The Language of Social Software

Abstract

Computer software is written in languages like C, Java or Haskell. In many cases social software is expressed in natural language. The paper explores connections between the areas of natural language analysis and analysis of social protocols, and proposes an extended program for natural language semantics, where the goals of natural language communication are derived from the demands of specific social protocols.

1 Introduction

Social software is an umbrella term for algorithms that are designed to regulated social interactions. The term was coined by Rohit Parikh in [18]. Prototypical examples of social algorithms are fair division protocols, of which 'I cut, you choose' is the simplest example. Other examples are election procedures, protocols for establishing common knowledge in communities, negotiation procedures, auction protocols, protocols for contract signing, and so on.

Social software analysis is the attempt to analyse social procedures with tools from logic and computer science. Compare Euclid's greatest common divisor algorithm with the 'I cut, you choose' procedure. Euclid's procedure is a formal recipe, and it can be analysed by formal means. The correctness of the recipe follows from the insight that if A and B are two positive natural numbers with A greater than B, then replacing A by $A-B$ does not change the set of common divisors of the pair.

Similarly, we can do a formal analysis of 'cut and choose' [23]. If X is a set, then a valuation function V for X is a function from $\mathcal{P}(X)$ to $[0,1]$ with the properties that $V(\emptyset) = 0$, $V(X) = 1$, and $A \subseteq B \subseteq X$ implies $V(A) \leq V(B)$. Suppose V_m and V_y are functions for my and your valuation of the contents of X. In general, the two of us will value the items in X differently. Indeed, as was already observed by Steinhaus in 1948, if the two parties have different estimations, then there exists a division that gives both

parties more than their due part; "this fact disproves the common opinion that differences in estimation make fair division difficult"[23].

To analyse this, first consider the case that your valuation is unknown to me, and vice versa. Then if I cut, the best I can do is pick sets $A, B \subseteq X$ with $A \cap B = \emptyset$, $A \cup B = X$, and $V_m(A) = V_m(B)$. If you choose, you will use V_y to pick the maximum of $\{V_y(A), V_y(B)\}$. It follows immediately that cutting guarantees a fair share, but no more than that, while choosing holds a promise for a better deal. So if you ever get the choice between cutting and choosing in a situation where both parties only know their own valuation, then it is to your advantage to leave the cutting to the other guy. But if the valuations are common knowledge, the situation is reversed, for then it is more advantageous to take the role of cutter, for the cutter can attempt to make a division in A and B with A slightly more valuable than B according to the valuation of the other party, while B is much more valuable than A according to his own valuation (I thank Rohit Parikh for an illuminating discussion about this). The example shows that issues of knowledge and ignorance are crucial for analysis of fair division protocols. Still, in traditional studies of fair division the role of knowledge is not taken into account, as is witnessed by the comprehensive study of 'cake cutting algorithms' in [20].

Social software is closely connected with game theory, action logic, epistemic logic and social choice theory. If a social protocol can be stated precisely, it can be analysed with formal means. There is no doubt that the formal analysis of social procedures has the potential to shed new light on well established interaction protocols. In this paper we will also link up with natural language analysis. In particular, we suggest that the study of social mechanisms may offer an extended agenda for natural language analysis, with the analysis of natural language communication in settings where something more definite than just information exchange is the focus: achievement of some well stated goals given by specific social protocols.

The structure of the paper is as follows. Section 2 introduces discourse situations as important ingredients of social protocols, and gives an example of a social software protocol that hinges on common knowledge. A logical language for a formal analysis of communication in given discourse situations is presented in Section 3, and it is shown how this language can be used to describe whether particular communications can establish common knowledge. This language is used in Section 4 to analyse presuppositions in terms of the concept of common knowledge. Section 5 adds factual change to the logical toolbox, and shows how this addition allows for the analysis of performative speech acts, and for an analysis of the communicative act

of asking a question as a performative. In Section 6 an example of a real protocol is analysed with the tools presented in Sections 3 and 5. Section 7 concludes with a program for applying dynamic epistemic logic to natural language analysis.

2 Discourse Situations in Social Protocols

Discourse situations in natural language communication are set up to achieve broadly conceived social goals, like establishment of common knowledge. Let us look at a famous example of a social protocol: the judgement of Solomon. Here is the well-known story of the stratagem Solomon used to settle a dispute about a child. Each woman claims that the child is hers, and that the child of the other woman died.

> 23 Then said the king: The one saith, My child is alive, and thy child is dead. And the other answereth: Nay; but thy child is dead, and mine liveth. 24 The king therefore said: Bring me a sword. And when they had brought a sword before the king, 25 Divide, said he, the living child in two, and give half to the one and half to the other. 26 But the woman, whose child was alive, said to the king; (for her bowels were moved upon her child) I beseech thee, my lord, give her the child alive, and do not kill it. But the other said: Let it be neither mine nor thine; but divide it. 27 The king answered, and said: Give the living child to this woman, and let it not be killed; for she is the mother thereof. 28 And all Israel heard the judgment which the king had judged, and they feared the king, seeing that the wisdom of God was in him to do judgment.
>
> *From the First Book of Kings, Third Chapter*

For a rational reconstruction of this, see [17]. The key to the reconstruction is that the two women are forced to reveal their valuation of the child, by stating how much money they are willing to pay for it, or by how much community service they are willing to put up with in order to get it. Now suppose the child is worth A to the real mother and B to the pretender. We shall assume that A is much larger than B.

Solomon makes the following announcement: "I will ask one of you if you are willing to give the child to the other. If the answer is yes, the case is settled. If not, I will ask the the same question to the other person. Again, if the answer is yes, the case is settled. If both of you refuse to give up the

child, then I will have to sell it for what it is worth. I will toss a coin, and the one who gets the child will have to pay $\frac{A+B}{2}$, and the other pays a fine."

If the women act rationally, one of them will give up the child, which settles the case. It should be noted that this modified protocol is immune to strategic behaviour in a way that the original stratagem is not. For Solomon's original ploy hinged on the surprise effect: in a second dispute about a child, after "all Israel heard the judgement which the kings had judged", both parties in the dispute would no doubt exclaim that the child should stay alive, even if this meant that they would have to give it up.

What interests us here is that Solomon's announcement uses natural language, and that the fact that the announcement creates common knowledge is crucial to the mechanism. The common knowledge about what is going to happen makes it possible for the women to make a rational decision.

