Acknowledgements

The preCQE-prototype has been developed by the CIE-team at chair 6 of the Department of Computer Science at the Technical University Dortmund since April 2010. Members of the CIE development team have been so far Joachim Biskup, Lorenzo Benet González, Martin Bring, Michael Bulinski, Christine Dahn, Katharina Diekmann, Nils Maybaum, Ralf Menzel, Marcel Preuß, Torsten Schlotmann, Dirk Schalge, Karl Stelzner, Cornelia Tadros, Lena Wiese and Jaouad Zarouali.
The preCQE algorithm, in its parameterized form, as described in the article “Publishing Inference-Proof Relational Data: An Implementation and Experiments”, has been implemented as part of the larger Controlled Interaction Execution (CIE) framework prototype. In this data set, in the directory preCQE-source, we provide all the preCQE specific files, which are in the Java package edu.udo.cs.ls6.cie.server.censor.precqe. Furthermore we provide those files from package edu.udo.cs.ls6.cie.server.examples that contain the examples used in the article.

1 Package edu.udo.cs.ls6.cie.server.censor.precqe

PreCQE.java is the central class. It contains the method createInferenceProofInstance that performs the preCQE algorithm. There are also several set... methods that can be used to set the different parameters discussed in the article.

PreCQETester.java contains the code to run the experiments. It parses the command line for the different parameters or sets of parameters to use and then calls createInferenceProofInstance on a PreCQE instance for which the parameters have been appropriately set. It prints the output in CSV format.

PreCQEPParameters.java contains the list of all available parameters.

PreCQEPParameterFactory.java is the common interface for all factory classes used to implement parameters. Its implementing classes and interfaces are:

1. PreCQESolverFactory.java
2. SplitAlgorithmFactory.java
3. CaseAlgorithmFactory.java
4. ActiveDomainFactory.java
5. MarkedDatabaseFactory.java

1.1 Solver

PreCQESolver.java is an abstract class that implements the framework for the main tree search procedure. There are three classes that implement different search strategies:

- DFSSolver.java
- PBSSolver.java
- GDFSSolver.java
It uses the Java class `java.util.concurrent.ThreadPoolExecutor` for the parallel search. The threads of the pool are instances of the class `PreCQEThread.java` and are generated using the class `PreCQEThreadFactory.java`.

The nodes of the search tree are represented by instances of the classes `TreeNode.java` and `TreeNodeCore.java`.

The class `Ancestry.java` provides a compact representation of the path to a node. This can be used to sort the nodes by their position within the search tree.

`PBSSolver` uses the help of the class `CountUpDownLatch.java` when waiting for the search to terminate.

`PreCQEAlgorithm.java` implements the main branch-and-bound procedure through its `ground` method. It is refined by `PreCQEWorker.java` that implements the `java.util.concurrent.Callable` interface. The class `PreCQEWorker` in turn is refined by the classes `DFSTask.java` and `PBSTask.java`, which are used by the matching solver implementations.

`Simplify.java` contains the code for simplification of formulas.

`EstimateLieLowerBound.java` estimates the lower bound of the lie count for a tree node to be examined. Auxiliary classes for this are `GetUnmarkedLiterals.java` and `LiteralSet.java`.

Further auxiliary classes to perform the `ground` method are

- `GetUnmarkedGroundLiterals.java`
- `InstantiationOrVariant.java`

1.2 Split

`SplitAlgorithm.java` is an abstract class that is the base of the different split algorithms. There are the subclasses

- `SplitDefault.java`
- `SplitDefaultOld.java` (obsolete)
- `SplitHighOccur.java`
- `SplitPureAbstract.java` (abstract superclass of the other `SplitPure...` classes)
- `SplitPureApprox.java`
- `SplitPureApproxGreedy.java`
- `SplitPureComplete.java`
1.3 Case Differentiation

CaseAlgorithm.java is an abstract class that is the base of the different case differentiation algorithms. There are the subclasses

- CaseDefault.java
- CaseShortestRelation.java
- CaseLeastQuantifiedVar.java
- CaseLeastVar.java

1.4 Active Domain

ActiveDomain.java is an abstract class that is the base of the different active domain algorithms. There are the subclasses

- SimpleActiveDomain.java
- DynamicActiveDomain.java
- FullActiveDomain.java

For the “dynamic” active domain the auxiliary classes ConstantsFinder.java, VariablesFinder.java, GetVariableOccurrences.java, Occurrence.java, Programme.java and CollectAction.java are used.

As a subordinate parameter the calculated active domain optionally can be sorted or shuffled. ActiveDomainOrderAction.java implements this.

1.5 Marked Database

MarkedSQLAppDatabase.java extends the CIE framework class SQLAppDatabase to provide for the storage of the markings of tuples. There are three implementing classes:

- MarkedSQLAppDatabasePlain.java
- MarkedSQLAppDatabaseSlim.java
- MarkedSQLAppDatabaseUsingSavepoints.java

Marker.java enumerates the different markers available. The class AtomMarking.java represents marked tuples, including the key of the tree node where they were marked. For lists of marked tuples, as they occur along the paths of the search tree, the class ImmutableStack.java, which provides for sharing of common prefixes, is used.

Position.java defines a (database) attribute by the relation name and the index of the attribute.
1.6 Miscellaneous

For debugging purposes a graphical output of the search tree, to be processed by GraphViz, can be generated. This is achieved through the interface GraphWriter.java and its implementing classes SimpleGraphWriter.java and NoOpGraphWriter.java.

The classes PreCQE.Censor.java, PreCQE.ViewCensor.java and PreCQE.NotificationHandler.java provide the integration of the preCQE algorithm into the CIE framework.

2 Package edu.udo.cs.ls6.cie.server.examples

The source code for the examples used in the article can be found in these classes:

- PreCQEMedicineV1.java
- PreCQEMedicineV2.java
- PreCQEMedicineV3.java
- PreCQEStudentV1_new.java
- PreCQEStudentV2_new.java
- PreCQEStudentV3_new.java
- PreCQEStudentQ3_new.java