International Journal of Chemical, Materials and Biomolecular Sciences
ISSN: 2415-6620
Vol:5, No:9, 2011

Development of a Simulator for Explaining Organic Chemical Reactions Based on Qualitative Process Theory

Alicia Y. C. Tang, Rukaini Hj. Abdullah, and Sharifuddin M. Zain

***Abstract*—**This paper discusses the development of a qualitative simulator (abbreviated QRiOM) for predicting the behaviour of organic chemical reactions. The simulation technique is based on the qualitative process theory (QPT) ontology. The modelling constructs of QPT embody notions of causality which can be used to explain the behaviour of a chemical system. The major theme of this work is that, in a qualitative simulation environment, students are able to articulate his/her knowledge through the inspection of explanations generated by software. The implementation languages are Java and Prolog. The software produces explanation in various forms that stresses on the causal theories in the chemical system which can be effectively used to support learning.

***Keywords*—**Chemical reactions, explanation, qualitative process theory, simulation

I. Introduction

In organic chemical reactions, one has to understand the many cognitive steps (the "mechanisms") involved before a stable product is formed. Understanding these cognitive steps is among the many difficulties faced by chemistry students. Traditional chemistry educational software is inadequate in promoting understanding such as why and how things happen. These programs do not "explain" simply because the results are obtained through chaining of rules or by searching the reaction routes that have been pre-coded in software. Consequently, some students, particularly weak learners would require additional learning aids such as a software tool to assist them in their learning. In conventional approach, reaction prediction is performed by finding a route through searching the entire state space. Explanation generation is also a great challenge to this type of programming paradigm. Traditional chemistry software is therefore inadequate to promote understanding or explain toward its results because traditional method does not link "reasoning" to problems. This paper describes the development of a qualitative simulator for the simulation of organic chemical reactions. The simulator is based on the qualitative reasoning framework described in [1]. QRiOM (Qualitative Reasoning in Organic Mechanism) is the product of the implementation of the framework. Qualitative process theory (QPT) [2] is used for the representation of chemical theories and chemical facts. QPT is well-known ontology for qualitative reasoning (QR). The development of QRiOM was motivated by a number of QR related systems reported in [3]-[11].

A. Y. C. Tang is with the University of Tenaga Nasional, Selangor, Malaysia (phone: 603-8921-2336; e-mail: aliciat@uniten.edu.my).
R. Abdullah is with the Department of Artificial Intelligence, Malaya University, Kuala Lumpur, Malaysia (email: rukaini@um.edu.my)
S. M. Zain is with the Department of Chemistry, Malaya University, Kuala Lumpur, Malaysia (e-mail: smzain@um.edu.my).

II. Previous Work

This work combines qualitative reasoning and ontologies in a simulation system, and generates explanations for learners from the system. In [12], "make-bond" and "break-bond" chemical bonding have been identified as two generic processes in the simulation of organic chemical reactions. From the analysis of various chemical reactions occurring under S_N1 and S_N2 mechanisms, the common set of chemical theories and behaviour for the generic processes have been identified, from which the model automation procedures are formulated. A set of QR algorithms used for the simulation of reactions have also been developed. The algorithms can cater for "select and sequence" ability, with the aid of the OntoRM ontology [13]. This paper will provide a few simulation results that serve as "explanation" to chemical phenomena related to organic reactions. Our approach for answer justification is based on causal reasoning. Overall, the issue of lack of explanation in chemistry software is addressed by embedding a causal explanation generator that produces explanation in various forms.

III. Software Architecture of QRIOM

Fig. 1 depicts the software modules implemented in QRiOM. Table 1 describes the role of the main modules in the simulator. The knowledge-base has a two layer structure. The purpose of the lower layer is to provide chemical facts to the simulator. This layer is called chemical instances (or basic facts). Instances refer to chemical elements and their chemical properties that do not change over time. The upper layer is the chemistry ontology for reaction mechanisms simulation. This tier is called OntoRM. The ontology defines the requirements and constraints when suggesting a reaction mechanism for a chemical equation simulation.

