Dynamic epistemic logic

Jan van Eijck and Hans van Ditmarsch

ESSLLI 2008
20th European Summer School in Logic, Language and Information
4–15 August 2008
Freie und Hansestadt Hamburg, Germany

Programme Committee. Enrico Franconi (Bolzano, Italy), Petra Hendriks (Groningen, The Netherlands), Michael Kaminski (Haifa, Israel), Benedikt Löwe (Amsterdam, The Netherlands & Hamburg, Germany) Massimo Poesio (Colchester, United Kingdom), Philippe Schlenker (Los Angeles CA, United States of America), Khalil Sima'an (Amsterdam, The Netherlands), Rineke Verbrugge (**Chair**, Groningen, The Netherlands).

Organizing Committee. Stefan Bold (Bonn, Germany), Hannah König (Hamburg, Germany), Benedikt Löwe (**chair**, Amsterdam, The Netherlands & Hamburg, Germany), Sanchit Saraf (Kanpur, India), Sara Uckelman (Amsterdam, The Netherlands), Hans van Ditmarsch (**chair**, Otago, New Zealand & Toulouse, France), Peter van Ormondt (Amsterdam, The Netherlands).

`http://www.illc.uva.nl/ESSLLI2008/`
`esslli2008@science.uva.nl`

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

ESSLLI 2008 is organized by the Universität Hamburg under the auspices of the *Association for Logic, Language and Information* (FoLLI). The *Institute for Logic, Language and Computation* (ILLC) of the *Universiteit van Amsterdam* is providing important infrastructural support. Within the Universität Hamburg, ESSLLI 2008 is sponsored by the Departments *Informatik*, *Mathematik*, *Philosophie*, and *Sprache, Literatur, Medien I*, the *Fakultät für Mathematik, Informatik und Naturwissenschaften*, the *Zentrum für Sprachwissenschaft*, and the *Regionales Rechenzentrum*. ESSLLI 2008 is an event of the *Jahr der Mathematik 2008*. Further sponsors include the *Deutsche Forschungsgemeinschaft* (DFG), the Marie Curie Research Training Site GLoRiClass, the European Chapter of the Association for Computational Linguistics, the *Hamburgische Wissenschaftliche Stiftung*, the Kurt Gödel Society, Sun Microsystems, the Association for Symbolic Logic (ASL), and the European Association for Theoretical Computer Science (EATCS). The official airline of ESSLLI 2008 is Lufthansa; the book prize of the student session is sponsored by *Springer Verlag*.

Jan van Eijck and Hans van Ditmarsch

Dynamic epistemic logic

Course material. 20th European Summer School in Logic, Language and Information (ESSLLI 2008), Freie und Hansestadt Hamburg, Germany, 4–15 August 2008

Material for ESSLLI'08 DEL Course

Put together by Hans van Ditmarsch and Jan van Eijck

1. ESSLLI'08 Course slides
   (Hans van Ditmarsch)
2. Playing Cards with Hintikka
   (Hans van Ditmarsch, Wiebe van der Hoek, Barteld Kooi)
3. Logics of Communication and Change
   (Johan van Benthem, Jan van Eijck, Barteld Kooi)
4. Propositional Dynamic Logic as a Logic of Belief Revision
   (Jan van Eijck and Yanging Wang)
5. DEMO — A Demo of Epistemic Modelling
   (Jan van Eijck)

Useful links:

- Additional material (Haskell implementation): `http://www.cwi.nl/~jve/courses/esslli08`
- The Haskell homepage: `http://www.haskell.org`
- The Hugs Haskell interpreter: `http://haskell.org/hugs`
- The GHC Haskell interpreter and compiler: `http://www.haskell.org/ghc`

ESSLLI08 Hamburg: Dynamic Epistemic Logic

lecturers: Hans van Ditmarsch & Jan van Eijck

Hans' part of the course: mornings & Friday

- Monday morning: epistemic logic
- Tuesday morning: public announcements
- Wednesday morning: action models
- Thursday morning: factual change
- Friday: logic puzzles & security

hans@cs.otago.ac.nz
http://www.cs.otago.ac.nz/staffpriv/hans/

Epistemic Logic

Ia: Epistemic Logic

Epistemic Logic

Anne draws one from a stack of three different cards 0, 1, and 2.
She draws card 0. She does not look at her card yet!
Card 1 is put back into the stack holder.
Card 2 is put (face down) on the table.
Anne now looks at her card.
What does Anne know?

- Anne holds card 0.
- Anne knows that she holds card 0.
- Anne does not know that card 1 is on the table.
- Anne considers it possible that card 1 is on the table.
- Anne knows that card 1 or card 2 is in the stack holder.
- Anne knows her own card.

Language

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi$$

Descriptions of knowledge

- There is one agent Anne: $\{a\}$
- Propositional variables q_a for 'card q $(0, 1, 2)$ is held by Anne.'
- $K_a\varphi$ expresses 'Anne knows that φ'.
- $\hat{K}_a\varphi$ $(\neg K_a \neg \varphi)$ expresses 'Anne considers it possible that φ'.

- Anne holds card 0: 0_a
- Anne knows that she holds card 0: $K_a 0_a$
- Anne does not know that card 1 is on the table: $\neg K_a 1_t$
- Anne considers it possible that card 1 is not on the table: $\hat{K}_a \neg 1_t$
- Anne knows that card 1 or card 2 is in the stack holder: $K_a(1_h \vee 2_h)$
- Anne knows her own card: $K_a 0_a \vee K_a 1_a \vee K_a 2_a$

Structures

A *Kripke model* is a structure $M = \langle S, R, V \rangle$, where

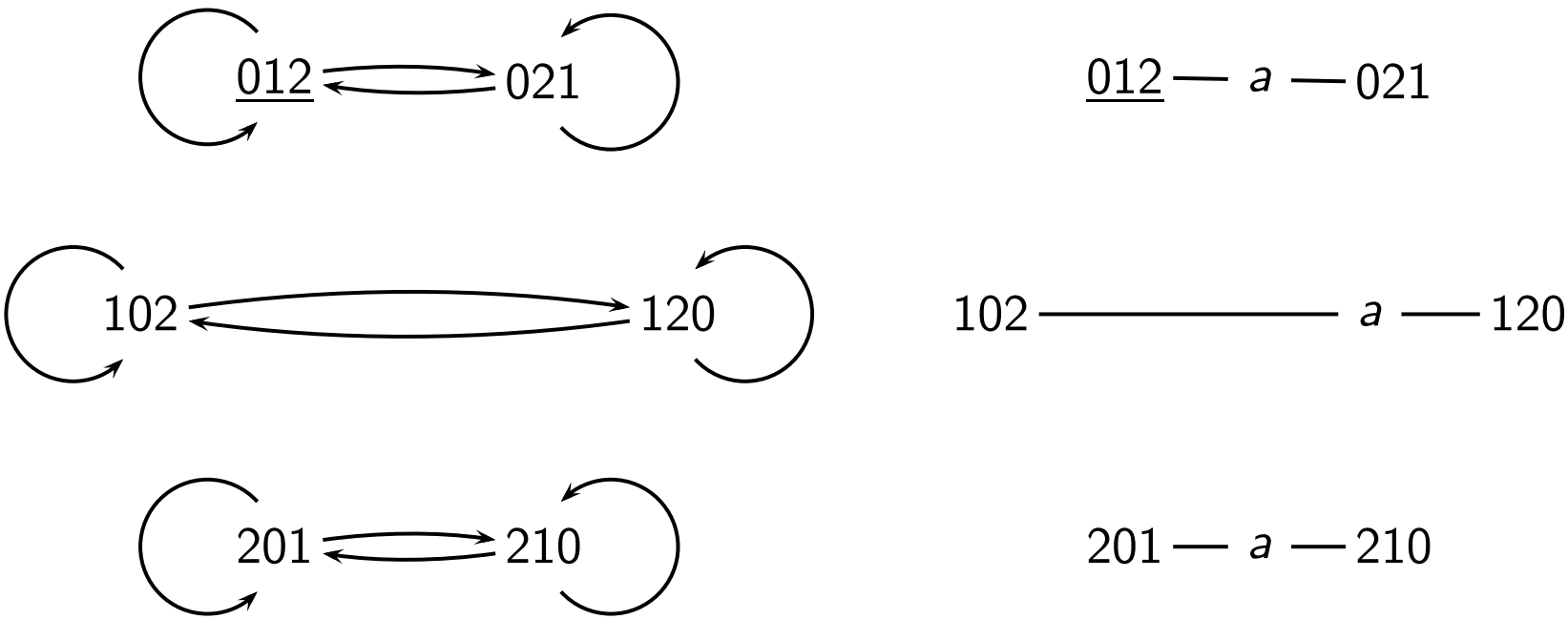
- *domain* S is a nonempty set of states;
- R yields an *accessibility relation* $R_a \subseteq S \times S$ for every $a \in A$;
- *valuation* (function) $V : P \to \mathcal{P}(S)$.

If all the relations R_a in M are equivalence relations, we call M an *epistemic model*. In that case, we write $\sim_a$ rather than R_a, and we represent the model as $M = \langle S, \sim, V \rangle$.

Epistemic state (M, s): epistemic model M with designated state s.

Example

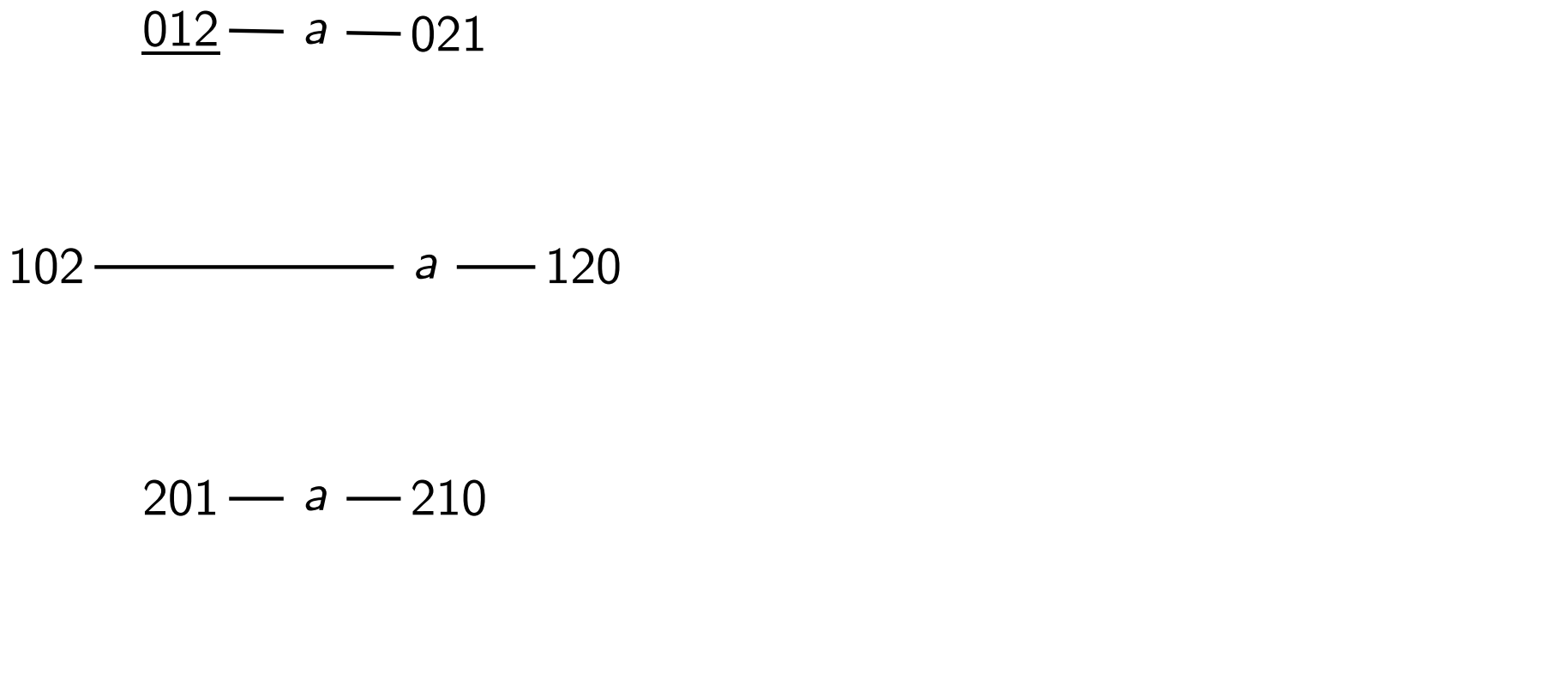
- $S = \{012, 021, 102, 120, 201, 210\}$
- $\sim_a = \{(012, 012), (012, 021), (021, 021), \dots\}$
- $V(0_a) = \{012, 021\},\ V(1_a) = \{102, 120\},\ \dots$



Truth

$$
\begin{array}{lll}
M, s \models p & \text{iff} & s \in V(p) \\
M, s \models (\varphi \wedge \psi) & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models \neg\varphi & \text{iff} & \text{not } (M, s \models \varphi) \\
M, s \models K_a\varphi & \text{iff} & \text{for all } t \text{ such that } s \sim_a t \text{ it holds that } M, t \models \varphi
\end{array}
$$

Example



$Hexa1, 012 \models K_a 0_a$

$\Leftrightarrow$

for all $t : 012 \sim_a t$ implies $Hexa1, t \models 0_a$

$\Leftarrow$

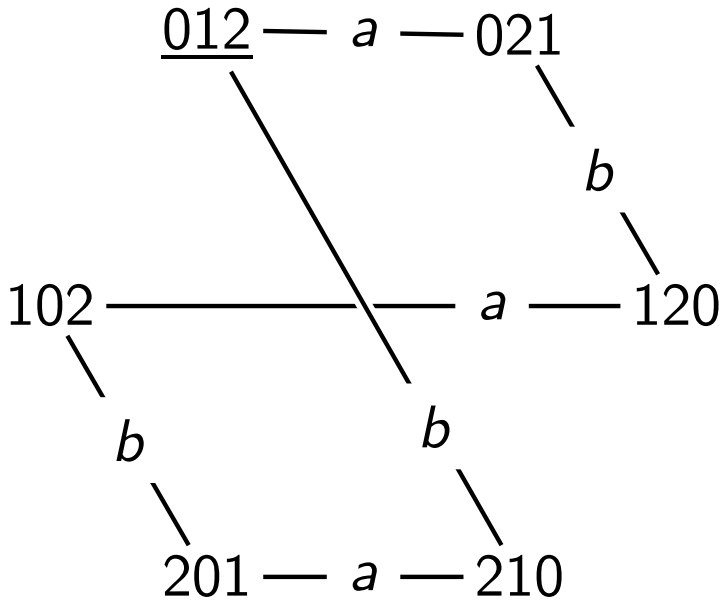
$Hexa1, 012 \models 0_a$ and $Hexa1, 021 \models 0_a$

$\Leftrightarrow$

$012 \in V(0_a) = \{012, 021\}$ and $021 \in V(0_a) = \{012, 021\}$

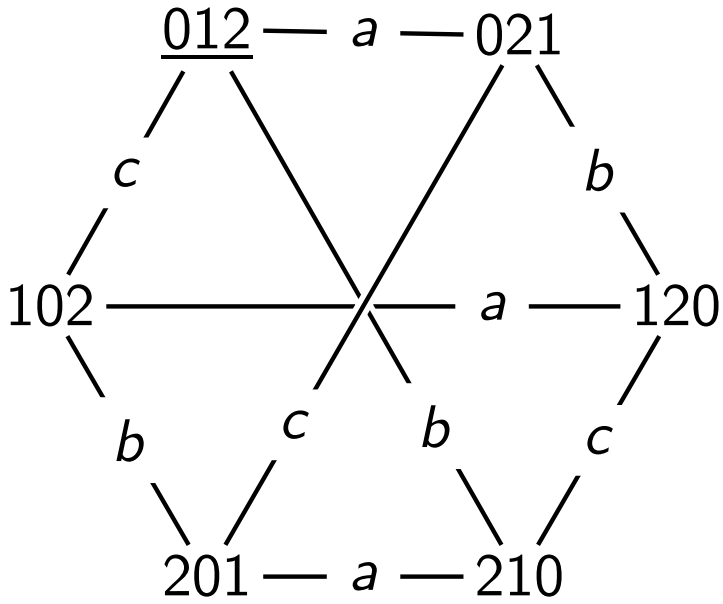
Two agents

Anne and Bill draw 0 and 1 from the cards 0, 1, 2. Card 2 is put (face down) on the table.



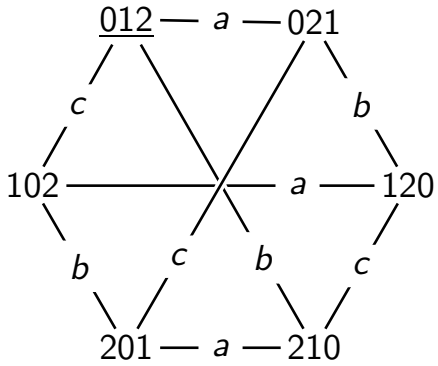
- Bill does not consider it possible that Anne has card 1: $\neg \hat{K}_b 1_a$
- Anne considers it possible that Bill considers it possible that she has card 1: $\hat{K}_a \hat{K}_b 1_a$
- Anne knows Bill to consider it possible that she has card 0: $K_a \hat{K}_b 0_a$

Three agents: Anne, Bill, Cath draw 0, 1, and 2



- Anne knows that Bill knows that Cath knows her own card: $K_a K_b(K_c 0_c \vee K_c 1_c \vee K_c 2_c)$
- Anne has card 0, but she considers it possible that Bill considers it possible that Cath knows that Anne does not have card 0: $0_a \wedge \hat{K}_a \hat{K}_b K_c \neg 0_a$

Example



$Hexa, 012 \models \hat{K}_a \hat{K}_b K_c \neg 0_a$

$\Leftarrow$

$Hexa, 021 \models \hat{K}_b K_c \neg 0_a$ because $012 \sim_a 021$

$\Leftarrow$

$Hexa, 120 \models K_c \neg 0_a$ because $021 \sim_b 120$

$\Leftrightarrow$

$Hexa, 120 \models \neg 0_a$ and $Hexa, 210 \models \neg 0_a$ $\sim_c(120) = \{120, 210\}$

$\Leftrightarrow$

$Hexa, 120 \not\models 0_a$ and $Hexa, 210 \not\models 0_a$

$\Leftrightarrow$

$120, 210 \notin V_{0_a} = \{012, 021\}$

Properties of knowledge

- $K_a\varphi \rightarrow \varphi$ — veridicality / truth axiom
- $K_a\varphi \rightarrow K_aK_a\varphi$ — positive introspection
- $\neg K_a\varphi \rightarrow K_a\neg K_a\varphi$ — negative introspection

Realistic assumptions for knowledge?

Axiomatization

all instantiations of propositional tautologies
$K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$
$K_a\varphi \to \varphi$
$K_a\varphi \to K_aK_a\varphi$
$\neg K_a\varphi \to K_a\neg K_a\varphi$
From φ and $\varphi \to \psi$, infer ψ
From φ, infer $K_a\varphi$

History

- von Wright 1951: An Essay in Modal Logic
- Hintikka 1962: Knowledge and Belief
- Aumann 1976: Agreeing to Disagree
- Fagin, Halpern, Moses and Vardi 1995: Reasoning about Knowledge
- Meyer and van der Hoek 1995: Epistemic Logic for AI and Computer Science

Common knowledge

Ib: Common knowledge

General knowledge and common knowledge

You forgot if you already passed the Channel Tunnel... When driving on a one-lane road, will you swerve to the left or to the right when other traffic approaches? How do you know that the other car knows that one is to drive on the left?

You are celebrating Sinterklaas (St. Nicholas) with family friends. How will you behave if its generally known that your 8-year old niece does not believe in Sinterklaas? And if it is common knowledge?

General knowledge and common knowledge

General knowledge:
$E_G\varphi := K_1\varphi \wedge K_2\varphi \wedge ... \wedge K_{\text{last}}\varphi$

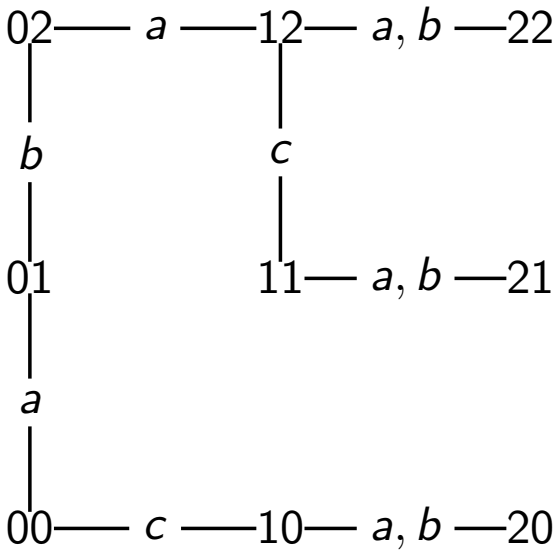
Common knowledge:
$C_G\varphi := \varphi \wedge E_G\varphi \wedge E_G E_G\varphi \wedge ...$
or
$C_G\varphi := \varphi \wedge K_1\varphi \wedge K_2\varphi \wedge K_1K_1\varphi \wedge K_1K_2\varphi \wedge \ldots K_1K_1K_1\varphi \ldots$

$C_G\varphi \leftrightarrow \varphi \wedge E_G C_G\varphi$

Computing transitive closure

$$\sim_B := (\bigcup_{a \in B} \sim_a)^*$$

R^* is the transitive and reflexive closure of a binary relation R: points s and t are R^*-related, if there is a path (of length 0 or more) of R-links between them.



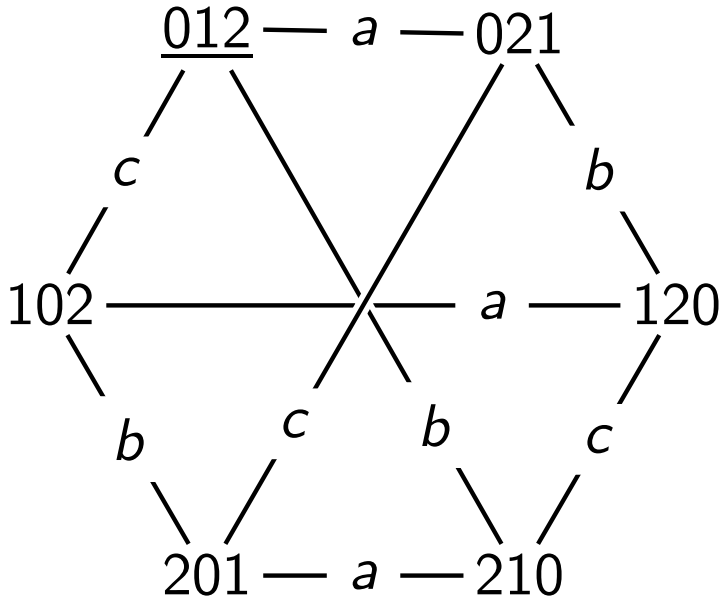
What is the partition on these nine states for a?
For group $\{a, b\}$? For group $\{a, c\}$? For group $\{a, b, c\}$?

Epistemic Logic with Common Knowledge

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid C_B\varphi$$

$$M, s \models C_B\varphi \quad \text{iff} \quad \text{for all } t : s \sim_B t \ \text{ implies } \ M, t \models \varphi$$

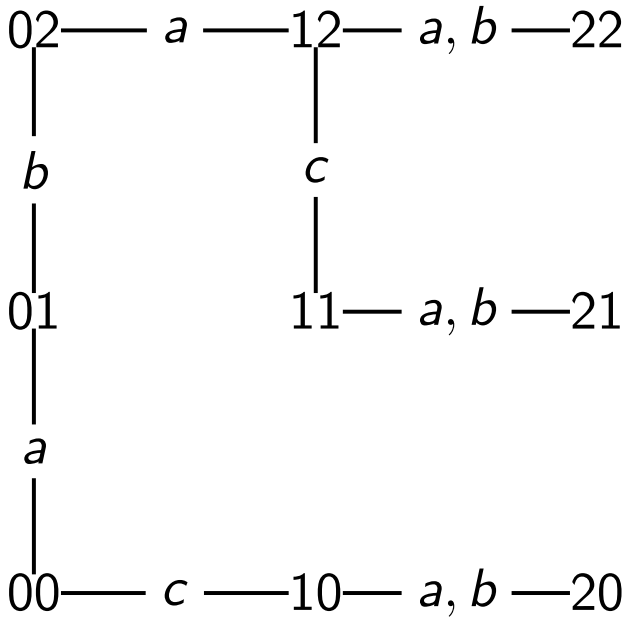
Example



$Hexa, 012 \models C_{abc}(K_a 0_a \vee K_a 1_a \vee K_a 2_a)$
(it is public knowledge that Anne knows her card)

$Hexa \models C_{ab}\varphi \rightarrow C_{bc}\varphi$
(a and b share the same knowledge as b and c)

Example



Which of the following are true / false:

$$11 \models K_c(x = 1)$$
$$11 \models C_{ac}(y \neq 0)$$
$$10 \models C_{ab}(x \geq 1)$$
$$02 \models C_{ab}((y = 2) \rightarrow C_{cb}(x > 0))$$

Axiomatization

$C_B(\varphi \to \psi) \to (C_B\varphi \to C_B\psi)$
$C_B\varphi \to (\varphi \wedge E_B C_B\varphi)$
$C_B(\varphi \to E_B\varphi) \to (\varphi \to C_B\varphi)$
From φ, infer $C_B\varphi$

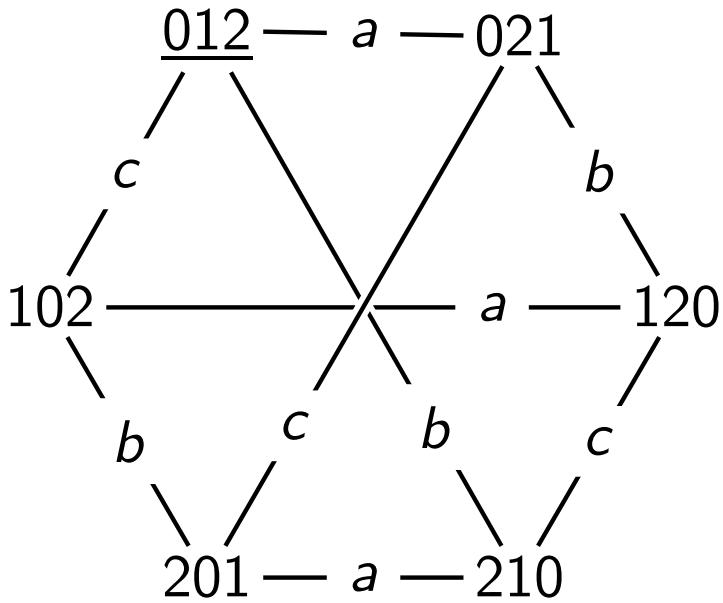
History

- Lewis 1969: Convention
- Friedell 1969: On the structure of shared awareness
- Aumann 1976: Agreeing to disagree
- Barwise 1988: Three views of common knowledge

Public announcements

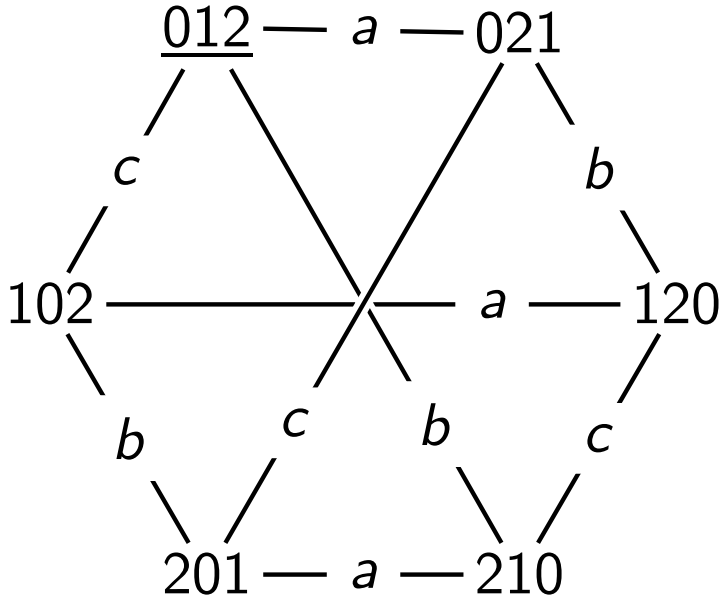
II: Public announcements

Example



- After Anne says that she does not have card 1, Cath knows that Bill has card 1.
- After Anne says that she does not have card 1, Cath knows Anne's card.
- Bill still doesn't know Anne's card after that.

Example



- After Anne says that she does not have card 1, Cath knows that Bill has card 1.
  $[\neg 1_a]K_c 1_b$
- After Anne says that she does not have card 1, Cath knows Anne's card.
  $[\neg 1_a](K_c 0_a \vee K_c 1_a \vee K_c 2_a)$
- Bill still doesn't know Anne's card after that:
  $[\neg 1_a]\neg(K_b 0_a \vee K_b 1_a \vee K_b 2_a)$

Public Announcements: language

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid C_B\varphi \mid [\varphi]\varphi$$

Public Announcements: semantics

The effect of the public announcement of φ is the restriction of the epistemic state to all states where φ holds. So, 'announce φ' can be seen as an epistemic state transformer, with a corresponding dynamic modal operator $[\varphi]$.

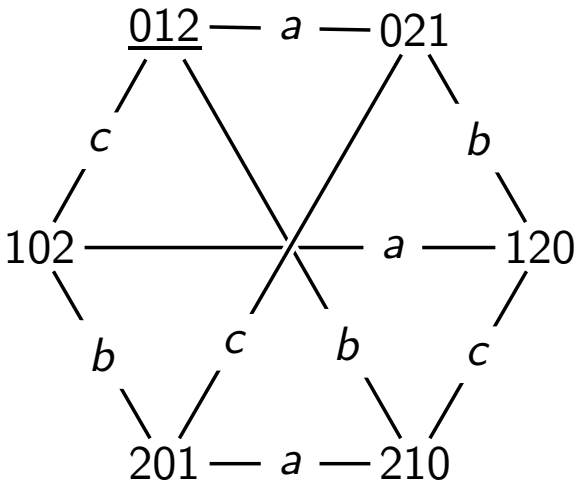
'φ is the announcement' means 'φ is publicly and truthfully announced'.

$$M, s \models [\varphi]\psi \text{ iff } (M, s \models \varphi \text{ implies } M|\varphi, s \models \psi)$$

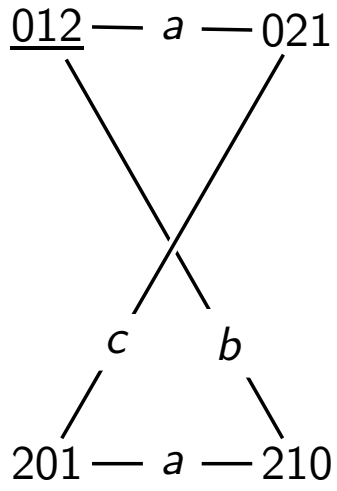
$M|\varphi := \langle S', \sim', V' \rangle$:

$$\begin{array}{lcl} S' & := & [\![\varphi]\!]_M \\ \sim'_a & := & \sim_a \cap ([\![\varphi]\!]_M \times [\![\varphi]\!]_M) \\ V'(p) & := & V(p) \cap [\![\varphi]\!]_M \end{array}$$

Example announcement in Hexa



$\Rightarrow$



$Hexa, 012 \models \langle \neg 1_a \rangle K_c 0_a$

$\Leftrightarrow$

$Hexa, 012 \models \neg 1_a$ and $Hexa|\neg 1_a, 012 \models K_c 0_a$

$\Leftarrow$

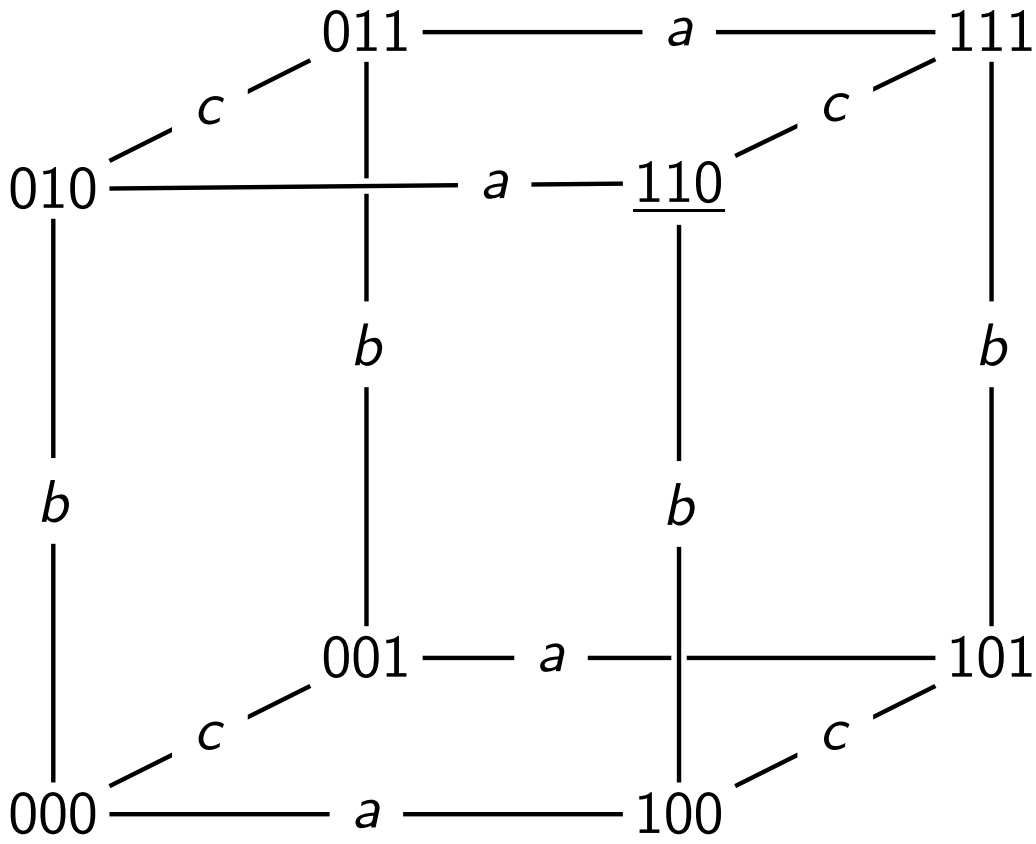
$012 \neq V(1_a)$ and $Hexa|\neg 1_a, 012 \models 0_a$

$\sim_c (012) = \{012\}$

Muddy Children

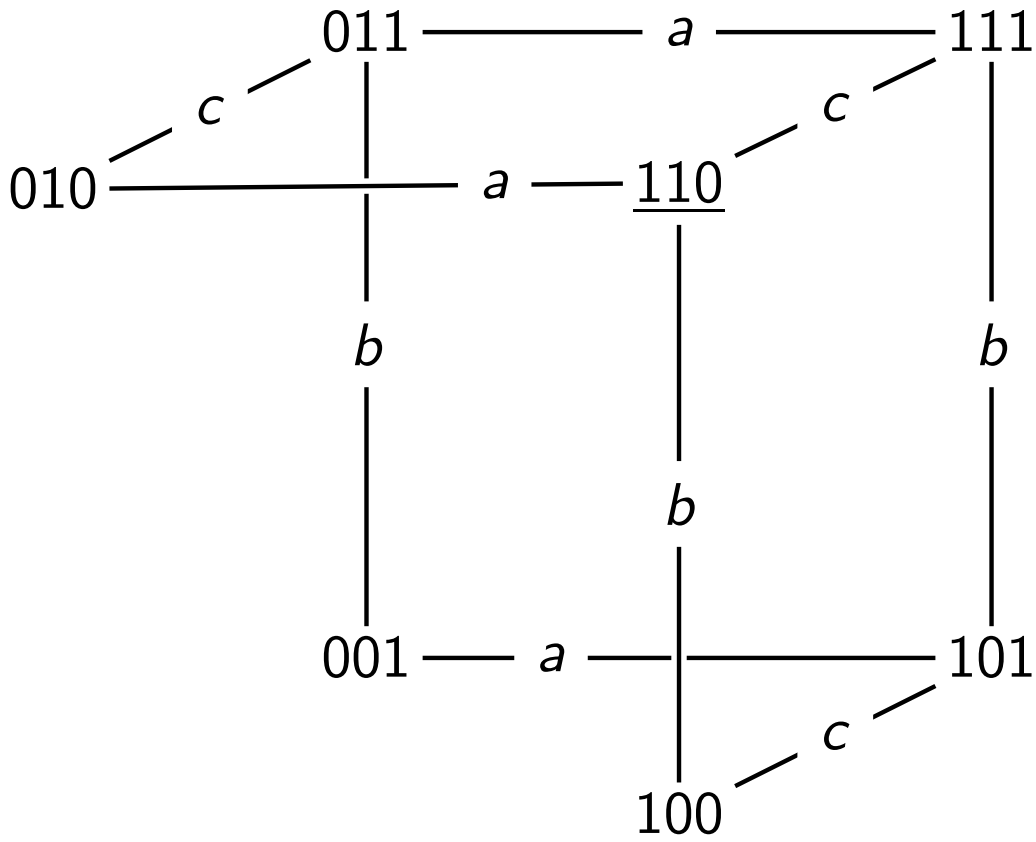
A group of children has been playing outside and are called back into the house by their father. The children gather round him. As one may imagine, some of them have become dirty from the play and in particular: they may have mud on their forehead. Children can only see whether other children are muddy, and not if there is any mud on their own forehead. All this is commonly known, and the children are, obviously, perfect logicians. Father now says: "At least one of you has mud on his or her forehead." And then: "Will those who know whether they are muddy please step forward." If nobody steps forward, father keeps repeating the request. What happens?

Muddy Children



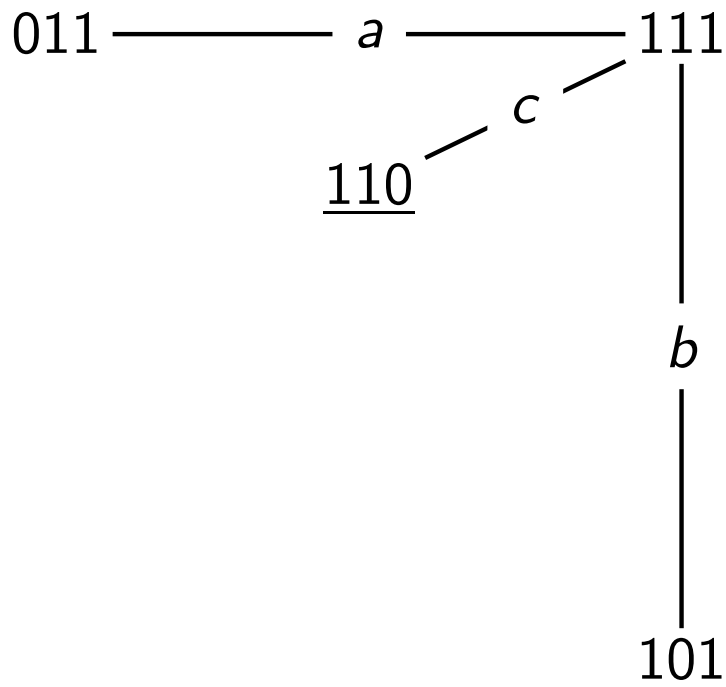
Given: The children can see each other

Muddy Children



After: At least one of you has mud on his or her forehead.

Muddy Children



After: Will those who know whether they are muddy please step forward?

Muddy Children

$\underline{110}$

After: Will those who know whether they are muddy please step forward?

Axiomatization

$[\varphi]p \leftrightarrow (\varphi \rightarrow p)$
$[\varphi]\neg\psi \leftrightarrow (\varphi \rightarrow \neg[\varphi]\psi)$
$[\varphi](\psi \wedge \chi) \leftrightarrow ([\varphi]\psi \wedge [\varphi]\chi)$
$[\varphi]K_a\psi \leftrightarrow (\varphi \rightarrow K_a[\varphi]\psi)$
$[\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$
From φ, infer $[\psi]\varphi$
From $\chi \rightarrow [\varphi]\psi$ and $\chi \wedge \varphi \rightarrow E_B\chi$, infer $\chi \rightarrow [\varphi]C_B\psi$

Every formula in the language of public announcement logic without common knowledge is equivalent to a formula in the language of epistemic logic.

