INSA Lyon

Département Informatique

Grammaires et Langages

Dossier de conception

- Développement d'un interpréteur pour le langage lutin -

31 mars 2016

Auteurs
Benoît Chabod
Baptiste Donet
Pierre Jarsaillon
Romain Mathonat
Nicolas Nativel
Jonathan Taws

Chef de projet
Antoine Chabert

H4311

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Table des matières

Table des figures

Liste des tableaux

Première partie

Livrables

1 Déroulement et répartition des tâches

Durant ce projet, nous devions implémenter un analyseur et interpréteur du langage de programmation *lutin*. Pour ce faire, nous avons suivi les étapes de travail proposées dans le sujet en attribuant des membres sur chacune d'elles, permettant ainsi de les paralléliser.

Dans un premier temps, tout le monde a réfléchi à la grammaire du langage, puis nous avons choisi celle qui était la plus simple et la plus compréhensible par l'équipe. Nous l'avons ensuite transformé en grammaire LR, visible dans la section 2.

Nous avons ensuite avancé en parallèle sur les tâches ci-dessous avant de nous consacrer à l'implémentation :

- Construction de l'automate LR correspondant (voir section 3) puis réalisation de la table des transitions (voir section 4) (par Jonathan, Nicolas et Romain)
- Conception de la structure de données sous forme de diagramme UML 5 (par Antoine, Pierre et Baptiste)
- Identification des expressions régulières (par Benoît)

Enfin, nous avons procédé à l'implémentation de cet analyseur et interpréteur de la manière suivante :

- Implémentation de l'analyseur lexical (par Jonathan)
- Implémentation de l'automate LR (par Antoine et Benoît)
- Réalisation de l'outil en ligne de commande (par Baptiste)
- Vérification statique du programme (par Pierre)
- Implémentation de toutes les classes symboles (par Nicolas et Romain)
- Conception de la partie optimisation et son implémentation (par Pierre et Baptiste)
- Conception et réalisation de la partie affichage du programme en mémoire (par Jonathan)
- Mise en place des gestions d'erreurs lexicales et syntaxiques et récupération (par Antoine, Benoît et Pierre)
- Implémentation des tests (par toute l'équipe)

À titre indicatif, pour terminer ce projet nous avons passé, en dehors des séances, en moyenne 8 heures chacun.

2 Grammaire du langage lutin

Vous trouverez dans le tableau ci-dessous (cf. tableau 1), l'ensemble des règles de notre grammaire pour le langage lutin, numérotées pour faciliter la compréhension de la table de transition (voir section 4). Le symbole ϵ représente l'élément vide.

Numéro	Règle
1	P → D I
2	D → D D'
3	D → ϵ
4	D' → var V ;
5	D' → const C ;
6	V → V , id
7	V → id
8	C → C , id = val
9	C → id = val
10	I → I I'
11	I → I'
12	I' → lire id ;
13	I' → ecrire E ;
14	I' → id := E ;
15	E → E + E
16	E → E * E
17	E → E - E
18	E → E / E
19	E → (E)
20	E → id
21	E → val

TABLE 1 – Grammaire du langage lutin

3 Automate LR

Vous trouverez ci-dessous (cf. figure 1) le schéma décrivant l'automate LR de notre grammaire permettant d'identifier les différents états et transitions avant l'implémentation.

