



Gilles Chabert <gilchab@gmail.com>

probleme avec la certification

2 messages

Bertrand Neveu <Bertrand.Neveu@sophia.inria.fr>**10 septembre 2008 11:56**

À: Gilles Chabert <gilchab@gmail.com>

Cc: Gilles Trombettoni <Gilles.Trombettoni@sophia.inria.fr>

Salut,

Voici encore un nouveau problème avec Ibex.

La certification marche assez mal, même pour des cas très simples.

Sur le problème suivant (1 equation du 2 second degré)

Variables

 $p1_x \text{ in } [-5,5];$

Constraints

 $(p1_x)^2 + 2*p1_x = 0 ;$ Ibex certifie une solution ($x=-2$) et pas ($x=0$), alors que la dérivée n'est pas nulle en 0.

Par contre, si la contrainte est factorisée,

 $p1_x * (p1_x + 2) = 0$, les 2 solutions sont certifiées.

ci joint la trace et les fichiers solver2.cc, exB1 et exB2.

A +,

Bertrand.

[neveu@boccanegra ibex-1.05]\$ solver2 exB1 $(p1_x^2 + (2*p1_x))=0$

contractor 0: 5 node(s) contractor 1: 1 node(s) contractor 2: 1 node(s) 5 boxes created.

total time : 0s

solution certifiée 0 (-1.9999999999999999[3,15])

solution non certifiée 0 (-0.0000000000000000[0,1])

[neveu@boccanegra ibex-1.05]\$ solver2 exB2 $(p1_x*(p1_x+2))=0$

contractor 0: 3 node(s) contractor 1: 2 node(s) contractor 2: 0 node(s) 3 boxes created.

total time : 0s

solution certifiée 0 ([0,0])

solution certifiée 1 ([-2,-2])


```
Variables
p1_x in [-5,5];
Constraints
(p1_x)^2 + 2*p1_x = 0 ;
```


end

```
Variables
p1_x in [-5,5];
Constraints
p1_x *(p1_x + 2) = 0 ;
```

end

3 pièces jointes

 **exB1**
1K

 **exB2**
1K

 **solver2.cc**
2K

Gilles Chabert <gilchab@gmail.com>

17 septembre 2009 15:36

À: Bertrand Neveu <Bertrand.Neveu@sophia.inria.fr>

Cc: Gilles Trombettoni <Gilles.Trombettoni@sophia.inria.fr>

Salut,

J'ai enfin clos ce bug!

Le problème était le suivant:

Mettons qu'on avait une boîte B à certifiée, obtenue par un solver.

Le problème était qu'un filtrage f (comme la 2B) pouvait m'avoir "ramené" une solution aux bornes de la boîte B (typiquement, en écrivant $x=0$, les bornes de $[x]$ deviennent *exactement* 0). Or le Newton, du fait que ce soit une itération "numérique" ne fait qu'approcher (certes très vite) une solution à epsilon.

Du coup, en gonflant la boîte à certifier, B qui devenait B', le résultat de Newton appliqué à B' (appelons-le B'') n'était jamais inclus B.

Ajoutons qu'appliquer le Newton à B directement n'était pas possible, car il faut observer une contraction "stricte" pour que cela prouve l'existence d'une solution.

Il était possible alors d'imaginer de faire quelque chose comme "je n'ai pas pu certifier B mais en revanche, j'ai certifié une boîte B'' " mais j'ai toujours trouvé ça très sale pour pas mal de raisons que vous imaginez.

En fait, il fallait simplement introduire 2 contracteurs (voir les paramètres de la classe Certification)

- le "prouver" (Newton) qui n'est utilisé qu'une fois pour observer une stricte contraction de la boîte gonflée B' (en B'')
- l' "entonnoir" (donc, la 2B par exemple) qui sert à "faire rentrer" B'' dans B en la filtrant.

J'aime bien le mot "entonnoir" .. un nouveau concept en CP ? :-)

J'ai joint au mail: le patch (.h et .cpp) à placer dans ibex/src et un exemple de code source (à partir de l'exemple soumis par Bertrand).

A+
Gilles

Le 10 septembre 2008 11:56, Bertrand Neveu <Bertrand.Neveu@sophia.inria.fr> a écrit :

[Texte des messages précédents masqué]

end

Variables

p1_x in [-5,5];

Constraints

p1_x *(p1_x + 2) = 0 ;

end

3 pièces jointes

 **lbexPaver.h**
12K

 **lbexPaver.cpp**
9K

 **bug018.cc**
1K
