A generic System

for the Implementation of Compilers

Dipl.-Inform. Peter Knauber

AG Programmiersprachen und Compiler

Universität Kaiserslautern

Contents

- **Motivation, Requirements**
- **Compiler Models**
- **A Library for Language Concepts**
- **The OCC-System**
- **Experiences**

Motivation

- ❏ **Complexity of compilers correlate with the complexity of languages to be translated**

- ❏ **Similar concepts in many languages**

- ➤ **Reuse of compiler components may reduce the overall complexity**

Specific Needs

Support the development of new languages:

- ➤ **Definition of the language**
- ➤ **Compiler prototype**
- ➤ **Reference for other compiler implementations**

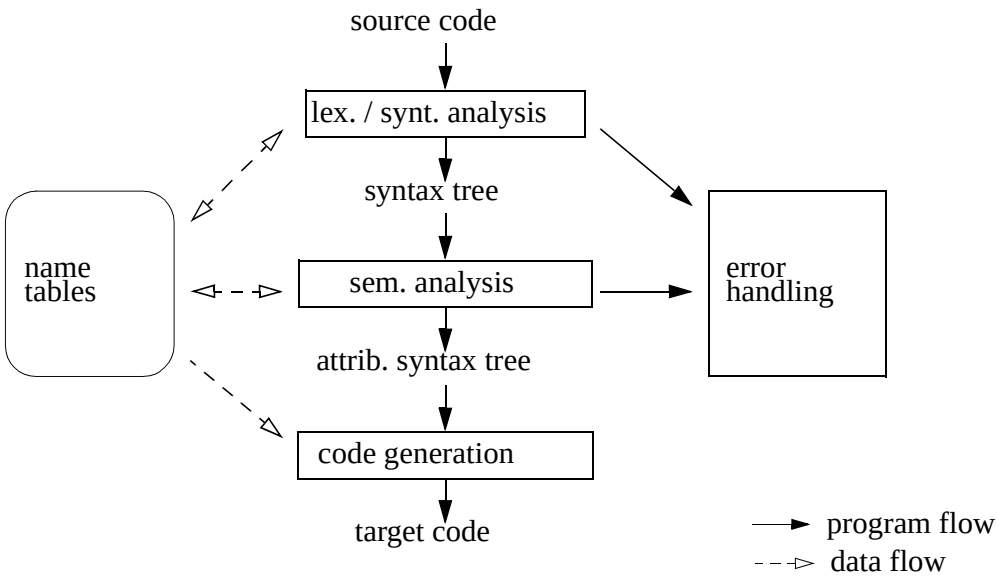
Idea for the solution: reuse of compiler components

Requirements

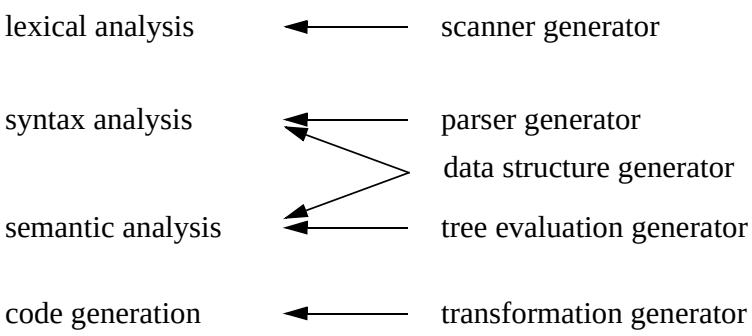
A system for language development support should …

- **support the reuse of already implemented concepts**
- **allow multiple combinations of language concepts**
- **enable the integration of new concepts**

