# Domaines abstraits numériques non relationnels

L'objectif de ce TP est d'implémenter des domaines abstraits numériques non relationnels dans l'analyseur jouet fourni pour calculer des invariants sur le langage du cours.

### **Préliminaires**

Récupérer l'analyseur utilisé pour le TP, décompresser l'archive et installer le mode Emacs :

- > tar xvzf tiny.tgz
- > cd tiny
- > make install-emacs-mode

Puis le compiler :

> make

Ouvrir un des fichiers d'exemple fournis :

> emacs examples/ex01.tiny

puis l'analyser, soit en tapant C-c C-a dans Emacs <sup>1</sup> soit en ligne de commande :

> src/tiny examples/ex01.tiny

Le domaine fourni par défaut n'a qu'une seule valeur, qui est interprétée comme  $\bot$ , d'où des résultats étranges. Remplacer src/tiny par bin/tiny-intervals, bin/tiny-kildall ou encore bin/tiny-parity pour essayer des domaines plus intéressants <sup>2</sup>. L'objectif de ces TP va justement être d'implémenter de tels domaines.

On notera enfin qu'on peut utiliser les options --verbose et --descending pour respectivement obtenir plus de détail sur le déroulement de l'analyse et effectuer des itérations descendantes une fois le point fixe atteint <sup>3</sup> :

> bin/tiny-intervals -v 4 -d 1 examples/ex01.tiny

Lancer quelques analyses avec les différents domaines fournis (intervalles, Kildall et parité) sur les fichiers du dossier examples et observer les résultats.

### Travail demandé

Dans la suite, nous décrivons des domaines abstraits numériques par un diagramme de Hasse donnant le treillis et une fonction de concrétisation  $\gamma$ . L'objectif est d'implémenter ces domaines abstraits dans l'analyseur sous forme d'un module OCAML dont la signature se trouve dans le fichier src/nonRelational.mli ou, plus agréable à lire, dans la documentation : src/doc/NonRelational.Domain.html.

Pour ce faire, copier les fichiers dummy.mli et dummy.ml:

<sup>1.</sup> Ou via le menu TINY  $\rightarrow$  Analyze.

<sup>2.</sup> Pour utiliser C-c C-a dans Emacs, on peut changer la valeur de la variable tiny-prog-name via le menu  $TINY \rightarrow Customize\ TINY\ mode.$ 

<sup>3.</sup> Options également disponibles dans le menu TINY d'Emacs.

- > cp src/domains/dummy.mli src/domains/monDomaine.mli
- > cp src/domains/dummy.ml src/domains/monDomaine.ml

et compléter monDomaine.ml (où monDomaine est un nom de votre choix pour votre nouveau domaine) puis modifier la première ligne de code du fichier src/analyze.ml afin que l'analyseur utilise le nouveau domaine. On peut alors recompiler l'analyseur (make dans le dossier src ou C-c C-c sur un fichier OCAML dans Emacs) puis tester (./tiny ../examples/ex01.tiny dans le dossier src ou C-c C-a sur un fichier Tiny dans Emacs).

Le fichier dummy.ml contient déjà des versions correctes — mais imprécises — de la plupart des fonctions. Il est possible — et conseillé — de les garder dans un premier temps. En particulier, on gardera widening = join et on ne cherchera à implémenter les sémantiques arrières backsem\_\* que lorsqu'on sera pleinement satisfait du reste et si l'on trouve un exemple le justifiant.

## • Le domaine des constantes de Kildall

Ce domaine, vu en cours, permet d'identifier des variables qui sont constantes à un point de programme donné. Il peut ainsi être utilisé pour simplifier les programmes dans un compilateur.

Question subsidiaire : que se passe t-il dans votre domaine sur examples/ex08.tiny)? On peut essayer d'y remédier à l'aide du module InfInt fourni (documentation : src/doc/InfInt.html).

#### • Le domaine des entiers modulo 2

Ce domaine permet de représenter la parité d'une valeur.

