SILICON IP – MORE THAN JUST DESIGN

VOLUME 17: IP CORE PROTECTION – Part I Soft IP, Chapter I-10: Computational Forensic Engineering (CFE)

Computational forensic engineering (CFE) is a non-signature based IP protection mechanism. It tries to identify whether the generated IP is coming from a familiar source or not.

Forensic engineering aims at providing both qualitative and quantitative evidence of substantial similarity between the design original and its copy. CFE is purported at identifying the entity that created a particular intellectual property (IP). Rather than relying on watermarking content or designs, the generic CFE methodology analyzes the statistics of certain features of a given IP and quantizes the likelihood that a well known source has created it.

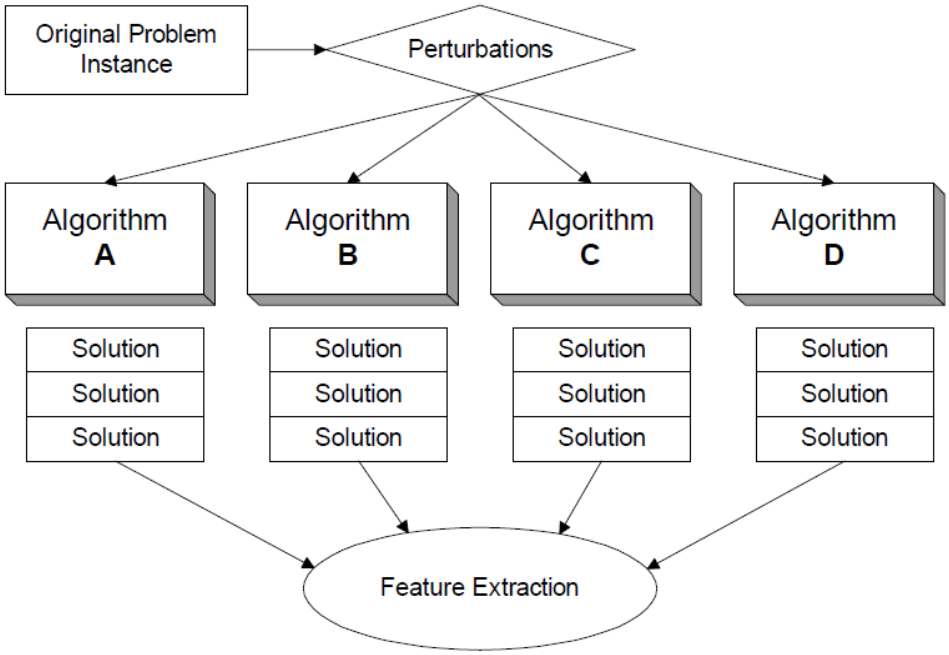
The generic CFE approach has four phases: feature and statistics data collection, feature extraction, entity clustering (clustering of heuristic properties for each analyzed tool), and decision validation.

In addition to IP protection, the developed CFE paradigm can have other potential applications: optimization algorithm selection and tuning, benchmark selection, and source-verification for mobile code.

FEATURE AND STATISTICS COLLECTION

The first phase can be divided into two subphases.

The *first subphase* is to identify and analyze relevant functional and structural properties of the problem. The properties are primarily obtained by analyzing solutions produced by various algorithms and identifying common features in solutions produced by a particular algorithm. For example, the Graph Coloring RLF algorithm is likely to have solutions with a large number of nodes in the graph to be colored with the same color, as well as some colors which only color one or two nodes.



Flow of the Generic Forensic Engineering Approach

The next step is to quantify properties by abstracting them to their numerical values. The goal is to eventually locate the solutions for each algorithm in an n-dimensional space. The dimensions are quantified properties which characterize solutions created by all considered algorithms. For example, for graph coloring solutions, we can find the cardinality of the largest independent set (IS) and normalize all other sets against it. Different coloring algorithms may produce solutions characterized by significantly different PDFs for this feature.

Third, features for which all considered algorithms display equivalent statistics will be discarded. A feature is considered as viable only if at least two algorithms have statistically distinct PDFs for it. For example, in the experimental results, the feature clausal stability for Satisfiability shows histograms which are different for each of the algorithms.

In the fourth step the principle component analysis will be conducted. Any subset of features which will provide the same information about the algorithms will be eliminated. The goal is to find the smallest set of features needed to fully classify the algorithms in order to improve efficiency and more importantly, statistical confidence.

Finally, fast algorithms for extraction of the selected properties will be applied. In some cases, extracting the features from the solutions is trivial. But in other cases, it can be complex and time consuming.

The *second subphase* is instance preprocessing. Order and lexical perturbations of an instance will be made. This is done to avoid any dependencies an algorithm may have on the naming or form of input, such as variable labels.

FEATURE EXTRACTION

All the perturbed instances will be run through each of the algorithms. Once all the solutions are obtained, the features will be extracted from them.

ALGORITHM CLUSTERING

The features will be placed into an n-dimensional space, and the results will be clustered.

VALIDATION

The final step is the application of non-parametric resubstitution software to establish the validity of the ability to distinguish distinct algorithms. Specifically, five hundred resubstituitions of 80% of the sample points will be run. Now, when a new solution is available, the generic flow and tools fully automatically determine which algorithm was used.

The figure above demonstrates the processing flow of the generic technique. The first level of flow chart shows the perturbation of a given instance in order to generate unbiased instances of the problem. Next, the instances are run on each of the candidate algorithms A, B, C, and D and the results are collected. From the solutions, relevant features are extracted. Given a large number of solutions for each of the algorithm, this process can take significant amount of time. Once the statistical and pattern features have been collected, algorithm clustering is performed.