IMPLEMENTING DYNAMIC-EPISTEMIC QUESTIONING
Engineering and Teaching Information Seeking via Dynamic Inquiry

Ştefan Minică
Institute for Logic, Language and Computation, University of Amsterdam, Science Park 904, Amsterdam, Netherlands

Keywords: Logic Education, Computer Supported Learning, Information Dynamics, Question, Inquiry, Epistemic Games.

Abstract: Asking questions in multi-agent environments is an interesting and complex phenomenon with potential applications for both education and information technologies. We approach the problem of building an adequate model for questioning phenomena using dynamic-epistemic formalism and we present an implementation of multi-agent dynamic-epistemic questioning using *Haskell*. We start by introducing a dynamic-epistemic logic for questions which extends previous results from (van Benthem and Minică, 2009). Next, an implementation for model-checking in epistemic-issue models is proposed based on a similar implementation for epistemic logic from (van Eijck, 2004). We conclude the paper by probing potential applications of this results in education and beyond by presenting an ongoing project of building an accessible and intuitive web-based graphical interface to be used in an electronic teaching environment for visualizing, designing and managing strategies for asking questions during abstract scientific inquiry and in cooperative or competitive scenarios of multi-agent goal-driven investigations and interrogative interactions.

1 INTRODUCTION

Questions are important for various reasons. Understanding their abstract structure is considered to be a significant problem in various fields of research. In information theory and AI, a formal model of questions is important for achieving the goal of implementing question-asking machines (Knuth, 2005). In logic, semantics and pragmatics of natural language the aim is to understand questioning by human users (Wiśniewski, 1996; Groenendijk, 2008; Ciardelli and Roelofsen, 2009). In epistemology and philosophy of science questions are studied because of their role in belief revision mechanisms (Baltag and Smets, 2008; Enqvist, 2009) and in interrogative processes of scientific inquiry (Hintikka et al., 2002; Genot, 2009). A dynamic epistemic approach to questions makes the interaction between questioning actions and knowledge explicit and shows how information changing events are relevant for a large variety of rational activities ranging from multi-agent interaction to scientific inquiry (Baltag, 2001; Unger and Giorgolo, 2008; van Benthem and Minică, 2009; Icard, 2009; Peliš, 2009).

2 QUESTIONING THEORY

This section presents problems and methods regarding modeling of questioning phenomena from a theoretical perspective. It will introduce a Dynamic Epistemic Logic of Questions (henceforth, *DELQ*) which adds some innovative features to contributions from (van Benthem and Minică, 2009).

The formal language used to describe questioning actions is given by the following BNF:

$$\varphi ::= \bot \mid p \mid i \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \Box_G\varphi \mid [\vec{\varphi}?]_H\psi \mid [!]_H\varphi$$

where p is a propositional symbol, i is a nominal symbol, $\Box \in \{K,R,Q,C\}$, an agent group G is a set of agent labels, singletons if $\Box \neq C$, $\vec{\varphi} = \langle \varphi_0, \ldots, \varphi_n \rangle$ is a vector of formulae, and $H = \langle \vec{G}_0, \ldots, \vec{G}_n \rangle$ is a vector of vectors containing groups of agents.

DELQ formulae are interpreted in Epistemic Issue Models (henceforth, *EIM*). *EIM*s are standard relational structures defined as tuples $M = \langle W, \overset{a}{\sim}, \overset{a}{\approx}, V, N \rangle$ where W_M is a set of epistemic alternatives, $\overset{a}{\sim}_M$ is a family of equivalence relations on W_M representing epistemic indistinguishability, $\overset{a}{\approx}_M$ is a family of binary relations, one for each agent $a \in \mathtt{A}$, representing issue-equivalence, $V_M : \mathtt{P} \to \wp(W_M)$ is a standard

Minică Ş. (2010).
IMPLEMENTING DYNAMIC-EPISTEMIC QUESTIONING - Engineering and Teaching Information Seeking via Dynamic Inquiry.
In *Proceedings of the 2nd International Conference on Computer Supported Education*, pages 367-372
DOI: 10.5220/0002857703670372

propositional valuation function, and $N_M : \mathtt{N} \to W$ is a naming function, for any given mutually disjoint countable sets $\mathtt{P}, \mathtt{N}, \mathtt{A}$ of propositional atoms, name symbols, and agent labels, respectively.

