9. Domain Specific Languages (DSL)

(under construction)

10. Summary
Questions to check understanding

1. Language properties - compiler tasks

1.1. Associate the compiler tasks to the levels of language definition.

1.2. Describe the structure of compilers and the interfaces of the central phases.

1.3. For each phase of compiler frontends describe its task, its input, its output.

1.4. For each phase of compiler frontends explain how generators can contribute to its implementation.

1.5. What specifications do the generators of (1.4) take and what do they generate?

1.6. What data structures are used in each of the phases of compiler frontends?

1.7. Give examples for feedback between compiler phases.

1.8. Java is implemented differently than many other languages, e.g. C++, what is the main difference?

2. Symbol specification and lexical analysis

2.1. Which formal methods are used to specify tokens?

2.2. How are tokens represented after the lexical analysis phase?

2.3. Which information about tokens is stored in data structures?

2.4. How are the components of the token representation used in later phases?

2.5. Describe a method for the construction of finite state machines from syntax diagrams.

2.6. What does the rule of the longest match mean?

2.7. Compare table-driven and directly programmed automata.

2.8. Which scanner generators do you know?

3. Context-free grammars and syntactic analysis

3.1. Which roles play concrete and abstract syntax for syntactic analysis?

3.2. Describe the underlying principle of recursive descent parsers. Where is the stack?

3.3. What is the grammar condition for recursive descent parsers?

3.4. Explain systematic grammar transformations to achieve the LL(1) condition.

3.5. Why are bottom-up parsers in general more powerful than top-down parsers?

3.6. Which information does a state of a LR(1) automaton represent?

3.7. Describe the construction of a LR(1) automaton.

3.8. Which kinds of conflicts can an LR(1) automaton have?

3.9. Characterize LALR(1) automata in contrast to those for other grammar classes.

3.10. Describe the hierarchy of LR and LL grammar classes.

3.11. Which parser generators do you know?

3.12. Explain the fundamental notions of syntax error handling.

3.13. Describe a grammar situation where an LR parser would need unbounded lookahead.

3.14. Explain: the syntactic structure shall reflect the semantic structure.

4. Attribute grammars and semantic analysis

4.1. What are the fundamental notions of attribute grammars?

4.2. Under what condition is the set of attribute rules complete and consistent?

4.3. Which tree walk strategies are related to attribute grammar classes?

4.4. What do visit-sequences control? What do they consist of?

4.5. What do dependence graphs represent?

4.6. What is an attribute partition; what is its role for tree walking?

4.7. Explain the LAG(k) condition.

4.8. Describe the algorithm for the LAG(k) check.

4.9. Describe an AG that is not LAG(k) for any k, but is OAG for visit-sequences.

4.10. Which attribute grammar generators do you know?

4.11. How is name analysis for C scope rules specified?

4.12. How is name analysis for Algol scope rules specified?

4.13. How is the creation of target trees specified?

5. Binding of names

5.1. How are bindings established explicitly and implicitly?

5.2. Explain: consistent renaming according to scope rules.

5.3. What are the consequences if defining occurence before applied occurence is required?

5.4. Explain where multiple definitions of a name could be reasonable?

5.5. Explain class hierarchies with respect to static binding.

5.6. Explain the data structure for representing bindings in the environment module.

5.7. How is the lookup of bindings efficiently implemented?

5.8. How is name analysis for C scope rules specified by attribute computations?

5.9. How is name analysis for Algol scope rules specified by attribute computations?

6. Type specification and analysis

6.1. What does „statically typed“ and „strongly typed“ mean?

6.2. Distinguish the notions „type“ and „type denotation“?

6.3. Explain the taxonomy of type systems.

6.4. How is overloading and coercion specified in Eli?

6.5. How is overloading resolved?

6.6. Distinguish Eli’s four identifier roles for type analysis?

6.7. How is type analysis for expressions specified in Eli?

6.8. How is name equivalence of types defined? give examples.

6.9. How is structural equivalence of types defined? give examples.

6.10. What are specific type analysis tasks for object-oriented languages?

6.11. What are specific type analysis tasks for functional languages?

7. , 8. Dynamic semantics and transformation

7.1. What are denotational semantics used for?

7.2. How is a denotational semantic description structured?

7.3. Describe semantic domains for the denotational description of an imperative language.

7.4. Describe the definition of the functions E and C for the denotational description of an imperative language.

7.5. How is the semantics of a while loop specified in denotational semantics?

7.6. How is the creation of target trees specified by attribute computations?

7.7. PTG is a generator for creating structured texts. Explain its approach.