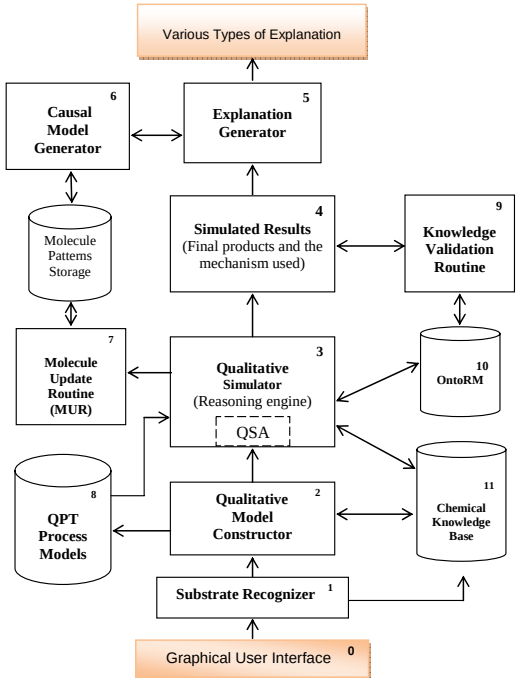
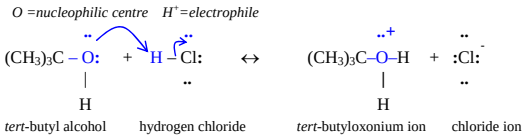


Fig. 1 Main software modules in QRiOM simulator

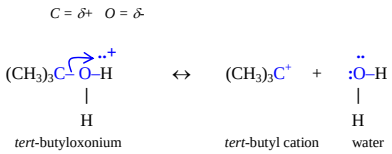
TABLE I
ROLES OF EACH MODULE IN QRIOM

	Roles
Module 1	This module checks user selection and returns the "type" of the input as either a nucleophile or an electrophile. From here on, an organic process may be determined.
Module 2	This module automates the construction of QPT models based on the identity of user inputs.
Module 3	This module does the actual reasoning and final product prediction.
Module 5	This module generates explanation on-the-fly based on causal reasoning.
Module 6	This module constructs causal graphs.
Module 7	This module keeps track of the structural change (pattern) of the substrate, from one organic reaction to another.
Module 9	The module validates the use of data *during* (to ensure the right data is passed to the reasoning engine) and *after* a simulation is completed (to ensure the results are predicted correctly).
Module 10	The reaction mechanism ontology that defines the basic chemical knowledge and chemical commonsense for S_N1 and S_N2.

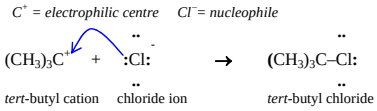
The chemical equation "$(CH_3)_3C–OH + HCl \rightarrow (CH_3)_3C–Cl + H_2O$" will be used as the illustration example. The "thought processes" for the chemical equation is depicted in Fig. 2. It is necessary to build process models to describe the "thought" before reasoning or simulation can be initiated. Fig. 3 shows a process model represented in QPT. The QPT model can be used to reproduce the behaviour of the first reaction step for the "$(CH_3)_3C–OH + HCl$" reaction (refer to part (a) of Fig. 2). Fig. 4 shows the simulation algorithm implemented in QRiOM.



(a) Reaction step 1



(b) Reaction step 2



(c) Reaction step 3

Name of the chemical process	Reactant 1	Reactant 2
Protonation (= "make-bond")	$(CH_3)_3COH$ (nucleophile)	H^+ (electrophile)
Dissociation	$(CH_3)_3C–OH_2^+$	
Capturing of anion by carbocation	$(CH_3)_3C^+$ (electrophile)	Cl^- (nucleophile)

(d) Reactants and their associated chemical processes

Fig. 2 The conversion of a tertiary alcohol to yield alkyl chloride can be described as a series of three small steps

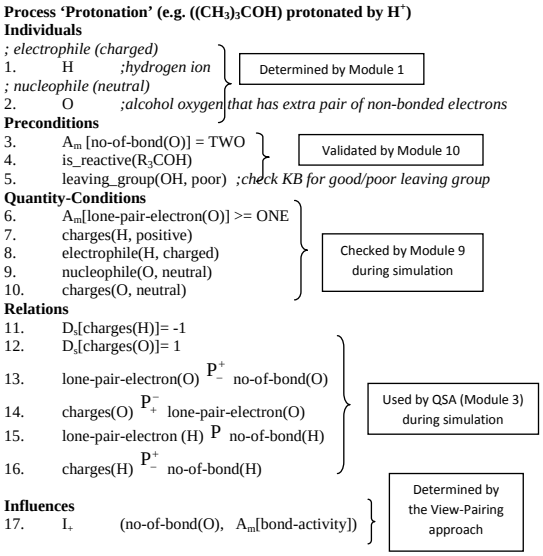


Fig. 3 QPT model for the "make-bond" process that links the proton to the oxygen atom in the tertiary alcohol substrate