The interpretation of *DELQ* formulae is given recursively at a world w in an *EIM* by this clauses:

$$\begin{array}{lll} M \models_w \bot & \text{iff} & w \in \varnothing \\ M \models_w p & \text{iff} & w \in V(p) \\ M \models_w i & \text{iff} & w \in N(i) \\ M \models_w \neg\varphi & \text{iff} & \text{not } M \models_w \varphi \\ M \models_w \varphi \wedge \psi & \text{iff} & M \models_w \varphi \text{ and } M \models_w \psi \\ M \models_w \Box_G \varphi & \text{iff} & \forall v : w \leftrightarrow v \Rightarrow M \models_v \varphi \\ M \models_w [\alpha]_H \varphi & \text{iff} & M \otimes E \models_{Z_{M\otimes E}(w)} \varphi \end{array}$$

$$\text{where: } \leftrightarrow = \begin{cases} (\overset{a}{\sim})_{a\in G} & \text{if } \Box = K, \\ (\overset{a}{\approx})_{a\in G} & \text{if } \Box = Q, \\ (\overset{a}{\sim} \cap \overset{a}{\approx})_{a\in G} & \text{if } \Box = R, \\ (\bigcup_{a\in G} \overset{a}{\sim})^* & \text{otherwise} \end{cases}$$

and $M \otimes E = \langle W_{M\otimes E}, \overset{a}{\sim}_{M\otimes E}, \overset{a}{\approx}_{M\otimes E}, X_{M\otimes E}, Z_{M\otimes E}\rangle$ is the *EIM* obtained via the product update mechanism given below for any Epistemic Questioning Action Model (henceforth, *EQAM*) $E = \langle S, \overset{a}{\sim}, \overset{a}{\approx}, X, Z\rangle$ such that, for any questioning action $[\alpha]_H$ and *EIM* M:

$$S = \begin{cases} \bigcup_{\varphi_i\in\vec{\varphi}} \{\varphi_i, \overline{\varphi}_i, \{\varphi_i, \overline{\varphi}_i\}\} & \text{if } \alpha = \vec{\varphi}?, \\ \{e \mid e = N_M(w), w \in W_M\} & \text{otherwise} \end{cases}$$

$$\overset{a}{\sim} = \begin{cases} \{(e,e') \mid \exists s, s' : e \in s \wedge e' \in s'\} & \text{if } \alpha = \vec{\varphi}?, \\ \{(e,e') \mid N_M^{-1}(e) \overset{a}{\approx} N_M^{-1}(e')\} & \text{otherwise} \end{cases}$$

$$\overset{a}{\approx} = \begin{cases} \{(\varphi_i, \overline{\varphi}_i) \mid \varphi_i \in \vec{\varphi}, a \notin \vec{G}_i(2)\} & \text{if } \alpha = \vec{\varphi}?, \\ \{(e,e') \mid e \in S, e' \in S\} & \text{otherwise} \end{cases}$$

$$X(e) = \begin{cases} \mathtt{prq}(e) & \text{if } \alpha = \vec{\varphi}?, \\ \mathtt{exe}(e) & \text{otherwise} \end{cases}$$

$$Z(e) = \begin{cases} \mathtt{str}(e) & \text{if } \alpha = \vec{\varphi}?, \\ e^\frown ! & \text{otherwise} \end{cases}$$

with $\mathtt{prq} : S \to \wp(W_M)$, $\mathtt{exe} : S \to \wp(W_M)$ given by:

$$\mathtt{prq}(e) = \begin{cases} [\![e]\!]_M & \text{if } e \in \{\varphi, \overline{\varphi}\} \text{ for some } \varphi \in \vec{\varphi}, \\ [\![h(e)]\!]_M & \text{if } e = \{\varphi, \overline{\varphi}\} \text{ for some } \varphi \in \vec{\varphi} \end{cases}$$

where, for every $\varphi_i \in \vec{\varphi}$ and $\vec{G}_i \in H$, $h(\{\varphi_i, \overline{\varphi}_i\}) =$

$$\bigwedge_{a\in G_i(0)} (\neg(K_a\varphi_i \vee K_a\neg\varphi_i) \wedge \widehat{K}_a \bigvee_{b\in G_i(1)} (K_b\varphi \vee K_b\neg\varphi))$$

$$\mathtt{exe}(e) = \{w \mid N_{M\otimes E}^{-1}(N_M(w)^\frown !) \overset{\mathtt{A}}{\sim} x,\ x \in W^*_{M\otimes E}\}$$

where $W^*_{M\otimes E} \subseteq W_{M\otimes E}$, and $\overset{\mathtt{A}}{\sim} = (\overset{a}{\sim})_{a\in\mathtt{A}}$.[1]