There is, by the way, a nice Indian version of this judgement procedure, as an Akbar and Birbal story [21]. In this version, Ramu and Shamu claimed ownership of the same mango tree, and decided to ask Birbal to settle the dispute. Birbal's verdict: "Pick all the fruits from the tree and divide them equally. Then cut down the tree and divide the wood." Ramu thought this was fair but Shamu was horrified, and Birbal declared Shamu the true owner. Again: the proclamation of the verdict uses natural language to create *common knowledge* between Ramu and Shamu.

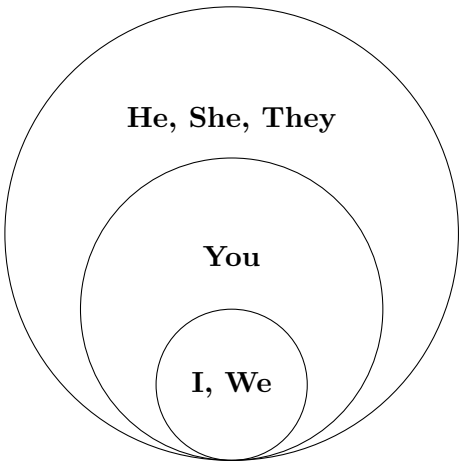


Figure 1: General Structure of a Discourse Situation.

The link between social software analysis and natural language analysis motivates a new focus for natural language semantics and pragmatics, for

analysis of social software calls for the study of natural language discourse situations where the discourse is part of a well-defined social mechanism.

Focussing on the fact that the verdicts in the Solomon case and the Birbal case are given in natural language, we can ask: what does the discourse convey to the two parties in the dispute, and how does the mechanism work?

- Natural language is used as a tool for creating (common) knowledge and changing (common) beliefs.
- Natural language also employs common knowledge and common belief to establish communication.
- Natural language is used to change the world by means of performatives: Solomon's announcement of how he is judging changes the world, because Solomon's words are uttered with the proper authority.

The three grammatical persons in natural language serve to refer to participants in a discourse, and they partition the discourse situation in three groups (see Figure 1):

- First person: **I, We**. Indicates the speaker, or the group represented by the speaker.
- Second person: **You**. Indicates the audience.
- Third person: **He, She, They.** indicates the outside world.

In many situations, the aim of discourse can be viewed as: create common knowledge between **Me** and **You**. In what follows, we will illustrate how dynamic epistemic logic, one of the key tools for analysing social situations, can be used to give an analysis of natural language discourse situations as well. We focus on the phenomena of presupposition processing and question answering.

3 DEL: A Language for Communication, Belief and Knowledge

As we have seen, an act of judgement can be viewed as a communication that changes the world, or, as philosophers of language call it, a performative action. To analyse what goes on in social protocols we need a logic for reasoning about knowledge, belief and communication in the presence of

operations that change the world. Dynamic epistemic logic or DEL (see [1] or [3] or the textbook treatment in [8]) offers these ingredients.

Fix a PDL style language for talking about epistemic plausibility. Assume p ranges over a set of basic propositions *Prop* and a over a set of agents *Ag*.

$$\begin{array}{rcl} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [\pi]\phi \\ \pi & ::= & a \mid a^{\smile} \mid ?\phi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

The intuition about the interpretation of a is that this gives the relation of (unconstrained) plausibility for agent a. Here $w \stackrel{a}{\rightarrow} v$ would express that agent a considers world v at least as plausible as world w. The advantage of interpreting a as plausibility rather than knowledge is that the typical properties of knowledge can be destroyed by actions of communcation and change that occur in social protocols. Assume Solomon knows a particular fact p. Then someone deceives him and makes that fact false. Solomon, who is unaware of this, still sticks to p, but can now no longer be said to have knowledge that p. This problem is avoided if no constraints at all are imposed on the plausibility relations for the agents involved in a social protocol.

These plausibility relations can be used to *define* knowledge. E.g., we can focus on individual knowledge of a by defining the operator $\sim_a$ as $(a \cup a^{\smile})^*$. This definition guarantees that the knowledge is reflexive, transitive and symmetric, even if no conditions are imposed on the plausibility relation $\stackrel{a}{\rightarrow}$ that interprets a. Next, the PDL operations can be used to define common knowledge, say between agents a and b. We stipulate that $\sim_{a,b}$ abbreviates $(\sim_a \cup \sim_b)^*$. This defines common knowledge between a and b as the reflexive transitive closure of the union of the knowledge relations for a and b. See [13] for an account for how this can be extended to (conditional) belief, and [10] for how it can be used to analyse multi-agent belief change.

This PDL language is to be interpreted in the usual PDL manner, with the relational expressions π interpreted as relations on the domain of the model. In particular, a is interpreted as the plausibility relation $\stackrel{a}{\rightarrow}$ given with the model. The intended interpretation of $a^{\smile}$ is the converse of this. Next, we interpret $\pi_1;\pi_2$ as relational composition, $\pi_1 \cup \pi_2$ as union of relations, and π^* as reflexive transitive closure.

Here is the formal version of the truth definition. Epistemic models $\mathbf{M}$ are triples (W, P, V), where W is a set of worlds, P is a function that assigns to each agent a a binary relation $\stackrel{a}{\rightarrow}$ on W, and V is a valuation function, i.e., V assigns to each world w in W a subset $V(w)$ of the set of basic propositions P.

Now let $\mathbf{M}$ be an epistemic model and let w be a world in the domain of $\mathbf{M}$. Then the notions of truth for formulas ϕ and relational interpretations of relation expressions π are given by simultaneous recursion, as follows:

$$\begin{array}{rcl}
\mathbf{M} \models_w \top & & \text{always} \\
\mathbf{M} \models_w p & \text{iff} & p \in V(w) \\
\mathbf{M} \models_w \neg\phi & \text{iff} & \text{not } \mathbf{M} \models_w \phi \\
\mathbf{M} \models_w \phi_1 \wedge \phi_2 & \text{iff} & \mathbf{M} \models_w \phi_1 \text{ and } \mathbf{M} \models_w \phi_2 \\
\mathbf{M} \models_w [\pi]\phi & \text{iff} & \text{for all } v \text{ with } (w,v) \in [\![\pi]\!]^{\mathbf{M}}, \mathbf{M} \models_w \phi
\end{array}$$

$$\begin{array}{rcl}
[\![a]\!]^{\mathbf{M}} & = & \{(w,v) \in W^2 \mid w \xrightarrow{a} v\} \\
[\![a^{\smile}]\!]^{\mathbf{M}} & = & \{(w,v) \in W^2 \mid v \xrightarrow{a} w\} \\
[\![?\phi]\!]^{\mathbf{M}} & = & \{(w,w) \in W^2 \mid \mathbf{M} \models_w \phi\} \\
[\![\pi_1;\pi_2]\!]^{\mathbf{M}} & = & \{(w,v) \in W^2 \mid \exists u \in W : (w,u) \in [\![\pi_1]\!]^{\mathbf{M}}, (u,v) \in [\![\pi_2]\!]^{\mathbf{M}}\} \\
[\![\pi_1 \cup \pi_2]\!]^{\mathbf{M}} & = & [\![\pi_1]\!]^{\mathbf{M}} \cup [\![\pi_2]\!]^{\mathbf{M}} \\
[\![\pi^*]\!]^{\mathbf{M}} & = & ([\![\pi]\!]^{\mathbf{M}})^{\star}
\end{array}$$

In the final clause, $([\![\pi]\!]^{\mathbf{M}})^{\star}$ expresses the reflexive transitive closure of the $[\![\pi]\!]^{\mathbf{M}}$ relation.

