Sémantique et Traduction des Langages

Majeure Sciences et Ingénierie du Logiciel

Marc Pantel

2020 – 2021

Organisation

- Cours : 10 séances – Marc Pantel
- TD : 8 séances – Marc Pantel/Neeraj Singh
- TP : 9 séances – Marc Pantel/Neeraj Singh
- Mini Projets (20%) en binôme : 2 séances de suivi + 1 test
  Finalisation travaux de TP sur langage fonctionnel miniML
  Finalisation travaux de TP sur langage impératif miniC
- Projet (40%) en quadrinôme : 5 séances de suivi + 1 test
  Extension du langage miniC avec technologies objets
- Examen (40%) : 1h30 avec documents
- Urgent : Constituer les quadrinômes et binômes associés
- Alternative 1 : Pas de confinement
  - Travaux Dirigées en présenciel
  - Travaux Pratiques à décider
- Alternative 2 : Confinement
  - Combinaison TD/TP à distance

Organisation

- Cours : 10 séances – Marc Pantel
- TD : 8 séances – Marc Pantel/Neeraj Singh
- TP : 9 séances – Marc Pantel/Neeraj Singh
- Mini Projets (20%) en binôme : 2 séances de suivi + 1 test
  Finalisation travaux de TP sur langage fonctionnel miniML
  Finalisation travaux de TP sur langage impératif miniC
- Projet (40%) en quadrinôme : 5 séances de suivi + 1 test
  Extension du langage miniC avec technologies objets
- Examen (40%) : 1h30 avec documents
- Urgent : Constituer les quadrinômes et binômes associés
- Alternative 1 : Pas de confinement
  - Travaux Dirigées en présenciel
  - Travaux Pratiques à décider
- Alternative 2 : Confinement
  - Combinaison TD/TP à distance

Organisation

- Cours : 10 séances – Marc Pantel
- TD : 8 séances – Marc Pantel/Neeraj Singh
- TP : 9 séances – Marc Pantel/Neeraj Singh
- Mini Projets (20%) en binôme : 2 séances de suivi + 1 test
  Finalisation travaux de TP sur langage fonctionnel miniML
  Finalisation travaux de TP sur langage impératif miniC
- Projet (40%) en quadrinôme : 5 séances de suivi + 1 test
  Extension du langage miniC avec technologies objets
- Examen (40%) : 1h30 avec documents
- Urgent : Constituer les quadrinômes et binômes associés
- Alternative 1 : Pas de confinement
  - Travaux Dirigées en présenciel
  - Travaux Pratiques à décider
- Alternative 2 : Confinement
  - Combinaison TD/TP à distance

Organisation

- Cours : 10 séances – Marc Pantel
- TD : 8 séances – Marc Pantel/Neeraj Singh
- TP : 9 séances – Marc Pantel/Neeraj Singh
- Mini Projets (20%) en binôme : 2 séances de suivi + 1 test
  Finalisation travaux de TP sur langage fonctionnel miniML
  Finalisation travaux de TP sur langage impératif miniC
- Projet (40%) en quadrinôme : 5 séances de suivi + 1 test
  Extension du langage miniC avec technologies objets
- Examen (40%) : 1h30 avec documents
- Urgent : Constituer les quadrinômes et binômes associés
- Alternative 1 : Pas de confinement
  - Travaux Dirigées en présenciel
  - Travaux Pratiques à décider
- Alternative 2 : Confinement
  - Combinaison TD/TP à distance

Organisation

- Cours : 10 séances – Marc Pantel
- TD : 8 séances – Marc Pantel/Neeraj Singh
- TP : 9 séances – Marc Pantel/Neeraj Singh
- Mini Projets (20%) en binôme : 2 séances de suivi + 1 test
  Finalisation travaux de TP sur langage fonctionnel miniML
  Finalisation travaux de TP sur langage impératif miniC
- Projet (40%) en quadrinôme : 5 séances de suivi + 1 test
  Extension du langage miniC avec technologies objets
- Examen (40%) : 1h30 avec documents
- Urgent : Constituer les quadrinômes et binômes associés
- Alternative 1 : Pas de confinement
  - Travaux Dirigées en présenciel
  - Travaux Pratiques à décider
- Alternative 2 : Confinement
  - Combinaison TD/TP à distance

Plan du cours

- Introduction
  - Rappels : Modélisation, Automates et Graphes, GLS
  - Architecture générale
  - Formes de sémantique
- Interprétation
  - Sémantique opérationnelle
  - Sémantique axiomatique
- Compilation
  - Table des Symboles, Arbre abstrait
  - Typage
  - Modèle mémoire, Génération de code
  - Sémantique translationnelle, dénotationnelle
- Vérification de correction

Plan du cours

- Introduction
  - Rappels : Modélisation, Automates et Graphes, GLS
  - Architecture générale
  - Formes de sémantique
- Interprétation
  - Sémantique opérationnelle
  - Sémantique axiomatique
- Compilation
  - Table des Symboles, Arbre abstrait
  - Typage
  - Modèle mémoire, Génération de code
  - Sémantique translationnelle, dénotationnelle
- Vérification de correction

