Activation and the Comprehension of Indirect Anaphors in Source Code

Sebastian Lohmeier

M.Sc. Informatik degree course
Technische Universität Berlin
sl@monochromata.de

Keywords: POP-II.B. program comprehension, POP-III.B. new language, POP-V.A. linguistics and mental models, POP-V.B. eye movements and questionnaire

Abstract

A reading experiment is proposed that combines eye tracking and a questionnaire to investigate the effects of the activation of background knowledge on the understanding of indirect anaphors in source code compared to reading Java source code with local variables and qualified expressions.

1. Introduction

Means of referring backwards to what was previously mentioned in a text are called *direct anaphors* in linguistics. Pronouns like *she*, *this*, and definite noun phrases like *the hard disk* can function as direct anaphors. While expressions like `this` exist in programming languages, they do not allow the programmer to choose a previously mentioned entity but work with a fixed entity instead. Lopes, Dourish, Lorenz, and Lieberherr (2003) and Knöll and Mezini (2006) proposed to include anaphors in general-purpose programming languages but did not cover examples like the following, in which *the hard disk* functions as an indirect instead of a direct anaphor:

(1) The computer did not boot. The hard disk had been formatted.

Here, *the hard disk* refers to a part of *the computer*. Indirect anaphors underspecify well-known information (e.g. that computers have hard-disks): the underspecified relation is not encoded in the text, but is taken from world knowledge and added to the reader's mental model of the text's meaning. Lohmeier (2011) describes an implementation of indirect anaphors in an extension of the Java programming language. With the experiment, I seek to identify when a relation learned from source code is well-known to an individual programmer. If that is the case, repeated statements of the relation could be omitted by use of an indirect anaphor when presenting source code to this programmer. Before the experiment is described, relevant prior work from linguistics is summarised, that was used to develop the experiment.

2. Linguistic Background

The proposed experiment is based on experiments on indirect anaphors, activation, and cohesion.

2.1. Indirect Anaphors

There are many ways of characterising indirect anaphors, which are also called bridging anaphors, associative anaphors or accommodations. I use the cognitive model of anaphors from Schwarz-Friesel (2007), which is based on a three-level semantics proposing that a reader constructs a mental representation while reading a text that integrates context-independent conceptual and lexical-semantic information from long-term memory in a text-world model (TWM) that represents context-specific current meanings in nodes and relations between them. In three-level semantics, words refer to nodes in the TWM which are called *referents*. Example (1) from above can be used to exemplify the model of Schwarz-Friesel. The indirect anaphor *the hard disk* refers to a TWM-node that I call HARD-DISK-1. This node is connected to a COMPUTER-1 node that was established when its so-called anchor *the computer* was read.

Both nodes are connected by a PART-OF-1 link from HARD-DISK-1 to COMPUTER-1 that is created through subconscious retrieval from the context-independent concept COMPUTER after reading *the hard disk*. The PART-OF-1 link is also said to function as *anaphora* relation. It increases the *coherence* of the sentence, the mental connection of the referents of a text. Because the *PART-OF-1* link is not expressed in the text (as could be done by using *its hard disk* instead of *the hard disk*), it is said to be *under-specified*. Using *its* instead of *the* would not only yield *coherence* but also *cohesion*: it would be visible in the text that it is coherent. Besides other forms, Schwarz-Friesel distinguishes between indirect anaphors based on part-of (i.e. meronymic) relations like in (1) and indirect anaphors that fill a verb role, as *the chalk* in