This logic is axiomatised by the standard PDL rules and axioms ([22, 16]) plus axioms that define the meanings of the relational converses $a^{\smile}$. The PDL rules and axioms are:

Modus ponens	and axioms for propositional logic
Modal generalisation	From $\vdash \phi$ infer $\vdash [\pi]\phi$

Normality	$\vdash [\pi](\phi \to \psi) \to ([\pi]\phi \to [\pi]\psi)$
Test	$\vdash [?\phi]\psi \leftrightarrow (\phi \to \psi)$
Sequence	$\vdash [\pi_1;\pi_2]\phi \leftrightarrow [\pi_1][\pi_2]\phi$
Choice	$\vdash [\pi_1 \cup \pi_2]\phi \leftrightarrow ([\pi_1]\phi \wedge [\pi_2]\phi)$
Mix	$\vdash [\pi^*]\phi \leftrightarrow (\phi \wedge [\pi][\pi^*]\phi)$
Induction	$\vdash (\phi \wedge [\pi^*](\phi \to [\pi]\phi)) \to [\pi^*]\phi$

The relation between the basic programs a and $a^{\smile}$ is expressed by the standard modal axioms for converse:

$$\vdash \phi \to [a]\langle a^{\smile}\rangle\phi \qquad \vdash \phi \to [a^{\smile}]\langle a\rangle\phi$$

Next step in the set-up is to add operators for communication. The most straightforward example is public announcement. Use $[!\phi_1]\phi_2$ to express the following:

If ϕ_1 is true then after public announcement of ϕ_1, it will be the case that ϕ_2 is true.

Let PDL+PA be the result of adding a clause $[!\phi]\phi$ to the definition of PDL.

Formally, $!\phi$ is interpreted as a model changing operation. It changes an epistemic model $\mathbf{M} = (W, P, V)$ to a model $\mathbf{M}^\phi = (W', P', V')$ given by

$$\begin{aligned} W' &= \{w \in W \mid \mathbf{M} \models_w \phi\} \\ P'(a) &= (\stackrel{a}{\rightarrow}) \cap W'^2 \\ V'(w) &= V(w). \end{aligned}$$

The semantics of $[!\phi_1]\phi_2$ is now given by:

$$\mathbf{M} \models_w [!\phi_1]\phi_2 \text{ iff } \mathbf{M} \models_w \phi_1 \text{ implies } \mathbf{M}^{\phi_1} \models_w \phi_2.$$

An important fact about public announcement is that everything that can be expressed in PDL+PA can already be expressed in PDL without announcements. In other words: the addition of public announcements $[!\phi_1]\phi_2$ to the language does not increase expressive power.

This can be shown by establishing that for every ϕ and every α there is a $T^\phi(\alpha)$ with the following property:

$$\mathbf{M} \models_w [!\phi][\alpha]\psi \text{ iff } \mathbf{M} \models_w [T^\phi(\alpha)][!\phi]\psi.$$

Here is the definition of this function (this is based on Van Benthem's analysis of public announcement in terms of relativized common knowledge; see [4]):

$$\begin{aligned} T^\phi(a) &= ?\phi\,;a \\ T^\phi(?\psi) &= ?(\phi \wedge [!\phi]\psi) \\ T^\phi(\alpha_1;\alpha_2) &= T^\phi(\alpha_1)\,;T^\phi(\alpha_2) \\ T^\phi(\alpha_1 \cup \alpha_2) &= T^\phi(\alpha_1) \cup T^\phi(\alpha_2) \\ T^\phi(\alpha^*) &= (?\phi\,;T^\phi(\alpha))^* \end{aligned}$$

But it turns out that we can be much more general than this, by using the definition of update models $\mathbf{A}$ and of the update product operation $\otimes$ following Baltag, Moss, Solecki [1]. An action model is like a preference model, but with the valuation replaced by a precondition map **pre**. The individual states in an action model are called actions, and the precondition map assigns to every action a formula of the language.

Updating a static model $\mathbf{M} = (W, P, V)$ with an action model $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre})$ results in new static model $\mathbf{M} \otimes \mathbf{A} = (W', P', V')$, where the new worlds are pairs (w, e) with $w \in W$ and $e \in E$, subject to the condition that (w, e) occurs in the update result iff the precondition of e in the action model holds in world w.

More precisely, the result of updating $\mathbf{M} = (W, P, V)$ with an action model $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre})$ is the model $\mathbf{M'} = (W', P', V')$ where $W' = \{(w, e) \mid w \in W, e \in E, \mathbf{M} \models_w \mathbf{pre}(e)\}$, P' is given by $\{(w, e)P'_i(w', e')$ iff wP_iw' and $e\mathbf{P}_ie'$, and $V'(w, e) = V(w)$. (w, e) is a distinguished world of $\mathbf{M'}$ iff w is a distinguished world of $\mathbf{M}$ and e is a distinguished event of $\mathbf{A}$.

The intuitive idea of distinguished states and distinguished events is that the actual state (or 'the real world') has to be one of the distinguished states. Likewise, the actual event (the event that actually takes place has to be one of the distinguished events. Using distinguished states makes it easy to indicate when an information update of a model is unsuccessful: this happens precisely when its set of distinguished states becomes empty, indicating that there is no room in the model for the actual world.