Plan du cours

- Introduction
  - Rappels : Modélisation, Automates et Graphes, GLS
  - Architecture générale
  - Formes de sémantique
- Interprétation
  - Sémantique opérationnelle
  - Sémantique axiomatique
- Compilation
  - Table des Symboles, Arbre abstrait
  - Typage
  - Modèle mémoire, Génération de code
  - Sémantique translationnelle, dénotationnelle
- Vérification de correction

Plan du cours

- Introduction
  - Rappels : Modélisation, Automates et Graphes, GLS
  - Architecture générale
  - Formes de sémantique
- Interprétation
  - Sémantique opérationnelle
  - Sémantique axiomatique
- Compilation
  - Table des Symboles, Arbre abstrait
  - Typage
  - Modèle mémoire, Génération de code
  - Sémantique translationnelle, dénotationnelle
- Vérification de correction

Rappels

- Modélisation :
  - Structure algébrique des langages
  - Spécification des langages :
    - Expressions régulières,
    - Grammaire (règles de production, EBNF, Conway)
- Automates et Théorie des Langages
  - Automates, Automates à piles, Analyseur descendant récursif
  - Générateurs d'analyseurs lexicaux et syntaxiques
- Ingénierie Dirigée par les Modèles
  - Métamodèles :
    - Représentation abstraite du langage (MOF),
    - Règles de bonne formation (OCL)
  - Syntaxe concrète texte : Xtext

Rappels

- Modélisation :
  - Structure algébrique des langages
  - Spécification des langages :
    - Expressions régulières,
    - Grammaire (règles de production, EBNF, Conway)
- **Automates et Théorie des Langages**
  - **Automates, Automates à piles, Analyseur descendant récursif**
  - **Générateurs d'analyseurs lexicaux et syntaxiques**
- Ingénierie Dirigée par les Modèles
  - Métamodèles :
    - Représentation abstraite du langage (MOF),
    - Règles de bonne formation (OCL)
  - Syntaxe concrète texte : Xtext

Rappels

- Modélisation :
  - Structure algébrique des langages
  - Spécification des langages :
    - Expressions régulières,
    - Grammaire (règles de production, EBNF, Conway)
- Automates et Théorie des Langages
  - Automates, Automates à piles, Analyseur descendant récursif
  - Générateurs d'analyseurs lexicaux et syntaxiques
- Ingénierie Dirigée par les Modèles
  - Métamodèles :
    - Représentation abstraite du langage (MOF),
    - Règles de bonne formation (OCL)
  - Syntaxe concrète texte : Xtext

Principes essentiels

Communication = Echange d'informations

Besoins :
- Représenter les informations possibles
- Reconnaitre une information
- Exploiter une information

Organisation stratifiée : information structurée

Informatique : Science du traitement de l'information

Computer science : Science de la « machine à calculer »

Essentiel :
- Description et manipulation de l'information (langage),
- Traitement d'une information quelconque,
- Traitement d'une manipulation quelconque

D'où :
- Description formelle du langage
- Génération automatique des outils de manipulation

Références bibliographiques

- Hopcroft, Ullman, Introduction to automata theory, languages and computation, Addison-Wesley, 1979.
- Stern, Fondements mathématiques de l'informatique, McGraw-Hill, 1990.
- Carton, Langages formels, calculabilité et complexité, Vuibert, 2008.
- Aho, Sethi, Ullman, Compilateurs : Principes, Techniques et Outils, InterEditions, 1989.
- Fisher, Leblanc, Crafting a compiler in ADA/in C, Benjamin Cummings, 1991.
- Wilhem, Maurer, Les compilateurs : Théorie, construction, génération, Masson, 1994.
- Appel, Modern Compiler Implementation in Java/ML/C, Cambridge University Press, 1998.
- Winskel, The formal semantics of programming languages : An introduction, MIT Press, 1993.
- Lämmel, Software Languages : Syntax, Semantics and Metaprogramming, Springer (under review), 2017.

Exemple : fichier /etc/hosts

- Fichier tel qu'il est affiché :

```
#␣Ceci␣est␣un␣commentaire

127.0.0.1       →hal9000␣localhost

#␣En␣voici␣un␣autre

147.127.18.144       →phoenix.enseeiht.fr
```

- Informations brutes : caractères

```
0000000    #  sp   C   e   c   i  sp   e   s   t  sp   u   n  sp   c   o
0000020    m   m   e   n   t   a   i   r   e  nl  nl   1   2   7   .   0
0000040    .   0   .   1  ht   h   a   l   9   0   0   0  sp   l   o   c
0000060    a   l   h   o   s   t  nl  nl   #  sp   E   n  sp   v   o   i
0000100    c   i  sp   u   n  sp   a   u   t   r   e  nl  nl   1   4   7
0000120    .   1   2   7   .   1   8   .   1   4   4  ht   p   h   o   e
0000140    n   i   x   .   e   n   s   e   e   i   h   t   .   f   r  nl
0000160
```

