Assistant pour les preuves de transformations de graphes (style Hoare)

En cours de développement - Version 3

Sébastien Andres - Thomas Ceresa - Rachid Echahed

Contents

1 Introduction

Ce guide d'utilisation vous fournira des instructions détaillées sur l'utilisation de notre application. L'application est en cours de développement et ceci est la version 3. Contact: `thomas.ceresa@etu.univ-lyon1.fr`

2 Prérequis

Interpréteur Python3 : cf. `https://www.python.org/`

2.1 Installation Linux (Ubuntu-Debian)

- pip3, outil d'installation des modules de Python (si besoin) : `sudo apt-get install python3-pip`
- ply, module pour l'analyse lexicale et syntaxique : `sudo pip3 install ply`
- z3 solver, module pour vérifier les formules : `sudo pip3 install z3-solver`
- Flask, mini-framework web : `sudo pip3 install flask`

2.2 Installation Windows

Normalement, pip est installé avec Python3, sinon suivre ces indications : `https://pip.pypa.io/en/stable/installation/`

Dans un terminal :

- `pip install ply`
- `pip install z3-solver`
- `pip install flask`

3 Exécution

Le fichier à exécuter est `mainFlask.py`. Il se trouve dans le répertoire `Interface`.
Utiliser l'IDE de votre convenance ; ou en console : `python3 mainFlask.py` (`py mainFlask.py` sous Windows)
Lorsque le programme est exécuté, un serveur web est installé sur la boucle locale avec le numéro de port 5000.
Écrire l'adresse `http://localhost:5000` dans la barre d'adresse d'un navigateur pour utiliser le logiciel.

4 Utilisation

L'application permet de saisir des preuves, de les enregistrer et de les lire. Dès l'écran d'accueil, choisissez de lire un fichier ou de le saisir manuellement.

4.1 Vue d'ensemble

Les preuves sont présentées comme suit:

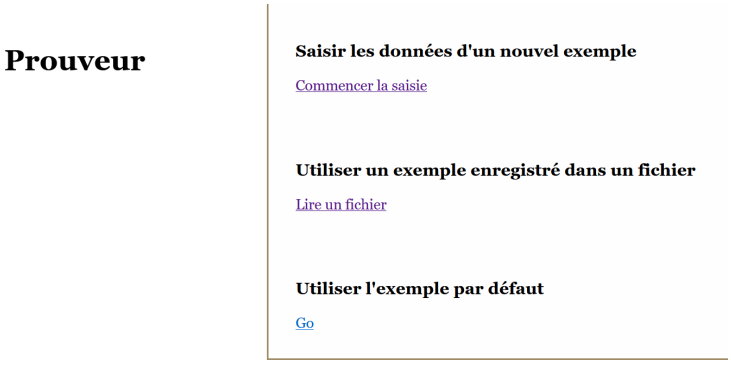

Figure 1: Écran d'accueil

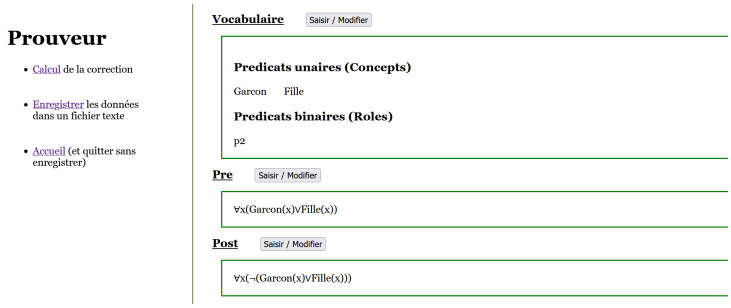

Figure 2: Vue d'ensemble partie I

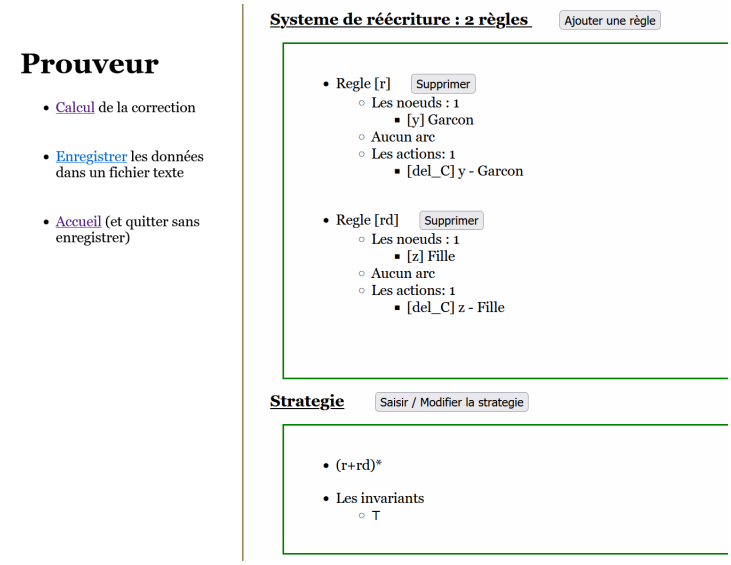

Figure 3: Vue d'ensemble partie II

4.2 Saisie du vocabulaire

La saisie du vocabulaire ou sa modification ouvre un formulaire dans lequel nous devons **ligne par ligne** indiquer les noms des concepts et des rôles.

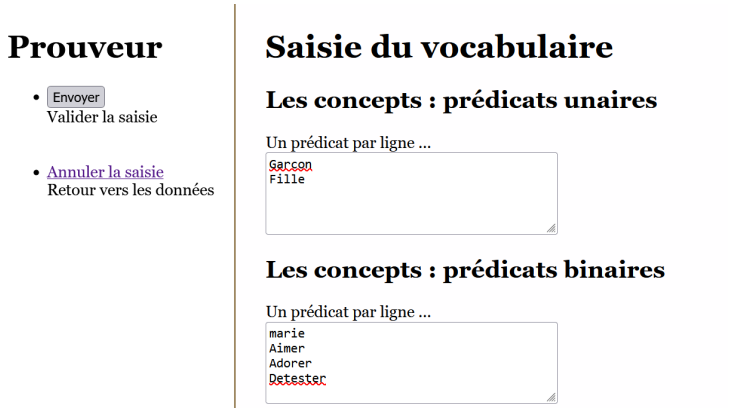

Figure 4: Exemple d'une saisie de vocabulaire

4.3 Saisie des conditions (Pre et Post)

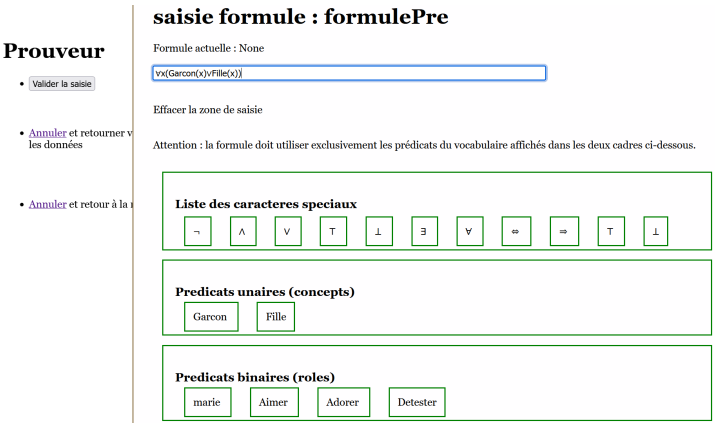

Figure 5: Saisie des conditions (Pre et Post)

Des boutons sont proposés pour faciliter l'écriture des caractères ascii. Attention: les "∃" ainsi que les "∀" doivent être entourés de parenthèses comme l'exemple ci-dessus. Les parenthèses des autres opérateurs peuvent être omises car le Lexer est souple – mais il est tout de même conseillé de les mettre pour votre compréhension et pour éviter toute ambiguïté.

4.4 Saisie d'une règle

4.4.1 Le lefthandside

On code, un par un, les nœuds et les arcs **ligne par ligne** en suivant un format bien précis expliqué dans le formulaire. Si l'on souhaite qu'un nœud soit constant (qu'il ne soit pas appliqué à un pattern-matching), un tilde '∼' doit être présent dans le nom du nœud. Ci-dessous, il y a un exemple d'une règle avec deux noeuds et un arc qui représente le fait qu'un homme appelé Gerome se marie avec une quelconque femme x qu'il aime déjà.