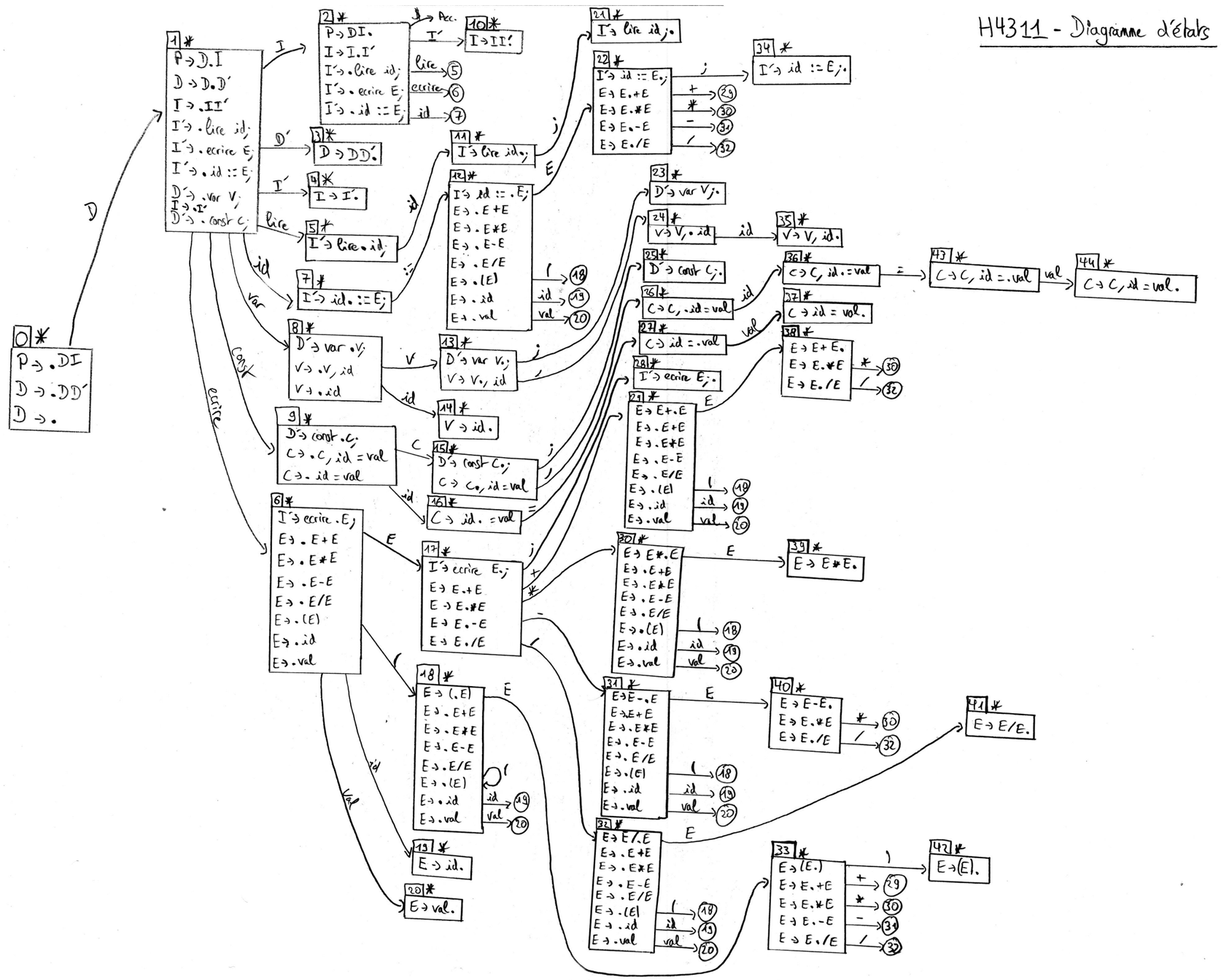

FIGURE 1 – Automate LR par rapport à la grammaire du langage lutin

4 Table des transitions LR

Avant d'implémenter l'ensemble des états, nous avons réalisé une table référençant toutes les transitions possibles d'un état à un autre en fonction des symboles analysés. Voici cette table de transition LR (cf tableau 4).