Analyse lexicale

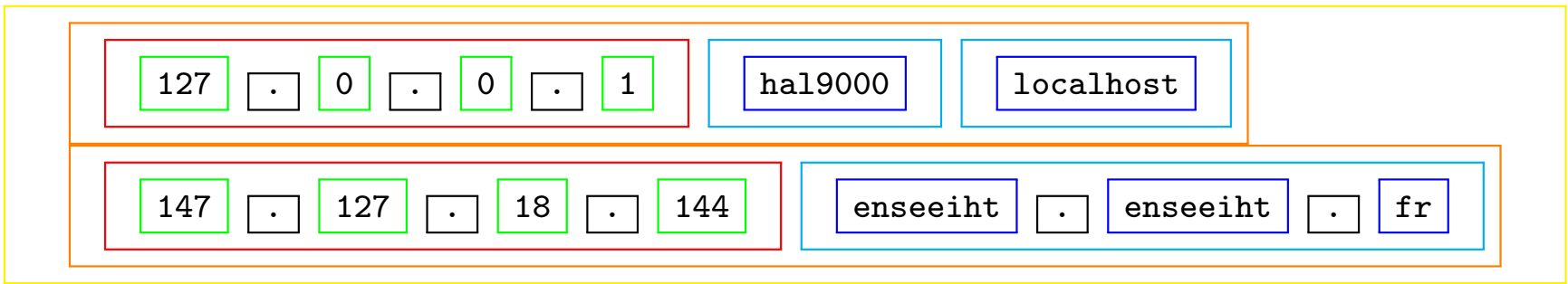
- Informations élémentaires : commentaire, nombre, identificateur, . (unités lexicales)
- Résultat de l'analyse lexicale :

`# Ceci est un commentaire` `127` `.` `0` `.` `0` `.` `1`
`hal9000` `localhost` `# En voici un autre` `147` `.`
`127` `.` `18` `.` `144` `enseeiht` `.` `enseeiht` `.` `fr`

- Spécification des unités lexicales : Expressions régulières
  - Commentaire : $\#[\hat{}\backslash n]^{\star}\backslash n$
  - Nombre : $[0-9]^{+}$
  - Identificateur : $[a-bA-B][a-bA-B0-9]^{\star}$

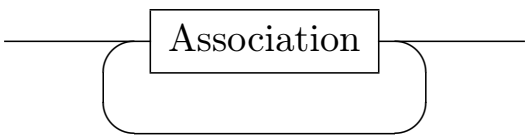
Analyse syntaxique

- Informations structurées (unités syntaxiques) :
  - Premier niveau : adresse IP, nom qualifié
  - Deuxième niveau : assocation
  - Troisième niveau : document

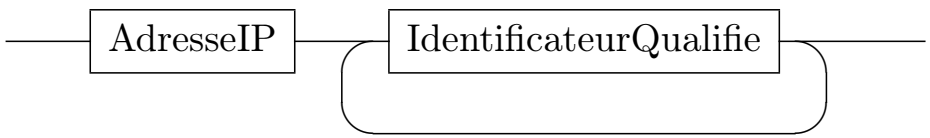


- Spécification des unités syntaxiques : Grammaires (notation de Conway)

