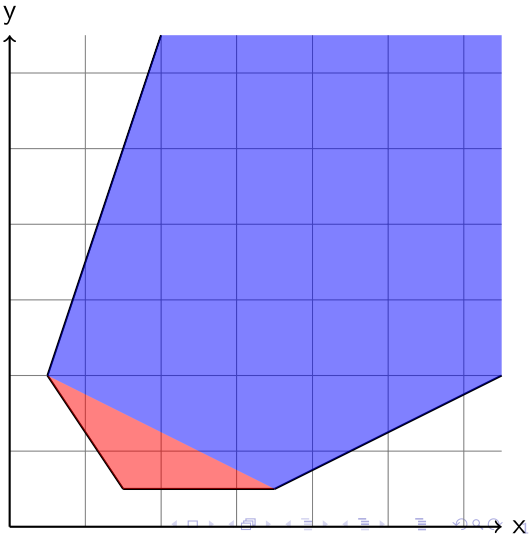
Tout polyhèdre peut-être décomposé en la somme :



Théorème de décomposition

Tout polyhèdre peut-être décomposé en la somme :

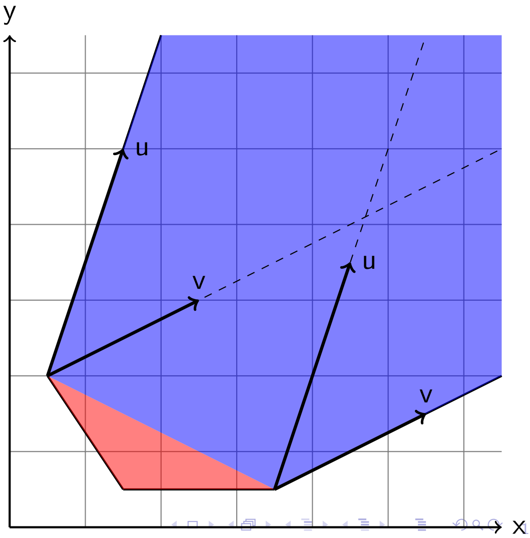
- d'un polytope



Théorème de décomposition

Tout polyhèdre peut-être décomposé en la somme :