	var	const	;	,	id	val	=	lire	ecrire	:=	+	-	*	/	(	)	$	P	D	D'	V	C	I	I'	E
0	R3	R3			R3			R3	R3										E1						
1	E8	E9			E7			E5	E6											E3			E2	E4	
2					E7			E5	E6								A							E10	
3	R2	R2			R2			R2	R2																
4					R11			R11	R11								R11								
5					E11																				
6					E19	E20									E18										E17
7										E12															
8					E14																E13				
9					E16																	E15			
10					R10			R10	R10								R10								
11			E21																						
12					E19	E20									E18										E22
13			E23	E24																					
14			R7	R7																					
15			E25	E26																					
16							E27																		
17			E28								E29	E31	E30	E32											
18					E19	E20									E18										E33
19			R20								R20	R20	R20	R20		R20									
20			R21								R21	R21	R21	R21		R21									
21					R12			R12	R12								R12								
22			E34								E29	E31	E30	E32											
23	R4	R4			R4			R4	R4																
24					E35																				
25	R5	R5			R5			R5	R5																
26					E36																				
27						E37																			
28					R13			R13	R13								R13								
29					E19	E20									E18										E38
30					E19	E20									E18										E39
31					E19	E20									E18										E40
32					E19	E20									E18										E41
33											E29	E31	E30	E32		E42									
34					R14			R14	R14								R14								
35			R6	R6																					
36							E43																		
37			R9	R9																					
38			R15								R15	R15	E30	E32		R15									
39			R16								R16	R16	R16	R16		R16									
40			R17								R17	R17	E30	E32		R17									
41			R18								R18	R18	R18	R18		R18									
42			R19								R19	R19	R19	R19		R19									
43						E44																			
44			R8	R8																					

Table 2 – Table des transitions LR

5 Structures de données

Afin de décrire au mieux la structure employée dans ce projet, vous trouverez ci-dessous le diagramme de classes simplifié dans le standard UML (cf. figure 5). En effet, afin de le rendre plus compréhensible, seules les classes les plus importantes sont visibles, les classes de type exception étant également regroupées dans un package. Vous trouverez en annexe (cf. figure 7) le diagramme de classes complet du programme.

