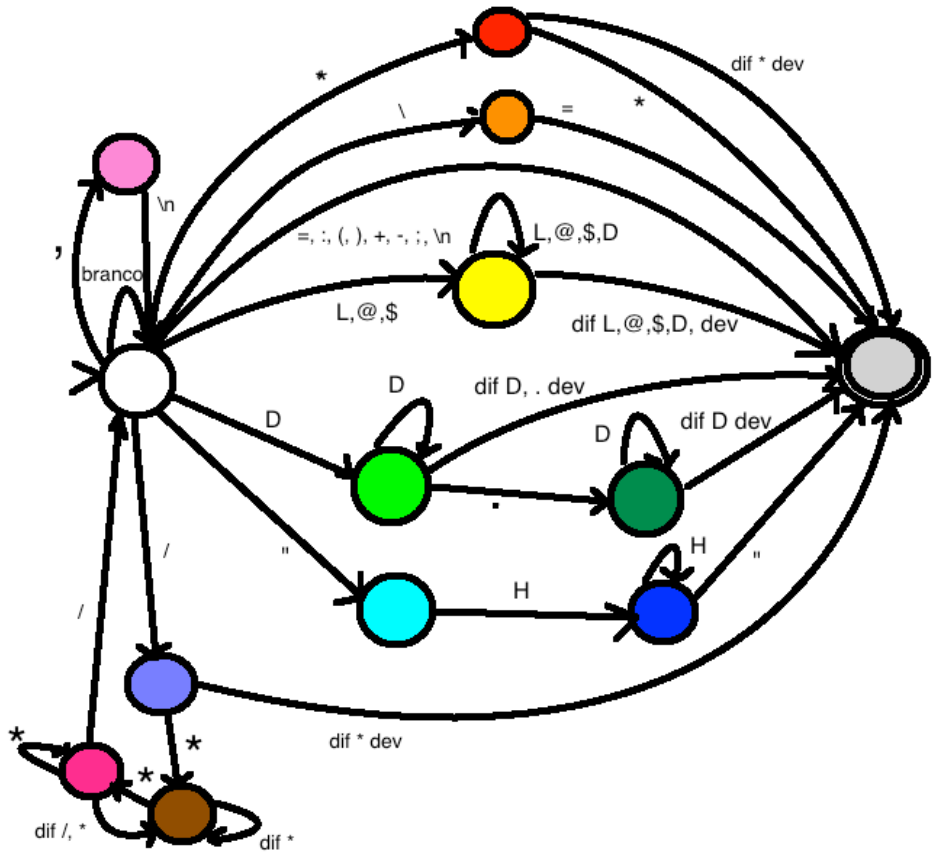
Pontifícia Universidade Católica de Minas Gerais – Departamento de Ciência da Computação
Disciplina: Compiladores
Prof. Alexei Machado

Primeira Prova

1)

Token	Lexema
id	(L|@|$)(L\|@\|$|D)*
const	D+(.D*|lambda) | "H+"
Q	\n | ;
=	=
REPEAT	(R|r)(E|e)(P|p)(E|e)(A|a)(T|t)
END	(E|e)(N|n)(D|d)
WHILE	(W|w)(H|h)(I|i)(L|l)(E|e)
\=	\=
IN	(I|i)(N|n)
:	:
**	**
(	(
)	)
*	*
/	/
+	+
-	-

2)



3)
Esta primeira versão da gramática, embora fatorada, não é LL(1) pois exp gera id:

```
S-> {C Q}+
C-> id = exp |
    REPEAT ( exp                        {C Q} END |
             id IN exp:exp[:exp]        {C Q} END |
             WHILE exp (=|\=) exp       {C Q} END
           )
exp-> [-] T {(+|-) T }
T-> F {(*|/) F }
F -> "("exp")" | const [E] |  id [E]
E-> ** (id | const | "("exp")")
```

Gerando uma gramática LL1 equivalente (substituindo exp por tudo que ele gera, para fatorar os ids):

```
S-> {C Q}+
C-> id = exp |
    REPEAT ( -T{(+|-) T}                        {C Q} END |
             "("exp")" {(*|/) F }{(+|-) T }{C Q} END |
             const [E]  {(*|/) F }{(+|-) T }{C Q} END |
             id [E]     {(*|/) F }{(+|-) T }{C Q} END |
             id IN exp:exp[:exp]            {C Q} END |
             WHILE exp (=|\=) exp           {C Q} END
           )
exp-> [-] T {(+|-) T }
T-> F {(*|/) F }
F -> "("exp")" | const [E] |  id [E]
E-> ** (id | const | "("exp")")
```

Agora basta fatorar os ids. O sufixo comum pode ser fatorado à direita mas não é obrigatório:

```
S-> {C Q}+
C-> id = exp |
    REPEAT(  -T{(+|-) T}                        |
             "("exp")" {(*|/) F }{(+|-) T }     |
             const [E]  {(*|/) F }{(+|-) T }    |
             id ( [E]{(*|/) F }{(+|-) T } | IN exp:exp[:exp] ) |
             WHILE exp (=|\=) exp
           ){C Q} END
exp-> [-] T {(+|-) T }
T-> F {(*|/) F }
F -> "("exp")" | const [E] |  id [E]
E-> ** (id | const | "("exp")")
```