1、LR(1)的含义

↓从左到右处理输入，→ 形成最右推导，使用先行的1个符号

自底向上的分析使用了显式栈，栈中包括已token和非终结符，以及一些其他信息。

刚开始栈是空的，输入串的末尾加了$

若最终栈中为$S，输入队列$，则分析成功

自底向上分析有两步动作：

①移进/移入(shift)：将终结符从输入的队列到栈顶。

②规约(reduce)：假设有BNF选择A→α，将栈顶部的串α归约为非终结符A。

2、扩充/增广(augmented)：

增加S'→S到文法中，且令S'为开始符号

3、LR(0)项(LR(0) item)：上下文无关文法右边带有区分位置的产生式选择. → LR(0)项是带有位置信息的产生式

用.区分这个位置.

LR(0)项构建：①把所有产生式展开(删除选择)

②在每一个产生式中的右侧子用位置添加.

项记录了特定文法规则右边识别中的中间步骤

4、初始项(initial item)

项目A→.α意味着未来将要利用A→α识别A

完整项(complete item)

项目A→α.意味着α现在位于分析栈的顶部.

5. LR(0)项的NFA

状态：LR(0)项作为自动机的状态来使用.

转换：2步

对于LR(0)项 $A \to \alpha . X \eta$，有

① 添加转换 $A \to \alpha . X \eta \xrightarrow{X} A \to \alpha X . \eta$

② 若X是非终结符，则对于每个初始项 $X \to . \beta$

添加转换 $A \to \alpha . X \eta \xrightarrow{\varepsilon} X \to . \beta$

开始状态：

初始项 $S' \to . S$ 为NFA的开始状态

接受状态：

NFA无接受状态（关于接受的信息都在接受状态中）

6. LR(0)项NFA → DFA：

子集构造法（方法一样）

核心项和闭包项

kernel item　　closure item

在LR(0)的DFA中，在ε闭包步骤中添加到状态中的项目为闭包项

引起状态作为转换目标的项目称为核心项。

核心项的重要性：

若有一个文法，核心项唯一地判断出状态以及它的转换，我们只需指出核心项就可以完整地表示出项目集合的DFA.

LR(0)分析算法 → 使用了构建好的NFA

该算法需记录DFA的当前状态，所以须修改分析栈以使不但能存储符号，而且还能存储状态。

修改分析栈

在压入一个符号后再将新的状态压入分析栈。

所有的分析栈开始时先压入为初始状态和开始状态或符号

开始时分析栈中和输入串尾添加$

$符号

动作定义

令当前状态S为栈顶状态，

移进：若状态S包含了格式A→α.Xβ的任何项目，其中X是一个终结符，则动作即为移进，把X压入栈中，后面压入的状态为包含项目A→αX.β的状态。

只要存在一个就行

先判断是否移进

规约：若状态S包含了任何完整项(A→α.)，则动作为用规则A→α归约，具体如下：

①将串α及它的所有对应状态从分析栈中弹出

②将A压入栈中，并压入包含B→αAβ的状态。

用S'→S和接受判断

LR(0)文法判断：根据LR(0)DFA 无移规和规规模即为LR(0)文法

当且仅当每个状态仅包括"移进项目"或仅包括单个完整项目的归约时，该文法才是LR(0)。

LR(0)分析表：五大列

纵轴是DFA的状态，横轴由是动作（规则（归约用）、输入（移进用）、Goto（移进用））

token输入　非终结符输入

状态	动作	规则	输入	Goto
0	移进			
1	规约	A'→A		
2				

新状态

规约的话，输入和Goto都是空的

移进的话，规则是空的

构造方法：以及LR(0)文法判定 和SLR(1)文法判定

均为隔后状态

①遍历每个状态，

②判断该状态的类型（移进或规约），即确定动作

③选择规约使用的规则或对不同的输入(token)和"Goto"(非终结符)确定新状态。

根据LR(0)分析移进和规约的定义

10、SLR(1)分析算法

简单LR(1)分析算法，仍使用LR(0)项的DFA.

Q.与LR(0)分析的区别：

①在移进前考虑输入的token以确保存在一个恰当的DFA.

②使用非终结符的Follow集合来决定是否执行一个归约。

动作定义

唯一 与LR(0)的不同

移进：若状态S包含了$A \to \alpha.X\beta$的任意项目，X是个终结符，且X是输入串中的下一个记号，则动作为将当前输入的token移到栈中，且被压入到栈中的新状态是包含了项目$A \to \alpha X.\beta$的状态。

归约：若状态S包含了完整项目$A \to \gamma.$，且输入串的的下一个token在Follow(A)中，则动作是用规则$A \to \gamma$规约。

用规则$S' \to S$归约与接受等价，且只有当下一个输入记号是\$时，这才发生。 Follow(S)包含\$

具体：①删除串γ和所有它的相应分析栈中的对应状态

②将A压入，并将包含项目$B \to \alpha.A\beta$的状态压入。

12、SLR(1)文法判定：

根据DFA

当且仅当每个状态S满足下面2条件：

移进-归约冲突 (1) 对于S中的任何项目$A \to \alpha.X\beta$，当X是终结符，且X在Follow(B)中时，S中没有完整项$B \to \gamma.$

规-规冲突 (2) 对于S中的任何两个完整项$A \to \alpha.$和$B \to \beta.$，Follow(A)∩Follow(B)为空。

11、SLR(1)分析表：

将LR(0)分析表的①动作规则列删除，将其内容合并至输入列的格子中。②且输入列再加一个\$

格式

状态	输入			Goto
	n	+	\$	E
0	s1			1
1		r(E→n)		2

s1：①移进 ②新状态是1

r(E→n)：用E→n规约

2：新状态是2

1. LR(1)的引入

2、增广文法 $S' \to S$

3、LR(0)项的定义及构建 ①原点选择 ②不同位置为点

4、初始项和完整项

5、LR(0)项的NFA构建 → 一个LR(0)项定义为一个状态 → 不说移入和规约

开始状态：$S' \to .S$

转换：两步

① $A \to \alpha . X\eta \xrightarrow{X} A \to \alpha X . \eta$

② 若X是非终结符，则又将每个 $X \to .\beta$，添加 ε

初始项

$A \to \alpha . X\eta \xrightarrow{\varepsilon} X \to .\beta$

6、LR(0)项NFA转DFA：子集构造

7、LR(0)分析表的结构和内容 七列：状态、动作、规则、输入token、Goto（非终结符）

构造 → 根据DFA

8、LR(0)分析算法：开始描述和动作定义

9、LR(0)文法判断 → 根据DFA，每个状态要么规约要么移入

10、SLR(1)分析算法 移进 考虑 输入的一个token

规约 $A \to \alpha$ 考虑 输入的一个token是否在Follow(A)集中

11、SLR(1)分析表结构和内容 三列、动作和规则并入输入的指中

12、SLR(1)文法判定 每个状态S

①若存在 $\alpha X\eta$，且X是终，且X在Follow(B)中，该状态S不含完整项 $B \to \gamma$.

②任意两个完整项 $A \to \alpha .$、$B \to \beta .$

$Follow(A) \cap Follow(B) = \phi$。

过程①增广文法

②LR(0)项构建

③