Traditional Compiler Model



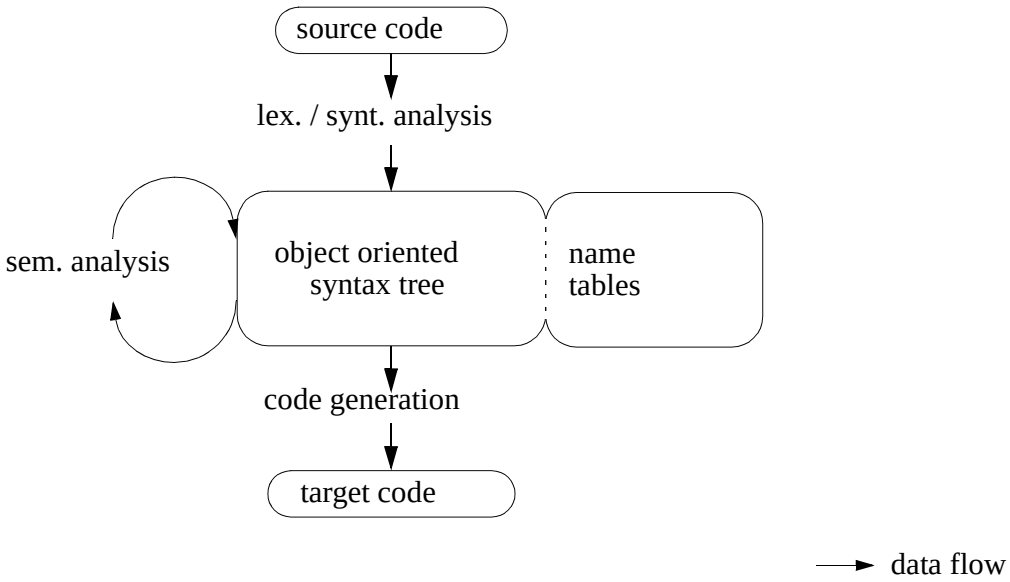
Support for traditional Compiler Generation



Resulting Problems:

- **Each concepts in each phase**
- **Reuse of functional libraries only**

Objectoriented Compiler Model



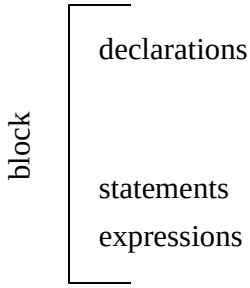
Concepts of Programming Languages

- **Structure is given by syntax**
- **Since ALGOL: block concept; examples:**

Algol 60, Algol 68:	**BEGIN** (declaration|statement)* **END**
Common Lisp:	**(let** declarations expressions **)**
Pascal, Modula-2, Oberon-2:	declarations **BEGIN** statements **END**
C, C++, Java:	**{** declarations statements **}**

Concepts

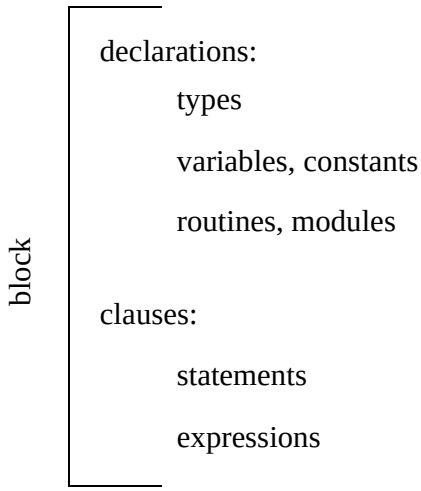
(up to now)