QPT-BASED SIMULATION ALGORITHM

Q_Simulation(QPT_model, OUTPUT)
1. Perform qualitative reasoning on the constructed QPT model
   1.1 Store the process's entry conditions
   1.2 Store the directly influenced process quantity
   1.3 Keep track of the state transition (handled by the QSA module)
2. IF process_stopping_condition = true THEN
   Store propagated effects in special purpose data structures
   Store new individuals in the VIS
   Update the VIS
   END_IF
3. Update the substrate's molecular structure (handled by the MUR module)
4. IF VIS contains reactive individuals THEN
   Determine a suitable chemical process
   Go to step 1
   ELSE
   Retrieve final product from the VIS
   Call OntoRM to check for validity of the predicted product
   Call OntoRM to check for the possible order of process execution
   Write the final product and the proposed mechanism to OUTPUT
   END_IF
5. Return OUTPUT

Fig. 4 The QPT-based simulation algorithm for chemical process reasoning

IV. Prototype Development

Fig. 5 shows the problem solving model of QRiOM (i.e. the protocol to interact with the software tool) while Fig. 6 gives the main interface of the qualitative simulator. QRiOM development is fully event-driven and object-oriented.

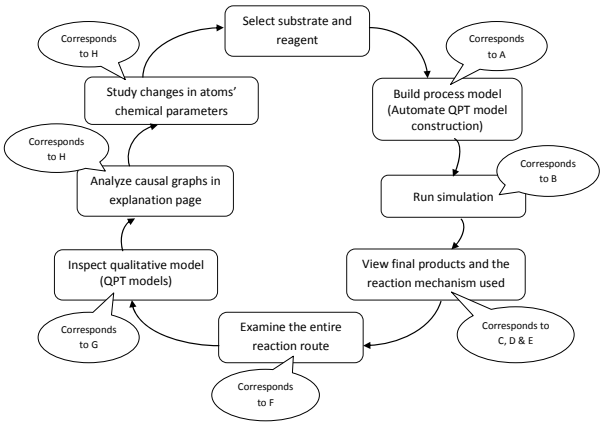


Fig. 5 Problem solving model (Labels A – H are in Fig. 6)

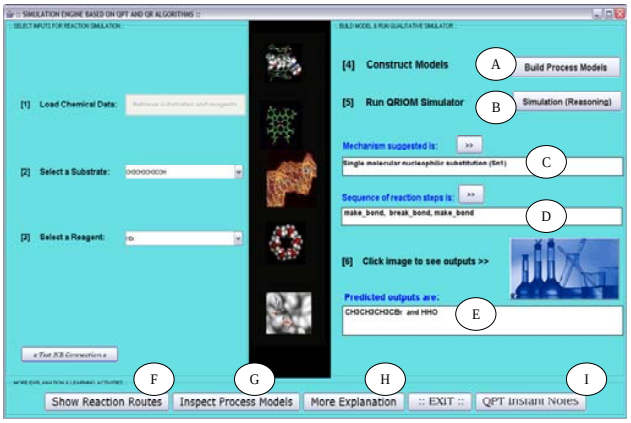


Fig. 6 The main interface of the simulator prototype

Apart from the final products, the simulator will produce the following results to explicate a phenomenon being asked: (1) qualitative model for organic processes, (2) view pairs used in the simulation, (3) causal graphs, (4) parameter state history for each atom that is involved in a reaction, and (5) overall structural change of the substrate. All of which are derived from the process model directly. As shall be shown, describing the domain knowledge in qualitative terms (via the modelling constructs of QPT) is sufficient to explain organic chemical phenomena. The following subsections present a few simulation outcomes (QPT model, view pairs, and causal graph) and the Java code for implementing them.

