

Introducing Natural Language Examples in a Course on Compiler Principles

Yin Chen

Abstract “Compiler principles” is widely regarded as the most difficult specialized course in software engineering major because of its difficult theory and abstract content. This chapter discusses how to introduce natural language examples into the classroom teaching, and therefore liberate students from abstract theory explanation.

Keywords Compiler principles • Classroom teaching • Natural language

1 Introduction

“Compiler principles” is an important specialized course of software engineering major. It is commonly regarded as the most difficult course to teach and learn because of its difficult theory and abstract content. So teachers should give some intuitional examples to students for better understanding. This chapter introduces some natural language examples into the course compiler principles. Natural language, which is relative to programming language, refers to the language people use in daily life. It is the language of human society, and is the language people most familiar with. In fact, programming language is built on the abstraction of natural language. The manners computer analyzes and processes programming language are similar to the manners people understand and use natural language. So, it couldn’t be better if we use natural language examples in course teaching.

Y. Chen (✉)

School of Software, Harbin Institute of Technology, Harbin, China
e-mail: chenyin@hit.edu.cn

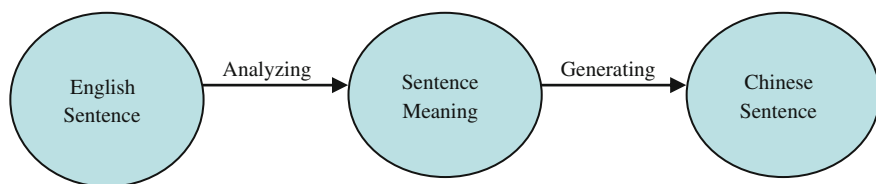


Fig. 1 English-Chinese translation process

This chapter attempts to introduce natural language examples into the course. Two examples shown in this chapter demonstrate that this method can liberate students from the abstract theory explanation.

2 A Natural Language Example in Course Introduction

In theory, compiling process usually includes five stages, which are lexical analysis, syntax analysis, semantic analysis, codes optimization and target code generation. This is usually the content teachers narrate in the course introduction. It is not difficult for students to memorize these five stages, but it is a little difficult for them to understand the dividing evidence for these stages. Teachers should advocate students to memorize based on understanding. Students need to know not only what it is, but also why it is so.

The main task of Compiler is to translate the source program written in programming language into equivalent target program written in machine language or assemble language. Since compiling essentially is a translation process, its working process is similar to foreign language translation. The only difference is that source language and target language are different [1].

Now, let us see what steps will be needed if we translate the English sentence “In the room, he broke a window with a hammer.” into Chinese (we may adopt heuristic method in course teaching which requests students themselves to think and summarize).

Generally, the translating procedure includes two steps (see Fig. 1). Firstly, we need to analyze the meaning of the source language (English) sentence which is called semantics. Next, we construct equivalent target language (Chinese) sentence based on the given semantics. The first step can be called “analysis procedure” which is from source language to semantics, this procedure only involves source language, but is independent of target language. The second step can be called “generating procedure” which is from semantics to target language, this procedure only involves target language, but is independent of source language. So, semantics is an intermedia which is independent of concrete language.

Analysis process can be further divided into several steps. Generally speaking, apart from core predicate verb, a sentence also includes some noun components,

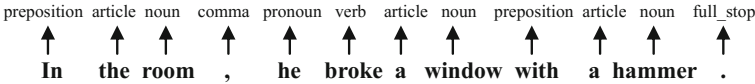


Fig. 2 Lexical analysis of the sentence

which are called entities. The essence of semantic analysis is to determine the relationship between these noun components and the core predicate verb. These relationships are called “case” in linguistics, including agentive case, dative case, time case, locative case, instrumental case, etc. Therefore, for a complex sentence, in order to acquire its meaning, we should firstly identify the core predicate verb and noun components in the sentence; in order to identify these components, we should firstly analyze the sentence structure (this process is called “syntax analysis”); in order to analyze the sentence structure, we should firstly determine the part-of-speech of each word in the sentence (this process is called “lexical analysis”). So, the analysis process can be further divided into three steps, which are lexical analysis, syntax analysis and semantic analysis.

For the above example, after lexical analysis, we get part of speech sequence corresponding to the sentence as shown in Fig. 2.