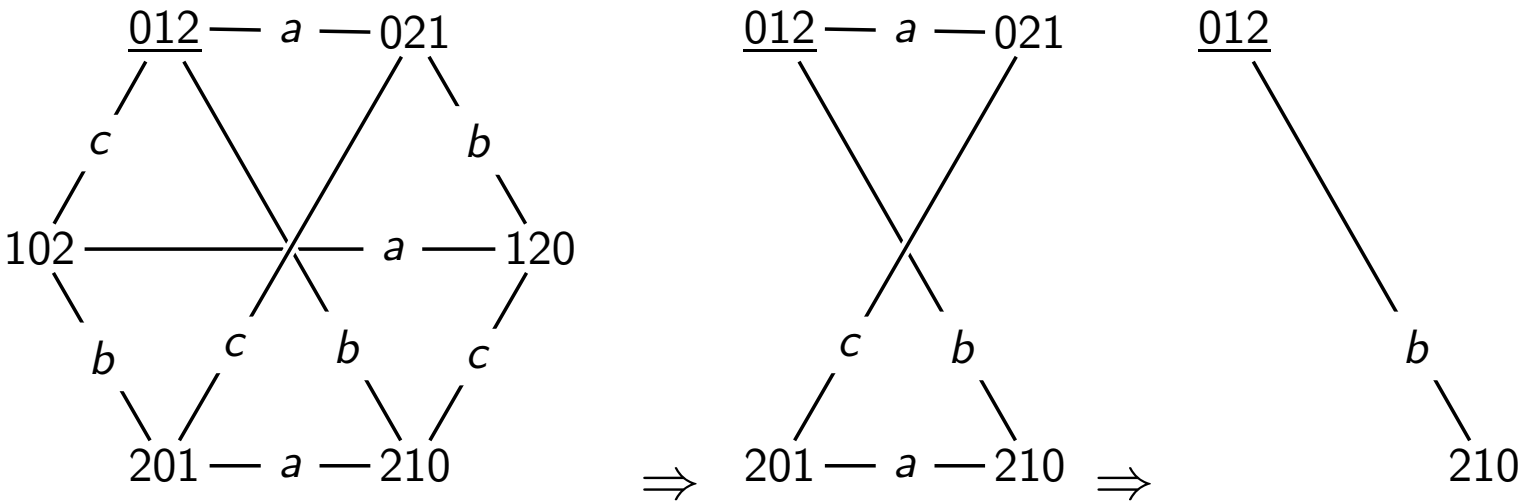
Sequence of announcements

Anne does not have card 1, and Cath now knows Anne's card.
Sequence of two announcements:

$$\neg 1_a \; ; \; (K_c 0_a \vee K_c 1_a \vee K_c 2_a)$$

Single announcement:

$$\neg 1_a \wedge [\neg 1_a](K_c 0_a \vee K_c 1_a \vee K_c 2_a)$$



Unsuccessful updates

Postulate of success:

$$\varphi \to \langle \varphi \rangle C_A \varphi$$

Announcement of a *fact* always makes it public:

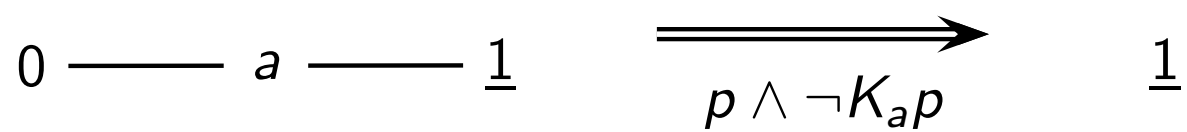
$$\models [p] C_A p$$

Announcements of non-facts do not have to make them public:

$$\not\models [\varphi] C_A \varphi$$

It can be even worse:

$$\models [p \wedge \neg K_b p] \neg (p \wedge \neg K_b p)$$



History

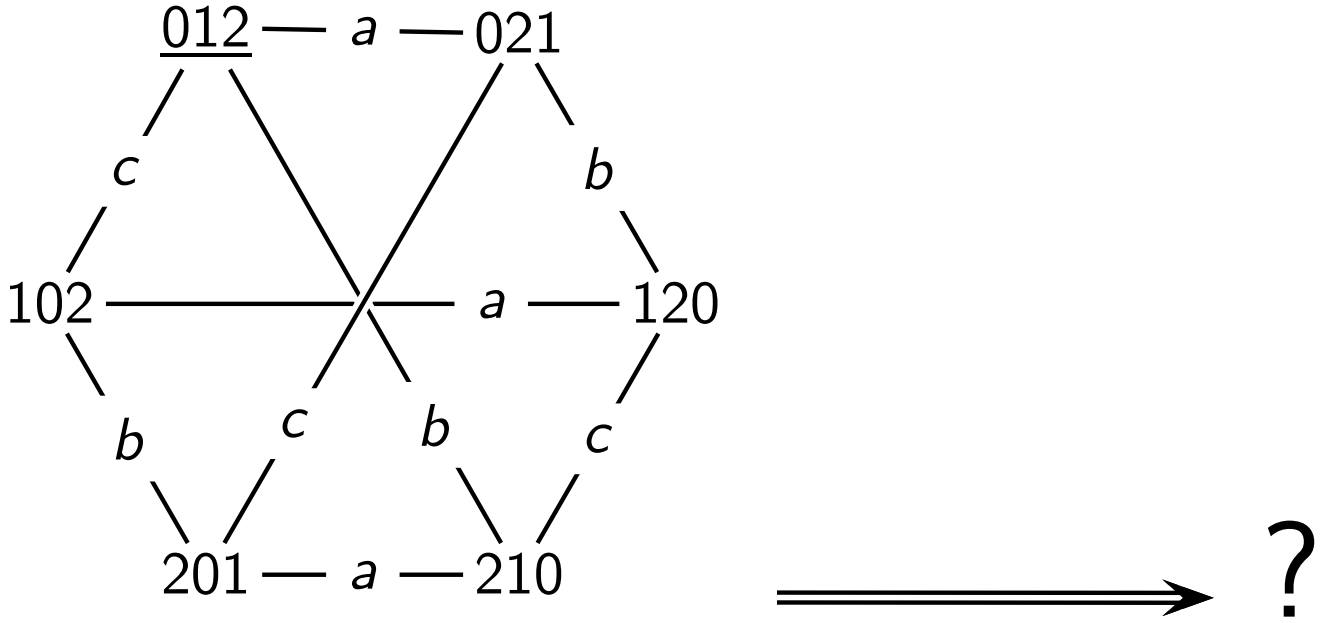
- Plaza 1989: Logics of Public Communications
- Gerbrandy & Groeneveld 1997: Reasoning about Information Change
- Baltag, Moss & Solecki 1998: The Logic of Common Knowledge, Public Announcements, and Private Suspicions
- van Ditmarsch, van der Hoek & Kooi 2007: Dynamic Epistemic Logic

Action models

III: Action models

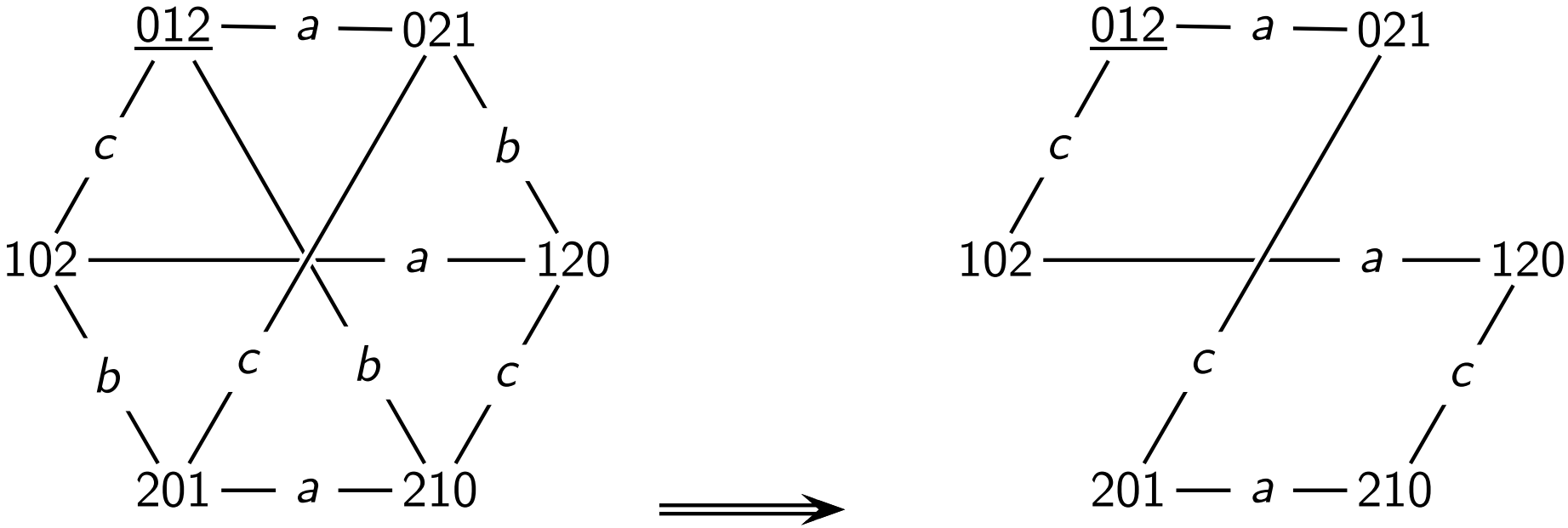
What we cannot do yet...

(Anne holds 0, Bill holds 1, and Cath holds 2.) Anne shows (only) Bill card 0. Cath cannot see the face of the shown card, but notices that a card is being shown.



What we cannot do yet...

(Anne holds 0, Bill holds 1, and Cath holds 2.) Anne shows (only) Bill card 0. Cath cannot see the face of the shown card, but notices that a card is being shown.



Epistemic modeling

- Given is an informal description of a situation
- The modeler tries to determine:
  - The set of relevant propositions
  - The set of relevant agents
  - The set of states
  - An indistinguishability relation over these worlds for each agent

Dynamic modeling

- Given is an informal description of a situation and an event that takes place in that situation.
- The modeler first models the epistemic situation, and then tries to determine:
  - The set of possible events
  - The preconditions for the events
  - An indistinguishability relation over these events for each agent

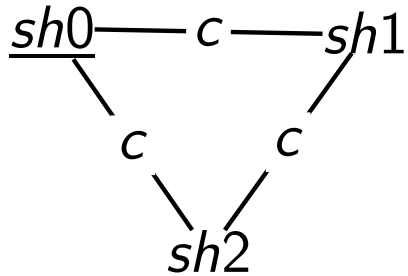
Action models

An action model M is a structure $\langle S, \sim, pre \rangle$

- S is a *finite* domain of action points or events
- $\sim_a$ is an equivalence relation on S
- pre : S $\longrightarrow \mathcal{L}$ is a preconditions function that assigns a precondition to each s $\in$ S.

Showing a card

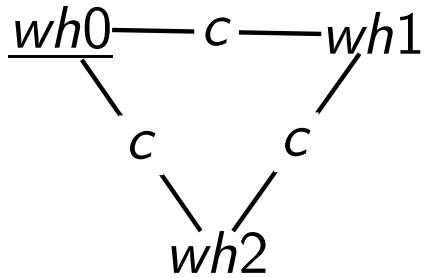
(Anne holds 0, Bill holds 1, and Cath holds 2.) Anne shows (only) Bill card 0. Cath cannot see the face of the shown card, but notices that a card is being shown.



- $\mathsf{S} = \{\mathsf{sh0}, \mathsf{sh1}, \mathsf{sh2}\}$
- $\sim_1 = \{(\mathsf{s}, \mathsf{s}) \mid \mathsf{s} \in \mathsf{S}\}$
- $\sim_2 = \{(\mathsf{s}, \mathsf{s}) \mid \mathsf{s} \in \mathsf{S}\}$
- $\sim_3 = \mathsf{S} \times \mathsf{S}$
- $\mathsf{pre}(\mathsf{sh0}) = 0_a$
- $\mathsf{pre}(\mathsf{sh1}) = 1_a$
- $\mathsf{pre}(\mathsf{sh2}) = 2_a$

Whispering

Bill asks Anne to tell him a card that he (Bill) doesn't have. Anne whispers in Bill's ear "I don't have card 2". Cath notices that the question is answered, but cannot hear the answer.



- $\mathsf{S} = \{\mathsf{wh0}, \mathsf{wh1}, \mathsf{wh2}\}$
- $\sim_1 = \{(\mathsf{s}, \mathsf{s}) \mid \mathsf{s} \in \mathsf{S}\}$
- $\sim_2 = \{(\mathsf{s}, \mathsf{s}) \mid \mathsf{s} \in \mathsf{S}\}$
- $\sim_3 = \mathsf{S} \times \mathsf{S}$
- $\mathsf{pre}(\mathsf{sh0}) = \neg 0_a$
- $\mathsf{pre}(\mathsf{sh1}) = \neg 1_a$
- $\mathsf{pre}(\mathsf{sh2}) = \neg 2_a$

What do you learn from an action?

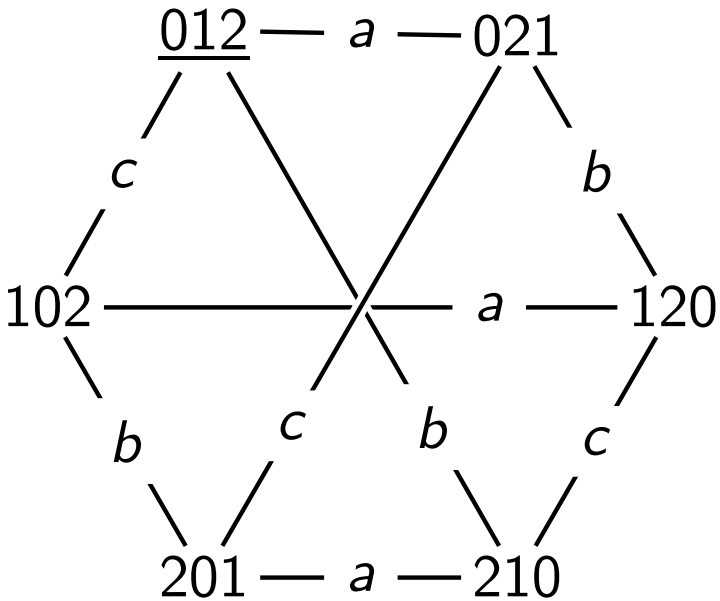
- Firstly, if you can distinguish two actions, then you can also distinguish the states that result from executing the action.
- Secondly, you do not forget anything due to an action. States that you could distinguish before an action are still distinguishable.

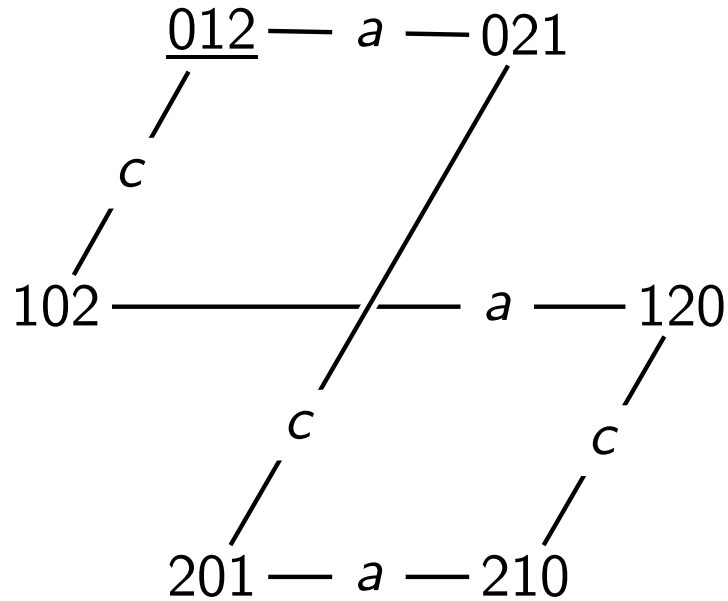
Product update

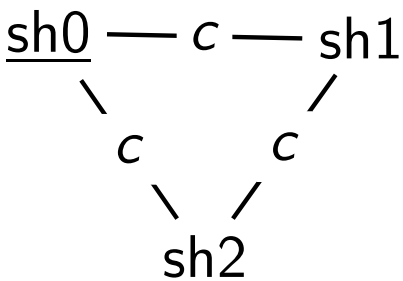
Given are an epistemic state (M, s) with $M = \langle S, \sim, V \rangle$ and an action model (M, s) with $\mathsf{M} = \langle \mathsf{S}, \sim, \mathsf{pre} \rangle$. The result of executing (M, s) in (M, s) is $(M', (s, \mathsf{s}))$ where $M' = \langle S', \sim', V' \rangle$ such that:

- $S' = \{(s, \mathsf{s}) \mid s \in S, \mathsf{s} \in \mathsf{S}, \text{ and } M, s \models \mathsf{pre}(\mathsf{s})\}$
- $(s, \mathsf{s}) \sim'_a (t, \mathsf{t})$ iff $(s \sim_a t$ and $\mathsf{s} \sim_a \mathsf{t})$
- $(s, \mathsf{s}) \in V'_p$ iff $s \in V_p$

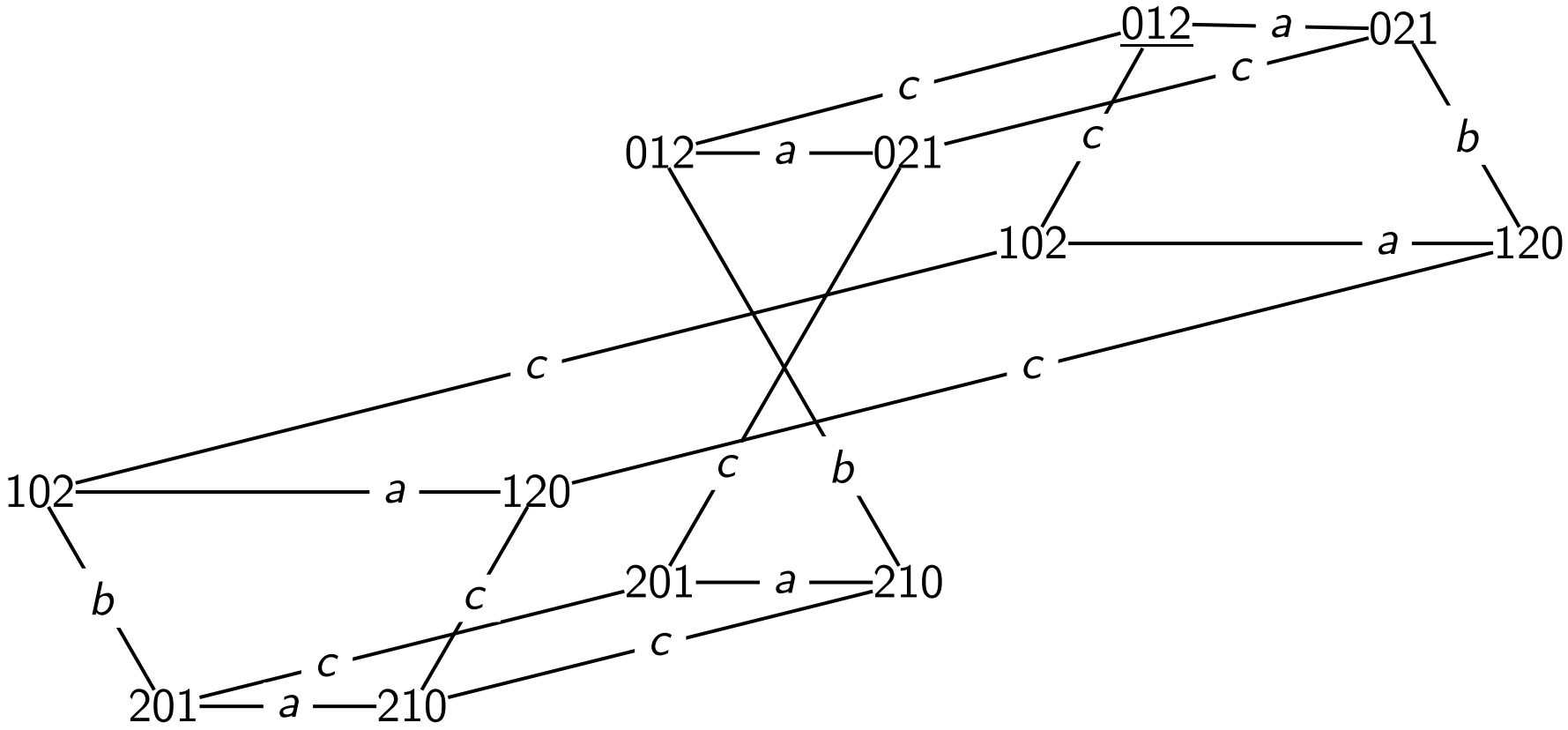
Anne shows card 0 to Bill







Anne whispers ‘not 0’ to Bill



Language

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid C_B\varphi \mid [\mathrm{M}, \mathrm{s}]\varphi$$

Semantics

$$
\begin{array}{lll}
M, s \models p & \text{:iff} & s \in V_p \\
M, s \models \neg\varphi & \text{:iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \text{:iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_a\varphi & \text{:iff} & \text{for all } s' \in S : s \sim_a s' \text{ implies } M, s' \models \varphi \\
M, s \models C_B\varphi & \text{:iff} & \text{for all } s' \in S : s \sim_B s' \text{ implies } M, s' \models \varphi \\
M, s \models [\mathrm{M}, \mathrm{s}]\varphi & \text{:iff} & \text{if } M, s \models \mathrm{pre(s)}, \text{ then } M \otimes \mathrm{M}, (s, \mathrm{s}) \models \varphi
\end{array}
$$

Syntax and semantics

- Are syntax and semantics clearly separated?

YES

Axiomatization

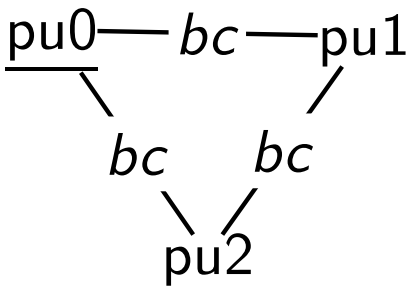
$[M, s]p \leftrightarrow (\mathrm{pre}(s) \rightarrow p)$
$[M, s]\neg\varphi \leftrightarrow (\mathrm{pre}(s) \rightarrow \neg[M, s]\varphi)$
$[M, s](\varphi \wedge \psi) \leftrightarrow ([M, s]\varphi \wedge [M, s]\psi)$
$[M, s]K_a\varphi \leftrightarrow (\mathrm{pre}(s) \rightarrow \bigwedge_{s \sim_a t} K_a[M, t]\varphi)$
$[M, s][M', s']\varphi \leftrightarrow [(M, s); (M', s')]\varphi$
From φ, infer $[M, s]\varphi$
Let (M, s) be an action model and let a set of formulas χ_t for every t such that $s \sim_B t$ be given. From $\chi_t \rightarrow [M, t]\varphi$ and $(\chi_t \wedge \mathrm{pre}(t)) \rightarrow K_a\chi_u$ for every $t \in S$ such that $s \sim_B t$, $a \in B$ and $t \sim_a u$, infer $\chi_s \rightarrow [M, s]C_B\varphi$.

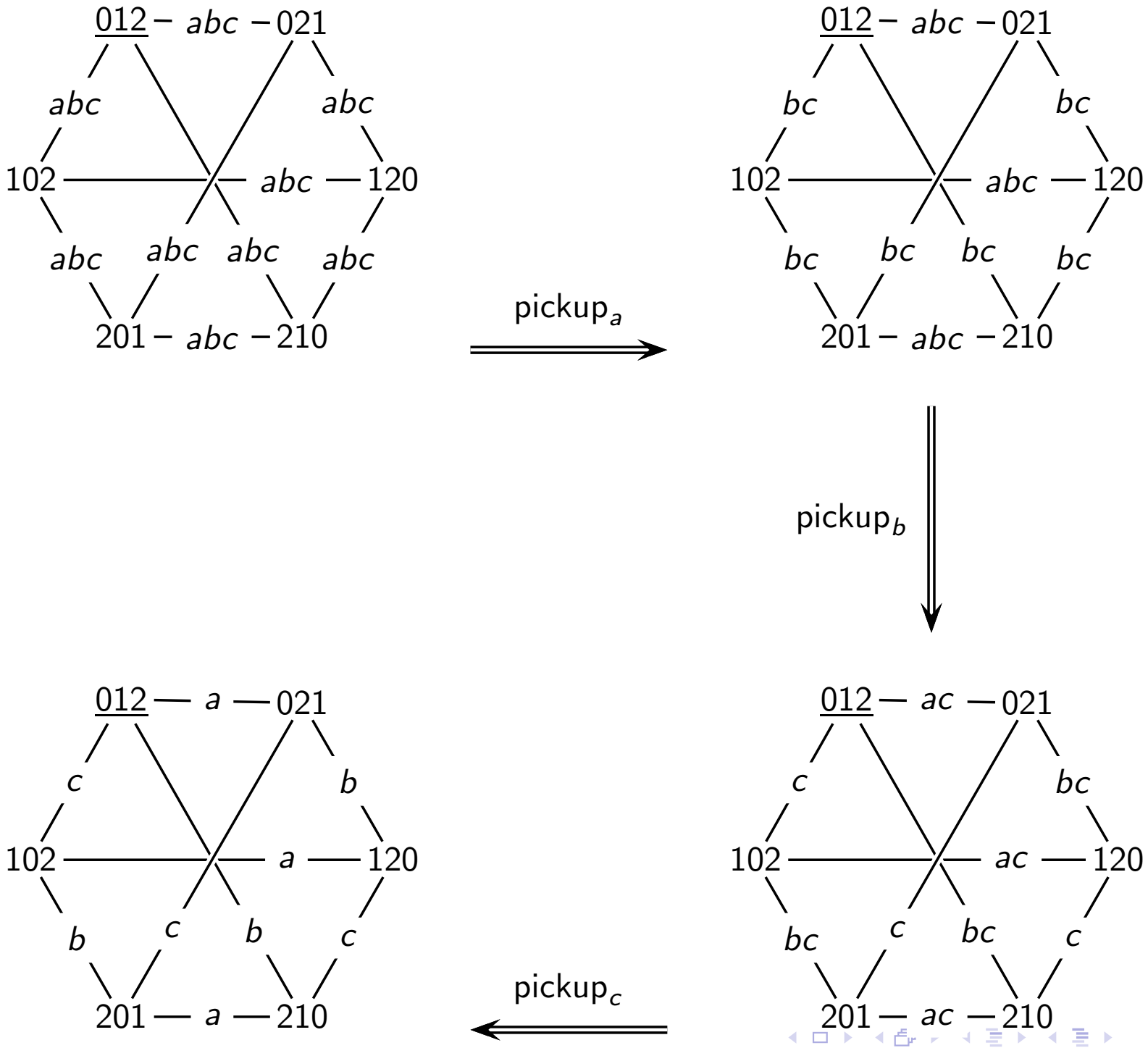
Every formula in the language of action model logic without common knowledge is equivalent to a formula in the language of epistemic logic.

Closing example: picking up cards

Three players Anne, Bill, Cath are each dealt one of cards $0, 1, 2$.

- pickup$_a$: Anne picks up her card and looks at it. It is card 0.
- pickup$_b$: Bill picks up his card and looks at it. It is card 1.
- pickup$_c$: Cath picks up her card and looks at it. It is card 2.




012 – abc – 021
abc
abc
102 – abc – 120
abc
abc
abc
abc
201 – abc – 210
pickup$_a$
012 – abc – 021
bc
bc
102 – abc – 120
bc
bc
bc
bc
201 – abc – 210
pickup$_b$
012 — ac — 021
c
bc
102 — ac — 120
bc
c
bc
c
201 — ac — 210
pickup$_c$
012 — a — 021
c
b
102 — a — 120
b
c
b
c
201 — a — 210

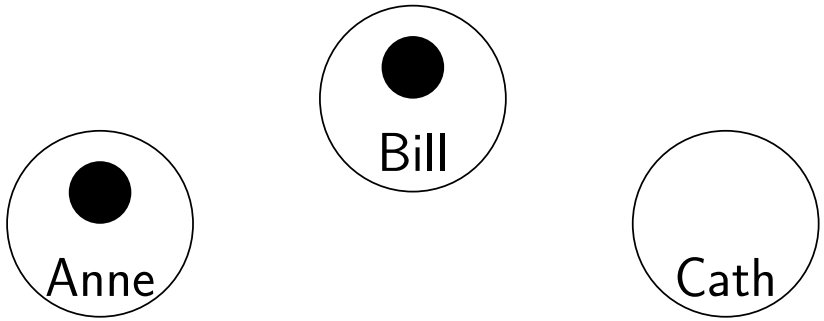
History

- Baltag, Moss & Solecki 1998: The Logic of Common Knowledge, Public Announcements, and Private Suspicions

Factual change

IV: Factual change

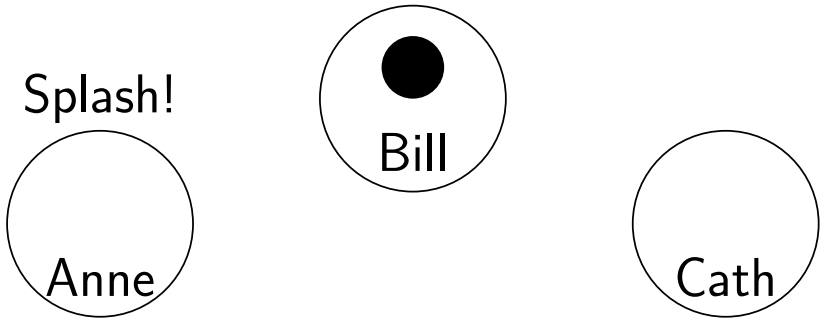
Factual change — Muddy Children again



There are three children, Anne, Bill, and Cath. Anne and Bill have mud on their foreheads. Father announces:

- At least one of you is muddy.
- If you know whether you are muddy, step forward. (Nobody steps forward.)
- If you know whether you are muddy, step forward. (Anne and Bill step forward.)

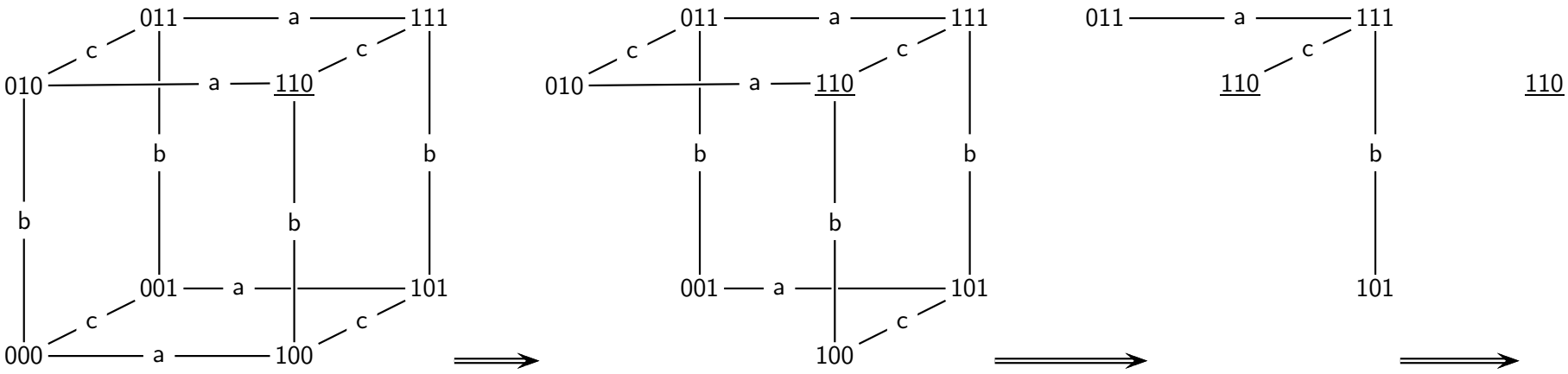
Cleaning Muddy Children



There are three children, Anne, Bill, and Cath. Anne and Bill have mud on their foreheads. Father announces:

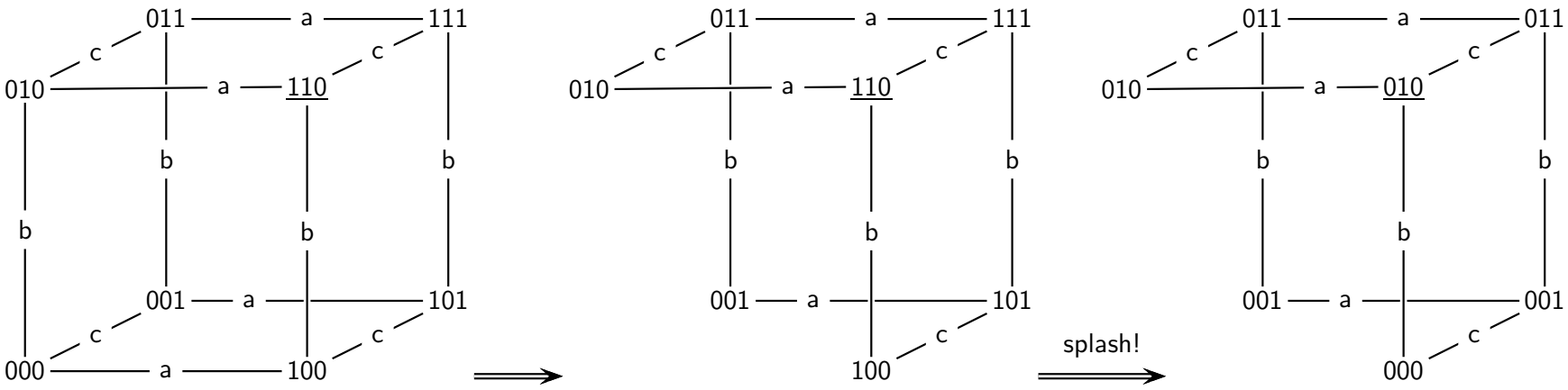
- At least one of you is muddy.
- **Splash!** *Father empties a bucket of water over Anne.*
- If you know whether you are muddy, step forward. (...?)
- If you know whether you are muddy, step forward. (...?)

Standard: Anne and Bill are muddy



- At least one child is muddy.
- Nobody steps forward.
- Anne and Bill step forward.

Non-standard: Anne and Bill are muddy, Anne is cleaned



- At least one child is muddy.
- *Father empties a bucket of water over Anne* (splash!)
- If you know whether you are muddy, step forward. (...?)
- If you know whether you are muddy, step forward. (...?)

Public factual change

Language

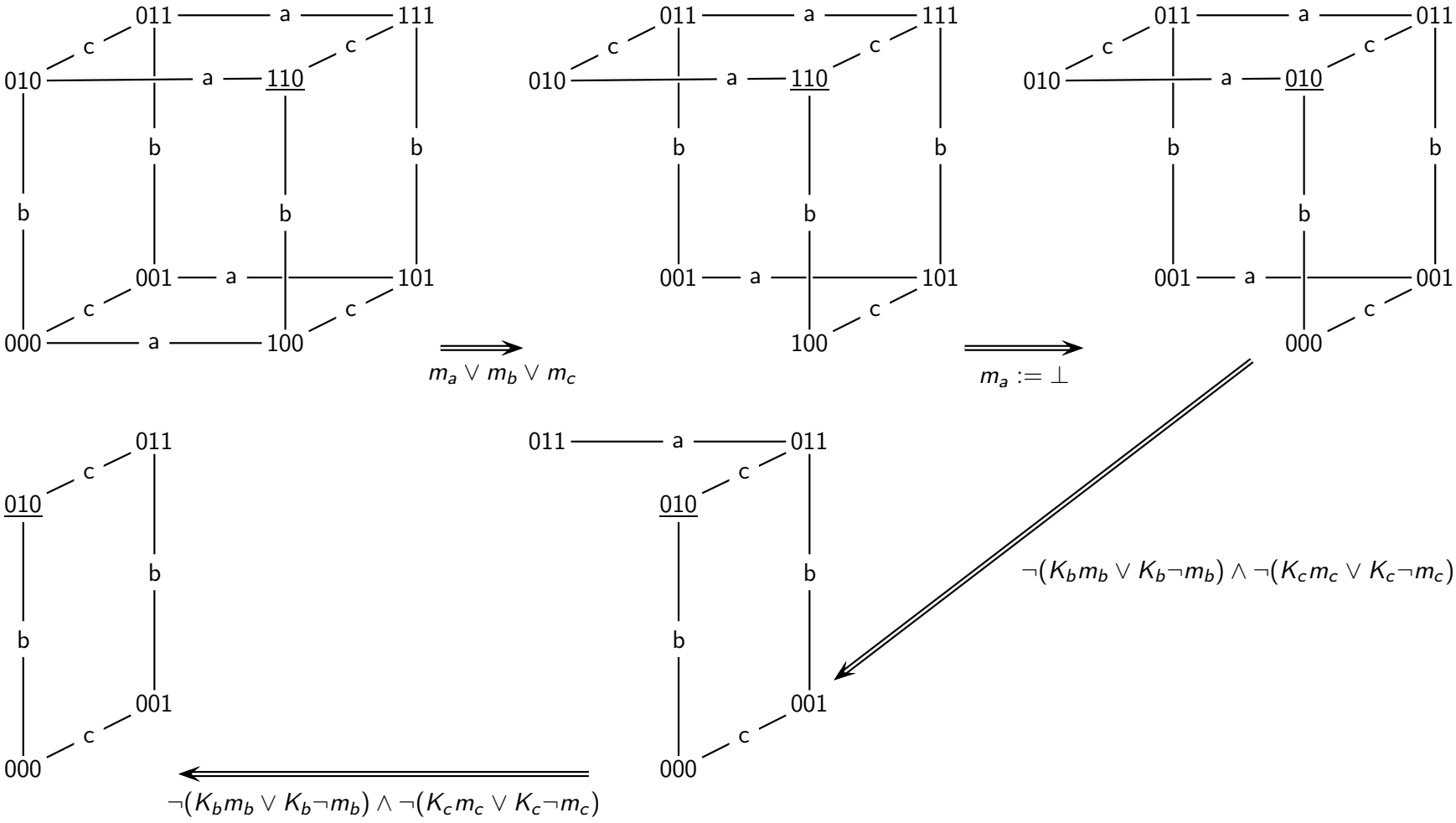
$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_a\varphi \mid C_A\varphi \mid [\varphi]\psi \mid [p := \varphi]\psi$$

Semantics

$$M, s \models [p := \varphi]\psi \quad \text{iff} \quad M_{p:=\varphi}, s \models \psi$$

$M_{p:=\varphi}$ is as M except that $V(p) = [\![\varphi]\!]_M$.

reduction principle: $[p := \varphi]p \leftrightarrow \varphi$.



At father's second request, Cath learns that Anne knows that she **was** *initially clean*

Factual change

Factual change with action models, more technique, and history: Jan

Logic puzzles

V: Logic puzzles and security protocols

- Russian Cards
- One hundred prisoners and a lightbulb

Public communication of secrets: Russian Cards

From a pack of seven known cards $0, 1, 2, 3, 4, 5, 6$ Alice (a) and Bob (b) each draw three cards and Eve (c) gets the remaining card. How can Alice and Bob openly (publicly) inform each other about their cards, without Eve learning of any of their cards who holds it?

Suppose Alice draws $\{0, 1, 2\}$, Bob draws $\{3, 4, 5\}$, and Eve 6.

Public communication of secrets: Russian Cards

From a pack of seven known cards $0, 1, 2, 3, 4, 5, 6$ Alice (a) and Bob (b) each draw three cards and Eve (c) gets the remaining card. How can Alice and Bob openly (publicly) inform each other about their cards, without Eve learning of any of their cards who holds it?

Suppose Alice draws $\{0, 1, 2\}$, Bob draws $\{3, 4, 5\}$, and Eve 6.

Bad:
Alice says "I have 012, or Bob has 012," and
Bob then says "I have 345, or Alice has 345."
Good:
Alice says "I have one of 012, 034, 056, 135, 246," and
Bob then says "Eve has card 6."

Card deals

Structures (interpreted system, Kripke model, state transition s.)

Players only know their own cards.
A hand of cards is a local state.
A deal of cards is a global state.

Logic (public announcement logic)

q_a		agent a holds card q.
ijk_a	$(i_a \wedge j_a \wedge k_a)$	agent a's hand of cards is $\{i, j, k\}$.