A. Java Snippets for the Construction of QPT Models

QPT model serves the educational objective of "knowledge articulation" of various aspects of a chemical reaction. When inspecting a QPT process model, students have to articulate relationships between entities and dependencies. Fig. 7 shows the Java code that retrieves the chemical facts from the chemical KB in order to construct a QPT model while a screenshot of model inspection page is shown in Fig. 8.

```
public void viewModel_actionPerformed(ActionEvent e) {

 amzi.ls.LogicServer ls = new amzi.ls.LogicServer();
          :
 String pn = gd.ProName;                                  Prepare the
 String ind1 = gd.View1;                                  individuals for the
 String Ind1Type = gd.Type1;                              chemical process
 String Ind2Type = gd.Type2;
 String ind2 = gd.View2;
          :
 jTextArea2.setFont(new java.awt.Font("Dialog", Font.BOLD, 12));
 jTextArea2.append("Process Activated: " + gd.ProNameList[0] + "\n");
 jTextArea2.append("\n" + "Individuals (The reacting units in this process/step)" + "\n");
 jTextArea2.setFont(new java.awt.Font("Dialog", Font.PLAIN, 10));
 jTextArea2.append("  " + gd.ViewList1_sn2[0] + "\t" + gd.ViewList2_sn2[0] + "\n");
 jTextArea2.setFont(new java.awt.Font("Dialog", Font.BOLD, 12));
 jTextArea2.append("\n" + "Quantity-Condition (Entry requirements to activate the process)" + "\n");
 jTextArea2.setFont(new java.awt.Font("Dialog", Font.PLAIN, 10));
          :
 ls.Load("bondKB.xpl");
 term1 = ls.CallStr("qty_cond("+gd.ProNameList[0]+", "+gd.TypeList1[0]+", X, Y, Z).");
          :                                                                    Based on the
  do                                                                          view's type,
  {                                                                           general set of
    jTextArea2.append("  " + ls.GetStrArg(term1, 3) + "(" + ls.GetStrArg(term1, 4) + ") "   chemical
                                                  + ls.GetStrArg(term1, 5) + "\n" );         theories are
                                                                              retrieved from
  } while (ls.Redo());                                                        the KB

  term2 = ls.CallStr("qty_cond("+gd.ProNameList[0]+", "+gd.TypeList2[0]+", X, Y, Z).");
          :
  do{
      jTextArea2.append("  " + ls.GetStrArg(term2, 3) + "(" + ls.GetStrArg(term2, 4) + ") "
      + ls.GetStrArg(term2, 5)+ "\n" );
   } while (ls.Redo());

jTextArea2.setFont(new java.awt.Font("Dialog", Font.BOLD, 12));                Prepare the
jTextArea2.append("\n" + "Influences (Direct effect caused by the process)" + "\n");   headings for the
jTextArea2.setFont(new java.awt.Font("Dialog", Font.PLAIN, 10));               QPT model, and
                                                                               the direct
if (gd.ProNameList[0].equals("make_bond"))                                     influence of the
      jTextArea2.append("  " + "A covalent bond is added (formed)" + "\n");    process
else jTextArea2.append("  " + "A covalent bond is removed (cleaved)" + "\n");

jTextArea2.setFont(new java.awt.Font("Dialog", Font.BOLD, 12));
jTextArea2.append("\n" + "Parameters dependency (Effects propagation)");
jTextArea2.setFont(new java.awt.Font("Dialog", Font.PLAIN, 10));
          :
 do{
                                                                               Display the
   jTextArea2.append("  " + ls.GetStrArg(term3, 3) + "(" + ls.GetStrArg(term3, 5) + ")  followed   effect
   ls.GetStrArg(term3, 4) + "(" + ls.GetStrArg(term3, 6) + ")" + "\n");        propagation
  } while (ls.Redo());                                                         caused by the
                                                                               process. These
                                                                               are the indirect
 jTextArea2.append("\n" + gd.ViewList2[0] + "  [" + gd.TypeList2[0] +"]" + ":" + "\n");   influences of a
 term3 = ls.CallStr("process_relations(make_bond, "+gd.TypeList2[0]+", P, Q, R, S).");   QPT model
          :
 do{

   jTextArea2.append("  " + ls.GetStrArg(term3, 3) + "(" + ls.GetStrArg(term3, 5) + ")  followed
   ls.GetStrArg(term3, 4) + "(" + ls.GetStrArg(term3, 6) + ")" + "\n");
  } while (ls.Redo());
          :
}
```

Fig. 7 The Java code for retrieving chemical theories of reacting species for constructing QPT model

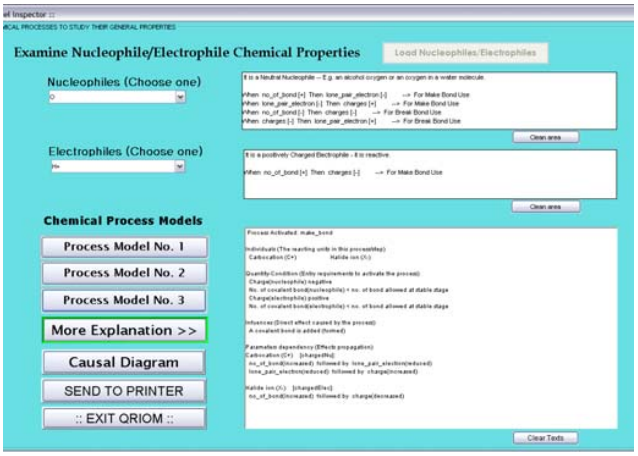


Fig. 8 The QPT model inspection page of QRiOM

B. *Java Snippets for Implementing the Reasoning Engine*

Quantity Space Analyzer (QSA) is one the important software modules in QRiOM. This module performs tasks such as updating and maintaining multiple data structures whenever an organic process is activated. Since the majority of chemistry students have difficulties identifying the right reacting units for processes activation, the tool will generate the whole set of reacting units (called "view pairs" in QPT) used in the entire simulation thus informing the learner of the types of reacting species that activated a given chemical process (Fig. 9). The Java code for "view structure updating" is presented in Fig. 10. These results can then be used to generate the necessary reaction route for the entire simulation of a chemical equation.

