# Case Based Reasoning

1. Elaborate the case based reasoning cycle :

There are four steps in case based reasoning which are retrieve, reuse, revision and retain. The first step is retrieve. The retrieve task start with a partial problem description and end when a best matching with previous case has been found. Its subtasks referred to as identify features, initially match, search and select, executed that order. The identification task basically comes up with a set of relevant problem descriptors, the goal of the matching task is to return to a set of cases that are sufficiently similar to the new case. In order to match cases based on semantic similarities and relative important of features an extensive body of general domain knowledge is needed to produce an explanation of why two cases match and how strong the match is. Syntactic similarities referred to as a “knowledge poor” approach and has its advantages in domain where general domain knowledge is difficult or impossible to acquire. On the other hand, semantic oriented approaches referred to as a “knowledge extensive” are able to use the contextual meaning of a problem description in its matching for domains where general domain knowledge is available. To identify a problem may involve simply noticing its input descriptors, but particularly for knowledge intensive method. An attempt is made to ‘understand’ the problem within its context. Unknown descriptor maybe disregarded or request to be explained by the user. To understand a problem involves to filter out noisy problem descriptors, to infer other relevant problem features, to check whether the feature value make sense within the context, to generate expectations of other features, etc. Checking of expectations may be done within the knowledge model (cases and general knowledge) or by asking the user. The task of finding a good match is typically split into two subtasks which are retrieves a set of plausible candidates and process of selecting the best one. Finding a set of matching cases is done by using the problem descriptors (input features) as indexes to the case memory in a direct or indirect way. There are in principle three ways of retrieving a case or a set of cases which are by following direct index pointers from problem features, by searching an index structures or by searching in a model of general domain knowledge. Cases may be retrieve solely from input features or features inferred from input. From a set of similar cases, a best match is chosen. This may have been done during initial match process but more often a set of case returned from that task. The selection process typically generates consequences and justify expectation. This may be done by using the system’s own model of general domain knowledge or by asking the user for confirmation and additional information.

The second step is reuse. Reuse of the retrieved case solution in context of the new case focuses on two aspects which are the differences among the past and the current and what part of retrieved case can be transferred to the new case. These are two subtasks which are copy and adapt. In simple classification tasks the differences are abstract away and the solution class of retrieved case is transferred to the new case as its solution class. This is a trivial type of reuse. Other system have to take into account differences in (a) and thus the reuse part in (b) cannot be directly transferred to the new case but require an adaptation process that takes into account those differences. In adapt, there are two main ways to reuse past case. First, reuse the past case solution. Second, reuse the past method that constructed the solution. Transformational reuse does not look how a problem is solved but focuses on the equivalent of solutions and this requires a strong domain-dependent model in the form of transformational operators plus a control regime to organize the operator application. Derivational reuse look at how the problem was solved in the retrieved case. The retrieved case holds information about the method used for solving the retrieved problem including the justification of the operator used, sub goals considered, alternative generated, failed search paths, etc.

The third step is revision. When a case of solution generated by the reuse phase is not correct, an opportunity for learning from failure arises. This phase is called revision and consists of two tasks which are evaluate the case solution generated by reuse and repair the case using domain specific knowledge. The evaluation task takes the result from applying the solution in the real environment. This is usually a step outside the CBR system, at least for a system in normal operation involves the application of a suggested solution to the real problem. The result from applying the solution may take some time to appear, depending on the type of applications. Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them. The best example is the CHEF system, where causal knowledge is used to generate an explanation of why certain goals of the solution plan were not achieves. CHEF learns the general situations that will cause the failure using an explanation-based learning technique. This is included into a failure memory that is used in the reuse phase to predict possible shortcomings of plans. This task uses the failure explanations to modify the solution in such a way the failure do not occur.

The last step is retain. This is the process of incorporating what is useful to retain from new problem solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves selecting which information from the case to retain, in what form to retain it, how to index the case for later retrieval from similar problems and how to integrate the new case in the memory structure. In CBR the case based is updated no matter how the problem was solved. If it solved by use of previous case, a new case may be built or the old case may be generalized to subsume the present case as well. If the problem was solved by other methods, including asking the user, an entirely new case will have to be constructed. A decision need to be made about what to use as the source of learning. Relevant problem of descriptors and problem solutions are obvious candidates. But an explanation or another form of justification of why a solution is a solution to the problems may also be marked for inclusion in a new case. The ‘indexing problem’ is a central and much focuses problem in case based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the space of indexes. Direct indexes, as previously mentioned, skip the latter step, but there is still the problem of identifying what type of indexes to use. This is actually the knowledge acquisition problem and should be analysed as part of domain knowledge analysis and modelling step. A trivial solution to the problem is of course to use all input features as indices. This is the approach of syntax-based method within instance-based and memory-based reasoning. This is the final step of updating the knowledge base with new case knowledge. If no new case and index set has been constructed, it is the main step of retain. By modifying the indexing of existing cases, CBR systems learn to become better similarity assessors. The turning of existing indexes is an important part of CBR learning. Index strengths or importance for a particular case or solution are adjusted due to the success or failure of using the case to solve the input problem. In knowledge-intensive approaches to CBR, learning may also take place within the general conceptual knowledge model. Thus, with a proper interface to the user a system may incrementally extend and refine its general knowledge model, as well as its memory of past cases, in the normal course of problem solving. This is inherent method in the PROTOS system, for example. All general knowledge in PROTOS is assumed to be acquired in such a bottom-up interaction with a competent user. The case just learned may finally be tested by re-entering the initial problem and see whether the system behaves as wanted.