Let PDL+U be the result of extending the PDL language with update operations $[A, S]\phi$, where A is an update model, and S is a subset of the state set of A (the set of distinguished events). Since formulas of PDL+U can occur as preconditions in action models, we now have to define update and truth by mutual recursion, as follows. We define the update of (M, U) with (A, S), where U is the set of distinguished states for model M and S is the set of distinguished events of action model A. Let

$$(M, U) \otimes (A, S)$$

be given by

$$((W', V', R'), U'),$$

where

$$\begin{array}{lcl} W' & := & \{(w, s) \mid w \in W_M, s \in W_A, M \models_w \mathrm{pre}_s\}, \\ V'(w, s) & := & V_M(w), \\ (w, s) \xrightarrow{i} (w', s') \in R' & :\equiv & w \xrightarrow{i} w' \in R_M, s \xrightarrow{i} s' \in R_A, \\ U' & := & (U \times S) \cap W', \end{array}$$

and where the truth definition is given by:

$$
\begin{array}{lcl}
M \models_w \top & & \text{always} \\
M \models_w p & :\equiv & p \in V_M(w) \\
M \models_w \neg\phi & :\equiv & \text{not } M \models_w \phi \\
M \models_w \phi_1 \wedge \phi_2 & :\equiv & M \models_w \phi_1 \text{ and } M \models_w \phi_2 \\
M \models_w [\alpha]\phi & :\equiv & \text{for all } w' \text{ with } w \xrightarrow{\alpha} w' \; M \models_{w'} \phi \\
M \models_w [A, S]\phi & :\equiv & M' \models_{(w,s)} \phi \text{ for all } s \in S \text{ with } M \models_w \mathbf{pre}_s, \\
 & & \text{where } M' = (M, \{w\}) \otimes (A, S),
\end{array}
$$

with $\xrightarrow{\alpha}$ given by

$$
\begin{array}{lcll}
\xrightarrow{i} & = & R_M(i) & \\
\xrightarrow{?\phi} & = & \{(x, x) \mid M \models_x \phi\} & \\
\xrightarrow{\alpha_1 \cup \alpha_2} & = & \xrightarrow{\alpha_1} \cup \xrightarrow{\alpha_2} & \\
\xrightarrow{\alpha_1;\alpha_2} & = & \xrightarrow{\alpha_1} \circ \xrightarrow{\alpha_2} & \text{(relational composition of } \xrightarrow{\alpha_1} \text{ and } \xrightarrow{\alpha_2}) \\
\xrightarrow{\alpha^*} & = & (\xrightarrow{\alpha})^* & \text{(transitive closure of } \xrightarrow{\alpha}).
\end{array}
$$

Note that if the static model $\mathbf{M}$ has a set of distinguished states U and the action model a set of distinguished events S, then the distinguished worlds of $\mathbf{M} \otimes A$ are the (w, s) with $w \in U$ and $s \in S$.

The following examples should clarify these definitions. We will represent worlds in static models as circles and events in action models as boxes. Distinguished worlds and events are shaded grey. This indicates that the actual world has to be one of the shaded worlds. Models without a shaded world leave no room for reality, so to speak: their set of candidates for the actual world is empty.

Figure 2: Static model and update model

Figure 2 gives an example pair of a static model with an update action. The static model, on the left, pictures the result of a hidden coin toss, with three onlookers, Alice, Bob and Carol. The update model represents a secret test to the effect that the toss is h. Here and henceforth, we leave out reflexive arrows, and use "—" for bidirectional links.

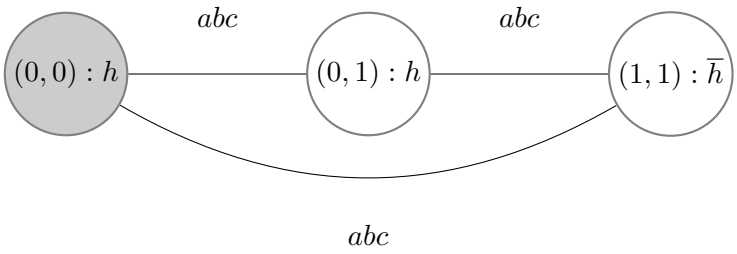


Figure 3: Result of the update in Figure 2.

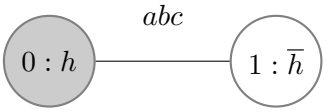


Figure 4: Bisimulation-minimal version of result of the update in Figure 2.

The literal result of the update is given in Figure 3. This can be contracted to a bisimulation minimal model: see Figure 4. The result of the update is that the distinction mark on the $\overline{h}$ world has disappeared, without any of a, b, c being aware of the change.

Factual change was added to update models in LCC [3], by means of propositional substitutions (see also [6]). A propositional binding is a map from proposition letters to formulas, represented by

$$\{p_1 \mapsto \phi_1, \ldots, p_n \mapsto \phi_n\}$$

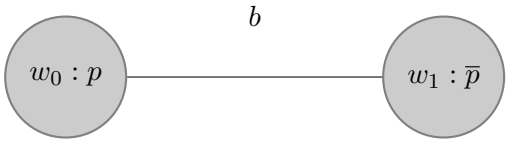
where the p_k are all different, and where no ϕ_k is equal to p_k. It is assumed that each p that does not occur in a lefthand side of a binding is mapped to itself.

Belief change can be added in a similar manner, by means of relational substitutions. A relational binding is a map from agents to program expressions, represented by

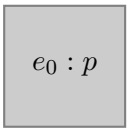
$$\{a_1 \mapsto \pi_1, \ldots, a_n \mapsto \pi_n\}$$

See [2] for further details on how this is used for belief revision.

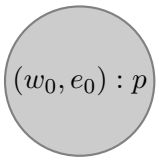
As an example of how public announcement creates common knowledge, consider the following situation where a knows whether p is true while b does not know. Both states of affairs might be actual:



Now an update action consisting of a public announcement of p takes place.

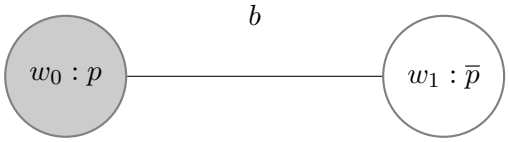


Here is the update result:



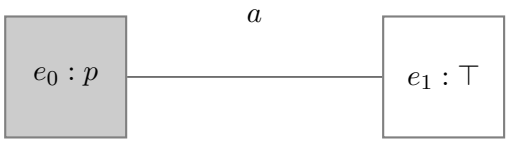
All agents now know that p is the case. In fact, p has become common knowledge: all agents know that all agents know that ... that p is the case.

