GUPU — un environnement pour l'enseignement de Prolog et de la PLC

Ulrich Neumerkel

Université technique de Vienne — Ulrich.Neumerkel@tuwien.ac.at

Langage : Prolog pur & extensions de la PLC, CLP(FD) de SICStus.
– restrictions syntaxiques pour remédier aux difficultés de la syntaxe édimbourgeoise
– standards de codage

Lectures sélectives à plusieurs niveaux
– lectures traditionnelles : déclaratives et opérationnelles
– raffiner par transformations :

généralisation : enlever des buts
père(Père) ←
    ***** ~~masculin(Père)~~,
    enfant_de(_E, Père).

spécialisation : ajouter des buts
~~conjoint_de(Époux, Épouse) ← **false**~~,
    ~~époux_épouse(Époux, Épouse)~~.
conjoint_de(Épouse, Époux) ←
    époux_épouse(Époux, Épouse).

+ facilitent les lectures des programmes plus volumineux
+ fidélité au code source, présentation simplifiée (rayer, cacher)
+ pas de nouveau formalisme comme les arbres de preuve, les traces
+ convient avec contraintes incomplètes

GUPU : Assertions — tester *puis* coder

Pratique d'écrire les tests *puis* coder (*extreme programming*)
– assertions *dans* le programme
– un seul commande pour sauvegarder, validation syntaxique, compilation, tests des assertions
– assertions positives : $\leftarrow$ *But*.

$\leftarrow$ ville(V). % assertion utilisée comme requête — substitutions *dans* le programme
@@ % V = aberdeen.
@@ % V = alcázar.
@@ % V = amsterdam.
@@ % V = ånge.
@@ % V = århus.
@@ ? Plusieurs solutions avec ESPACE

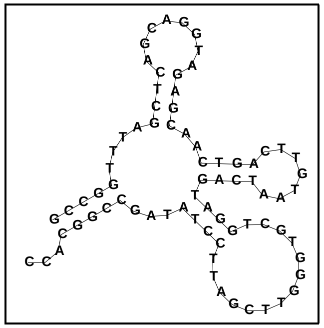
– assertions négatives : $\not\leftarrow$ masculin(P), féminin(P).
– assertions pour la terminaison universelle :
$\leftarrow$ *But*.
$\not\leftarrow$ *But*, false.
– assertions coûteuses : $\overset{\$}{\leftarrow}$ *But*.
– assertions infinies : $\overset{\infty}{\leftarrow}$ *But*.
infinité des solutions $\Rightarrow$ nonterminaison : $\overset{\infty}{\not\leftarrow}$ liste_sans(Xs,X), false.

Pré-notation automatisée
– pour renforcer l'écriture des assertions
– retour immédiat
– délais flexibles par ponderation
– intervalle approximatif 0%..100%

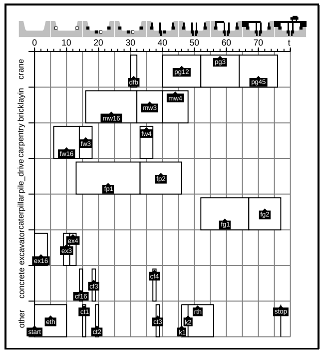
GUPU : Visualisateurs

- affichage des graphismes
- sans effet de bords
- assertions annotées : $\leftarrow$ *Visualisateur* $\lll$ *But.*

$\leftarrow$ ville(V).
$\leftarrow$ **ville_en_europe(V)** $\lll$ ville(V). % assertion annotée
...
@@ % V = paris.
...



$\stackrel{\$}{\leftarrow}$ repliement(F) $\lll$ location_repliement(210..285,F).
% F = [libre([]),liaison("GCCGG",[libre("TTTA"),liaison("GCTC",[libre("AGCAGGTA")]), ...]), ...].



$\stackrel{\$}{\leftarrow}$ bridgeplan(DB-Plan) $\lll$ db_plan(DB,Plan).
% DB = small, Plan = [task(start,0,0),task(ex16,0,4), ... task(k2,48,0),task(stop,77,0)].

GUPU : Explications — *slicing*

– fragments du programme, obtenus par transformations de programmes, *doivent* être changés
– facilite des lectures sélectives, sans interaction d'usager
– explication pour échec inattendu (insuffisance), solution inattendue (incorrection), non terminaison

échec inattendu

← frère_de(B, P). % insuffisance

frère_de(F, P) ←
    dif(F,P),
    masculin(F),
    enfant_de(F,V),
    masculin(V),
    enfant_de(P,V),
    enfant_de(F,M),
    enfant_de(P,M),
    féminin(V).

masculin(franz_I).
masculin(joseph_II).
masculin(leopold_II).

féminin(maria_theresia).

enfant_de(joseph_II, maria_theresia).
enfant_de(joseph_II, franz_I).
enfant_de(leopold_II, maria_theresia).
enfant_de(leopold_II, franz_I).

échec inattendu de la généralisation maximale

← frère_de(B, P). % insuffisance

frère_de(F, P) ←
  * ~~dif(F,P),~~
  * ~~masculin(F),~~
  * ~~enfant_de(F,V),~~
    masculin(V),
  * ~~enfant_de(P,V),~~
  * ~~enfant_de(F,M),~~
  * ~~enfant_de(P,M),~~
    féminin(V).

masculin(franz_I).
masculin(joseph_II).
masculin(leopold_II).

féminin(maria_theresia).

~~enfant_de(joseph_II, maria_theresia).~~
~~enfant_de(joseph_II, franz_I).~~
~~enfant_de(leopold_II, maria_theresia).~~
~~enfant_de(leopold_II, franz_I).~~