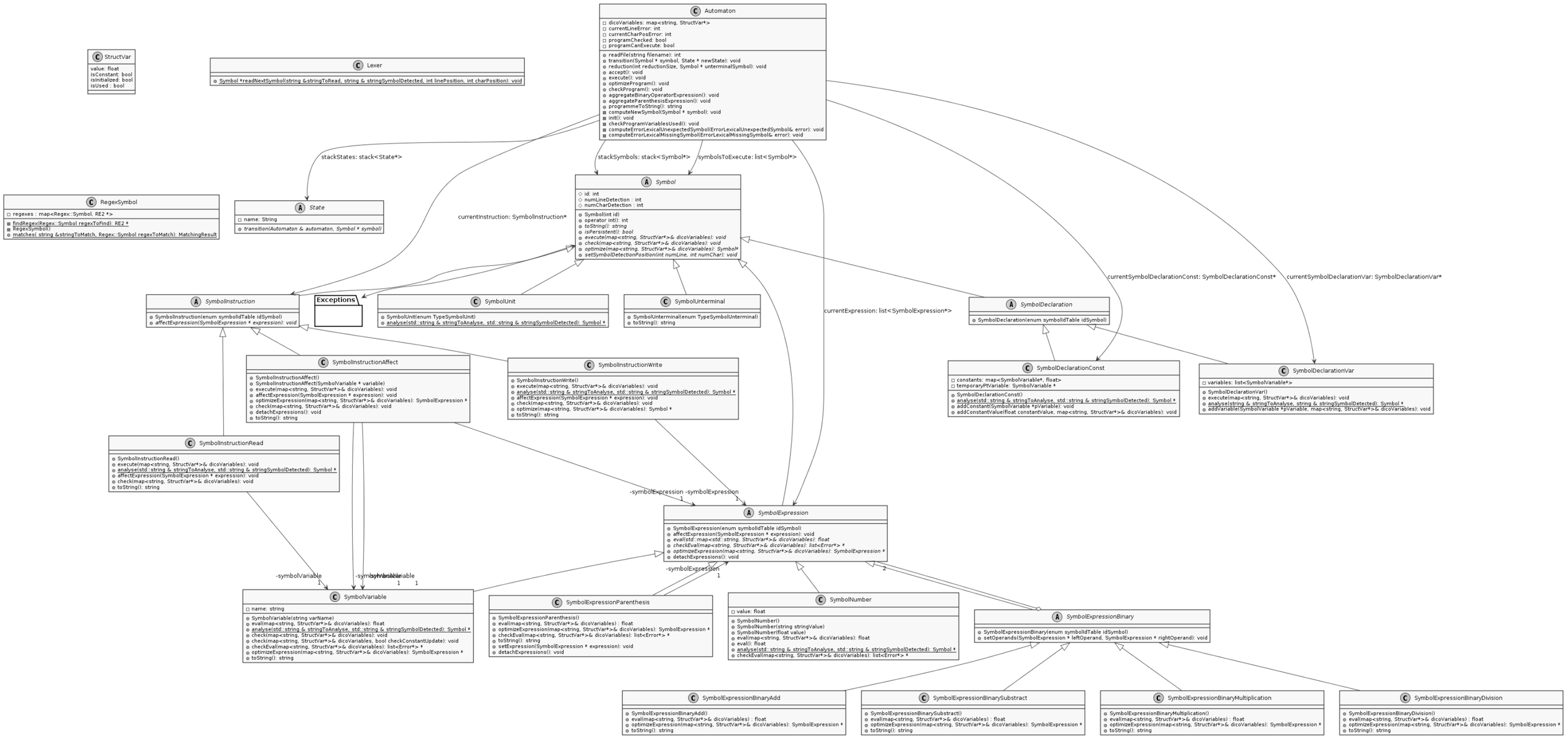

FIGURE 2 – Diagramme de classes des structures de données

Deuxième partie

Annexes

6 Gestion des erreurs

6.1 Listes des erreurs traitées

Erreur	Type	Code de retour
ERROR_SEMANTIC_VAR_NOT_USED	Semantic Warning	None
ERROR_WRONG_ARGUMENTS	Arguments	1
ERROR_FILE_NOT_FOUND	I/O	2
ERROR_SEMANTIC_VAR_ALREADY_DECLARED	Semtantic	3
ERROR_SEMANTIC_VAR_NOT_DECLARED	Semantic	4
ERROR_SEMANTIC_VAR_IS_CONST	Semantic	5
ERROR_SEMANTIC_VAR_NOT_INITIALIZED	Semantic	6
ERROR_LEXICAL_UNKNOWN_SYMBOL	Lexical	7
ERROR_LEXICAL_UNEXPECTED_SYMBOL	Lexical	8
ERROR_LEXICAL_MISSING_SYMBOL	Lexical	9
ERROR_SEMANTIC_INCORRECT_NUM_VALUE	Semantic	10
ERROR_SEMANTIC_DIVISION_ZERO	Semantic	11