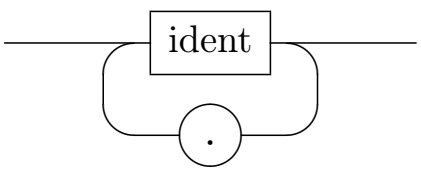
Document



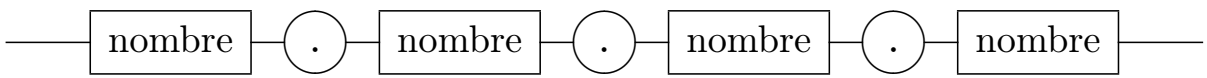
Association



IdentificateurQualifie

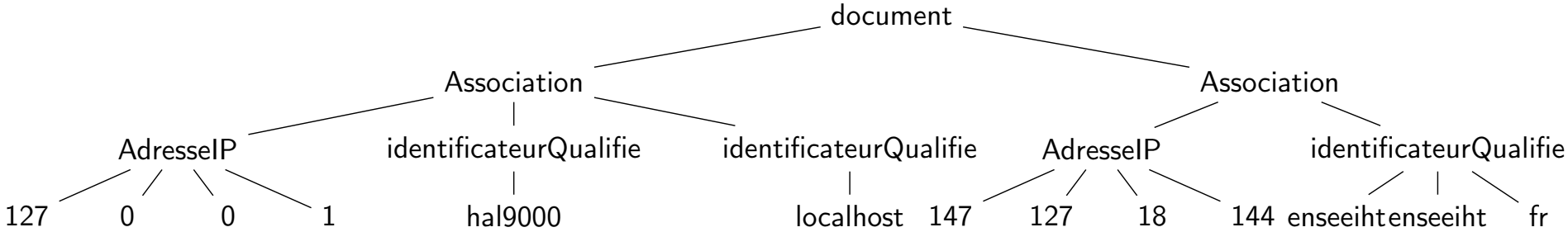


AdresseIP

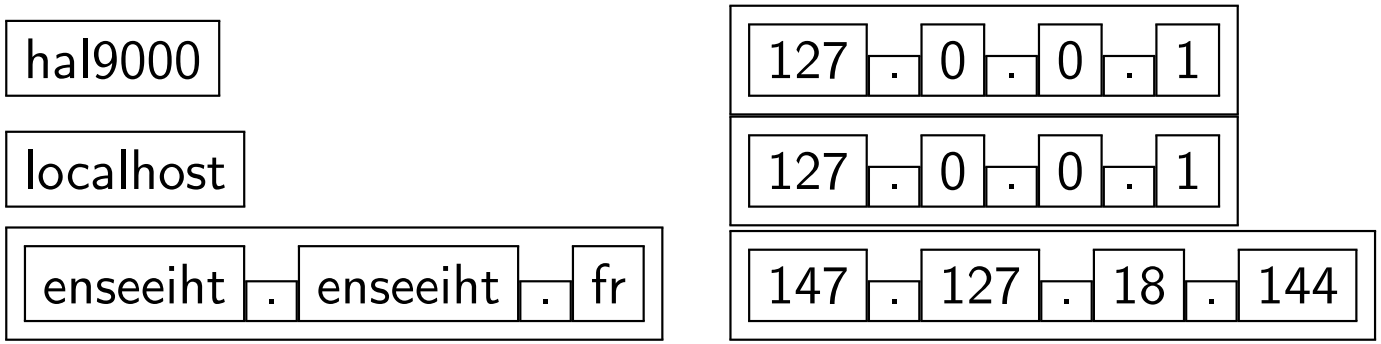


Analyse sémantique

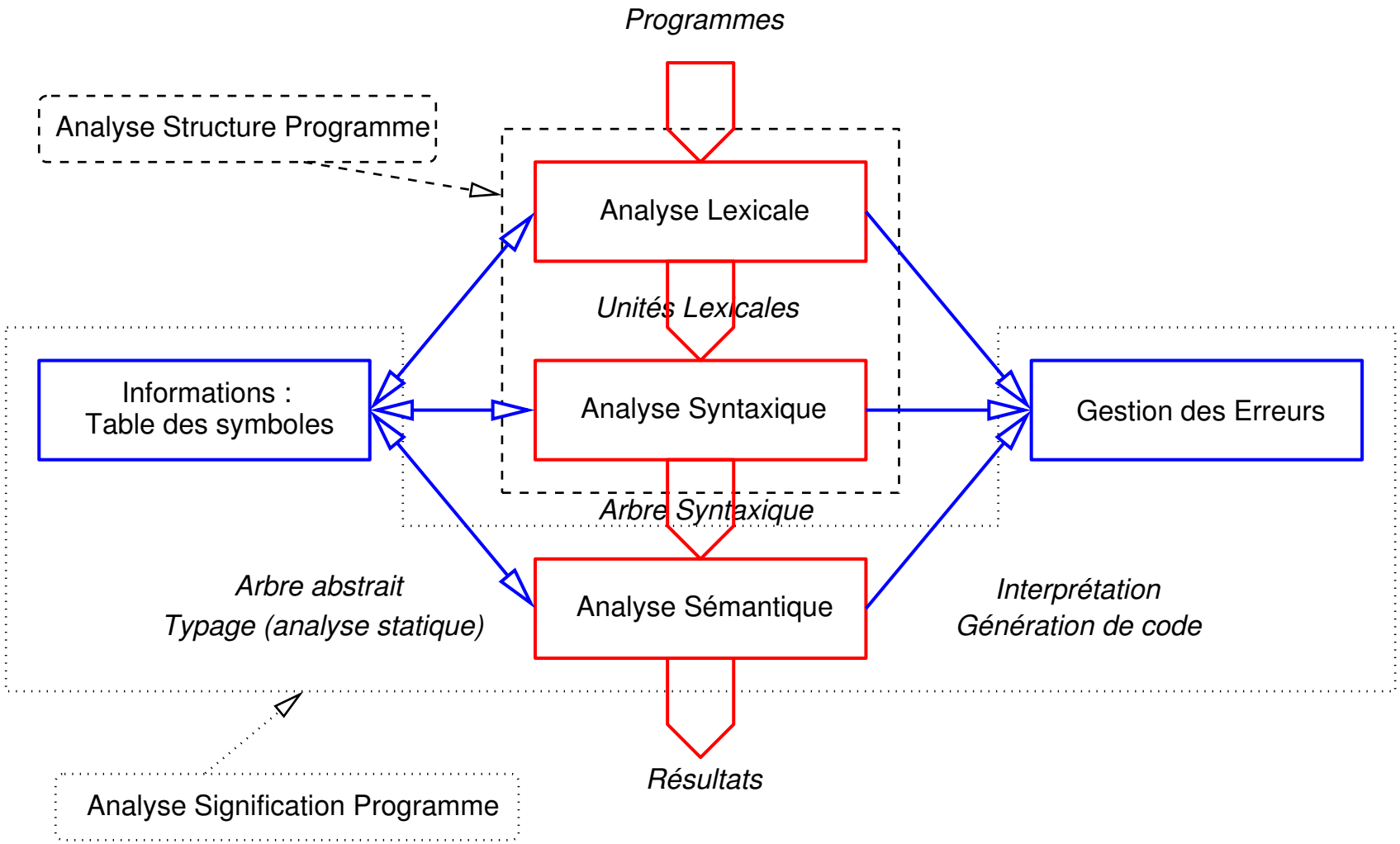
- Structure arborescente associée :