(2) The teacher was busy writing an exercise on the blackboard. However, she was disturbed by a loud scream from the back of the class and the chalk dropped on the floor. (Garrod & Terras, 2000, 529)

where the referent of *the chalk* fills the INSTRUMENT role of the referent of its anchor *writing*. Example (2) shows an excerpt of one of the 24 texts that Garrod and Terras (2000) constructed to consolidate earlier research by comparing the understanding of indirect and direct anaphors as well as dominant and non-dominant instruments (besides further conditions). (2) is an example of a non-dominant indirect anaphor. When *with chalk* is appended to the first sentence in (2), the anaphor *the chalk* in the second sentence will pick up that mention of chalk and be a direct anaphor instead of an indirect one. The texts for the conditions with dominant anaphors used a pen instead of chalk because writing is more commonly performed with a pen than with chalk. Besides fixation duration, Garrod and Terras (2000) measured regression-path reading-time for a word, that is the sum of the durations of all fixations from the initial fixation on the word, until the eyes fixate a word to the right of the word. They found that for dominant relations like the one between *writing* and *pen*, subjects' regression-path reading-times separately calculated for the anaphor and for the following verb were not longer for indirect anaphors than for direct anaphors. For non-dominant relations, like the one between *writing* and *chalk*, regression-path reading-times on the verb following the anaphor were 48 ms longer on average for indirect anaphors than for direct anaphors. This means that in terms of regression-path reading-times, the comprehension of indirect anaphors is harder than understanding of direct ones if the relation between the anaphor and anchor is not well-known. This argument requires a task design that avoids problem solving so that delayed effects on eye movements stem from discourse integration only and not from e.g. resolving logical puzzles. It can then also be assumed that the *midas touch* problem does not occur where subjects gaze at a certain location while thinking about a problem instead of perceiving their environment.

2.2. Activation

How well a relation is known may be modelled via its activation in mind, a theoretical construct that describes how probable and fast recall of a referent may be. Rayner, Raney, and Pollatsek (1995) report that words are read faster at later re-occurrences in a text: gaze durations for rare words were about 50 ms shorter from the third encounter on, compared to the initial occurrence; gaze durations for frequent words were about 10 ms shorter from the second encounter on. The dominance of a relation between a verb and an instrument could therefore be manipulated by repeatedly exposing subjects to expressions of the relation in texts.

2.3. Cohesion

The use of indirect anaphors reduces the cohesion of a text, i.e. does not show in the text the coherence relations a reader is to establish during reading. In a study to replicate the so-called *reverse cohesion effect*, O'Reilly and McNamara (2007) increased the cohesion in a biology text on cell mitosis presented to college students by replacing pronouns by noun phrases as well as by adding elaborations and sentence connectives. Subjects answered comprehension questions that asked for information from a single sentence (text-based questions) or required integration of information from two sentences (inference-based questions). O'Reilly and McNamara (2007) found that readers with high prior knowledge answered text-based questions better for low cohesion texts than for high-cohesion texts if their comprehension skill was low. They explain this *reverse cohesion effect* by hypothesising that low-skill readers skim more and therefore likely miss more detail in high-cohesion texts compared to low-cohesion texts. While the effect was not present for readers with high comprehension skill, the study confirmed the possibility that reduced cohesion can, depending on prior knowledge and comprehension skill, potentially improve comprehension as measured by comprehension questions. The same effect may occur when cohesion is reduced in source code through use of indirect anaphors subject to prior knowledge and (program) comprehension skill.