It is well-known that message exchange cannot create common knowledge [15]. Here is a DEL version of the story that is often used to illustrate this, the coordinated attack problem. Two generals a, b will succeed if they attack together at a certain time, otherwise they will be defeated. Here is a picture of the situation where a has decided to attack (p), but b does not know this:

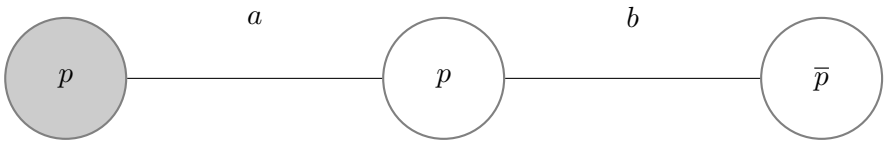


For an appropriate DEL encoding of what goes on we needs to find the right update action. Admittedly, this is more art than science, but the following is reasonable. The update action for general a consists of sending a message p. Only, he cannot be sure that his message will get across, for the messenger

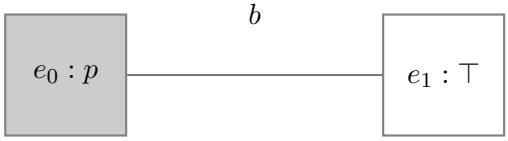
has to cross ennemy territory and might get captured. This creates an uncertainty for a about which event is going to take place, an uncertainty that is captured by the link with an event where nothing happens. Let us say that *in fact* the messenger gets through, although a does not know this:



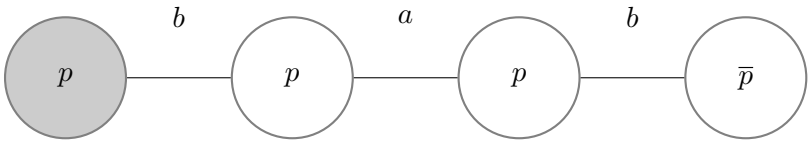
The situation after this first message from general a gets across is like this:



The update action for general b is symmetric. b will send an acknowledgement of p, but just like the other general, he cannot be sure that this message gets across:



The situation after this second event is like this:



And so it goes on ...

We can contrast this with real life situations where co-presence creates common knowledge. A prime example of this is cash withdrawal from a bank. You withdraw a large amount of money from your bank account and have it paid out to you in cash by the cashier. Typically, what happens is

this. The cashier looks at you earnestly to make sure she has your full attention, and then she slowly counts out the banknotes for you: one thousand (counting ten notes while saying *one*, *two*, *three*, ..., *ten*), two thousand (counting another ten notes), three thousand (ten notes again), and four thousand (another ten notes). This ritual creates common knowledge that forty banknotes of one hundred euros each were paid out to you. To see that this is different from mere knowledge, consider the alternative where the cashier counts out the money out of sight, puts it in an envelope, and hands it over to you. At home you open the envelope and count the money. Then the cashier and you have knowledge about the amount of money that is in the envelope. But the amount of money is not common knowledge among you. In order to create common knowledge you will have to insist on counting the money while the cashier is looking on, making sure that you have her full intention. For suppose you fail to do that. On recounting the money at home you discover there has been a mistake. One banknote is missing. Then the situation is as follows: the cashier believed that she knew there were forty banknotes. You now know there are only thirty-nine. How are you going to convince your bank that a mistake has been made, and that it is their mistake?

If you reflect on this, you see that what happens here is different from what happens when money is paid out to you by an ATM. The machine counts the money, dispenses it, and you can count it afterwards. You have no part in the counting process by the machine, and the machine has no part in your counting process, so this procedure does *not* create common knowledge between you and the machine. Confidence has to be created in other ways: by making the machines so reliable that errors seldom occur, by carefully building a reputation that complaints about malfunctioning ATMs are always taken seriously, and so on.

And here again an analogy with natural language analysis suggests itself. Paying out money is a performative act, and it can be analysed as such with the DEL toolset: counting out *one*, *two*, ... *ten*, and handing the notes to me is a communicative action that also changes the world. Technically, this is done with an update action consisting of a public announcement together with a public change (a substitution that changes certain truth values in the world).

4 Presuppositions and Common Knowledge

.

A presupposition of an utterance is an implicit assumption about the world or a background belief shared by speaker and hearer in a discourse.

Shall we do it again? (1)

Presupposition: we have done it before.

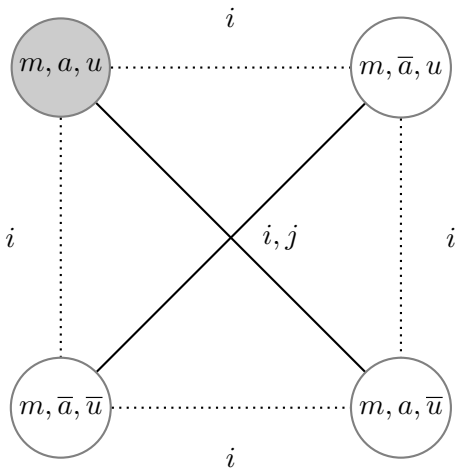
Jan is a bachelor. (2)

First presupposition: 'Jan' refers to a male person. (True in the Netherlands and Poland, false in the United Kingdom.) Second presupposition: 'Jan' refers to an adult. Third presupposition: 'bachelor' presupposes 'male' and 'adult'. To analyse this in the spirit of DEL [11], we need DEL with public announcements. Let $[!\phi]\psi$ express that after public announcement of ϕ, ψ holds. Formally:

$$M \models_w [!\phi]\psi \text{ iff } (M \models_w \phi \text{ implies } M \mid \phi \models_w \psi).$$

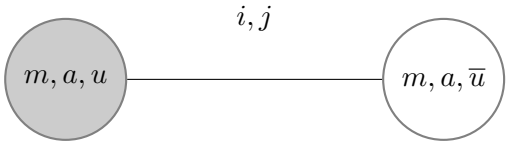
Now consider the special case of an update of the form "it is common knowledge between i and j that ϕ", i.e., an update of the form $![\sim_{i,j}]\phi$. Then we have the following:

- In case ϕ is already common knowledge, this update does not change the model.
- In case ϕ is not yet common knowledge, the update leads to a model without actual worlds.



Solid lines for i, j accessibilities, dotted lines for i accessibilities. m for 'male', a for 'adult', u for 'unmarried'. Note that j does not know about u and i does not know about a, u. Note also that $[\sim_{ij}]m$ holds, but $[\sim_{ij}]a$ and $[\sim_{ij}]u$ do not hold.

To analyze presupposition in terms of common knowledge, we can view presuppositions as pieces of common knowledge shared between speaker and hearer in a discourse. In the following model, m and a are common knowledge between i and j:



An update with 'bachelor' conveys that 'male' and 'adult' are presupposed (common knowledge), while 'unmarried' is conveyed:

$$[\sim_{ij}](m \wedge a) \wedge u$$

Here is the update result:

Note the following fact about public announcement of common knowledge:

$$M \models_w [!{\sim_{ij}}]\phi]\psi \text{ iff } M \models_w [\sim_{ij}]\phi \rightarrow \psi.$$

This says that public announcement of common knowledge has the force of an implication.

$$M \models_w [!([\sim_{ij}]\phi \wedge \phi')]\psi \text{ iff } M \models_w [!{\sim_{ij}}]\phi][!\phi']\psi.$$

This says that putting a presupposition before an assertion has the same effect as lumping them together.