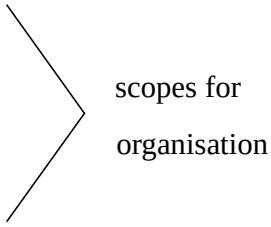


Declarations

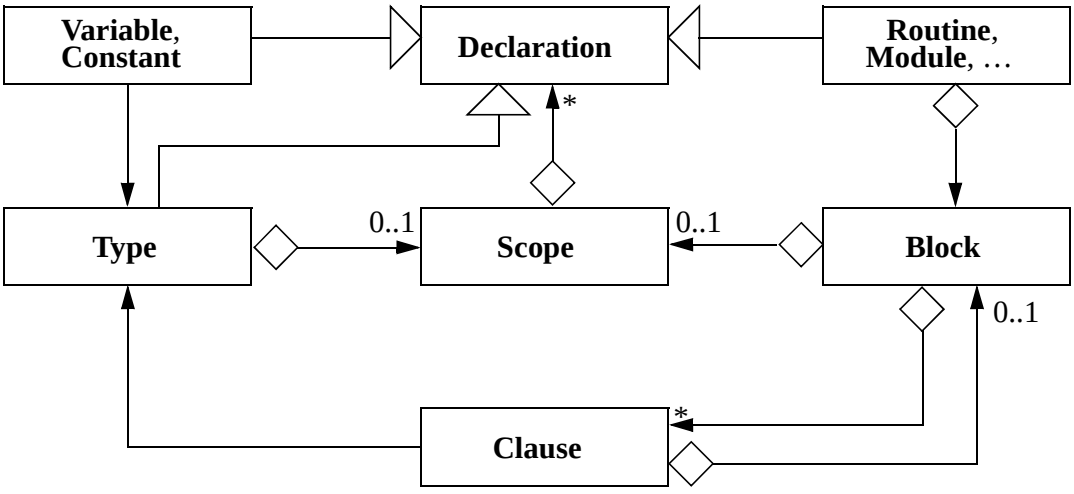
- **Constants**
- **Types**
- **Variables**
- **Routines: procedures, functions, methods, tasks, coroutines, …**
- **Modules: interfaces, definition modules, implementation modules, programs, classes, …**

Concepts





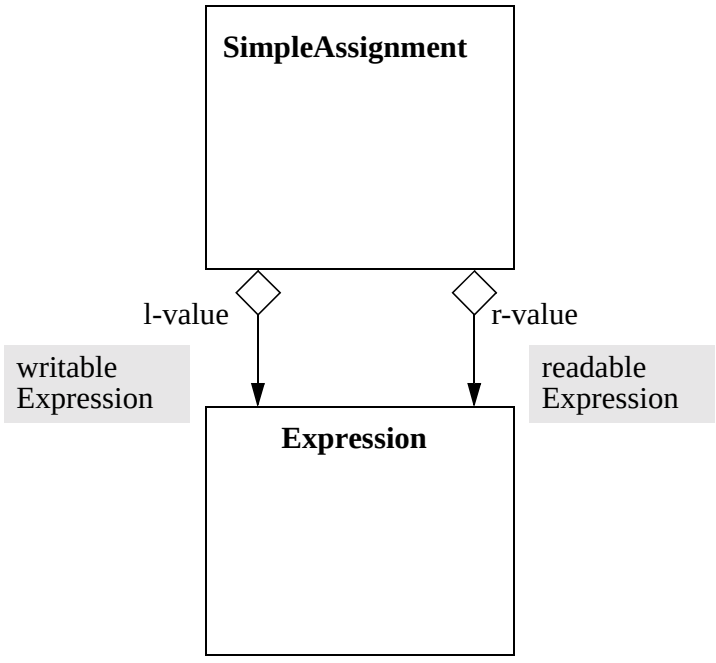
Concepts of imperative / objectoriented Languages