Epistemic postconditions

Bob informs Alice	aknowsbs	$\bigwedge(ijk_b \rightarrow K_a ijk_b)$
Alice informs Bob	bknowsas	$\bigwedge(ijk_a \rightarrow K_b ijk_a)$
Eve remains ignorant	cignorant	$\bigwedge(\neg K_c q_a \wedge \neg K_c q_b)$

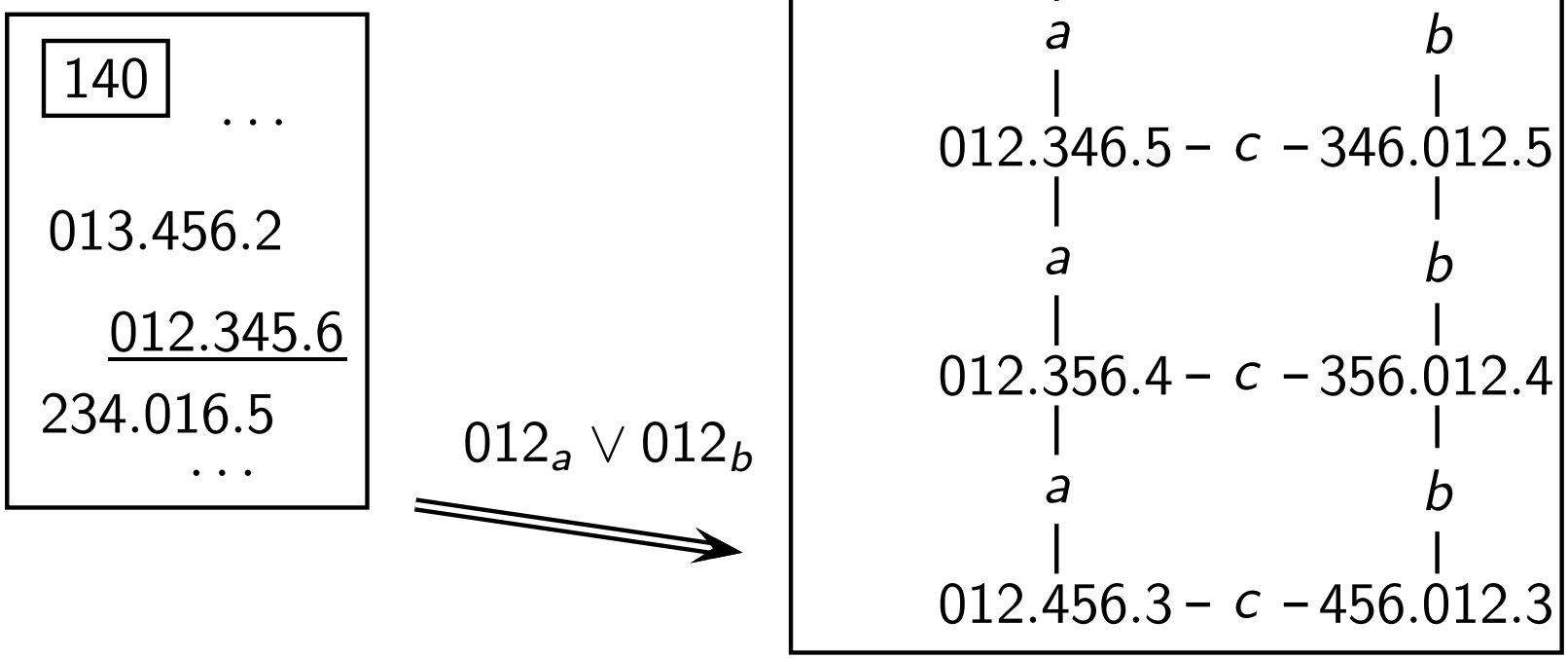
Public communication of secrets: bad

An insider says "Alice has $\{0,1,2\}$ *or Bob has* $\{0,1,2\}$*."*

$$012.345.6 \models [012_a \vee 012_b]\text{cignorant}$$

Alice says "I have $\{0,1,2\}$ *or Bob has* $\{0,1,2\}$*."*

$$012.345.6 \not\models [K_a(012_a \vee 012_b)]\text{cignorant}$$



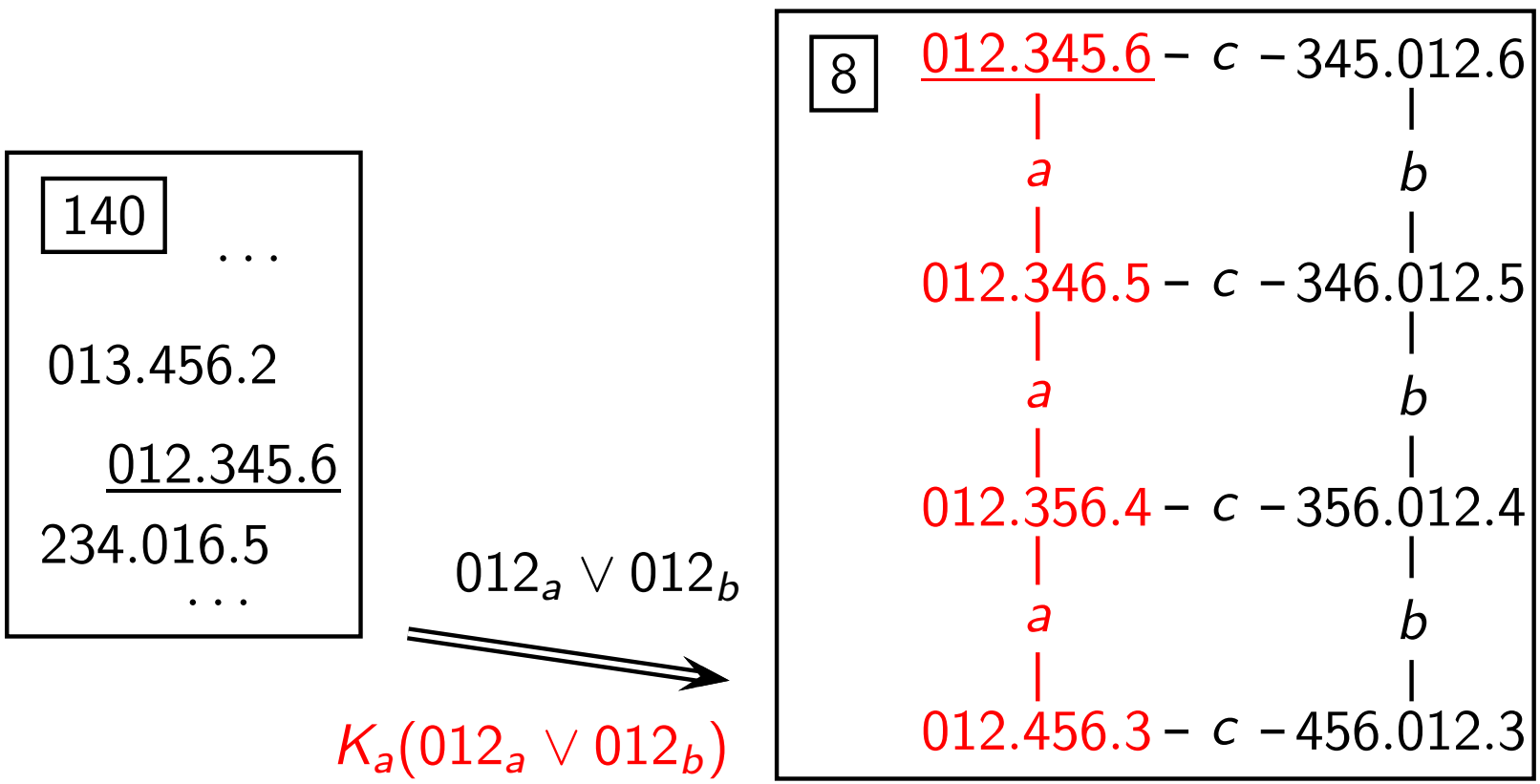
Public communication of secrets: bad

An insider says "Alice has $\{0,1,2\}$ *or Bob has* $\{0,1,2\}$*."*

$$012.345.6 \models [012_a \vee 012_b]\text{cignorant}$$

Alice says "I have $\{0,1,2\}$ *or Bob has* $\{0,1,2\}$*."*

$$012.345.6 \not\models [K_a(012_a \vee 012_b)]\text{cignorant}$$



Public communication of secrets: also bad

Alice says "I don't have card 6."

$$012.345.6 \models [K_a \neg 6_a]\text{cignorant}$$
$$012.345.6 \not\models [K_a \neg 6_a] K_a\text{cignorant}$$

Public communication of secrets: almost good

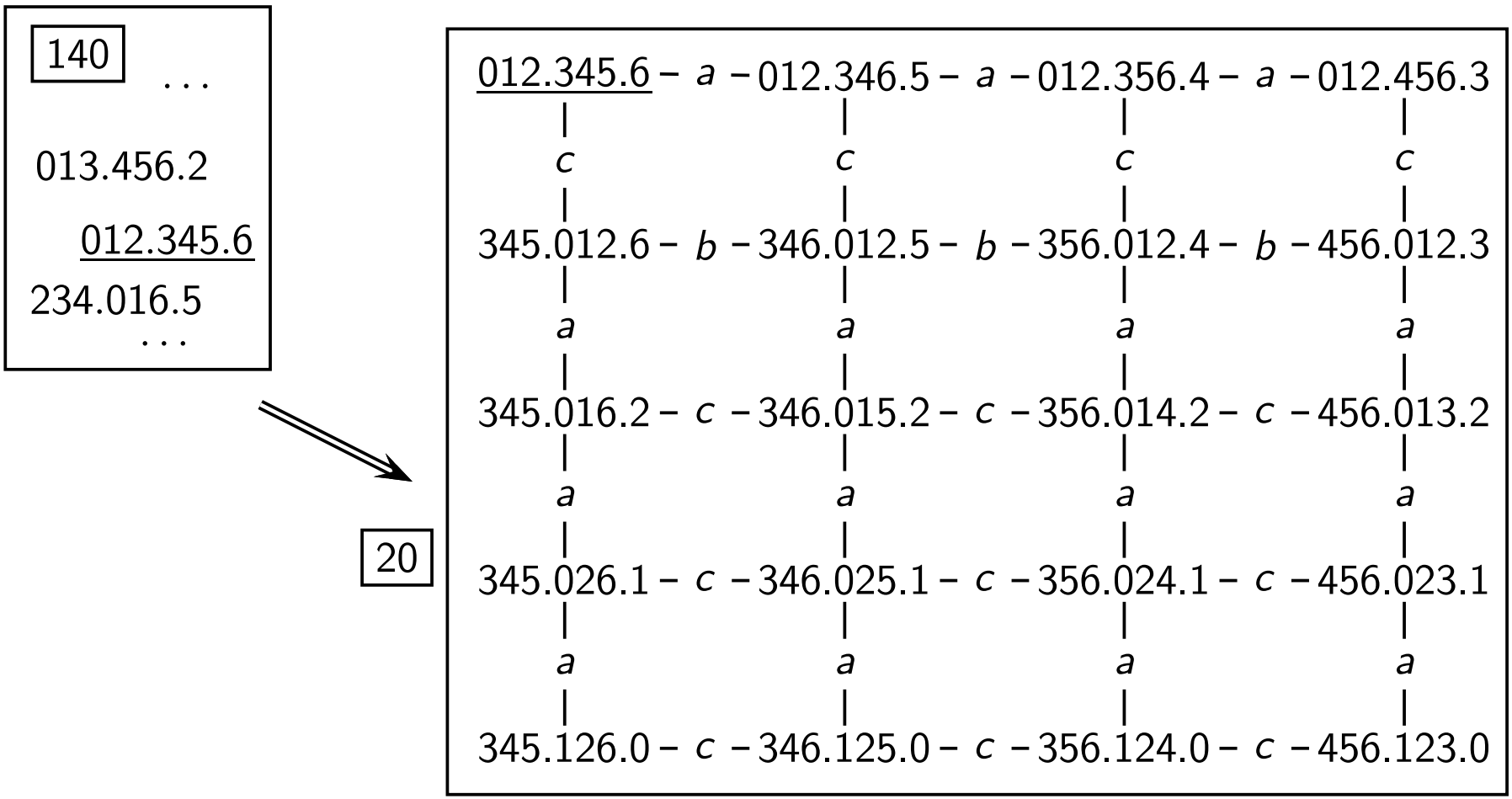
Alice says "I have $\{0,1,2\}$*, or I have none of these cards."*
Eve is ignorant after Alice's announcement.
Alice knows that Eve is ignorant.
Eve doesn't know that Alice knows that Eve is ignorant.
But Eve may assume that Alice knows that Eve is ignorant.
That is informative for Eve!

$$012.345.6 \models [K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a))]\text{cignorant}$$
$$012.345.6 \models [K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a))]K_a\text{cignorant}$$
$$012.345.6 \not\models [K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a))]K_c K_a\text{cignorant}$$
$$012.345.6 \models [K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a))][K_a\text{cignorant}]\neg\text{cignorant}$$

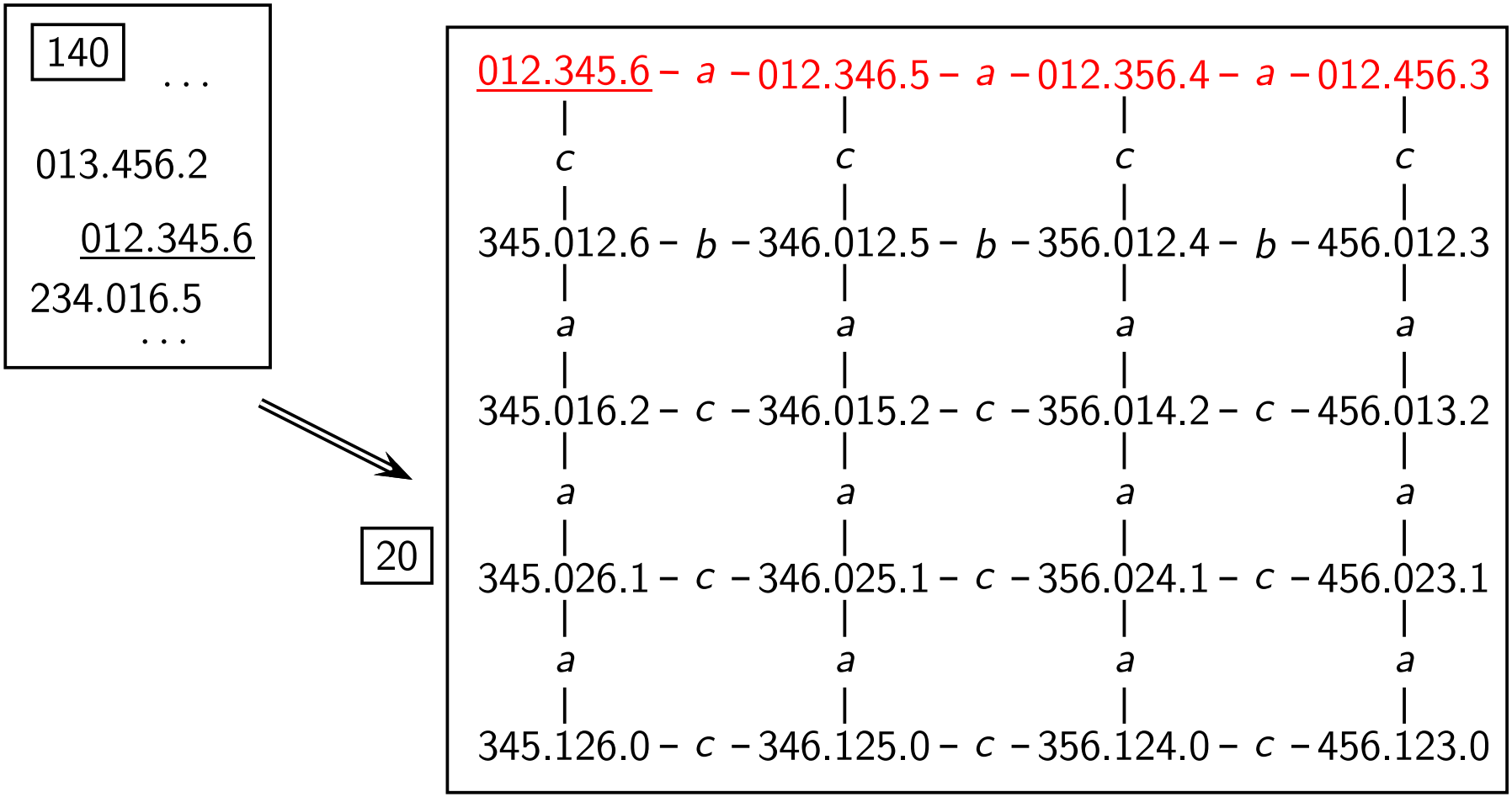
$$012.345.6 \models [K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a))][K_a\text{cignorant}]\neg K_a\text{cignorant}$$

Alice reveals her cards, *because* she intends to keep them secret.

Public communication of secrets: almost good



Public communication of secrets: almost good



Public communication of secrets

Safe announcements guarantee public preservation of ignorance.

$[\varphi]$	announcement of φ (by an observer)
$[K_a\varphi]$	announcement of φ (by agent/Alice)
$[K_a\varphi \wedge [K_a\varphi]C_{abc}\text{cignorant}]$	safe announcement of φ
$[K_a\varphi][C_{abc}\text{cignorant}]$	

Good protocols produce finite sequences of safe announcements s.t.

$$C_{abc}(\text{aknowsbs} \wedge \text{bknowsas} \wedge \text{cignorant})$$

One hundred prisoners and a lightbulb

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 that is the only way in which they can communicate). The light is initially switched off. There is no fixed order of interrogation, or interval between interrogations, and the same prisoner may be interrogated again at any stage. 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. While still in the dining room, and before the prisoners go to their isolation cells (forever), can the prisoners agree on a protocol that will set them free?

Playing cards with Hintikka
An introduction to dynamic epistemic logic[*]

H. P. van Ditmarsch
Department of Computer Science,
University of Otago
hans@cs.otago.ac.nz

W. van der Hoek
Department of Computer Science,
The University of Liverpool
wiebe@csc.liv.ac.uk

B. P. Kooi
Department of Philosophy,
University of Groningen
barteld@philos.rug.nl

Received by Greg Restall
Published October 6, 2005
http://www.philosophy.unimelb.edu.au/ajl/2005

Abstract: This contribution is a gentle introduction to so-called dynamic epistemic logics, that can describe how agents change their knowledge and beliefs. We start with a concise introduction to epistemic logic, through the example of one, two and finally three players holding cards; and, mainly for the purpose of motivating the dynamics, we also very summarily introduce the concepts of general and common knowledge. We then pay ample attention to the logic of public announcements, wherein agents change their knowledge as the result of public announcements. One crucial topic in that setting is that of unsuccessful updates: formulas that become false when announced. The Moore-sentences that were already extensively discussed at the conception of epistemic logic in Hintikka's 'Knowledge and Belief' (1962) give rise to such unsuccessful updates. After that, we present a few examples of more complex epistemic updates.

[*]Professor Jaakko Hintikka kindly gave permission to use his name in the title. He also observed that "My late wife Merrill was one of the best female blackjack players in the world and a championship level bridge player. Hence twenty years ago you would have been well advised to specify which Hintikka you refer to in your title!" Section 5 is partly based on a chapter of [vDvdHK06], and Section 6 is partly based on a section of [vDK05]. We thank an anonymous referee for very detailed helpful comments.

Our closing observations are on recent developments that link the 'standard' topic of (theory) belief revision, as in 'On the Logic of Theory Change: partial meet contraction and revision functions', by Alchourron et al. (1985), to the dynamic epistemic logics introduced here.

1 INTRODUCTION

Imagine three players Anne, Bill, and Cath, each holding one card from a stack of three (known) cards clubs, hearts, and spades, such that they know their own card but do not know which other card is held by which other player. Assume that the actual deal is that Anne holds clubs, Bill holds hearts and Cath holds spades. Now Anne announces that she does not have hearts. What was known before this announcement, and how does this knowledge change as a result of that action? Before, Cath did not know that Anne holds clubs, but afterwards she knows that Anne holds clubs. This is because Cath can reason as follows: "I have spades, so Anne must have clubs or hearts. If she says that she does not have hearts, she must therefore have clubs." Bill knows that Cath now knows Anne's card, even though he does not know himself what Anne's card is. Both before and after, players know which card they hold in their hands. Note that the *only* change that appears to have taken place is epistemic change, and that no factual change has taken place, such as cards changing hands. How do we describe such an information update in an epistemic setting? We can imagine various other actions that affect the knowledge of the players, for example, the action where Anne *shows* her clubs card to Bill, in such a way that Cath sees that Anne is doing that, but without seeing the actual card. How does that affect the knowledge of the players about each other? After that action, Cath still does not know whether Anne holds clubs or hearts. But Cath now knows that Bill knows Anne's card.

This contribution is a gentle introduction to so-called dynamic epistemic logics, that can describe how agents change their knowledge and beliefs. We start with a concise introduction to epistemic logic, through the example of one, two and finally three players holding cards; and, mainly for the purpose of motivating the dynamics, we also very summarily introduce the concepts of general and common knowledge. We then pay ample attention to the logic of public announcements, wherein agents change their knowledge as the result of, indeed, public announcements. One crucial topic in that setting is that of unsuccessful updates: formulas that become false when announced. The Moore-sentences that were already extensively discussed at the conception of epistemic logic in [Hin62] give rise to such unsuccessful updates. After that, we present a few examples of more complex epistemic updates. Our closing observations are on recent developments that link the 'standard' topic of (theory) belief revision [AGM85] to the dynamic epistemic logics introduced here.

2 ONE AGENT

We introduce epistemic logic by a simple example, even simpler than the one in the introduction. Suppose there is only one player: Anne.

> *Anne draws one card from a stack of three different cards clubs, hearts, and spades. Suppose she draws the clubs card—but she does not look at her card yet; and that one of the remaining cards is put back into the stack holder, suppose that is the hearts card; and that the remaining card is put (face down) on the table. That must therefore be the spades card! Anne now looks at her card.*

What does Anne know? We would like to be able to evaluate system descriptions such as:

- Anne holds the clubs card.
- Anne knows that she holds the clubs card.
- Anne does not know that the hearts card is on the table.
- Anne can imagine that the hearts card is not on the table.
- Anne knows that the hearts card or the spades card is in the stack holder.
- Anne knows her own card.
- The card on the table is not held by Anne.
- Anne knows that she holds one card.

Facts about the state of the world are in this case facts about card ownership. We describe such facts by atoms such as $\mathit{Clubs}_{\mathrm{a}}$ standing for 'the clubs card is held by Anne', and similarly $\mathit{Clubs}_{\mathrm{h}}$ for 'the clubs card is in the stack holder', and $\mathit{Clubs}_{\mathrm{t}}$ for 'the clubs card is on the table', etc. The standard propositional connectives are $\wedge$ for 'and', $\vee$ for 'or', $\neg$ for 'not', $\rightarrow$ for 'implies', and $\leftrightarrow$ for 'if and only if'. A formula of the form $\mathsf{K}\varphi$ expresses that 'Anne knows that φ', and a formula of the form $\hat{\mathsf{K}}\varphi$ ($\hat{\mathsf{K}}$ is the dual of K) expresses that 'Anne can imagine that φ'. The informal descriptions above become

- Anne holds the clubs card: $\mathit{Clubs}_{\mathrm{a}}$
- Anne knows that she holds the clubs card: $\mathsf{K}\mathit{Clubs}_{\mathrm{a}}$
- Anne does not know that the hearts card is on the table: $\neg\mathsf{K}\mathit{Hearts}_{\mathrm{t}}$
- Anne can imagine that the hearts card is not on the table: $\hat{\mathsf{K}}\neg\mathit{Hearts}_{\mathrm{t}}$
- Anne knows that the hearts card or the spades card is in the stack holder: $\mathsf{K}(\mathit{Hearts}_{\mathrm{h}} \vee \mathit{Spades}_{\mathrm{h}})$
- Anne knows her own card: $\mathsf{K}\mathit{Clubs}_{\mathrm{a}} \vee \mathsf{K}\mathit{Hearts}_{\mathrm{a}} \vee \mathsf{K}\mathit{Spades}_{\mathrm{a}}$

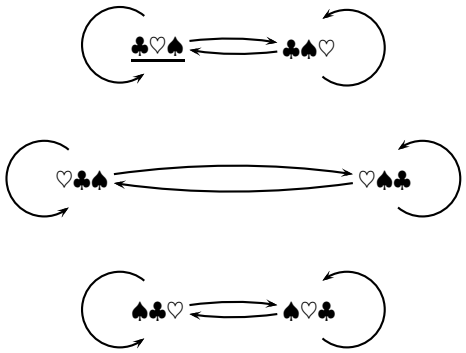


Figure 1: A pointed Kripke model, also known as an epistemic state, that represents Anne's knowledge of the card deal where Anne holds clubs, hearts is in the stack holder, and spades is on the table. The actual state is underlined.

- The card on the table is not held by Anne:

$$(\mathit{Clubs}_{\mathrm{t}} \to \neg \mathit{Clubs}_{\mathrm{a}}) \wedge (\mathit{Hearts}_{\mathrm{t}} \to \neg \mathit{Hearts}_{\mathrm{a}}) \wedge (\mathit{Spades}_{\mathrm{t}} \to \neg \mathit{Spades}_{\mathrm{a}})$$

- Anne knows that she holds one card:

$$\begin{aligned} \mathsf{K}\big((\mathit{Clubs}_{\mathrm{a}} \to (\neg \mathit{Hearts}_{\mathrm{a}} \wedge \neg \mathit{Spades}_{\mathrm{a}})) \wedge \\ (\mathit{Hearts}_{\mathrm{a}} \to (\neg \mathit{Clubs}_{\mathrm{a}} \wedge \neg \mathit{Spades}_{\mathrm{a}})) \wedge \\ (\mathit{Spades}_{\mathrm{a}} \to (\neg \mathit{Hearts}_{\mathrm{a}} \wedge \neg \mathit{Clubs}_{\mathrm{a}}))\big) \end{aligned}$$

So far, so good. Now how are we going to interpret these formulas? The operator K can be interpreted as a modal operator, of the 'necessity'—or $\Box$—type, on structures that are Kripke models. Formally, an epistemic state, or information state, is a pointed relational structure consisting of a set of 'states of the world', a binary relation of 'accessibility' between states, and a factual description of the states—i.e., a valuation of facts on all states. In our example, the states are card deals. The deal where Anne holds the clubs card, the hearts card is in the stack holder and the spades card is on the table, we give the 'name' ♣♡♠, etc. By identifying states with deals, we have implicitly specified the evaluation of facts in the state with the name ♣♡♠. The binary relation of accessibility between states expresses what the player knows about the facts. For example, if deal ♣♡♠ is actually the case, Anne holds the clubs card, and in that case she can imagine that not ♣♡♠ but ♣♠♡ is the case, wherein she also holds the clubs card. We say that state ♣♠♡ is accessible from state ♣♡♠ for Anne, or that (♣♡♠, ♣♠♡) is in the accessibility relation. Also, she can imagine the actual deal ♣♡♠ to be the case, so ♣♡♠ is 'accessible from itself': the pair (♣♡♠, ♣♡♠) must also be in the accessibility relation.

Continuing in this way, we get the accessibility relation in Figure 1. This structure can formally be described as a pointed Kripke model $(\mathit{Hexa}_{\mathrm{a}}, ♣♡♠)$ where the model $\mathit{Hexa}_{\mathrm{a}} = \langle \mathsf{S}, \mathsf{R}, \mathsf{V} \rangle$ consists of a domain S, accessibility relation

R and valuation V such that

$$
\begin{aligned}
&S = \{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit, \heartsuit\clubsuit\spadesuit, \heartsuit\spadesuit\clubsuit, \spadesuit\clubsuit\heartsuit, \spadesuit\heartsuit\clubsuit\}\\
&R = \{(\clubsuit\heartsuit\spadesuit, \clubsuit\heartsuit\spadesuit), (\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit), (\clubsuit\spadesuit\heartsuit, \clubsuit\spadesuit\heartsuit), \dots\}\\
&V(\mathit{Clubs}_a) = \{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit\}\\
&V(\mathit{Hearts}_a) = \{\heartsuit\clubsuit\spadesuit, \heartsuit\spadesuit\clubsuit\}\\
&\dots
\end{aligned}
$$

The states where a given atom is true are identified with a subset of the domain: Clubs_a—for 'Anne holds the clubs card'—is only true in states $\{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit\}$, etc. A standard modal language inductively defined by $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \Box\varphi$ can now interpreted on this structure—let's stick to the familiar $\Box$ for a little while, before we write K for that. The crucial clause in the interpretation of formulas is the one for the modal operator: $M, s \models \Box\varphi$ if and only if for all t, if $R(s,t)$, then $M, t \models \varphi$. For $M, s \models \varphi$ read 'state s of model M satisfies formula φ', or 'φ is true in state s of model M'. For example, we can now compute that in the epistemic state $(\mathit{Hexa}_a, \clubsuit\heartsuit\spadesuit)$ it is indeed true that Anne knows that she holds the clubs card:

> We have that $\mathit{Hexa}_a, \clubsuit\heartsuit\spadesuit \models \Box\mathit{Clubs}_a$ if and only if (for all states s, if $R(\clubsuit\heartsuit\spadesuit, s)$ then $\mathit{Hexa}_a, s \models \mathit{Clubs}_a$). The last is implied by $\mathit{Hexa}_a, \clubsuit\heartsuit\spadesuit \models \mathit{Clubs}_a$ and $\mathit{Hexa}_a, \clubsuit\spadesuit\heartsuit \models \mathit{Clubs}_a$, as the only states that are accessible from $\clubsuit\heartsuit\spadesuit$ are $\clubsuit\heartsuit\spadesuit$ itself and $\clubsuit\spadesuit\heartsuit$: we have $R(\clubsuit\heartsuit\spadesuit, \clubsuit\heartsuit\spadesuit)$ and $R(\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit)$. Finally, $\mathit{Hexa}_a, \clubsuit\heartsuit\spadesuit \models \mathit{Clubs}_a$ because $\clubsuit\heartsuit\spadesuit \in V(\mathit{Clubs}_a) = \{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit\}$, and, similarly, $\mathit{Hexa}_a, \clubsuit\spadesuit\heartsuit \models \mathit{Clubs}_a$ because $\clubsuit\spadesuit\heartsuit \in V(\mathit{Clubs}_a) = \{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit\}$. Done! From now on, we will always write K for $\Box$.

It turns out that Anne's accessibility relation is an equivalence relation. If one assumes certain properties of knowledge, this is always the case. The properties are that 'what you know is true', which is formalized by the schema $K\varphi \rightarrow \varphi$; that 'you are aware of your knowledge', which is formalized by the schema $K\varphi \rightarrow KK\varphi$, and that 'you are aware of your ignorance', which is formalized by the schema $\neg K\varphi \rightarrow K\neg K\varphi$. These properties may be disputed for various reasons, for example, without the requirement that what you know is true, we get a notion of belief instead of knowledge. For now, also for the sake of a simple exposition, we will stick to the properties of knowledge and see where they get us. Together, they enforce that in epistemic logic the accessibility relation is always an equivalence relation. This is somewhat differently expressed, by saying that what a player / agent cannot distinguish between induces a *partition* on the set of states, i.e., a set of equivalence classes that cover the entire domain. For equivalence relations, we write $\sim$ instead of R, and we write this 'infix', i.e., we write $\clubsuit\heartsuit\spadesuit \sim \clubsuit\spadesuit\heartsuit$ instead of $R(\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit)$. In the case of equivalence relations a simpler visualization is sufficient: we only need to link visually the states that are in the same class. If a state is not linked to

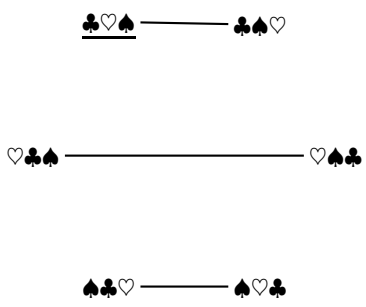


Figure 2: A simpler visualization of the epistemic state where Anne holds clubs, hearts is in the stack holder, and spades is on the table. The actual state is underlined.

others, it must be a singleton equivalence class (reflexivity always holds). For $(\mathit{Hexa}_{\mathrm{a}}, \clubsuit\heartsuit\spadesuit)$ we get the visualization in Figure 2.

One might ask: why not restrict ourselves in the model to the two deals $\clubsuit\heartsuit\spadesuit$ and $\clubsuit\spadesuit\heartsuit$ only? The remaining deals are inaccessible anyway from the actual deal! From an agent's point of view this is arguably right, but from a modeller's point of view the six-point model is preferable: this model works regardless of the actual deal.

The dual of 'know' is 'can imagine that' (or 'considers it possible that'): $\hat{\mathrm{K}}\varphi \;:= \neg\mathrm{K}\neg\varphi$, so that 'can imagine that' means 'not knowing that not'. For example, 'Anne can imagine that the hearts card is not on the table' is described by $\hat{\mathrm{K}}\neg\mathit{Hearts}_{\mathrm{t}}$ which is true in epistemic state $(\mathit{Hexa}_{\mathrm{a}}, \clubsuit\heartsuit\spadesuit)$, because from deal $\clubsuit\heartsuit\spadesuit$ Anne can access deal $\clubsuit\heartsuit\spadesuit$ for which $\neg\mathit{Hearts}_{\mathrm{t}}$ is true, as the spades card is on the table in that deal. There appears to be no generally accepted notation for 'can imagine that'. The 'hat' in the notation $\hat{\mathrm{K}}\varphi$—the notation we will keep using—is reminiscent of the diamond in $\Diamond\varphi$. Other notations for $\hat{\mathrm{K}}\varphi$ are $\mathrm{M}\varphi$ and $\mathrm{k}\varphi$.

3 MORE AGENTS

Many features of formal dynamics can be presented based on the single-agent situation. For example, the action of Anne picking up the card from the table that has been dealt to her, is a significantly complex epistemic action. But a proper and more interesting perspective is that of the multi-agent situation. This is because players may now have knowledge about each others' knowledge, so that for even a single fact the Kripke models representing that knowledge can become arbitrarily complex. For a start, let's move from one to two players in the three cards situation:

> *Anne and Bill both draw one card from the cards clubs, hearts, and spades. The remaining card is put (face down) on the table. Suppose Anne draws*

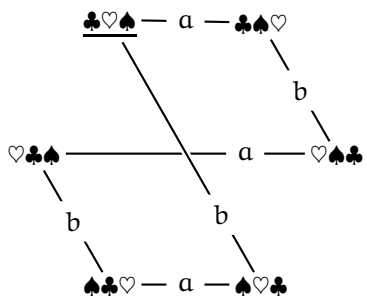


Figure 3: Anne and Bill both draw one card from the cards clubs, hearts, and spades. The remaining card is put (face down) on the table. Anne draws the clubs card and Bill draws the hearts card.

the clubs card and Bill draws the hearts card.

The epistemic operator K with corresponding access $\sim$, to describe Anne's knowledge, now has to be different from an epistemic operator and corresponding access for Bill. The distinction can easily be made by labelling an operator, and access, with the agent that it is knowledge and access for. If we take $\mathfrak{a}$ for Anne, and $\mathfrak{b}$ for Bill, this results in equivalence relations $\sim_{\mathfrak{a}}$ and $\sim_{\mathfrak{b}}$ and corresponding knowledge operators $\mathsf{K}_{\mathfrak{a}}$ and $\mathsf{K}_{\mathfrak{b}}$. Bill's access on the domain is different from Anne's: whereas Anne cannot tell deals ♣♡♠ and ♣♠♡ apart, Bill instead cannot tell deals ♣♡♠ and ♠♡♣ apart, etc. The resulting model $\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}$ is depicted in Figure 3. We can now describe in the epistemic language that, for example:

- Bill cannot imagine that Anne has the hearts card: $\neg\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$
- Anne can imagine that Bill can imagine that she has the hearts card: $\hat{\mathsf{K}}_{\mathfrak{a}}\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$
- Anne knows Bill can imagine that she has the clubs card: $\mathsf{K}_{\mathfrak{a}}\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Clubs}_{\mathfrak{a}}$

The formula $\hat{\mathsf{K}}_{\mathfrak{a}}\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$ is true in epistemic state $(\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}, ♣♡♠)$—formally, $(\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}, ♣♡♠) \models \hat{\mathsf{K}}_{\mathfrak{a}}\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$. This can be shown as follows.

> We have that ♣♡♠ $\sim_{\mathfrak{a}}$ ♣♠♡ and that ♣♠♡ $\sim_{\mathfrak{b}}$ ♡♠♣. In the last state, we have $(\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}, ♡♠♣) \models \mathit{Hearts}_{\mathfrak{a}}$. From that and ♣♠♡ $\sim_{\mathfrak{b}}$ ♡♠♣ follows $(\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}, ♣♠♡) \models \hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$, and from that and ♣♡♠ $\sim_{\mathfrak{a}}$ ♣♠♡ follows $(\mathit{Hexa}_{\mathfrak{a}\mathfrak{b}}, ♣♡♠) \models \hat{\mathsf{K}}_{\mathfrak{a}}\hat{\mathsf{K}}_{\mathfrak{b}}\mathit{Hearts}_{\mathfrak{a}}$.

For three cards and three agents, we get the model *Hexa* pictured in Figure 4, and we can now describe in the epistemic language that:

- Anne knows that Bill knows that Cath knows her own card

$$\mathsf{K}_{\mathfrak{a}}\mathsf{K}_{\mathfrak{b}}(\mathsf{K}_{\mathfrak{c}}\mathit{Clubs}_{\mathfrak{c}} \vee \mathsf{K}_{\mathfrak{c}}\mathit{Hearts}_{\mathfrak{c}} \vee \mathsf{K}_{\mathfrak{c}}\mathit{Spades}_{\mathfrak{c}})$$

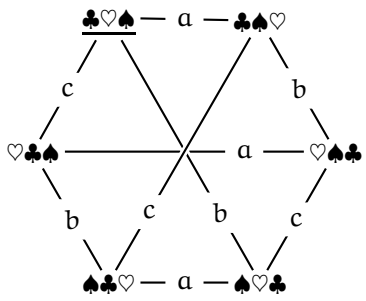


Figure 4: The epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ for the card deal where Anne holds clubs, Bill holds hearts, and Cath holds spades.

- Anne has the clubs card, but Anne can imagine that Bill can imagine that Cath knows that Anne does not have the clubs card: $\mathit{Clubs}_a \wedge \hat{K}_a\hat{K}_b K_c \neg \mathit{Clubs}_a$

The structures we will use throughout this presentation can now be introduced formally as follows:

Definition 1 (Epistemic structures) An *epistemic model* $M = \langle S, \sim, V\rangle$ consists of a *domain* S of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \to \mathcal{P}(S \times S)$, and a *valuation* $V : P \to \mathcal{P}(S)$. For $s \in S$, (M, s) is an *epistemic state*.

For $\sim(n)$ we write $\sim_n$ and for $V(p)$ we write V_p. So, access $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_p. Relative to a set of agents N and a set of atoms P, the language of multiagent epistemic logic is inductively defined by $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi$. We need some further extensions of the language, but all these will be interpreted on the structures presented in Definition 1.

4 common knowledge

The first extension of the language is with epistemic operators for *groups* of agents. We will add common knowledge operators. As we aim to focus on *dynamic* epistemics in this contribution, and not on dynamic *epistemics*, this will be a lightning quick introduction to 'common knowledge'. For more information, see [FHMV95, MvdH95].

In the epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ of Figure 4 both Anne and Bill know that the deal of cards is *not* $\spadesuit\clubsuit\heartsuit$: both $K_a\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$ and $K_b\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$ are true. If a group of agents all individually know that φ, we say that φ is general knowledge. The modal operator for general knowledge of a group G is E_G. For an arbitrary subset $G \subseteq N$ of the set of agents N, we define $E_G\varphi := \bigwedge_{n\in G} K_n\varphi$. So in this case we have

that $E_{ab}\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$—par 'abus de langage' we write E_{ab} instead of $E_{\{a,b\}}$. Now even though φ may be generally known, that does not imply that agents know *about each other* that they know φ. For example, $K_bK_a\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$ is *false* in $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$: Bill can imagine Anne to have the spades card instead of clubs. In that case, Anne can imagine that the card deal is $\spadesuit\clubsuit\heartsuit$. So $\hat{K}_a\hat{K}_b(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$ is true, and therefore $K_bK_a\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c)$ is false. For other examples, one can construct formulas that are true to some extent $K_aK_bK_cK_aK_aK_b\varphi$ but no longer if one adds one more operator at the start, e.g., $K_bK_aK_bK_cK_aK_aK_b\varphi$ is false. A formula φ is *common knowledge* for a group G, notation $C_G\varphi$, if it holds for arbitrary long stacks of individual knowledge operators (for individuals in that group). If, for example, $G = \{a, b, c\}$, we get something (involving an enumeration of all finite stacks of knowledge operators) like $C_{abc}\varphi := \varphi \wedge K_a\varphi \wedge K_b\varphi \wedge K_c\varphi \wedge K_aK_a\varphi \wedge K_aK_b\varphi \wedge K_aK_c\varphi \wedge \ldots K_aK_aK_a\varphi \ldots$. Alternatively, we may see common knowledge as the conjunction of arbitrarily many applications of general knowledge: $C_G\varphi := \varphi \wedge E_G\varphi \wedge E_GE_G\varphi \wedge \ldots$. Such infinitary definitions are frowned upon. Therefore common knowledge C_G is added as a primitive operator to the language, whereas general knowledge is typically defined (for a finite set of agents) by the notational abbreviation above. Instead, common knowledge is defined semantically, by an operation on the accessibility relations for the individual agents in the group (namely transitive closure of their union). By way of validities involving common knowledge, that are mentioned at the end of this section, any single conjunct from the right-hand side of the infinitary definition of common knowledge is then entailed, and thus we avoid having to define it in an infinitary way.

The semantics of common knowledge formulas is: $C_G\varphi$ is true in an epistemic state (M, s) if φ is true in any state s_m *that can be reached by a finite path of linked states* $s \sim_{n_1} s_1 \sim_{n_2} s_2 \sim_{n_3} \cdots \sim_{n_m} s_m$, with all of $n_1, \ldots, n_m \in G$ (and not necessarily all different). Mathematically, 'reachability by a finite path' is the same as 'being in the transitive reflexive closure'. If we define $\sim_G$ as $(\bigcup_{n\in G})^*$—which is that reflexive transitive closure—then we interpret common knowledge as

$$M, s \models C_G\varphi \text{ if and only if for all } t : s \sim_G t \text{ implies } M, t \models \varphi$$

If all individual accessibility relations are equivalence relations, $\sim_G$ is also an equivalence relation [MvdH95]. Common knowledge for the entire group N of agents is called *public knowledge*.

