APRON - Activités à Vérimag

- **Réduction de la dimension de l'espace en analyse de relations linéaires**
- **Elargissement et Accélération**
- **Un domaine pour la vérification des systèmes sur puce**

Réduction de la dimension de l'espace

[Thèse de David Merchat, en cours de rédaction]

Les représentations des polyèdres peuvent être exponentielles en la dimension de l'espace.

Exemple : n-hypercube, $2n$ contraintes, 2^n sommets

Deux approches :

- **découverte précoce des équations linéaires**
- **factorisation cartésienne**

Equations linéaires

- **idée de base : effectuer d'abord une analyse d'équations linéaires [Karr76]. Utiliser les équations pour éliminer des variables.**
- **utilisation au cours de l'analyse : avant tout calcul coûteux sur les polyèdres, calculer les équations satisfaites dans le résultat, et éliminer les variables**

Résultats décevants : en fait, ce travail est fait implicitement dans les opérations de toute bonne bibliothèque

Factorisation cartésienne

Si un polyèdre P est un produit cartésien de polyèdres

$$P = P_1 \times P_2 \times \ldots \times P_\ell$$

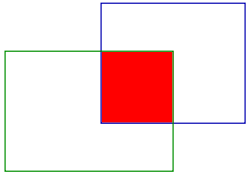
(variables indépendantes), son système générateur (sommets, rayons, droites) peut être réduit logarithmiquement

exemple: hypercube $\longrightarrow$ intervalles

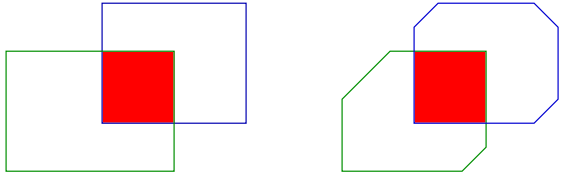
Détection de produit cartésien: **Facile sur le système de contraintes (mise sous forme bloc-diagonale)**

Opérations sur les produits cartésiens : **Facile pour presque toutes les opérations ($\cap$, transformations affines, élargissement, test du vide, inclusion)**

Exemple: **l'intersection de 2 polyèdres factorisés de la même façon est au moins aussi factorisée :**



Exemple: **l'intersection de 2 polyèdres factorisés de la même façon est au moins aussi factorisée :**



L'enveloppe convexe :

$$(P_X \times P_Y) \sqcup (Q_X \times Q_Y)$$

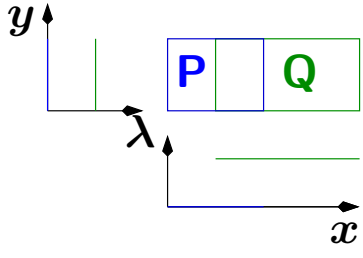
soient $R_{X,\lambda} = (P_X \wedge \lambda = 0) \sqcup (Q_X \wedge \lambda = 1)$,
et $S_{Y,\lambda} = (P_Y \wedge \lambda = 0) \sqcup (Q_Y \wedge \lambda = 1)$

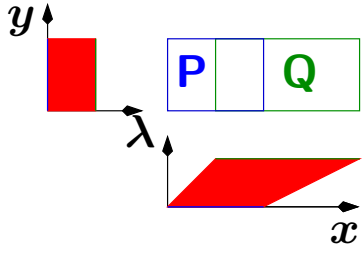
- **Si λ est borné (par des expressions non constantes)**
  - **supérieurement dans $R_{X,\lambda}$,**
  - **et inférieurement dans $S_{Y,\lambda}$,**

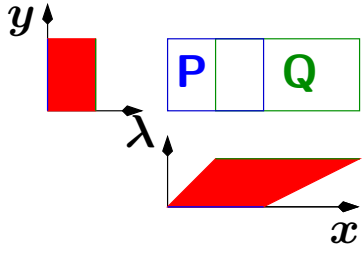
  ou inversement,
  alors X et Y ne sont pas indépendantes dans
  $(P_X \times P_Y) \sqcup (Q_X \times Q_Y) = \exists\lambda.R_{X,\lambda} \wedge S_{Y,\lambda}$