Given an *EIM* $M = (\langle W, \overset{a}{\sim}, \overset{a}{\approx}, V\rangle, W^*)$ together with a designated set of worlds $W^* \subseteq W$, an *EQAM* $E = \langle S, \overset{a}{\sim}, \overset{a}{\approx}, X, Z\rangle$, the product update model $M \otimes E$ is defined as $M_\otimes = (\langle W_\otimes, \overset{a}{\sim}_\otimes, \overset{a}{\approx}_\otimes, V_\otimes, N_\otimes\rangle, W^*_\otimes)$ with:

$$W_\otimes = \{(w,e) \mid w \in W, e \in S\},$$

$$\mathtt{N}_\otimes = \{n^\frown e \mid (N^{-1}(n), Z(e)) \in W_\otimes\},$$

$$V_\otimes(p) = \{(w,e) \mid w \in V(p), e \in S\},$$

$$N_\otimes(m) = \{(V(n), Z^{-1}(e)) \mid n^\frown e = m\},$$

$$\overset{a}{\sim}_\otimes = \{((w,e),(w',e')) \mid w \in X(e,e') \Leftrightarrow w' \in X(e,e'),$$

$$w \in X(e) \Leftrightarrow w' \in X(e'), w \overset{a}{\sim} w', e \overset{a}{\sim} e'\},$$

$$\overset{a}{\approx}_\otimes = \{((w,e),(w',e')) \mid w \overset{a}{\approx} w', e \overset{a}{\approx} e'\},$$

$$W^*_\otimes = \{(w,e) \mid w \in W^*, e \in E\}.$$

Figure 1 gives an intuitive illustration of the way this mechanism works for a concrete example.

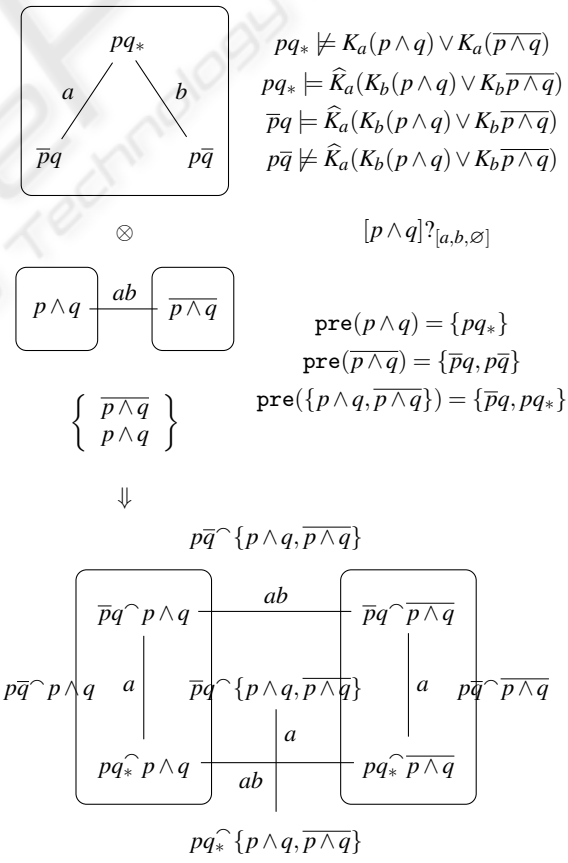

Figure 1: Product update for multi-agent questions.

[1] Conditions for executing both questioning and resolution actions can be different in a variety of specific contexts.

3 IMPLEMENTATION

This section presents an implementation for *DELQ* using *Haskell*. It will describe an *Haskell* program implementing model-checking for *DELQ* based on an already existing *Haskell* implementation for multi-agent Dynamic Epistemic Logic (henceforth, *DEL*) called Dynamic Epistemic MOdeling (henceforth, *DEMO*) developed in (van Eijck, 2004).

