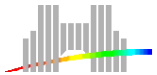
Master Informatique Fondamentale - M1 Compilation

Au delà des grammaires hors contexte

Paul Feautrier

ENS de Lyon
Paul.Feautrier@ens-lyon.fr
perso.ens-lyon.fr/paul.feautrier

4 février 2007



Arbre d'analyse syntaxique, I

On associe un type à chaque production du langage :
Exemple :

```
cond : IF expr THEN stat
     | IF expr THEN stat ELSE stat
     ;
```

Il y a deux productions, cond1 et cond2 :

```
type cond1 = {predicat: expr;
              true_part: stat;
              ...
             }

and cond2 = {predicat: expr;
             true_part: stat;
             false_part: stat;
             ...
            }
```

Arbre d'analyse syntaxique, II

... et un type à chaque terminal :

```
and cond = Cond1 of cond1
         | Cond2 of cond2

and expr = ...
and stat = ...
;;
```

- Il est possible de rajouter de l'information dans les structures `cond1` et `cond2`.
- Il est possible d'optimiser la structure de l'arbre.

Exemples d'optimisation

Factorisation

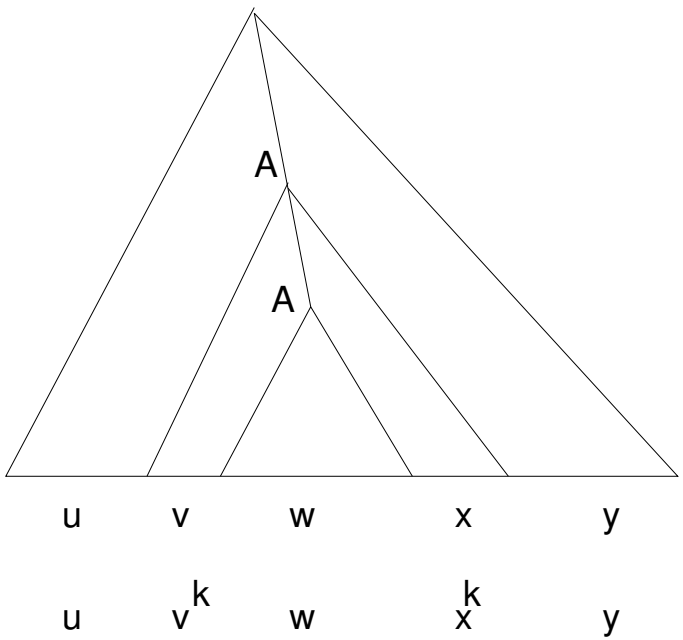
```
type cond = {predicat: expr;
             true_part : stat;
             false_part: stat
            }
and stat = Nop
         | ...
and expr = {...
            mutable sorte: cType;
           }
```

Réification

```
 and args = {arg1 : expr;
             arg2 : expr
            }
 and expr = Plus of args
          | Minus of args
          | Times of args
...
```

```
and expr = {op: string;
            arg1: expr;
            arg2: expr;
            mutable sorte: cType;
            ...
           }
```

Certains langages n'ont pas de grammaire hors contexte



Lemme de pompage pour les grammaires hors contexte.

- Exemple : $\{a^n b^n c^n \mid n \geq 0\}$.
- Il est possible de faire grandir en synchronisme deux des sous-mots a^n, b^n ou c^n, mais pas les trois à la fois.

Grammaires attribuées

- Un attribut est une valeur quelconque que l'on attache à un nœud de l'arbre d'analyse syntaxique.
- En Ocaml, c'est une composante supplémentaire des structures qui matérialisent les nœuds.
- Les attributs ne peuvent en général pas être calculés au vol. Ils doivent donc être déclarés `mutables` et on doit prévoir une valeur par défaut : `"" [] Nop None`.
- On peut annoter les productions par des règles de calcul d'attributs.
- Intérêt : aller au delà des grammaires hors contexte, par exemple pour spécifier la sémantique ou des vérifications supplémentaires.

Exemple : la sémantique d'un nombre binaire

```
  nonzero : UN              {$0.val = 1}
| nonzero : nonzero ZERO    {$0.val = 2*$1.val}
| nonzero : nonzero UN      {$0.val = 2*$1.val + 1}
;

bin : ZERO                  {$0.val = 0}
    | nonzero               {$0.val = $1.val}
```

Notations :

- `$0$` le (nœud) membre gauche de la production
- `$1`, `$2`, ... les nœuds du membre droit de la production
- `$x.id` l'attribut `id` du nœud x.