Figure 6: Le lefthandside de l'exemple

4.4.2 Le righthandside

Le rhs fonctionne de manière similaire. **Ligne par ligne**, on indique des actions élémentaires sous certains formats précis du formulaire. Cf. Figure 7 (ci-dessous)

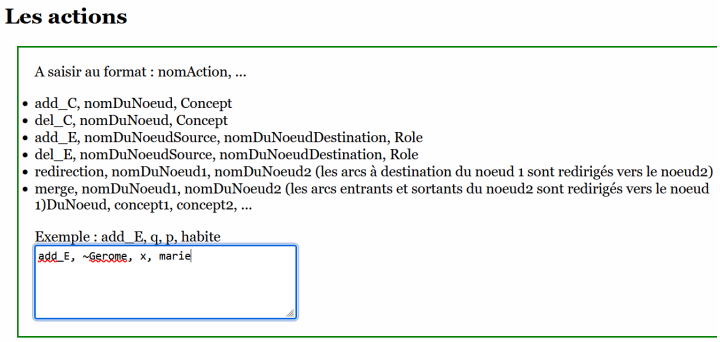

Figure 7: Le righthandside de l'exemple

4.5 Correction

4.5.1 Formule

Quand l'utilisateur a bien vérifié toutes les données de sa preuve, il clique sur "Calcul de correction". Il est alors dirigé vers une page qui contient entre autre la formule de correction.

Remarque: dans l'application des règles et des strategies, les noeuds des lhs, qui sont des variables, sont précédé par un $\exists$ et renommés par "nomDeLeurRegle_nomDuNoeud_i". "i" étant le nombre de fois que la règle est appliqué.

Formule de correction à prouver par Z3

```
Implies(ForAll(x, Or(Garcon(x), Fille(x))),
    And(Exists(rd_z_1,
        Exists(r_y_1,
            And(And(True, True),
              And(Implies(And(Not(Or(Fille(rd_z_1),
                    Garcon(r_y_1))),
                    True),
                    ForAll(x,
                    Not(Or(Garcon(x),
                    Fille(x))))),
                  Implies(And(Or(Fille(rd_z_1),
                    Garcon(r_y_1)),
                    True),
                    And(Exists(rd_z_0,
                    Implies(Fille(rd_z_0),
                    True)),
                    Exists(r_y_0,
                    Implies(Garcon(r_y_0),
                    True))))))))),
      True))
```

Figure 8: Le righthandside de l'exemple

4.5.2 Réponse & Modele brute

Une réponse courte est aussi donnée à l'utilisateur pour dire si la preuve est valide ou non. Dans un cas où elle n'est pas valide, le modèle z3 (le contre exemple) est laissé pour les utilisateurs les plus agairis qui connaissent z3 et/ou qui sont dans la recherche.

4.5.3 Contre-exemple

Ces modèles brutes de z3 sont souvent incompréhensible, ou du moins pas parlant. Le logiciel possède alors un affichage de contre-exemple. Pour chaque role ou concept, ça indique quels noeuds ou arcs les possèdent. Dans l'exemple, ci-dessous, Figure 10, Katerine, Allonzo ainsi que Trycia sont des noeuds du leftHandSide.

Remarque :

- Le contre exemple est un graphe G (il peut en avoir plusieur) qui ne satisfait pas la preuve ;
- les '...' signifient tous les autres noeuds (attributs possibles et imaginables) ;
- les 'Noeud i' signifient des noeuds qu'on ne connait pas le nom, mais qui suffisent à montrer que la preuve est fausse pour le graphe G ;
- des éléments comme x!i ou elem!i peuvent apparaître dans les contre-exemples. Il s'agit de noeud souvent important dans le contre-exemple, mais parfois non. Ce sont des variables comme les Noeud i, cependant.

