

TD Feuille 5 — Analyse statique d'intervalles

Le but de cet exercice est de définir une analyse statique vérifiant l'absence de débordements arithmétiques dans les entier signés 32 bits. Dans l'exercice, toutes les variables sont implicitement de type `int` et donc, par invariant de typage, comprise entre $M_- = -2^{31}$ et $M_+ = 2^{31} - 1$. On considère ici un domaine abstrait non-relationnel \mathbb{D} dérivé d'un domaine de valeurs abstraites \mathbb{V} , où chaque valeur abstraite est soit l'intervalle vide noté \perp , soit un couple de $\mathbb{Z} \times \mathbb{Z}$ noté $[a, b]$, avec $M_- \leq a \leq b \leq M_+$, qui représente l'ensemble des entiers v tels que $a \leq v \leq b$.

Pour simplifier les notations, on considère que $[a, b]$ avec $a > b$ se "simplifie" en \perp .

Exercice 1 (Mise en place du treillis borné de \mathbb{V}).

1. Définir \sqcup, \sqcap et \sqsubseteq .
2. A quoi correspond \top ? Le treillis est-il de hauteur finie ?

Exercice 2 (Opérateurs arithmétiques dans \mathbb{V}). Pour calculer la valeur abstraite des termes (dans un certain état abstrait), on pense à définir les opérations suivantes sur les valeurs abstraites :

$$[a_1, b_1] + [a_2, b_2] \quad -[a, b] \quad [a_1, b_1] * [a_2, b_2] \quad [a, b]/2$$

Ici $a/2$ représente la division entière par 2 de C . On rappelle qu'elle vérifie $(-a)/2 = -(a/2)$.

Pour simplifier, on va autoriser ces opérations à produire des intervalles qui débordent de $[M_-, M_+]$. L'idée est que l'analyse d'une affectation $x := T$ sera de toute façon précédée d'un **assert** générant une obligation de preuve que l'interprétation $T[A]$ de T dans l'état abstrait courant A satisfait $T[A] \sqsubseteq [M_-, M_+]$. Et *in fine*, on définira

$$A[x := T] \stackrel{def}{=} A \oplus \{x \mapsto T[A] \sqcap [M_-, M_+]\}$$

► **Question 1.** Définir les opérateurs ci-dessus. Comment se comporte \perp vis-à-vis de ces opérateurs ?

► **Question 2.** On suppose deux variables x et y dans un état abstrait tel que $A(x) = [10, 100]$ et $A(y) = [-10, 5]$. Que vaut $A[x := ((x + y) * (-y))/2]$?

► **Question 3.** Que vaut $A[x := 42]$?

Exercice 3 (Interprétation abstraite des gardes). Définir les gardes suivantes dans un contexte où x_1 et x_2 sont variables entières, et c est un littéral entier.

1. $A \sqcap (x_1 < c)$
2. $A \sqcap (x_1 < x_2)$
3. $A \sqcap (x_1 == x_2)$
4. $A \sqcap (x_1 != x_2)$

Exercice 4 (Vérification d'une recherche dichotomique). On considère maintenant le code ci-dessous dans un contexte où v a une valeur dans `int` inconnue et "`int f(int x);`" est une fonction externe qu'on suppose sans effet de bord observable, comme EVA le fait en l'absence de spécification ACSL explicite.

```

1  l = 0;
2  u = 10;
3  while (l < u) {
4    m = (u + 1) / 2;
5    r = f(m);
6    if (r < v) l = m + 1;
7    else if (r > v) u = m - 1;
8    else l=u=m;
9  }
10 if (l==u && v==f(l)) return l;
11 else return -1;

```

► **Question 1.** Insérer dans le programme les assertions RTE qui garantissent l'absence de débordement arithmétique. Puis, colorier ces assertions avec une analyse du programme qui prend \sqcup comme opérateur d'élargissement (*widening* en anglais).

► **Question 2.** Pour éviter un fameux bogue le jour où u sera initialisé à une valeur supérieure à 2^{30} , on change la ligne 4 en

```
m = 1 + (u - 1) / 2;
```

Qu'est-ce que ça change pour notre analyse par intervalle (en conservant ici la valeur initiale $u==10$)? Expliquer le phénomène. Comment améliorer la situation?

► **Question 3.** (Question optionnelle : à sauter si manque de temps). Pour corriger le manque de précision à la question précédente, on procède à des transformations symboliques sur les expressions contenant $/2$ (car c'est un pattern qui apparaît dans les algos "diviser-pour-régner" qu'on aimerait supporter autant que possible). Ce genre de transformation symbolique est par exemple appliqué dans l'analyseur *Astrée* sur les patterns qu'on trouve en programmation embarquée (e.g. calculs flottants de traitement du signal).

Ici, nos transformation symboliques consistent d'abord à remplacer $/2$ avec une fraction exacte, avec $x/2 = \frac{x-x\%2}{2}$. On "simplifie" ensuite les polynômes multivariés sur le corps \mathbb{Q} . Dans notre exemple, cela conduit à "simplifier" $1+(u-1)/2$ en $\frac{1}{2}1 + \frac{1}{2}u + \frac{-1}{2}(u-1)\%2$.

On peut alors sur-approximer le calcul d'un tel polynôme dans les intervalles de $\mathbb{Q} \times \mathbb{Q}$. Typiquement, on prend $[a, b]\%2 = [-1, 1]$ quand $a < 0$ et $b > 0$. Finalement, au moment de l'affectation, on restreint l'intervalle de rationnels $[q_1, q_2]$ en intervalles d'entiers $[[q_1], [q_2]]$ (puisque on sait que la valeur exacte est un entier de $[q_1, q_2]$).

Refaites l'analyse du programme ainsi transformé.

Exercice 5 (Opérateurs d'élargissements et rétrécissement). On considère le code ci-dessous, dans un contexte où N est un littéral entier fixé.

```

1  i = 0;
2  while (i < N) i++;
3  assert (i==N);

```

- **Question 1.** Faire l'analyse quand $N=4$ avec \sqcup comme opérateur d'élargissement.
- **Question 2.** Faire la même question avec $N=10000$. Pour accélérer la convergence, on prend comme opérateur d'élargissement

$$[a_1, b_1] \vee [a_2, b_2] \stackrel{\text{def}}{=} \begin{cases} [a_1, b_1] & \text{si } a_1 \leq a_2 \text{ et } b_2 \leq b_1 \\ [a_1, M_+] & \text{si } a_1 \leq a_2 \text{ et } b_1 \leq b_2 \\ [M_-, b_1] & \text{si } a_2 \leq a_1 \text{ et } b_2 \leq b_1 \\ \top & \text{sinon} \end{cases}$$

- **Question 3.** Pour gagner en précision, on ne s'arrête plus au premier invariant de boucle J trouvé par élargissement : on essaye d'abord de voir si $A_0 \sqcup A_1$ est un invariant de boucle où A_0 est l'état initial de la boucle et A_1 l'état obtenu après un tour de boucle à partir de J . En effet, comme J est un invariant, on sait que $A_0 \sqsubseteq J$ et $A_1 \sqsubseteq J$. Donc, dans un treillis, $A_0 \sqcup A_1 \sqsubseteq J$. Par conséquent, $A_0 \sqcup A_1$ s'appelle un *rétrécissement* de J . Reste à savoir si c'est lui aussi un invariant de boucle...

Commencez par ajouter au système d'inférence $\# \{A\} S \{A'\}$ une règle qui vérifie qu'un état abstrait I est un invariant de boucle, avant de le propager pour la suite du calcul. Puis appliquer votre règle pour mettre en œuvre la suggestion ci-dessus.