TOM

Anne-Claire Lonchamp
Yoann Toussaint

Equipe PROTHEO

Sommaire

- Présentation de Prothéo
- Tom
  - Présentation des mécanismes de base
  - Tom et Vas
  - Tom et XML
  - Strategies
- Industrialisation et intégration de Tom dans un environnement Java
- Extension du langage et notions de contraintes de filtrage

Equipe PROTHEO

Spécification et vérification de logiciels

- Contraintes
- Règles de réécriture
- Stratégies

TOM

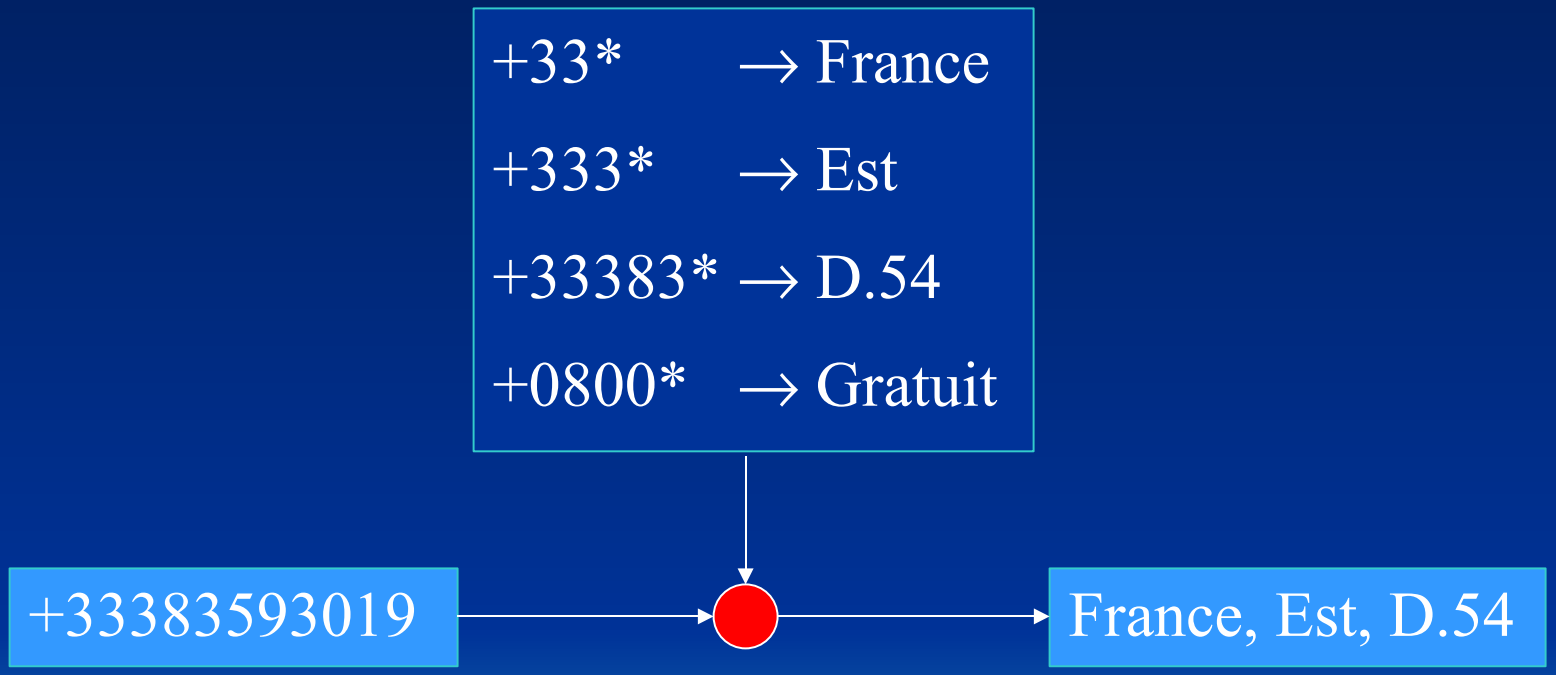
http://tom.loria.fr/

Programmation par règles
Application à XML

Qu'est ce que Tom ?

- Un langage qui permet de :
  - Construire des structures arborescentes.
  - Reconnaître des motifs dans une structure de donnée.
- Lorsqu'un motif est reconnu :
  - On dit que le motif « filtre » vers la structure.
  - On peut exécuter une action, écrite en C ou Java.