3. Indirect Anaphors in Object-Oriented Programming

As has been done in Lohmeier (2011), class declarations in object-oriented source code, e.g. in Java, can be interpreted as conceptual structures: the field and accessor method declarations of a class can describe part-of relations. Likewise, the arguments and the return type from a method declaration can be interpreted as verb roles that are used to anchor an indirect anaphor in a method invocation expression which is based on the declaration of the method. An indirect anaphor is anchored in a method invocation expression when the returned value is not assigned to a local variable – otherwise the anaphor would be a direct one and be related to the local variable. To anchor indirect anaphors in arguments of a method invocation expression, it would be necessary to omit method arguments, and use some kind of default value, which is not possible in Java. As in Lohmeier (2011), I will therefore use the return value only to anchor indirect anaphors in method invocations. In an extension of the Java programming language an indirect anaphor has a type and can be anchored in expressions with a type that declares a field or accessor method of the type of the anaphor. An indirect anaphor can also be anchored in a method invocation expression if the return type of the method is compatible to the type of the anaphor. Indirect anaphors are marked with a dot preceding the type name used to find an anchor. E.g. `.ServiceID` can anchor in a method invocation that returns a `.ServiceID` or in an expression of a type that declares a field of type `ServiceID` or defines an accessor that returns a `ServiceID`. The activation of the relation between indirect anaphor and anchor can be manipulated by varying how often or how recently programmers were shown the declaration or use of such fields or (accessor) methods. Direct anaphors can also be used in source code. Their syntax is identical to the syntax of indirect anaphors. The type name that is part of the direct anaphor is used to find an expression or parameter of that type within the method body or constructor body that contains the anaphor. The value of the expression will at runtime be used as the value of the direct anaphor. Direct anaphors in source code enable repeated reference to values made accessible via indirect anaphors. Direct anaphors are introduced together with indirect anaphors in this experiment because omitting them removes a potential source of confusion lying in the fact that direct and indirect anaphors share the same syntax but are understood differently.

4. Experiment

Participants are assigned to one of four groups that balance potential position effects of item sequence (items 1-20 followed by items 21-40 vs. items 21-40 followed by items 1-20) and sequence of exposition to target expression type (using indirect anaphors in the first and not the second block of items vs. not using them in the first and using them in the last block). The experiment comprises the following factors: a between-subject factor of program comprehension skill (high vs. low), as well as three within-subject factors of target expression type (indirect anaphors vs. local variables or qualified access), the activation (high vs. low) of the target relation that is under-specified in indirect anaphors and given in the source code otherwise, and the type of comprehension questions (text-based vs. inference question). I do not treat the different kinds of indirect anaphors (anchored in a field or accessor vs. anchored in the return value of a method) as separate conditions because both cases involve the under-specification of a direct relation between indirect anaphor and antecedent. The following dependent variables will be analysed: regression-path reading-time on the target expression, regression-path reading-time on the word following the target expression, task duration (the time a subject takes to complete a task), and error rate in comprehension questions. Based on the review of prior work in linguistics, I expect the following effects.

A. Regression-path reading-time on an indirect anaphor or the following word will be shorter, the more active – i.e. more recently and frequently presented – the underspecified relation, analogous to the dominance effect found by Garrod and Terras (2000).
B. If a target relation is highly activated, regression-path reading-times will be equivalent for both types of target expressions, like the results of Garrod and Terras (2000) for dominant verb arguments.
C. For programmers with low program comprehension skill and for highly activated relations, the error rate for text-based comprehension questions is expected to be lower for the test tasks with indirect anaphors than for the control tasks with local variables and qualified expressions, in line with the findings of O'Reilly and McNamara (2007) for readers with high background knowledge and low comprehension skill. High background knowledge is here equated with high activation of relations.
D. For highly active relations, duration could be lower for test tasks than for corresponding control tasks because under-specification reduces the amount of text to be read and Garrod and Terras (2000) found that indirect anaphors for dominant relations are gazed at as long as the control targets.
E. Alternatively, task duration could be higher for test tasks with anaphors than for corresponding control tasks, if indirect anaphors are generally harder to understand than normal Java code.

4.1. Participants

I intend to test 30 subjects with practical experience in larger software projects using Java or Scala – both students enrolled in a Master of computer science degree course and professional programmers.

4.2. Apparatus and Procedure

Participants are seated in a dedicated laboratory room in front of an SMI iView X Hi-Speed 1250 tower-mounted eye tracker using a scan rate of 500 Hz. The stimuli are displayed in a customised version of the Eclipse IDE that uses a plug-in (http://monochromata.de/eyeTracking/) to control the eye tracker and correlate eye movement data with words displayed in the IDE's editor with syntax highlighting. The Eclipse IDE is customised to hide the application toolbar and disable markers and annotations in the editors (e.g. for errors shown because the editor is not able to parse indirect anaphors). The editors are put into a disabled state that removes the context menu and prevents text selection.

