Projet de Caml

Licence d'informatique

—Programmation fonctionnelle—

Calcul des séquents

Le but de ce projet est d'écrire un programme en CAML permettant d'évaluer efficacement une formule. Pour cela, on utilisera la méthode des séquents vue en td.

Définition du type formule

On utilisera le type terme suivant :

```
type var = string;;
type const = string;;
type terme = Variable of var
           | Constant of const * terme list;;
```

Ainsi, le terme $x+3$ sera écrit `Constant ("+", [Variable ("x"); Constant ("3", [])])` (noter qu'une constante d'individu comme 3 est une fonction d'arité 0, donc la liste des arguments est vide).

On utilisera le type `formule` suivant :

```
type formule = Predicate of const * terme list
             | Forall of var * formule
             | Exists of var * formule
             | Or of formule * formule
             | And of formule * formule
             | Implies of formule * formule
             | Not of formule;;
```

Par exemple, la formule $\forall x, x < 0$ s'écrit :
`Forall("x", (Predicate( "<", [Variable "x";Constant( "0", [])])))`

- Écrire une fonction `formule_to_string` prenant une formule et renvoyant une chaîne de caractères représentant cette formule. Par exemple, pour la formule précédente :

```
# formule_to_string(Forall("x", (Predicate("<", [Variable "x";Constant("0", [])]))));;
- : string = "pour tout x, (<(x,0))"
```

Définition du type séquent

Rappelons qu'un séquent est défini par deux ensembles de formules, l'ensemble des antécédents (à gauche) et l'ensemble des conséquents (à droite). On utilisera le type `sequent` suivant :

```
type sequent = {gauche:formule list; droite:formule list};;
```

Par exemple, le séquent $A \wedge B \Rightarrow A \vee B$ est défini de la façon suivante :

```
{gauche= [Or(Predicate("A",[]),Predicate("B",[]))];
 droite=[And(Predicate("A",[]),Predicate("B",[]))]};;
```

- Écrire une fonction `sequent_to_string` prenant un séquent et renvoyant une chaîne de caractères représentant ce séquent. Par exemple, pour le séquent précédent :

```
sequent_to_string ({gauche= [Or(Predicate("A",[]),Predicate("B",[]))];
droite=[And(Predicate("A",[]),Predicate("B",[]))]});;
- : string = "(A) or (B) => (A) and (B)"
```

Calcul des séquents pour les propositions

On rappelle les règles du calcul de séquents pour les propositions :

$$G_\vee : \frac{\Gamma,F\Rightarrow\Delta \quad \Gamma,G\Rightarrow\Delta}{\Gamma,F\vee G\Rightarrow\Delta} \qquad D_\vee : \frac{\Gamma\Rightarrow\Delta,F,G}{\Gamma\Rightarrow\Delta,F\vee G}$$

$$G_\wedge : \frac{\Gamma,F,G\Rightarrow\Delta}{\Gamma,F\wedge G\Rightarrow\Delta} \qquad D_\wedge : \frac{\Gamma\Rightarrow\Delta,F \quad \Gamma\Rightarrow\Delta,G}{\Gamma\Rightarrow\Delta,F\wedge G}$$

$$G_\rightarrow : \frac{\Gamma\Rightarrow\Delta,F \quad \Gamma,G\Rightarrow\Delta}{\Gamma,F\rightarrow G\Rightarrow\Delta} \qquad D_\rightarrow : \frac{\Gamma,F\Rightarrow\Delta,G}{\Gamma\Rightarrow\Delta,F\rightarrow G}$$

$$G_\neg : \frac{\Gamma\Rightarrow\Delta,F}{\Gamma,\neg F\Rightarrow\Delta} \qquad D_\neg : \frac{\Gamma,G\Rightarrow\Delta}{\Gamma\Rightarrow\Delta,\neg G}$$

- Écrire une fonction qui vérifie qu'un séquent est un axiome.
- Écrire une fonction qui réduit un séquent selon la règle adéquate quand c'est possible. Cette fonction devra renvoyer une liste de séquents (prémisses de la règle).
- Écrire une fonction qui effectue et affiche le calcul des séquents d'une proposition. Par exemple, lorsqu'on applique cette fonction à la formule $A \wedge B \Rightarrow A \vee B$, on obtient :

```
to prove:  => ((a) and (b)) -> ((a) or (b))
to prove: (a) and (b) => (a) or (b)
to prove: a,b => (a) or (b)
to prove: a,b => a,b
is axiom
- : bool = true
```

Calcul des séquents pour les prédicats

On rajoute les règles de calcul de séquents pour les quantificateurs $\exists$ et $\forall$:

$$G_\forall : \frac{\Gamma,A(t),\forall x A(x)\Rightarrow\Delta}{\Gamma,\forall x A(x)\Rightarrow\Delta} \qquad D_\forall : \frac{\Gamma\Rightarrow A(a),\Delta}{\Gamma\Rightarrow\forall x A(x),\Delta} \quad \text{où } a \text{ est une variable fraîche}$$

$$G_\exists : \frac{\Gamma,A(a)\Rightarrow\Delta}{\Gamma,\exists x A(x)\Rightarrow\Delta} \qquad D_\exists : \frac{\Gamma\Rightarrow A(t),\exists x A(x),\Delta}{\Gamma\Rightarrow\exists x A(x),\Delta} \quad \text{et } t \text{ est un terme}$$

Écrire une fonction qui effectue et affiche le calcul des séquents d'un prédicat. Vous préciserez dans votre rapport l'algorithme pour l'application des règles $G\forall$: et $D\exists$: (choix de la constante t).