Programming Whitout Code: An Approach & Environment To Conditions-on-Data Programming

Ph. Larvet
Former Research Engineer, Alcatel-Lucent Bell Labs
June 2017
I Have A Dream

Specifications, expressed in Natural Language

Generated code
The Present Approach

Spec of a complex system to be validated

Mechanism for validation or simulation

= A Program!
What Is A Program?

Within the context of OO programming, a Program is a set of interacting Objects.

- OBJECT1
  - Data
  - Methods (operations)

- OBJECT2
  - Data
  - Methods (operations)

- OBJECT3
  - Data
  - Methods (operations)

- OBJECT4
  - Data
  - Methods (operations)
What is an Object?

- An Object is a code structure that encapsulates **data** and **methods** that work on them.

```
OBJECT
  Data
  Methods
```

- Variables, information owned by the Object.
- Operations that process object’s data.
Methods Contain Statements

 Statements (instructions) can be divided in four main groups:

- **Accessors** = to get data from other objects
- **Calculations** = to compute new values of variables & data
- **Tests** = to organize the internal logic of the method (if… then… else; do… loop; while…; repeat... until; etc.)
- **Generators of events** = to call methods on other objects
The Bricks Of A Method

- => A, T, C, G are fundamental bricks to build a method

- Do these letters remind you something?

- (Let’s open a small parenthesis)
A Protein Is Made Of Bricks

Sample: amino-acids sequence in human insulin
Final Bricks Are A,T,C,G

- Each amino-acid (Gly, Val, Ala, Glu, Pro, etc.) is a set of A,C,T,G
A, T, C, G: The Bricks of Life
A,C,T,G To Build Programs

- As a Protein is a Program, a Program can be built as a set of A,C,T,G
  (Let’s close now the small parenthesis)

- From a detailed point of view, A,C,T,G can be seen as equalities (equations) or « conditions on data » (COD)
An Accessor Is An Equality

```
myData = OBJECT2.get(Data);
```
A Calculation Is An Equality

g = 0;
delta = (b*b) - (4*a*c);
i++;
query = "SELECT *
FROM customers
WHERE cID = 256";
A Generator Is An Equality

- An event Generator inside a method of OBJ1 is a call of a method on OBJ2

```c
result = OBJECT2.funct2(param);
```
A Test Is A Condition On Data

if (var1 == val01 &&
    var2 > val02) {
    g = 0;
    i++;
}

can be expressed as:

var1 = val01 AND var2 > val02
=> g = 0 AND i++
A Test Is A Condition On Data (2)

while (content < demand) {
    //operation
    j++; 
    out = display(TAB[j]); 
}

can be expressed as:

content < demand => j++ AND out = display(TAB[j])

and we need a mechanism to express or implement the boucle
First Summary

- We propose to simplify the writing of programs by expressing only
  - equalities
  - conditions on data (COD)

- So, we don’t write any more complex algorithms in terms of « code », but only CODs

- And we need a mechanism to implement the dynamics => HOW TO DO THIS ?
A COD Is A Rule

- A condition on data, expressed as
  - \( var1 = val01 \Rightarrow g = 0 \) AND \( i++ \)
- can be seen as a « production rule » in a rule-based system (expert system) :
  - PREMISE : \( var1 = val01 \)
  - CONCLUSION : \( g = 0 \) AND \( i = i + 1 \)
CODs Are Distributed Rules

- Main differences between a rule-based system and the COD approach:
  - in our COD approach, CODs express conditions on equalities (i.e. equations) instead of predicates or « logical production rules »
  - rules (i.e. CODs) are not stored in a unique rule database but are distributed into all objects because a COD belongs to a given object
  - idem, facts (i.e. Data) are not stored in a unique fact database but are distributed into objects, because each object stores its own data
The COD Environment

- OBJECT1
  - Equalities
  - CODs

- OBJECT2
  - Equalities
  - CODs

- OBJECT3
  - Equalities
  - CODs

- OBJECT4
  - Equalities
  - CODs

- INFERENCE ENGINE
Inside The Inference Engine

• 'on balaie les "faits" de TabN
• 'et on cherche dans les "règles" des objets si certaines prémisses peuvent décl. des concl.
• 'on répète le traitement jusqu'à ce qu'aucune nouvelle concl ne soit décl.

nouv_conclusion = False

• 'on balaie les RLists des objets pour chercher une éventuelle règle à déclencher
  For i = 0 To nbObj
  • 'on balaie la RList de chaque objet
  For j = 0 To RList(i).ListCount - 1
  reg$ = RList(i).List(j)
  • prem$ = gauche(reg$, ";=") 'extraction de la prémisses
  If InStr(prem$, " AND ") > 0 Then
    spreml$ = gauche(prem$, " AND ")
    sprems$ = droite(prem$, " AND ")
    If prem_verifiee(spreml$) = True And prem_verifiee(sprems$) = True Then
      decl_concl i, j
      nouv_conclusion = True
    End If
  Else
    If prem_verifiee(prem$) = True Then
      decl_concl i, j
      nouv_conclusion = True
    End If
  End If
  Next
  Next

• If nouv_conclusion = True Then moteur
Building An Example

- Filling tank
- Reactor
- Emptying electro-valve
- Storage tank
- Filling electro-valve
- Heating regulator

Demo!
Object Model of the Example