Table 3 – Liste des erreurs prises en compte par le programme

6.2 Structure des erreurs

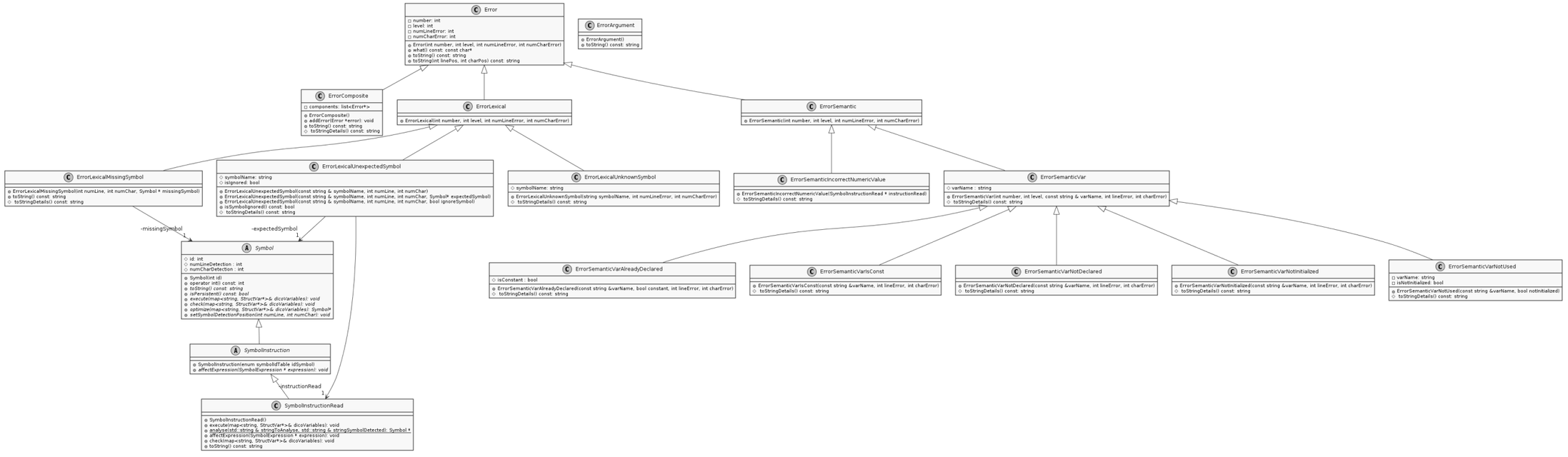

Figure 3 – Diagramme de classe des exceptions (erreurs)

7 Structures de données complètes

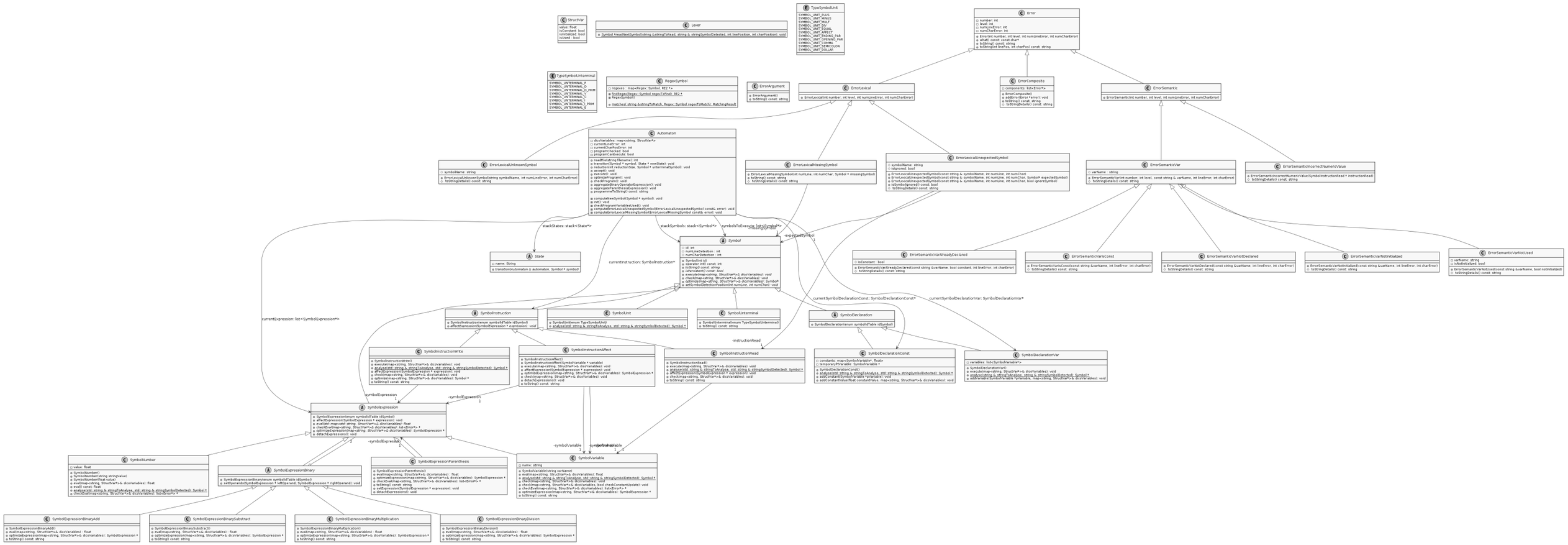

FIGURE 4 – Diagramme de classes des structures de données