In the model *Hexa*, access for any subgroup of two players, or for all three, is the entire model. For such groups G, $C_G\varphi$ is true in an epistemic state (Hexa, t) iff φ is valid on the model *Hexa*—a formula is valid on a model M, notation $M \models \varphi$, if and only if for all states s in the domain of M: $M, s \models \varphi$. For example, we have that:

- It is public knowledge that Anne knows her card:

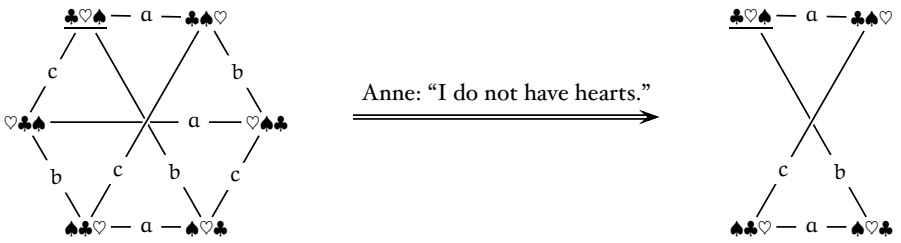


Figure 5: On the left, the epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ for the card deal where Anne hold clubs, Bill holds hearts, and Cath holds spades. The actual deal is underlined. On the right, the effect of Anne saying that she does not have hearts.

$$\mathit{Hexa} \models \mathsf{C}_{abc}(\mathsf{K}_a\mathit{Clubs}_a \vee \mathsf{K}_a\mathit{Hearts}_a \vee \mathsf{K}_a\mathit{Spades}_a)$$

- Anne and Bill share the same knowledge as Bill and Cath:
  $\mathit{Hexa} \models \mathsf{C}_{ab}\varphi \to \mathsf{C}_{bc}\varphi$

Valid principles for common knowledge are $\mathsf{C}_G(\varphi \to \psi) \to (\mathsf{C}_G\varphi \to \mathsf{C}_G\psi)$ (distribution of C_G over $\to$), and $\mathsf{C}_G\varphi \to (\varphi \wedge \mathsf{E}_G\mathsf{C}_G\varphi)$ (use of C_G), and $\mathsf{C}_G(\varphi \to \mathsf{E}_G\varphi) \to (\varphi \to \mathsf{C}_G\varphi)$ (induction). Some grasp of group concepts of knowledge is important to understand the effects of public announcements, but we will not pay more attention here to these concepts.

5 PUBLIC ANNOUNCEMENTS

We now move on to the dynamics of knowledge. Suppose Anne says that she does not have the hearts card. She then makes public to all three players that all deals where Hearts_a is true can be eliminated from consideration. This results in a restriction of the model *Hexa* as depicted in Figure 5. The public announcement "I do not have hearts" can be seen as an epistemic 'program' with 'precondition' $\neg\mathit{Hearts}_a$, that is interpreted as a 'state transformer' of the original epistemic state, exactly as a program in *dynamic* modal logic. Given some program π, in dynamic logic $[\pi]\psi$ means that after every execution of π (state transformation induced by π), formula ψ holds. For announcements we want something of the form $[\varphi]\psi$, meaning that after (every) announcement of φ, formula ψ holds.

We *appear* to be moving away slightly from the standard paradigm of modal logic. So far, the accessibility relations were between states in a given model underlying an epistemic state. But all of a sudden, we are confronted with an accessibility relation between epistemic states as well: "I do not have hearts" induces a(n) (epistemic) state transition such that the pair of epistemic states in Figure 5 is in that relation. The epistemic states take the role of the points or

worlds in a seemingly underspecified domain of 'all possible epistemic states'. By lifting accessibility between points in the original epistemic state to accessibility between epistemic states, we can get the dynamic and epistemic accessibility relations 'on the same level' again, and see this as an 'ordinary structure' on which to interpret a perfectly ordinary multimodal logic. A crucial point is that this 'higher-order structure' is induced by the initial epistemic state and the actions that can be executed there, and not the other way round. So it's standard modal logic after all.

Anne's announcement "I do not have hearts" is a simple epistemic action in various respects. It is public, and therefore not private or even something else. It is truthful, and not merely introspective or even weaker; in that sense it describes change of knowledge only and not change of belief. It is deterministic, i.e. a state transformer; other actions, of which we will see an example, are non-deterministic.

The effect of the public announcement of φ is the restriction of the epistemic state to all worlds where φ holds. So, 'announce φ' can indeed be seen as an information state transformer, with a corresponding dynamic modal operator $[\varphi]$. We now formally introduce the language with all the operators we have seen so far.

Definition 2 (public announcement language) Given a set of agents N and a set of atoms P, let $p \in P$, $n \in N$, and $G \subseteq N$ be arbitrary. The language of public announcements is inductively defined as

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

Definition 3 (semantics) Given an epistemic model $M = \langle S, \sim, V\rangle$, we define:

$$\begin{array}{lll}
M, s \models p & : \text{iff} & s \in V_p \\
M, s \models \neg\varphi & : \text{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & : \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_n\varphi & : \text{iff} & \text{for all } t \in S : s \sim_n t \text{ implies } M, t \models \varphi \\
M, s \models C_G\varphi & : \text{iff} & \text{for all } t \in S : s \sim_G t \text{ implies } M, t \models \varphi \\
M, s \models [\varphi]\psi & : \text{iff} & M, s \models \varphi \text{ implies } M|\varphi, s \models \psi
\end{array}$$

where $M|\varphi := \langle S', \sim', V'\rangle$ is defined as follows:

$$\begin{array}{lcl}
S' & := & \{s' \in S \mid M, s' \models \varphi\} \\
\sim'_n & := & \sim_n \cap (S' \times S') \\
V'_p & := & V_p \cap S'
\end{array}$$

In other words: the model $M|\varphi$ is the model M restricted to all the states where φ holds, including access between states. The interpretation of the dual $\langle\varphi\rangle$ of $[\varphi]$ will be obvious: $M, s \models \langle\varphi\rangle\psi$ if and only if $M, s \models \varphi$ and $M|\varphi, s \models \psi$.

Formula φ is valid on model M, notation $M \models \varphi$, if and only if for all states s in the domain of M: $M, s \models \varphi$. Formula φ is valid, notation $\models \varphi$, if and only if for all models M (of the class of models for the given parameters of N and P): $M \models \varphi$. A proof system for this logic originates with and is proved sound and complete in [BMS98], with precursors (namely completeness results for the logic *with* announcements but *without* common knowledge) in [Pla89] and [GG97].

After Anne's announcement that she does not have hearts, Cath knows that Anne has clubs (see Figure 5). We can verify this with a semantic computation as follows:

> In order to prove that $\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \models [\neg \mathit{Hearts}_a]K_c \mathit{Clubs}_a$, we have to show that $\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \models \neg \mathit{Hearts}_a$ implies $\mathit{Hexa}|\neg \mathit{Hearts}_a, \clubsuit\heartsuit\spadesuit \models K_c \mathit{Clubs}_a$. As it is indeed the case that $\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \models \neg \mathit{Hearts}_a$, it only remains to show that $\mathit{Hexa}|\neg \mathit{Hearts}_a, \clubsuit\heartsuit\spadesuit \models K_c \mathit{Clubs}_a$. The set of states that is equivalent to $\clubsuit\heartsuit\spadesuit$ for Cath, is the singleton set $\{\clubsuit\heartsuit\spadesuit\}$. So it is sufficient to show that $\mathit{Hexa}|\neg \mathit{Hearts}_a, \clubsuit\heartsuit\spadesuit \models \mathit{Clubs}_a$, which follows trivially from $\clubsuit\heartsuit\spadesuit \in V_{\mathit{Clubs}_a} = \{\clubsuit\heartsuit\spadesuit, \clubsuit\spadesuit\heartsuit\}$.

The semantics of public announcement is actually slightly imprecise. Consider what happens if in "$M, s \models [\varphi]\psi$ if and only if $M, s \models \varphi$ implies $M|\varphi, s \models \psi$" the formula φ is false in M, s. In that case, $M|\varphi, s \models \psi$ is undefined, because s is now not part of the domain of the model $M|\varphi$. Apparently, we 'informally' use that an implication 'antecedent implies consequent' in the meta-language is not just true when the antecedent is false or the consequent is true, in the standard binary sense, where both antecedent and consequent are defined. But we also use that the implication is true when the antecedent is false even when the consequent is undefined. A more precise definition of the semantics of public announcement, that does not have that informality, is: $M, s \models [\varphi]\psi$ if and only if for all (M', t) such that $(M, s)[\![\varphi]\!](M', t)$: $(M', t) \models \psi$. In this definition, $(M, s)[\![\varphi]\!](M', t)$ holds if and only if $M' = M|\varphi$ and $s = t$. The general definition of the interpretation of epistemic actions, of which 'announcement' is just an example, has a very similar form.

To give the reader a feel for what goes in this logic we give some of its valid principles. In all cases we only give motivation and we refrain from proofs.

If an announcement can be executed, there is only one way to do it:

$$\langle\varphi\rangle\psi \to [\varphi]\psi \text{ is valid}$$

This is a simple consequence of the functionality of the state transition semantics for the announcement. Of course, the converse $[\varphi]\psi \to \langle\varphi\rangle\psi$ does not hold. Take $\varphi = \psi = \bot$ ($\bot$ is 'falsum'). We now have that $[\bot]\bot$ is valid (for trivial reasons) but $\langle\bot\rangle\bot$ is, of course, always false, for the same trivial reason

that no epistemic state satisfies $\bot$! Related to the functionality and partiality of 'announcement' are that all of the following are equivalent:

- $\varphi \to [\varphi]\psi$
- $\varphi \to \langle\varphi\rangle\psi$
- $[\varphi]\psi$

A sequence of two announcements can always be replaced by a single, more complex announcement. Instead of first saying 'φ' and then saying 'ψ' you may as well have said for the first time 'φ and after that ψ'. It is expressed in

$$[\varphi \wedge [\varphi]\psi]\chi \text{ is equivalent to } [\varphi][\psi]\chi$$

This turns out to be a quite useful feature for analyzing announcements that are made with specific intentions; or, more generally, conversational implicatures à la Grice. Those intentions tend to be postconditions ψ that supposedly hold after the announcement. So the (truthful) announcement of φ with the intention of achieving ψ corresponds to the announcement $\varphi \wedge [\varphi]\psi$.

For an example sequence of two announcements, consider the following announcement, supposedly made by some outsider that has full knowledge of the epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ (alternatively, such an agent can be modelled as a player with the identity relation for access):

An outsider says: "The deal of cards is neither $\spadesuit\clubsuit\heartsuit$ *nor* $\heartsuit\spadesuit\clubsuit$*."*

This is formalized as $\neg(\mathit{Spades}_a \wedge \mathit{Clubs}_b \wedge \mathit{Hearts}_c) \wedge \neg(\mathit{Hearts}_a \wedge \mathit{Spades}_b \wedge \mathit{Clubs}_c)$. Abbreviate this announcement as one. See Figure 6 for the result of the announcement of one. Observe that none of the three players Anne, Bill, and Cath know the card deal as a result of this announcement! Now imagine that the players know (publicly) that the outsider made the announcement one in the happy knowledge of not revealing the deal of cards to anyone! For example, he might have been boasting about his logical prowess and the players might inadvertently have become aware of that. In other words, it becomes known that the announcement one was made *with the intention of keeping the players ignorant of the card deal*. Ignorance of the card deal (whatever the deal may have been) can be described as some long formula that is a conjunction of eighteen parts and that starts as $\neg K_a(\mathit{Clubs}_a \wedge \mathit{Hearts}_b \wedge \mathit{Spades}_c) \wedge \neg K_b(\mathit{Clubs}_a \wedge \mathit{Hearts}_b \wedge \mathit{Spades}_c) \wedge \neg K_c(\mathit{Clubs}_a \wedge \mathit{Hearts}_b \wedge \mathit{Spades}_c) \wedge \ldots$ and that we abbreviate as two. The formula two is false in all states (in the model resulting from the announcement of one) that are a singleton equivalence class for at least one player, and true anywhere else. So it's only true in state $\clubsuit\heartsuit\spadesuit$. For the result of the announcement of two, see again Figure 6. Observe that in the epistemic state resulting from two, all players now know the card deal! So in that epistemic state two is false. Now what does it mean that the players have become aware of the intention of the outsider? This means that although the

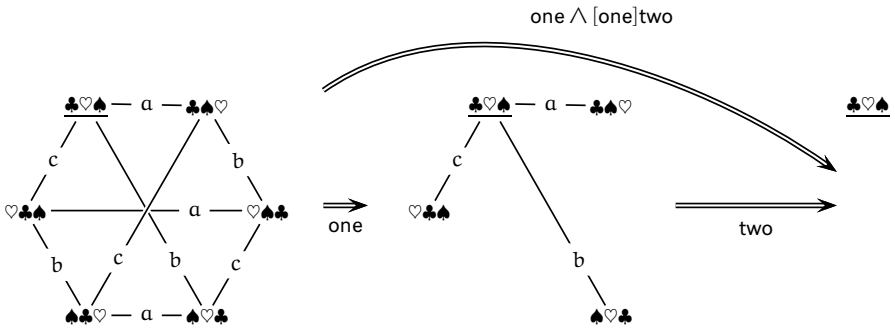


Figure 6: Two announcements in sequence, replaced by one.

outsider was actually saying one, he really meant 'one, and after that two', in other words, he was saying one ∧ [one]two. See again Figure 6. Unfortunately, $\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \models$ [one ∧ [one]two]¬two. The outsider could have kept the card deal a secret, but by intending to keep it a secret—and the assumption that this intention is public knowledge—he was, after all, actually revealing the secret.

The relation of the announced formula to the pre- and postconditions of the announcement is not trivial. To start with, $[\varphi]\mathsf{K}_n\psi$ is not equivalent to $\mathsf{K}_n[\varphi]\psi$. This is a consequence of the fact that $[\varphi]$ is a partial function. A simple counterexample is the following: in $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ it is true that after 'every' announcement of 'Anne holds hearts', Cath knows that Anne holds clubs. This is because that announcement cannot take place in that epistemic state. In other words, we have

$$\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \models [\mathit{Hearts}_a]\mathsf{K}_c\mathit{Clubs}_a$$

On the other hand, it is false that Cath knows that after the announcement of Anne that she holds the hearts card (which she can imagine to take place), Cath knows that Anne holds the clubs card. On the contrary: Cath then knows that Anne holds the hearts card! So we have

$$\mathit{Hexa}, \clubsuit\heartsuit\spadesuit \not\models \mathsf{K}_c[\mathit{Hearts}_a]\mathit{Clubs}_a$$

If we make $[\varphi]\mathsf{K}_n\psi$ conditional to the truth of the announcement, an equivalence indeed holds:

$$[\varphi]\mathsf{K}_n\psi \text{ is equivalent to } \varphi \to \mathsf{K}_n[\varphi]\psi$$

The relationship between announcement and knowledge can be formulated in various ways. One or the other may appeal more to the intuitions of the reader. Often, the 'diamond'-versions of axioms correspond better to one's intuitions than the 'box'-versions. It may sharpen the modeller's wits to realize that all of the following validities express the same equivalence:

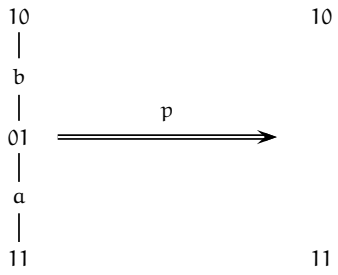


Figure 7: A state transition illustrating what a and b commonly know before and after the announcement of p.

- $[\varphi]K_n\psi \leftrightarrow (\varphi \rightarrow K_n[\varphi]\psi)$
- $\langle\varphi\rangle K_n\psi \leftrightarrow (\varphi \wedge K_n(\varphi \rightarrow \langle\varphi\rangle\psi))$
- $\langle\varphi\rangle \hat{K}_n\psi \leftrightarrow (\varphi \wedge \hat{K}_n\langle\varphi\rangle\psi)$

If we restrict ourselves to the logic of announcements *without* common knowledge, every formula is logically equivalent to one in the logic without announcements. But for the logic of announcements *with* common knowledge, this is no longer the case [BMS98]. Apart from conceptual reasons, such as having a natural specification language for dynamics, *that*, one might say, is the real validation of this logical tool. Let us take a closer look at a principle relating announcements and common knowledge.

The simple generalization of the principle $[\varphi]K_n\psi \leftrightarrow (\varphi \rightarrow K_n[\varphi]\psi)$ relating announcement and individual knowledge would be $[\varphi]C_N\psi \leftrightarrow (\varphi \rightarrow C_N[\varphi]\psi)$. This happens to be invalid. The following countermodel M demonstrates this clearly.

Consider a model M for two agents a and b and two facts p and q. Its domain is $\{11, 01, 10\}$, where 11 is the state where p and q are both true, 01 the state where p is false and q is true, and 10 the state where p is true and q is false. Agent a cannot tell 11 and 01 apart, whereas b cannot tell 01 and 10 apart. So the partition for a on the domain is $\{11, 01\}, \{10\}$ and the partition for b on the domain is $\{11\}, \{01, 10\}$. See Figure 7.

Now consider the instance $[p]C_{ab}q \leftrightarrow (p \rightarrow C_{ab}[p]q)$ of this supposed principle. The left side of the equivalence is true in state 11 of M, whereas the right side is false in that state. We show that as follows. First, $M, 11 \models [p]C_{ab}q$ is true in 11, because $M, 11 \models p$ and $M|p, 11 \models C_{ab}q$. For the result of the announcement of p in $(M, 11)$, see Figure 7. The model $M|p$ consists of two disconnected states; obviously, $M|p, 11 \models C_{ab}q$, because $M|p,11 \models q$ and 11 is now the only reachable state from 11.

On the other hand, we have that $M, 11 \not\models p \to C_{ab}[p]q$, because $M, 11 \models p$ but $M, 11 \not\models C_{ab}[p]q$. The last is because $11 \sim_{ab} 10$ (because $11 \sim_a 01$ and $01 \sim_b 10$), and $M, 10 \not\models [p]q$. When evaluating q in $M|p$, we are now in the *other* disconnected part of $M|p$, where q is false: $M|q, 10 \not\models q$.

Fortunately there are also other ways to get common knowledge after an announcement. The general principle is: If $\chi \to [\varphi]\psi$ and $\chi \wedge \varphi \to E_N\chi$ are valid, then $\chi \to [\varphi]C_N\psi$ is valid as well.

6 UNSUCCESSFUL UPDATES

After announcing φ, φ may remain true but may also have become false! This will not come as a surprise to those familiar with so-called Moore-sentences, that are already discussed in detail in the original presentation of epistemic logic in [Hin62]. This states that you cannot know that some fact is true and that you do not know that. In other words, $K(p \wedge \neg Kp)$ is inconsistent in epistemic logic. This can easily be seen by the following argument: from $K(p \wedge \neg Kp)$ follows $Kp \wedge K\neg Kp$, so follows Kp. But also, from $Kp \wedge K\neg Kp$ follows $K\neg Kp$, and from that, with 'truthfulness', follows $\neg Kp$. Together, Kp and $\neg Kp$ are inconsistent.

Within the setting of the logic of public announcements this can be redescribed as follows: after the truthful announcement (in some given epistemic state) of $(p \wedge \neg Kp)$, this formula can no longer be true (in the resulting epistemic state). In [Ger99] this sort of announcement was called an *unsuccessful update*: you say something "because it's true," but unfortunately, that was not a very successful thing to do, because now it's false!

For a different example, consider the result of Anne announcing in the epistemic state (*Hexa*, ♣♡♠): "I hold the clubs card and (at the time I am saying this) Bill does not know that". This is an announcement of $\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a$ (or of, equivalently, $K_a(\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a)$; note that mixing epistemic operators for *different* agents does not make it 'Moore'). After this announcement, Bill now knows that Anne holds the clubs card, so $K_b\mathit{Clubs}_a$ has become true, and therefore $\neg(\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a)$ as well. The reader can simply check in Figure 8 that after this announcement the formula $\neg(\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a)$ indeed holds, and therefore $\mathit{Hexa}, ♣♡♠ \models [\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a]\neg(\mathit{Clubs}_a \wedge \neg K_b\mathit{Clubs}_a)$.

We appear to be deceived by some intuitive, but incorrect, communicative expectations. If an agent truthfully announces φ to a group of agents, it appears *on first sight* to be the case that (s)he 'makes φ common knowledge' that way: in other words, if φ holds, then after announcing that, $C_N\varphi$ holds. In other words, $\varphi \to [\varphi]C_N\varphi$ *appears* to be valid. This expectation is unwarranted, because the truth of *epistemic* (non-propositional) parts of the formula may be influenced by its announcement. On the other hand—it's not that our intuition

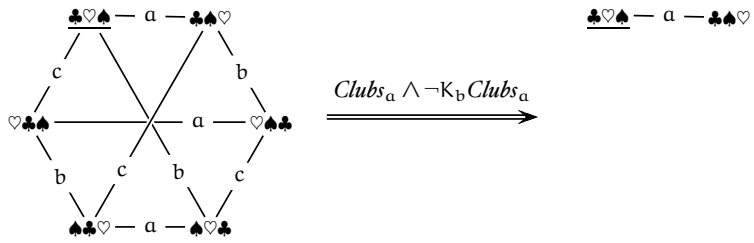


Figure 8: Anne says to Bill: "(*I hold clubs and*) *You don't know that I hold clubs*."

is *that* stupid—sometimes the expectation *is* warranted after all: the formulas that always become common knowledge after being announced, can be called *successful*. What are the possibilities?

After announcing φ, φ sometimes remains true and sometimes becomes false, and this depends both on the formula *and* on the epistemic state. Consider an epistemic state for one atom p and two agents, Anne and Bill again, where Anne knows the truth about p but Bill doesn't. This epistemic state is formally defined as $(\mathit{Letter}, 1)$, where the model *Letter* has domain $\{0, 1\}$, where p is true in state 1: $V_p = \{1\}$, and such that Anne can distinguish 1 from 0 but Bill cannot, so access $\sim_a$ for a is the identity $\{(0,0),(1,1)\}$ and access $\sim_b$ for b is the universal relation $\{(0,0),(1,1),(0,1),(1,0)\}$. The model is called *Letter* because it can be seen as the result of Bill seeing Anne read a *letter* which contains the truth about p. If in this epistemic state Anne says, truthfully: "I know that p," then after this announcement of $K_a p$ it *remains true* that $K_a p$:

$$\mathit{Letter}, 1 \models [K_a p] K_a p$$

This is, because in *Letter* the formula $K_a p$ is true in state 1 only, so that the model $\mathit{Letter}|K_a p$ consists of the singleton state 1, with reflexive access for a and b. It also becomes common knowledge that Anne knows p: we have that $\mathit{Letter}, 1 \models [K_a p] C_N K_a p$; although in this particular case of a singleton model, that is not very informative. We therefore also have $\mathit{Letter} \models K_a p \to [K_a p] C_N K_a p$ and $K_a p \to [K_a p] C_N K_a p$ is indeed valid.

But it is not always the case that announced formulas remain true. In the given epistemic state $(\mathit{Letter}, 1)$, Anne could on the other hand have said as well, to Bill: "You don't know that p." The actual implicature in this case is "Fact p is true and you don't know that." After this announcement of $K_a(p \wedge \neg K_b p)$, that also only succeeds in state 1, Bill knows that p, therefore $K_a(p \wedge \neg K_b p)$ is now *no longer* true

$$\mathit{Letter}, 1 \models [K_a(p \wedge \neg K_b p)] \neg K_a(p \wedge \neg K_b p)$$

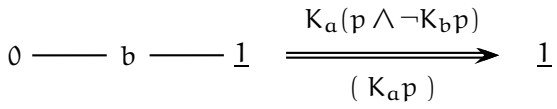


Figure 9: A simple unsuccessful update: (p and) you don't know that p. The annoucement 'I know that p'—between brackets—induces the same state transition.

and so it is certainly not commonly known.

$$\mathit{Letter}, 1 \models [\mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)]\neg \mathsf{C}_N \mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)$$

So $\mathsf{K}_a(p \wedge \neg \mathsf{K}_b p) \rightarrow [\mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)]\mathsf{C}_N \mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)$ is definitely not valid.

The epistemic state transition induced by this announcement is depicted in Figure 9. The announcement of $\mathsf{K}_a p$ induces the same state transition. Incidentally, like here, for a given state transition there is always a formula that induces it and *remains* true, an interesting result by van Benthem [vB02].

In this case, we not only have that $\mathsf{K}_a p$ remains true after being announced and that $\mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)$ becomes false, but also that $[\mathsf{K}_a p]\mathsf{K}_a p$ is valid, and $[\mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)]\neg \mathsf{K}_a(p \wedge \neg \mathsf{K}_b p)$ is valid. In between these extremes of 'always successful' and 'always unsuccessful' there are also formulas that sometimes remain true, and at other times—given other epistemic states—become false after an announcement. A typical example is 'not stepping forward' in the well-known Muddy Children problem [FHMV95]. The 'announcement' (implicitly, by not stepping forward) that none of the children know whether they are muddy, remains true in all epistemic states for this problem *except* the last one, in which it is an unsuccessful update: after that the muddy children know that they are muddy, and step forward. The following terminology describes all those nuances.

Definition 4 (success) A formula φ in the language of public announcements is *successful* if and only if $[\varphi]\varphi$ is valid. A formula is *unsuccessful* if and only if it is not successful. Given an epistemic state (M, s), φ is a *successful update* in (M, s), if and only if $M, s \models \langle\varphi\rangle\varphi$; and φ is an *unsuccessful update* in (M, s), if and only if $M, s \models \langle\varphi\rangle\neg\varphi$.

In the definitions, the switch between the 'box' and the 'diamond' versions of the announcement operator may puzzle the reader. In the definition of a successful *formula* we need the 'box'-form: $\langle\varphi\rangle\varphi$ is invalid for all φ except $\top$ ($\top$ stands for 'verum', 'truth'). But in the definition of a successful *update* we need the 'diamond'-form: otherwise, whenever the announcement formula is false in an epistemic state, $[\varphi]\neg\varphi$ would therefore be true, and we would be forced to call that φ an unsuccessful update. That would not capture the intuititive meaning of 'unsuccessful update', which is a property of an epistemic state transition. We must therefore assume that the announcement formula can

indeed be truthfully announced. This explains the difference between the two definitions.

Announcements of (therefore true) successful formulas (the validity of $[\bot]\bot$ is considered atypical) are always successful updates, but sometimes successful updates are on formulas that are unsuccessful. The first will be obvious: if a successful *formula* φ is true in an epistemic state (M, s), then $\langle\varphi\rangle\varphi$ is also true in that state, so it is also a successful *update*. The last is less obvious: formulas may be successful updates in one epistemic state, but unsuccessful updates in another, and from the latter follows that they are unsuccessful *formulas* [vDK05].

We can link our intuitions about 'success' to the definition of a successful formula in a surprisingly elegant way: A formula $[\varphi]\varphi$ is valid, if and only if $[\varphi]C_N\varphi$ is valid, if and only if $\varphi \to [\varphi]C_N\varphi$ is valid. So the successful formulas 'do what we want them to do': if true, they become common knowledge when announced. What formulas are successful? An answer to this question is not obvious, because some inductive ways to construct the class of successful formulas fail: even if φ and ψ are successful, $\neg\varphi$, $\varphi \wedge \psi$, or $\varphi \to \psi$ may be unsuccessful. For example, both p and $\neg Kp$ are successful formulas, but, as we have seen, $p \wedge \neg Kp$ is not. A partial answer to that question and further information on unsuccessful updates, including examples, can be found in [vDK05].

7 EPISTEMIC ACTIONS

Some epistemic actions are more complex than public announcements, where the effect of the action is always a restriction on the epistemic model. Let us reconsider the epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$ for three players Anne, Bill and Cath, each holding one of clubs, hearts, and spades; and wherein Anne holds clubs, Bill holds hearts, and Cath holds spades. And consider again one of the example actions in the introduction:

> *Anne shows (only) to Bill her clubs card. Cath cannot see the face of the shown card, but notices that a card is being shown.*

As always in this epistemic (and not doxastic) setting, it is assumed that it is publicly known what the players can and cannot see or hear. Call this action showclubs. The epistemic state transition induced by this action is depicted in Figure 10. Unlike after public announcements, in the showclubs action we cannot eliminate any state. Instead, all b-links between states have now been severed: whatever the actual deal of cards, Bill will know that card deal and cannot imagine any alternatives. Let us show the intuitive acceptability of the resulting epistemic state. After the action showclubs, Anne can imagine that Cath can imagine that Anne has clubs. That much is obvious, as Anne has

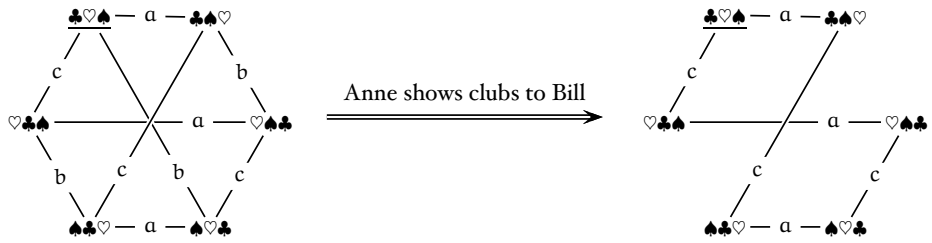


Figure 10: On the left, the Kripke model for three players each holding one card. On the right, the effect of Anne showing her clubs card to Bill.

clubs anyway. But Anne can also imagine that Cath can imagine that Anne has hearts, because Anne can imagine Cath to have spades, and so not to know whether Anne has shown clubs or hearts; so it might have been hearts. It is even the case that Anne can imagine that Cath can imagine that Anne has spades, because Anne can imagine Cath not to have spades but hearts instead, in which case Cath would not have known whether Anne has shown clubs or spades; so it might have been spades. Note that, even though for Cath there are only two 'possible actions'—showing clubs or showing hearts—none of the *three* possible actions can apparently be eliminated 'from public consideration'. The descriptions of the action showclubs and of the other 'possible actions', where Anne shows hearts or spades to Bill instead, should obviously be related: in Figure 10, this merely means shifting the point from one state to another.

But it can become even more complex. Imagine the following action, rather similar to the showclubs action:

> *Anne whispers into Bill's ear that she does not have the spades card, given a (public) request from Bill to whisper into his ear one of the cards that she does not have.*

This is the action whispernospades. Given that Anne has clubs, she *could* have whispered "no hearts" or "no spades". And whatever the actual card deal was, she could always have chosen between two such options. We expect an epistemic state to result that reflects that choice, and that therefore consists of $6 \times 2 = 12$ different states. It is depicted in Figure 11. The reader may ascertain that the desirable postconditions of the action whispernospades indeed hold. For example, given that Bill holds hearts, Bill will now have learnt from Anne what Anne's card is, and thus the entire deal of cards. So there should be no alternatives for Bill in the actual state (the underlined state ♣♡♠ 'at the back' of the figure—for convenience, different states for the same card deal have been given the same name). But Cath does not *know* that Bill knows the card deal, as Cath can imagine that Anne actually whispered "no hearts" instead.

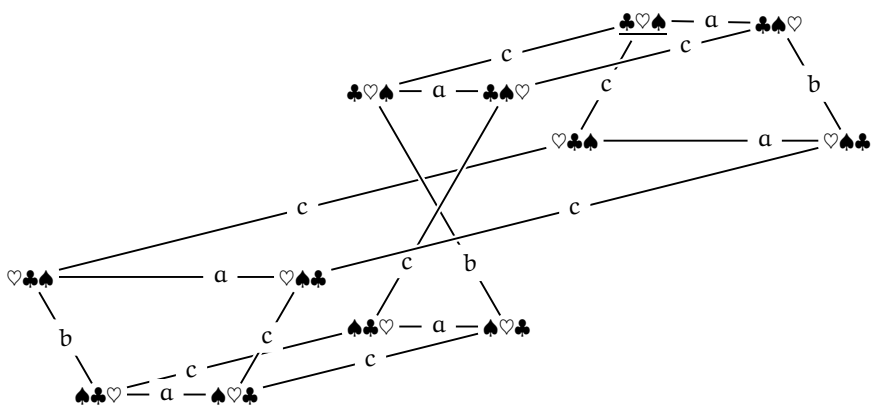


Figure 11: Anne whispers into Bill's ear that she does not have the spades card, given a (public) request from Bill to whisper into his ear one of the cards that she does not have.

That would have been something that Bill already knew, as he holds hearts himself—so from that action he would not have learnt very much. Except that Cath could then have imagined him to know the card deal... Note that in Figure 11 there is also another state named ♣♡♠, 'in the middle', so to speak, that is accessible for Cath from the state ♣♡♠ 'at the back', and that satisfies that Bill doesn't know that Anne has clubs.

From the point of view of dynamic epistemics, a public announcement is a simple form of epistemic action: it results in a restriction of the domain. The showhearts action results in a refinement of accessibility relations given the same domain. The whispernospades action results in increased complexity of the underlying epistemic model, reflecting non-deterministic choice. To be able to model such actions a generalization of the approach used in the public announcement logic of [Pla89] was needed. Plaza's work was more fully appreciated in the latter half of the 1990s, when subsequent, partially independent, developments took place. A stream of publications appeared around the year 2000 [GG97, Ger99, LR99b, BMS98, Bal99, vD00, vB01, BM04, vD02, tC02, K0003, vDvdHK03]. Gerbrandy was unfamiliar with the work of Plaza at the time of his seminal publication [GG97]. It models the dynamics of introspective agents, and therefore in particular changes in belief (and, as a special case, knowledge). Its basis is a different insight into dynamics, namely along the lines of work in dynamic semantics by [Vel96]. The approach in [vD00, vD02, vDvdHK03] might be called a relational action language, wherein epistemic states resulting from computing the effects of actions for subgroups (such as 'Anne and Bill' in the case of three cards) are used in the computations of the effects of the action for larger groups that contain

that smaller group, and finally, the effects of the action for the public (such as 'Anne, Bill, and Cath'). A different approach, and a conceptually very appealing solution, is to see a semantic action as some kind of Kripke model, an 'action model' so to speak, and action execution as a restricted modal product ('the next epistemic state') of the current epistemic state and the action model. This was first presented in [BMS98, Bal99] and its semantics recently appeared in final version in [BM04].

A crucial concept in the [vDvdHK03] approach is the 'learn' operator. This is a dynamic variant of the 'common knowledge' operator. Let's see what it means, by paraphrasing the action showclubs in a way that brings this action closer to its description as an epistemic action.

> *Anne and Bill learn that Anne holds clubs, whereas Anne, Bill and Cath learn* [that either Anne and Bill learn that Anne holds clubs, or that Anne and Bill learn that Anne holds hearts, or that Anne and Bill learn that Anne holds spades].

In other words: Anne, Bill and Cath learn that Anne and Bill learn which card Anne holds, and, actually, Anne and Bill learn that Anne holds clubs. The choice made from the three alternatives by subgroup $\{a, b\}$ is known to them only, and is hidden from c, who only knows what the three alternatives are. The description of this action in the relational action approach is

$$L_{abc}(!L_{ab}?\mathit{Clubs}_a \cup L_{ab}?\mathit{Hearts}_a \cup L_{ab}?\mathit{Spades}_a)$$

In this description, 'L' stands for 'learning', the '!' indicates which of the three alternatives really happens, '$\cup$' stands for non-deterministic choice, and '?' stands for 'a test on' (the truth of the formula following it). The whispernospades action is described as

$$L_{abc}(L_{ab}?\neg\mathit{Clubs}_a \cup L_{ab}?\neg\mathit{Hearts}_a \cup !L_{ab}?\neg\mathit{Spades}_a)$$

Note that in this case the first option could not have been chosen, and that instead the third option has been chosen. To explain this in reasonable detail, or any of the other approaches, is beyond this introduction. For details see the references.

Some rather simple actions cannot be modelled in any of the current dynamic epistemic approaches. For example, given that the action descriptions in all mentioned approaches are entirely based on the properties of the current epistemic state, one cannot distinguish between different ways in which that current state came about. Anne may *only* want to show a clubs card if some past action of Bill involved showing a spades card. But the action descriptions cannot distinguish between epistemic states that have the same (bisimilar) epistemic description but different action histories! In view of modelling game

strategies, such expanded expressive power is of course essential. For another example, given the scenario where Anne receives a letter and Bill sees her reading it, suppose that the letter did not contain the truth about a single fact but contained a natural number. So instead of one fact we have infinitely many facts. Before she reads the letter, the epistemic model for that consists of infinitely many points, with universal access for both Anne and Bill, no problem at all. It is also clear what the model looks like after Anne reads the letter: Anne's access is now the identity, and Bill's is still the universal relation. But the action describing that Anne reads the letter, which transforms the former into the latter, has an infinitely long description, because there are infinitely many alternatives: a problem.

8 BELIEF, TIME, REVISION

This section presents different perspectives and other approaches. Instead of knowledge change we may want to model belief change; knowledge change can also be seen as emerging from the temporal progression of some epistemic state, using temporal and epistemic operators instead; we can see knowledge change as some kind of (deductively closed) *theory* change: a matter that has been thoroughly investigated under the header of 'belief revision'; and there are logics that combine knowledge and belief, and degrees of belief, and probability, and changes to some or all of those.

BELIEF: We discussed knowledge change only and not belief change—with 'knowledge as true belief'. This was just for expository purposes. Belief change can be modelled in the same way. With the exception of the approach originating in Van Ditmarsch PhD. thesis [vD00], that so far only applies to knowledge, all mentioned approaches for dynamic epistemics only assume arbitrary accessibility relations. They therefore apply as well to structures that satisfy the properties of belief. A typical sort of epistemic action that can only be modelled in this setting is the *private announcement to a subgroup only*: Suppose that in epistemic state $(\mathit{Hexa}, \clubsuit\heartsuit\spadesuit)$, Anne shows Bill her clubs card, as before, but now *without* Cath noticing anything at all. In the state resulting from that action, Bill knows the card deal, as before, but Cath incorrectly believes that Bill does not know that. Such private announcements to groups are the main topic of Gerbrandy's PhD. thesis [Ger99].

TIME: In temporal epistemic approaches we may express the information that Bill knows that Anne holds clubs after she said that she does not have spades, as, for example, $\mathsf{X}\mathsf{K}_b \mathit{Clubs}_a$, or $\mathsf{K}^1_b \mathit{Clubs}_a$. We then assume an underlying structure of the corresponding epistemic state transitions, for example corresponding to some such transitions in a run of an interpreted system. We cannot express the content of the action in the temporal operator. In $\mathsf{X}\mathsf{K}_b \mathit{Clubs}_a$, X is the (modal) temporal 'next' operator, which is interpreted as follows '$\mathsf{X}\mathsf{K}_b \mathit{Clubs}_a$ is

true in the current state, if in the next state (as determined by the underlying structure) $\mathsf{K}_b \mathit{Clubs}_a$ is true. In $\mathsf{K}^1_b \mathit{Clubs}_a$ we do something similar, only that in this case K^1_b is the operator describing what Bill knows at point 1 in time. Temporal epistemic logics have been fairly successful. Their computational properties are well-known and proof tools have been developed. See for example—we give just some arbitrary references here—[vdM98, DFW98, HvdMV04]. The main difference with the dynamic epistemic approach is that the temporal epistemic description takes as models systems consisting of many epistemic states together with their whole (deterministic) history and future development. Instead, in dynamic epistemics a single epistemic state—a point in that temporal structure so to speak—is sufficient: its further development is induced by the description of the action to be executed there. This may be seen as an advantage of the dynamic epistemic approach. But there are also definite advantages to the temporal epistemic approach. Consider again the Moore-sentences. After Anne announces to Bill: "(I hold clubs and) You do not know that I hold clubs," there is nothing inconsistent in the truth of $\mathsf{K}^1_b(\mathit{Clubs}_a \wedge \neg \mathsf{K}^0_b \mathit{Clubs})$: at point 1 in time, Bill knows that Anne holds clubs and that at point 0 in time he *did* not know that Anne holds clubs.

BELIEF REVISION: In belief revision the emphasis is on theories of objective (i.e., non-epistemic) beliefs that are changed due to expansions, contractions, or revisions, typically from the point of view of a single agent. Let's consider the point of view of Bill in 'three cards'. In this case his 'beliefs' are his justified true beliefs: his knowledge. At the outset he knows that he holds hearts, but he does not know the ownership of other cards. Therefore we may assume that Hearts_b is part of his set of current beliefs T. General descriptions are also part of that theory T of current beliefs, for example rules expressing that a card can only be held by a single player: exclusive disjunction of Spades_a, Spades_b, and Spades_c; and sentences describing single card ownership: $\mathit{Hearts}_b \rightarrow \neg \mathit{Spades}_b$, ...; etc. Suppose the new information is 'Anne does not hold spades'. As Bill's current beliefs were consistent with both Spades_a and $\neg \mathit{Spades}_a$, the belief change taking place here is an *expansion* and not a revision. The revised theory $\mathsf{T}+\neg \mathit{Spades}_a$ should contain the 'new information' $\neg \mathit{Spades}_a$, and we also expect Bill to be able to derive Clubs_a from that.

A general framework to describe such belief expansion in an epistemic setting, and also contractions and revisions, is given in [Seg99b]. See also [Seg99a, LR99a]. As far as the logical language is concerned, this follows more or less the following pattern:

For the example just given, Bill's beliefs φ are described by all $\mathsf{K}_b\varphi$ that are true in the current epistemic state. That Hearts_b is part of his beliefs corresponds to the truth of $\mathsf{K}_b \mathit{Hearts}_b$. That both Clubs_a and $\neg \mathit{Clubs}_a$ are absent from his beliefs, corresponds to the truth of both $\neg \mathsf{K}_b \mathit{Clubs}_a$ and $\neg \mathsf{K}_b \neg \mathit{Clubs}_a$ in the current state of information, before Anne's announcement. And that Clubs_a is believed by Bill after the announcement, is described by the truth of

$\mathsf{K}_b \mathit{Clubs}_a$ in the resulting epistemic state. The expansion with $\neg \mathit{Spades}_a$ corresponds to Anne's public announcement of $\neg \mathit{Spades}_a$, after which $\mathsf{K}_b \neg \mathit{Spades}_a$ is indeed true.