Attributs synthétisés, hérités

- Un attribut de la forme `$0.a = f($1.b,...,$n.c)` est dit synthétisé. Il ne dépend que d'attributs situés en dessous de lui dans l'arbre syntaxique.
- Un attribut de la forme `$n.a = f($0.x,...,$0.z)` est dit hérité.
- Les attributs qui ne sont ni synthétisés ni hérités sont mixtes.
- Si on utilise un analyseur ascendant, on peut calculer les attributs synthétisés "au vol".
- L'arbre sysntaxique peut être considéré comme un attribut synthétisé particulier.
- Dans les autres cas, il faut une phase indépendante pour le calcul des attributs.

Une autre version du binaire

```
bin : bits        {$1.pds = 1;
                   $0.val = $1.val}
    ;

bits : bits bin  {$1.pds = 2*$0.pds;
                  $0.val = $1.val + $2.val;
                  $2.pds = $0.pds}

     | bit        {$1.pds = 2*$0.pds;
                   $0.val = $1.val}
     ;

bit : ZERO        {$0.val = 0}
    |             {$0.val = $0.pds}
    ;
```

Ordonnancement d'une grammaire attribuée

- Construire l'arbre syntaxique.
- Dans les règles de calcul des attributs, remplacer les $x par les noms des nœuds correspondants.
- Résoudre le système d'équations ainsi obtenu en suivant les dépendances.

Exemple : traduction dirigée par la syntaxe, I

Machine cible Un ordinateur RISC avec un grand nombre de registres `R0, R1, .....` 5 types d'instructions :

- Lecture de la mémoire : `LOAD Rn, adresse`
- Chargement d'une constante : `LOADI, Rn, valeur`
- Ecriture en mémoire : `STORE Rn, adresse`
- Addition : `ADD Ra,Rb,Rc`
- Multiplication `MULT Ra,Rb,Rc`

Attributs

- `code` le code engendré par un sous-arbre de l'arbre syntaxique (synthétisé)
- `registre` le registre qui contient le résultat du code ci-dessus (hérité).
- `id` le texte d'un identificateur (engendré par Lex)
- `val` la valeur d'un entier, engendré par Lex

Traduction dirigée par la syntaxe, II

Grammaire

```
stats : stat            {$0.code = $1.code}
      | stats stat      {$0.code = $1.code ^ $2.code} ;
stat : IDENT EQUAL rhs SEMICOLON
       {$0.code = $3.code ^ "STORE R1," ^ $1.id;
        $3.registre = 1}
rhs : IDENT             {$0.code = "LOAD " ^ $0.registre ^ "," $1.id}
    | INT               {$0.code = "LOADI " ^ $0.registre ^ "," $1.val}
    | LPAR rhs PLUS rhs RPAR
      {$2.registre = $0.registre;
       $4.registre = $0.registre + 1;
       $0.code = $2.code ^ $4.code
                 ^ "ADD " $0.registre, $0.registre $1.registre}
    | LPAR rhs MULT rhs RPAR
      {$2.registre = $0.registre;
       $4.registre = $0.registre + 1;
       $0.code = $2.code ^ $4.code
                 ^ "MULT " $0.registre, $0.registre $1.registre};
```

Yacc comme synthétiseur d'attributs

- De nombreux outils pour la compilation de grammaires attribuées ont été construits dans les années 80 (exemple : DELTA de Bernard Lorho).
- Ces outils ont été abandonnés parce que peu efficaces (nombreuses opérations de recopie) et difficile à utiliser (calculs locaux seulement).
- Dans chaque cas particulier, il est plus efficace de construire un programme *ad hoc* qui peut être optimisé de diverses façons (tables non locales, ordonnancement *a priori*, etc).
- Cependant, Yacc conserve un système d'attributs seulement synthétisés.
- Il n'y a qu'un seul attribut par nœud, mais ce n'est pas un problème si le langage cible est assez puissant.
- Tout le langage cible est disponible pour le calcul des attributs.
- En général, on utilise cet outil pour construire l'arbre d"analyse syntaxique du programme.