After being welcomed, subjects complete a program comprehension skill questionnaire. The results of the questionnaire are used to balance the average program comprehension skill questionnaire score between the four groups of the experiment. An introduction to anaphors in source code and a test on anaphors in source code are performed before the eye tracker is calibrated and the participants perform the main test that comprises 40 items. To limit problem solving during the main test in favour of reading comprehension, subjects are instructed to work quickly, but not quicker than is required to work accurately. Subjects are informed that they are going to read source code that does not contain any errors. After the main test, subjects complete comprehension questions and have 5 minutes to write a short summary of important relations in the source code. The procedure takes about 90 minutes per subject.

4.3. Materials

The items were derived from source code of the Apache River project (http://river.apache.org/) that is sufficiently specialised not to be known by subjects and to enable an effective manipulation of activation levels. Subjects of all groups receive 40 items each; 32 of the items contain anaphors. The average activation of the underspecified relations of indirect anaphors is balanced between different kinds of indirect anaphors, as are the number of occurences of the different kinds of indirect anaphors. Four fixed lists of items are used. The items with direct and indirect anaphors were constructed by replacing local variables and qualified expressions by indirect anaphors and by replacing local variable declarations and uses of local variables by direct anaphors. The items with indirect anaphors were arranged such that each low-activation relation was explicated only once, three items before the item with the indirect anaphor under-specifying the relation. Each high-activation relation was explicated more than once in

```
private void addOne(ServiceRegistrar registrar) {
  LookupLocator loc = registrar.getLocator();
  String host = loc.getHost();
  if (loc.getPort() != Constants.discoveryPort)
    host += ":" + loc.getPort();
  JRadioButtonMenuItem reg =
        new RegistrarMenuItem(host, registrar.getServiceID());
  reg.addActionListener(wrap(new Lookup(registrar)));
}

private void addOne(ServiceRegistrar registrar) {
  LookupLocator loc = registrar.getLocator();
  String host = loc.getHost();
  if (loc.getPort() != Constants.discoveryPort)
    host += ":" + loc.getPort();
  JRadioButtonMenuItem reg =
        new RegistrarMenuItem(host, .ServiceID);
  reg.addActionListener(wrap(new Lookup(registrar)));
}
```

Fig. 1. A method that is part of the source code of an item with a control target expression (a qualified expression, top) and test target expression (bottom). The bottom shows the test target expression `.ServiceID` that functions as an indirect anaphor. The indirect anaphor is anchored in the expression `registrar` that has a type that declares an accessor method `getServiceID()` that returns a `ServiceID` instance.

items preceding the item containing the indirect anaphor. Each item is followed by a yes-no question with feedback on the correctness of the response to keep subjects focussed on the task. The question is displayed when the subject clicks a button to replace the source code with the yes-no question. Subjects are able to freely view the source code of the current item, of all previous items and the yes-no question of the current item repeatedly by clicking buttons to show them. Presentation of an item and its yes-no question is finished when the user clicks either the *yes* or the *no* button to register her answer. Task duration of an item is defined as the time a subject takes to answer the yes-no question since the first time the source code of the item was displayed.

The source code of each item fits into the editor without requiring scrolling. It contains a package declaration and class or interface declaration with a single method or constructor declaration. Import statement are collapsed and cannot be expanded. All comments have been removed. Statements were inserted or modified in order to control confounding variables. See Figure 1 for an example method declaration from an item. Code from a class or interface declaration can occur in more than one item, with or without partial overlap between the items. The distance between anchor and indirect anaphor was balanced so that the anchor could not be identified solely based on its distance from the indirect anaphor. Because the experiment will be conducted in Germany and I do not assume that there are valid sources of word frequencies for German programmers reading English source code, I did instead balance the frequencies of words used as indirect anaphors and anchors within the text comprised of all items. I also balanced the length of words used as indirect anaphors and anchors.