A major difference between belief revision and dynamic epistemics is that the latter, and not the former, allows higher-order belief change. In 'three cards' we have that from Anne's announcement that she does not have spades, Cath does not gain any factual knowledge, but learns that Bill now knows Anne's card. So the revision of Cath's beliefs should involve adding a non-objective formula $\mathsf{K}_b \mathit{Clubs}_a \vee \mathsf{K}_b \mathit{Hearts}_a \vee \mathsf{K}_b \mathit{Spades}_a$, because in the new epistemic state it is true that $\mathsf{K}_c(\mathsf{K}_b \mathit{Clubs}_a \vee \mathsf{K}_b \mathit{Hearts}_a \vee \mathsf{K}_b \mathit{Spades}_a)$. This general issue of updating 'non-objective' formulas was neglected by classical belief revision theory, partly because of complications in the form of 'Moore'-problems. An expansion with "(I hold clubs and) You do not know that I hold clubs," can never be successful; and 'success' happens to be a deeply entrenched postulate for acceptable theory revision. It was unclear how the standard AGM postulates should be generalized to include such cases.

A second important difference between dynamic epistemics and belief revision concerns not expansion but actual 'revision' of (possibly wrong) beliefs, i.e. updating with a formula that is inconsistent with prior beliefs. This is typically analyzed in depth by belief revision, but neglected by dynamic epistemics. Recent advances in that have been made in [vDL03, Auc03], motivated to an important extent by seminal work from Spohn [Spo88].

It suffices to give a simple example of where this comes in handy. Consider, once again, but for the last time now, the action showclubs wherein Anne shows clubs to Bill only, with Cath noticing that. Now imagine that Cath considers it more likely that Anne shows hearts than that Anne shows clubs. And assume that Cath's beliefs—as is common within a 'belief revision' setting—are determined by the things she considers most normal / most likely. With each agent we can associate a whole set of operators for all of belief, and different degrees of belief, and knowledge, and interpret these on 'doxastic epistemic' models, that carry a set of accessibility relations per agent. In the resulting state of information we can describe that: even though Bill knows that Anne holds clubs—$\mathsf{K}_b \mathit{Clubs}_a$—Cath believes that Bill knows that Anne holds hearts—$\mathsf{B}_c \mathsf{K}_b \mathit{Hearts}_a$. Further actions, for example Anne putting her clubs card face up on the table, then result in Cath retracting her belief in $\mathsf{K}_b \mathit{Hearts}_a$ and 'expanding' her beliefs with $\mathsf{K}_b \mathit{Clubs}_a$ instead, so we then end up with $\mathsf{B}_c \mathsf{K}_b \mathit{Clubs}_a$ again. For details, see [vD05, Auc03]. These approaches—they may incorporate infinitely many degrees of belief—also suggest overlap with approaches combining knowledge and probability [FH94, Hal03], and the dynamics of that [Koo03].

References

[AGM85] C.E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[Auc03] G. Aucher. A combined system for update logic and belief revision. Master's thesis, ILLC, University of Amsterdam, Amsterdam, the Netherlands, 2003.

[Bal99] A. Baltag. A logic of epistemic actions. In W. van der Hoek, J.-J. Meyer, and C. Witteveen, editors, *(Electronic) Proceedings of the ESSLLI 1999 workshop on Foundations and Applications of Collective Agent-Based Systems*. Utrecht University, 1999.

[BM04] A. Baltag and L.S. Moss. Logics for epistemic programs. *Knowledge, Rationality, & Action (Synthese)*, 139:165–224, 2004. 1–60.

[BMS98] A. Baltag, L.S. Moss, and S. Solecki. The logic of common knowledge, public announcements, and private suspicions. In I. Gilboa, editor, *Proceedings of the 7th conference on theoretical aspects of rationality and knowledge (TARK 98)*, pages 43–56, 1998.

[DFW98] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for temporal logics of knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[FH94] R. Fagin and J.Y. Halpern. Reasoning about knowledge and probability. *Journal of the Association for Computing Machinery*, 41(2):340–367, 1994.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[Ger99] J.D. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.

[GG97] J.D. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language, and Information*, 6:147–169, 1997.

[Hal03] J.Y. Halpern. *Reasoning about Uncertainty*. MIT Press, Cambridge MA, 2003.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.

[HvdMV04] J.Y. Halpern, R. van der Meyden, and M.Y. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33(3):674–703, 2004.

[Koo03] B.P. Kooi. *Knowledge, Chance, and Change*. PhD thesis, University of Groningen, 2003. ILLC Dissertation Series DS-2003-01.

[LR99a] S. Lindström and W. Rabinowicz. DDL unlimited: dynamic doxastic logic for introspective agents. *Erkenntnis*, 50:353–385, 1999.

[LR99b] A.R. Lomuscio and M. D. Ryan. An algorithmic approach to knowledge evolution. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(2), 1999. Special issue on Temporal Logic in Engineering.

[MvdH95] J.-J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science. Cambridge Tracts in Theoretical Computer Science 41*. Cambridge University Press, Cambridge, 1995.

[Pla89] J.A. Plaza. Logics of public communications. In M.L. Emrich, M.S. Pfeifer, M. Hadzikadic, and Z.W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[Seg99a] K. Segerberg. Default logic as dynamic doxastic logic. *Erkenntnis*, 50:333–352, 1999.

[Seg99b] K. Segerberg. Two traditions in the logic of belief: bringing them together. In H.J. Ohlbach and U. Reyle, editors, *Logic, Language, and Reasoning*, pages 135–147, Dordrecht, 1999. Kluwer Academic Publishers.

[Spo88] W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states. In W.L. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, volume II, pages 105–134, 1988.

[tC02] B. ten Cate. Internalizing epistemic actions. In Maricarmen Martinez, editor, *Proceedings of the NASSLLI-2002 student session*, Stanford University, 2002.

[vB01] J.F.A.K. van Benthem. Logics for information update. In J.F.A.K. van Benthem, editor, *Proceedings of TARK VIII*, pages 51–88, Los Altos, 2001. Morgan Kaufmann.

[vB02] J.F.A.K. van Benthem. One is a lonely number: on the logic of communication. Technical report, ILLC, University of Amsterdam, 2002. Report PP-2002-27 (material presented at the Logic Colloquium 2002).

[vD00] H.P. van Ditmarsch. *Knowledge games*. PhD thesis, University of Groningen, 2000. ILLC Dissertation Series DS-2000-06.

[vD02] H.P. van Ditmarsch. Descriptions of game actions. *Journal of Logic, Language and Information*, 11:349–365, 2002.

[vD05] H.P. van Ditmarsch. Prolegomena to dynamic logic for belief revision. *Knowledge, Rationality, & Action (Synthese)*, 2005. To appear.

[vDK05] H.P. van Ditmarsch and B.P. Kooi. The secret of my success. *Synthese*, 2005. To appear.

[vDL03] H.P. van Ditmarsch and W.A. Labuschagne. A multimodal language for revising defeasible beliefs. In E. Álvarez, R. Bosch, and L. Villamil, editors, *Proceedings of the 12th International Congress of Logic, Methodology, and Philosophy of Science (LMPS)*, pages 140–141. Oviedo University Press, 2003.

[vdM98] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[vDvdHK03] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Concurrent dynamic epistemic logic. In V.F. Hendricks, K.F. Jørgensen, and S.A. Pedersen, editors, *Knowledge Contributors*, pages 45–82, Dordrecht, 2003. Kluwer Academic Publishers. Synthese Library Volume 322.

[vDvdHK06] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Dynamic epistemic logic. Graduate textbook, scheduled to appear, 2006.

[Vel96] F. Veltman. Defaults in update semantics. *Journal of Philosophical Logic*, 25:221–261, 1996.

The *Australasian Journal of Logic* (ISSN 1448-5052) disseminates articles that significantly advance the study of logic, in its mathematical, philosophical or computational guises. The scope of the journal includes all areas of logic, both pure and applied to topics in philosophy, mathematics, computation, linguistics and the other sciences.

Articles appearing in the journal have been carefully and critically refereed under the responsibility of members of the Editorial Board. Only papers judged to be both significant and excellent are accepted for publication.

The journal is freely available at the journal website at

http://www.philosophy.unimelb.edu.au/ajl/.

All issues of the journal are archived electronically at the journal website.

SUBSCRIPTIONS Individuals may subscribe to the journal by sending an email, including a full name, an institutional affiliation and an email address to the managing editor at ajl-editors@unimelb.edu.au. Subscribers will receive email abstracts of accepted papers to an address of their choice. For institutional subscription, please email the managing editor at ajl-editors@unimelb.edu.au.

Complete published papers may be downloaded at the journal's website at http://www.philosophy.unimelb.edu.au/ajl/. The journal currently publishes in pdf format.

SUBMISSION The journal accepts submissions of papers electronically. To submit an article for publication, send the LaTeX source of a submission to a member of the editorial board. For a current list of the editorial board, consult the website.

Logics of Communication and Change

Johan van Benthem [a] Jan van Eijck [b] Barteld Kooi [c]

[a] *ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands & Philosophy Department, Stanford University*

[b] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands & Uil-OTS, Trans 10, 3512 JK, Utrecht, The Netherlands*

[c] *Department of Philosophy, University of Groningen, Oude Boteringestraat 52, 9712 GL, Groningen, The Netherlands*

Abstract

Current dynamic epistemic logics for analyzing effects of informational events often become cumbersome and opaque when common knowledge is added for groups of agents. Still, postconditions involving common knowledge are essential to successful multi-agent communication. We propose new systems that extend the epistemic base language with a new notion of 'relativized common knowledge', in such a way that the resulting full dynamic logic of information flow allows for a compositional analysis of all epistemic postconditions via perspicuous 'reduction axioms'. We also show how such systems can deal with factual alteration, rather than just information change, making them cover a much wider range of realistic events. After a warm-up stage of analyzing logics for public announcements, our main technical results are expressivity and completeness theorems for a much richer logic that we call LCC. This is a dynamic epistemic logic whose static base is propositional dynamic logic (PDL), interpreted epistemically. This system is capable of expressing all model-shifting operations with finite action models, while providing a compositional analysis for a wide range of informational events. This makes LCC a serious candidate for a standard in dynamic epistemic logic, as we illustrate by analyzing some complex communication scenarios, including sending successive emails with both 'cc' and 'bcc' lines, and other private announcements to subgroups. Our proofs involve standard modal techniques, combined with a new application of Kleene's Theorem on finite automata, as well as new Ehrenfeucht games of model comparison.

Key words: epistemic logic, update, dynamic logic, common knowledge, reduction axioms, product update, finite automata, Kleene's Theorem

Email addresses: johan@science.uva.nl (Johan van Benthem), jve@cwi.nl (Jan van Eijck), B.P.Kooi@rug.nl (Barteld Kooi).

1 Introduction

Epistemic logic deals with what agents consider possible given their current information. This includes knowledge about facts, but also *higher-order information* about information that other agents have. A prime example is *common knowledge*. A formula φ is common knowledge if everybody knows φ, everybody knows that everybody knows that φ, and so on. *Common belief* is an important related notion. Indeed, although this paper is mainly written in 'knowledge' terminology, everything we say also holds, with minor technical modifications, when describing agents' *beliefs*, including common belief.

Dynamic epistemic logics analyze changes in both basic and higher-order information. One of the main attractions of such systems is their transparent analysis of effects of communicative actions in the format of an equivalence between epistemic postconditions and preconditions. A typical example concerns knowledge of an agent after and before a public announcement:

$$[\varphi]\Box_a\psi \leftrightarrow (\varphi \rightarrow \Box_a[\varphi]\psi).$$

This axiom says that after the announcement that φ agent a knows that ψ iff φ implies that agent a knows that after φ is announced ψ will be true. We call such principles *reduction axioms*, because the announcement operator is 'pushed through' the epistemic operator, in such manner that on the right hand side the complexity of the formula in the scope of the announcement is less that the complexity of the formula in the scope of the announcement on the left hand side. This reduction axiom describes the interaction between the announcement operator and the epistemic operator. If there is a reduction axiom for each logical operator in the language, such a set of axioms make logical systems particularly straightforward. For instance, the logic of public announcements without common knowledge has an easy completeness proof by way of a translation that follows the reduction axioms. Formulas with announcements are translated to provably equivalent ones without announcements, and completeness follows from the known completeness of the epistemic base logic. Thus, the dynamic logic of the announcement operator is fully characterized by the reduction axioms.

This is the technical way of putting things. But more importantly, reduction axioms like the one above also reflect a desirable *methodology*: they allow for *compositional analysis* of the epistemic effects of informational events. This is particularly helpful with more complex scenarios, where it is not at all easy to describe just what agents should know, or not, after some communication has taken place: say, a round of emails involving both public 'cc' and half-private 'bcc' lines. A dynamic epistemic logic with a complete set of reduction axioms has an ideal 'harmony' between its static and dynamic parts allowing for complete compositional analysis. So, it is worth finding such systems whenever

they exist. Finally, more locally, specific reduction axioms also express interesting assumptions about the interplay of events and knowledge. E.g., it has often been observed that the above one for public announcement embodies a form of 'Perfect Recall': the event of announcement does not add or delete uncertainty lines for agents among those worlds which they consider possible.

In this light, there is a problem with common knowledge for *groups* in societies of communicating agents. Understanding group knowledge as it is gained or lost, is at the heart of analyzing epistemic update. But existing dynamic epistemic logics have no compositional reduction axioms for achieving common knowledge, and this infelicity has been there from the start. Now, the seminal paper [1] does treat common knowledge per se, but not on a reductive pattern, and its completeness proof is correspondingly messy. Indeed, reduction axioms are not available, as the logic with epistemic updates is more expressive than the logic without them. We think this is an infelicity of design, and our main aim in this paper is to show how compositional analysis is feasible by some judicious language extension, restoring the proper harmony between the static and dynamic features of the system.

In Section 2 we first look at examples of the general kinds of information change that we are interested in. These include public announcement, but also communication involving privacy and partial observation, and indeed, observation of any sort of event that carries information. We also include real physical actions changing the world. Before we give a system that deals with all these phenomena, we first look at a pilot case that illustrates many issues in a simpler setting, the logic PAL of *public announcements*. Section 3 gives a new and complete set of reduction axioms for public announcement logic with common knowledge, obtained by strengthening the base language with an operator of relativized common knowledge, as first proposed in [3]. Moreover, since languages with model-shifting operators like $[\varphi]$ are of independent logical interest, we develop the model theory of PAL a bit further, using new game techniques for epistemic languages with fixed-point operators for common knowledge to investigate its expressive power. Section 4, the heart of this paper, then proposes a new dynamic epistemic LCC dealing with the general case of updating with finite structures of events, generalizing the standard reference [1] to include a much wider range of epistemic assertions, as well as factual change. What the section demonstrates, in particular, is that PDL (the well-known system of propositional dynamic logic), when interpreted epistemically, can serve as a basis for a rich and expressive logic of communication that allows for smooth compositional analysis of common knowledge after epistemic updates. To avoid confusion with PDL in its non-epistemic uses for analyzing actions, we will call our version here LCC.

A general approach that reduces dynamic epistemic logic to propositional dynamic logic was first proposed using finite automata techniques in [18], using a

variant of propositional dynamic logic called 'automata PDL'. The new techniques used in the present paper (cf. [11]) work directly in epistemic PDL, using the idea behind Kleene's Theorem for regular languages and finite automata to find the relevant reduction axioms inductively by means of 'program transformations'. This analysis does not just yield the meta-theorems of completeness and decidability that we are after. It can also be used in practice to actually compute valid axioms analyzing common knowledge following specific communicative or informational events. Section 5 analyzes some of our earlier communication types in just this fashion, obviating the need for earlier laborious calculations 'by hand' (cf. [27]). Finally, Section 6 draws conclusions, and indicates directions for further research.

The broader context for this paper are earlier systems of dynamic epistemic logic, with [26], [14], and [1] as key examples of progressively stronger systems, while [8] is a source of inspiring examples. Reduction axioms were already used to prove completeness for dynamic epistemic logics in [14] and [1]. Another major influence is the work of [13] on common knowledge in computational settings, using a more general temporal-epistemic framework allowing also for global protocol information about communicative processes. Connections between the two approaches are found, e.g., in [21]. Further references to the literature on epistemic actions and to many challenging open problems in the general landscape of update logics can be found in [5].

Even though the main thrust of this paper may seem technical, our proposal is much more than just a trick for smoothing completeness proofs, or for finding a new model-theoretic play-ground. It also addresses a significant design issue of independent interest: what is the most convenient and transparent epistemic language for describing information flow for groups of agents in a compositional manner? Our main logic LCC in Section 4 is meant as a serious proposal for a standard.

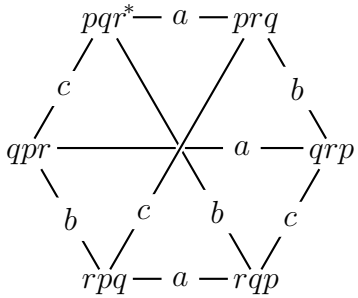
2 Modelling Effects of Communication and Change

Epistemic update logics are about the effects of general communication, and indeed, they describe the logic of observing any kind of information-bearing event. But in practice, it is helpful to look at more constrained scenarios. A good source of examples are basic actions in *card games*. Game moves then involve looking at a card, showing a card to someone (with or without other players looking on), exchanging cards with someone (with or without other players looking on), and perhaps even changing the setting in more drastic ways (cf. [8]). This is not just a frivolous move toward parlour games. One can think of 'card events' as a sort of normal form for any type of informational activity — and one which has the additional virtue of evoking vivid intuitions.

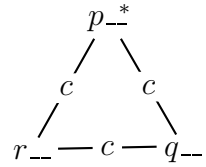
Moreover, scenarios involving the interplay of information and ignorance are not just logician's puzzles for their own sake: managing the right mixtures of information and ignorance is absolutely essential to human intelligence, and to the functioning of civilized societies.

In this section we list some examples involving combinations of epistemic and actual change that we think any full-fledged dynamic-epistemic logic should be able to deal with. The simplest scenario here is public announcement of some fact P, which merely requires elimination of all worlds in the current model where P does not hold. But general communication can be much more complex — just think of whispers in a lecture theatre. This requires updates of the initial information model beyond mere elimination of worlds. Some updates even make the current model bigger, as alternatives can multiply. Since all these actions involve groups of agents, understanding both individual and common knowledge following certain events is clearly essential.

Card Showing A simple card showing situation goes as follows. Alice, Bob and Carol each hold one of the cards p q, r. The actual deal is: Alice holds p, Bob holds q, Carol holds r. Assuming all players looked at their own cards, but have kept them hidden from the others, this situation is modelled as follows (xyz represents the situation where Alice holds x, Bob holds y and Carol holds z, xyz—a—$x'y'z'$ represents the fact that Alice cannot distinguish xyz from $x'y'z'$, and xyz^* indicates that xyz is the situation that actually is the case):

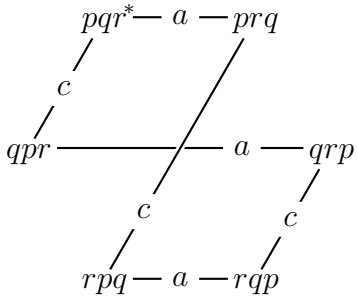


Now assume Alice shows her card p to the whole group. This public event eliminates all worlds from the initial model that conflict with the new information. Thus, out of the six given worlds only two remain: pqr and prq. In the resulting model, Bob and Carol know all the cards, while Alice only knows that she has p — and both these facts are common knowledge. Now consider a 'semi-public' action of Alice showing her card to Bob, with Carol looking on (Carol sees that a card is shown, but does not see which card). Here is a major new idea, due to [1]. We first picture the new event itself as an *update model* whose structure is similar to that of epistemic models in general:

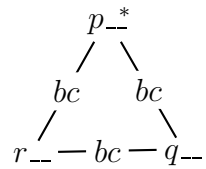


According to this picture, the card that Alice shows to Bob is card p, but for all Carol knows, it might have been q or r. In actual fact it cannot be r, as that is the card which Carol holds and has just inspected, but this information is not part of the update action. Note that the events depicted cannot occur in just any world. Alice can only show card p when she actually holds p, she can show q when she actually holds q and she can show r when she actually holds r. The latter information is encoded in so-called *preconditions*, and indeed, the fundamental reason why occurrences of events carry information for us is that we know their preconditions.

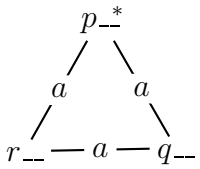
Now for the update from the preceding event. Intuitively, we want a new information model arising from the initial one that the agents were in plus the update model containing the relevant actions. The new worlds are then old worlds plus the most recent event attached. Moreover, our intuitions tell us what the desired result of this update should look like, viz.:



Card Inspection Next, consider acts of observation. Suppose the three cards are dealt to Alice, Bob and Carol, but are still face down on the table. The following picture describes an update model for Alice's inspecting her own card and discovering it to be p, with the others just looking on.

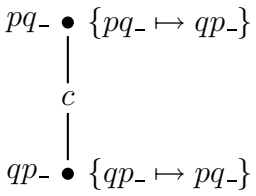


And here is the related update model of Alice picking up her card and showing it to the others, without taking a look herself:

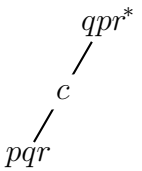


In all these cases we have clear intuitions about what the outcomes of the updates should be, and these underlie the technical proposals made in Section 4.

Card Exchange Next, we add a case where real physical action takes place, which is not purely informational. Suppose in the initial situation, where Alice holds p, Bob holds q and Carol holds r, Alice and Bob exchange cards, without showing the cards to Carol. To model this, we need an update that also changes the state of the world. For that, we need to change the valuation for atomic facts. This may be done by using 'substitutions' which reset the truth values of those atomic statements that are affected by the action:



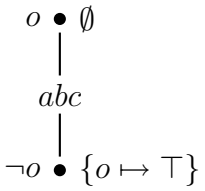
Note that the diagram now indicates both the earlier *preconditions* for events or actions, and *postconditions* for their successful execution. Here is the result of applying this update model in the initial situation, where all players have looked at their cards and the actual deal is pqr:



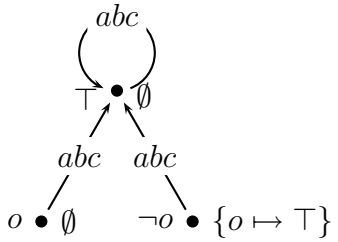
So far, we have looked at card examples, where actions are meant to be communicative, with some intentional agent. But it is important to realize that dynamic epistemic logic can also serve as a system for analyzing arbitrary *observations* of events that carry some information to observers, whether intended or not. That is, it is a logic of perception as much as of communication, and it is useful for modelling the essence of what goes on in very common everyday actions. We conclude with two illustrations in the latter mode.

Opening a window The precondition for opening a window is that the window is closed. To make this into an update that can always be performed,

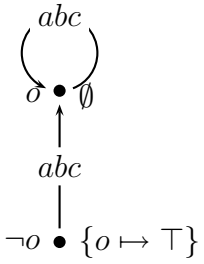
we specify different actions depending on the state of the window. If it is open, nothing needs to be done; if it is closed, then open it. This much is standard dynamic logic, as used in describing transition systems in computer science. But now we are in a setting where agents may have different 'epistemic access' to what is taking place. E.g., assuming the relevant event is invisible to all (Alice, Bob and Carol), it can be modelled as follows:



Again, we see both pre- and postconditions, with the latter depending on the former. Note that this action can make agents 'out of touch with reality', perhaps through laziness in observation. Such a mismatch can also result from being actively misled. If the window is opened in secret, its update model looks as follows (o denotes an open window; $o \mapsto \top$ is the substitution making o true).



Further variations on this update model are possible. E.g., the window is in fact opened, while everyone is told that it was already open. Here is the corresponding update:



Fiddling with a light switch Fiddling with a light switch is an update that depends on the actual situation as follows: if the light is on, then switch it off, if it is off, then switch it on. If this fiddling is done in a way such that the result is visible to all, then here is its update model:

$$\top \bullet \{o \mapsto \neg o\}$$

If the fiddling (and its result) are kept secret, the corresponding update model looks like the above opening, but now with the new substitution.

Remark: Two Limitations The preceding examples suggest that we can view an update model with several event as a disjunction of instructions under conditions, i.e., as a kind of *structured program* 'if the window is open then do A or if the window is ajar, then do B, or, if the window is wide open, then do C ....' This suggests a set-up where update models are built from simple actions by means of the regular operations of choice, sequence and iteration — and perhaps even concurrent composition of events. We will not pursue this approach here. Indeed, natural though it is, it transcends the boundaries of our analysis. E.g., Miller and Moss [23] show that just adding finite iteration $*$ of announcements already leads to an undecidable dynamic logic, not effectively reducible to its decidable base.

Another boundary that we will not cross is the *atomic form* of our postconditions for world-changing actions. In more general scenarios, captured by logics with operators in the spirit of 'See To It That φ', one may want to define some action as having some complex epistemic effect described by arbitrary φ, such as 'make sure that only courageous people know the true state of affairs'. Modeling complex postconditions raises some delicate technical issues, orthogonal to our main concerns here.

3 Logics of Public Announcement

Many of the issues we want to deal with in the full-fledged logic of epistemic updates are also present in the logic of the simplest form of communicative action: public announcement logic. The corresponding update idea that announcing a proposition φ removes all worlds where φ does not hold goes back far into the mists of logical folklore, and it has been stated explicitly since the 1970s by Stalnaker, Heim, and others. The same idea also served as a high-light in the work on epistemic logic in computer science (cf. [13]). Its first implementation as a dynamic-epistemic logic seems due to Plaza [26].

Section 3.1 is a brief introduction to public announcement logic (PAL) as usually stated. In Section 3.2 we give a new base logic of *relativized common knowledge*, EL-RC. This extension was first proposed in [3], which analyzed updates as a kind of *relativization* operator on models. Restricted or 'bounded' versions of logical operators like quantifiers or modalities are very common in

semantics, and we will provide further motivation below. The resulting epistemic logic with relativized common knowledge is expressive enough to allow a reduction axiom for common knowledge. A proof system is defined in Section 3.3, and shown to be complete in Section 3.4. The system is extended with reduction axioms for public announcements in Section 3.5. The existence of reduction axioms for public announcements and relativized common knowledge suggests that this new logic is more expressive than the epistemic logics with public announcements proposed elsewhere in the literature. Hence it is interesting to investigate the expressive power of this new logic with characteristic model comparison games. These games are provided in Section 3.6. This technique is then used to investigate the expressive power of relativized common knowledge in Section 3.7, settling all issues of system comparison in this area. Finally, complexity issues are briefly discussed in Section 3.8.

Once again, our systems work for agents' beliefs as well as knowledge in more constrained models. In our informal explanations, we will use either notion, as seems best for getting points across.

3.1 Language and Semantics of PAL

A public announcement is an epistemic event where all agents are told simultaneously and transparently that a certain formula holds right now. This is modeled by a modal operator $[\varphi]$. A formula of the form $[\varphi]\psi$ is read as 'ψ holds after the announcement of φ'. If we also add an operator $C_B\varphi$ to express that φ is common knowledge among agents B, we get public announcement logic with common knowledge (PAL-C). The languages $\mathscr{L}_{\mathsf{PAL}}$ and $\mathscr{L}_{\mathsf{PAL\text{-}C}}$ are interpreted in standard models for epistemic logic.

Definition 1 (Epistemic Models) *Let a finite set of propositional variables P and a finite set of agents N be given. An* epistemic model *is a triple $M = (W, R, V)$ such that*

- *$W \neq \emptyset$ is a set of possible worlds,*
- *$R : N \to \wp(W \times W)$ assigns an accessibility relation $R(a)$ to each agent a,*
- *$V : P \to \wp(W)$ assigns a set of worlds to each propositional variable.*

In epistemic logic the relations $R(a)$ are usually equivalence relations. In this paper we treat the general modal case without such constraints — making 'knowledge' more like *belief*, as observed earlier. But our results also apply to the special modal S5-case of equivalence relations. The semantics are defined with respect to models with a distinguished 'actual world': M, w.

Definition 2 (Semantics of PAL and PAL-C) *Let a model M, w with $M = (W, R, V)$ be given. Let $a \in N$, $B \subseteq N$, and $\varphi, \psi \in \mathscr{L}_{\mathsf{PAL}}$. For atomic proposi-*

tions, negations, and conjunctions we take the usual definition. The definitions for the other operators run as follows:

$$M, w \models \Box_a \varphi \text{ iff } M, v \models \varphi \text{ for all } v \text{ such that } (w, v) \in R(a)$$
$$M, w \models [\varphi]\psi \text{ iff } M, w \models \varphi \text{ implies } M|\varphi, w \models \psi$$
$$M, w \models C_B \varphi \text{ iff } M, v \models \varphi \text{ for all } v \text{ such that } (w, v) \in R(B)^+$$

where $R(B) = \bigcup_{a \in B} R(a)$*, and* $R(B)^+$ *is its transitive closure. The updated model* $M|\varphi = (W', R', V')$ *is defined by restricting* M *to those worlds where* φ *holds. Let*

$$[\![\varphi]\!] = \{v \in W | M, v \models \varphi\}.$$

Now $W' = [\![\varphi]\!]$*,* $R'(a) = R(a) \cap (W \times [\![\varphi]\!])$*, and* $V'(p) = V(p) \cap [\![\varphi]\!]$*.*

Here, mostly for convenience, we chose to define common knowledge as a transitive closure, as in [13]. In [22], common knowledge is defined as the *reflexive* transitive closure. Our results will work either way, with minimal adaptations.

A completeness proof for public announcement logic without an operator for common knowledge (PAL) is straightforward.

Definition 3 (Proof System for PAL) *The proof system for* PAL *is that for multi-modal S5 epistemic logic plus the following reduction axioms:*

$$\begin{aligned} \textit{Atoms} &\vdash [\varphi]p \leftrightarrow (\varphi \rightarrow p) \\ \textit{Partial Functionality} &\vdash [\varphi]\neg\psi \leftrightarrow (\varphi \rightarrow \neg[\varphi]\psi) \\ \textit{Distribution} &\vdash [\varphi](\psi \wedge \chi) \leftrightarrow ([\varphi]\psi \wedge [\varphi]\chi) \\ \textit{Knowledge Announcement} &\vdash [\varphi]\Box_a\psi \leftrightarrow (\varphi \rightarrow \Box_a[\varphi]\psi \end{aligned}$$

as well as the following rules of inference:

(Announcement generalization) *From* $\vdash \psi$*, infer* $\vdash [\varphi]\psi$*.*

The formulas on the left of these equivalences are of the form $[\varphi]\psi$. In **Atoms** the announcement operator no longer occurs on the right-hand side. In the other reduction axioms formulas within the scope of an announcement are of higher complexity on the left than on the right. Note that the **Distribution** axiom is the well known K-axiom from modal logic. When applied successively, these axioms turn every formula of the dynamic language into an equivalent static one, thus showing the earlier 'harmony' between the static and dynamic parts of the total system.

This system allows for compositional analysis of the epistemic effects of statements made in groups of agents. In this light, even the technical reduction to

the static part has a more general thrust worth pointing out. What it says is that knowing the current knowledge of agents in some models suffices, in principle, for knowing the effects of nay announcement actions that could occur. Thus, in the terminology of [6], the static language is rich enough to *pre-encode* all dynamic effects. This is a powerful idea which also occurs in conditional logic, and in reasoning with conditional probabilities. One way of understanding our next topic is as a move towards achieving pre-encoding for common knowledge, too. Here is why this requires work. For public announcement logic with a common knowledge operator (PAL-C), a completeness proof via reduction axioms is impossible. There is no such axiom for formulas of the form $[\varphi]C_B\psi$, given the results in [1].

3.2 Relativized Common Knowledge: EL-RC

Even so, the semantic intuitions for achieving common knowledge by announcement are clear. If φ is true in the old model, then every B-path in the new model ends in a ψ world. This means that in the old model every B-path that consists exclusively of φ-worlds ends in a $[\varphi]\psi$ world. To facilitate this, we introduce a new operator $C_B(\varphi, \psi)$, which expresses that

every B-path which consists exclusively of φ-worlds ends in a ψ world.

We call this notion *relativized common knowledge*. A natural language paraphrase might be 'if φ is announced it becomes common knowledge among B that ψ was the case before the announcement.' A shorter paraphrase of $C_B(\varphi, \psi)$ that we will use henceforth is 'ψ is φ-relative common knowledge among group B.' Henceforth we consider only such φ-relative or φ-conditional common knowledge of agents, just as one does in logics of doxastic *conditionals*, where $A \implies B$ means something like "if I were to learn that A, I would believe that B." Yet another helpful analogy may be with the well-known 'Until' of temporal logic. A temporal sentence 'φ until ψ' is true iff there is some point in the future where ψ holds and φ is true up to that point. All these readings show that the new notion has some concrete intuition behind it. Its other virtue, as we shall see presently, is mathematical elegance.

Definition 4 (Language and Semantics of EL-RC) *The language of* EL-RC *is that of* EL*, together with the operator for relativized common knowledge, with semantics given by:*

$$M, w \models C_B(\varphi, \psi)$$

iff

$$M, v \models \psi \text{ for all } v \text{ such that } (w, v) \in (R(B) \cap (W \times [\![\varphi]\!]))^+$$

where $(R(B) \cap (W \times [\![\varphi]\!]))^+$ *is the transitive closure of* $R(B) \cap (W \times [\![\varphi]\!])$.

Note that φ-relative common knowledge is *not* what results from a public update with φ. E.g., $[p]C_B\Diamond_a\neg p$ is *not* equivalent to $C_B(p, \Diamond_a\neg p)$, for $[p]C_B\Diamond_a\neg p$ is always false, and $C_B(p, \Diamond_a\neg p)$ holds in models where every B path through p worlds ends in a world with an a successor with $\neg p$. In Section 3.7 we will show that $C_B(p, \Diamond_a\neg p)$ cannot be expressed in PAL-C.

The semantics of the other operators is standard. Ordinary common knowledge can be defined with the new notion: $C_B\varphi \equiv C_B(\top, \varphi)$.

3.3 Proof System for EL-RC

Relativized common knowledge still resembles common knowledge, and so we need just a slight adaptation of the usual axioms.

Definition 5 (Proof System for EL-RC) *The proof system for* EL-RC *has these axioms:*

$$
\begin{array}{rl}
\textit{Tautologies} & \textit{All instantiations of propositional tautologies} \\
\Box\ \textit{Distribution} & \vdash \Box_a(\varphi \to \psi) \to (\Box_a\varphi \to \Box_a\psi) \\
C\ \textit{Distribution} & \vdash C_B(\varphi, \psi \to \chi) \to (C_B(\varphi, \psi) \to C_B(\varphi, \chi)) \\
\textit{Mix} & \vdash C_B(\varphi, \psi) \leftrightarrow E_B(\varphi \to (\psi \wedge C_B(\varphi, \psi))) \\
\textit{Induction} & \vdash (E_B(\varphi \to \psi) \wedge C_B(\varphi, \psi \to E_B(\varphi \to \psi))) \to C_B(\varphi, \psi)
\end{array}
$$

and the following rules of inference:

(Modus Ponens) *From* $\vdash \varphi$ *and* $\vdash \varphi \to \psi$ *infer* $\vdash \psi$.
(*$\Box$ **Necessitation) *From* $\vdash \varphi$ *infer* $\vdash \Box_a\varphi$.
(*C **Necessitation) *From* $\vdash \varphi$ *infer* $\vdash C_B(\psi, \varphi)$.

In the **Mix** *and the* **Induction** *axiom, the notation* $E_B\varphi$ *is an abbreviation of* $\bigwedge_{a\in B} \Box_a\varphi$ *(*everybody *believes, or knows,* φ*).*

These axioms are all sound on the intended interpretation. In particular, understanding the validity of the relativized versions **Mix** *and* **Induction** *provides the main idea of our analysis.*

Next, a proof consists of a sequence of formulas such that each is either an instance of an axiom, or it can be obtained from formulas that appear earlier in the sequence by applying a rule. If there is a proof of φ*, we write* $\vdash \varphi$.

Remark It may also be helpful to write $C_B(\varphi, \psi)$ as a sentence in propositional dynamic logic PDL: $[(\bigcup_{a \in B} a; ?\varphi)^+]\psi$. Our proof system essentially follows the usual PDL-axioms for this formula. This technical observation is the key to our more general system LCC in Section 4 below.

3.4 Completeness for EL-RC

To prove completeness for our extended static language EL-RC, we follow [16], [13]. The argument is standard, and our main new point is just that the usual proof in the literature actually yields information about a richer language than is commonly realized.

For a start, we take maximally consistent sets with respect to finite fragments of the language that form a canonical model for that fragment. In particular, for any given formula φ we work with a finite fragment called the *closure* of φ. This is the appropriate analogue of the *Fisher-Ladner closure* from the PDL literature (see [16]).

Definition 6 (Closure) *The* closure *of φ is the minimal set Φ such that*

(1) $\varphi \in \Phi$,
(2) Φ is closed under taking subformulas,
(3) If $\psi \in \Phi$ and ψ is not a negation, then $\neg\psi \in \Phi$,
(4) If $C_B(\psi, \chi) \in \Phi$, then $\Box_a(\psi \to (\chi \wedge C_B(\psi, \chi))) \in \Phi$ for all $a \in B$.

Definition 7 (Canonical Model) *The* canonical model M_φ *for φ is the triple $(W_\varphi, R_\varphi, V_\varphi)$ where*

- $W_\varphi = \{\Gamma \subseteq \Phi \mid \Gamma$ *is maximally consistent in* $\Phi\}$*;*
- $(\Gamma, \Delta) \in R_\varphi(a)$ *iff* $\psi \in \Delta$ *for all* ψ *with* $\Box_a\psi \in \Gamma$*;*
- $V_\varphi(p) = \{\Gamma \mid p \in \Gamma\}$.

Next, we show that a formula in such a finite set is true in the canonical model where that set is taken to be a world, and vice versa.

Lemma 8 (Truth Lemma) *For all $\psi \in \Phi$, $\psi \in \Gamma$ iff $M_\varphi, \Gamma \models \psi$.*

Proof By induction on ψ. The cases for propositional variables, negations, conjunction, and individual epistemic operators are straightforward. Therefore we focus on the case for relativized common knowledge.

From left to right. Suppose $C_B(\psi, \chi) \in \Gamma$. If there is no Δ such that $(\Gamma, \Delta) \in (R(B) \cap (W_\varphi \times [\![\psi]\!]))^+$, then $(M_\varphi, \Gamma) \models C_B(\psi, \chi)$ holds trivially.

Otherwise, take a $\Delta \in W_\varphi$ such that $(\Gamma, \Delta) \in (R(B) \cap (W_\varphi \times [\![\psi]\!]))^+$. We

have to show that $\Delta \models \chi$, but we show something stronger, namely that $\Delta \models \chi$ and $C_B(\psi, \chi) \in \Delta$. This is done by induction on the length of the path from Γ to Δ. The base case is a path of length 1. From our assumption it follows that $\psi \in \Delta$. Our assumption that $C_B(\psi, \chi) \in \Gamma$ implies that $\vdash \delta_\Gamma \to \Box_a(\psi \to (\chi \wedge C_B(\psi, \chi)))$ by the **Mix** axiom. The formula χ is also in Φ. Therefore $\chi \in \Delta$. By applying the induction hypothesis we get $(M_\varphi, \Delta) \models \chi$. We already assumed that $C_B(\psi, \chi) \in \Phi$, therefore also $C_B(\psi, \chi) \in \Delta$. So we are done with the base case.

Now suppose that the path to Δ is of length $n+1$. There must be a path from Γ of length n to a Θ in $(R(B) \cap (W_\varphi \times [\![\psi]\!]))^+$ such that $(\Theta, \Delta) \in R(a)$ for some $a \in N$ and $(M_\varphi, \Delta) \models \psi$. By the induction hypothesis $C_B(\psi, \chi) \in \Theta$. Now we can apply the same reasoning as in the base case to conclude that $(M_\varphi, \Delta) \models \chi$ and $C_B(\psi, \chi) \in \Delta$.

From right to left. Suppose $(M_\varphi, \Gamma) \models C_B(\psi, \chi)$. Now consider the set Λ:

$$\Lambda = \{\delta_\Delta | (\Gamma, \Delta) \in (R(B) \cap (W_\varphi \times [\![\psi]\!]))^+\}$$

Let $\delta_\Lambda = \bigvee_{\Delta \in \Lambda} \delta_\Delta$ We have to show that

$$\vdash \delta_\Lambda \to E_B(\psi \to \delta_\Lambda) \tag{1}$$

Observe that if Λ is empty, then it follows trivially, because an empty disjunction is equivalent to a contradiction.

Otherwise note that for every $a \in B$, for every $\Delta \in \Lambda$ and every $\Delta' \in \overline{\Lambda}$ (where $\overline{\Lambda}$ is the complement of Λ) either $\psi \notin \Delta'$, or there is a formula $\varphi_{\Delta\Delta'}$ such that $\Box_a \varphi_{\Delta\Delta'} \in \Delta$ and $\varphi_{\Delta\Delta'} \notin \Delta'$. From this it follows in both cases that

$$\vdash \delta_\Lambda \to E_B(\psi \to \neg\delta_{\overline{\Lambda}})$$

It can also be shown that $\vdash \delta_\Lambda \vee \delta_{\overline{\Lambda}}$, and therefore we get (1). By necessitation we get

$$\vdash C_B(\psi, \delta_\Lambda \to E_B(\psi \to \delta_\Lambda))$$

By applying the induction axiom we can deduce

$$\vdash E_B(\psi \to \delta_\Lambda) \to C_B(\psi, \delta_\Lambda)$$

Given that $\vdash \delta_\Lambda \to \chi$, we get

$$\vdash E_B(\psi \to \delta_\Lambda) \to C_B(\psi, \chi)$$

It is also the case that $\vdash \delta_\Gamma \to E_B(\psi \to \delta_\Lambda)$. Therefore $C_B(\psi, \chi) \in \Gamma$. $\Box$

The completeness theorem follows in a straightforward way from this lemma.