The *Haskell* implementation for the theory introduced in the previous section proceeds along the expected lines. Even though some design choices and a number of coding details are noteworthy the implementation is, essentially, a translation from the previous notation into *Haskell* functions and datatype definitions. We refrain from listing all the code lines and will focus on presenting a selection of main features.

We start by implementing a data structure for the formulae in the static fragment of our language:

```
data Formq = Top | Prop Prop | Neg Formq
 | Conj [Formq] | Disj [Formq]|K Agent Formq
 | X Agent Formq | O Agent Formq
 | CK [Agent] Formq |W Formq deriving (Eq,Ord)
```

*EIM*s are represented by the following data structure:

```
data QEM state = Qm [state] [Agent]
  [(Agent,state,state)] [(Agent,state,state)]
  [(state,[Prop])] [(state,String)]
  [state] deriving (Eq,Show)
```

The function `isTrueAt` gives the interpretation of the language in a relational structure. It is defined as:

```
trueAt :: ( Ord state ) =>
  QEM state -> state -> Formq -> Bool
trueAt m w Top = True
trueAt m@(Qm _ _ _ _ vl _ _) w (Prop p)=
 elem p (concat [props|(w',props)<-vl,w'== w])
trueAt m w (Neg f) = not (isTrueAt m w f)
trueAt m w (Conj fs) = and (map (trueAt m w) fs)
trueAt m w (Disj fs) = or (map (trueAt m w) fs)
trueAt m@(Qm _ _ acc _ _ _ _) w (K agt f) =
and(map(flip (trueAt m) f)(rightS (rlk agt m) w))
trueAt m@(Qm _ _ acc eqq _ _ _) w (X agt f) =
and(map(flip (trueAt m) f)(rightS (rlx agt m) w))
trueAt m@(Qm _ _ _ eqq _ _ _) w (O agt f) =
and(map(flip (trueAt m) f)(rightS (rlq agt m) w))
trueAt m@(Qm dom ags acc _ _ _ _) w (CK agts f)=
and(map(flip (trueAt m) f)(comAlts ac agts dm w))
trueAt m w (W f) = True
```

We use a similar data structure to represent *EQAM*s:

```
data QAM act = Qa [act] [Agent]
 [(Agent,act,act)] [(Agent,act,act)]
 [(act,[Int])] [(String,act)] deriving(Eq,Show)
```

To make some computations easier we store both epistemic events and their names in *EQAM* states:

```
type State = (Formq, String)
type Quac = ([Formq], Hrup)
type Hrup = [[[Agent]]]
```

Questioning actions are then pairs of lists of formulae and lists of agents that are relevant for the action. The following definition provides the construction of a relational structure that represents a questioning action:

```
quac :: (Show state, Eq state) =>
  QEM state -> Quac -> QAM State
quac m@(Qm dom ags acc eqq val nam act) qk =
(Qa dom' ags' acc' eqq' pex nam')
 where
   dom' = fdom m qk
   ags' = ags
   acc' = fsim m qk
   eqq' = fapr m qk
   pex  = fxpr m qk
   nam' = fnam m qk
```

EQAM components are constructed by auxiliary functions using as input *EIM*s and a questioning acts:

```
fdom :: (Show state) =>
  QEM state -> Quac -> [State]
fdom m@(Qm dom _ _ _ _ _ _) qk
  | fst qk == [W Top] = zip (map (\x->Top)
      [1..(length dom)]) (map show dom)
  | otherwise = (zip (fst qk) (map show
      (fst qk))) ++ (zip (map Neg (fst qk))
      (map show (map Neg (fst qk)))) ++
      (zip (map W (fst qk)) (map show
      (map W (fst qk))))
```

We only give here for illustration the code of the functions computing two of the components of an *EQAM*:

```
fsim :: (Show state, Eq state) =>
  QEM state->Quac->[(Agent, State, State)]
fsim m@(Qm _ ags acc eqq _ _ _) qk
 | fst qk == [W Top] = [(x,y,z) |
 x<-ags, y<-(fdom m qk), z<-(fdom m qk),
 (x,(who m (snd y)),(who m (snd z)))`elem`eqq]
 | otherwise = [(x,y,z) | x <- ags,
 y<-(fdom m qk),z<-(fdom m qk),(not('?' `elem`
 (show(fst y)))),(not('?'`elem`(show(fst z))))]
```

