

Chapitre 1

Conception du lexeur

I Conventions de notation

I.1 Types de *Token*

Dans cette partie, on désignera par i (avec $i \in \mathbb{N}^*$) le numéro de la ligne où le *Token* a été lu.

1. Retours chariot & délimitations de blocs
 - Token indiquant un saut de ligne : **Token(NEWLINE, i)** ;
 - Token signalant le début d'une indentation : **Token(BEGIN, i)** ;
 - Token marquant la fin d'une indentation : **Token(END, i)** ;
 - Token indiquant la fin du fichier source : **Token(EOF, i)**.
2. Identificateur, représentant une variable, une fonction ou un paramètre : **Token(IDENTIFIER, <identificateur>, i)**.
3. Mot-clé : **Token(KEYWORD, <mot clé>, i)**. On rappelle que les mots-clé, réservés par le langage et ne pouvant pas être utilisés comme identifiants, sont les suivants : *and, or, if, else, for, in, not, True, False, print, def, return* et *None*.
4. Opérateurs binaires (utilisés pour faire des opérations binaires) :
 - **Token(PLUS, i)** ;
 - **Token(MINUS, i)** ;
 - **Token(MULTIPLY, i)** ;

- **Token**(**FLOOR_DIVIDE**, i);
 - **Token**(**MODULO**, i)¹;
 - **Token**(**LESS_EQUAL**, i);
 - **Token**(**GREATER_EQUAL**, i);
 - **Token**(**LESS**, i);
 - **Token**(**GREATER**, i);
 - **Token**(**NOT_EQUAL**, i);
 - **Token**(**EQUAL**, i);
 - **Token**(**AND**, i);
 - **Token**(**OR**, i).
5. Opérateurs unaires (c'est-à-dire utilisés pour faire des opérations unaires) : **Token**(**UNARY_MINUS**, i) et **Token**(**NOT**, i).
6. Opérateur d'assignation ("**:=**") : **Token**(**ASSIGNMENT**, i).
7. Types pris en charge par le langage : **Token**(**INTEGER**, **<int>**, i) et **Token**(**STRING**, **<str>**, i).
8. Délimiteurs :
- Crochet ouvrant ("**[**") : **Token**(**LBRACKET**, i);
 - Crochet fermant ("**]**") : **Token**(**RBRACKET**, i);
 - Parenthèse ouvrante ("**(**") : **Token**(**LPAREN**, i);
 - Parenthèse fermante ("**)**") : **Token**(**RPAREN**, i);
 - Virgule : **Token**(**COMMA**, i);
 - Point-virgule : **Token**(**COLON**, i).

I.2 Ensembles

Pour alléger les notations dans les automates qui suivent, on propose les notations suivantes :

- α désignera tout caractère alphanumérique (minuscule ou majuscule) :

$$\alpha = [a - z] \mid [A - Z].$$

1. On considère l'opérateur "*modulo*" comme étant l'opérateur binaire qui associe à deux entiers naturels le reste de la division euclidienne du premier par le second.

- \mathcal{D} désignera tout chiffre :

$$\mathcal{D} = [0 - 9].$$

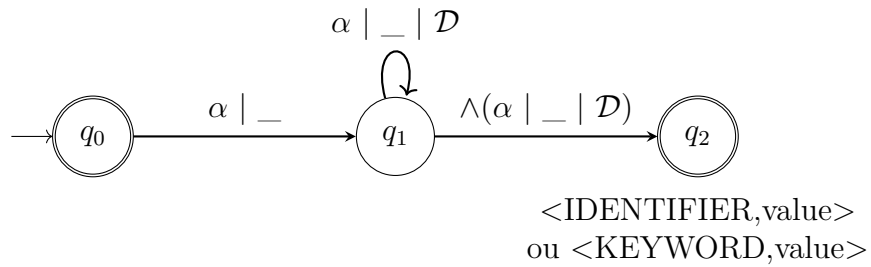
- \mathcal{O} désignera tout opérateur formé d'un seul symbole :

$$\mathcal{O} = [+ \mid - \mid * \mid \%].$$

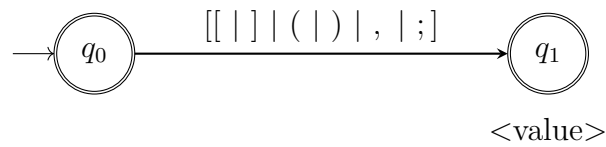
- \mathcal{A} désignera n'importe quel symbole reconnu par la grammaire.