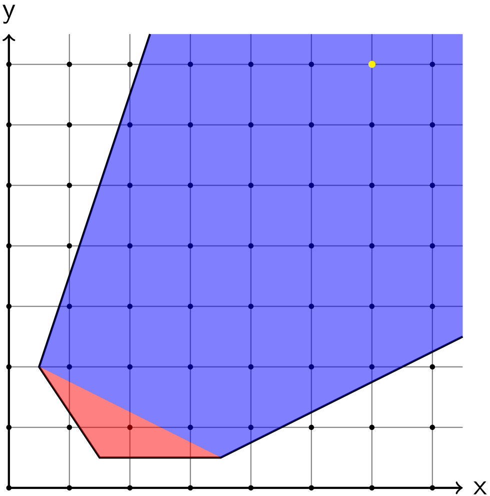
- d'un polytope
- et d'un cône fini



$$P = Q + \mathrm{cone}\ \{c_0, \ldots, c_{s-1}\}$$

$$c_0, \ldots, c_{s-1} \in \mathbb{Z}^n$$

$$x \in P$$

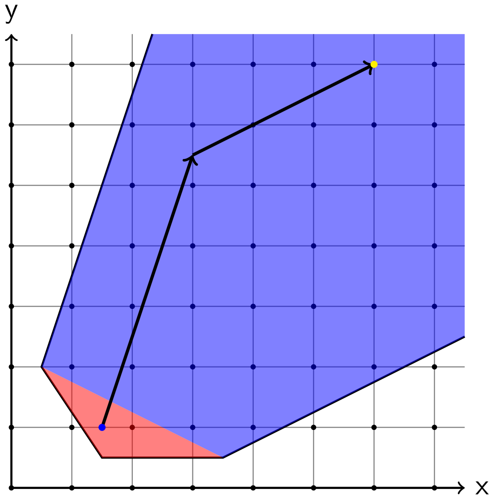


$$P = Q + \mathrm{cone}\,\{c_0, \ldots, c_{s-1}\}$$

$$c_0, \ldots, c_{s-1} \in \mathbb{Z}^n$$

$$x \in P$$

$$x = q + \sum \lambda_i c_i$$

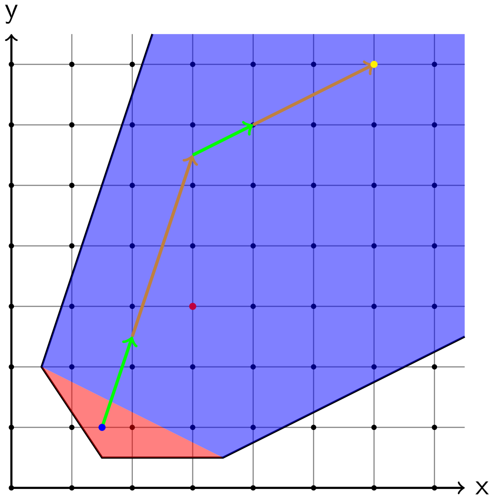


$$P = Q + \mathrm{cone}\ \{c_0, \ldots, c_{s-1}\}$$

$$c_0, \ldots, c_{s-1} \in \mathbb{Z}^n$$

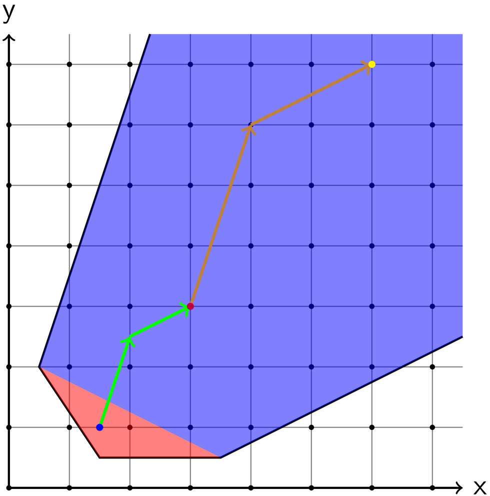
$$x \in P$$

$$\begin{aligned} x = q \quad &+ \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i \\ &+ \sum \lfloor \lambda_i \rfloor c_i \end{aligned}$$



$$P = Q + \mathrm{cone}\,\{c_0, \ldots, c_{s-1}\}$$

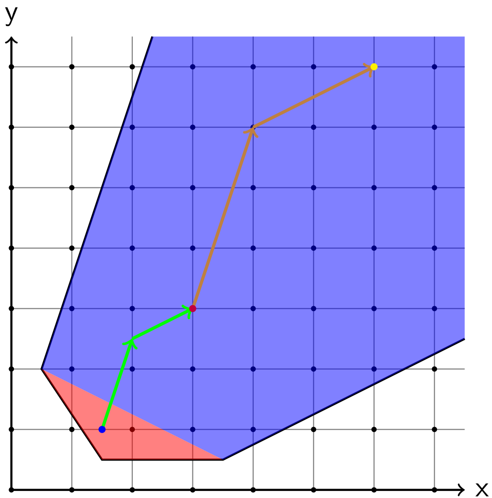
$$y = q + \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i$$



$$P = Q + \mathrm{cone}\ \{c_0, \dots, c_{s-1}\}$$

$$y = q + \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i$$

$$y \in \mathbb{Z}^n$$

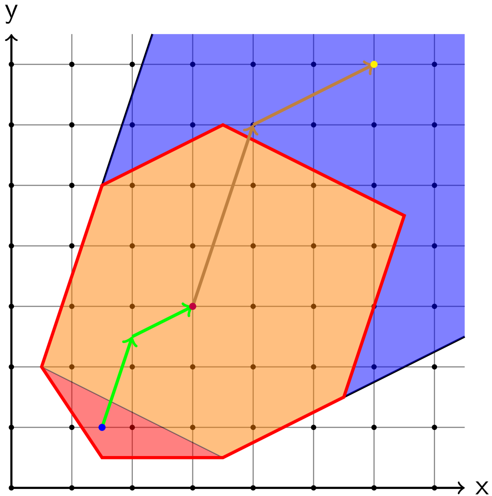


$$P = Q + \text{cone}\ \{c_0, \ldots, c_{s-1}\}$$

$$y = q + \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i$$

$$y \in \mathbb{Z}^n$$

$$y \in \{q + \sum \lambda_i c_i \mid \quad q \in Q, \\ 0 \leqslant \lambda_i \leqslant 1\}$$

