

上下文无关语 言 姚刚

目录 上下文无关文 无关 分析与二义 性 下文程序设计 语言

第五章 上下文无关语<u>言</u>

姚刚

中国科学院信息工程研究所



目录

上下文无关语 言 姚刚

目录 上下文无关文 法 分析与二义性 上下文无关语

●上下文无关文法

2 分析与二义性

③ 上下文无关语言和程序设计语言



上下文无关语

扩大语言族

言 姚刚 目录 上下文无关文 上下文无关文 上下文无关诗 上下文无关诗计

因为正则语言可以有效描述某种简单模 式, 所以我们可以比较简单地找到非正 则语言的例子。在程序设计语言中, 有 一种简单的嵌套结构, 如(())和((()))的 形式,而(()则不会出现。这意味着程 序设计语言的某些属性需要某些正则语 言不具有的性质。 为此, 我们需要扩大语言族, 着导致我 们考虑上下文无关(context-free)语言及 其文法。



正则文法的限制

上下文无关语 姚刚 上下文无关文

正则文法的产生式有两方面的限制:

- 左侧必须是一个简单变量:
- 右侧有固定的形式。

为了创造出更强大的文法, 我们放宽这 些限制:保留左侧的限制,允许右侧是 任何形式, 我们就获得了上下文无关文 法。



上下文无关文法

上下文无关语 言 姚刚

上下文无关文

上下文无关语言和程序设计语言

定义 (上下文无关文法)

文法G = (V, T, S, P)称为是上下文无关的(context-free),如果P中的产生式具有形式 $A \to x$,其中 $A \in V$, $x \in (V \cup T)^*$ 。语言L是上下文无关的,当且仅当存在一个上下文无关文法G,使得L = L(G)。



上下文无关文法

上下文无关语 言

姚刚

目录 上下文无关文

分析与二义性 上下文无关语 言和程序设计 语言 每个正则文法都是上下文无关文法,因此正则语言也是上下文无关语言。 $\{a^nb^n\}$ 是上下文无关语言,但不是正则

语言。因此,正则语言族是上下文无关语言族的真子集。

告 · 族的其 · 朱。 上下文无关文法得名于 · 句型中出现的产

上下 又 允 关 又 法 得 名 于 句 型 中 出 现 的 产 生 式 的 左 部 可 以 在 任 何 时 候 被 替 换 , 这 种 替 换 不 依 赖 于 句 型 中 的 其 他 符 号 。

例子

上下文无关语 姚刚

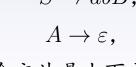
上下文无关文

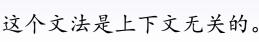


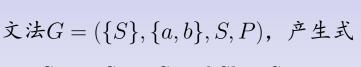


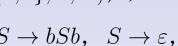
式

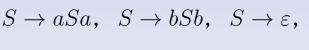
$S \to abB$, $A \to aaBb$,

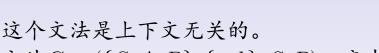












文法
$$G = (\{S, A, B\}, \{a, b\}, S, P)$$
,产生

$$A \to \varepsilon$$
, $B \to bbAa$,



例子

上下文无关语言
姚剛

语言 $L = \{a^n b^m : n \neq m\}$ 是上下文无关的。

上下文无关文

注

线性文法和正则文法都是上下文无关的,但是上下文无关文法不一定是线性的。这个例子给出的文法是上下文无关的,不是线性的,但是不能说这个语言就不是线性的。

分析与二义 上下文无关 言和程序设 语言



推导过程

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关设计 语言 在上下文无关文法中,推导可能包括由 多个变量构成的句型。这样,在替换变量的时候,我们可以进行选择。 为了表示使用的是哪个产生式,我们将 产生式进行编号,并在⇒上标出相应的 编号。



例子

上下文无关语 姚刚 上下文无关文

文法 $G = (\{S, A, B\}, \{a, b\}, S, P),$ 产生 式 \circ $S \to AB$.

考察文法生成的语言,并给出aab的推

 \bullet $A \rightarrow aaA$, \bullet $A \to \varepsilon$. \bullet $B \to Bb$.

导过程。

 \bullet $B \to \varepsilon$.