Next, according to English grammar, we get the structure of this sentence as shown in Fig. 3. According to the syntax analysis result, we can identify the core predicate verb and noun phrases in the sentence.

Next, we would analyze the sentence’s meaning according to semantic rules. In linguistics, case grammar [2] is used to express the semantic concept in the deep structure of grammar system. Case grammar is a linguistic theory proposed by the American linguist Fillmore in 1966 [3]. Simply put, the case grammar can be regarded as grammar with case. For example, if the English case grammar has such a semantic rule as shown in Fig. 4.

Then, we can get the meaning of the above English sentence. That is, the core event described by the sentence is “break”, the agent of this event is “he”, the object of this event is “a window”, the instrumental used in the event is “a hammer”, location of the event is “In the room”. Semantic results of case grammar may be expressed by case framework. For example, the semantic analysis result of the above sentence is represented by the following case framework.

```
[break
  [case-frame
    agent: he
    object: a window
    locative: in the room
    instrument: a hammer
  ]
[modals
```

Fig. 3 Syntax analysis of the sentence

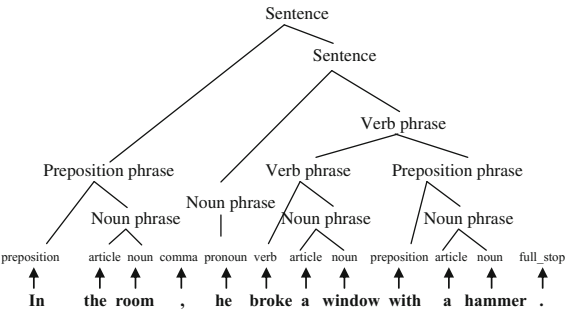
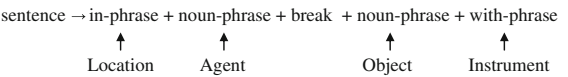


Fig. 4 Example for semantic rule



time: past
voice: active

]

]

Now the analysis process has been accomplished, and the generating process is just opposite process to the analysis process, which constructs the structure of the Chinese (target language) sentence and the concrete word sequence, based on the case frame.

In fact, the principle of compiler is similar to translation process. It is also divided into these stages as lexical analysis, syntax analysis, semantic analysis and target code generation. Their difference lies in the following aspects:

1. In English-Chinese translation, the task of lexical analysis is to determine the type of each word in the sentence. Here, word type refers to part-of-speech including nouns, verbs, adjectives, prepositions, pronouns, adverbs, etc.; The task of compiler's lexical analysis is also to recognize words and words' types in the source program. However, word's type here includes keyword, identifier, operator, bounded symbol, constant etc.
2. In English-Chinese translation, the task of syntax analysis is to analyze the structure of English sentence, and to identify the core predicate verb and noun components; However, the task of compiler's syntax analysis is to analyze structure of program language (source language)'s sentence, and to identify the key part (such as "var", "proc", ":", "if", "else", "switch", "while", "for", "call") and other components of the sentence.
3. In English-Chinese translation, the task of semantic analysis is to determine the relationship between each noun component and the core predicate of English sentence based on semantic rules (case grammar), and therefore represent the

Table 1 Analogy between English-Chinese translation and compiling process

English-Chinese translation		Compiling
Lexical analysis	Determine part-of-speech (nouns, verbs, adjectives, prepositions, pronouns, adverbs, etc.) of each word in English sentence	Recognize words and words' types (keywords, identifiers, operator, bounded symbol, constant etc.) in source program
Syntax analysis	Analyze the structure of English sentence, and identify the core predicate verb and noun components	Analyze structure of program language's sentence, and identify the key part (such as "var", "proc", ":", "=", "if", "else", "switch", "while", "for", "call") and other components of the sentence
Semantic analysis	Determine the relationship between each noun components and the core predicate of english sentence based on semantic rules (case grammar), and therefore represent the analysis result by case framework	Transfer the source program into intermediate code according to syntax-directed definitions (SDD)
Target code generation	Construct Chinese sentence's structure and the concrete word sequence based on case frame	Generate machine language or assembly language program according to intermediate code

analysis result by case framework; However, the task of compiler's semantic analysis is to transfer the source program into intermediate code (such as three address instruction) according to syntax-directed definitions (SDD). Similarly to case framework, intermediate code has fixed format and is independent of concrete language. In three address instruction, the operator is corresponding to the core predicate verb of natural language sentence, and each operand (source, target) is corresponding to various case of core predicate verb.