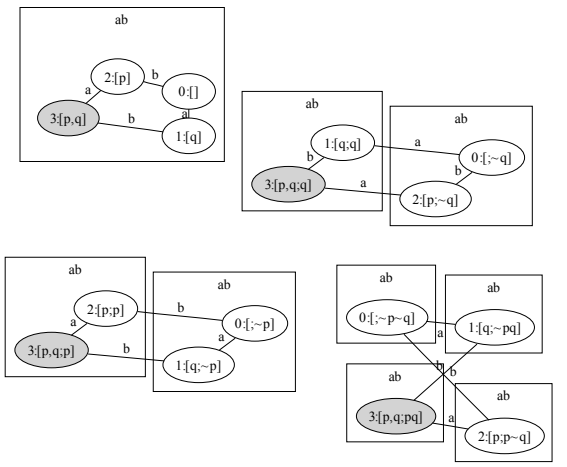

Figure 2: Example of *EIM*s before and after an updated with one or two questions in *Graphviz* generated format.

We supplement these abstract definitions by an intuitive illustration in *Graphviz* output before a resolution action is performed, see Figure 2.

The remaining functions implement the product update rule in a way that goes along the implementation for the standard product update rule for informative actions from *DEMO* (van Eijck, 2004). Most of the details are standard in DEL and we will skip them here, there are also some specific details that differ when questioning actions are considered. Some of the aspects that emerge when interrogative events are introduced are studied in (Baltag, 2001). Some other aspects are even more problematic both conceptually and technically when multiple questions are considered, in particular, there is no composition principle for questions and for complex questioning actions sequential or parallel execution lead to different outcomes. We do not pursue this direction here, a more detailed discussion of these and other related aspects can be found in (van Benthem and Minică, 2009).

4 APPLICATIONS

This section explores possible applications of results in previous section. We describe first applications in teaching and computer supported education and we also mention some further theoretical developments towards the end of the section.

A first possible application concerns the role of question-management actions in interactive education. We present ongoing work towards building a web application with a user-friendly graphical interface linked with *Haskell* and *Graphviz* server based programs in order to produce interactive and intuitive visual representations (*SVG* output) for computations using questioning actions that are changing epistemic-issue structures. The workflow diagram for this application is given in Figure 3. In what follows we will describe empirical data about the application as it is at this stage of development. Another possible application, interdependent on the first, is design and management of questioning strategies.

The first direction in which we intent to continue and develop results form previous sections is an immediate application inside an teaching-supporting electronic environment. We are aware of previous research in this direction that make the case for the advantages of having such teaching tools for a variety of domains and subjects. There is no reason to believe that the specific domain of logic for interactive interrogative actions is an exception from this point of view. Moreover, a suitable teaching environment for both interactive questioning and abstract issue management has the potential of producing useful applications in many other scientific domains. As long as recurrent interrogative structures are engaged in a specific field of study, there is good evidence to believe that a dynamic approach to questioning might be useful for identifying and understanding such patterns.

Some specific aspects of *DELQ* could make this task maybe more challenging. For instance, by the very fact that reasoning about epistemic and interrogative realities is done inside relational structures, an effective educational environment would have to provide feedback in both text messages and formulae describing computations and a more demanding graphical format. This is still an emerging practice but basic teaching experience seems to indicate that any type of feedback that would be relevant and efficient for teaching purposes should at least contain basic visualizations and processing of graph structures.

Besides allowing to perform model checking in epistemic structures, *DEMO* is also designed for producing output describing epistemic content of relational structures in a format that can be graphically displayed and used to enhance intuitively thrusted computations about interesting phenomena involving knowledge and interactive information flow. Given their very similar relational structure, it is reasonable to expect that *EIM*'s are also suitable candidates for a similar approach. There are also software solutions that support web-based viewers for *Graphviz* output. WebDot is such an example of a scripted service that could be used for visualizing graphs in HTML documents to be used as teaching tools.

