Analyse syntaxique de la machine de Turing +

Présenté par : A.Kenza, F.Liza, W.Alice

Objectif

input : Prend en entrée un fichier .TSplus

- Vérifier la syntaxe de la machine de Turing de del Vigna + et afficher des messages d'erreur s'il y a des erreurs dans le script spécifié. Si pas d'erreur, affiche "Fichier Valide".

- Crée en sortie, si la syntaxe est correcte, un fichier TSV spécifiant la liste des tokens, leur type, l'instruction à laquelle ils appartiennent, le type de l'instruction, et leur position dans l'instruction s'il s'agit d'une instruction de type "opération" ou "test".

Tokens autorisés

Dans MTdV:

- G, D, 0, 1, fin, si (0), si (1), boucle, }, #, I, P, % (commentaire)

Dans MTdV+:

- entiers [0-29].
- nom de variable ([a-z]|[A-Z]|_)([0-9]|[a-z]|[A-Z]|_)* : un nom de variable peut contenir des chiffres, des lettres minuscules ou majuscules et le symbole _, mais ne peut pas commencer par un chiffre. De plus, il ne peut pas être un autre mot clé (ex boucle = 3).
- mot clé ‘si’ → pour faire des test sur les valeurs et les variables.
- opérateurs : =, +, *, == , nous envisageons l’ajout de: <, >, - (avec erreur si négatif mais c’est le problème du programmeur, pas le nôtre).

Type des tokens

- fin, si (0), si (1), boucle, }, #, I, P, % : “**trivial**”

  => il s’agit de tokens qui ne changent pas de sens par rapport à la MTdV.

- G, D, 0, 1 : “**mémoire**”

  => tokens qui touchent à la mémoire.

- si : “**complexe**”
- x, y, ma_variable : “**var**”
- +, *, =, == : “**op**”
- [0-29] : “**val**”

  => afin de savoir s’il s’agit de “val” ou “mémoire” pour 0 et 1, il faut regarder le contexte du token (désambiguisation).

Nature des erreurs

Dans MTdV, nous avions les erreurs suivantes:

- Mauvais chemin vers le fichier (404 Not Found :( )
- Token non autorisé : *token* (ligne n)
- Programme terminé sans avoir rencontré #.
- Tokens trouvés après le mot-clé # : *token_suivant* (ligne n)
- } rencontré hors d'une *boucle* ou d'un *si* (ligne n)
- Pas d'accolade fermante après *si* (ligne n)
- Pas d'accolade fermante après *boucle* (ligne n)
- Pas de *fin* dans la *boucle* (ligne n)

Ainsi que le warning:

- *fin* rencontré à l'extérieur d'une boucle, le programme pourrait s'arrêter avant la fin.

Nature des erreurs

Nous allons ajouter les erreurs suivantes:

- Opération binaire incomplète (il manque un des trois : *gauche, opérateur, ou droite)* (ligne n*)*
- Affectation incomplète (*variable, =, partie de droite)* (ligne n*)*
- Entier dépassant la limite autorisée (<0 ou > 29) : *entier* (ligne n*)*
- Token non autorisé à gauche de l'affectation : *token* (ligne n)
- Token non autorisé à droite de l'affectation : *token* (ligne n)
- Token de type entier non autorisé à cette position : *token* (ligne n)
- Nom de variable non autorisé en dehors d'une affectation ou d'un test : *token* (ligne n)

etc, voir les diapos suivantes.

Ainsi que le nouveau warning :

- Mot-clé [G D 1 0] rencontré (ligne n). Cela pourrait corrompre la mémoire du programme.

Nature des erreurs : test

- Un test doit commencer par le mot-clé si : *token* (ligne n)
- Un test doit être entouré par des parenthèses. syntaxe : si (operation1 test operation2)
- Mot-clé non autorisé dans un test : *token* (ligne n)
- Un test d'égalité ne peut pas contenir deux fois l'opérateur == : *token* (ligne n)

Nature des erreurs : opération

- Test ou affectation incomplète. (ligne n)
- Token inattendu dans une opération
- Opérateur attendu (+ ou *) dans l'opération. Opération binaire incomplète. (ligne n)
- Valeur ou variable attendue dans l'opération. Opération binaire incomplète (ligne n)
- Entier dépassant la limite autorisée. Valeurs autorisées : [0-29]
- Une opération ne peut pas se terminer par un opérateur : *token* (ligne n)

Nature des erreurs : affectation

- Côté gauche de l'affectation ne peut pas être un mot-clé. *token* (ligne n)
- Côté gauche de l'affectation doit être une variable. *token* (ligne n)
- Une affectation doit avoir un seul symbole à gauche du =. *token* (ligne n)
- Côté droite de l'affectation ne peut pas contenir un mot-clé. *token* (ligne n)
- Token non autorisé dans une affectation. *token* (ligne n)

Exemple d'output

n°instruction	token	type_token	type_instruction	position_affectation	scope_boucle
0	x	variable	affectation	G	-
0	=	operation	affectation	M	-
0	1	valeur	affectation	D	-
1	si(0)	trivial	test	-	-
2	si	complexe	test	-	-
2	x	variable	test	G	-
2	==	operation	test	M	-
2	3	valeur	test	D	-

Exemple de programme valide

```
% Script basique MTdV+
x = 0
x = x + 1
y = x * 2
boucle
   si (x == 2)
   fin }
   x = y
}
#
```

Output

Un fichier de type ‘tsv’ comme tel sera passé aux autres groupes.

id_instruction	token	type_token	type_instruction	position_operate	scope_boucle
0	%	trivial	MTdV	-	-
1	x	variable	affectation	G	-
1	=	operateur	affectation	M	-
1	0	valeur	affectation	D	-
2	x	variable	affectation	G	-
2	=	operateur	affectation	M	-
2	x	variable	affectation	D	-
2	+	operateur	affectation	D	-
2	1	valeur	affectation	D	-
3	y	variable	affectation	G	-
3	=	operateur	affectation	M	-
3	x	variable	affectation	D	-
3	*	operateur	affectation	D	-
3	2	valeur	affectation	D	-
4	boucle	trivial	MTdV	-	8
4	si	complexe	si	-	8
4	x	variable	test	G	8
4	==	operateur	test	M	8
4	2	valeur	test	D	8
4	fin	trivial	MTdV	-	8
5	}	trivial	MTdV	-	8
6	x	variable	affectation	G	8
6	=	operateur	affectation	M	8
6	y	variable	affectation	D	8
7	}	trivial	MTdV	-	8
8	#	trivial	MTdV	-	-


G1
G2
G3
G4
15 minutes de pause !