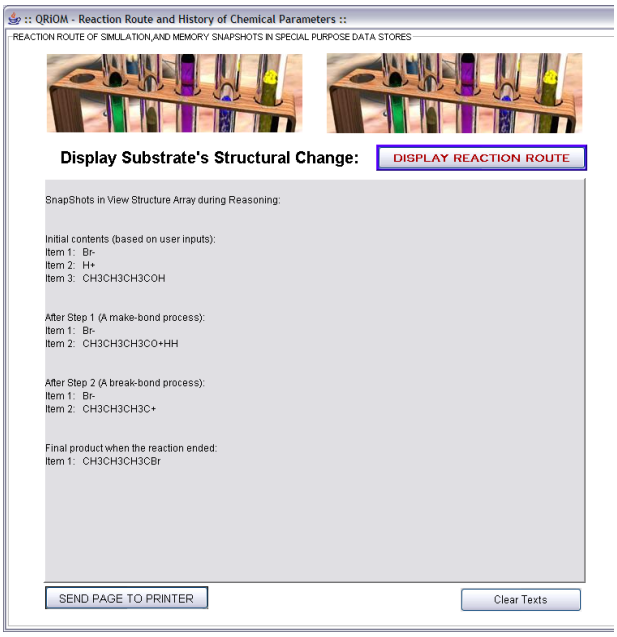


Fig. 9 The choice of reacting units for each reaction step and the intermediates produced are displayed for further inspection

```
          :
if (Subst_1_ChargedHistory[4].equals("pos"))
    {  appendedCharge1 = StartMaterialTable[0].concat("+");   // Check its charge's state
       theStr1 = "CH3CH3CH3".concat(appendedCharge1);}
else if (Subst_1_ChargedHistory[4].equals("neg"))
        {   appendedCharge1 = StartMaterialTable[0].concat("-");
            theStr1 = "CH3CH3CH3".concat(appendedCharge1);}
else theStr1 = "CH3CH3CH3".concat(StartMaterialTable[0]);

if (Agent_2_ChargedHistory[4].equals("pos"))
        {  appendedCharge1 = StartMaterialTable[1].concat("+");
           theStr2 = theStr1.concat(appendedCharge1);}
else if (Agent_2_ChargedHistory[4].equals("neg"))
          {  appendedCharge1 = StartMaterialTable[1].concat("-");
             theStr2 = theStr1.concat(appendedCharge1);}
 else  theStr2 = theStr1.concat(StartMaterialTable[1]);

ViewStructureArr[0] = theStr2;                              // Update the contents of VIS
          :
```

Fig. 10 The Java statements for updating the VIS in order to suggest the next organic process in the qualitative simulation environment