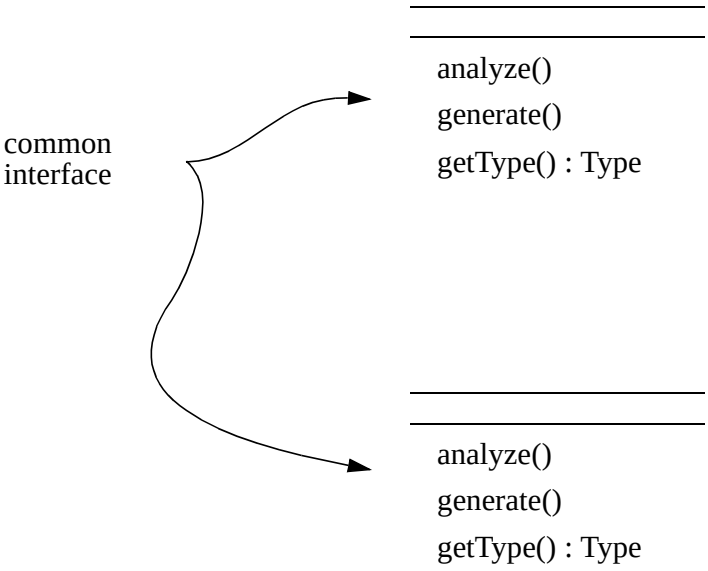


Languages

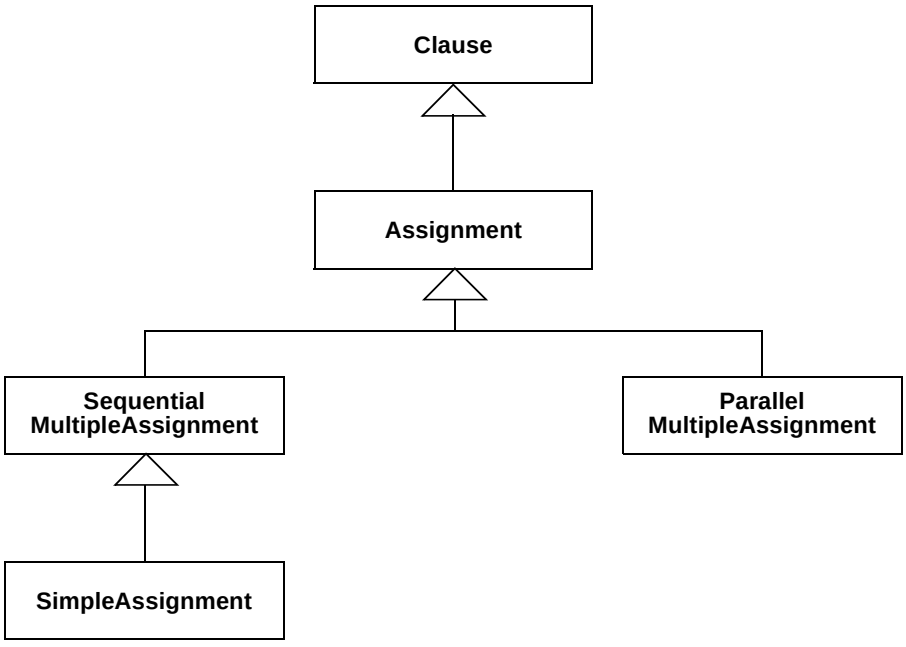
- **FORTRAN**
- **ALGOL 60, 68**
- **Common Lisp**
- **PL/I**
- **Simula 67, Smalltalk-80**
- **C, ANSI C, C++, Java**
- **Pascal, Modula-2, Oberon-2**