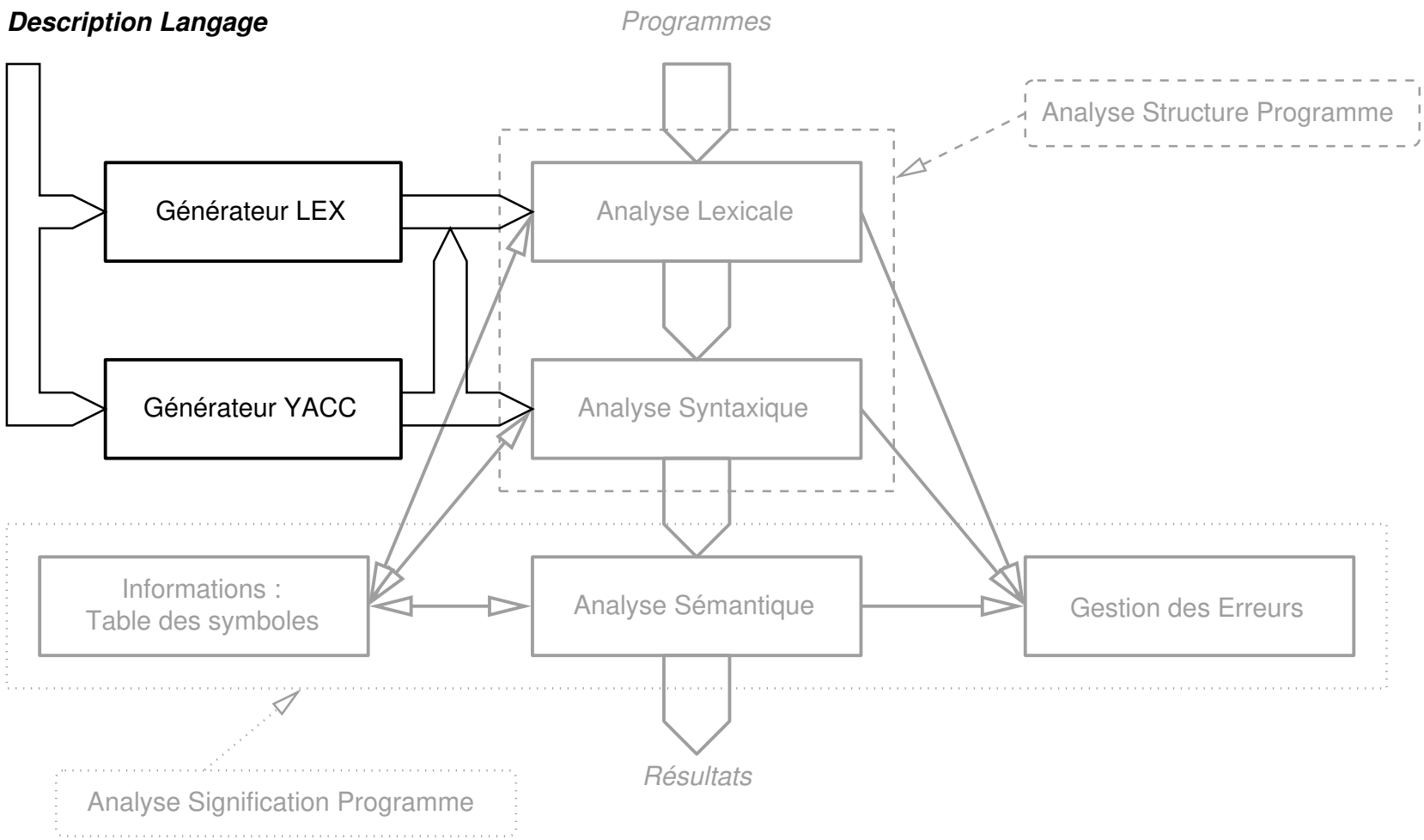
- Exploitation des informations : association nom qualifié/adresse IP (unités sémantiques)



