

Forth and German Academia Report of a Field Trip

Klaus Schleisiek
kschleisiek at freenet.de

The Event

38th workshop of the **Programming Languages and Computation Concepts** section of Gesellschaft für Informatik.

At present, **Type Checking** is the pig that is chased thru the computer science village.

After it became clear to me that this implies an **automatic stack checker for Forth**, I am all for it.

One presentation dealt with the design of an extensible language. The chosen implementation strategy was very complex.

New Trend

It seems that there is a growing discontent of conventional compiling strategies that use **Phrase Structure Grammar**, characterized by BNF specifications.

Instead **Dependency Grammar** based on a **dictionary** or **lexicon** is considered to be the more flexible approach.

It has been shown that both **phrase structure grammar** and **dependency grammar** cover the same set of linguistic constructs, namely **context free grammars**.

Scheme, Forth, Prolog, Smalltalk, APL, and LISP are examples of **dependency grammar systems**.

See: <https://dl.acm.org/doi/10.1145/3133850.3133859>

Open Issues

In the paper, these topics are considered "open issues":

1. User Defined Data Structures
 - **Create ... Does>**
2. Nesting Lexicons and Introducing Scopes
 - **Vocabulary Tree**
3. Handling Ambiguity
 - **Vocabularies and Redefinitions**
4. Dynamic Binding
 - **evaluate**
5. Higher Order Words (Metaprogramming)
 - **immediate**

Academia and Forth

The academic world does not know about the simplicity of the Forth approach.

Therefore, I am going to hold a presentation next year:

"Poor Man's Compilers - How Forth Treats its Source Code"

Conclusion

The **academic computer science** and the **Forth** communities use different terminology.

We don't understand each other.

We have to learn their terminology in order to be understood.

volksForth will be re-engineered to serve as a **Model Forth System** in order to understand how it works.