

# LPP-v6 – Notes Développeur

P.Raymond

## Table des matières

<b>1</b>	<b>Ligne de commande et analyse syntaxique</b>	<b>1</b>
1.1	Fichiers acceptés . . . . .	1
1.2	Analyse syntaxique . . . . .	1
<b>2</b>	<b>Analyse des déclarations</b>	<b>2</b>
<b>3</b>	<b>Analyse référentielle et typage de surface</b>	<b>2</b>

## 1 Ligne de commande et analyse syntaxique

### 1.1 Fichiers acceptés

Les fichiers supportés sont :

- Les fichiers `foo.lus` à la syntaxe Lustre-v6 comportant un ensemble de déclarations de packages et de modules.
- Les fichiers `foo.lus` contenant uniquement un corps de package à la syntaxe v6 ; un tel fichier est considéré comme la déclaration implicite d'un package dont le nom est `foo`, et dont tous les items sont exportés (`provided`).

Restrictions/à faire :

- pour l'instant on prend un seul fichier, mais ça sera facilement adaptable

### 1.2 Analyse syntaxique

Elle produit un ensemble de déclarations de packages et de modèles dans une structure syntaxique abstraite (module **Syntaxe**), qui constitue le **source brut**.

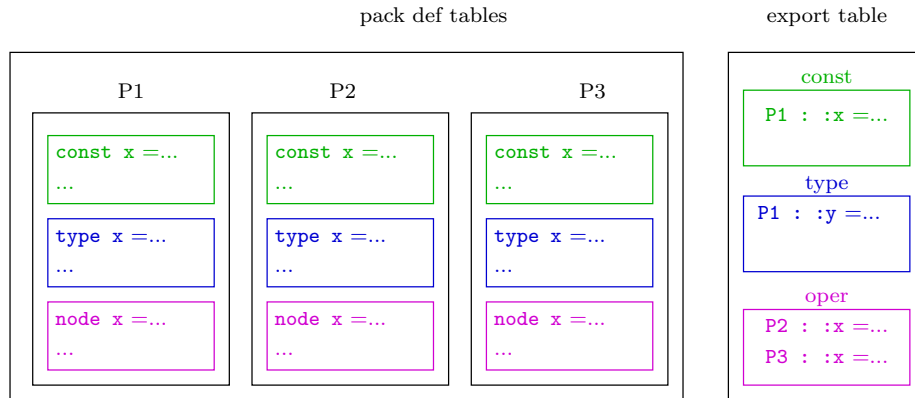


FIG. 1 – La table des sources

## 2 Analyse des déclarations

C'est la première phase de compilation qui consiste à vérifier la cohérence des déclarations modulo l'organisation en "name-space" (les packages).

Cette phase produit une nouvelle version du source (module **SrcTab**) :

- unicité des déclarations de modèles et de packages
- instantiation (purement syntaxique) des modèles
- identification des items exportés et de leurs définitions abstraites (e.g. un type fourni par un package est présenté abstrait pour les utilisations extérieures).

Au final on a une table de sources bien organisée (**SrcTab**) mais qui renvoie encore à des infos purement syntaxiques (**Syntaxe**), voir figure 1 :

- les trois tables d'export (const, type, oper) sont indexées par des identificateurs absolus (pack+nom) et pointent sur des définitions éventuellement abstraites : c'est la seule vision qu'on en a de l'extérieur, et l'analyse doit être faite uniquement modulo cette info.
- la table des définitions de pack, chacun comportant ses tables d'items (const, type, oper) ; les identifiants sont relatifs et les définitions concrètes.

## 3 Analyse référentielle et typage de surface

À cause de la récursivité statique (notamment) il est pratiquement impossible de faire du typage "fin" (et a fortiori de la compilation) modulaire. Le typage fin et la compilation proprement dite sera donc faite à la demande : on compile tout ce qui est nécessaire à l'exécution d'un nœud principal désigné.

Cependant, on effectue tout de même une analyse préliminaire indépendante d'un main particulier :

- chaque référence à un ident pointe bien sur un item de nature attendue,
- 

PAS CLAIR : A revoir ...