9. Summary
Questions to check understanding

Language properties - compiler tasks

1. Associate the compiler tasks to the levels of language definition.
2. Describe the structure of compilers and the interfaces of the central phases.
3. Give examples for feedback between compiler phases.
4. Which compiler tasks can be solved by generators? Explain what they generate.
5. Java is implemented differently than many other languages, e.g. C++,
   what is the main difference?

Lecture Programming Languages and Compilers SS 2007 / Slide 901

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Symbol specification and lexical analysis

6. Which formal methods are used to specify tokens?
7. How are tokens represented after the lexical analysis phase?
8. Describe a method for the construction of finite state machines from syntax diagrams.
9. What does the rule of the longest match mean?
10. Compare table-driven and directly programmed automata.
11. Which scanner generators do you know?

Lecture Programming Languages and Compilers SS 2007 / Slide 902

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Context-free grammars and syntactic analysis

12. Compare concrete and abstract syntax.
13. Describe the underlying principle of recursive descent parsers. Where is the stack?
14. How is tree construction achieved bottom-up in a recursive descent parser? how top-down?
15. What is the grammar condition for recursive descent parsers?
16. Why are bottom-up parsers in general more powerful than top-down parsers?
17. In which order do LR parsers construct derivations?
18. Which information does a state of a LR(1) automaton represent?
19. Describe the construction of a LR(1) automaton.
20. Which kinds of conflicts can an LR(1) automaton have?
21. Characterize LALR(1) automata in contrast to those for other grammar classes.
22. Describe the hierarchy of LR and LL grammar classes.
23. Which parser generators do you know?
24. Explain the fundamental notions of syntax error handling.
25. Describe a grammar situation where an LR parser would need unbounded lookahead.
26. Explain: the syntactic structure shall reflect the semantic structure.

Lecture Programming Languages and Compilers SS 2007 / Slide 903

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Attribute grammars and semantic analysis

27. What are the fundamental notions of attribute grammars?
28. Which tree walk strategies are related to attribute grammar classes?
29. What do visit-sequences control? What do they consist of?
30. What do dependence graphs represent?
31. What is an attribute partition; what is its role for tree walking?
32. Explain the LAG(k) condition.
33. Describe the algorithm for the LAG(k) check.
34. Which attribute grammar generators do you know?
35. How is type checking of expressions specified?
36. How is name analysis for C scope rules specified?
37. How is name analysis for Algol scope rules specified?
38. How is the creation of target trees specified?

Lecture Programming Languages and Compilers SS 2007 / Slide 904

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Binding of names

39. How are bindings been established explicitly and implicitly?
40. Explain: consistent renaming according to scope rules.
41. What are the consequences if defining occurence before applied occurence is required?
42. Explain where multiple definitions of a name could be reasonable?
43. Explain class hierarchies with respect to static binding.
44. Explain the data structure for representing bindings in the environment module.
45. How is the lookup of bindings efficiently implemented?
46. How is name analysis for C scope rules specified by attribute computations?
47. How is name analysis for Algol scope rules specified by attribute computations?

Lecture Programming Languages and Compilers SS 2007 / Slide 905

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Type specification and analysis

48. Which language properties are specified for a statically typed language?
49. What are the tasks of type analysis?
50. Give some characteristic properties of specific types.
51. What is coercion, and in which situations has the compiler to consider it?
52. How is overloading resolved?
53. How is type checking of expressions specified by attribute computations?
54. What are specific type analysis tasks for object-oriented languages?
55. What are specific type analysis tasks for functional languages?

Lecture Programming Languages and Compilers SS 2007 / Slide 906

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic

Dynamic semantics and transformation

56. Describe semantic domains for the denotational description of an imperative language.
57. Describe the definition of the functions E and C for the denotational description of an imperative language.
58. How is the semantics of a while loop specified in denotational semantics?
59. How is the creation of target trees specified by attribute computations?
60. PTG is a generator for creating structured texts. Explain its approach.

Lecture Programming Languages and Compilers SS 2007 / Slide 907

Objectives:

Questions for repetition

In the lecture:

Answer some questions for demonstration

Questions:

More questions can be found along with the slides of this topic