More concretely, one specific feature of *DEMO* is the fact that it can output epistemic models in *Graphviz* format. Nodes of the graph represent possible worlds, edges between nodes represent indistinguishability relations, information about the agents and the valuation is represented by labels on edges and worlds. For our present purpose, because the issue relation, used to represend questions, is also an equivalence relation as is the epistemic indistinguishability relation, a representation of both as links between nodes becomes problematic. In order to reflect the conceptual difference between them we decided to use clustering of nodes for questions and edges for knowledge.

In order to obtain output in the previously specified graphical format we use the `fdp` layout program. Equivalence classes formed by issue relations are grouped in one cluster and labels are used to represent the agents. Figure 2 representa an *EIM* in which two agents a and b are ignorant about two propositions p and q. The initial frame, including all the alternatives, represents a universal issue relation. Ask-

ing a question (in this example p? or q?) splits the domain in two partition cell, represented here as clusters. Asking two questions in parallel splits the domain in four clusters at once.

At the moment all these are still under development and there are many aspects that can be improved. We mention some directions for future development in comparison to other existing established approches in the next section. Interliked with these we mention here some more theoretical aspects that could be attached once such a software architecture is in place. Its functionality could easily be expanded in other relevant directions in close interaction and by incorporating input from other related topics that are studied in DEL. Questioning actions are even more relevant in strategic interactions and over structured sequences or long-term processes of inquiry. Results and techniques used in studies of multi-agent strategic interaction in an epistemic framework such as public announcement games from (Ågotnes and van Ditmarsch, 2009) could be extended to include questioning actions and further enrich the present approach.

It is also often the case that the real practice of inquiry and scientific investigation also shows that there are various aspects that lead to various restrictions with regard to which kind of questioning actions are available in a certain context. Experimental designs come with complex patterns of dependence between what, how, and when is available for investigation. (van Benthem et al., 2009) introduces protocols to study information flow during structured sequences of epistemic actions. Such techniques are also relevant for questioning actions and could be used to study long-term design and management of questioning strategies in scientific inquiry.

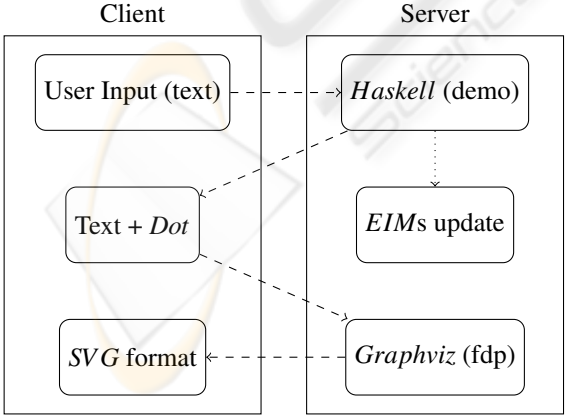

Figure 3: Workflow diagram for *DELQ* update.

5 COMPARISONS

There already exist previous software architectures that have both a web-based accessible and intuitive user interface and interactive feedback provided by a *Haskell* program running on a remote server to the student performing derivations in order to solve a specific exercise. (Gerdes et al., 2008) gives an exposition of such such a successful project. It uses a web-based user-friendly interface that communicates, using an *Ajax*-based solution, with *Haskell* functions computing next steps in derivations and giving feedback about strategic applications of rules in further steps in the form of text messages. Strategic feedback is also provided about which next steps are better in a specific exercise. Our approach does not include strategic feedback but such a feature is a desideratum for future development.

We are also aware of other possibilities for a suitable framework providing users access to computing operations on relational structures in a visual graphical format. One such example of an existing software solution that is easily available and might be also relevant in our context is *SAGE*. It is an open source software used for visualizing algebraic and geometric structures, its main features are described in (Stein and Joyner, 2005).

While previously mentioned approaches focus on algebraic structure of graphs in our approach relational structures are used as support for intensional reasoning using modal logic. There exist also previous approaches implementing reasoning with modal logic. *LoTREC* is such an example of a generic tableau-based theorem prover for intensional reasoning (Farias del Cerro et al., 2001). Compared with this our approach fosuses more on model checking tasks. But there exists an underlying calculs of issues and knowledge that could serve as a basis for adding validity testing and other syntactic functionality to our approach. On the other hand, the graphical representation of relational structures could be easily used as teaching aid for other related logical task.