The comprehension questionnaire contains 20 open-ended questions, 10 text-based and 10 inference-based ones. 5 text-based and 5 inference-based questions target or include knowledge of the relations under-specified in indirect anaphors in the test tasks resp. the relations explicated in the control tasks.

4.4. Data Analysis

The planned analysis comprises the effect of target expression type, program comprehension skill, question type and activation ($2 \times 2 \times 2 \times 2$) on error rate, the effect of target expression type and activation (2×2) on regression-path reading-time of the target expression or the subsequent word and the effect of target expression type and activation (2×2) on task duration.

5. Discussion

Because I want to examine basic effects during reading, I do not aim at high external validity. Consequently, I do not claim that the results are directly transferrable to programming tasks that involve both reading and writing; I do not consider how often indirect anaphors identical to the ones I tested would occur in source code written by programmers themselves; I do not speculate about the frequency with which the activation levels I created through my manipulation occur in practice. Such aspects may be studied in the future.

A number of relevant outcomes are expected from the experiment. Data collected for hypothesis A will show whether there is a measurable and a significant difference on-line between indirect anaphors with under-specified relations at different activation levels. Data for hypothesis B will permit an initial statement on whether there are forms of indirect anaphors that are not harder to understand for programmers than are local variables and qualified expression used instead of the indirect anaphors. Data for hypothesis C will supplement this on-line result with off-line data from comprehension questions that will show whether certain forms of indirect anaphors can improve comprehension for certain kinds of programmers. Data for the two alternative hypotheses D and E will show whether these comprehension-related effects are paralleled by an overall decrease or increase of time duration. If data for hypotheses B and C shows that indirect anaphors are beneficial in certain situations, it will be necessary to investigate how the presentation of source code in integrated development environments like Eclipse can be personalised. Display could be personalised by continuous gathering of eye tracking data to determine whether the relation under-specified in a given indirect anaphor can be well-known enough to make a specific programmer benefit from the presentation of the indirect anaphor at that point in time. In such cases, indirect anaphors will be displayed by the IDE instead of local variables or qualified expressions. In other situations, when it is assumed that there will be comprehension difficulties arising from the use of the indirect anaphor that do not support the work of the programmer, a local variable or a qualified expression will be displayed instead of the indirect anaphor.

6. Acknowledgements

I thank Nele Russwinkel for providing the eye tracker used to prepare the experiment and Arthur Jacobs who provides the eye tracker for the experiment. I am grateful to Lutz Prechelt who took a lot of time to discuss the experiment and to the three anonymous reviewers for their helpful comments.

7. References

Garrod, S., & Terras, M. (2000). The contribution of lexical and situational knowledge to resolving discourse roles: Bonding and resolution. *Journal of Memory and Language*, *42*, 526–544.

Knöll, R., & Mezini, M. (2006). Pegasus: first steps toward a naturalistic programming language. In *Companion to the 21st ACM SIGPLAN symposium on object-oriented programming, systems, languages & applications (OOPSLA '06)* (pp. 542–559). New York: ACM.

Lohmeier, S. (2011). *Continuing to shape statically-resolved indirect anaphora for naturalistic programming.* Retrieved 2014/06/01, from http://monochromata.de/shapingIA/

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

O'Reilly, T., & McNamara, D. S. (2007). Reversing the reverse cohesion effect: Good texts can be better for strategic, high-knowledge readers. *Discourse Processes*, *43*(2), 121–152.

Rayner, K., Raney, G. E., & Pollatsek, A. (1995). Eye movements and discourse processing. In R. F. Lorch & E. J. O'Brien (Eds.), *Sources of coherence in reading* (pp. 9–35). Hillsdale: Erlbaum.

Schwarz-Friesel, M. (2007). Indirect anaphora in text: A cognitive account. In M. Schwarz-Friesel, M. Consten, & M. Knees (Eds.), *Anaphors in text : cognitive, formal and applied approaches to anaphoric reference* (pp. 3–20). Amsterdam: Benjamins.