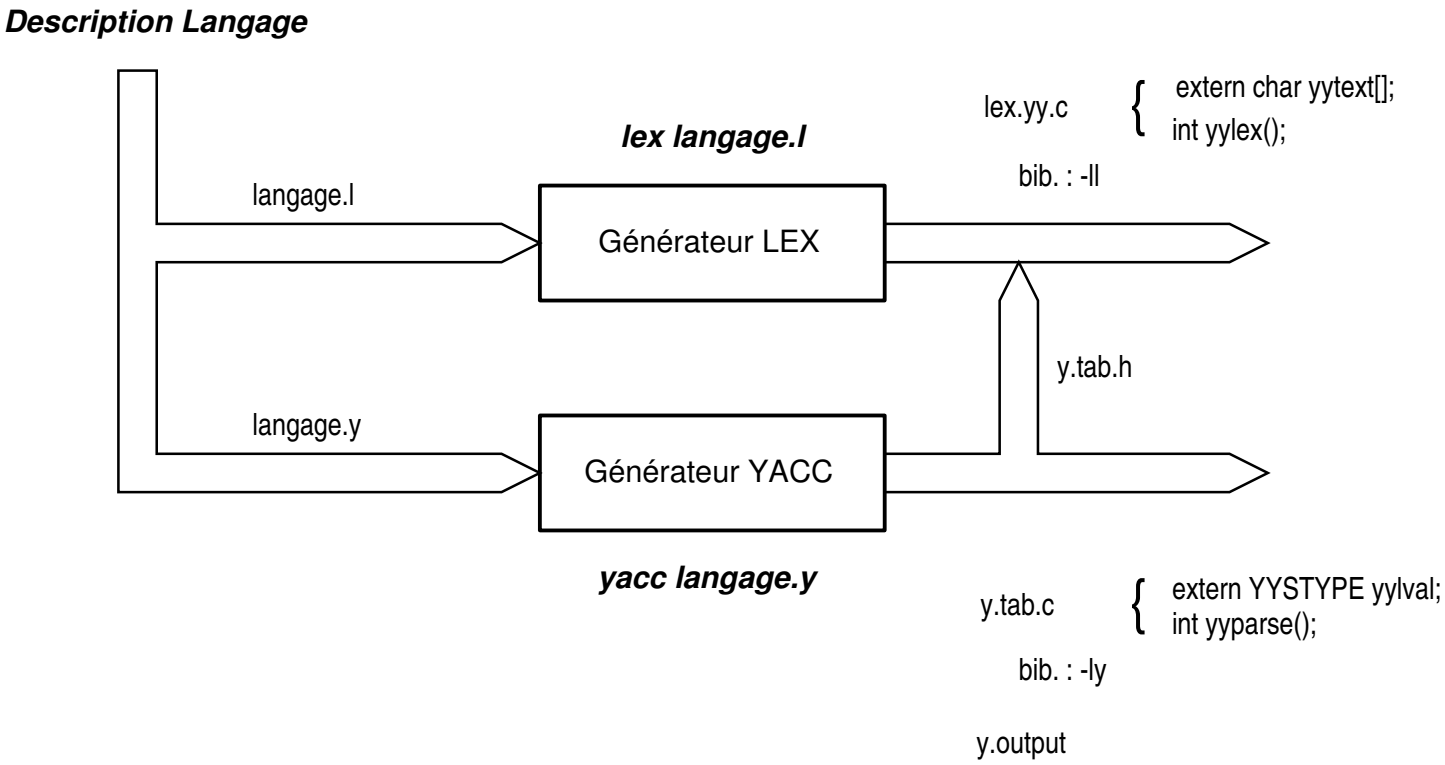
Structure d'un outil



Exemple lex et yacc



Exemple lex et yacc



Définitions

- Caractère/Symbole : Unité élémentaire d'information
- Unité lexicale (lexème, mot) : Séquence de caractères
- Unité syntaxique (arbre syntaxique, syntème, phrase) : Arbre d'unités lexicales
- Unité sémantique : diverses (arbre abstrait, table des symboles, type, code généré, résultat évaluation, ...)

Comment organiser les informations ?

- Objectif : Exploitation des informations
- Règle : Choisir le bon niveau de précision
- Unité lexicale : Bloc élémentaire d'information pertinente
- Unité syntaxique : Elément structurant de l'information

Sémantique formelle des langages

- Objectif : Modélisation la sémantique avec des outils mathématique
- Atteindre la qualité de la modélisation de la syntaxe
- Etudier la cohérence et la complétude
- Prouver la correction des outils
- Générer automatiquement les outils
- Différentes formes :
  - Sémantique opérationnelle : Mécanisme d'exécution des programmes
  - Sémantique axiomatique : Mécanisme de vérification des programmes
  - Sémantique translationnelle : Traduction vers un autre langage équipé d'une sémantique formelle
  - Sémantique dénotationnelle : Traduction vers un formalisme mathématique
- Validation des sémantique par étude équivalence entre formes

miniML

- Expressions : sans effets de bord, similaire dans tous les langages
- Partie fonctionnelle : sans effets de bord
- Partie Impérative : effets de bord, y compris dans les expressions et la partie fonctionnelle

miniML

- Expressions : sans effets de bord, similaire dans tous les langages
- Partie fonctionnelle : sans effets de bord
- Partie Impérative : effets de bord, y compris dans les expressions et la partie fonctionnelle

miniML

- Expressions : sans effets de bord, similaire dans tous les langages
- Partie fonctionnelle : sans effets de bord
- Partie Impérative : effets de bord, y compris dans les expressions et la partie fonctionnelle

miniML

- Expressions :