Within this framework we can analyse presupposition projection, by checking how sequential composition of statements affects the presuppositions of the components. For a trivial but illustrative example, let us consider the update without presupposition $!m$ (the statement **male**) followed

by the update for **bachelor**). Using C for $[\sim_{ij}]$ and $C(\phi, \psi)$ for $[!\phi][\sim_{ij}]\psi$, we get:

$$\begin{aligned}
[!m][!(C(m \wedge a) \wedge u)]\chi &\leftrightarrow [!(m \wedge [!m](C(m \wedge a) \wedge u))]\chi \\
&\leftrightarrow [!(m \wedge [!m]Cm \wedge [!m]Ca \wedge [!m]u)]\chi \\
&\leftrightarrow [!(m \wedge [!m]Ca \wedge [!m]u)]\chi \\
&\leftrightarrow [!(m \wedge C(m, a) \wedge m \to u)]\chi \\
&\leftrightarrow [!(C(m, a) \wedge m \wedge u)]\chi
\end{aligned}$$

So the presuppositional part of the combined statement is $C(m, a)$, and the assertional part is $m \wedge u$.

Next, look at what the literature calls presupposition accommodation. Suppose p is common knowledge. Then updating with statement $!(Cp \wedge q)$ has the same effect as updating with $!q$. Suppose on the other hand that p is true in the actual world but not yet common knowledge. Then updating with $!(Cp \wedge q)$ will lead to an inconsistent state, but updating with $!p$ followed by an update with $!(Cp \wedge q)$ will not. Accommodation of the presupposition would consist of replacement of $!(Cp \wedge q)$ by $[!p][!(Cp \wedge q)]$. By invoking the Gricean maxim 'be informative' one can explain why $[!p][!(Cp \wedge q)]$ is *not* appropriate in contexts where p is common knowledge.

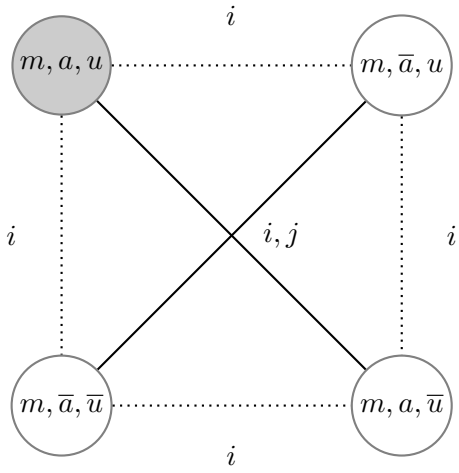
5 Performatives and Question Answering

In order to analyse performative acts of communication, we need to use the operation of public change. For this we use $[p := \phi]\psi$, with the semantic stipulation that this is true in world w of M if ψ is true in world $w^{p:=[\![\phi]\!]_w}$ of $M^{p:=[\![\phi]\!]}$. Here $w^{p:=[\![\phi]\!]_w}$ is the result of changing the valuation of w in such manner that p gets value $[\![\phi]\!]_w$. Thus, $p := \phi$ changes the model M to $M^{p:=[\![\phi]\!]}$. Note that the command $p := \phi$ also makes sense if p occurs in ϕ. An example is the command $p := \neg p$, which just swaps the truth value of p in every valuation in the model. Here are some examples of performative speech acts.

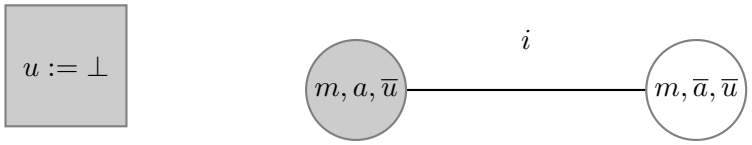
I call you Adam. (3)

I declare you man and wife. (4)

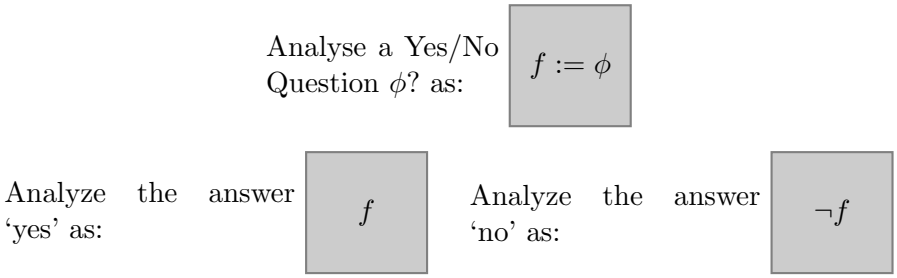
Let's illustrate the modelling of the marriage ceremony in DEL with public changes. We suppose the initial situation looks like this:



Below left is the action model for public change, and right the result of updating the above model with this.



Other important ingredients in social protocol are questioning and question answering. We illustrate how DEL can be used to give an analysis of the notion of appropriateness of a yes/no question. See [14] for a standard account. Our DEL analysis is based on the treatment in the last chapter of [12]. Other relevant work in this area can be found in [**?**]. Let f be a propositional variable for *question focus*.



Question: 'Is Johnny married?' (5)

Answer: 'Johnny is not an adult.' (6)

This answer is *appropriate*: updating with this answer makes 'John is not married' common knowledge. The update entails the answer 'no'.

Thus, if the question modelled as an update with the public change action $f := \phi$, then the answer ψ is appropriate if either updating with ψ has the effect that f becomes common knowledge, or updating with ψ has the effect that $\neg f$ becomes common knowledge.

6 Social Software Protocol Analysis with DEL

Finally, let us turn to the analysis of an example social software protocol. We will first give a DEL analysis of a protocol that solves a riddle, and next ask a meta-question: how can we explain that the agents in the protocol agree to adopt the protocol? It will turn out that a rational explanation can be given, but this requires an extension of DEL. Here is the riddle.

> A group of 100 prisoners, all together in the prison dining area, are told that they will be all put in isolation cells and then will be interrogated one by one in a room containing a light with an on/off switch. The prisoners may communicate with one another by toggling the light-switch (and in no other way). The light is initially switched off. There is no fixed order of the interrogations. Every day one prisoner will get interrogated. At any stage every prisoner will be interrogated again sometime.
>
> When interrogated, a prisoner can either do nothing, or toggle the light-switch, or announce that all prisoners have been interrogated. If that announcement is true, the prisoners will (all) be set free, but if it is false, they will all be executed. Can the prisoners agree on a protocol that will set them free?

