Presentation 33

Department of Computer Science
University of Petrozavodsk

Using Formal Grammars for Describing Documentation Set of a Department

Timophey V. Nazarov, Dr. Yury A. Bogoyavlenskiy

Timophey.Nazarov@cs.karelia.ru, Yury.Bogoyavlenskiy@cs.karelia.ru

Abstract

The goal of this work is to apply the mechanism of the formal grammars for describing documentation set of a department. So we get the concept of the metastructure. The metastructure of a department is a structure of structures of some objects. For putting in order this set we use structuring of the documents by hierarchical sections. On level of the formal grammars this lead to using of substitution grammars. An example of this approach is presented.

MOTIVATION

Our task is to separate the structure of a document set from implementation details and tools. In our case we need to describe array of documents: a structure of structures. Thus we use concept of *metastructure*.

Practical tasks examples:

- Getting the course work list: student, topic, science advisor
- Getting information about department staff
- Getting extended information about department staff (for privileged users)

CONCEPT

1. Example

```
<department>::=<staff><students>...

<staff>::=<staff member>+
<staff member>::=<name><e-mail>
        <rank><appointment>

<students>::=<student>+
<student>::=<name><e-mail><group number>

...
```

2. Grammar properties

- regularity

  We are facing with groups of similar objects. For example, students groups.

```
<students>::=<student>+
```

- finite-choice

  In each case we have concrete result. For example,

```
<staff>::=<staff member>+

<staff member>::=<name><e-mail>
 <rank><appointment>

<staff> => <staff member> =>
 <name><e-mail><rank><appointment> =>
 "Ivan Petrov" "ivan@petrov.ru"
 "master of mathematics" "lecturer"
```

- hierarchy

  In practice we have a lot of parts of documentation (staff, students, etc.) So we are to structure this information at formal level. We can do this by grammar substitution mechanism. In our case it looks as follows.

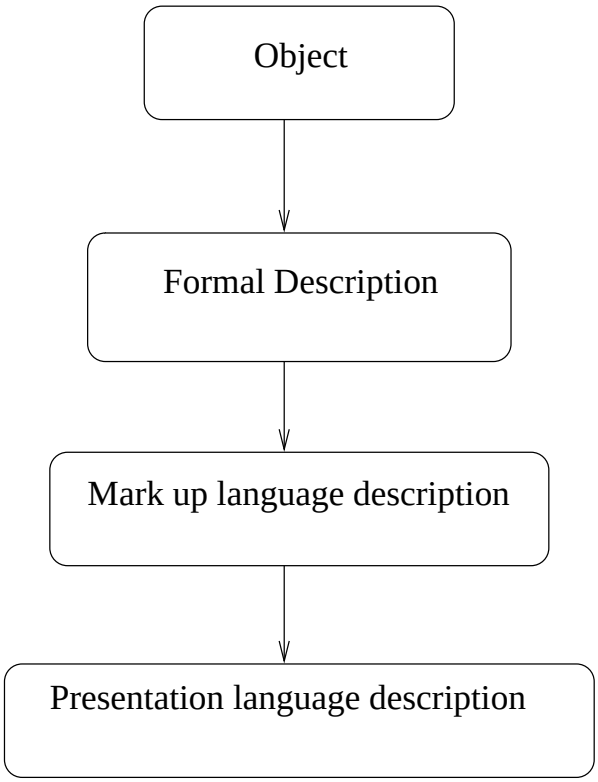
```
<department>::="staff" "students"...
```

So we draw out non-terminals `staff` and `students` as terminal symbols. In sub-grammars we use ones as non-terminals.

```
<staff>::=<staff member>+
```

Finally, in grammar operations we put non-terminal `staff` at main grammar as terminal symbol making substitution.

3. Object abstract level scheme



CONCLUSION

- Approach advantages:
  1. It allows to reduce labour intensity of other components of information system.

2. It allows to use powerful mechanism of the formal grammars.
3. It separates description of a structure from realization details and tools.

Disadvantages: labour intensity in making grammars.

- Current state
  - We applied the idea of using formal grammars to documentation set of the department of computer science.
  - We choose grammars with concrete properties.
  - We demonstrated the suggested approach on example of the department of computer science of Petrozavodsk state university.