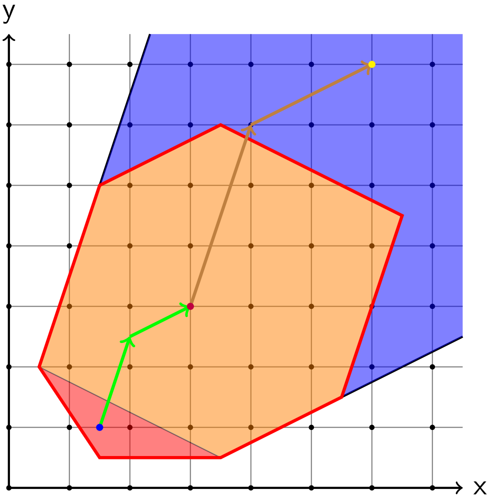


$$P = Q + \mathrm{cone}\,\{c_0, \ldots, c_{s-1}\}$$

$$y = q + \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i$$

$$y \in \mathbb{Z}^n$$

$$y \in D$$

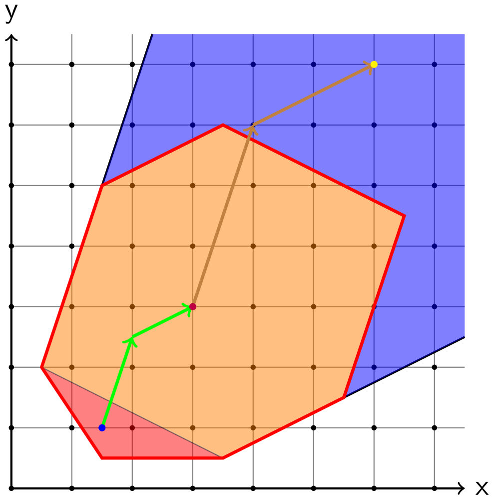


$$P = Q + \mathrm{cone}\ \{c_0, \ldots, c_{s-1}\}$$

$$y = q + \sum (\lambda_i - \lfloor \lambda_i \rfloor) c_i$$

$$y \in \mathbb{Z}^n$$

$y \in D,\ D$ est borné



Existence d'une solution entière

- P possède un point entier si et seulement si D possède un point entier.

Existence d'une solution entière

- P possède un point entier si et seulement si D possède un point entier.
- On peut borner les éléments de D en fonction des éléments du polytope Q et des c_i.

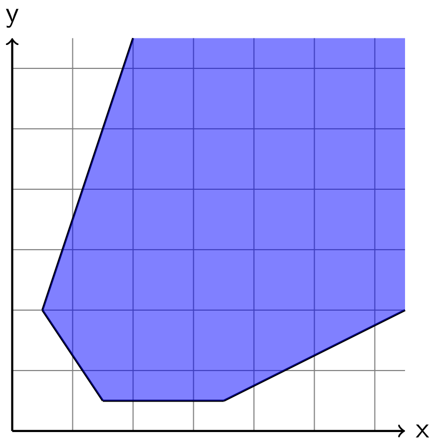
Existence d'une solution entière

- P possède un point entier si et seulement si D possède un point entier.
- On peut borner les éléments de D en fonction des éléments du polytope Q et des c_i.

Posons m le plus grand coefficient du problème linéaire.

Théorème de décomposition généralisé

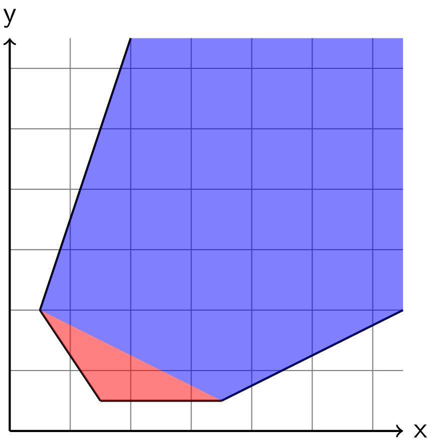
Tout polyhèdre peut-être décomposé en la somme :



Théorème de décomposition généralisé

Tout polyhèdre peut-être décomposé en la somme :

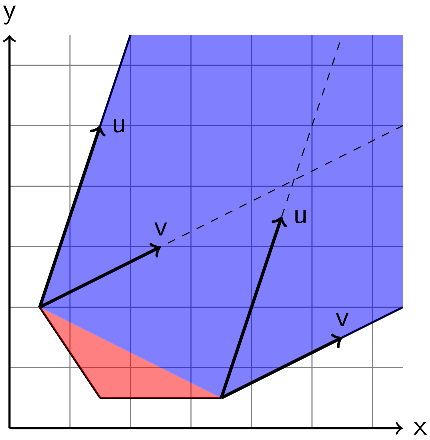
- d'un polytope dans $[-B, B]^n$



Théorème de décomposition généralisé

Tout polyhèdre peut-être décomposé en la somme :