推导顺序

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关设计 言和程序设计 在上下文无关文法中,不同的推导可能 得到相同的句子,而且使用了相同的产 生式。它们的不同仅仅是使用产生式的 顺序不同。

为了去掉这些不相关的因素, 我们要求变量按照固定的顺序被替换。



最左推导与最右推导

上下文无关语 言 姚刚

日录 上下文无关文 法 分析与二义性 上言和程序设计

定义 (最左推导与最右推导)

如果每次都替换句型中最左端的变量,那么这种推导称为最左的(leftmost)。如果每次都替换句型中最右端的变量,那么这种推导称为最右的(rightmost)。

文法 $G = (\{S, A, B\}, \{a, b\}, S, P)$, 产生 式 $S \rightarrow aAB$, $A \rightarrow bBb$, $B \rightarrow A|\varepsilon$, 给出abbbb的最左推导与最右推导。



另一种推导方式

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关语言和程序设计 语言 我们给出另一种表示推导过程的方式,叫做推导树(derivation tree),这种方式与产生式P的使用顺序无关。

推导树是一个有序树, 其中父结点标记 的是产生式左部的符号, 子结点标记的 是该产生式右部的符号。



推导树

上下文无关语 言 姚刚

目录 上下文无关文

分析与二义性 上下文无关语 言和程序设计

定义 (推导树)

G = (V, T, S, P)是一个上下文无关文法。当且仅当下列性质成立时,一个有序树才能称为是G的推导树:

- 。根结点标记为S。
- 。每个叶结点有一个属于 $T \cup \{ε\}$ 的标记。
- · 每个中间结点(不是叶结点的结点)都有一个属于V的标记。



推导树

上下文无关语 言

姚刚

目录

上下文无关文

上下文无关语言和程序设计

定义 (推导树(续))

- •如果一个结点有标记 $A \in V$,它的子结点标记 $A \in V$,它的子结点标记(从左到右)为 a_1, a_2, \cdots, a_n ,那么,P中一定存在包含形式为 $A \to a_1 a_2 \cdots a_n$ 的产生式。
 - •标记为 ε 的叶结点没有兄弟,即:一个结点标记为 ε ,则这个结点是叶结点,并且是它的父结点的唯一子结点。



部分推导树

上下文无关语 言 姚刚

目录 |上下文无关文

分析与二义性 上下文无关语 言和程序设计 语言

定义

如果一棵树拥有部分推导树的后三条性 质,但是第一条性质不成立,并且把第 二条性质替换为:

• 每个叶结点都有一个属于 $V \cup T \cup \{\varepsilon\}$ 的标记。

那么,这样的一棵树就称为部分推导树(partial derivation tree)。



树的果(yield)

上下文无关语 言 姚刚

上下文无关文 法

分析与一义性 上下文无关语 言和程序设计 语言 通过从左向右地读出树上的叶结点获得的符号串,忽略遇到的任何 ε ,称为树的果。

"从左到右"的术语描述可以给出一个准确的意义。果是终结符构成的符号串,是按照深度优先遍历树时遇到的字母顺序构成的,深度优先遍历总是先遍历最左部没有被遍历过的分支。



上下文无关语 言 姚刚 目录

分析与二义性 上下文无关语 言和程序设计 语言 文法 $G = (\{S, A, B\}, \{a, b\}, S, P)$, 产生式

 $S \to aAB$, $A \to bBb$, $B \to A|\varepsilon$,

给出abbbb的推导树。

定理

上下文无关语 言

姚刚

目录 上下文无关文

~ 分析与二义性

上下文无关语 言和程序设计 语言

定理

设G = (V, T, S, P)是一个上下文无关文 法,那么对于每个 $w \in L(G)$, G中总存 在一个推导树,它的果是w。反过来. 任何推导树的果都属于语言L(G)。 同样,如果 t_G 是G的任何一个部分推导 树. 且根结点标记为S, 那么 t_G 的果就 是G的的一个句型。



上下文无关语

推导树与产生式的顺序

■ 姚剛 目录 上下文无关文 法 分析与二义性 上下文无关语 计 语言 推导树指出了获取句子的过程中用到 了哪些产生式,但是没有给出这些产生 式应用的顺序。推导树能够表示任何推 导, 反映了顺序是无关的。 对于 $w \in L(G)$ 都有一个推导, 但是不 能说一定有最左推导和最右推导。有 了推导树, 通过把树看成是如下方法 构造的,即总是由树的最左边变量开 始扩展,来得到最左推导。任何 $w \in$ L(G), 既有最左推导, 也有最右推导。



分析

上下文无关语 言 姚刚

分析与二义性

人们在学习自然语言的时候都熟悉一 种方式,就是通过文法推导解释一条语 句, 我们称之为分析(parsing)。 分析是描述语句结构的一种形式。每当 我们需要理解语义的时候, 例如要把一 种语言翻译成另一种语言. 分析是非常 重要的。



