Gala de GAFs* à gogo

(*Generalized Anti-Frame rules)

François Pottier

Grenoble, 12–13 mars 2009

L'opérateur $\otimes$ et la règle frame

L'opérateur $\cdot \otimes I$ ajoute à *toutes* les flèches au sein de son premier argument un *effet* sur la zone de mémoire décrite par la capacité I.

$$(\chi_1 \longrightarrow \chi_2) \otimes I = (\chi_1 \otimes I) * I \longrightarrow (\chi_2 \otimes I) * I$$

La *règle frame* $\chi \leq \chi \otimes I$ permet l'utilisation d'un "Objet" par un "Client" dont les effets sur I sont inconnus de l'Objet. Par exemple,

$$\begin{array}{rl} map : & (int \longrightarrow int) \longrightarrow list\, int \longrightarrow list\, int \\ \leq & (int * I \longrightarrow int * I) \longrightarrow list\, int * I \longrightarrow list\, int * I \end{array}$$

La règle anti-frame

La règle anti-frame, *à l'inverse*, permet l'utilisation d'un Objet doté d'un état interne I par un Client qui en ignore l'existence.

$$\frac{\Gamma \otimes I \vdash (\chi \otimes I) * I}{\Gamma \vdash \chi}\ \mathsf{af}$$

Par exemple, si $I = \{x : \mathit{ref}\ \mathit{int}\}$, on peut cacher une référence x tout en exportant des méthodes *get* et *set*:

$$\frac{\vdash (\mathit{unit} * I \longrightarrow \mathit{int} * I) \times (\mathit{int} * I \longrightarrow \mathit{unit} * I) * I}{\vdash (\mathit{unit} \longrightarrow \mathit{int}) \times (\mathit{int} \longrightarrow \mathit{unit})}$$

L'*invariant* I est satisfait *à tout instant* lorsque le Client a la main.

Un exemple problématique

On souhaite *cacher* x et *vérifier* statiquement l'assertion ligne 8.

```
1  let mk () =
2    let x = ref 0 in
3    let m f =
4      x := !x + 1;
5      let v1 = !x in
6      f();
7      let v2 = !x in
8      assert (v1 = v2);
9      x := !x - 1
10   in m
```

La règle anti-frame le permet-elle? ...

Un exemple problématique

Non. Parce que x évolue, *l'invariant* I ne peut pas en fixer la valeur.

```
1  let mk () =
2    let x = ref 0 in          — soit I = { x: ref int }
3    let m f =                 — m: (unit * I → unit * I) * I → unit * I
4      x := !x + 1;            — accès à x permis, car nous possédons I
5      let v1 = !x in
6      f();                    — le type de f indique que f préserve I
7      let v2 = !x in
8      assert (v1 = v2);       — rien ne prouve que ceci soit vrai...
9      x := !x - 1
10   in m                      — à l'extérieur, m: (unit → unit) → unit
```

En fait, m et f exigent *l'un quelconque* des invariants de la *famille* $I\,i = \{\sigma : \mathit{ref}\ \mathit{int}\ i\}$, et le *préservent...*

Une GAF

Généralisons l'opérateur $\otimes$:

$$(\chi_1 \longrightarrow \chi_2) \otimes I = \forall i. \big((\chi_1 \otimes I) * I\ i \longrightarrow (\chi_2 \otimes I) * I\ i \big)$$

Modifions très légèrement la règle anti-frame en conséquence:

$$\frac{\Gamma \otimes I \vdash (\chi \otimes I) * \exists i. I\ i}{\Gamma \vdash \chi}\ \mathit{gaf}$$

L'exemple résolu

La règle généralisée s'applique parfaitement à cet exemple:

```
1  let mk () =
2    let x = ref 0 in          —  soit l i = { x: ref int i }
3    let m f =                 —  m: ∀ i. (∀ j. unit * l j → unit * l j) * l i → ur
4      x := !x + 1;            —  nous possédons l i, pour un certain i
5      let v1 = !x in          —  v1: int (i + 1)
6      f();                    —  le type de f indique que f préserve l (i + 1)
7      let v2 = !x in          —  v2: int (i + 1)
8      assert (v1 = v2);       —  succès garanti
9      x := !x - 1             —  nous rendons l i, pour le même i
10   in m                      —  à l'extérieur, m: (unit → unit) → unit
```

Conclusion

En résumé:

- Cette règle généralisée semble intéressante.
- On peut la généraliser encore et y incorporer une notion de monotonie.

Questions:

- Ces règles sont-elles correctes?
- Quelle est la GAF ultime?