There are several ways for solving this [7]; we will discuss the simplest protocol that works, and then move on to the meta question: how can the prisoners agree on adopting the protocol?

For $n \leq 2$ the riddle is uninteresting, so assume there are $n > 2$ prisoners The n prisoners appoint one among them as the *counter*. All prisoners except the counter act as follows: the first time they enter the room when the light is off, they switch it on; on all subsequent occasions, they do nothing. The counter acts as follows: The first $n - 2$ times that the light is on when he enters the interrogation room, he turns it off. Then the next time he enters the room when the light is on, he (truthfully) announces that everybody has been interrogated.

For simplicity, let us assume there are three prisoners $0, 1, 2$, with 0 acting as counter. Let e_0, e_1, e_2 be the interrogation events of the three prisoners. Let p express that the light is on. For example: if the light is on and if event e_0 (interrogation of the counter) takes place, then afterwards the light is off, and the counter knows that it is off:

$$p \to [e_0]K_0\neg p.$$

Let q_i, for $i = 1, 2$, express that prisoner i has been interrogated at least once. Then the following is true:

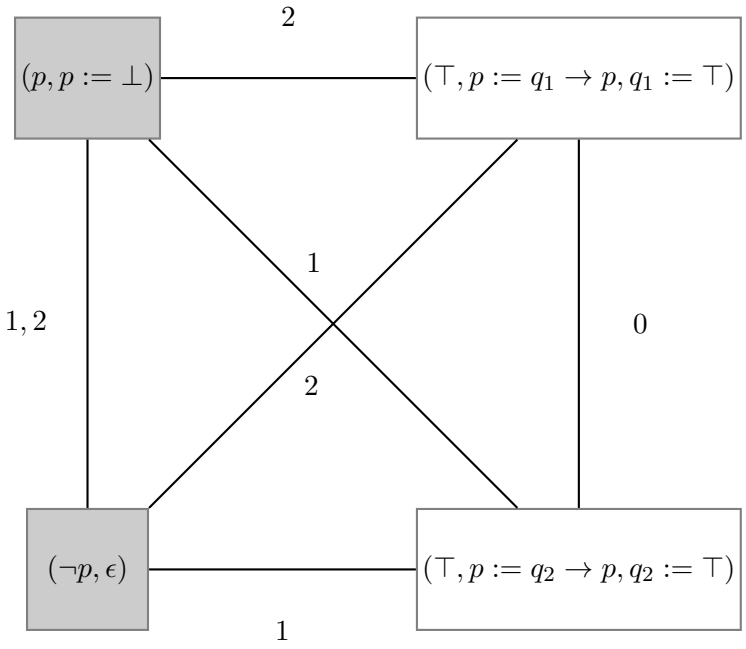
$$[e_1]q_1.$$

And similarly for prisoner 2. After the events of their interrogation, the prisoners have been interrogated at least once.

The key to the correct modelling of the protocol is the construction of the update model. The update model for the counter update should distinguish between two cases: the light is on (p) or the light is off ($\neg p$). We call these events e_{0p} and $e_{0\overline{p}}$. This distinction is crucial, for we want to model the fact that the counter *learns* from what he sees when he enters the interrogation room. In case the counter finds the light on, he switches it off ($p := \bot$); in case he finds the light off, he does nothing (ϵ). The counter can of course distinguish between these two events, for he sees whether the light is on or off, but the other two prisoners will confuse them.

For the two other prisoners it only matters what they do, and what they do is conditional on whether they have been interrogated before. Consider prisoner 1. A bit of reflection shows that the following action takes place: $p := q_1 \to p, q_1 := \top$. The assignment $p := q_1 \to p$ ensures that if the light is on nothing changes, and if the light is off, it is turned on on condition that q_1 is false. The assignment $q_1 := \top$ makes q_1 true, to express that at least one interrogation of 1 has taken place. The treatment of the event for the other non-counting prisoner is similar.

From the perspective of the counter, the update model looks like this (this is in fact the implementation of update e_0):



Note that $1, 2$ cannot see the difference between interrogation of 0 with the light on or off. $1, 2$ also cannot see the difference between the counter interrogation and the interrogation of the other non-counting prisoner. Finally, the counter cannot tell the interrogations of the other two prisoners apart.

From the perspective of prisoner 1 the update model looks similar, but now with the event where $p := q_1 \to p, q_1 := \top$ takes place as the actual event. Call this update e_1. Finally, update e_2 is given by the same action model, but now with the event where $p := q_2 \to p, q_2 := \top$ takes place as the actual event.

It is not hard to see that this gives a correct modelling of what takes place during the interrogations, so it gives us a semantics for $[e_i]\phi$, with i ranging over $1, 2, 3$. Note that the update model can be viewed as a formal version of the protocol that the prisoners have agreed on.

Now let us turn to the meta question. Why should the prisoners agree on this protocol? After all, it is a matter of life and death to them. Intuitively, the explanation is that it is common knowledge that if the interrogation sequence is fair (in the sense that every prisoner will always be interrogated again at some day in the future), then at some point in the future the counter will know that all have been interrogated. Also, it is common knowledge

that the counter will never falsely believe that the prisoners have all been interrogated. Can we express such facts in DEL? No, for DEL does not allow temporal reasoning. But we can express them in the extension of DEL with temporal operators (call this DEL + LTL):

$$\begin{array}{rcl} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [\pi]\phi \mid [e]\phi \mid \\ & & Ne \mid F\phi \mid G\phi \mid P\phi \mid H\phi \\ \pi & ::= & a \mid a^{\smile} \mid ?\phi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \\ a & ::= & 0 \mid 1 \mid 2 \\ e & ::= & e_0 \mid e_1 \mid e_2 \end{array}$$

The intended semantics of $[e]\phi$ is the DEL semantics: either update with event e fails, or in the updated model ϕ holds. The intended semantics of Ne is that the next event is e. The meanings of F, P, G, H are the usual ones from linear time logic (LTL; see [19]).