4. In English-Chinese translation, the task of generating process is to construct Chinese (target language) sentence's structure and the concrete word sequence based on case frame; However, the task of compiler's generating process is to generate machine language or assembly language program according to intermediate code.

The above analogical processes are summarized in Table 1. Through the analogy between compiler's stages and natural language translation process, students can deeply understand the dividing evidence for compiler's stages, and thus better understand the working process of compiler.

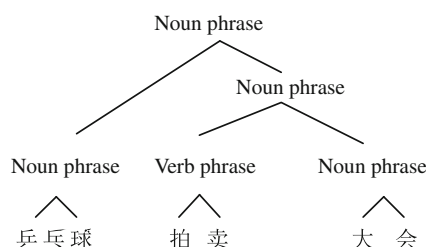
3 A Natural Language Example in “Handle” Teaching and Learning

“Handle” is also an abstract concept in learning process of students. The key problem of bottom-up analysis is how to correctly recognize handle. Simply put, handle is the symbol string that should be reduced each time in bottom-up analysis process. Handle is the body (right part) of a production. In the LR analysis method, handle is defined as “left-most direct phrase”. “Left” is easy for students to understand, so the key problem is how to correctly identify direct phrase. According to the definition of a direct phrase—assuming $S \Rightarrow * \alpha \gamma \beta$ (i.e. $\alpha \gamma \beta$ is sentence pattern), if $S \Rightarrow * \alpha A \beta$, and $A \Rightarrow \gamma$ (i.e. $A \rightarrow \gamma \in P$), then γ is called direct phrase of sentence pattern $\alpha \gamma \beta$ corresponding to variable A —it can be seen that direct phrase must be right part of a certain production. However, on the contrary direction, the right part of a production is not necessarily a direct phrase (it can be called direct phrase only when it meet the “if...” conditions). That is to say “right part of a production” is a necessary condition of “direct phrase”, but not a sufficient condition. So, in bottom-up analysis process, it is not always the case to perform the reduction operation when meeting the right part of a production. The right part can be reduced only when it meets certain conditions and is assuredly a direct phrase. In order to illustrate in what circumstances an incorrect handle will be incorrectly recognized, we again introduce a natural language example to help students understand it.

For example, if we have the following Chinese grammar:

1. noun phrase \rightarrow noun phrase + noun phrase
2. noun phrase \rightarrow “大” + “会”

Fig. 5 Parse tree of the sentence “乒乓球拍卖大会”



3. noun phrase → “乒乓” + “球”
4. noun phrase → “乒乓” + “球” + “拍”
5. verb phrase → verb phrase + noun phrase
6. verb phrase → “卖”
7. verb phrase → “拍” + “卖”

For the sentence “乒乓球拍卖大会”, according to the parse tree shown in Fig. 5, “乒乓球” and “拍 卖” is its direct phrase. However, “乒 乓 球 拍” and “卖” are not the sentence’s directly phrases although they respectively are right part of productions (4) and (6). But In some other sentences they may be directly phrases, such as in the sentence “商店里的乒乓球拍卖完了”. This is why the definition of direct phrase emphasizes that direct phrase is relative to a specific sentence pattern.

4 Conclusions

“Compiler principles” is commonly regarded as the most difficult specialized course in software engineering major because of its difficult theory and abstract content. This chapter attempts to introduce natural language examples into the course. Two examples shown in this chapter demonstrate that this method can liberate students from the abstract theory explanation.

References

1. Li, D., & Shi, H. (2008). Study and exploration of the teaching methods for course "Compiler Principle". *Computer Education*, 8, 103-104.
2. Fillmore, C. J. (1969). Towards a modern theory of case. In D. Reibel & S. Shane (Eds.), *Modern studies in English* (pp. 361-375). Englewood Cliffs, N.J.: Prentice Hall.
3. Fillmore, C. J. (1968). The case for case. In E. Bach & R. Harms (Eds.), *Universals in linguistic theory*. New York: Holt, Rinehart and Winston.

Software Engineering Education for a Global E-Service
Economy

State of the Art, Trends and Developments

Motta, G.; Wu, B. (Eds.)

2014, XV, 162 p. 36 illus., Hardcover

ISBN: 978-3-319-04216-9