Fig. 11 gives the Java snippets for updating changes in parameters' states. The values assigned to the chemical parameters during simulation are recorded in special purpose data structures for future retrieval. One such structure is the atom property table (Fig. 12).

```
            :
ls.Init("");
ls.Load("chemkb.xpl");

t1 = ls.CallStr("qpropAPTable(make_bond, chargedElec, P3, Q3, R3, S3);   // Prepare to retrieve chemical theories
if (t1 == 0)
  {       :
     ls.Close();
     return;
  }

do{
      if (n1 == 1){
            qtyArrTemp[0] = ls.GetStrArg(t1, 3);     qtyArrTemp[1] = ls.GetStrArg(t1, 4);
            signArrTemp[0] = ls.GetStrArg(t1, 5);   signArrTemp[1] = ls.GetStrArg(t1, 6);}
      if (n1 == 2){
            qtyArrTemp[2] = ls.GetStrArg(t1, 3);     qtyArrTemp[3] = ls.GetStrArg(t1, 4);
            signArrTemp[2] = ls.GetStrArg(t1, 5);   signArrTemp[3] = ls.GetStrArg(t1, 6);}
            n1++;
    } while (ls.Redo());

if ((individual_2_type.equals("chargedElec")) && (bActivity.equals("make_bond")))
  {
     if ((signArrTemp[0].equals("plus")) && (qtyArrTemp[0].equals("charge")))
      {  for (int t=0; t<=2; t++)
           if (charge[t].equals(smt_0_ChargeVal)) index = t;
               gdat.Sub_1_ChargedHistory[4] = Subst_1_ChargedHistory[4]
               = charge[++index];  smt_0_ChargeVal = Subst_1_ChargedHistory[4];}
         else if ((signArrTemp[0].equals("minus")) && (qtyArrTemp[0].equals("charge")))
            {  for (int t=0; t<=2; t++)
                 if (charge[t].equals(smt_0_ChargeVal)) index = t;
                   gdat.Sub_1_ChargedHistory[4] = Subst_1_ChargedHistory[4] = charge[--index];
                   smt_0_ChargeVal = Subst_1_ChargedHistory[4];}

     if ((signArrTemp[0].equals("plus")) && (qtyArrTemp[0].equals("lone_pair_electron")))
      {  for (int t=0; t<=4; t++)
           if (String.valueOf(lone_pair[t]).equals(smt_0_LPVal)) index = t;
               gdat.Sub_1_LonePairHistory[4] = Subst_1_LonePairHistory[4] =
                  String.valueOf(lone_pair[++index]);
               smt_0_LPVal = Subst_1_LonePairHistory[4];}
         else if ((signArrTemp[0].equals("minus")) && (qtyArrTemp[0].equals("lone_pair_electron")))
            {  for (int t=0; t<=4; t++)
                 if (String.valueOf(lone_pair[t]).equals(smt_0_LPVal)) index = t;
                   gdat.Sub_1_LonePairHistory[4] = Subst_1_LonePairHistory[4] =
                          String.valueOf(lone_pair[--index]);
                   smt_0_LPVal = Subst_1_LonePairHistory[4];}

     if ((signArrTemp[0].equals("plus")) && (qtyArrTemp[0].equals("no_of_bond")))
      {  for (int t=0; t<=4; t++)
           if (String.valueOf(bond[t]).equals(smt_0_BondVal)) index = t;
               gdat.Sub_1_BondHistory[3] = Subst_1_BondHistory[3] = String.valueOf(bond[++index]);
               smt_1_BondVal = Subst_2_BondHistory[4];}
         else if ((signArrTemp[0].equals("minus")) && (qtyArrTemp[0].equals("no_of_bond")))
            {  for (int t=0; t<=4; t++)
                 if (String.valueOf(bond[t]).equals(smt_0_BondVal)) index = t;
                   gdat.Sub_2_BondHistory[4] = Subst_2_BondHistory[4] = String.valueOf(bond[--index]);
               smt_0_BondVal = Subst_1_BondHistory[4];}

                        :
                        :
  }
```

The chemical theories (stored as qprop) for the identified organic process in the chemical KB are retrieved and stored in temporary arrays

The states of the chemical parameter are updated based on the qualitative proportionalities retrieved earlier.

Fig. 11 The Java code for updating the chemical parameters' states of each atom during simulation

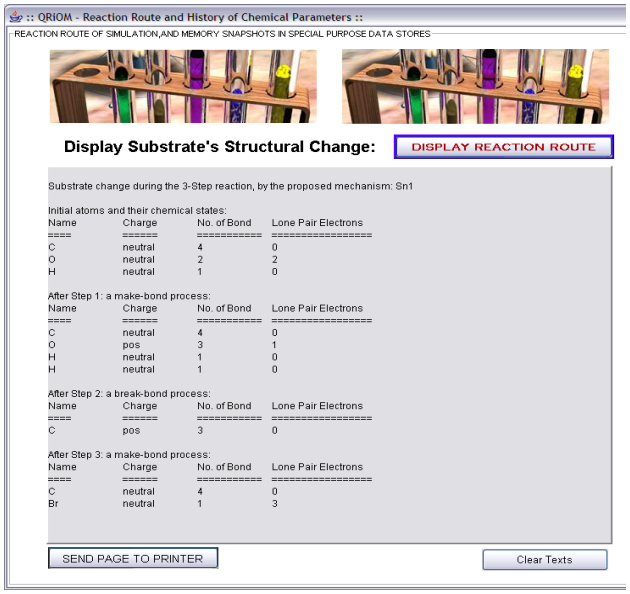


Fig. 12 The chemical states possessed by each reacting unit during simulation are stored in atom property table

C. *Java Snippets for Causal Graph Generator*

Algorithms determine what the behaviour is, not an explanation of it. An explanation of system behaviour may take many forms. An example is causal accounts (or causality). Causal account is a kind of explanation that is consistent with our intuitions of how systems function. The explanation used by QRiOM is achieved by tracing the cause-effect propagation through the modelling constructs of QPT. For example, during each reaction simulation, a causal graph (Fig. 13) is generated that shows the use of the qualitative proportionality statements (or functional dependency) in the QPT models. Fig. 14 shows the Java code that generates causal graphs for explaining the cause-effect interaction among all the parameters.