6 CONCLUSIONS

We have presented a theoretical approach to questioning that builds on previous results in the field and introduces some innovative features in *DELQ*.

We have shown how these new features can be implemented in a *Haskell* program by extending previous implementations of similar aspects in *DEMO*.

We have presented ongoing work on a project of developing an educational electronic environment

that makes use of these results. Various possible solutions available for similar problems have been considered and their relevance for questioning actions have been assessed. Some of the difficulties and challenges for fulfilling such a project have also been considered.

Based on all these aspects we have argued for the potential benefits of a software architecture used for modeling questioning actions and visualizing epistemic issue structures in an electronic environment. We have found good reasons to believe that such a solution is very relevant and useful for educational purposes and could be used for teaching and enhancing important questioning skills.

We have also made a case for potential further extensions towards using our framework in understanding theoretical aspects, building practical skills and enhancing cognitive abilities, involved in strategic design and long-term management of questioning in multi-agent interaction or scientific investigation.

REFERENCES

Ågotnes, T. and van Ditmarsch, H. (2009). What will they say? - public announcement games. In *Proc. of Logic, Game Theory and Social Choice 6*. Tsukuba, Japan.

Baltag, A. (2001). Logics for insecure communication. In *Proc. of TARK'01*. Morgan Kaufmann.

Baltag, A. and Smets, S. (2008). A qualitative theory of dynamic interactive belief revision. *Texts in logic and games*, 3:9–58.

Ciardelli, I. and Roelofsen, F. (2009). Generalized inquisitive logic. In *Proc. of TARK-11*, pages 71–80. ACM.

Enqvist, S. (2009). Interrogative Belief Revision in Modal Logic. *Journal of Phil. Logic*, 38(5):527–548.

Farias del Cerro, L., Fauthoux, D., Gasquet, O., Herzig, A., Longin, D., and Massacci, F. (2001). Lotrec: the generic tableau prover for modal and description logics. In *International Joint Conference on Automated Reasoning*, LNCS, page 6. Springer Verlag.

Genot, E. (2009). The game of inquiry: the interrogative approach to inquiry and belief revision theory. *Synthese*, 171(2):271–289.

Gerdes, A., Heeren, B., Jeuring, J., and Stuurman, S. (2008). Feedback services for exercise assistants. In *Proc. of the 7th European Conference on e-Learning*. Agia Napa, Cyprus.

Groenendijk, J. (2008). Inquisitive semantics: Two possibilities for disjunction. In *7th Tbilisi Symposium on Language, Logic, and Computation.* Springer.

Hintikka, J., Halonen, I., and Mutanen, A. (2002). Interrogative logic as a general theory of reasoning. *Handbook of the Logic of Argument and Inference: The Turn Towards the Practical*, page 295.

Icard, T. (2009). Inquisitive Announcement Logic. *manuscript*.

Knuth, K. (2005). Toward question-asking machines: the logic of questions and the inquiry calculus. In *10th International Workshop on Artificial Intelligence and Statistics, Barbados*. Citeseer.

Peliš, M. (2009). Epistemic Logic and Questions.

Stein, W. and Joyner, D. (2005). Sage: System for algebra and geometry experimentation. In *ACM SIGSAM Bulletin, vol. 39*. ACM.

Unger, C. and Giorgolo, G. (2008). Interrogation in Dynamic Epistemic Logic. *13th ESSLLI Student Session*.

van Benthem, J., Gerbrandy, J., Hoshi, T., and Pacuit, E. (2009). Merging frameworks of interaction. In *Journal of Philosophical Logic*. Springer.

van Benthem, J. and Minică, Ş. (2009). Toward a dynamic epistemic logic of questions. In *Proc. of LORI'09, Second International Workshop on Logic, Rationality and Interaction, LNCS/FoLLI-LNAI 5834*. Springer.

van Eijck, J. (2004). Dynamic epistemic modelling. In *CWI Report SEN-E0424*. Available online from: http://www.cwi.nl/ jve/papers/04/demo/DEMO.pdf.

Wiśniewski, A. (1996). The logic of questions as a theory of erotetic arguments. *Synthese*, 109(1):1–25.