Switching Algorithm – readme file

How to use:

The API that the user should use is displayed at the class:"**SwitchingDiagnosticEngine**"
By 3 methods:

- **FindDiagnosisHaltByFirstDiagnosis**
- **FindDiagnosisHaltByTime**
- **FindDiagnosisHaltByQuantiy**

All this methods gets as parameters: an observation, an initial set of conflicts, an initial set of diagnoses and a unique parameter if needed (i.e. time or quantity).
The return value of all this methods is '**DiagnosisSet**'.

The Main Algorithm:

The main algorithm is in the class "**SwitchingAlgorithm**" where the main method is "**FindDiagnosis**".
As explained above – the user should not use directly this class, he should use the wrapper class "**SwitchingDiagnosticEngine**".

All the configuration also reside in the "SwitchingAlgorithm" class, include:

- <u>Testing Configuration</u>
  In the end of this class there is a static class named "TestingEnvironment" – the developer can put there the desired model files (System model file, Observation file and Diagnosis file) if he wants to use our 'test' classes.

- <u>Data Structure Configuration</u>
  We implemented 2 data structures to support this algorithm, a simple one called: ' SetsDataStructure' and another one that implemented a little bit like a prefix tree called ' CompSetTree'. The user can choose each of this structures to use.

  For using the Simple Data Structure do the following:
  1. Uncomment line 22 –
     private readonly SetsDataStructure _diagnosisesSetDataStructure;
  2. Comment line 23 –
     private readonly CompSetTree.CompSetTree _diagnosisesSetDataStructure;
  3. Uncomment line 24 –
     private readonly SetsDataStructure _conflictsSetDataStructure;
  4. Comment line 25-
     private readonly CompSetTree.CompSetTree _conflictsSetDataStructure;
  5. Comment line 37-
     this._conflictsSetDataStructure = new SetsDataStructure("Conflicts");
  6. Comment line 38-
     this._conflictsSetDataStructure = new CompSetTree.CompSetTree();
  7. Comment line 46-
     this._diagnosisesSetDataStructure = new SetsDataStructure("Diagnosises");
  8. Comment line 47-
     this._diagnosisesSetDataStructure = new CompSetTree.CompSetTree();

  9. Change the method signature at line 265 from:
     **private void AddComponentToSet(SetsDataStructure sets, List<Gate> gates,
     bool needToBeSatisfied)**

To:

private void AddComponentToSet(CompSetTree.CompSetTree sets, List<Gate> gates, bool needToBeSatisfied)

For using the Tree Data Structure just do the **opposite** in the 10 steps above.
*All the lines mentioned are in "**SwitchingAlgorithm"** class**.**

- Solver configuration

We implemented a mock for testing purposes for the "**ConstraintSystemSolver**".
So similar to the data structures the user/developer can choose if he want to use the real "**ConstraintSystemSolver**" or the mock one(when using the mock you must specify the files in the "**TestingEnvironment**" class as discussed above).
To change between those 2 just go to "**SwitchingAlgorithm**" class and choose 1 of the lines:
Line 29 –

public static ConstraintSystemSolverMock Solver = ConstraintSystemSolverMock.getInstance();

For the mock.
Or
Line 30 -

public static ConstraintSystemSolver Solver = ConstraintSystemSolver.Instance;

For the real solver.
(Comment the un-wanted line).
* We use the mock most of the time.

For any further question:

Niv – nivave@post.bgu.ac.il
Tal – talvis@post.bgu.ac.il