Theorem 9 (Completeness for EL-RC) $\models \varphi$ *iff* $\vdash \varphi$.

Proof Let $\nvdash \varphi$, i.e. $\neg\varphi$ is consistent. One easily finds a maximally consistent set Γ in the closure of $\neg\varphi$ with $\neg\varphi \in \Gamma$, as only finitely many formulas matter. By the Truth Lemma, $M_{\neg\varphi}, \Gamma \models \neg\varphi$, i.e., $M_{\neg\varphi}, \Gamma \not\models \varphi$.

The soundness of the proof system can easily be shown by induction on the length of proofs, and we do not provide its straightforward details here. □

3.5 Reduction Axioms for PAL-RC

Next, let PAL-RC be the dynamic epistemic logic with both relativized common knowledge and public announcements. Its semantics combines those for PAL and EL-RC. We want to find a reduction axiom for $[\varphi]C_B(\psi, \chi)$, the formula that expresses that after public announcement of φ, every ψ path leads to a χ world. Note that $[\varphi]C_B(\psi, \chi)$ holds exactly in those worlds where every $\varphi \wedge [\varphi]\psi$ path ends in a world where $[\varphi]\chi$ is true. This observation yields the following proof system for PAL-RC:

Definition 10 (Proof System for PAL-RC**)** *The proof system for* PAL-RC *is that for* EL-RC *plus the reduction axioms for* PAL*, together with* **C-Red***:*

$$[\varphi]C_B(\psi, \chi) \leftrightarrow (\varphi \to C_B(\varphi \wedge [\varphi]\psi, [\varphi]\chi)) \quad \textit{(common knowledge reduction)}$$

as well as an inference rule of necessitation for all announcement modalities.

It turns out that PAL-RC is no more expressive than EL-RC by a direct translation, where the translation clause for $[\varphi]C_B(\psi, \chi)$ relies on the above insight:

Definition 11 (Translation from PAL-RC **to** EL-RC**)** *The function t takes a formula from the language of* PAL-RC *and yields a formula in the language of* EL-RC*.*

$$\begin{array}{llll}
t(p) & = p & t([\varphi]p) & = t(\varphi) \to p \\
t(\neg\varphi) & = \neg t(\varphi) & t([\varphi]\neg\psi) & = t(\varphi) \to \neg t([\varphi]\psi) \\
t(\varphi \wedge \psi) & = t(\varphi) \wedge t(\psi) & t([\varphi](\psi \wedge \chi)) & = t([\varphi]\psi) \wedge t([\varphi]\chi) \\
t(\Box_a\varphi) & = \Box_a t(\varphi) & t([\varphi]\Box_a\psi) & = t(\varphi) \to \Box_a t([\varphi]\psi) \\
t(C_B(\varphi, \psi)) & = C_B(t(\varphi), t(\psi)) & t([\varphi]C_B(\psi, \chi)) & = C_B(t(\varphi) \wedge t([\varphi]\psi), t([\varphi]\chi)) \\
 & & t([\varphi][\psi]\chi) & = t([\varphi]t([\psi]\chi))
\end{array}$$

The translation induced by these principles can be formulated as an inside-out procedure, replacing innermost dynamic operators first. To see that it

terminates, we can define a notion of complexity on formulas such that the complexity of the formulas is smaller on the right hand side. We have added the final axiom here for its independent interest, even though it is not strictly necessary for this procedure. As observed in [4], it describes the effect of *sequential composition* of announcements, something which can also be stated as an independently valid law of public announcement saying that the effect of first announcing φ and then ψ is the same as announcing one single assertion, viz. the conjunction $\varphi \wedge [\varphi]\psi$. Standard programming styles for performing the reduction (cf. [9]) do include the final clause in any case. As to its admissibility, note that, when the last clause is called, the innermost application will yield a formula of lesser complexity. The following theorems can be proved by induction on this complexity measure.

Theorem 12 (Translation Correctness) *For each dynamic-epistemic formula φ of* PAL-RC *and each semantic model M, w,*

$$M, w \models \varphi \text{ iff } M, w \models t(\varphi).$$

Theorem 13 ('PAL-RC = EL-RC') *The languages* PAL-RC *and* EL-RC *have equal expressive power.*

Theorem 14 (Completeness for PAL-RC) $\models \varphi$ *iff* $\vdash \varphi$.

Proof The proof system for EL-RC is complete (Theorem 9), and every formula in $\mathscr{L}_{\text{PAL-RC}}$ is provably equivalent to its translation in $\mathscr{L}_{\text{EL-RC}}$, given the reduction axioms. □

3.6 Model Comparison Games for EL-RC

The notion of relativized common knowledge is of independent interest, just as irreducibly binary general quantifiers (such as *Most A are B*) lead to natural completions of logics with only unary quantifiers. It is important to investigate the relation between the logic of epistemic logic with relativized common knowledge with public announcement logic with common knowledge. We provide some more information through characteristic games. Model comparison games for languages with individual modalities are well-known, but dealing with common knowledge: i.e., arbitrary finite iterations, requires some nice twists. These games will be used in the next section to investigate the expressivity of EL-RC relative to PAL-C.

Definition 15 (The EL-RC Game) *Let two epistemic models $M = (W, R, V)$ and $M' = (W', R', V')$ be given. Starting from each $w \in W$ and $w' \in W'$, the* n-round EL-RC game *between Spoiler and Duplicator is given as follows. If $n = 0$ Spoiler wins if w and w' differ in their atomic properties, otherwise Du-*

plicator wins. Otherwise Spoiler can initiate one of the following two scenarios in each round:

$\Box_a$***-move*** *Spoiler chooses a point x in one model which is an a-successor of the current w or w'. Duplicator responds with a matching successor y in the other model. The output is x, y.*

RC_B***-move*** *Spoiler chooses a B-path $x_0 \dots x_k$ in either of the models with x_0 the current w or w'. Duplicator responds with a B-path $y_0 \dots y_m$ in the other model, with $y_0 = w'$. Then Spoiler can (a) make the end points x_k, y_m the output of this round, or (b) he can choose a world y_i (with $i > 0$) on Duplicator's path, and Duplicator must respond by choosing a matching world x_j (with $j > 0$) on Spoilers path, and x_j, y_i becomes the output.*

The game continues with the new output states. If these differ in their atomic properties, Spoiler wins — otherwise, a player loses whenever he cannot perform a move while it is his turn. If Spoiler has not won after all n rounds, Duplicator wins the whole game.

Definition 16 (Modal Depth) *The* modal depth *of a formula is defined by:*

$$d(\bot) = d(p) = 1$$
$$d(\neg\varphi) = d(\varphi)$$
$$d(\varphi \wedge \psi) = \max(d(\varphi), d(\psi))$$
$$d(\Box_a\varphi) = d(\varphi) + 1$$
$$d(C_B(\varphi, \psi)) = \max(d(\varphi), d(\psi)) + 1$$

If two models M, w and M', w' have the same theory up to depth n, we write $M, w \equiv_n M', w'$.

The following result holds for all logical languages that we use in this paper. Recall that our stock of propositional letters is finite.

Lemma 17 (Propositional finiteness) *For every n, up to modal depth n, there are only finitely logically non-equivalent propositions.*

Theorem 18 (Adequacy of the EL-RC Game) *Duplicator has a winning strategy for the n-round game from M, w, M', w' iff $M, w \equiv_n M', w'$.*

Proof The proof is by induction on n. The base case is obvious, and all inductive cases are also standard in modal logic, except that for relativized common knowledge. As usual, perspicuity is increased somewhat by using the dual existential modality $\hat{C}_B(\varphi, \psi)$. From left to right the proof is straightforward.

From right to left. Suppose that $M, w \equiv_{n+1} M', w'$. A winning strategy for

Duplicator in the $(n+1)$-round game can be described as follows. If Spoiler makes an opening move of type [$\Box_a$-move], then the usual modal argument works. Next, suppose that Spoiler opens with a finite sequence in one of the models: say M, without loss of generality. By the Lemma 17, we know that there is only a finite number of complete descriptions of points up to logical depth n, and each point s in the sequence satisfies one of these: say $\Delta(s,n)$. In particular, the end point v satisfies $\Delta(v,n)$. Let $\Delta(n)$ be the disjunction of all formulas $\Delta(s,n)$ occurring on the path. Then, the initial world w satisfies the following formula of modal depth $n+1$: $\hat{C}_B(\Delta(n), \Delta(v,n))$. By our assumption, we also have $M', w' \models \hat{C}_B(\Delta(n), \Delta(v,n))$. But any sequence witnessing this by the truth definition is a response that Duplicator can use for her winning strategy. Whatever Spoiler does in the rest of this round, Duplicator always has a matching point that is n-equivalent in the language. □

Thus, games for $\mathscr{L}_{\mathsf{EL\text{-}RC}}$ are straightforward. But it is also of interest to look at the language $\mathscr{L}_{\mathsf{PAL\text{-}C}}$. Here, the shift modality $[\varphi]$ passing to definable submodels requires a new type of move, not found in ordinary Ehrenfeucht games, where players can decide to change the current model. The following description of what happens is 'modular': a model changing move can be added to model comparison games for ordinary epistemic logic (perhaps with common knowledge), or for our EL-RC game. By way of explanation: we let Spoiler propose a model shift. Players first discuss the 'quality' of that shift, and Duplicator can win if it is deficient; otherwise, the shift really takes place, and play continues within the new models. This involves a somewhat unusual sequential composition of games, but perhaps one of independent interest.

Definition 19 (The PAL-C Game) *Let the setting be the same as for the n-round game in Definition 15. Now Spoiler can initiate one of the following scenario's each round*

$\Box_a$***-move*** *Spoiler chooses a point x in one model which is an a-successor of the current w or w', and Duplicator responds with a matching successor y in the other model. The output of this move is x, y.*

C_B***-move*** *Spoiler chooses a point x in one model which is reachable by a B-path from w or w', and Duplicator responds by choosing a matching world y in the other model. The output of this move is x, y.*

$[\varphi]$***-move*** *Spoiler chooses a number $r < n$, and sets $S \subseteq W$ and $S' \subseteq W'$, with the current $w \in S$ and likewise $w' \in S'$.* Stage 1*: Duplicator chooses states s in $S \cup S'$, $\overline{s}$ in $\overline{S} \cup \overline{S'}$ (where $\overline{S}$ is the complement of S). Then Spoiler and Duplicator play the r-round game for these worlds. If Duplicator wins this subgame, she wins the n-round game.* Stage 2*: Otherwise, the game continues in the relativized models $M|S, w$ and $M'|S', w'$ over $n-r$ rounds.*

The definition of depth is extended to formulas $[\varphi]\psi$ as $d([\varphi]\psi) = d(\varphi) + d(\psi)$.

Theorem 20 (Adequacy of the PAL-RC Game) *Duplicator has a winning strategy for the n-round game on M, w and M', w' iff $M, w \equiv_n M', w'$ in $\mathscr{L}_{\mathsf{PAL\text{-}RC}}$.*

Proof We only discuss the inductive case demonstrating the match between announcement modalities and model-changing steps. From left to right, the proof is straightforward.

From right to left. Suppose that M, w and M', w' are equivalent up to modal depth $n+1$. We need to show that Duplicator has a winning strategy. Consider any opening choice of S, S' and $r < n+1$ made by Spoiler. *Case 1*: Suppose there are two points s, $\overline{s}$ that are equivalent up to depth r. By the induction hypothesis, this is the case if and only if Duplicator has a winning strategy for the r-round game starting from these worlds and so has a winning strategy in Stage 1. *Case 2*: Duplicator has no such winning strategy, which means that Spoiler has one — or equivalently by the inductive hypothesis, every pair $s, \overline{s}$ is distinguished by some formula $\varphi_{s\overline{s}}$ of depth at most r which is true in s and false in $\overline{s}$. Observe that $\delta_s = \bigwedge_{\overline{s} \in \overline{S} \cup \overline{S'}} \varphi_{s\overline{s}}$ is true in s and false in $\overline{S} \cup \overline{S'}$. Note that there can be infinitely many worlds involved in the comparison, but finitely many different formulas will suffice by the Lemma 17, which also holds for this extended language. Further, the formula $\Delta_S = \bigvee_{s \in S} \delta_s$ is true in S and false in $\overline{S} \cup \overline{S'}$. A formula $\Delta_{S'}$ is found likewise, and we let Δ be $\Delta_{S'} \vee \Delta_S$. It is easy to see that Δ is of depth r and defines S in M and S' in M'. Now we use the given language equivalence between M, w and M', w' with respect to all depth $(n+1)$-formulas $\langle \Delta \rangle \psi$ where ψ runs over all formulas of depth $(n+1) - r$. We can conclude that $M|\Delta, w$ and $M'|\Delta, w'$ are equivalent up to depth $(n+1) - r$, and hence Duplicator has a winning strategy for the remaining game, by the inductive hypothesis. So in this case Duplicator has a winning strategy in Stage 2. □

In the next section we will use this game to show that EL-RC is more expressive than PAL-C. For now, we will give an example of how this game can be played.

Definition 21 *Let the model $M(n) = (W, R, V)$ be defined by*

- $W = \{x \in \mathbb{N} \mid 0 \leq x \leq n\}$
- $R = \{(x, x-1) \mid 1 \leq x \leq n\}$
- $V(p) = W$

These models are simply lines of worlds. They can all be seen as submodels of the entire line of natural numbers (where $W = \mathbb{N}$). The idea is that Spoiler cannot distinguish two of these models if the line is long enough. The only hope that Spoiler has is to force one of the current worlds to an endpoint and the other not to be an endpoint. In that case Spoiler can make a $\Box$-move in the world that is not an endpoint and Duplicator is stuck. This will not succeed if the lines are long enough. Note that a C-move does not help Spoiler. Also

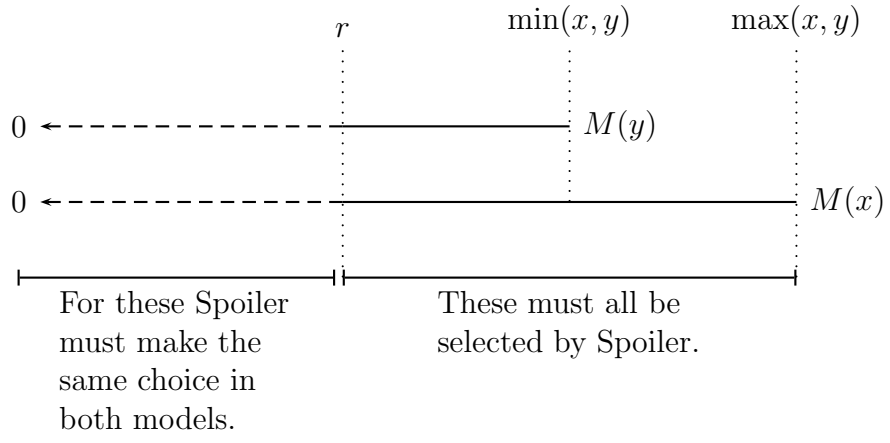


Fig. 1. Illustration of the proof of Lemma 22.

a $[\varphi]$-move will not help Spoiler. Such a move will shorten the lines, but that will cost as many rounds as it shortens them, so Spoiler still loses if they are long enough. The following Lemma captures this idea.

Lemma 22 *For all m, n, and all $x \leq m$ and $y \leq n$ Duplicator has a winning for the* PAL-C *game for $M(m), x$ and $M(n), y$ with at most $\min(x, y)$ rounds.*

Proof If $x = y$, the proof is trivial. We proceed by induction on the number of rounds. Suppose the number of rounds is 0. Then x and y only have to agree on propositional variables. They must agree, since p is true everywhere.

Suppose that the number of rounds is $k+1$ (i.e. $\min(x, y) = k+1$). Duplicator's strategy is the following. If Spoiler chooses to play a $\Box$-move, he moves to $x-1$ (or to $y-1$). Duplicator responds by choosing $y-1$ (or $x-1$). Duplicator has a winning strategy for the resulting subgame by the induction hypothesis.

Suppose Spoiler chooses to play a C-move. If Spoiler chooses a $z < \min(x, y)$, then Duplicator also chooses z. Otherwise, Duplicator takes just one step (the minimum she is required to do). Duplicator has a winning strategy for the resulting subgame by the induction hypothesis.

Suppose Spoiler chooses to play a $[\varphi]$-move. Spoiler chooses a number of rounds r and some S and S'. Observe that for all $z < \min(x, y)$ it must be the case that $z \in S$ iff $z \in S'$. Otherwise, Duplicator has a winning strategy by the induction hypothesis by choosing z and z. Moreover for all $z \geq r$ it must be the case that $z \in S \cup S'$. Otherwise, Duplicator has a winning strategy by the induction hypothesis for $\min(x, y)$ and z. In Stage 2 the resulting subgame will be for two models bisimilar to models to which the induction hypothesis applies. The number of rounds will be $(k+1)-r$, and the lines will be at least $\min(x, y) - r$ long (and $(\min(x, y) = k + 1)$. This is sketched in Figure 1. $\Box$

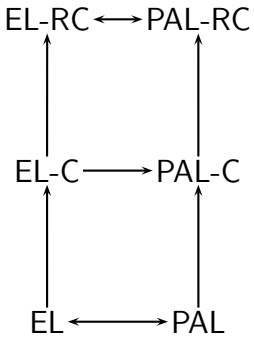


Fig. 2. Expressive power of static and dynamic epistemic logics.

3.7 Expressivity Results

In this section we investigate the expressive power of the logics under consideration here. Reduction axioms and the accompanying translation tell us that two logics have equal expressive power. But our inability to find a compositional translation from one logic to another does not imply that those logics have different expressive power. Here are some known facts. Epistemic logic with common knowledge is more expressive than epistemic logic without common knowledge. In [1] it was shown that public announcement logic with common knowledge is more expressive than epistemic logic with common knowledge. This can also be shown using the results on PDL in [15]. In this section we show that relativized common knowledge logic is more expressive than public announcement logic with common knowledge. The landscape of expressive power is summarized in Figure 2. All arrows are strict.

In general one logic L is more expressive than another logic L' ($\mathsf{L}' \longrightarrow \mathsf{L}$ in Figure 2) if there is a formula in the language of L which is not equivalent to any formula in the language of L' (and every formula in the language of L' is equivalent to some formula in the language of L). So, in order to show that EL-RC is more expressive than PAL-C we need to find a formula in $\mathscr{L}_{\mathsf{EL\text{-}RC}}$ which is not equivalent to any formula in $\mathscr{L}_{\mathsf{PAL\text{-}C}}$. The formula

$$C(p, \neg\Box p)$$

fits this purpose. This will be shown in Theorem 27.

We can show that this formula cannot be expressed in $\mathscr{L}_{\mathsf{PAL\text{-}C}}$ by using model comparison games. We will show that for any number of rounds there are two models such that Duplicator has a winning strategy for the model comparison game, but $C(p, \neg\Box p)$ is true in one of these models and false in the other.

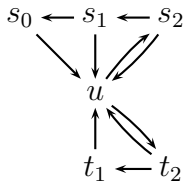
In Definition 25 we provide the models that EL-RC can distinguish, but PAL-C cannot. Since the model comparison game for PAL-C contains the $[\varphi]$-move, we also need to prove that the relevant submodels cannot be distinguished by

PAL-C. We deal with these submodels first in the next definition and lemma.

Definition 23 *Let the model* $M(m,n) = (W,R,V)$ *where* $0 < n \leq m$ *be defined by*

- $W = \{s_x \mid 0 \leq x \leq m\} \cup \{t_x \mid n \leq x \leq m\} \cup \{u\}$
- $R = \{(s_x, s_{x-1}) \mid 1 \leq x \leq m\} \cup \{(t_x, t_{x-1}) \mid n+1 \leq x \leq m\} \cup \{(w,u) \mid w \in W \setminus \{u\}\} \cup \{(u,s_m),(u,t_m)\}$
- $V(p) = W \setminus \{u\}$

The picture below represents $M(2,1)$.



Let us call these models 'hourglasses'. The idea is that Spoiler cannot distinguish the top line from the bottom line of these models if they are long enough. Note that apart from u this model consists of two lines. So if Spoiler plays $\Box$-moves on these lines, Duplicator's strategy is the same as for the line models described above. If he moves to u, Duplicator also moves to u, and surely Duplicator cannot lose the subsequent game in that case. In these models a C-move is very bad for Spoiler, since all worlds are connected by the reflexive transitive closure of R. A $[\varphi]$-move will either yield two lines which are too long, or it will be a smaller hourglass model, which will still be too large, since the $[\varphi]$-move reduces the number of available moves. The Lemma below captures this idea.

In what follows, w_x is a variable ranging over s_x and t_x. And if t_x does not exist it refers to s_x.

Lemma 24 *For all* m, n *and all* $x \leq m$ *and* $y \leq m$ *Duplicator has a winning strategy for the public announcement game for* $M(m,n), w_x$ *and* $M(m,n), w_y$ *with at most* $\min(x,y) - n$ *rounds.*

Proof We prove the case when $w_x = s_x$ and $w_y = t_y$ (the other cases are completely analogous) by induction. Suppose the number of rounds is 0. Then s_x and t_y only have to agree on propositional variables. They do agree, since p is true in both.

Suppose that the number of rounds is $k+1$. Duplicator's winning strategy is the following. If Spoiler chooses to play a $\Box$-move, he moves to s_{x-1} (or to t_{y-1}), or to u. In the last case Duplicator responds by also choosing u, and has a winning strategy for the resulting subgame. Otherwise Duplicator moves to

the t_{y-1} (or s_{x-1}). Duplicator has a winning strategy for the resulting subgame by the inductive hypothesis.

Suppose Spoiler plays a C-move. If Spoiler moves to w, then Duplicator moves to the same w, and has a winning strategy for the resulting subgame.

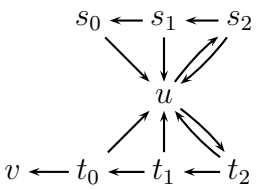
Suppose Spoiler chooses to play a $[\varphi]$-move. Spoiler chooses a number of rounds r and some S. Since there is only one model, Spoiler only chooses one subset of W. Moreover for all $z \geq \min(x, y) - n - r$ it must be the case that $w_z \in S$. Otherwise, Duplicator has a winning strategy by the induction hypothesis for s_x and w_z. In Stage 2 the result will be two models bisimilar to models to which the inductive hypothesis applies, or to which Lemma 22 applies. $\square$

Lastly consider the following class of models.

Definition 25 *Let the model $M^+(m, n) = (W, R, V)$ where $0 < n \leq m$ be defined by*

- $W = \{s_x \mid n \leq x \leq m\} \cup \{t_x \mid 0 \leq x \leq m\} \cup \{v, u\}$
- $R = \{(s_x, s_{x-1}) \mid n + 1 \leq x \leq m\} \cup \{(t_x, t_{x-1}) \mid 1 \leq x \leq m\} \cup$
  $\{(w, u) \mid w \in W \setminus \{v, u\}\} \cup \{(t_0, v)\} \cup \{(u, s_m), (u, t_m)\}$
- $V(p) = W \setminus \{u\}$

The picture below represents $M^+(2, 0)$.



In these 'hourglasses with an appendage', the idea is that Duplicator cannot distinguish the top line from the bottom line of these models when they are long enough. Apart from v, the model is just like a hourglass. So the only new option for Spoiler is to force one of the current worlds to v, and the other to another world. Then Spoiler chooses a $\Box$-move and takes a step from the non-v world and Duplicator is stuck at v. However if the model is large enough v is too far away. Again a C-move does not help Spoiler, because it can be matched exactly by Duplicator. Reducing the model with a $[\varphi]$-move will yield either a hourglass (with or without an appendage) or two lines, for which Spoiler does not have a winning strategy. This idea leads to the following Lemma.

Lemma 26 *For all m, n and all $x \leq m$ and $y \leq m$ Duplicator has a winning strategy for the public announcement game for $M^+(m, n), w_x$ and $M^+(m, n)$, w_y with at most $\min(x, y) - n$ rounds.*

Proof The proof is analogous to the proof of Lemma 24. □

This Lemma now yields to the following theorem.

Theorem 27 EL-RC *is more expressive than* PAL-C.

Proof Suppose PAL-C is just as expressive as EL-RC. Then there is a formula $\varphi \in \mathscr{L}_{\mathsf{PAL\text{-}C}}$ with $\varphi \equiv C(p, \neg\Box p)$. Suppose $d(\varphi) = n$. In that case we would have $M^+(n,0), s_n \models \varphi$ and $M^+(n,0), t_n \not\models \varphi$, contradicting Lemma 26. Hence, EL-RC is more expressive. □

3.8 Complexity Results

Update logics are about processes that manipulate information, and hence they raise natural questions of *complexity*, as a counterpoint to the expressive power of communication and observation scenarios. In particular, all of the usual complexity questions concerning a logical system make sense. *Model checking* asks where a given formula is true in a model, and this is obviously crucial to computing updates. *Satisfiability testing* asks when a given formula has a model, which corresponds to consistency of conversational scenarios in our dynamic epistemic setting. Or, stating the issue in terms of *validity*: when will a given epistemic update always produce some global specified effect? Finally, just as in basic modal logic, there is a non-trivial issue of *model comparison*: when do two given models satisfy the same formulas in our language, i.e., when are two group information states 'the same' for our purposes? As usual, this is related with checking for *bisimulation*, or in a more finely-grained version, the existence of winning strategies for Duplicator in the above model comparison games.

Now technically, the translation of Definition 11 combined with known algorithms for model checking, satisfiability, validity, or model comparison for epistemic logic yield similar algorithms for public announcement logic. But, in a worst case, the length of the translation of a formula is exponential in the length of the formula. E.g., the translation of φ occurs three times in that of $[\varphi]C_B(\psi, \chi)$. Therefore, a direct complexity analysis is worth-while. We provide two results plus some references.

Lemma 28 *Deciding whether a finite model M, w satisfies $\varphi \in \mathscr{L}_{\mathsf{EL\text{-}RC}}$ is computable in polynomial time in the length of φ and the size of M.*

Proof The argument is an easy adaptation of the usual proof for PDL or common knowledge with common knowledge: see [16, p.202] and [13, p.91]. □

This algorithm does not suffice for the case with public announcements. The truth values of φ and ψ in the given model do not fix that of $[\varphi]\psi$. We must also know the value of ψ in the model restricted to φ worlds.

Lemma 29 *Deciding whether a finite model M, w satisfies $\varphi \in \mathscr{L}_{\mathsf{PAL\text{-}RC}}$ is computable in polynomial time in the length of φ and the size of M.*

Proof Again there are at most $|\varphi|$ subformulas of φ. Now we make a binary tree of these formulas which splits with formulas of the form $[\psi]\chi$. On the left subtree all subformulas of ψ occur, on the right all those of χ. This tree can be constructed in time $\mathcal{O}(|\varphi|)$. Labeling the model is done by processing this tree from bottom to top from left to right. The only new case is when we encounter a formula $[\psi]\chi$. In that case we have already processed the left subtree for ψ. Now we first label those worlds where ψ does not hold as worlds where $[\psi]\chi$ holds, then we process the right subtree under $[\psi]\chi$ where we restrict the model to worlds labeled as ψ-worlds. After this process we label those worlds that were labeled with χ as worlds where $[\psi]\chi$ holds and the remaining as worlds where it does not hold. We can see by induction on formula complexity that this algorithm is correct.

Also by induction on φ, this algorithm takes time $\mathcal{O}(|\varphi| \times \|M\|^2)$. The only difficult step is labeling the model with $[\psi]\chi$. By the induction hypothesis, restricting the model to ψ takes time $\mathcal{O}(|\psi| \times \|M\|^2)$. We simply remove (temporarily) all worlds labelled $\neg\varphi$ and all arrows pointing to such worlds. Again by the induction hypothesis, checking χ in this new model takes $\mathcal{O}(|\psi| \times \|M\|^2)$ steps. The rest of the process takes $\|M\|$ steps. So, this step takes over-all time $\mathcal{O}(|[\psi]\chi| \times \|M\|^2)$. □

Moving on from model checking, the satisfiability and the validity problem of epistemic logic with common knowledge are both known to be EXPTIME-complete. In fact, this is true for almost any logic that contains a transitive closure modality. Satisfiability and validity for PDL are also EXPTIME-complete. Now there is a linear time translation of the language of EL-RC to that of PDL. Therefore the satisfiability and validity problems for EL-RC are also EXPTIME-complete. For PAL-RC and even PAL-C, however, the complexity of satisfiability and validity is not settled by this. Lutz [20] shows that satisfiability in PAL is PSPACE-complete, using a polynomial-time translation from dynamic-epistemic to purely epistemic formulas. The latter is unlike the translation underpinning our reduction axioms, in that it is not meaning-preserving. The same method probably extends to PAL-C and PAL-RC.

Finally, the complexity of model comparison for finite models is the same as that for ordinary epistemic logic, viz. PTIME. The reason is that even basic modal equivalence on finite models implies the existence of a bisimulation, while all our extended languages are bisimulation-invariant.

This completes our in-depth analysis of a redesigned dynamic logic of public announcement. Having shown the interest of such a system, we now consider more powerful versions, covering a much wider range of phenomena.

4 A New Logic of Communication and Change

The examples in the Section 2 set a high ambition level for a dynamic epistemic logic updating with events involving both communication and actual change. As we explained, systems of this general sort were proposed in [1,2], but without reduction axioms for common knowledge. We now proceed to a version which can deal with common knowledge, generalizing the compositional methodology for epistemic logic with announcements of Section 3.

In Section 4.1 we introduce *update models* and specify their execution on epistemic models in terms of 'product update'. The only difference with the references above is our addition of fact-changing actions by substitutions. In Section 4.2 we review propositional dynamic logic (PDL) under its epistemic/doxastic interpretation, written henceforth as E-PDL. In Section 4.3 we then present our dynamic epistemic *logic of communication and change* LCC as an extension of E-PDL with dynamic modalities for update models. In Section 4.4 we show that LCC is in harmony with E-PDL through a semantic analysis of epistemic postconditions, and in Section 4.5 we present a proof system for LCC in terms of reduction axioms based on this insight [11,10].

From a technical perspective, the proofs to follow are not just simple generalizations of those for public announcements. In [18] a correspondence between update models and *finite automata* is used to obtain reduction axioms in a dynamic epistemic logic based on so-called 'automata PDL', a variant of E-PDL. Our main new idea here is that this can be stream-lined by analyzing the automata inductively inside E-PDL itself [11], using the well-known proof of Kleene's theorem, stating that languages generated by nondeterministic finite automata are regular [17]. The main theorems to follow use an inductive 'program transformation' approach to epistemic updates whose structure resembles that of Kleene's translation from finite automata to regular expressions. This technique for deriving compositional reduction axioms may be of independent interest beyond the present setting.

4.1 Update Models and their Execution

When viewed by themselves, communicative or other information-bearing scenarios are similar to static epistemic models, in that they involve a space of

possible events and agents' abilities to distinguish between these. In [1] this observation is used as the engine for general update of epistemic models under epistemic actions. In particular, individual events come with *preconditions* holding only at those worlds where they can occur.

Of course, events normally do not just signal information, they also change the world in more concrete ways. Before describing the update mechanism, we enrich our dynamic models with postconditions for events that really change the world. For this purpose we use 'substitutions' that effect changes in valuations at given worlds, serving as *postconditions* for events to occur.

Definition 30 (Substitutions) *$\mathscr{L}$ substitutions are functions of type $\mathscr{L} \to \mathscr{L}$ that distribute over all language constructs, and that map all but a finite number of basic propositions to themselves. $\mathscr{L}$ substitutions can be represented as sets of bindings*

$$\{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$$

where all the p_i are different . If σ is a $\mathscr{L}$ substitution, then the set $\{p \in P \mid \sigma(p) \neq p\}$ is called its domain*, notation $dom(\sigma)$. Use ϵ for the identity substitution. Let $SUB_{\mathscr{L}}$ be the set of all $\mathscr{L}$ substitutions.*

Definition 31 (Epistemic Models under a Substitution) *If $M = (W, V, R)$ is an epistemic model and σ is a $\mathscr{L}$ substitution (for an appropriate epistemic language $\mathscr{L}$), then V^{σ}_{M} is the valuation given by $\lambda p \cdot [\![\sigma(p)]\!]^{M}$. In other words, V^{σ}_{M} assigns to w the set of worlds w in which $\sigma(p)$ is true. For $M = (W, V, R)$, call M^{σ} the model given by (W, V^{σ}_{M}, R).*

Definition 32 (Update Models) *An update model for a finite set of agents N with a language $\mathscr{L}$ is a quadruple $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub})$ where*

- $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$ *is a finite non-empty set of events,*
- $\mathsf{R} : N \to \wp(\mathsf{E}^2)$ *assigns an accessibility relation $\mathsf{R}(a)$ to each agent $a \in N$.*
- $\mathsf{pre} : \mathsf{E} \to \mathscr{L}$ *assigns a precondition to each event,*
- $\mathsf{sub} : \mathsf{E} \to SUB_{\mathscr{L}}$ *assigns a $\mathscr{L}$ substitution to each event.*

A pair U, e is an update model with a distinguished actual event $\mathsf{e} \in \mathsf{E}$.

In these definitions, $\mathscr{L}$ can be any language that can be interpreted in the models of Definition 1. Note that an 'action model' in the sense of [1] is a special update model in our sense, where sub assigns the identity substitution ϵ to every event. Our substitutions then take the original action model philosophy one step further. In particular, our notion of update execution will reset both basic features of information models: epistemic accessibility relations, but also the propositional valuation. Section 4.3 then presents a dynamic logic for communication and real change based on these general update models.

Remark: Information Change and Real Change Note that the world-changing feature is modular here. Readers only interested in epistemic information flow can think of models in the original dynamic epistemic style. All of our major results hold in this special case, and proofs proceed by merely skipping base steps or inductive steps involving the substitutions.

Executing an update is now modeled by the following *product construction.*

Definition 33 (Update Execution) *Given a static epistemic model* $M = (W, R, V)$*, a world* $w \in W$*, an update model* $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub})$ *and an action state* $\mathsf{e} \in \mathsf{E}$ *with* $M, w \models \mathsf{pre}(\mathsf{e})$*, we say that the result of* executing U, e in M, w *is the model* $M \circ \mathsf{U}, (w, \mathsf{e}) = (W', R', V'), (w, \mathsf{e})$ *where*

- $W' = \{(v, \mathsf{f}) \mid M, v \models \mathsf{pre}(\mathsf{f})\}$,
- $R'(a) = \{((v, \mathsf{f}), (u, \mathsf{g})) \mid (v, u) \in R(a)$ *and* $(\mathsf{f}, \mathsf{g}) \in \mathsf{R}(a)\}$,
- $V'(p) = \{(v, \mathsf{f}) \mid M, v \models \mathsf{sub}(\mathsf{f})(p)\}$

Once again, Definitions 32 (with all substitutions set equal to ϵ) and 33 provide a semantics for the logic of epistemic actions LEA of [1]. The basic epistemic language $\mathscr{L}_{\mathsf{LEA}}$ can then be extended with dynamic modalities $[\mathsf{U}, \mathsf{e}]\varphi$, where a U is any *finite update model* for $\mathscr{L}_{\mathsf{LEA}}$. These say that 'every execution of U, e yields a model where φ holds':

$$M, w \models [\mathsf{U}, \mathsf{e}]\varphi \text{ iff } M, w \models \mathsf{pre}(\mathsf{e}) \text{ implies that } M \circ \mathsf{U}, (w, \mathsf{e}) \models \varphi$$

In [1] an axiomatic system for LEA is presented with a, somewhat complicated, completeness proof, without reduction axioms for common knowledge. Our analysis to follow will improve on this.

To see what is needed, observe that, again, the semantic intuition about the crucial case $M, w \models [\mathsf{U}, \mathsf{e}]C_B\varphi$ is clear. It says that, if there is a B-path $w_0, \ldots, w_n$ (with $w_0 = w$) in the static model and a matching B-path $\mathsf{e}_0, \ldots, \mathsf{e}_n$ (with $\mathsf{e}_0 = \mathsf{e}$) in the update model with $M, w_i \models \mathsf{pre}(\mathsf{e}_i)$ for all $i \leq n$, then $M, w_n \models \varphi$. To express all this in the initial static model, it turns out to be convenient to choose a representation of complex epistemic assertions that meshes well with update models.

Now, the relevant finite paths in static models involve strings of agent accessibility steps and tests on formulas. And these finite traces of actions and tests are precisely the sort of structure whose study led to the design of *propositional dynamic logic* (PDL). Initially, PDL was designed for the analysis of programs. In what follows, however, we will give it an epistemic interpretation.

The language of propositional dynamic logic and all further information about its semantics and proof theory may be found in [16], which also has references to the history of this calculus, and its original motivations in computer science. We briefly recall some major notions and results.

Definition 34 (PDL, Language) *Let a set of propositional variables P and a set of relational atoms N be given, with p ranging over P and a over N. The language of* PDL *is given by:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi$$
$$\pi ::= a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

We employ the usual abbreviations: $\bot$ is shorthand for $\neg\top$, $\varphi_1 \vee \varphi_2$ is shorthand for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2$ is shorthand for $\neg(\varphi_1 \wedge \varphi_2)$, $\varphi_1 \leftrightarrow \varphi_2$ is shorthand for $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$, and $\langle\pi\rangle\varphi$ is shorthand for $\neg[\pi]\neg\varphi$.

Definition 35 (PDL, Semantics) *The semantics of* PDL *over P, N is given in models $M = (W, R, V)$ for signature P, N. Formulas of* PDL *are interpreted as subsets of W, relational atoms a as binary relations on W (with the interpretation of relational atoms a given as $R(a)$), as follows:*

$$\begin{aligned}
[\![\top]\!]^M &= W \\
[\![p]\!]^M &= V(p) \\
[\![\neg\varphi]\!]^M &= W \setminus [\![\varphi]\!]^M \\
[\![\varphi_1 \wedge \varphi_2]\!]^M &= [\![\varphi_1]\!]^M \cap [\![\varphi_2]\!]^M \\
[\![\ [\pi]\varphi]\!]^M &= \{w \in W \mid \forall v \text{ if } (w,v) \in [\![\pi]\!]^M \text{ then } v \in [\![\varphi]\!]^M\}
\end{aligned}$$

$$\begin{aligned}
[\![a]\!]^M &= R(a) \\
[\![?\varphi]\!]^M &= \{(w,w) \in W \times W \mid w \in [\![\varphi]\!]^M\} \\
[\![\pi_1;\pi_2]\!]^M &= [\![\pi_1]\!]^M \circ [\![\pi_2]\!]^M \\
[\![\pi_1 \cup \pi_2]\!]^M &= [\![\pi_1]\!]^M \cup [\![\pi_2]\!]^M \\
[\![\pi^*]\!]^M &= ([\![\pi]\!]^M)^*
\end{aligned}$$

Here $([\![\pi]\!]^M)^*$ is the reflexive transitive closure of binary relation $[\![\pi]\!]^M$. If $w \in W$ then we use $M, w \models \varphi$ for $w \in [\![\varphi]\!]^M$, and we say that φ is true at w. A PDL formula φ is *true* in a model if it holds at every state in that model.

These definitions specify how formulas of PDL can be used to make assertions about PDL models. E.g., the formula $\langle a \rangle \top$ says that the current state has an $R(a)$-successor. Truth of $\langle a \rangle \top$ in a model says that $R(a)$ is serial.

Note that ? is an operation for mapping formulas to programs. Programs of the form $?\varphi$ are called *tests*; they are interpreted as the identity relation, restricted to the states s satisfying the formula φ.

If $\sigma = \{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$ is a PDL substitution, we use φ^σ for $\sigma(\varphi)$ and π^σ for $\sigma(\pi)$. We can spell out φ^σ and π^σ, as follows:

$$\begin{array}{llll} \top^\sigma & = \top & a^\sigma & = a \\ p^\sigma & = \sigma(p) & (?\varphi)^\sigma & = ?\varphi^\sigma \\ (\neg\varphi)^\sigma & = \neg\varphi^\sigma & (\pi_1;\pi_2)^\sigma & = \pi_1^\sigma;\pi_2^\sigma \\ (\varphi_1 \wedge \varphi_2)^\sigma & = \varphi_1^\sigma \wedge \varphi_2^\sigma & (\pi_1 \cup \pi_2)^\sigma & = \pi_1^\sigma \cup \pi_2^\sigma \\ ([\pi]\varphi)^\sigma & = [\pi^\sigma]\varphi^\sigma & (\pi^*)^\sigma & = (\pi^\sigma)^*. \end{array}$$

The following holds by simultaneous induction on the structure of formulas and programs:

Lemma 36 (Substitution) *For all* PDL *models M, all* PDL *formulas σ, all* PDL *programs π, all* PDL *substitutions σ:*

$$M, w \models \varphi^\sigma \text{ iff } M^\sigma, w \models \varphi.$$

$$(w, w') \in [\![\pi^\sigma]\!]^M \text{ iff } (w, w') \in [\![\pi]\!]^{M^\sigma}.$$

This is just the beginning of a more general model-theory for PDL, which is bisimulation-based just like basic modal logic.