分析

上下文无关语 言 姚刚

目录 上下文无关文

分析与二义性 上下文无关语 言和程序设计 给定文法G, 我们学习用G可以推导出符号串的集合。

在实际中,我们也关注文法的分析性质:给定终结符构成的符号串w,我们想要知道w是否属于L(G)。

成员资格判定算法可以告诉我们w是否属于L(G)。术语分析描述的是寻找 $w \in L(G)$ 的推导过程中使用的一系列产生式。



分析过程

上下文无关语 言 姚刚

目录 上下文无关文 法

分析与二义性 上下文无关语

给定L(G)中的符号串w, 我们系统地构 造所有可能的(比如:最左)推导,看看 推导出的结果是否能够匹配w。 首先看所有具有形式 $S \to x$ 的产生式, 从而发现可以由S一步推导出的x。 如果结果都不匹配w,那么将所有可以 应用的产生式都用在每个x 的最左边的 变量上。这样得到一个句型的集合, 或 许其中有的句型就能推导出w。



上下文无关语

分析过程(续)

姚剛 目录 上下文无关文 分析与二义性 上言和定 语言 在后续的过程中,考察所有最左边的变 量,应用所有可以应用的产生式。如果 句型不能推出w,就抛弃它。因而,每 个过程都会获得一些可能的句型。 这样, 首先获得使用一个简单产生式 而生成的句型, 然后获得使用两步推导 可以得到的句型·····如果 $w \in L(G)$, 那么,它一定有有限长度的最左推导。 因此,这种方法最后会给出w的最左推 早。



穷举搜索分析

上下文无关语 言 姚刚

上下文无关 法

分析与二义性 上下文无关语 上面所述的这个分析方法,我们称之为穷举搜索方法(exhaustive search paring)。它是自顶向下分析(top-down paring)的一种形式,我们可以把它看成是自根部向下构造推导树。

文法 $G = (\{S\}, \{a, b\}, S, P)$,产生式 $S \to SS|aSb|bSa|\varepsilon$,

分析符号串w = aabb。



穷举搜索的缺陷

```
上下文无关语
言
姚刚
```

上下文无关文 **分析与二义性** 上下文无关语 主言和程序设计 语言

- 最明显的缺陷是过程冗长,不适用于需要使用高效率分析的地方。
- ② 用这种方法分析 $w \in L(G)$ 时,如果得不到属于L(G)的符号串,就可能永远都不会停止。除非使用某种方式使它停下来。



限制文法

上下文无关语 : 姚剛 目录 上下文无关文

分析与二义性 上下文无关语 言和程序设计 如果限制文法的形式, 穷举搜索分析的 不终止问题就相对容易解决。 我们看到使用 $S \rightarrow \varepsilon$ 的产生式会减少推 导出的后继句型的长度, 因此不容易判 断出什么时候停止推导。如果没有这样 的产生式,困难就少些。 事实上, 我们要消除两种形式的产生 式: $A \to \varepsilon nA \to B$ 。后面我们会看 到,这种限制不影响最终的文法表达能 力。



```
上下文无关语言
姚刚
```

文法
$$G = (\{S\}, \{a,b\}, S, P)$$
,产生式
$$S \rightarrow SS|aSb|bSa|ab|ba$$
,

能够生成前面例子中的语言,除了空串。

定理

上下文无关语 言 姚刚

日录

分析与二义性

上下文无关语 言和程序设计 语言

定理

假设文法G = (V, T, S, P)是上下文无关文法,并且不具备下面形式的产生式:

$$A \to \varepsilon$$
 或者 $A \to B$,

其中, $A, B \in V$ 。那么由穷举搜索分析法可以生成一个算法,它能完成:对于任何 $w \in \Sigma^*$,要么生成w的一种分析,要么告诉我们不可能存在分析。



产生句型的数量的上限

言 姚刚

上下文无关语

口水 上下文无关文 法

分析与二义性 上下文无关语 言和程序设计 穷举搜索分析法得到的句型数量,没有 准确的结果,但是可以给出一个严格的 上限。

如果我们只限制最左推导,一步推导后,得到的句型不超过|P|个,两步推导后,得到的句型不超过|P|2个... 由上面定理,推导过程不超过|w|步, 因此,得到的句型不超过

 $M = |P| + |P|^2 + \dots + |P|^{2|w|} \circ$



定理

上下文无关语 言

姚刚

目录 上下文无关:

[~] 分析与二义性

上下文无关语 言和程序设计 语言 定理

每个上下文无关文法都存在一个算法,它对于任何 $w \in L(G)$,都可以和 $|w|^3$ 成正比的步数进行分析。

上面的算法,工作量与符号串长度的 三次幂成比例增加,虽然比指数算法 好些,但是也不是太有效。我们想要的 是和符号串的长度成正比的时间复杂度 的分析法,我们把这种方法称为线性时 间(linear time)分析算法。



