Computational Linguistics Vice Versa

Sebastian Lohmeier
Technische Universität Berlin
sl@monochromata.de

25. PPIG, Doctoral Consortium, June 25, 2014

What do I want to know?

- How does source code structure influence program comprehension?
- Can NL text structures shorten source code without making it harder to comprehend?

What is programming, then?

Computational Linguistics

–

Linguistics of Programming

What is programming, then?

Computational Linguistics

–

Cognitive Linguistics of Programming

What is programming, then?

Computational Linguistics

–

Cognitive Linguistics of Object-Oriented Programming

What is programming, then?

Use code structure to influence program comprehension:

```
void addOne(ServiceRegistrar reg) {
  new RegistrarMenu(host, reg.getServiceID());
}
```

vs.

```
void addOne(ServiceRegistrar reg) {
  new RegistrarMenu(host, .ServiceID);
}
```

What is programming, then?

A cognitive model

- based on the information-system metaphor of cognition
- integrates information from long-term memory
- the parts of the representation have activation values
- activation influences fixation durations and regressions

A formal model (a compiler)

- is restricted to detect and reject referential ambiguity

What about naturalistic programming?

- “the primitive abstractions in programming languages should be drawn from the study of Natural Languages”;
- “We don’t advocate implementing English! The languages we are proposing are naturalistic, but not natural.” (Lopes et al., 2003, 199,204)

```
create (a hidden file with "hello.txt" as
    name)
```

(Knöll et al., 2011, 37)

What to do?

- *Implement* a cognitive and a formal model of (indirect) anaphors in Java
- Shorten source code, but make it harder to understand for some programmers
- Use eye tracking to figure out when background knowledge is active enough to permit comprehension of the indirect anaphor
- Create a source code editor that only displays indirect anaphors when background knowledge is sufficiently active

- I.e. apply cognitive linguistics to implement a new programming language feature

Discussion

*

References I

Knöll, R., Gasiunas, V., and Mezini, M. (2011). Naturalistic types. In *ONWARD '11: Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software*, pages 33–47.

Lopes, C. V., Dourish, P., Lorenz, D. H., and Lieberherr, K. (2003). Beyond AOP: toward naturalistic programming. *SIGPLAN Notices*, 38(12):34–43.