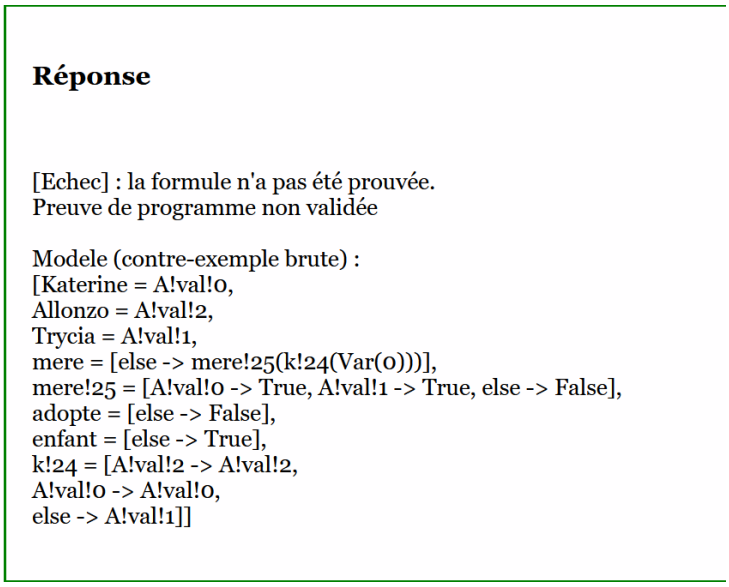

Figure 9: Reponse corte et modele brute

Figure 10: Un exemple de contre-exemple

5 Exemple d'utilisation

Pour illustrer, nous allons nous arrêter à un exemple qui couvre beaucoup de cas d'utilisation. (Notez que plusieurs exemples sont laissés à l'utilisateur dans le dossier "Exemple". Ce dernier est invité à les générer dans l'application, dès l'écran d'accueil, en sélectionnant dans le menu "Lire un fichier".)

Soit L un left-hand-side, dont le seul nœud est k. k a comme unique concept *pred*. Soit $\alpha =$ `del_C(k, pred)` "on supprime le concept 'pred' d'un nœud". Soit $r = (L, \alpha)$.

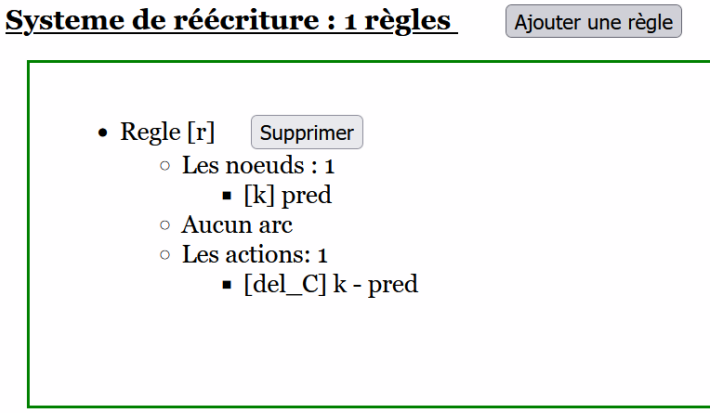

Figure 11: La règle r

Et on veut observer :

1. $\{pred(x)\}r\{\neg pred(x)\}$
2. $\{\forall x, pred(x)\}r\{\forall x, \neg pred(x)\}$
3. $\{\forall x, pred(x)\}r\{\exists x, \neg pred(x)\}$
4. $\{\forall x, pred(x)\}r^*_{\{\top\}}\{\forall x, \neg pred(x)\}$

En d'autres termes, nous voulons montrer qu'à la fin, la règle supprime bien le concept du (des) nœud(s).
Il est très important de comprendre que le 1. va être prouvé par le logiciel. Quand l'utilisateur ne précise pas la quantification de la variable x, ça reviendrait à dire que x est une variable libre. On peut aussi dire que 1. est équivalent à $\{\exists x, pred(x)\}r\{\exists x, \neg pred(x)\}$. Ce qui est tout le temps vrai, car s'il existe au moins un nœud x et qu'on applique r, à la fin il existera forcément un nœud x' qui n'aura pas le concept *pred*.