II Sous-automates

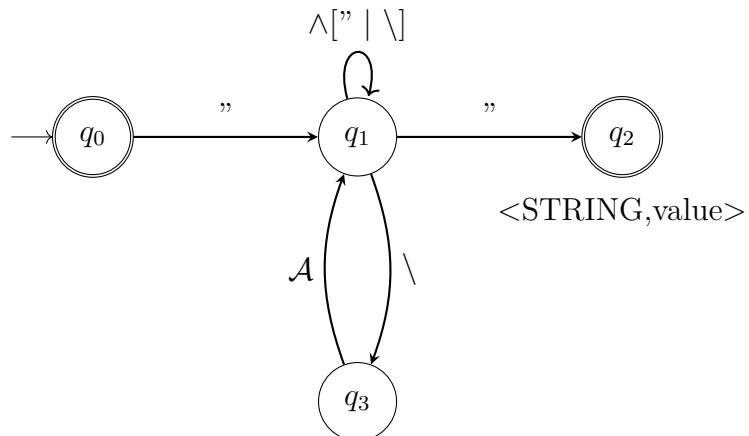
II.1 Identificateurs & mots-clés



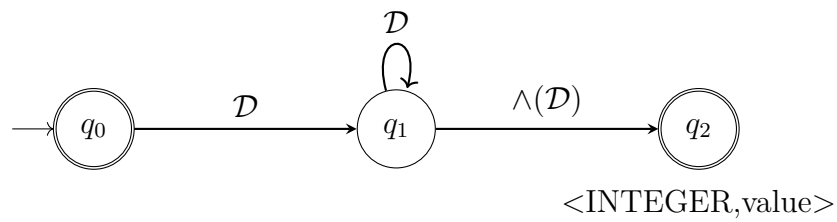
II.2 Délimiteurs



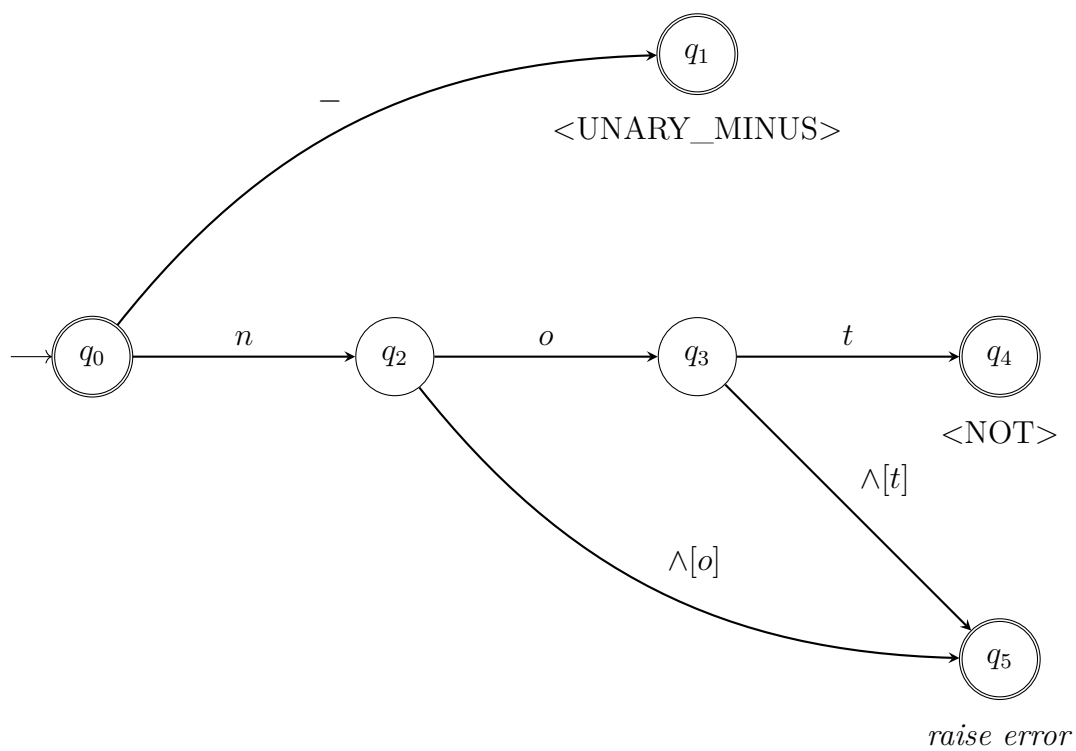
II.3 Chaînes de caractère



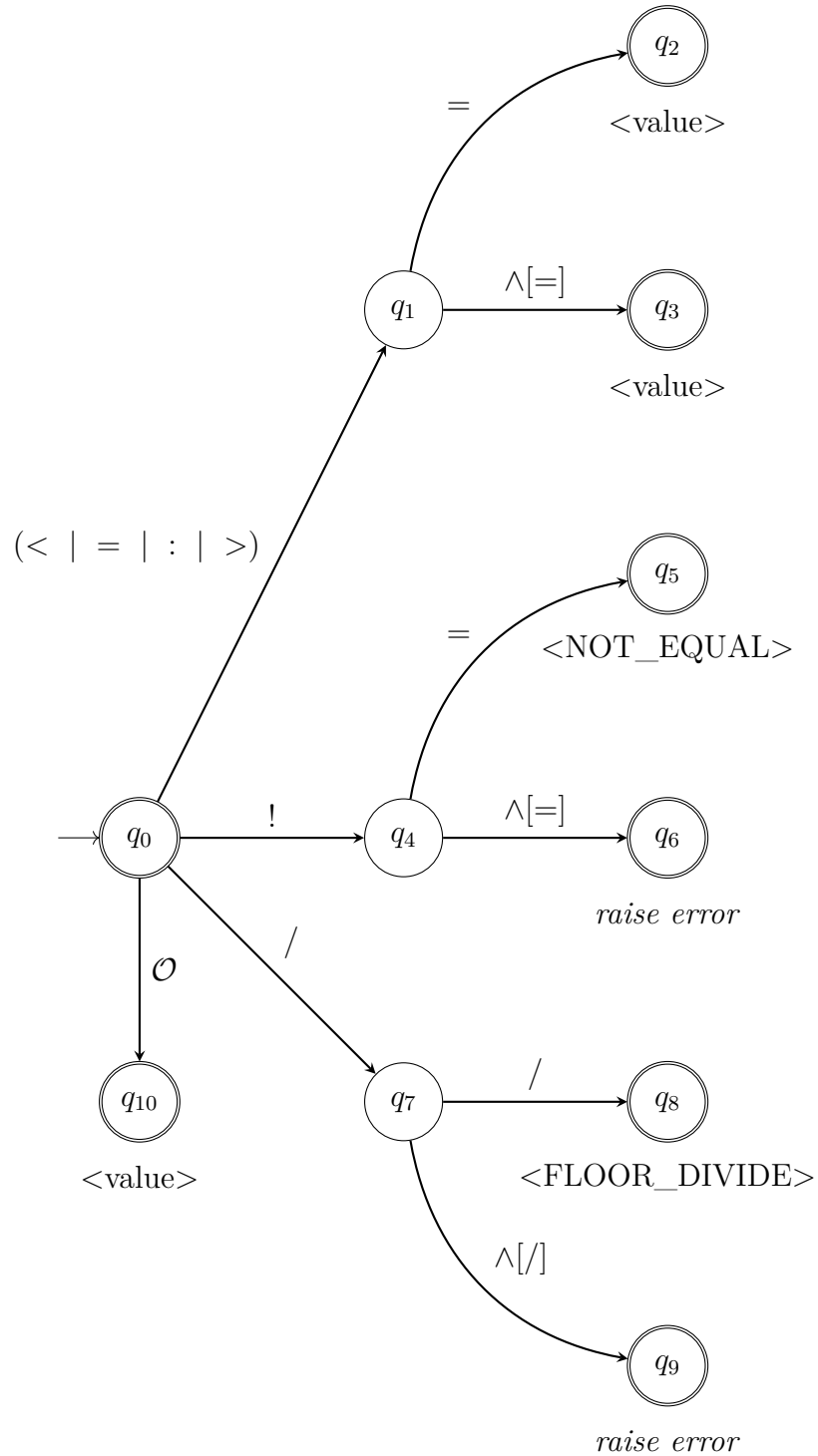
II.4 Entiers



II.5 Opérateurs unaires

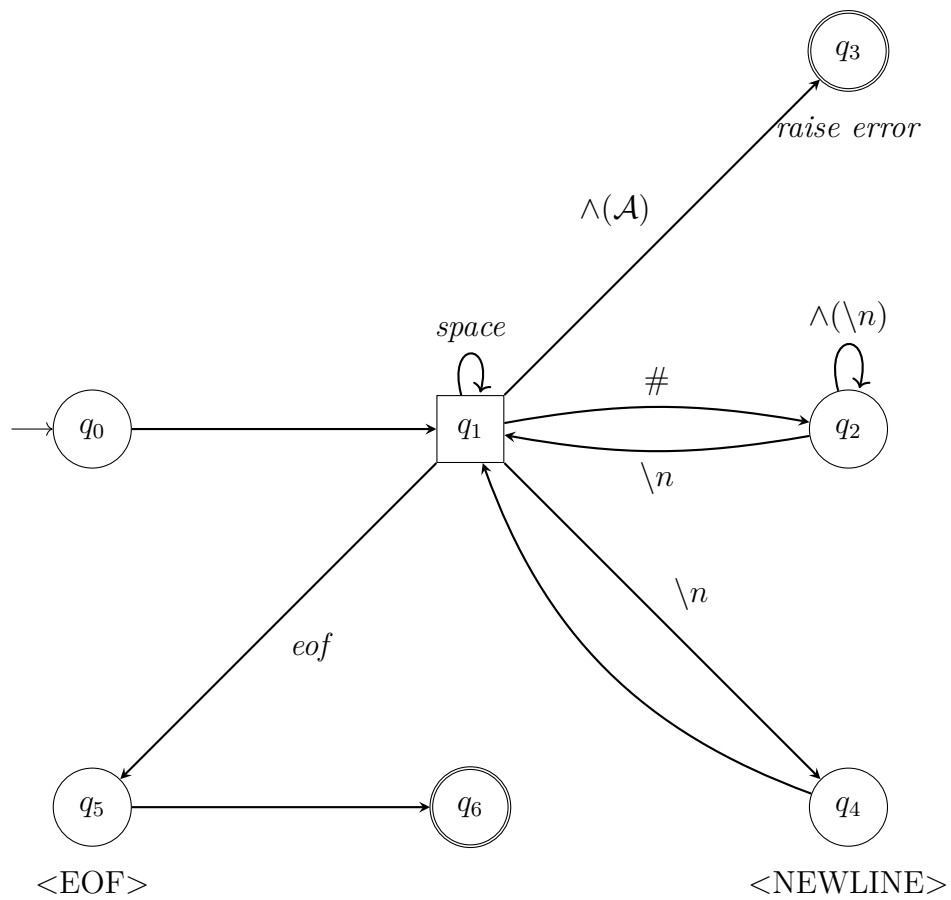


II.6 Opérateurs binaires



III Automate fini déterministe

Dans cette section, on va détailler l'automate fini déterministe correspondant au lexeur du "Mini-Python". Ce dernier implémentera