**NICAD: Accurate Detection of Near-Miss Intentional Clones. Using Flexible Pretty-Printing and Code Normalization**

In this paper the authors introduce a new approach for clone detection, by presenting their tool, NiCad. It represents a novel approach on clone detection, by using a source transformation system, TXL, and an efficient text line comparison technique. Although it could be very useful at detecting a lot other types of clones, this paper mainly presents the results related to near-miss intentional clones, at function level.

The proposed approach is a multi-pass one, which is parser-based and language-specific, using simple text line comparison to achieve good time and space complexity. It uses flexible pretty-printing, code normalization and filtering together with comparison algorithm and clustering of potential clones for achieving a good level of clone detection and good memory and time usage.

NICAD has 6 main components: Extraction of Potential Clones, Flexible Pretty-Printing, Flexible Code Normalization, Flexible Code Filtering, Comparing the potential Clones, and Output Generation.

The Extraction of Potential Clones step has as purpose extracting possible candidates for clones that respect the minimal length of the clone. This minimal length is important for reducing the amount of further computations, and for presenting a more readable result at the end (ignoring clones that are too small for being useful). The implementation of this step is pretty straightforward, by using TXL extract function [^]. This will return a set of all embedded instances of one grammatical type, into a bigger one (function definitions from the entire source code for example).

Flexible Pretty Printing is a novel approach in clone detection. It uses the TXL’s agile parsing, for breaking different part of a statement into several lines that will be analysed by text comparison component. This allows different parts of a statement to be compared at different granularities. So, an item to be compared may contain one token or several tokens, according to the used pretty-printing rules. This is an advantage above the other clone detection tools, since there can be detected as clones just parts of a statement. Also, because of using TXL, this statement breaking is pretty easy to implement, by using just a modified version of the grammar.

Flexible Code Normalization is possible because of the TXL’s replacement rules. This allows normalization of different parts of a grammar element, removing the limitation to global replacement from other tools. By using TXL patterns NICAD provides flexibility in applying the normalization, by choosing only to normalize within certain type of statement or within a certain level of nesting.

Flexible Code Filtering is done with the help of TXL’s rules, being simple and efficient. All it has to do is to replace the uninteresting statements (defined by using a TXL pattern), with empty ones. The comparison of the potential clones is done with a Longest Common Subsequence algorithm, for comparing the text lines of potential clones. This algorithm will compare the pretty-printed and normalized sequences of text lines from clone, as items. Then it is determined the number of unique items in each potential clone. Then it is computed the percentage of unique items for each potential clones, as number of unique items / total number of items. If this percentage is below a certain threshold, then the sequences are considered to be clones of each other.

NICAD provides the results in two different representations: the textual report of the clone class information, where each clone class is shown with the corresponding name and line numbers, or the visual representation, which generates an HTML page showing the first code segment as an exemplar for each class clone. In the rest of the paper there are presented the experimental results, which were pretty good. The tool was used for finding clones into Abyss and Weltab.