Example “SimpleAssignment”: objectoriented Translation

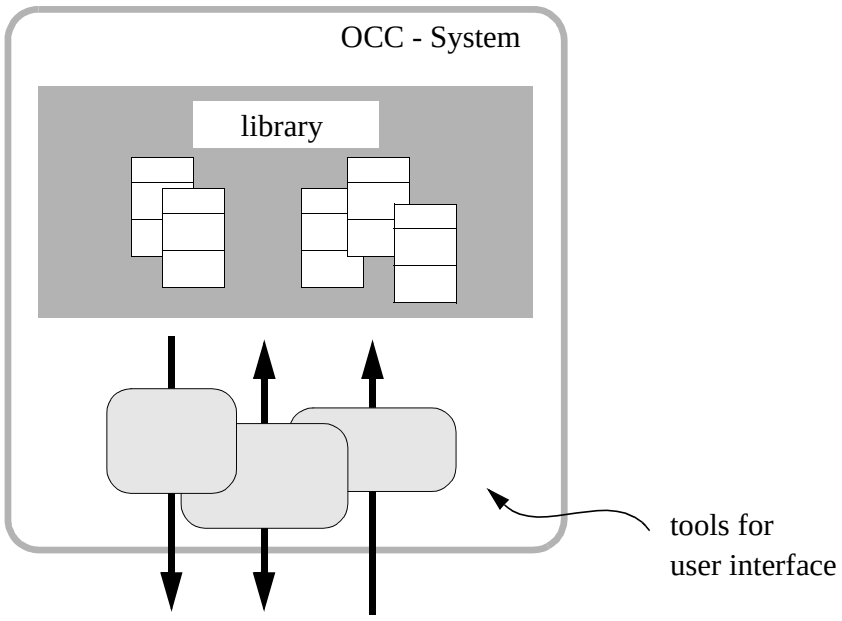




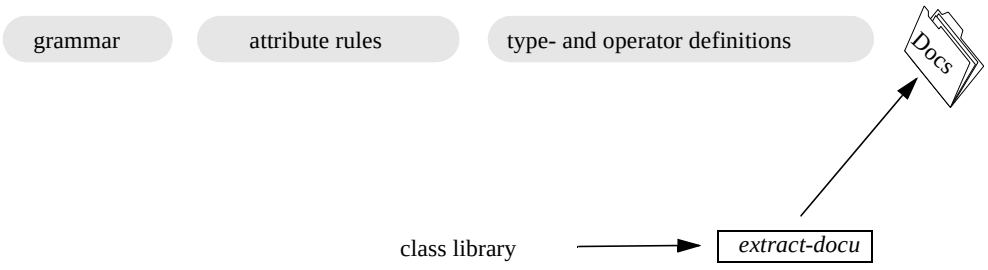
Example “SimpleAssignment”: Inheritance = Specialisation

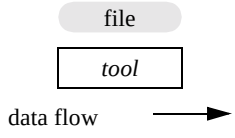


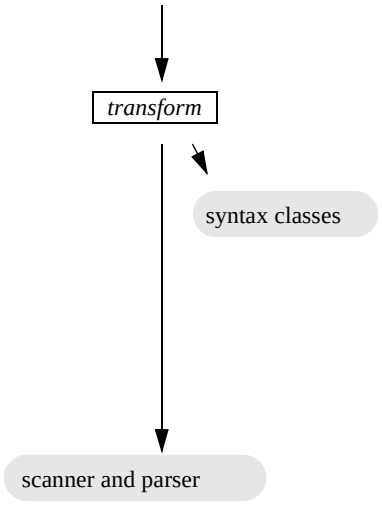
Objectoriented Compiler Construction

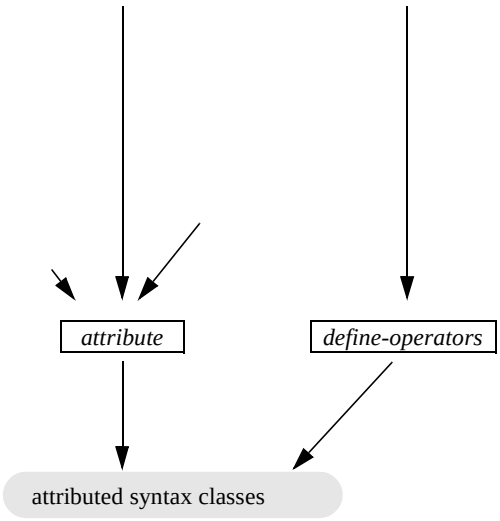


Using the OCC-System






transform
syntax classes
scanner and parser


attribute
define-operators
attributed syntax classes

Experiences from the Prototype

- **Disadvantage**
  - **Inefficiency of the generated compilers**

- **Advantages**
  - **Rapid development**
  - **Concentration on specific / new concepts**

Remarks

- **Implementation language: W-Lisp**

- **Complexity**

- **Debugging, depth of class hierarchy ≤ 6**