Par contre, le 2. est faux. Et l'application évidemment ne le prouve pas. Car, si on applique une fois **et une seule fois** r aux graphes qui ont plusieurs nœuds (ayant le concept *pred*), on applique r à un seul nœud. Donc, la postcondition $\{\forall x, \neg pred(x)\}$ n'est pas respectée. L'application donne, d'ailleurs, un contre-exemple où il y a plein de nœuds qui ont le concept *pred*, représenté par '...' :

Le 3. est cependant prouvé car, à la fin, il y a bien un seul nœud qui a été changé.

Enfin, le 4. a été prouvé. En effet, dans ce cas, on applique r tant que c'est possible, c'est-à-dire sur tous les nœuds. Ce qui explique pourquoi il n'y a plus aucun nœud qui a le concept *pred*. Voici la réponse du 4. :

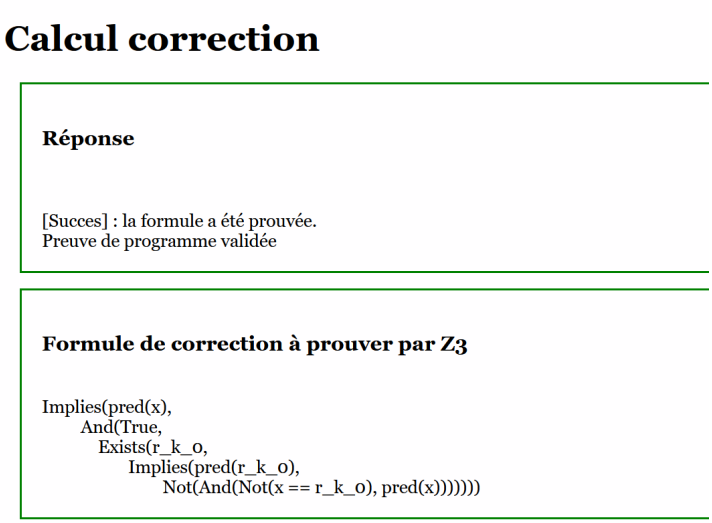

Figure 12: Réponse de la correction du 1.

Figure 13: Contre-Exemple du 2.

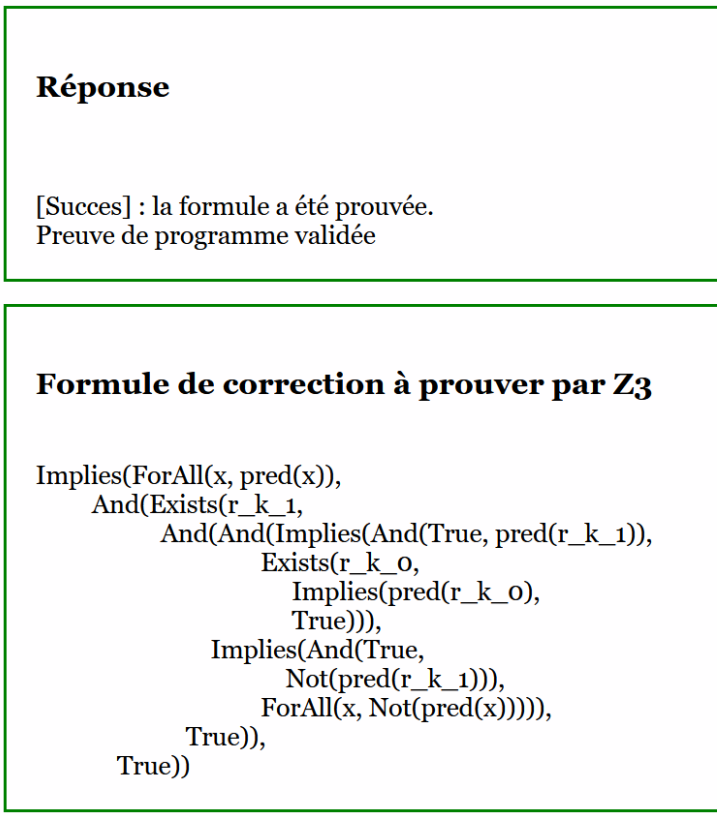

Figure 14: Réponse de la correction du 4.