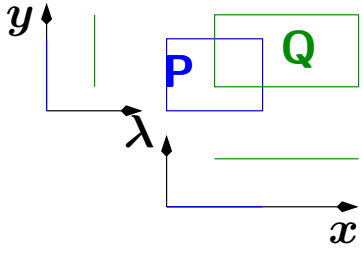
- **Sinon**
  $(P_X \times P_Y) \sqcup (Q_X \times Q_Y) = (\exists\lambda.R_{X,\lambda}) \times (\exists\lambda.S_{Y,\lambda})$

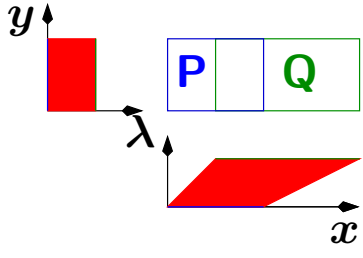

P
Q

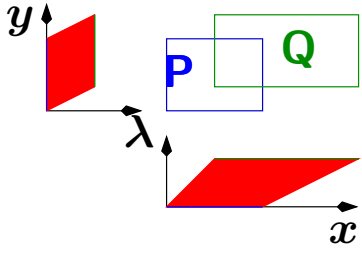

y
P
Q
λ
x

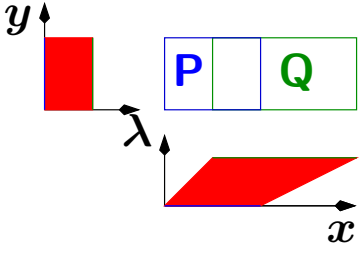

y
P
Q
λ
x

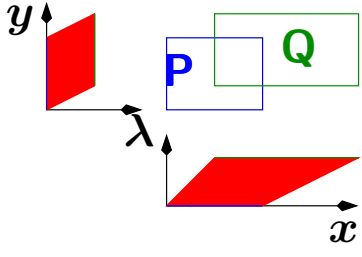

y
P
Q
λ
x

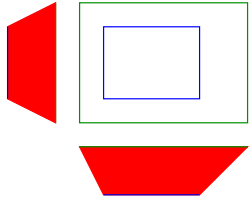

y
P
Q
λ
x

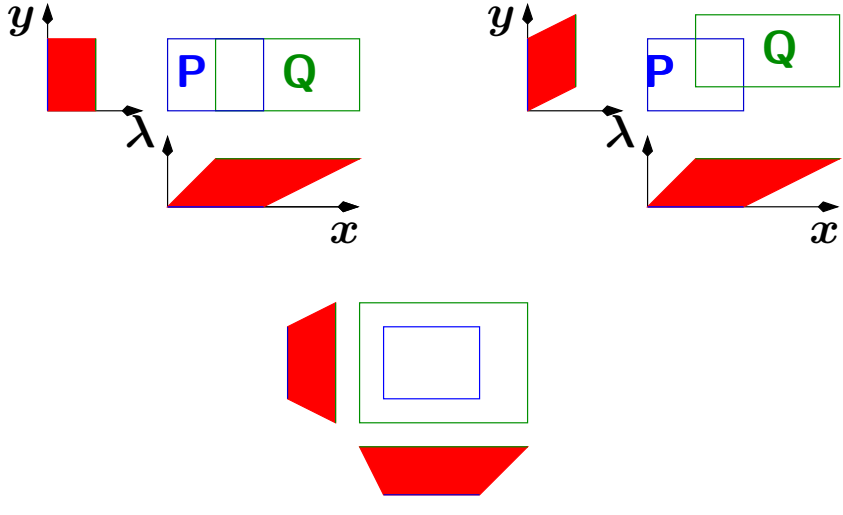

y
P
Q
λ
x


y
P
Q
λ
x


y
P
Q
λ
x

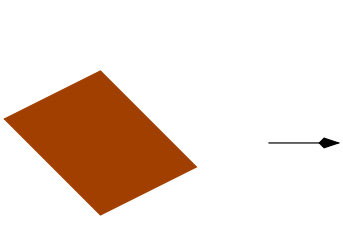

y
P
Q
λ
x





Très efficace, on gagne même quand le résultat n'est pas factorisé

Changement de base



$$2 \leq x + y \leq 5$$
$$-4 \leq x - 2y \leq 2$$

$$x' = x + y - 2$$
$$y' = x - y + 4$$
$$0 \leq x' \leq 3$$
$$0 \leq y' \leq 6$$

- **implementé au dessus de la PPL (bibliothèque de Parme)**
- **compatible avec les matrices creuses?**

