

---

# Syntaktische Mehrdeutigkeiten

## Erkennung, Vermeidung und Auflösung

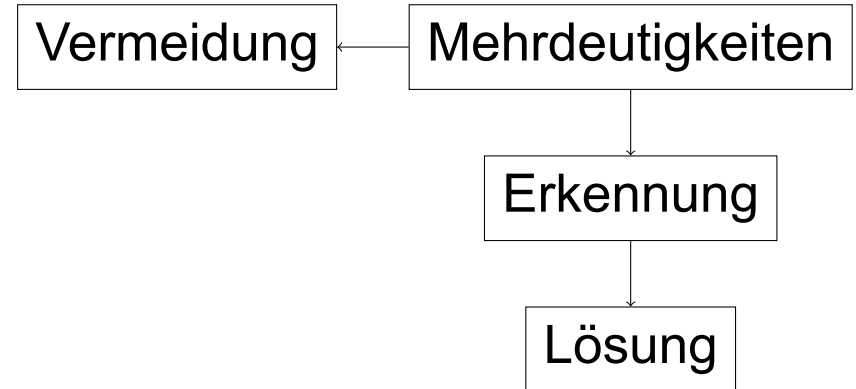
Lennart Protte

14.06.2024

# Motivation

## Zielsetzung

Entwicklung von Strategien zur **Erkennung** und **Lösung** von Mehrdeutigkeiten oder zur **Vermeidung** ihrer Entstehung.



## Beispiel

Manchmal lassen sich Mehrdeutigkeiten nicht vermeiden. [5]  
Dann ist es notwendig diese zu Erkennen und Aufzulösen.

# Probleme und Herausforderungen

---

## Problemstellungen

- Bei komplexen Programmiersprachen lassen sich Mehrdeutigkeiten kaum vermeiden.
- Es existiert kein Algorithmus, welcher stets alle Mehrdeutigkeiten erkennen kann.[4]
- Mehrdeutigkeiten können nur selten Algorithmisch aufgelöst werden.

## Herausforderungen

- Heuristiken und Verfahren zur Erkennung von Mehrdeutigkeiten.
- Entwicklung von Strategien zur Vermeidung von Mehrdeutigkeiten.
- Konzeption praxistauglicher Strategien um Mehrdeutigkeiten aufzulösen.

# Vermeidung von Mehrdeutigkeiten

## Design eindeutiger Programmiersprachen [6]

- Symbole
- Bezeichner
- Vorrangsregeln
- Assoziativität

### Mehrdeutiger C++ Code

```
class A {public: void func() {}};  
class B {public: void func() {}};  
class C: public A, public B {};  
int main() {  
    C c;  
    c.func();  
    return 0;  
}
```

### Eindeutiger C++ Code

```
class A {public: void func() {}};  
class B {public: void func() {}};  
class C: public A, public B {};  
int main() {  
    C c;  
    c.A::func();  
    return 0;  
}
```

# Erkennung von Mehrdeutigkeiten

## Methoden zur Erkennung von Mehrdeutigkeiten

- Suchbasierte Mehrdeutigkeitserkennung
- Analyse der Parsing-Tabelle
- Analyse der Grammatik

## Suchbasierte Mehrdeutigkeitserkennung [4]

	ACLA		AMBER		AmbiDexter		<i>dynamic1</i>	
	Sen	Amb	Sen	Amb	Sen	Amb	Sen	Amb
Boltzmann	-	15	58	14	27	21	1554664	2671
Altered PL	-	11	10	9	15	15	281	88
Mutated	-	15	15	6	15	15	4392	502

# Auflösung von Mehrdeutigkeiten

## Chomsky Normalform

Für  $A, B, C \in N$ ,  $S \in \text{Startsymbol}$ ,  
 $a \in T$  und  $B, C \neq S$ .

$$A \rightarrow BC$$

$$A \rightarrow a$$

$$S \rightarrow \varepsilon$$

- Mehrdeutigkeiten können durch die Umformung in eine CNF aufgelöst werden.[1]
- Es vereinfacht eine weitere Analyse der Grammatik.

## Algorithmus von Vasudevan und Tratt

$$E ::= E * E \quad (links)$$

$$> E + E \quad (links)$$

$$| \text{ num}$$

Kann unter gegebener Assoziativität und Vorrangsregeln eine eindeutige Grammatik erzeugen. [4]

- **ANTLR, YACC, Bison:** Parser-Generator-Tools, können Mehrdeutigkeiten erkennen und melden.
- **Earley-Parser:** kann jede kontextfreie Grammatik parsen, sinnvoll bei Mehrdeutigkeiten.[3]
- **Chart-Parser:** kann Mehrdeutige Grammatiken in polynomieller Zeit parsen.
- **Probabilistische kontextfreie Grammatiken (PCFGs):** Erweiterung der kontextfreien Grammatiken, fügt Wahrscheinlichkeiten zu den Produktionen hinzu

## ANTLR

```
start  : statement* EOF ;
statement: 'if' condition 'then' statement ('else' statement)?? ;

condition: '(' STRING ')';
STRING   : '"' .*? '"';
WS       : [ \t\r\n]+ -> skip ;
```

### Erläuterung des ?? Operators [2]

Der ?? Operator gibt dem Else die niedrigste Priorität, womit es mit dem äußersten If gematched wird.



## Maschinelles Lernen zur Erkennung von Mehrdeutigkeiten

- Für komplexe Grammatiken effizienter
- Risiko von Fehlentscheidungen der KI
- hohe Ressourcenintensität

## Natürliche Sprache anstatt gewöhnlichem Quellcode

- intuitiver für den Menschen
- komplexe Grammatiken erforderlich
- Die natürliche Sprache ist voller Mehrdeutigkeiten

# Quellen

---

[1] R. Kemp.

Mehrdeutigkeiten kontextfreier grammatiken.

In Jacques Loeckx, editor, *Automata, Languages and Programming*, pages 534–546, Berlin, Heidelberg, 1974. Springer Berlin Heidelberg.

[2] Terence Parr.

The definitive antlr reference.

<https://theswissbay.ch/pdf/Gentoomen2024>.

[Online; accessed 12-June-2024].

[3] Siyuan Qi, Baoxiong Jia, and Song-Chun Zhu.

Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction, 2018.

[4] Naveneetha Vasudevan and Laurence Tratt.

Detecting ambiguity in programming language grammars.

In Martin Erwig, Richard F. Paige, and Eric Van Wyk, editors, *Software Language Engineering*, pages 157–176, Cham, 2013. Springer International Publishing.

[5] John Watrous.

Parse trees, ambiguity, and chomsky normal form.

<https://cs.uwaterloo.ca/watrous/ToC-notes/ToC-notes.08.pdf>, 2020.

[Online; accessed 17-April-2024].

[6] John Watrous.

Parse trees, ambiguity, and chomsky normal form.

<https://student.cs.uwaterloo.ca/cs241/slides/sylvie/Sylvie-L13.pdf>, 2024.

[Online; accessed 17-April-2024].