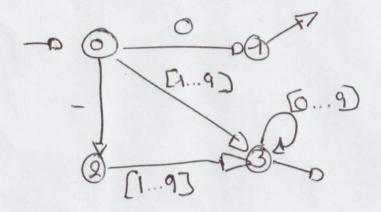
D17

LANGUAGES ET AUTOTTATES.

Partie 1

1. Automate d'état fini minimal pour OI[-18][1.9][0.9]*)



2. Automate d'état fini minimal pour [a...] [[a...z] [[a...z]] *
[a...z], [[a...z] [[a...z] [[a...z]] [[a.

Partie 2

2.1.1

da grammaire 6 n'est pas LL(1) car effe contient des règles recoursives gauches:

Les Expression & - DExpression OpArithma Expression 1

(Expression) |

VoirNum

DEXPRESSION — Not Bear |

Not Bexpression |

Bexpression and Bexpression |

Bexpression or Bexpression |

Conolition.

2.1.2 Pour rendre la grammaire LL(1), on commence par retirer les règles recursives.

Los Expression - Expression Ophrith Expression 1 (Expression) 1

VarNum

Expression of (Expression) Expression' | Varbon Expression' Expression' - OpArith Expression Expression' | &

Lo BExpression - no Val Bool | not BExpression |

BExpression and Bexpression |

BExpression or BExpression |

Condition

BEXPression — ValBool BEXPression' |

not BEXPression BEXPression' |

Conclition BEXPression

BEXPression BEXPression' |

and BEXPression BEXPression' |

2.1.2 suite On peut ensièle factoriser les règles obtenus. Expression Expression to Texpression Expression Texpression - (Expression) | Var Num Expression' - Op Arith Expression Expression') & BExpression BEXPression - OBEXPression BEXPression GBEXPRESSION - PValBOOP | not BEXPRESSION | Condition BExpression' BExpression' - TBExpression BExpression' E TBExpression -p or BExpression and BExpression TBEXPression - TBEXPression BEXPression TOExpression TBExpression - n or and

Maintenant, if faut introduire fa notion de priorité sur les operationes. Il en existe deux types, les operateurs lagiques (or, and et not), et arithmetique (+,-,x,÷).

d'ordre sera:

logique: not, puis and, puis enfin or

Transformation des operateurs arithmetiques

des règles concernées sont Expression, Expression et Texpression

Expression - D TExpression Expression'

Expression' - D OPPP TExpression Expression' | E

TExpression - D NExpression Texpression'

TExpression' - D OPP NExpression Texpression' | E

NExpression - D Var Num | (Expression)

OPPP - D + |
OPP - D * | /

Transformation des operateurs arithmetics fagiques

des règles concernées sent Béxpression, Béxpression, TBéxpression, TBéxpression, TBéxpression,

BEXPRESSION - Cambé Expression Béxpression Bexpression or Cambé Expression Béxpression of le Cambé Expression - O Not Expression Anal Expression Anal Expression | Cambé Expression - O and a Not Expression And Expression | Cambé Expression - O not Not Expression | Bool Expression - O Val Bool | Canalihan

2.1.2 suite

Enfin, une grammaire LL(1) doit être factorisées. Il reste donc à factoriser les règles qui le penvent.

LOLI-OI; LIE

ローロ: ロ1E

Lo Affectation - n ident (- Expression | ident (- Val Bool

Affectation - rident (- Affectation!

Affectation' - rident (- Affectation!

Expression | Val Book.

Lo Var Num -o ident | ident [Expression] entier

Var Num - o ident Var Num' | Expression] | E

2.1.2 suite

Avec touts ceci, l'on obtient donc une grammaire 6':

5 -p program ident begin LI end LI TO ILI' II'm; LIIE I To Affectation | While | For | I) | break Affectation - rident (- Affectation' Affectation - D Expression | ValBool While -s while BExpression do LI end For - n for ident from Valeur to Valeur do LI end If n if BEXpression Hen II doe II end VolBool - True | false BEXPRESSION - CombExpression BEXPression BEXPRESSION - n or BEXPRESSION (Control Pression BEXPRESSION) (C CombExpression -p NotExpression And Expression Not Expression - p not Not Expression | Rood Expression And Expression - p and CombExpression | & Real Expression - To Val Bool | Condition Expression to Texpression Expression Expression - 0 Oppe Expression | E TExpression - NExpression TExpression TEXPRESSION - OPP TEXPRESSION & NExpression - o Var Num (Expression) Opp - * / Oppp -0+1-Var Num -o ident Var Num entier Var Num' - D [Expression] /4 Valeur -o ident Enter entier Condition -o Expression Ophel Expression Ophel - 0 <= | < | > | = | !=

2.1.2 suite

Pour Pinir, il faut tester que notre nouvelle grammaire 6'est bien LL(1) avec l'algorithme premier et suivant

PT	Premier	Suivent
. 5	program	\$
巨	ident, while, for, if, break	end, else
江'	٤,;	end, else
I	Metatrost ident, while for if break	end, else,;
Affectation	ident	end, d'se,;
Affectation'	ident, entier, (, true, folse	end, else,
while	while	end, efse,;
For	for	erd, else, ;
P	if	end, else, :
uspear	true, false	and, or, do, and, then, else,;
BEXPRESSION	ident, entier, (, true, folse, not	do, Hen
BExpression'	a, or	do, Hen
Combexpression	indent, entier, (, true, false, not	or, do, then
- chesoro!	really entirer, Chrose Polos	
merc pres non	e, and	and, or, do, then
RadExpression	ident, onlier, (, true, false	or, do, then
Expression		and, or, do, then
Expression!	ident, entier, (3,), <=, <, >, >=,=, =, and, or, do, end, Hen, else,;
	٤,-	-17, -17, 7, =, =, =, and, or, do, end, then elso.
TExpression	indent, enhier, (7),+,-,<=,<,>>=,=,!=, and, ar, do, Hen, else,; 7,)+,-,<=,<,>=,=,!=, and, or, do, Hen, else,; 7),*/<=/
Texpression'	2,*,1	7,),+,-, <=, <, >, >=,=,1=, and as do llos als
DExpression	ident, entier, (7), */ <= < >> = - 1: and, or, and, then, else, ;
Opp	*,/	indent entire (
OPPP	+ -	indent, enhier, (ident, enhier, (
VarNum	ident, enhier	3),*/,+,-(5/32)
Var Num'		7,),*,/,+,-,<=,<,>=,=,l=, and, or, do, end, then, etc.; 3),*,/,+,-,<=<>>==l=, and, or, do, end, then, etc.;
Valeur	ident, entier	7),*, 1, +, -, <=, <>,>=,=,!=, and, or, do, end, Hun, else,; elo, to
Conclibion	ident, entier, (and, or, do, then
Ophel	ζ=,ς,),>=,=,l=	indent, enhier, (