les sous-automates détaillés précédemment, qui seront tous appelés simultanément à l'état 1 (représenté rectangulaire) de l'automate ci-dessous. Ainsi, n'importe quelle unité lexicale sera reconnue et verra son *Token* ajouté tant qu'aucun des autres symboles "`\n`", "`#`" ou un caractère non-accepté par le langage.



Chapitre 2

Conception du parseur

I Grammaire LL(2)

On propose la grammaire suivante, obtenue à partir de celle-ci dessus, dont les règles figurent en annexe (document "grammar.pdf") :

$$G = (\mathcal{N}, \mathcal{T}, \rightarrow, S), \quad \text{où}$$

$$\mathcal{N} = \{S, S', S'', A, B, B', C, C', D, D', E, E', E'', E''', F, G, G', H, H', I, I'\}$$

et $\mathcal{T} = \alpha \cup \mathcal{D} \cup \{+; -; *; /; \% ; (; [;) ;] ; \backslash, ", :, !, =\}.$

II Élagage de l'arbre

II.1 Suppression des éléments lexicaux non-essentiels

De nombreux *tokens* dans l'arbre à sa sortie du parseur sont purement syntaxiques et n'affectent pas réellement la logique sous-jacente. Il s'agit des *tokens* du types "**Newline**". On va donc les éliminer en premier.