Exemple


+33* → France
+333* → Est
+33383* → D.54
+0800* → Gratuit
+33383593019
France, Est, D.54

Mécanisme de base (1)

- `f(g(a))
- permet de construire le terme

Mécanisme de base (2)

```
t = `f(g(a))
match(t) {
    f(g(x)) → { Action }
}
```

- Permet de filtrer le motif f(g(x)) vers le terme f(g(a))
- Execute l'action avec x = a

Outil puissant

- Grande expressivité
- Types de données abstrait :
  - Mapping
  - Ancrage formel
- Prouveur :
  - Coq
  - Zenon

Outil puissant

- Filtrage associatif avec élément neutre :

```
%match(Term subject) {
        f(x,x) -> {return `x;}
        _      ->  {return `y;}
}
```

Tom et Vas

- Vas : permet de définir un type de donnée abstrait.

```
%vas{
    module Term
    imports
    public sorts Term ListTerm

    abstract syntax
    a -> Term
    b -> Term
    f(s1:Term,s2:Term) -> Term
    concTerm(Term*) -> ListTerm
}
```

Outil de programmation

- Tom permet de programmer en C ou Java
- Plugin pour Eclipse
- Utilisation de la plateforme Gforge :
  - CVS
  - http://gforge.inria.fr/projects/tom/

Manipulation de listes

- (x*,a,y*) : recherche l'élement a dans la liste
- (_*,x,_*) if P(x) : recherche un élement x ayant la propriété P
- f(X1*,x,x,Y*) $\rightarrow$ f(X*,x,Y*)
  - permet d'éliminer les doublons
- f(X*,x,y,Y*) $\rightarrow$ f(X*,y,x,Y*) if y<x
  - permet de trier une liste

Extension XML

- <A> <B state=« idle »/> </A>
  - permet de trouver un nœud B ayant un attribut state=« idle » dans un document XML
- <A>

  <B state=« active »/>

  <B state=« idle »/>

  <B state=« active »/>

  </A>

Construction et transformation

- t = `xml(<A> <A> <B/> </A> </A>)
- Exemple de transformation :
  - <Person name=n/> → <Personne nom=n/>
- Plus généralement
  - Morceau de document → Morceau' de document

Application à XML

```
Node sort(Node subject) {
    %match(subject) {
        <Persons>(X1*,p1,X2*,p2,X3*)</Persons> -> {
            if(`compare(p1,p2) > 0) {
                return sort(`xml(<Persons>X1* p2 X2* p1 X3*</Persons>));
            }
        }
    }
    return subject;
}
```

Application à XML

```
int compare(Node t1, Node t2) {
    %match(t1, t2) {
      <Person Age=a1><FirstName> n1 </FirstName></Person>,
      <Person Age=a2><FirstName> n2 </FirstName></Person>
      -> { return `a1.compareTo(`a2); }
    }
    return 0;
  }
```

Stratégies

```
%strategy RewriteSystem extends Identity{
      visit Term {
            f(x,x) -> {return `x;}
            }
}
VisitableVisitor rule = new RewriteSystem;
Term subject = `f(a,a);
MuTraveler.init(rule).visit(subject);
```

Stratégies

- Différentes stratégies :
  - BottomUp
  - BottomUpId
  - OnceBottomUp
  - Innermost
  - InnermostId

Industrialisation et intégration de Tom dans un environnement Java

- Travail :
  - Réalisation d'un analyseur syntaxique permettant de rendre uniforme les constructions de Java et les constructions de Tom

Industrialisation et integration de Tom dans un environnement Java

- Intérêts:
  - Utilisation de Tom plus facile
  - Optimisations plus faciles
  - Typage statique plus puissant
  - Messages d'erreurs plus précis

Industrialisation et intégration de Tom dans un environnement Java

Exemple:

```
String dico;
%match(String dico) {
"hello" -> {return "bonjour";}
}
```

Extensions du langage et notions de contraintes de filtrage

- La notion de contrainte de filtrage
- Le 'when', une première implémentation

Syntaxe :

```
%match(T s) {
   p when c -> {...}
}
```

Extensions du langage et notions de contraintes de filtrage

- Vers des contraintes plus algébriques (%op allégé pour typer les fonctions java)
- Vers des contraintes multiples (c1^c2^...^cn)

TP à 14h !!!!

- Questions, détails techniques,…
  - Comment ca marche ?
  - Que peut-on faire ?
  - A quoi cela sert ?
  - Est-il possible de …?
- Travaux Pratiques à 14h !!!