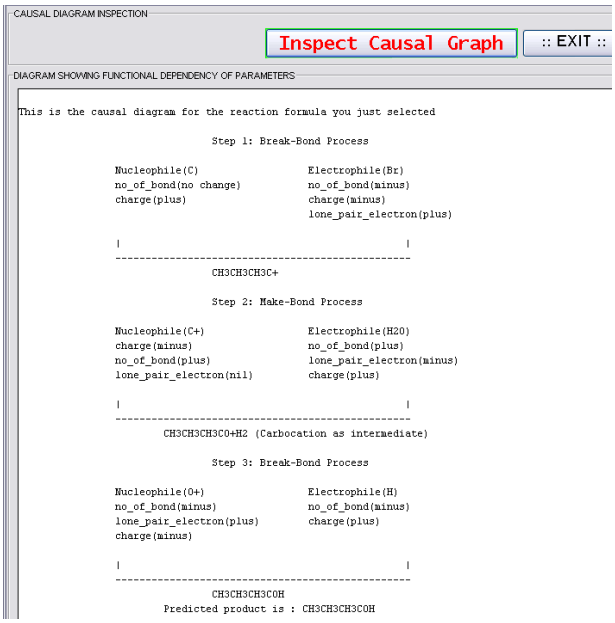


Fig. 13 A causal graph generated by QRiOM that enables learners to examine the cause-effect relationships of chemical parameters during reasoning

```
public void causalGraph_actionPerformed(ActionEvent e) {
  globalDataEG gd = new globalDataEG();

  jCG.setText(" ");
  jCG.append("\nThis is the causal diagram for the reaction formula you just selected\n");

if (gd.procInvolved[0].equals("make_bond")){
  jCG.append("Step 1: Make-Bond Process\n");
  jCG.append("Nucleophile"  + "(" + gd.smn + ")" + "\t\t\t" + "Electrophile" +  "(" + gd.aat00 + ")" +"\n" );

   for(int y=0; y<=2; y++){
     if(!(gd.par1[y].equals("nil")))
      if (y==1)
         jCG.append("\t\t" + gd.par1[y] + "(" + gd.sign1[y] + ")" );
       else  jCG.append("\t\t" + gd.par1[y] + "(" + gd.sign1[y] + ")" );
     else   jCG.append("\t\t\t\t");
     if(!(gd.par2[y].equals("no change")))
       { if (y==2)
           jCG.append("\t\t\t" + gd.par2[y] + "(" + "no change"  + ")" + "\n");
         else if (y==1)
           jCG.append("\t" + gd.par2[y] + "(" + gd.sign2[y]  + ")" + "\n");
         else jCG.append("\t\t" + gd.par2[y] + "(" + gd.sign2[y] + ")" + "\n");}
     else  jCG.append("\n");

  }
          :
          :
} // End Generate Causal Graph
```

These codes prepare the values taken by each main parameter during the entire reasoning and simulation. The set of values are then formatted onto the appropriate user graphical interfaces

Fig. 14 The Java code that keeps track of the values assigned to each parameter during reasoning. The set of values will be retrieved and formatted to graphical user interface

When the outputs of causal graph, QPT model, and atom property table are examined together, the students are expected to relate various aspects in a reaction such that they are able to explain an organic reaction in a more elaborate way. This will lead to an improvement in one's conceptual understanding of the subject.

V. Conclusion

A qualitative simulation environment that enables students to articulate his/her knowledge through the inspection of explanations generated by software has been described. The implementation of the main modules in the reasoning engine has also been presented as a collection of Java snippets. QRiOM is the first chemistry education software that can generate multiple forms of textual explanation via QPT-based reasoning. QRiOM is viewed as useful and effective by chemistry learners, consistent with the fact that students' conceptual understanding is improved [14].