简单文法

上下文无关语 言 姚刚

目录 上下文无关³

分析与二义性

上下文无关语 言和程序设计 语言

定义 (简单文法)

上下文无关文法G = (V, T, S, P)是简单文法 $(simple\ grammer\ in s-grammer)$,如果它的产生式形式都是

 $A \rightarrow ax$,

其中, $A \in V$, $a \in T$, $x \in V^*$, 并且, 任何对(A,a)在P中至多出现一次。



例子

文法G

上下文无关语 姚刚

分析与二义性

不是简单文法。

间复杂度范围内分析完。

如果G是简单文法,那么L(G)中的任何

符号串w,都可以在与|w|成正比的时

 $S \rightarrow aS|bSS|aSS|c$

是简单文法:而文法

 $S \to aS|bSS|c$



上下文无关语

二义性

- 姚刚 目录 上下文无关文 **分析与二 又** 生 一下文程序设计 对任何给定的 $w \in L(G)$, 穷举搜索分析一定会产生一棵w的推导树。我们说"一棵", 而不是特指某棵的原因在于可能存在大量不同的推导树。

定义 (二义性)

如果对某个 $w \in L(G)$, 至少存在两棵不同的推导树,那么,上下文无关文法G称作是二义性的(ambiguous)。也就是说,二义性意味着存在两个或者两个以上的最左或最右推导。



```
上下文无关语
  姚刚
分析与二义性
```

```
已知文法G = (V, T, S, P), 其中V =
{E, I}, T = {a, b, c, +, *, (,)},  产生式
```

$$E o I$$
, $E o E + E$, $E o (E)$, $E o a|b|c$,

fa + b * c的两棵推导树。



去除二义性

上下文无关语 言 姚刚

法

上下文无关语言和程序设计

注

在程序设计语言中, 每条语句只应该有一种解释, 要去除二义性, 通常是通过重写一个等价的、无二义性的文法来达到这个目的。

重写上面例子中的文法。



去除二义性

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关语 言和程序设计 在一般情况下,一个给定的上下文无关 文法是否是二义性的,或者两个上下文 无关文法是否等价,这两个问题都是非 常困难的。

事实上,不存在通用的算法来解决这些问题。



无二义性与固有二义性

上下文无关语 言 姚刚

日录

运 分析与二义性

上下文无关语 言和程序设计 语言

定义

如果一个上下文无关语言L存在一个无二义性文法,那么L就是无二义性的。如果每个生成L的文法都是二义性的,那么这个语言是固有二义性的 $(inherently\ ambiquous)$ 。

语言 $L = \{a^nb^nc^m\} \cup \{a^nb^mc^m\}$ 是一个具有固有二义性的上下文无关语言,其中n和m是非负整数。



形式语言的应用

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关设计 产和程序设计 连章 形式语言理论的一个最重要应用是程序设计语言的定义和它们的解释器、编译器的构造。

这里的基本问题是准确地定义程序设计 语言,并使用这个定义作为书写有效、 可靠的翻译程序的起点。



文法定义程序设计语言

上下文无关语 言

姚刚

目录 上下文无关:

分析与二义性 上下文无关语 言和程序设计 语言 通常,我们习惯使用一种特殊的文法 来定义语言,这种文法称为巴克斯-诺 尔形式(Backus-Naur Form, BNF),这种 形式在本质上和我们使用的表示方式相 同,但是表现形式不同。

在BNF中,变量放在尖括号中表示,终 结符不用特殊标记。BNF也使用辅助符 号,例如|等。



文法定义程序设计语言

上下文无关语 言 姚刚

上下文无关文 法 分析与二义性 上下文无关语 言和程序设计 一些程序设计语言的很多部分使用了上下文无关文法的严格方式表示,容易受到定义的影响。 然而,并不是一个典型程序设计语言的

所有特性都可以被简单文法表示。 此外,程序设计语言的规约必须是无二 义性的。



上下文无关语

研究的问题

姚刚

法 分析与二义性 上下文无关语 言和程序设计 语言 程序设计语言的语义是一个复杂的问题。程序设计语义的规约不像上下文无关文法这样优雅而简洁。

无论从程序设计语言方面,还是从形式语言理论方面,定义程序设计语言语义 还是一个正在研究中的问题。



上下文无关语 言

姚刚

- ', 上下文无关文 法

分析与二义性

上下文无关语 言和程序设计 语言

谢谢!

主讲人: 姚刚

电子邮箱: yaogang@iie.ac.cn