$$
\begin{array}{lcl}
\textit{Expr} & \rightarrow & \textit{Ident} \\
 & | & \textit{Const} \\
 & | & \textit{Expr Binaire Expr} \\
 & | & \textit{Unaire Expr} \\
 & | & (\ \textit{Expr}\ ) \\
 & & \\
\textit{Const} & \rightarrow & \textit{entier} \mid \textit{booleen} \\
 & & \\
\textit{Unaire} & \rightarrow & - \\
 & & \\
\textit{Binaire} & \rightarrow & + \mid - \mid * \mid / \mid \% \mid \& \mid \mid \\
 & | & == \mid != \mid < \mid <= \mid > \mid >=
\end{array}
$$

miniML

- Partie Fonctionnelle :

$$
\begin{array}{rcl}
\mathit{Expr} & \rightarrow & \cdots \\
& | & \texttt{let}\ \mathit{Ident}\ \texttt{=}\ \mathit{Expr}\ \texttt{in}\ \mathit{Expr} \\
& | & \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Expr}\ \texttt{else}\ \mathit{Expr} \\
& | & \texttt{fun}\ \mathit{Ident}\ \texttt{->}\ \mathit{Expr} \\
& | & \texttt{(}\mathit{Expr}\ \texttt{)}\ \mathit{Expr} \\
& | & \texttt{let rec}\ \mathit{Ident}\ \texttt{=}\ \mathit{Expr}\ \texttt{in}\ \mathit{Expr}
\end{array}
$$

- Partie Impératives :

$$
\begin{array}{rcl}
\mathit{Expr} & \rightarrow & \texttt{ref}\ \mathit{Expr} \\
& | & \texttt{!}\ \mathit{Expr} \\
& | & \mathit{Expr}\ \texttt{:=}\ \mathit{Expr} \\
& | & \mathit{Expr}\ \texttt{;}\ \mathit{Expr} \\
& | & \texttt{while}\ \mathit{Expr}\ \texttt{do}\ \mathit{Expr}\ \texttt{done}
\end{array}
$$

Interprétation : Principes généraux

- Programme qui exécute un programme (émulateur, machine virtuelle, ...)
- Langage hôte/support : Langage de programmation de l'interprète
- Représenter le programme comme une donnée
- Représenter l'exécution comme des données et algorithmes
  - Résultats de l'exécution (dont intermédiaires)
  - Ne pas oublier les erreurs d'exécution (résultats possibles)
  - Pour chaque construction exécutable du langage, identifier :
    1. Variantes selon résultats intermédiaires
    2. Étapes dans chaque variante
    3. Contraintes entre les étapes

Interprétation : Principes généraux

- Programme qui exécute un programme (émulateur, machine virtuelle, ...)
- Langage hôte/support : Langage de programmation de l'interprète
- Représenter le programme comme une donnée
- Représenter l'exécution comme des données et algorithmes
  - Résultats de l'exécution (dont intermédiaires)
  - Ne pas oublier les erreurs d'exécution (résultats possibles)
  - Pour chaque construction exécutable du langage, identifier :
    1. Variantes selon résultats intermédiaires
    2. Étapes dans chaque variante
    3. Contraintes entre les étapes

Interprétation : Principes généraux

- Programme qui exécute un programme (émulateur, machine virtuelle, ...)
- Langage hôte/support : Langage de programmation de l'interprète
- Représenter le programme comme une donnée
- Représenter l'exécution comme des données et algorithmes
  - Résultats de l'exécution (dont intermédiaires)
  - Ne pas oublier les erreurs d'exécution (résultats possibles)
  - Pour chaque construction exécutable du langage, identifier :
    1. Variantes selon résultats intermédiaires
    2. Étapes dans chaque variante
    3. Contraintes entre les étapes

Représentation des programmes et de l'exécution

- Programmes :
  - Arbres abstraits : Abstraction de l'arbre de dérivation (arbre syntaxique)
  - Structure de graphe (relation définition/utilisation)
    - Approche objet : Métamodèles
    - Approche fonctionnelle : Structure d'arbres + Tables des symboles
- Exécution :
  - Valeurs : Exploiter les types de base du langage hôte (booléen, entier, flottant, caractère, chaîne de caractère, ...)
  - Déclarations : Utilisation d'un dictionnaire (table des symboles)
  - Mémoire : Adresses et Espace de données associé

Représentation des programmes et de l'exécution

- Programmes :
  - Arbres abstraits : Abstraction de l'arbre de dérivation (arbre syntaxique)
  - Structure de graphe (relation définition/utilisation)
    - Approche objet : Métamodèles
    - Approche fonctionnelle : Structure d'arbres + Tables des symboles
- Exécution :
  - Valeurs : Exploiter les types de base du langage hôte (booléen, entier, flottant, caractère, chaîne de caractère, ...)
  - Déclarations : Utilisation d'un dictionnaire (table des symboles)
  - Mémoire : Adresses et Espace de données associé

Application à miniML

- Arbre abstrait : voir vidéo séparée
- Valeurs :

$$\begin{array}{lll} \mathit{Valeur} & \rightarrow & \mathit{Const} \\ & | & \bot \end{array}$$

- Algorithme d'exécution : voir vidéo séparée

Sémantique Opérationnelle

- Objectif : Décrire formellement les mécanismes d'exécution des programmes d'un langage
- Principe :
  - Exploiter la syntaxe du langage
  - Décrire l'exécution comme une transformation des programmes
- Notation : Règles de déduction
  - Soient $J_1, \ldots J_n$ et J des jugements :