One striking feature of PDL is that its set of validities is decidable, with a perspicuous axiomatization. We display it here, just to fix thoughts — but no details will be used in what follows.

Theorem 37 *The following axioms and inference rules are complete for* PDL*:*

$$\begin{array}{rl} \textit{(K)} & \vdash [\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi) \\ \textit{(test)} & \vdash [?\varphi_1]\varphi_2 \leftrightarrow (\varphi_1 \to \varphi_2) \\ \textit{(sequence)} & \vdash [\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi \\ \textit{(choice)} & \vdash [\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi \\ \textit{(mix)} & \vdash [\pi^*]\varphi \leftrightarrow \varphi \wedge [\pi][\pi^*]\varphi \\ \textit{(induction)} & \vdash (\varphi \wedge [\pi^*](\varphi \to [\pi]\varphi)) \to [\pi^*]\varphi \end{array}$$

and the following rules of inference:

(Modus Ponens) *From* $\vdash \varphi_1$ *and* $\vdash \varphi_1 \to \varphi_2$, *infer* $\vdash \varphi_2$.
(Modal Generalisation) *From* $\vdash \varphi$, *infer* $\vdash [\pi]\varphi$.

In the rest of this paper, we are going to use PDL for a very special purpose, viz. as a *rich epistemic language*. This may be confusing at first sight, since the objects in our update models are events, and hence one might naturally think of a propositional dynamic logic for sequences of these. The latter use would be close to describing operational structure on update models viewed as *programs*, which we noted in Section 2, but then decided to forego. We ask the reader to firmly resist this association henceforth, and focus instead on the following epistemic perspective ([24,28]). To make the distinction even clearer, we will often refer to propositional dynamic logic in this epistemic guise as E-PDL.

Atomic relations will be epistemic accessibilities of single agents. Compositions like $b_1; b_2$ then express the 'levels of knowledge' of Parikh: if φ expresses that b_1 wants b_2 to pick up the children, then $[b_1; b_2]\varphi$ states that b_1 knows that b_2 knows what is expected of him (a precondition for being at ease about the arrangement). Next, if $B \subseteq N$ and B is finite, we use B as shorthand for $b_1 \cup b_2 \cup \cdots$. Under this convention, the general knowledge operator $E_B\varphi$ takes the shape $[B]\varphi$, while the common knowledge operator $C_B\varphi$ appears as $[B^*]\varphi$, i.e., $[B]\varphi$ expresses that it is general knowledge among agents B that φ, and $[B^*]\varphi$ expresses that it is common knowledge among agents B that φ. In the special case where $B = \emptyset$, B turns out equivalent to $?\bot$, the program that always fails. In the same vein, common *belief* among agents B that φ can be expressed as $[B; B^*]\varphi$. But E-PDL is much richer than these notions, in that it also allows for much more complex combinations of agent accessibility relations, corresponding to some pretty baroque 'generalized agents'. We have found no practical use for these at present, but they are the price that we cheerfully pay for having a language living in expressive harmony with its dynamic superstructure — as will be described now.

4.3 LCC*, a Dynamic Logic of Communication and Change*

Now we have all the ingredients for the definition of the logic of communication and change.

Definition 38 (LCC, Language) *The language* $\mathscr{L}_{\mathsf{LCC}}$ *is the result of adding a clause* $[\mathsf{U}, \mathsf{e}]\varphi$ *for update execution to the language of* E-PDL*, where* U *is an update model for* $\mathscr{L}_{\mathsf{LCC}}$.

Definition 39 (LCC, Semantics) *The semantics* $[\![\varphi]\!]^M$ *is the standard se-*

mantics of PDL*, with the meaning of* $[\mathsf{U},\mathsf{e}]\varphi$ *in* $M=(W,R,V)$ *given by:*

$$[\![[\mathsf{U},\mathsf{e}]\varphi]\!]^M = \{w \in W \mid \textit{if } M,w \models \mathsf{pre}(\mathsf{e}) \textit{ then } (w,\mathsf{e}) \in [\![\varphi]\!]^{M\circ\mathsf{U}}\}.$$

We have to check that the definition of execution of update models is well behaved. The following theorems state that it is, in the sense that it preserves epistemic model bisimulation and update model bisimulation (the corresponding theorems for LEA are proved in [1]).

Theorem 40 *For all* PDL *models* M,w *and* N,v *and all formulas* $\varphi \in \mathscr{L}_{\mathsf{LCC}}$

$$\textit{If } M,w \leftrightarrows N,v \textit{ then } w \in [\![\varphi]\!]^M \textit{ iff } v \in [\![\varphi]\!]^N$$

This theorem must be proved simultaneously with the following result.

Theorem 41 *For all* PDL *models* M,w *and* N,v*, all update models* U,e*:*

$$\textit{If } M,w \leftrightarrows N,v \textit{ then } M\circ\mathsf{U},(w,\mathsf{e}) \leftrightarrows N\circ\mathsf{U},(v,a).$$

Proof We prove both results simultaneously by induction on formulas φ and the preconditions of the relevant update models.

- Proof of Theorem 40: the base case for propositional variables and the cases for negation, conjunction, and program modalities is standard. The only interesting case is for formulas of the form $[\mathsf{U},\mathsf{e}]\varphi$. Suppose $w \in [\![[\mathsf{U},\mathsf{e}]\varphi]\!]^M$. Therefore $w \in [\![\mathsf{pre}(\mathsf{e})]\!]^M$ implies $(w,\mathsf{e}) \in [\![\varphi]\!]^{M\circ\mathsf{U}}$. By the induction hypothesis $w \in [\![\mathsf{pre}(\mathsf{e})]\!]^M$ iff $v \in [\![\mathsf{pre}(\mathsf{e})]\!]^N$ and $M\circ\mathsf{U},(w,\mathsf{e}) \leftrightarrows N\circ\mathsf{U},(v,\mathsf{e})$. Then, by applying the induction hypothesis to $M\circ\mathsf{U},(w,a)$ and $N\circ\mathsf{U},(v,\mathsf{e})$, we infer $v \in [\![\mathsf{pre}(\mathsf{e})]\!]^N$ implies $(v,\mathsf{e}) \in [\![\varphi]\!]^{N\circ\mathsf{U}}$. By the semantics this is equivalent to $v \in [\![[\mathsf{U},\mathsf{e}]\varphi]\!]^N$. The other way around is completely analogous.
- Proof of Theorem 41: Let B be a bisimulation witnessing $M,w \leftrightarrows N,v$. Then the relation C between $W_M \times \mathsf{E}_\mathsf{U}$ and $W_N \times \mathsf{E}_\mathsf{U}$ defined by

  $$(w,\mathsf{e})C(v,\mathsf{f}) \text{ iff } wBv \text{ and } \mathsf{e}=\mathsf{f}$$

  is a bisimulation.

  The induction hypothesis guarantees that (w,e) exists iff (v,f) exists.

  Suppose $(w,\mathsf{e})C(v,\mathsf{f})$. Then wBv and $\mathsf{e}=\mathsf{f}$. The only non-trivial check is the check for sameness of valuation. By wBv, w and v satisfy $V_M(w)=V_N(v)$. By $\mathsf{e}=\mathsf{f}$, e and f have the same substitution σ. By the fact that w and v are bisimilar, by the induction hypothesis we have that $w \in [\![\varphi]\!]^M$ iff $v \in [\![\varphi]\!]^N$. Thus, by $V_M(w)=V_N(s)$ and the definition of V_M^σ and V_N^σ, we get $V_M^\sigma(w)=V_N^\sigma(v)$.

□

Theorem 42 *For all* PDL *models* M, w*, all update models* U_1, e *and* U_2, f*:*

$$\textit{If } \mathsf{U}_1, \mathsf{e} \leftrightarrows \mathsf{U}_2, \mathsf{f} \textit{ then } M \circ \mathsf{U}_1, (w, \mathsf{e}) \leftrightarrows M \circ \mathsf{U}_2, (w, \mathsf{f}).$$

Proof Let R be a bisimulation witnessing $\mathsf{U}_1, \mathsf{e} \leftrightarrows \mathsf{U}_2, \mathsf{f}$. Then the relation C between $W_M \times \mathsf{E}_{\mathsf{U}_1}$ and $W_M \times \mathsf{E}_{\mathsf{U}_2}$ given by

$$(w, \mathsf{e}) C (v, \mathsf{f}) \text{ iff } w = v \text{ and } \mathsf{e} R \mathsf{f}$$

is a bisimulation.

Suppose $(w, \mathsf{e}) C (v, \mathsf{f})$. Then $w = v$ and $\mathsf{e} R \mathsf{f}$. Again, the only non-trivial check is the check for sameness of valuation. By $\mathsf{e} R \mathsf{f}$, the substitutions σ of e and τ of f are equivalent. By $w = v$, $V_M(w) = V_M(v)$. It follows that $V_M^\sigma(w) = V_M^\sigma(v)$, i.e., (w, e) and (v, f) have the same valuation. □

4.4 Expressive Power of LCC

Now, if we have designed things well, the dynamic system just defined should be in harmony with its static substructure. In particular, we expect reduction axioms for compositional analysis of the effects of arbitrary update models: $[\mathsf{U}, \mathsf{e}][\pi]\varphi$. These will then, if one wants to phrase this somewhat negatively, 'reduce LCC to E-PDL.' As before, the quest for such principles starts with an attempt to describe what is the case after the update in terms of what is the case before the update. In case of LCC, epistemic relations can take the shape of arbitrary E-PDL programs. So we must ask ourselves how we can find, for a given relation $[\![\pi]\!]^{M \circ \mathsf{U}}$ a corresponding relation in the original model M, w.

A formula of the form $\langle \mathsf{U}, \mathsf{e}_i \rangle \langle \pi \rangle \varphi$ is true in some model M, w iff there is a π-path in $M \circ \mathsf{U}$ leading from (w, e_i) to a φ world (v, e_j). That means there is some path $w \dots v$ in M and some path $\mathsf{e}_i \dots \mathsf{e}_j$ in U such that $(M, w) \models \mathsf{pre}(\mathsf{e}_i)$ and $\dots$ and $(M, v) \models \mathsf{pre}(\mathsf{e}_j)$ and of course $(M, v) \models \langle \mathsf{U}, \mathsf{e}_j \rangle \varphi$. The program $T_{ij}^{\mathsf{U}}(\pi)$, to be defined below, captures this. A $T_{ij}^{\mathsf{U}}(\pi)$-path in the original model corresponds to a π-path in the updated model. But in defining $T_{ij}^{\mathsf{U}}(\pi)$ we cannot refer to a model M. The definition of the transformed program $T_{ij}^{\mathsf{U}}(\pi)$ only depends on π, U, e_i and e_j. These program transformers are used in the reduction axiom, which can be formulated as follows:

$$[\mathsf{U}, \mathsf{e}_i][\pi]\varphi \leftrightarrow \bigwedge_{j=0}^{n-1} [T_{ij}^{\mathsf{U}}(\pi)][\mathsf{U}, \mathsf{e}_j]\varphi.$$

The remainder of this section is directed towards showing that this axiom is sound (Theorem 48). Our main new technical contribution in this paper lies in the machinery leading up to this.

The program transformer T^{U}_{ij} is defined as follows:

Definition 43 (T^{U}_{ij} Program Transformers)

$$T^{\mathsf{U}}_{ij}(a) = \begin{cases} ?\mathsf{pre}(\mathsf{e}_i); a & \textit{if } \mathsf{e}_i R(a) \mathsf{e}_j, \\ ?\bot & \textit{otherwise} \end{cases}$$

$$T^{\mathsf{U}}_{ij}(?\varphi) = \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\varphi) & \textit{if } i = j, \\ ?\bot & \textit{otherwise} \end{cases}$$

$$T^{\mathsf{U}}_{ij}(\pi_1; \pi_2) = \bigcup_{k=0}^{n-1} (T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2))$$

$$T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2) = T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2)$$

$$T^{\mathsf{U}}_{ij}(\pi^*) = K^{\mathsf{U}}_{ijn}(\pi)$$

where $K^{\mathsf{U}}_{ijn}(\pi)$ is given by Definition 44.

We need the additional program transformer K^{U}_{ijn} in order to build the paths corresponding to the transitive closure of π in the updated model step by step, where we take more and more worlds of the update model into account. Intuitively, $K^{\mathsf{U}}_{ijk}(\pi)$ is a (transformed) program for all the π paths from (w, e_i) to (v, e_j) that can be traced through $M \circ \mathsf{U}$ while avoiding a pass through intermediate states with events e_k and higher (this is the thrust of Definition 44). Here, a π path from (w, e_i) to (v, e_j) is a path of the form $(w, \mathsf{e}_i), (v, \mathsf{e}_j)$ (in case $i = j$), or $(w, \mathsf{e}_i) \xrightarrow{\pi} \cdots \xrightarrow{\pi} (v, \mathsf{e}_j)$. Intermediate states are the states at positions $\cdots$ where a π step ends and a π step starts. Note that the restriction only applies to intermediate states. States passed in the execution of π may involve events e_m with $m > k$. A given intermediate state e_r may occur more than once in a π path.

Just as the definition of $T^{\mathsf{U}}_{ij}(\pi)$ does not refer to a concrete model M, also $K^{\mathsf{U}}_{ijn}(\pi)$ does not depend on a concrete model M. We only need to be concerned about the paths from e_i to e_j that could be the event components in a π-path in the updated model. Thus, $K^{\mathsf{U}}_{ij0}(\pi)$ is a program for all the paths from e_i to e_j that can be traced through U without stopovers at intermediate states that could yield a π path in an updated model. If $i = j$ it either is the skip action or a direct π loop, and otherwise it is a direct $T^{\mathsf{U}}_{ij}(\pi)$ step. This explains the base case in the following notion:

Definition 44 (K^{U}_{ijk} Path Transformers) *$K^{\mathsf{U}}_{ijk}(\pi)$ is defined by recursing on k, as follows:*

$$K^{\mathsf{U}}_{ij0}(\pi) = \begin{cases} ?\top \cup T^{\mathsf{U}}_{ij}(\pi) & \textit{if } i = j, \\ T^{\mathsf{U}}_{ij}(\pi) & \textit{otherwise} \end{cases}$$

$$K^{\mathsf{U}}_{ij(k+1)}(\pi) = \begin{cases} (K^{\mathsf{U}}_{kkk}(\pi))^* & \textit{if } i = k = j, \\ (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi) & \textit{if } i = k \neq j, \\ K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^* & \textit{if } i \neq k = j, \\ K^{\mathsf{U}}_{ijk}(\pi) \cup (K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi)) & \textit{otherwise} \\ & (i \neq k \neq j). \end{cases}$$

Concrete applications of Definitions 43 and 44 are found in Section 5. The next theorem states that the program transformation yields all the paths in the original model that correspond to paths in the updated model.

Theorem 45 (Program Transformation into E-PDL) *For all update models* U *and all* E-PDL *programs* π*, the following equivalence holds:*

$$(w, v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \textit{ iff } ((w, \mathsf{e}_i), (v, \mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}}.$$

To prove Theorem 45 we need two auxiliary results.

Lemma 46 (Constrained Kleene Path) *Suppose*

$$(w, v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \textit{ iff } ((w, \mathsf{e}_i), (v, \mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}}.$$

Then $(w, v) \in [\![K^{\mathsf{U}}_{ijk}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$ *iff there is a* π *path from* (w, e_i) *t* (v, e_j) *in* $M \circ \mathsf{U}$ *that does not have intermediate states* $\cdots \xrightarrow{\pi} (u, \mathsf{e}_r) \xrightarrow{\pi} \cdots$ *with* $r \geq k$*.*

Proof We use induction on k, following the definition of K^{U}_{ijk}, distinguishing a number of cases.

(a) *Base case* $k = 0$*, subcase* $i = j$: A π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does not visit any intermediate states is either empty or a single π step

from (w, e_i) to (v, e_j). Such a path exists iff

$$
\begin{array}{ll}
 & ((w, \mathsf{e}_i).(v, \mathsf{e}_j)) \in [\![?\top \cup \pi]\!]^{M \circ \mathsf{U}} \\
\text{iff (assumption)} & (w, v) \in [\![T^{\mathsf{U}}_{ij}(?\top \cup \pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff (definition } T^{\mathsf{U}}_{ij}) & (w, v) \in [\![?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\top) \cup T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff } (i = j, & (w, v) \in [\![?\top \cup T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\quad \text{so } \mathsf{pre}(\mathsf{e}_i) = \mathsf{pre}(\mathsf{e}_j)) & \\
\text{iff (definition } K^{\mathsf{U}}_{ij0}) & (w, v) \in [\![K^{\mathsf{U}}_{ij0}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M.
\end{array}
$$

(b) *Base case* $k = 0$*, subcase* $i \neq j$: A π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does not visit any intermediate states is a single π step from (w, e_i) to (v, e_j). Such a path exists iff

$$
\begin{array}{ll}
 & ((w, \mathsf{e}_i).(v, \mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}} \\
\text{iff (assumption)} & (w, v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff (definition } K^{\mathsf{U}}_{ij0}) & (w, v) \in [\![K^{\mathsf{U}}_{ij0}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M.
\end{array}
$$

(c) *Induction step.* Assume that $(w, v) \in [\![K^{\mathsf{U}}_{ijk}(\pi); ?\mathsf{pre}(\mathsf{e}_j]\!]^M$ iff there is a π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does not pass through any pairs (u, e) with $\mathsf{e} \in \{\mathsf{e}_k, \ldots, \mathsf{e}_{n-1}\}$.

We have to show that $(w, v) \in [\![K^{\mathsf{U}}_{ij(k+1)}(\pi); ?\mathsf{pre}(\mathsf{e}_j]\!]^M$ iff there is a π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does not pass through any pairs (u, e) with $\mathsf{e} \in \{\mathsf{e}_{k+1}, \ldots, \mathsf{e}_{n-1}\}$.

Case $i = k = j$. A π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does not pass through any pairs (u, e) with $\mathsf{e} \in \{\mathsf{e}_{k+1}, \ldots, \mathsf{e}_{n-1}\}$ now consists of an arbitrary composition of π paths from e_k to e_k that do not visit any intermediate states with event component e_k or higher. By the induction hypothesis, such a path exists iff $(w, v) \in [\![(K^{\mathsf{U}}_{kkk}(\pi))^*; ?\mathsf{pre}(\mathsf{e}_j]\!]^M$ iff (definition of $K^{\mathsf{U}}_{ij(k+1)}$) $(w, v) \in [\![K^{\mathsf{U}}_{ij(k+1)}(\pi); ?\mathsf{pre}(\mathsf{e}_j]\!]^M$.

Case $i = k \neq j$. A π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ that does pass through any pairs (u, e) with $\mathsf{e} \in \{\mathsf{e}_{k+1}, \ldots, \mathsf{e}_{n-1}\}$ now consists of a π path starting in (w, e_k) visiting states of the form (u, e_k) an arbitrary number of times, but never visiting states with event component e_k or higher in between, and ending in (v, e_k), followed by a π path from (u, e_k) to (v, e_j) that does not visit any pairs with event component $\mathsf{e} \in \{\mathsf{e}_k, \ldots, \mathsf{e}_{n-1}\}$. By the induction hypothesis, a π path from (w, e_k) to (u, e_k) of the first kind exists iff $(w, u) \in [\![(K^{\mathsf{U}}_{kkk}(\pi))^*; ?\mathsf{pre}(\mathsf{e}_k)]\!]^M$. Again by the induction hypothesis, a path

from (u, e_k) to (v, e_j) of the second kind exists iff $(u, v) \in [\![K^{\mathsf{U}}_{kjk}; ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$. Thus, the required path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$ exists iff $(w, v) \in [\![(K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$, which, by the definition of $K^{\mathsf{U}}_{ij(k+1)}$, is the case iff $(w, v) \in [\![K^{\mathsf{U}}_{ij(k+1)}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$.

The other two cases are similar. □

Lemma 47 (General Kleene Path) *Suppose* $(w, v) \in [\![T^{\mathsf{U}}_{ij}(\pi)); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$ *iff there is a* π *step from* (w, e_i) *to* (v, e_j) *in* $M \circ \mathsf{U}$.

Then $(w, v) \in [\![K^{\mathsf{U}}_{ijn}(\pi)); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$ *iff there is a* π *path from* (w, e_i) *to* (v, e_j) *in* $M \circ \mathsf{U}$.

Proof Suppose $(w, v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M$ iff there is a π path from (w, e_i) to (v, e_j) in $M \circ \mathsf{U}$. Then, assuming that U has states $\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}$, an application of Lemma 46 yields that $K^{\mathsf{U}}_{ijn}(\pi)$ is a program for all the π paths from (w, e_i) to (v, e_j) that can be traced through $M \circ \mathsf{U}$, for stopovers at any (u, e_k) with $0 \leq k \leq n-1$ are allowed. □

Lemma 47 explains the use of K^{U}_{ijn} in the clause for π^* in Definition 43. Now, we can clinch matters:

Proof of Theorem 45. This time, we use induction on the structure of π.

Base case a:

$$
\begin{aligned}
& (w, v) \in [\![T^{\mathsf{U}}_{ij}(a); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff } & (w, v) \in [\![?\mathsf{pre}(\mathsf{e}_i); a; ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \text{ and } \mathsf{e}_i R(a) \mathsf{e}_j \\
\text{iff } & M, w \models \mathsf{pre}(\mathsf{e}_i), (w, v) \in [\![a]\!]^M, \mathsf{e}_i R(a) \mathsf{e}_j \text{ and } M, v \models \mathsf{pre}(\mathsf{e}_j) \\
\text{iff } & ((w, \mathsf{e}_i), (v, \mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}}.
\end{aligned}
$$

Base case $?\varphi$, *subcase* $i = j$:

$$
\begin{aligned}
& (w, v) \in [\![T^{\mathsf{U}}_{ij}(?\varphi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff } & (w, v) \in [\![?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\varphi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff } & w = v \text{ and } M, w \models \mathsf{pre}(\mathsf{e}_i) \text{ and } M, w \models [\mathsf{U}, \mathsf{e}_i]\varphi \\
\text{iff } & w = v \text{ and } M, w \models \mathsf{pre}(\mathsf{e}_i) \text{ and } M, w \models \mathsf{pre}(\mathsf{e}_i) \text{ implies } M \circ \mathsf{U}, (w, \mathsf{e}_i) \models \varphi \\
\text{iff } & w = v \text{ and } M \circ \mathsf{U}, (w, \mathsf{e}_i) \models \varphi \\
\text{iff } & ((w, \mathsf{e}_i), (v, \mathsf{e}_j)) \in [\![?\varphi]\!]^{M \circ \mathsf{U}}.
\end{aligned}
$$

Base case $?\varphi$, subcase $i \neq j$:

$$\begin{aligned} & (w,v) \in [\![T^{\mathsf{U}}_{ij}(?\varphi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\ \text{iff } & (w,v) \in [\![?\bot]\!]^M \\ \text{iff } & ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![?\varphi]\!]^{M\circ\mathsf{U}}. \end{aligned}$$

Induction step: Now consider any complex program π and assume for all components π' of π that:

$$(w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi'); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \text{ iff } ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi']\!]^{M\circ\mathsf{U}}.$$

We have to show:

$$(w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \text{ iff } ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi]\!]^{M\circ\mathsf{U}}.$$

Here are the three relevant program operations: $\pi = \pi_1;\pi_2$:

$$\begin{aligned} & (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi_1;\pi_2); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\ \text{iff } & (w,v) \in [\![\textstyle\bigcup_{k=0}^{n-1}(T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2)); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\ \text{iff } & \text{for some } k \in \{0,\ldots,n-1\} (w,v) \in [\![T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\ \text{iff } & \text{for some } k \in \{0,\ldots,n-1\} \text{ and some } u \in W \\ & (w,u) \in [\![T^{\mathsf{U}}_{ik}(\pi_1)]\!]^M \text{ and } (u,v) \in [\![T^{\mathsf{U}}_{kj}(\pi_2); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\ \text{iff (ih)} & \text{for some } k \in \{0,\ldots,n-1\} \text{ and some } u \in W \\ & (w,u) \in [\![T^{\mathsf{U}}_{ik}(\pi_1)]\!]^M \text{ and } ((u,\mathsf{e}_k),(v,\mathsf{e}_j)) \in [\![\pi_2]\!]^{M\circ\mathsf{U}} \\ \text{iff } & \text{for some } k \in \{0,\ldots,n-1\} \text{ and some } u \in W \\ & (w,u) \in [\![T^{\mathsf{U}}_{ik}(\pi_1)]\!]^M, \text{ and} \\ & M,u \models \mathsf{pre}(\mathsf{e}_k) \text{ and } ((u,\mathsf{e}_k),(v,\mathsf{e}_j)) \in [\![\pi_2]\!]^{M\circ\mathsf{U}} \\ \text{iff } & \text{for some } k \in \{0,\ldots,n-1\} \text{ and some } u \in W \\ & (w,u) \in [\![T^{\mathsf{U}}_{ik}(\pi_1); \mathsf{pre}(\mathsf{e}_k)]\!]^M \text{ and } ((u,\mathsf{e}_k),(v,\mathsf{e}_j)) \in [\![\pi_2]\!]^{M\circ\mathsf{U}} \\ \text{iff (ih too)} & \text{for some } k \in \{0,\ldots,n-1\} \text{ and some } u \in W \\ & ((w,\mathsf{e}_i),(u,\mathsf{e}_k)) \in [\![\pi_1]\!]^{M\circ\mathsf{U}} \text{ and } ((u,\mathsf{e}_k),(v,\mathsf{e}_j)) \in [\![\pi_2]\!]^{M\circ\mathsf{U}} \\ \text{iff } & ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi]\!]^{M\circ\mathsf{U}}. \end{aligned}$$

$\pi = \pi_1 \cup \pi_2$:

$$
\begin{array}{ll}
 & (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff} & (w,v) \in [\![(T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2)); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff} & (w,v) \in [\![(T^{\mathsf{U}}_{ij}(\pi_1); ?\mathsf{pre}(\mathsf{e}_j)) \cup (T^{\mathsf{U}}_{ij}(\pi_2) ?\mathsf{pre}(\mathsf{e}_j))]\!]^M \\
\text{iff} & (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi_1); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \text{ or } (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi_2); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff (ih)} & ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi_1]\!]^{M \circ \mathsf{U}} \text{ or } ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi_2]\!]^{M \circ \mathsf{U}} \\
\text{iff} & ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}}.
\end{array}
$$

$\pi = \pi^*$:

$$
\begin{array}{ll}
 & (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi^*); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff (definition } T^{\mathsf{U}}_{ij}) & (w,v) \in [\![K^{\mathsf{U}}_{ijn}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M \\
\text{iff (ih, Lemma 47)} & \text{there is a } \pi \text{ path from } (w,\mathsf{e}_i) \text{ to } (v,\mathsf{e}_j) \text{ in } M \circ \mathsf{U} \\
\text{iff} & ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi^*]\!]^{M \circ \mathsf{U}}.
\end{array}
$$

□

Theorem 48 (Reduction Equivalence) *Suppose that the model* U *has* n *states* $\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}$. *Then:*

$$M, w \models [\mathsf{U}, \mathsf{e}_i][\pi]\varphi \text{ iff } M, w \models \bigwedge_{j=0}^{n-1} [T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U}, \mathsf{e}_j]\varphi.$$

Proof The result is derived by the following chain of equivalences:

$$
\begin{array}{ll}
 & M, w \models [\mathsf{U}, \mathsf{e}_i][\pi]\varphi \\
\text{iff} & M, w \models \mathsf{pre}(\mathsf{e}_i) \text{ implies } M \circ \mathsf{U}, (w, \mathsf{e}_i) \models [\pi]\varphi \\
\text{iff} & \forall v \in W, j \in \{0, \ldots, n-1\} \text{ with } ((w,\mathsf{e}_i),(v,\mathsf{e}_j)) \in [\![\pi]\!]^{M \circ \mathsf{U}}, \\
 & \quad M \circ \mathsf{U}, (v, \mathsf{e}_j) \models \varphi \\
\text{iff (Thm 45)} & \forall v \in W, j \in \{0, \ldots, n-1\} \text{ with } (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi); ?\mathsf{pre}(\mathsf{e}_j)]\!]^M, \\
 & \quad M \circ \mathsf{U}, (v, \mathsf{e}_j) \models \varphi \\
\text{iff} & \forall v \in W, j \in \{0, \ldots, n-1\} \text{ with } (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi)]\!]^M, \\
 & \quad M, v \models \mathsf{pre}(\mathsf{e}_j) \text{ implies } M \circ \mathsf{U}, (v, \mathsf{e}_j) \models \varphi \\
\text{iff} & \forall v \in W, j \in \{0, \ldots, n-1\} \text{ with } (w,v) \in [\![T^{\mathsf{U}}_{ij}(\pi)]\!]^M, \\
 & \quad M, v \models [\mathsf{U}, \mathsf{e}_j]\varphi \\
\text{iff} & M, w \models \textstyle\bigwedge_{j=0}^{n-1} [T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U}, \mathsf{e}_j]\varphi
\end{array}
$$

□

What the Reduction Equivalence tells us is that LCC is equivalent to E-PDL, and hence, that a proof system for LCC can be given in terms of axioms that reduce formulas of the form $[\mathsf{U},\mathsf{e}]\varphi$ to equivalent formulas ψ with the property that their main operator is not an update modality for U. First, we state the former model-theoretic expressiveness result, which is the first main theorem of this paper:

Theorem 49 ('LCC = E-PDL') *The languages of* LCC *and* E-PDL *have equal expressive power.*

The earlier-mentioned similarity between finite automata and our current approach is most striking in the definition of the transformation for starred programs. The definition of transformed π^* paths in terms of operators $K_{ijk}(\pi)$ resembles the definition of sets of regular languages L_k generated by moving through a non-deterministic finite automaton without passing through states numbered k or higher, in the well-known proof of Kleene's Theorem. Textbook versions of its proof can be found in many places, e.g., [19, Theorem 2.5.1].

Another, more technical, interpretation of Theorems 11, 12 is that they establish a strong *closure property* for propositional dynamic logic: not just under the usual syntactic relativizations to submodels, but also under syntactic counterparts for much more general model transformations.

4.5 Reduction Axioms and Completeness for LCC

The results from the previous section point the way to appropriate reduction axioms for LCC. In the axioms below $p^{\mathsf{sub}}(\mathsf{e})$ is $\mathsf{sub}(\mathsf{e})(p)$ if p is in the domain of $\mathsf{sub}(\mathsf{e})$, otherwise it is p.

Definition 50 (Proof System for LCC**)** *The proof system for* LCC *consists of all axioms and rules of* PDL*, plus the following reduction axioms:*

$$\begin{aligned}
[\mathsf{U},\mathsf{e}]\top &\leftrightarrow \top\\
[\mathsf{U},\mathsf{e}]p &\leftrightarrow (\mathsf{pre}(\mathsf{e}) \to p^{\mathsf{sub}(\mathsf{e})})\\
[\mathsf{U},\mathsf{e}]\neg\varphi &\leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi)\\
[\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2) &\leftrightarrow ([\mathsf{U},\mathsf{e}]\varphi_1 \wedge [\mathsf{U},\mathsf{e}]\varphi_2)\\
[\mathsf{U},\mathsf{e}_i][\pi]\varphi &\leftrightarrow \bigwedge_{j=0}^{n-1} [T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U},\mathsf{e}_j]\varphi.
\end{aligned}$$

plus inference rules of necessitation for all update model modalities.

The last, and most crucial, of the reduction axioms in the given list is based on program transformation. Incidentally, if slightly more general updates with so-called 'multiple pointed update models' (cf. [12]) are added to the language, we would need this additional reduction axiom:

$$[\mathsf{U},\mathsf{W}]\varphi \leftrightarrow \bigwedge_{\mathsf{e}\in\mathsf{W}} [\mathsf{U},\mathsf{e}]\varphi$$

As before with logics for public announcement, these reduction axioms also drive a *translation procedure*. The results of Section 4.4 tell us that LCC is no more expressive than E-PDL; indeed, program transformations provide the following translation:

Definition 51 (Translation) *The function t takes a formula from the language of* LCC *and yields a formula in the language of* PDL.

$$\begin{array}{lll}
t(\top) & = \top & r(a) = a \\
t(p) & = p & r(B) = B \\
t(\neg\varphi) & = \neg t(\varphi) & r(?\varphi) = ?t(\varphi) \\
t(\varphi_1 \wedge \varphi_2) & = t(\varphi_1) \wedge t(\varphi_2) & r(\pi_1;\pi_2) = r(\pi_1);r(\pi_2) \\
t([\pi]\varphi) & = [r(\pi)]t(\varphi) & r(\pi_1 \cup \pi_2) = r(\pi_1) \cup r(\pi_2) \\
t([\mathsf{U},\mathsf{e}]\top) & = \top & r(\pi^*) = (r(\pi))^*. \\
t([\mathsf{U},\mathsf{e}]p) & = t(\mathsf{pre}(\mathsf{e})) \rightarrow p^{\mathsf{sub}(\mathsf{e})} & \\
t([\mathsf{U},\mathsf{e}]\neg\varphi) & = t(\mathsf{pre}(\mathsf{e})) \rightarrow \neg t([\mathsf{U},\mathsf{e}]\varphi) & \\
t([\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2)) & = t([\mathsf{U},\mathsf{e}]\varphi_1) \wedge t([\mathsf{U},\mathsf{e}]\varphi_2) & \\
t([\mathsf{U},\mathsf{e}_i][\pi]\varphi & = \bigwedge_{j=0}^{n-1}[T_{ij}^{\mathsf{U}}(r(\pi))]t([\mathsf{U},\mathsf{e}_j]\varphi) & \\
t([\mathsf{U},\mathsf{e}][\mathsf{U}',\mathsf{e}']\varphi & = t([\mathsf{U},\mathsf{e}]t([\mathsf{U}',\mathsf{e}']\varphi)) &
\end{array}$$

The correctness of this translation follows from direct semantic inspection, using the program transformation corollary for the translation of $[\mathsf{U},\mathsf{e}_i][\pi]\varphi$ formulas. The clause for iterated update modalities gives rise to exactly the same comments as those made for PAL-RC in Section 3.5.

As for deduction in our system, we note the following result:

Theorem 52 (Completeness for LCC) $\models \varphi$ *iff* $\vdash \varphi$.

Proof The proof system for PDL is complete, and every formula in the language of LCC is provably equivalent to a PDL formula. □

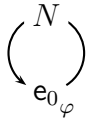
5 Analyzing Major Communication Types

Our analysis of LCC has been abstract and general. But the program transformation approach has a concrete pay-off! It provides a systematic perspective on communicative updates that occur in practice. For public announcement and common knowledge, it was still possible to find appropriate reduction axioms by hand. Such axioms can also be generated automatically, however, by program transformation, as we will now show. This method then allows us to deal with much more complicated cases, such as secret group communication and common belief, or subgroup announcement and common knowledge, where axiom generation by hand is infeasible. For a border-line case, see [27] for a direct axiomatization of the logic of subgroup communication with common knowledge — a topic conspicuously absent from, e.g., [14]. Our analysis obviates the need for this laborious, and error-prone, work.

The following generated axioms may look unwieldy, illustrating the fact that E-PDL functions as an assembler language for detailed analysis of the higher level specifications of communicative updates in terms of update models. But upon closer inspection, they make sense, and indeed, for simple communicative scenarios, they can be seen to reduce to EL-RC.

5.1 Public Announcement and Common Knowledge

The update model for public announcement that φ consists of a single state e_0 with precondition φ and epistemic relation $\{(\mathsf{e}_0, \mathsf{e}_0)\}$ for all agents. Call this model P_φ.



We are interested how public announcement that φ affects common knowledge in a group of agents B, i.e., we want to compute $[P_\varphi, \mathsf{e}_0][B^*]\psi$. For this, we need $T_{00}^{P_\varphi}(B^*)$, which equalled $K_{001}^{P_\varphi}(B)$.

To work out $K_{001}^{P_\varphi}(B)$, we need $K_{000}^{P_\varphi}(B)$, and for $K_{000}^{P_\varphi}(B)$, we need $T_{00}^{P_\varphi}(B)$, which turns out to be $\bigcup_{b\in B}(?\varphi; b)$, or equivalently, $?\varphi; B$. Working upwards from this, we get:

$$K_{000}^{P_\varphi}(B) = ?\top \cup T_{00}^{P_\varphi}(B) = ?\top \cup (?\varphi; B),$$

and therefore:

$$\begin{aligned} K_{001}^{P_\varphi}(B) &= (K_{000}^{P_\varphi}(B))^* \\ &= (?\top \cup (?\varphi; B))^* \\ &= (?\varphi; B)^*. \end{aligned}$$

Thus, the reduction axiom for the public announcement update P_φ with respect to the program for common knowledge among agents B, works out as follows:

$$\begin{aligned} [P_\varphi, \mathsf{e}_0][B^*]\psi &\leftrightarrow [T_{00}^{P_\varphi}(B^*)][P_\varphi, \mathsf{e}_0]\psi \\ &\leftrightarrow [K_{001}^{P_\varphi}(B)][P_\varphi, \mathsf{e}_0]\psi \\ &\leftrightarrow [(?\varphi; B)^*][P_\varphi, \mathsf{e}_0]\psi. \end{aligned}$$

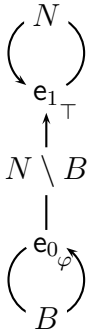
This expresses that every B path consisting of φ worlds ends in a $[P_\varphi, \mathsf{e}_0]\psi$ world, i.e., it expresses what is captured by the special purpose operator $C_B(\varphi, \psi)$ from Section 3.2.

5.2 Secret Group Communication and Common Belief

The logic of secret group communication is the logic of email 'cc' (assuming that emails arrive immediately and are read immediately). The update model for a secret group message to B that φ consists of two possible events $\mathsf{e}_0, \mathsf{e}_1$, where e_0 has precondition φ and e_1 has precondition $\top$, and where the accessibilities T are given by:

$$T = \{\mathsf{e}_0 R(b) \mathsf{e}_0 \mid b \in B\} \cup \{\mathsf{e}_0 R(a) \mathsf{e}_1 \mid a \in N \setminus B\} \cup \{\mathsf{e}_1 R(a) \mathsf{e}_1 \mid a \in N\}.$$

The actual event is e_0. The members of B are aware that φ gets communicated; the others think that nothing happens. In this thought they are mistaken, which is why 'cc' updates generate KD45 models: i.e., 'cc' updates make knowledge degenerate into belief.



We work out the program transformations that this update engenders for common knowledge among a group of agents D. Call the update model CC_φ^B.

We will have to work out $K_{002}^{\mathrm{CC}_\varphi^B} D$, $K_{012}^{\mathrm{CC}_\varphi^B} D$, $K_{112}^{\mathrm{CC}_\varphi^B} D$, $K_{102}^{\mathrm{CC}_\varphi^B} D$.

For these, we need $K_{001}^{\mathrm{CC}_\varphi^B} D$, $K_{011}^{\mathrm{CC}_\varphi^B} D$, $K_{111}^{\mathrm{CC}_\varphi^B} D$, $K_{101}^{\mathrm{CC}_\varphi^B} D$.

For these in turn, we need $K_{000}^{\mathrm{CC}_\varphi^B} D$, $K_{010}^{\mathrm{CC}_\varphi^B} D$, $K_{110}^{\mathrm{CC}_\varphi^B} D$, $K_{100}^{\mathrm{CC}_\varphi^B} D$.

For these, we need:

$$
\begin{aligned}
T_{00}^{\mathrm{CC}_\varphi^B} D &= \bigcup_{d \in B \cap D} (?\varphi; d) \ = ?\varphi; (B \cap D) \\
T_{01}^{\mathrm{CC}_\varphi^B} D &= \bigcup_{d \in D \setminus B} (?\varphi; d) \ = ?\varphi; (D \setminus B) \\
T_{11}^{\mathrm{CC}_\varphi^B} D &= D \\
T_{10}^{\mathrm{CC}_\varphi^B} D &= ?\bot
\end{aligned}
$$

It follows that:

$$
\begin{aligned}
K_{000}^{\mathrm{CC}_\varphi^B} D &= ?\top \cup (?\varphi; (B \cap D)) \\
K_{010}^{\mathrm{CC}_\varphi^B} D &= ?\varphi; (D \setminus B) \\
K_{110}^{\mathrm{CC}_\varphi^B} D &= ?\top \cup D, \\
K_{100}^{\mathrm{CC}_\varphi^B} D &= ?\bot
\end{aligned}
$$

From this we can work out the K_{ij1}, as follows:

$$
\begin{aligned}
K_{001}^{\mathrm{CC}_\varphi^B} D &= (?\varphi; (B \cap D))^* \\
K_{011}^{\mathrm{CC}_\varphi^B} D &= (?\varphi; (B \cap D))^*; (D \setminus B) \\
K_{111}^{\mathrm{CC}_\varphi^B} D &= ?\top \cup D \\
K_{101}^{\mathrm{CC}_\varphi^B} D &= ?\bot.
\end{aligned}
$$

Finally, we get K_{002} and K_{012} from this:

$$
\begin{aligned}
K_{002}^{\mathrm{CC}_\varphi^B} D &= K_{001}^{\mathrm{CC}_\varphi^B} D \cup K_{011}^{\mathrm{CC}_\varphi^B} D; (K_{111}^{\mathrm{CC}_\varphi^B} D)^*; K_{101}^{\mathrm{CC}_\varphi^B} D \\
&= K_{001}^{\mathrm{CC}_\varphi^B} D \quad \text{(since the right-hand expression evaluates to } ?\bot) \\
&= (?\varphi; (B \cap D))^* \\
K_{012}^{\mathrm{CC}_\varphi^B} D &= K_{011}^{\mathrm{CC}_\varphi^B} D \cup K_{011}^{\mathrm{CC}_\varphi^B} D; (K_{111}^{\mathrm{CC}_\varphi^B} D)^* \\
&= K_{011}^{\mathrm{CC}_\varphi^B} D; (K_{111}^{\mathrm{CC}_\varphi^B} D)^* \\
&= (?\varphi; (B \cap D))^*; (D \setminus B); D^*.
\end{aligned}
$$

Thus, the program transformation for common belief among D works out as follows:

$$
[\mathrm{CC}_\varphi^B, \mathsf{e}_0][D^*]\psi
$$

$$
\leftrightarrow
$$

$$
[(?\varphi; (B \cap D))^*][\mathrm{CC}_\varphi^B, \mathsf{e}_0]\psi \wedge [(?\varphi; (B \cap D))^*; (D \setminus B); D^*][\mathrm{CC}_\varphi^B, \mathsf{e}_1]\psi.
$$

This transformation yields a reduction axiom that shows that EL-RC also suffices to provide reduction axioms for secret group communication.