- d'un polytope dans $[-B, B]^n$
- et d'un cône fini de base entière dans $[-B, B]^n$

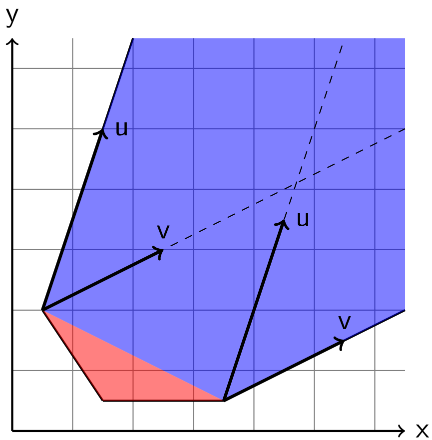


Théorème de décomposition généralisé

Tout polyhèdre peut-être décomposé en la somme :

- d'un polytope dans $[-B, B]^n$
- et d'un cône fini de base entière dans $[-B, B]^n$

$B = n! \cdot m^n$



Théorème

Il existe une solution entière à un problème linéaire si et seulement il existe une solution entière bornée par $(n+1)! \cdot m^n$.

Formalisation des résultats concernant les inégalités linéaires

Formalisation de nombreux résultats issus de *Theory of linear and integer programming* [3].

Formalisation des résultats concernant les inégalités linéaires

Formalisation de nombreux résultats issus de *Theory of linear and integer programming* [3].

- Théorème fondamental des inégalités linéaires
- Théorème de Farkas-Minkowsky-Weyl
- Théorème de décomposition
- Théorème de Meyer
- ...

Formalisation des résultats concernant les inégalités linéaires

Formalisation de nombreux résultats issus de *Theory of linear and integer programming* [3].

- Réalisé en collaboration avec René Thiemann et Max Haslbeck.

Formalisation des résultats concernant les inégalités linéaires

Formalisation de nombreux résultats issus de *Theory of linear and integer programming* [3].

- Réalisé en collaboration avec René Thiemann et Max Haslbeck.
- Publié dans *Archive of Formal Proofs* (AFP)

Formalisation de l'algorithme branch-and-bound

Preuve de la correction et de la terminaison d'un algorithme branch-and-bound.

- Utilisation de solveur de problèmes linéaires précédent
- Utilisation des résultats sur les inégalités linéaires

Conclusion

- Formalisation de résultats concernant les inégalités linéaires

Conclusion

- Formalisation de résultats concernant les inégalités linéaires
- Notamment, à propos de bornes sur l'existence d'une solution

Conclusion

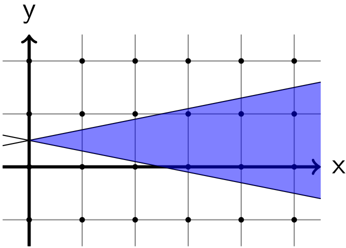
- Formalisation de résultats concernant les inégalités linéaires
- Notamment, à propos de bornes sur l'existence d'une solution
- Preuve de la terminaison et de la correction d'un algorithme pour résoudre les MILP.

Travaux futurs

- Utilisation de l'interface incrémentale

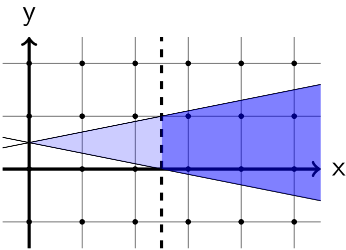
Travaux futurs

- Utilisation de l'interface incrémentale
- Utilisation de coupes de Gomory



Travaux futurs

- Utilisation de l'interface incrémentale
- Utilisation de coupes de Gomory



Travaux futurs

- Utilisation de l'interface incrémentale
- Utilisation de coupes de Gomory
- Développement d'une interface incrémentale

Ralph Bottesch, Max W. Haslbeck, and René Thiemann.
Verifying an incremental theory solver for linear arithmetic in Isabelle/HOL.
In *Proceedings of the 12th International Symposium on Frontiers of Combining Systems*, volume 11715 of *LNAI*, 2019.
To appear.

Bruno Dutertre and Leonardo de Moura.
A fast linear-arithmetic solver for DPLL(T).
In *Proceedings of the 18th International Conference on Computer Aided Verification*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.

Alexander Schrijver.
Theory of linear and integer programming.
John Wiley & Sons, 1998.