	Notation	Signification
Déduction	$\dfrac{J_1 \quad J_n}{J}$	si J_1 et ...et J_n sont valides alors J est valide
Axiome	$\dfrac{}{J}$	J est valide

- Jugement d'exécution à grand pas : $\gamma \vdash e \Rightarrow v$
  - γ : environnement (association *Ident* / *Valeur*)
  - e : expression (*Expr*)
  - v : valeur (*Valeur*)
- Partie haute : Étapes intermédiaires (appels récursifs dans interpréte miniML)
- Partie basse : Construction traitée par la règle

miniML : Constantes et Accès identificateur

- Constante : Valeur ne change pas

$$\frac{}{\gamma \vdash \mathit{entier} \Rightarrow \mathit{entier}}$$

$$\frac{}{\gamma \vdash \mathit{booleen} \Rightarrow \mathit{booleen}}$$

- Identificateur : Accès à l'environnement
  - Présent : Transmission valeur associée

$$\frac{x \in \gamma \quad \gamma(x) = v}{\gamma \vdash x \Rightarrow v}$$

  - Absent : Cas d'erreur

$$\frac{x \notin \gamma}{\gamma \vdash x \Rightarrow \bot_{\mathit{undef}}}$$

miniML : Opérateur Unaire

- Étape préliminaire : Calcul du paramètre
- Variante 1 : Résultat correct du bon type

$$\frac{\gamma \vdash e \Rightarrow v \quad v \neq \bot \quad v \in dom\, op \quad v' = op\, v}{\gamma \vdash op\ e \Rightarrow v'}$$

- Variante 2 : Résultat erroné

$$\frac{\gamma \vdash e \Rightarrow v \quad v = \bot_c}{\gamma \vdash op\ e \Rightarrow \bot_c}$$

- Variante 3 : Résultat correct du mauvais type

$$\frac{\gamma \vdash e \Rightarrow v \quad v \neq \bot \quad v \notin dom\, op}{\gamma \vdash op\ e \Rightarrow \bot_{type}}$$

miniML : Opérateur Binaire

- Étapes préliminaires : Calcul des paramètres
- Question : Y a t'il un ordre particulier ?
- En absence d'effets de bord : Non, concurrence/parallélisme possible
- Variante 1 : Résultats corrects du bon type

$$\frac{\begin{array}{ll}\gamma \vdash e_1 \Rightarrow v_1 & v_1 \neq \bot \\ \gamma \vdash e_2 \Rightarrow v_2 & v_2 \neq \bot\end{array} \quad v_1 \times v_2 \in dom\, op \quad v = v_1\ op\ v_2}{\gamma \vdash e_1\ op\ e_2 \Rightarrow v}$$

- Variante 2 : Résultat(s) erroné(s)

$$\frac{\gamma \vdash e_1 \Rightarrow v_1 \quad v_1 = \bot_c}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_c} \qquad \frac{\gamma \vdash e_2 \Rightarrow v_2 \quad v_2 = \bot_c}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_c}$$

  - Que se passe t'il si deux erreurs se produisent de natures différentes ?
  - Définir une règle qui explicite ce cas

- Variante 3 : Résultat correct du mauvais type

$$\frac{\gamma \vdash e_1 \Rightarrow v_1 \quad \gamma \vdash e_2 \Rightarrow v_2 \quad v_1 \neq \bot \quad v_2 \neq \bot \quad v_1 \times v_2 \notin dom\, op}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_{type}}$$

miniML : Opérateur Binaire Droite à Gauche

- Imposons un ordre d'évaluation de droite à gauche (celui de OCaML)
- Variante 1 : Résultats corrects du bon type

$$\frac{\gamma \vdash e_2 \Rightarrow v_2 \quad v_2 \neq \bot \qquad \gamma \vdash e_1 \Rightarrow v_1 \quad v_1 \neq \bot \qquad v_1 \times v_2 \in dom\, op \quad v = v_1\ op\ v_2}{\gamma \vdash e_1\ op\ e_2 \Rightarrow v}$$

Attention : Cette règle n'impose pas d'ordre

- Variante 2 : Résultat(s) erroné(s)

$$\frac{\gamma \vdash e_2 \Rightarrow v_2 \quad v_2 = \bot_c}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_c} \qquad \frac{\gamma \vdash e_2 \Rightarrow v_2 \quad v_2 \neq \bot \qquad \gamma \vdash e_1 \Rightarrow v_1 \quad v_1 = \bot_c}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_c}$$

  - 2 erreurs ne peuvent plus se produire en même temps

- Variante 3 : Résultat correct du mauvais type

$$\frac{\gamma \vdash e_1 \Rightarrow v_1 \quad \gamma \vdash e_2 \Rightarrow v_2 \quad v_1 \neq \bot \quad v_2 \neq \bot \quad v_1 \times v_2 \notin dom\, op}{\gamma \vdash e_1\ op\ e_2 \Rightarrow \bot_{type}}$$