References

[1] A.Y.C. Tang, S.M. Zain, and N.A. Rahman, "Design and Development of a Qualitative Simulator for Learning Organic Reactions," International Journal of Computers, Vol. 3, Issue 1, pp. 96-103, 2009.
[2] K.D. Forbus, "Qualitative Process Theory," Artificial Intelligence, 24: 85-168, 1984.
[3] A. Bouwer and B. Bredeweg, "VisiGarp: Graphical Representation of Qualitative Simulation Models," In Moore, J.D., Luckhardt Redfield, G., Johnson, J.L. (eds.) Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future. IOS Press/Ohmsha, 2001, pp. 294-305, Japan, Osaka.
[4] A. Bouwer and B. Bredeweg, "Generating Structured Explanations of System Behaviour Using Qualitative Simulations," Looi, C.K. McCalla, G., Bredeweg, B. and Breuker, B. (Eds.), Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology, 2005, pp. 756-758, IOS press, Amsterdam.
[5] T. Horiguchi and T. Hirashima, "Robust Simulator, a Method of Simulating Learners' Erroneous Equations for Making Error-Based Simulation," in Proceedings of the Intelligent Tutoring Systems, 2006, pp. 655-665.
[6] T. Horiguchi, T. Hirashima, and M. Okamoto, "Conceptual Changes in Learning Mechanics by Error-based Simulation," in Proceedings of the International Conference on Computers in Education (ICCE), 2005, Singapore, pp. 138-145.
[7] T. Hirashima, T. Horiguchi, A. Kashihara, and J. Toyoda, "Error-based Simulation for Error-visualization and its Management," International Journal of Artificial Intelligence in Education, vol. 9, pp. 17-31, 1998.
[8] K.D. Forbus, J. Everett, L. Ureel, M. Brokowski, J. Baher, and S. Kuehne, "Distributed Coaching for an Intelligent Learning Environment," in Proceedings of International Workshop on Qualitative Reasoning, 1998, Cape Cod.
[9] K.D. Forbus, P. Whalley, J. Everett, L. Ureel, M. Brokowski, J. Baher, and S. Kuehne, "CyclePad: An Articulate Virtual Laboratory for Engineering Thermodynamics," Artificial Intelligence Journal, 114: 297-347, 1999.
[10] K.D. Forbus, "Articulate Software for Science and Engineering Education," in Forbus, K.D., Feltovich, P., Canas, A. (eds.) Smart Machines in Education: The Coming Revolution in Educational Technology, 2001, AAAI Press.
[11] S.M.F.D. Syed Mustapha, J.S. Pang, and S.M. Zain, "QALSIC: Towards Building an Articulate Educational Software using Qualitative Process Theory Approach in Inorganic Chemistry for High School Level," Intl. J. of AIED, 15(3): 229-257, 2005.
[12] A.Y.C. Tang, S.M.F.D. Syed Mustapha, R. Abdullah, S.M. Zain, and N. A. Rahman, "Towards automating QPT model construction for reaction mechanism simulation," in the 21st International Workshop on Qualitative Reasoning, C. Price and N. Snooke (Eds), June 2007, Aberystwyth, United Kingdom, pp. 27-29.
[13] A.Y.C. Tang and S.M. Zain, "OntoRM: Ontology for Supporting the Simulation of Organic Reactions in a Qualitative Reasoning Environment," in the proceedings of the 2nd International Conference on Research Challenges in Computer Science, December 2010, Shanghai, China.
[14] A.Y.C. Tang, S.M. Zain, and R. Abdullah, "Development and Evaluation of Chemistry Educational Software for Learning Organic Reactions Using Qualitative Reasoning," Intl. J. of Education and Information Technologies, Issue 3, Vol. 4, pp. 9-138, 2010.

Alicia Y.C. Tang is a senior lecturer at the University of Tenaga Nasional, Malaysia. She obtained her Ph.D. from University of Malaya in 2011. Her research fields include Qualitative Reasoning, Agent Technology, Data Mining, and AI in Education. Alicia is a member of APSCE, IEEE, and AIED Society.

Rukaini Hj. Abdullah received her Ph.D. from University of Leeds, United Kingdom. She is attached to the Faculty of Computer Science and IT, University of Malaya. Her research interests are Natural Language Processing, AI in Education, Information Retrieval, and Computer Based Learning.

Sharifuddin Mohd. Zain received his Ph.D. in 1995 from Imperial College, London, United Kingdom. He is an Associate Professor in the Department of Chemistry at University of Malaya, Malaysia. His research interests include Computational Chemistry, Application of Computers in Chemical Research and Education, Artificial Intelligence in Chemistry, Molecular Spectroscopy, and Environmental Modelling. Sharifuddin is leading several projects supported by the Ministry of Science, Technology and Innovation (MOSTI), Malaysia. He is also a Regional Editor for the Journal of Global Environmental Engineering.