Formally, interpretation takes place with respect to infinite sequences of events. If σ is such a sequence and if n is a positive natural number, then we use σ_n is the n-th event of the sequence. So σ looks like $\sigma_1, \sigma_2, \ldots$ Then $M_{\sigma,n}$ is the model which results from doing updates $\sigma_1, \ldots, \sigma_n$ on the initial model (where the light is off and everyone knows that). Since all updates are functional, this is well-defined. Here is an example of an interrogation sequence where the prisoners get interrogated indefinitely in the sequence $0, 1, 2$:

$$e_0, e_1, e_2, e_0, e_1, e_2, e_0, e_1, e_2, e_0, e_1, e_2, \ldots$$

The truth definition for this language defines the relation $(\sigma, n) \models \phi$, as follows. $(\sigma, n) \models p$ iff p is true in $M_{\sigma,n}$. Booleans and epistemic operations are interpreted as usual, using $M_{\sigma,n}$ for (σ, n). The clauses for the temporal operators run like this:

$$\begin{array}{lll} (\sigma, n) \models Ne & \text{iff} & \text{if } \sigma_{n+1} = e \text{ (the next event in the sequence } \sigma \text{ equals } e) \\ (\sigma, n) \models F\phi & \text{iff} & \text{if for some } m > n,\ (\sigma, m) \models \phi. \\ (\sigma, n) \models G\phi & \text{iff} & \text{if for all } m > n,\ (\sigma, m) \models \phi. \\ (\sigma, n) \models P\phi & \text{iff} & \text{if for some } m < n,\ (\sigma, m) \models \phi. \\ (\sigma, n) \models H\phi & \text{iff} & \text{if for all } m < n,\ (\sigma, m) \models \phi. \end{array}$$

With this, we can express what it means that an interrogation sequence is fair:

$$G(FNe_0 \wedge FNe_1 \wedge FNe_2).$$

Knowledge of 0 that prisoners 1 and 2 have been interrogated:

$$[\sim_0](PNe_1 \wedge PNe_2).$$

Correctness of the protocol:

$$G(FNe_0 \wedge FNe_1 \wedge FNe_2) \rightarrow F[\sim_0](PNe_1 \wedge PNe_2).$$

Common knowledge of correctness of the protocol:

$$[\sim_{012}](G(FNe_0 \wedge FNe_1 \wedge FNe_2) \rightarrow F[\sim_0](PNe_1 \wedge PNe_2)).$$

An implementation of epistemic model checking for this example is available from the author upon request [9].

7 Conclusions

Common knowledge and common belief are central notions in (natural language) discourse analysis and in social software. An interesting program for natural language semantics would be to analyze discourse as sequences of public announcements to the discourse participants, using the tools from dynamic epistemic logic. Public announcement logic can also shed light on the analysis of presupposition, presupposition projection and presupposition accommodation. Yes/no questions can be analyzed as public change of focus. Appropriate answers can be analysed in terms of 'same update effect'. A program for DEL applied to natural language analysis would be to extend the hints given above to a full semantic/pragmatic theory of questions and answers. Finally on the agenda is the program of investigating the logic of DEL + LTL, comparing with the more general perspective of [5], analyzing further social software protocols in DEL + LTL, and develop model checking tools for this language.

Acknowledgement I wish to thank two anonymous *Synthese* reviewers for useful advice on presentation.

References

[1] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In I. Bilboa, editor, *Proceedings of TARK'98*, pages 43–56, 1998.

[2] J. van Benthem and F. Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 14(2):157–182, 2007.

[3] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[4] Johan van Benthem. Information update as relativization. Technical report, ILLC, Amsterdam, 2000. Available from `http://staff.science.uva.nl/~johan/Upd=Rel.pdf`.

[5] Johan van Benthem, Jelle Gerbrandy, Tomohiro Hoshi, and Eric Pacuit. Merging frameworks for interaction. *Journal of Philosophical Logic*, 38(5):491–526, 2009.

[6] Hans van Ditmarsch and Barteld Kooi. Semantic results for ontic and epistemic change. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, Texts in Logic and Games, pages 87–117. Amsterdam University Press, 2008.

[7] Hans van Ditmarsch, Jan van Eijck, and William Wu. One hundred prisoners and a lightbulb — logic and computation. Twelfth International Conference on the Principles of Knowledge Representation and Reasoning, Toronto, Canada, May 2010.

[8] H.P. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2006.

[9] Jan van Eijck. DEMO — a demo of epistemic modelling. In Johan van Benthem, Dov Gabbay, and Benedikt Löwe, editors, *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, number 1 in Texts in Logic and Games, pages 305–363. Amsterdam University Press, 2007.

[10] Jan van Eijck and Floor Sietsma. Multi-agent belief revision with linked plausibilities. In G. Bonanno, B. Loewe, and W. van der Hoek, editors, *Logic and the Foundations of Game and Decision Theory – LOFT 8*, volume 6006 of *Lecture Notes in Artificial Intelligence*. Springer, 2010.

[11] Jan van Eijck and Christina Unger. The epistemics of presupposition projection. In Maria Aloni, Paul Dekker, and Floris Roelofsen, editors, *Proceedings of the Sixteenth Amsterdam Colloquium, December 17–19, 2007*, pages 235–240, Amsterdam, December 2007. ILLC.

[12] Jan van Eijck and Christina Unger. *Computational Semantics with Functional Programming*. To appear with Cambridge University Press, 2010.

[13] Jan van Eijck and Yanjing Wang. Propositional Dynamic Logic as a logic of belief revision. In Wilfrid Hodges and Ruy de Queiros, editors, *Proceedings of Wollic'08*, number 5110 in Lecture Notes in Artificial Intelligence, pages 136–148. Springer, 2008. `http://dx.doi.org/10.1007/978-3-540-69937-8_13`.

[14] J. Groenendijk and M. Stokhof. *Studies on the Semantics of Questions and the Pragmatics of Answers*. PhD thesis, University of Amsterdam, 1984.

[15] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing (PODS)*, pages 50–61, 1984. A newer version appeared in the *Journal of the ACM*, vol. 37:3, 1990, pp. 549–587.

[16] D. Kozen and R. Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14:113–118, 1981.

[17] J. Moore. Implementation, contracts, and renegotiation in environments with complete information. In J.-J. Laffont, editor, *Advances in Economic Theory — 6th World Congress*, volume I, Cambridge, 1992. Cambridge University Press.

[18] Rohit Parikh. Social software. *Synthese*, 132:187–211, 2002.

[19] A. Pnueli. A temporal logic of programs. *Theoretical Computer Science*, 13:45–60, 1981.

[20] Jack Robertson and William Webb. *Cake-Cutting Algorithms*. A.K. Peters, 1998.

[21] Clifford Sahwney. *50 Wittiest Tales of Birbal*. Unicorn Books, 2004.

[22] K. Segerberg. A completeness theorem in the modal logic of programs. In T. Traczyck, editor, *Universal Algebra and Applications*, pages 36–46. Polish Science Publications, 1982.

[23] H. Steinhaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.