Elargissement et accélération

[Thèse de Laure Gonnord, en cours]

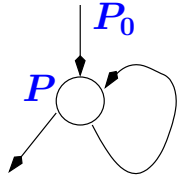
Problème :

Calculer $F^*(X_0) = X_0 \sqcup F(X_0) \sqcup F^2(X_0) \sqcup \ldots$

Elargissement: De X_0**,** $X_0 \sqcup F(X_0)$**, extrapoler** $F^*(X_0)$

Accélération [Boigelot, Finkel, ...]: Dans certains cas, on sait calculer exactement F^*

Accélération sur les polyèdres

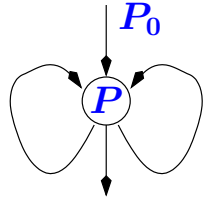


$g \rightarrow X := X + B$

$$F^*(P_0) = \exists X_0 \in P_0 \cap g,\ \exists k \in \mathbb{N},$$
$$X = X_0 + kB \wedge g(X + (k-1)B)$$

Plus dur...

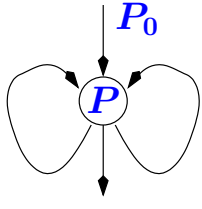
$g_2 \to X := X + B_2$



$g_1 \to X := X + B_1$

On ne sait pas calculer $(F_1 \sqcup F_2)^*$

Plus dur...

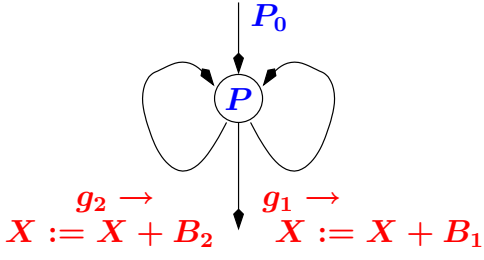
$g_2 \rightarrow X := X + B_2$  $g_1 \rightarrow X := X + B_1$

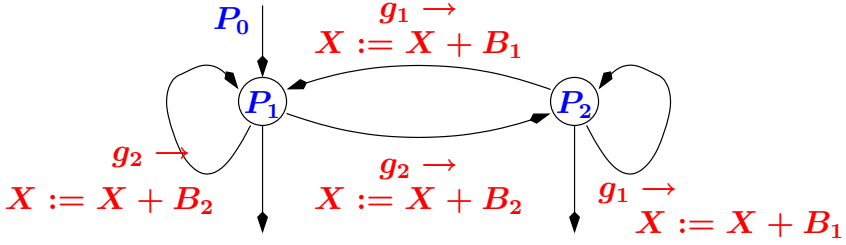
On ne sait pas calculer $(F_1 \sqcup F_2)^*$

mais on peut poser:

$$P = P \nabla \left(F_1^*(P) \sqcup F_2^*(P)\right)$$

Autre piste: partitionnement





Vérification des systèmes sur puce

DEA de Mathias Péron, démarrage

Systèmes sur puces, en System-C (thèses avec ST)

Problèmes de vérification avec des entiers qui représentent des adresses (mémoire, composants,. . .)

Polyèdres trop puissants, intervalles trop faibles

Vérification des systèmes sur puce

DEA de Mathias Péron, démarrage

Systèmes sur puces, en System-C (thèses avec ST)

Problèmes de vérification avec des entiers qui représentent des adresses (mémoire, composants,. . .)

Polyèdres trop puissants, intervalles trop faibles

intervalles $+ \{=, \neq\}$
$(+ \{<, \leq, \geq, >\}? +$ **"uninterpreted functions"?)**