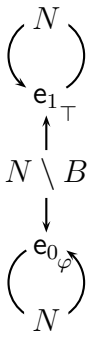
5.3 Group Messages and Common Knowledge

Finally, we consider group messages. This example is one of the simplest cases that shows that program transformations gives us reduction axioms that are no longer feasible to give by hand.

The update model for a group message to B that φ consists of two states $\mathsf{e}_0, \mathsf{e}_1$, where e_0 has precondition φ and e_1 has precondition $\top$, and where the accessibilities T are given by:

$$
\begin{aligned}
T = {} & \{\mathsf{e}_0 R(b) \mathsf{e}_0 \mid b \in B\} \cup \\
& \{\mathsf{e}_1 R(b) \mathsf{e}_1 \mid b \in B\} \cup \\
& \{\mathsf{e}_0 R(a) \mathsf{e}_1 \mid a \in N \setminus B\} \cup \\
& \{\mathsf{e}_1 R(a) \mathsf{e}_0 \mid a \in N \setminus B\}.
\end{aligned}
$$

This captures the fact that the members of B can distinguish the φ update from the $\top$ update, while the other agents (the members of $N \setminus B$) cannot. The actual event is e_0. Call this model G_φ^B.



A difference with the 'cc' case is that group messages are S5 models. Since updates of S5 models with S5 models are S5, group messages engender common knowledge (as opposed to mere common belief). Let us work out the program transformation that this update engenders for common knowledge among a group of agents D.

We will have to work out $K_{002}^{G_\varphi^B}D$, $K_{012}^{G_\varphi^B}D$, $K_{112}^{G_\varphi^B}D$, $K_{102}^{G_\varphi^B}D$.

For these, we need $K_{001}^{G_\varphi^B}D$, $K_{011}^{G_\varphi^B}D$, $K_{111}^{G_\varphi^B}D$, $K_{101}^{G_\varphi^B}D$.

For these in turn, we need $K_{000}^{G_\varphi^B}D$, $K_{010}^{G_\varphi^B}D$, $K_{110}^{G_\varphi^B}D$, $K_{100}^{G_\varphi^B}D$.

For these, we need:

$$
\begin{aligned}
T_{00}^{G_\varphi^B}D &= \bigcup_{d\in D}(?\varphi;d) = ?\varphi;D,\\
T_{01}^{G_\varphi^B}D &= \bigcup_{d\in D\setminus B}(?\varphi;d) = ?\varphi;(D\setminus B),\\
T_{11}^{G_\varphi^B}D &= D,\\
T_{10}^{G_\varphi^B}D &= D\setminus B.
\end{aligned}
$$

It follows that:

$$
\begin{aligned}
K_{000}^{G_\varphi^B}D &= ?\top \cup (?\varphi;D),\\
K_{010}^{G_\varphi^B}D &= ?\varphi;(D\setminus B),\\
K_{110}^{G_\varphi^B}D &= ?\top \cup D,\\
K_{100}^{G_\varphi^B}D &= D\setminus B.
\end{aligned}
$$

From this we can work out the K_{ij1}, as follows:

$$K_{001}^{G_\varphi^B} D = (?\varphi; D)^*,$$
$$K_{011}^{G_\varphi^B} D = (?\varphi; D)^*; ?\varphi; D \setminus B,$$
$$K_{111}^{G_\varphi^B} D = ?\top \cup D \cup (D \setminus B; (?\varphi; D)^*; ?\varphi; D \setminus B),$$
$$K_{101}^{G_\varphi^B} D = D \setminus B; (?\varphi; D)^*.$$

Finally, we get K_{002} and K_{012} from this:

$$\begin{aligned} K_{002}^{G_\varphi^B} D &= K_{001}^{G_\varphi^B} D \cup K_{011}^{G_\varphi^B} D; (K_{111}^{G_\varphi^B} D)^*; K_{101}^{G_\varphi^B} D \\ &= (?\varphi; D)^* \cup \\ &\quad (?\varphi; D)^*; ?\varphi; D \setminus B; (D \cup (D \setminus B; (?\varphi; D)^*; ?\varphi; D \setminus B))^*; \\ &\qquad\qquad D \setminus B; (?\varphi; D)^*, \\ K_{012}^{G_\varphi^B} D &= K_{011}^{G_\varphi^B} D; (K_{111}^{G_\varphi^B} D)^* \\ &= (?\varphi; D)^*; ?\varphi; D \setminus B; (D \cup (D \setminus B; (?\varphi; D)^*; ?\varphi; D \setminus B))^*. \end{aligned}$$

Abbreviating $D \cup (D \setminus B; (?\varphi; D)^*; ?\varphi; D \setminus B)$ as π, we get the following transformation for common knowledge among D after a group message to B that φ:

$$[G_\varphi^B, \mathsf{e}_0][D^*]\psi$$
$$\leftrightarrow$$
$$[(?\varphi; D)^* \cup ((?\varphi; D)^*; ?\varphi; D \setminus B; \pi^*; D \setminus B; (?\varphi; D)^*)][G_\varphi^B, \mathsf{e}_0]\psi \wedge$$
$$[(?\varphi; D)^*; ?\varphi; D \setminus B; \pi^*][G_\varphi^B, \mathsf{e}_1]\psi.$$

This formula makes it clear that, although we can translate every formula of LCC to PDL, higher order descriptions using update models are more convenient for reasoning about information change.

One interesting side-effect of this bunch of illustrations is that it demonstrates the computational character of our analysis. Indeed, the above axioms were found by a machine! Cf. [9] on the use of computational tools in exploring the universe of iterated epistemic updates.

6 Conclusion and Further Research

Dynamic-epistemic logics provide systematic means for studying exchange of factual and higher-order information. In this many-agent setting, common knowledge is an essential concept. We have presented two extended languages

for dynamic-epistemic logic that admit explicit reduction axioms for common knowledge resulting from an update: one (PAL-RC) for public announcement only, and one (LCC with its static base E-PDL) for general scenarios with information flow. These systems make proof and complexity analysis for informative actions more perspicuous than earlier attempts in the literature. Still, PAL-RC and LCC are just two extremes on a spectrum, and many further natural update logics may lie in between. We conclude by pointing out some further research topics that arise on our analysis.

- *Downward in expressive power from* E-PDL. Which weaker language fragments are in 'dynamic-static harmony', in the sense of having reduction axioms for compositional analysis of update effects, and the corresponding meaning-preserving translation? Our program transformer approach does work also with certain restrictions on tests in our full logic LCC, but we have not yet been able to identify natural intermediate levels.
- *Upward in expressive power from* E-PDL. Which *richer* languages are in dynamic-static harmony? A typical candidate is the epistemic μ*-calculus*, which allows arbitrary operators for defining smallest and greatest fixed-points. Indeed, we have a proof that the results of Section 3 extend to the calculus PAL-μ for public announcements, which takes the complete epistemic μ*-calculus* for its static language (allowing no binding into announcement positions). Our conjecture is that our expressivity and axiomatization results of Section 4 also extend to the full μ-calculus version of LCC. Cf. [7] for a first proposal.
- *Other notions of group knowledge.* Another test of our methodology via reduction axioms are further notions of group knowledge. For instance, instead of common knowledge, consider *distributed group knowledge* $D_B\varphi$ consisting of those statements which are available implicitly to the group, in the sense of φ being true at every world reachable from the current one by the intersection of all epistemic accessibility relations. The following simple reduction holds for public announcements: $[\varphi]D_B\psi \leftrightarrow (\varphi \rightarrow D_B[\varphi]\psi)$. We have not yet investigated our full system LCC extended with distributed knowledge.
- *Program constructions over update models.* One can add the usual regular operations of composition, choice, and iteration over update models, to obtain a calculus describing effects of more complex information-bearing events. It is known that this extension makes PAL undecidable, but what about partial axiomatizations in our style? One can also look at such extensions as moving toward a still richer *epistemic temporal logic* with future and past operators over universes of finite sequences of events starting from some initial model. This would be more in line with the frameworks of [13,25], to which our analysis might be generalized.
- *Alternative questions about update reasoning.* With one exception, our reduction axioms are all *schematically valid* in the sense that substituting arbitrary formulas for proposition letters again yields a valid formula. The

exception is the base clause, which really only holds for atomic proposition letters p. As discussed in [4], this means that certain schematically valid laws of update need not be derivable from our axioms in an explicit schematic manner, even though all their concrete instances will be by our completeness theorem. An example is the schematic law stating the associativity of successive announcements. It is not known whether schematic validity is decidable, even for PAL, and no complete axiomatization is known either. This is just one instance of open problems concerning PAL and its ilk for public announcement (cf. the survey in [5]), which all return for LCC, with our program transformations as a vehicle for generalizing the issues.

- *Belief revision.* Even though our language can describe agents' beliefs, the product update mechanism does not describe genuine belief revision. New information which contradicts current beliefs just leads to inconsistent beliefs. There are recent systems, however, which handle belief revision as update of plausibility rankings in models [6]. But so far, these systems only handle beliefs of single agents. Our analysis of common belief might be added on top of these, to create a more richly structured account of 'group-based belief revision'.
- *General logical perspectives.* Languages with relativizations are very common in logic. Indeed, closure under relativization is sometimes stated as a defining condition on logics in abstract model theory. Basic modal or first-order logic as they stand are closed under relativizations $[A]\varphi$, often written $(\varphi)^A$. The same is true for logics with fixed-point constructions, like PDL (cf. [3]) or the modal μ-calculus. E.g., computing a relativized least fixed-point $[A]\mu p.\varphi(p)$ works much as evaluation of $\mu p.\varphi(p) \wedge A$ – which actually suggests a corresponding dynamic epistemic reduction axiom (cf. [7]). The setting of Section 4 lifts relativization to some sort of 'update closure' for general logical languages, referring to *relative interpretations* in definable submodels of products. Languages with this property include again first-order logic and its fixed-point extensions, as well as fragments of the μ-calculus, and temporal **UNTIL** logics.

Acknowledgements Tomasz Sadzik and other participants of the *Stanford-ILLC Seminar* in December 2004 made many helpful comments and asked inspiring questions. Floris Roelofsen performed a careful proof-reading of our manuscript. Three anonymous referees of this Journal gave valuable feedback on presentation and reported errors. Thank you all.

References

[1] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. Technical Report SEN-R9922,

CWI, Amsterdam, 1999. Many updates.

[2] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.

[3] J. van Benthem. Information update as relativization. ILLC, University of Amsterdam, 2000. Available from `http://staff.science.uva.nl/~johan/upd=Rel.pdf`.

[4] J. van Benthem. One is a lonely number: on the logic of communication. Technical Report PP-2002-27, ILLC, Amsterdam, 2002. To appear in P. Koepke et al., ed. *Colloquium Logicum, Münster, 2002*, AMS Publications, Providence.

[5] J. van Benthem. Open problems in logical dynamics. ILLC Prepublication PP-2005-06, ILLC Amsterdam, 2005.

[6] J. van Benthem. Dynamic logics of belief revision. ILLC, University of Amsterdam, 2006. Available from `http://staff.science.uva.nl/~johan/DL-BR-new.pdf`.

[7] J. van Benthem, J. van Eijck, and B. Kooi. A fixed-point look at update logic, 2005. Available from `staff.science.uva.nl/~johan/Redux-Fix++.pdf`.

[8] H. P. van Ditmarsch. *Knowledge Games*. PhD thesis, ILLC, Amsterdam, 2000.

[9] J. van Eijck. Dynamic epistemic modelling. Technical Report SEN-E0424, CWI, Amsterdam, December 2004. Available from `http://db.cwi.nl/rapporten/`.

[10] J. van Eijck. Guarded actions. Technical Report SEN-E0425, CWI, Amsterdam, December 2004. Available from `http://db.cwi.nl/rapporten/`.

[11] J. van Eijck. Reducing dynamic epistemic logic to PDL by program transformation. Technical Report SEN-E0423, CWI, Amsterdam, December 2004. Available from `http://db.cwi.nl/rapporten/`.

[12] J. van Eijck, J. Ruan, and Th. Sadzik. Action emulation, 2006. Available from `www.cwi.nl/~jve/papers/06/ae/ae.pdf`.

[13] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, Massachusetts, 1995.

[14] J. Gerbrandy. *Bisimulations on Planet Kripke*. ILLC Dissertation Series, Amsterdam, 1999.

[15] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II: extensions of classical logic*, volume 165 of *Synthese Library Studies in epistemology, logic, methodology, and philosophy of science*, chapter 10, pages 497–604. D. Reidel Publishing Company, Dordrecht Boston Lancaster, 1984.

[16] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. Foundations of Computing. MIT Press, Cambridge, Massachusetts, 2000.

[17] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, N.J, 1956.

[18] B. Kooi and J. van Benthem. Reduction axioms for epistemic actions. In R. Schmidt, I Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *AiML-2004: Advances in Modal Logic*, number UMCS-04-9-1 in Technical Report Series, pages 197–211. University of Manchester, 2004.

[19] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 1981.

[20] C. Lutz. Complexity and succinctness of public announcement logic. In *Proceedings AAMAS-06: Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, Hakodate, Japan, 2006.

[21] R. van der Meyden. Constructing finite state implementations of knowledge based programs with perfect recall. In *Intelligent Agent Systems: Theoretical and Practical Issues*, volume 1209 of *Lecture Notes in Artificial Intelligence*, pages 135–152. Springer, 1997.

[22] J.-J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, Cambridge, 1995.

[23] J.S. Miller and L.S Moss. The undecidability of iterated modal relativization. *Studia Logica*, 79(3):373–407, 2005.

[24] R. Parikh. Logics of knowledge, games and dynamic logic. In M. Joseph, editor, *Foundations of Software Technology and Theoretical Computer Science*, pages 202–222. Springer, 1984.

[25] R. Parikh and R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12:453–467, 2003.

[26] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[27] J. Ruan. Exploring the update universe. Master's thesis, ILLC, Amsterdam, 2004.

[28] H. Tuominen. Translations from epistemic into dynamic logic. In Yves Kodratoff, editor, *ECAI-88: Proceedings of the 8th European conference on Artificial Intelligence*, pages 586–588, 1988.

Propositional Dynamic Logic as a Logic of Belief Revision

Jan van Eijck and Yanjing Wang

Center for Mathematics and Computer Science (CWI)
Kruislaan 413 1098 SJ Amsterdam, The Netherlands
{jve,y.wang}@cwi.nl

Abstract. This paper shows how propositional dynamic logic (PDL) can be interpreted as a logic for multi-agent belief revision. For that we revise and extend the logic of communication and change (LCC) of [9]. Like LCC, our logic uses PDL as a base epistemic language. Unlike LCC, we start out from agent plausibilities, add their converses, and build knowledge and belief operators from these with the PDL constructs. We extend the update mechanism of LCC to an update mechanism that handles belief change as relation substitution, and we show that the update part of this logic is more expressive than either that of LCC or that of doxastic/epistemic PDL with a belief change modality. It is shown that the properties of knowledge and belief are preserved under any update, and that the logic is complete.

Keywords: PDL, epistemic dynamic logic, belief revision, knowledge update.

1 Introduction

Proposals for treating belief revision in the style of dynamic epistemic logic (see Gerbrandy [15], van Ditmarsch [12], van Benthem [6,10], and Baltag, Moss and coworkers [3,1,2], or the textbook treatment in [13]) were made in [8] and [7], where it is suggested that belief revision should be treated as relation substitution. This is different from the standard action product update from [3], and it suggests that the proper relation between these two update styles should be investigated.

We propose a new version of action product update that integrates belief revision as relation substitution with belief update by means of action product. We show that this allows to express updates that cannot be expressed with action product only or with relation substitution only.

We graft this new update mechanism on a base logic that can express knowledge, safe belief, conditional belief, and plain belief, and we show that the proper relations between these concepts are preserved under any update. The completeness of our logic is also provided.

Our main source of inspiration is the logic of communication and change (LCC) from [9]. This system has the flaw that updates with non-S5 action models may

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 136–148, 2008.

destroy knowledge or belief. If one interprets the basic relations as knowledge relations, then updating with a lie will destroy the S5 character of knowledge; similarly, if one interprets the basic relations as belief, the relational properties of belief can be destroyed by malicious updates. Our redesign does not impose any relational conditions on the basic relations, so this problem is avoided. Our completeness proof is an adaptation from the completeness proof for LCC. The treatment of conditional belief derives from [11]. Our work can be seen as a proposal for integrating belief revision by means of relation substitution, as proposed in [7] with belief and knowledge update in the style of [3].

2 PDL as a Belief Revision Logic

A preference model $\mathbf{M}$ for set of agents Ag and set of basic propositions $Prop$ is a tuple (W, P, V) where W is a non-empty set of worlds, P is a function that maps each agent a to a relation P_a (the preference relation for a, with wP_aw' meaning that w' is at least as good as w), and V is a map from W to $\mathcal{P}(Prop)$ (a map that assigns to each world a $Prop$-valuation). A distinctive preference model is a pair consisting of a preference model and a set of distinctive states in that model. The intuitive idea is that the actual world is constrained to be among the distinctive worlds. This information is typically not available to the agents, for an agent's knowledge about what is actual and what is not is encoded in her P_a relation (see below).

There are no conditions at all on the P_a. Appropriate conditions will be imposed by constructing the operators for belief and knowledge by means of PDL operations.

We fix a PDL style language for talking about preference (or: plausibility). Assume p ranges over $Prop$ and a over Ag.

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

We use $PROG$ for the set of program expressions (expressions of the form π) of this language.

This is to be interpreted in the usual PDL manner, with $[\![\pi]\!]^{\mathbf{M}}$ giving the relation that interprets relational expression π in $\mathbf{M} = (W, P, V)$. $[\pi]\phi$ is true in world w of $\mathbf{M}$ if for all v with $(w, v) \in [\![\pi]\!]^{\mathbf{M}}$ it holds that ϕ is true in v. We adopt the usual abbreviations of $\bot$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$ and $\langle\pi\rangle\phi$.

The following additional abbreviations allow us to express knowledge, safe belief, conditional belief and plain belief:

Knowledge. $\sim_a$ abbreviates $(a \cup a^{\smile})^*$.
Safe belief. $\geq_a$ abbreviates a^*.
Conditional belief. $[\rightarrow_a^{\phi}]\psi$ abbreviates $\langle\sim_a\rangle\phi \rightarrow \langle\sim_a\rangle(\phi \wedge [\geq_a](\phi \rightarrow \psi))$.
Plain belief. $[\rightarrow_a]\phi$ abbreviates $[\rightarrow_a^{\top}]\phi$. (note: it follows that $[\rightarrow_a]\phi$ is equivalent to $\langle\sim_a\rangle[\geq_a]\phi$).

We will occasionally use $\leq_a$ for the converse of $\geq_a$.

Safe belief is belief that persists under revision with true information (see Stalnaker [20]). The definition of $[\rightarrow_a^\phi]\psi$ (conditional belief for a, with condition ϕ) is from Boutillier [11] This definition, also used in [5], states that conditional to ϕ, a believes in ψ if either there are no accessible ϕ worlds, or there is an accessible ϕ world in which the belief in $\phi \rightarrow \psi$ is safe. The definition of $[\rightarrow_a^\phi]\psi$ matches the well-known accessibility relations $\rightarrow_a^P$ for each subset P of the domain, given by:

$$\rightarrow_a^P := \{(x, y) \mid x \sim_a y \wedge y \in \mathrm{MIN}_{\leq_a} P\},$$

where $\mathrm{MIN}_{\leq_a} P$, the set of minimal elements of P under $\leq_a$, is defined as

$$\{s \in P : \forall s' \in P(s' \leq_a s \Rightarrow s \leq_a s')\}.$$

This logic is axiomatised by the standard PDL rules and axioms ([19,18]) plus axioms that define the meanings of the converses $a^\smile$ of basic relations a. The PDL rules and axioms are:

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

Normality	$\vdash [\pi](\phi \rightarrow \psi) \rightarrow ([\pi]\phi \rightarrow [\pi]\psi)$
Test	$\vdash [?\phi]\psi \leftrightarrow (\phi \rightarrow \psi)$
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	$\vdash (\phi \wedge [\pi^*](\phi \rightarrow [\pi]\phi)) \rightarrow [\pi^*]\phi$

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

$$\vdash \phi \rightarrow [a]\langle a^\smile\rangle\phi \qquad \vdash \phi \rightarrow [a^\smile]\langle a\rangle\phi$$

Any preference relation P_a can be turned into a pre-order by taking its reflexive transitive closure ${P_a}^*$. So our abbreviation introduces the $\geq_a$ as names for these pre-orders. The knowledge abbreviation introduces the $\sim_a$ as names for the equivalences given by $(P_a \cup {P_a}^\smile)^*$. If the P_a are well-founded, $\mathrm{MIN}_{\leq_a} P$ will be non-empty for non-empty P. Wellfoundedness of P_a is the requirement that there is no infinite sequence of different $w_1, w_2, \ldots$ with $\ldots P_a w_2 P_a w_1$. Fortunately, we do not have to worry about this relational property, for the canonical model construction for PDL yields finite models, and each relation on a finite model is well-founded.

This yields a very expressive complete and decidable PDL logic for belief revision, to which we can add mechanisms for belief update and for belief change.

Theorem 1. *The above system of belief revision PDL is complete for preference models. Since the canonical model construction for PDL yields finite models, it is also decidable.*

Knowledge is S5 (equivalence), safe belief is S4 (reflexive and transitive), plain belief is KD45 (serial, transitive and euclidean). Note that the following is valid:

$$\langle\sim_a\rangle[\geq_a]\phi \to [\sim_a]\langle\geq_a\rangle\langle\sim_a\rangle[\geq_a]\phi$$

This shows that plain belief is euclidean.

3 Action Model Update

We give the definition of action models $\mathbf{A}$ and of the update product operation $\otimes$ from Baltag, Moss, Solecki [3]. An action model is like a preference model for Ag, with the difference that the worlds are now called *actions* or *events*, and that the valuation has been replaced by a map **pre** that assigns to each event e a formula of the language called the *precondition* of e. From now on we call the preference models *static models*.

Updating a static model $\mathbf{M} = (W, P, V)$ with an action model $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre})$ succeeds if the set

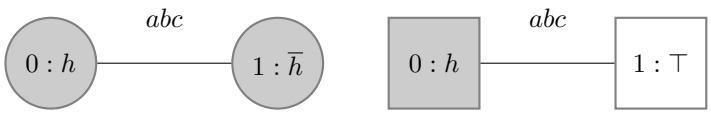
$$\{(w, e) \mid w \in W, e \in E, \mathbf{M}, w \models \mathbf{pre}(e)\}$$

is non-empty. The update result is a new static model $\mathbf{M} \otimes \mathbf{A} = (W', P', V')$ with

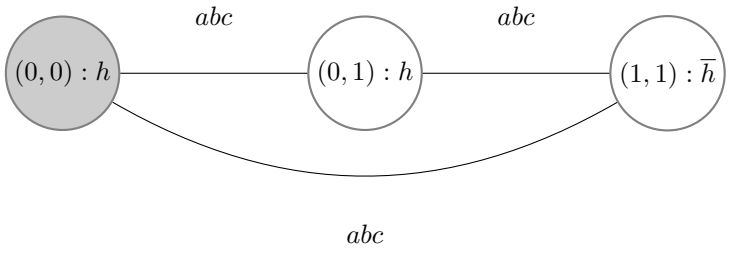
- $W' = \{(w, e) \mid w \in W, e \in E, \mathbf{M}, w \models \mathbf{pre}(e)\}$,
- P'_a is given by $\{(w, e), (v, f)) \mid (w, v) \in P_a, (e, f) \in \mathbf{P}_a\}$,
- $V'(w, e) = V(w)$.

If the static model has a set of distinctive states W_0 and the action model a set of distinctive events E_0, then the distinctive worlds of $\mathbf{M} \otimes A$ are the (w, e) with $w \in W_0$ and $e \in E_0$.

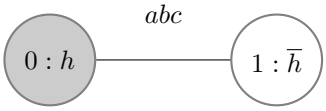
Below is an example pair of a static model with an update action. The static model, on the left, pictures the result of a hidden coin toss, with three onlookers, Alice, Bob and Carol. The model has two distinctive worlds, marked in grey; h in a world means that the valuation makes h true, $\overline{h}$ in a world means that the valuation makes h false in that world. The P_a relations for the agents are assumed to be equivalences; reflexive loops for a, b, c at each world are omitted from the picture.



The action model represents a secret test whether the result of the toss is h. The distinctive event of the update is marked grey. The P_i relations are drawn, for three agents a, b, c. The result of the update is shown here:



This result can be reduced to the bisimilar model below:



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.

4 Adding Factual Change and Belief Change

Factual change was already added to update models in LCC. We will now also add belief change. Let an action model with both changes be a quintuple.

$$A = (E, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where $E, \mathbf{P}, \mathbf{pre}$ are as before, **Sub** is a function that assigns a propositional binding to each $e \in E$, and **SUB** is a function that assigns a relational binding to each $e \in E$. A propositional substitution is a map from proposition letters to formulas, represented by a finite set of bindings.

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

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 left-hand side of a binding is mapped to itself.

Similarly, a relational substitution is a map from agents to program expressions, represented by a finite set.

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

where the a_j are agents, all different, and where the π_j are program expressions from the PDL language. It is assumed that each a that does not occur in the left-hand side of a binding is mapped to a. Use ϵ for the identity propositional or relational substitution.

Definition 1 (Update execution). *The update execution of static model* $\mathbf{M} = (W, P, V)$ *with action model* $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$ *is a tuple:* $\mathbf{M} \circledast \mathbf{A} = (W', P', V')$ *where:*

- $W' = \{(w, e) \mid \mathbf{M}, w \vDash \mathbf{pre}(e)\}$.
- P'_a *is given by*

$$\{((w_1, e_1), (w_2, e_2)) \mid \textit{there is a } \mathbf{SUB}(e_1)(a) \textit{ path from } (w_1, e_1) \textit{ to } (w_2, e_2) \textit{ in } \mathbf{M} \otimes \mathbf{A}\}.$$

- $V'(p) = \{(w, e) \in W' \mid \mathbf{M}, w \vDash \mathbf{Sub}(e)(p)\}$.

Note: the definition of P'_a refers to paths in the old style update product.

Consider the suggestive upgrade $\sharp_a \phi$ discussed in Van Benthem and Liu [8] as a relation changer (*uniform* relational substitution):

$$\sharp_a \phi =_{\text{def}} \ ?\phi; a; ?\phi \ \cup ?\neg\phi; a; ?\neg\phi \ \cup ?\neg\phi; a; ?\phi.$$

This models a kind of belief change where preference links from ϕ worlds to $\neg\phi$ worlds for agent a get deleted. It can be modelled as the following example of public belief change.

Example 1 (Public Belief Change). Action model

$$G = (\{e\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where:

- For all the $i \in Ag$, $\mathbf{P}_i = \{(e, e)\}$.
- $\mathbf{pre}(e) = \top$.
- $\mathbf{Sub}(e) = \epsilon$.
- $\mathbf{SUB}(e) = \{a \mapsto \ \sharp_a \phi, b \mapsto \ \sharp_b \phi\}$.

Note that our action model and its update execution implement the *point-wise* relation substitutions which is more powerful than merely upgrading the relations *uniformly* everywhere in the model, as the following example shows:

Example 2 (Non-public Belief Change). Action model

$$G' = (\{e_0, e_1\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where:

- For all $i \in Ag$, if $i \neq b$ then $\mathbf{P}_i = \{(e_0, e_0), (e_1, e_1)\}$, $\mathbf{P}_b = \{(e_0, e_0), (e_1, e_1), (e_0, e_1), (e_1, e_0)\}$
- $\mathbf{pre}(e_0) = \mathbf{pre}(e_1) = \top$.
- $\mathbf{Sub}(e_0) = \mathbf{Sub}(e_1) = \epsilon$.
- $\mathbf{SUB}(e_0) = \{a \mapsto \ \sharp_a \phi\}$, $\mathbf{SUB}(e_1) = \epsilon$.

Assume e_0 is the actual event.

This changes the belief of a while b remains unaware of the change.

Let PDL$^+$ be the result of adding modalities of the form $[\mathbf{A}, e]\phi$ to PDL, with the following interpretation clause:

$$\mathbf{M}, w \models [\mathbf{A}, e]\phi \ \text{ iff } \ \mathbf{M}, w \models \mathbf{pre}(e) \text{ implies } \mathbf{M} \circledast \mathbf{A}, (w, e) \models \phi.$$

Theorem 2 (Soundness and Completeness for PDL$^+$). $\vDash \phi$ *iff* $\vdash \phi$.

Proof. Completeness can be proved by a patch of the LCC completeness proof in [9] where the action modalities are pushed through program modalities by program transformations. See the first Appendix.

5 Expressivity of Action Update with Changes

Although PDL$^+$ reduces to PDL, just like LCC, the new action update mechanism (the model transformation part) is more *expressive* than classic product update and product update with factual changes, as we will show in this section. Call a function on epistemic models that is invariant for bisimulation a model transformer. Then each update can be viewed as a model transformer, and a set of model transformers corresponds to an update mechanism. If U is an update mechanism, let $\mathrm{Tr}(U)$ be its set of model transformers.

Definition 2. *Update mechanism U_1 is* less expressive *than update mechanism U_2 if $Tr(U_1) \subset Tr(U_2)$.*

First note that the classical product update (with factual changes) has the *eliminative* nature for relational changing: according to the definition, the relations in the updated model must come from relations in the static model. For example, it is not possible, by product update, to introduce a relational link for agent a to a static model where the a relation was empty. However, we can easily do this with an uniform relation substitution $a \mapsto ?\top$. Thus we have:

Proposition 1. *Relational substitution can express updates that cannot be expressed with action product update(with factual changes) alone, so relational substitution is not less expressive than action product update.*

On the other hand, relational substitution alone cannot add worlds into a static model, while the classical product update mechanism can copy sets of worlds. Therefore it is not hard to see:

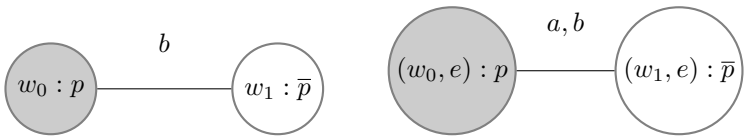
Proposition 2. *Action product update can express updates that cannot be expressed with relational substitution alone, so action product update is not less expressive than relational substitution.*

Our action update with both relational and factual changes combines the power of product update and propositional/relational substitutions. Thus according to Propositions 1 and 2, it is more expressive than relational eliminative product update with factual changes in LCC, and more expressive than propositional/relation changing substitution *simpliciter*. Moreover, we can prove a even stronger result for the case of $S5$ updates.

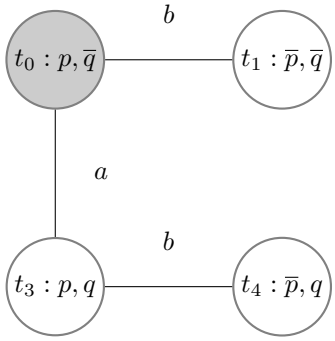
Theorem 3. *In the class of $S5$ model transformers, action product update with factual changes is less expressive than action update with both factual and relational changes.*

Proof. Let $\mathbf{A}$ be the action model $(\{e\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$ where $\mathbf{P}_a = \{(e, e)\} = \mathbf{P}_b; \mathbf{pre}(e) = \top; \mathbf{Sub}(e) = \epsilon; \mathbf{SUB}(e) = \{a \mapsto b\}$. It is easy to see that this action model will change the relation a as b uniformly while keeping the updated model being still $S5$, if the static model is indeed $S5$. We now show that it does not have a corresponding action model in LCC style (only factual changes) which can give the bisimilar updated result for every static model.

First consider the following static $S5$ model $\mathbf{M}$ (left) and its updated model $\mathbf{M} \circledast \mathbf{A}$(right) (reflexive loops are omitted):



For a contradiction, suppose there is a LCC action model $\mathbf{A}'$ with distinctive event e' such that $M_1 \circledast \mathbf{A}, (w_0, e) \underline{\leftrightarrow} \mathbf{M} \otimes \mathbf{A}', (w_0, e')$. Then according to the definition of bisimulation, there must be an $a-$link from (w_0, e') to a $\overline{p}$ world (s, e'') in $\mathbf{M} \otimes \mathbf{A}'$. According to the definition of $\otimes$, $(w_0, s) \in p_a$ in $\mathbf{M}$ and $(e', e'') \in \mathbf{P}_a$ in $\mathbf{A}'$. Thus $s = w_0$ and $\mathbf{M}, w_0 \vDash pre(e') \wedge pre(e'')$. Let us consider the following $S5$ model $\mathbf{M}'$ which consists of two copies of $\mathbf{M}$ with an $a-$link in between:



where q does not show up in $pre(e')$ and $pre(e'')$. Thus it is not hard to see that $pre(e') \wedge pre(e')$ holds on t_0 and t_3. Then $\mathbf{M}' \otimes \mathbf{A}', (t_0, e')$ must has an a link from a $\overline{q}$ world (t_0, e') to a q world (t_3, e''), while in $\mathbf{M}' \circledast \mathbf{A}, (t_0, e)$ there is no such link. Thus $\mathbf{M}' \circledast \mathbf{A}, (t_0, e)$ and $\mathbf{M}' \otimes \mathbf{A}', (t_0, e')$ are not bisimilar. Contradiction.

An example illustrating the use of the new belief revision update mechanism is worked out in the second Appendix. This example also shows the difference in expressive power for the achievement of common knowledge between knowledge update and belief revision.

6 Future Work

Several update mechanisms for dynamic epistemic logic have been proposed in the literature. A very expressive one is the action-priority upgrade proposed in [4,5]. Comparing the expressiveness of our update with factual and relation change with that of their mechanism is future work.

The new update mechanism proposed above is grafted on a doxastic/epistemic logic that does not impose any conditions on the basic preference relations. Thus, any update will result in a proper epistemic model. This situation changes as

soon as one imposes further conditions. E.g., if the basic preferences are assumed to be locally connected, then one should restrict the class of update models to those that preserve this constraint. For each reasonable constraint, there is a corresponding class of model transformers that preserve this constraint. Finding syntactic characterizations of these classes is future work.

We are interested in **model checking** with doxastic/epistemic PDL and updates/upgrades in the new style, and we are currently investigating its complexity. We intend to use the logic, and the new update/upgrade mechanism, in the next incarnation of the epistemic model checker DEMO [14].

Acknowledgments. We have profited from discussions with Johan van Benthem, Hans van Ditmarsch and Barteld Kooi, and we thank two anonymous referees for their comments. The second author is supported by the Dutch Organisation for Scientic Research (NWO) under research grant no. 612.000.528 (VEMPS).

References

1. Baltag, A.: A logic for suspicious players: epistemic action and belief-updates in games. Bulletin of Economic Research 54(1), 1–45 (2002)
2. Baltag, A., Moss, L.S.: Logics for epistemic programs. Synthese 139(2), 165–224 (2004)
3. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Bilboa, I. (ed.) Proceedings of TARK 1998, pp. 43–56 (1998)
4. Baltag, A., Smets, S.: Conditional doxastic models: A qualitative approach to dynamic belief revision. Electronic Notes in Theoretical Computer Science (ENTCS) 165, 5–21 (2006)
5. Baltag, A., Smets, S.: A qualitative theory of dynamic interactive belief revision. In: Bonanno, G., van der Hoek, W., Wooldridge, M. (eds.) Texts in Logic and Games. Amsterdam University Press (to appear, 2008)
6. van Benthem, J.: Language, logic, and communication. In: van Benthem, J., Dekker, P., van Eijck, J., de Rijke, M., Venema, Y. (eds.) Logic in Action, pp. 7–25. ILLC (2001)
7. van Benthem, J.: Dynamic logic for belief revision. Journal of Applied Non-Classical Logics 2, 129–155 (2007)
8. van Benthem, J., Liu, F.: Dynamic logic of preference upgrade. Journal of Applied Non-Classical Logics 14(2) (2004)
9. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation 204(11), 1620–1662 (2006)
10. van Benthem, J.: One is a lonely number: on the logic of communication. Technical Report PP-2002-27, ILLC, Amsterdam (2002)
11. Boutilier, C.: Toward a logic of qualitative decision theory. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) Proceedings of the 4th International Conference on Principle of Knowledge Representation and Reasoning (KR 1994), pp. 75–86. Morgan Kaufmann, San Francisco (1994)
12. van Ditmarsch, H.: Knowledge Games. PhD thesis, ILLC, Amsterdam (2000)

13. van Ditmarsch, H.P., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2006)
14. van Eijck, J.: DEMO — a demo of epistemic modelling. In: van Benthem, J., Gabbay, D., Löwe, B. (eds.) Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop, number 1 in Texts in Logic and Games, pp. 305–363. Amsterdam University Press (2007)
15. Gerbrandy, J.: Bisimulations on planet Kripke. PhD thesis, ILLC (1999)
16. Gray, J.: Notes on database operating systems. In: Bayer, R., Graham, R.M., Seegmuller, G. (eds.) Operating Systems: an advanced course. LNCS, vol. 66, Springer, Berlin (1978)
17. Halpern, J.Y., Moses, Y.O.: Knowledge and common knowledge in a distributed environment. In: Proceedings 3rd ACVM Symposium on Distributed Computing, pp. 50–68 (1984)
18. Kozen, D., Parikh, R.: An elementary proof of the completeness of PDL. Theoretical Computer Science 14, 113–118 (1981)
19. Segerberg, K.: A completeness theorem in the modal logic of programs. In: Traczyck, T. (ed.) Universal Algebra and Applications, pp. 36–46. Polish Science Publications (1982)
20. Stalnaker, R.C.: On logics of knowledge and belief. Philosophical Studies 128, 169–199 (2006)

Appendix 1: Soundness and Completeness of PDL$^+$

To define the proper program transformation for PDL$^+$ we need a function $^\cup$ that maps each PDL program to its converse (in the obvious sense that the interpretation of $\pi^\cup$ is the converse of that of π):

$$\begin{array}{ll} (a^\smile)^\cup & = a \\ (?\phi)^\cup & = ?\phi \\ (\pi_1;\pi_2)^\cup & = {\pi_2}^\cup;{\pi_1}^\cup \\ (\pi_1 \cup \pi_2)^\cup & = {\pi_1}^\cup \cup {\pi_2}^\cup \\ (\pi^*)^\cup & = (\pi^\cup)^* \end{array}$$

What is needed to get a completeness proof is a redefinition of the epistemic program transformation operation $T^{\mathbf{A}}_{ij}$ used in the LCC completeness to push an action model modality $[\mathbf{A}, e]$ through an epistemic program modality $[\pi]$.

$$\begin{array}{rcl} \underline{T}^{\mathbf{A}}_{ij}(a) & = & \begin{cases} ?pre(e_i);\mathbf{SUB}(e_i)(a) & \text{if } e_i \mapsto_{\mathbf{SUB}(e_i)(a)} e_j \text{ in } \mathbf{A} \\ ?\bot & \text{otherwise} \end{cases} \\ \underline{T}^{\mathbf{A}}_{ij}(a^\smile) & = & \begin{cases} ?pre(e_i);(\mathbf{SUB}(e_i)(a))^\cup & \text{if } e_i \mapsto_{(\mathbf{SUB}(e_i)(a))^\cup} e_j \text{ in } \mathbf{A} \\ ?\bot & \text{otherwise} \end{cases} \\ \underline{T}^{\mathbf{A}}_{ij}(?\phi) & = & \begin{cases} ?(pre(e_i) \wedge [\mathbf{A}, e_i]\phi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \\ \underline{T}^{\mathbf{A}}_{ij}(\pi_1;\pi_2) & = & \bigcup_{k=0}^{n-1}(\underline{T}^{\mathbf{A}}_{ik}(\pi_1);\underline{T}^{\mathbf{A}}_{kj}(\pi_2)) \\ \underline{T}^{\mathbf{A}}_{ij}(\pi_1 \cup \pi_2) & = & \underline{T}^{\mathbf{A}}_{ij}(\pi_1) \cup \underline{T}^{\mathbf{A}}_{ij}(\pi_2)) \\ \underline{T}^{\mathbf{A}}_{ij}(\pi^*) & = & K^{\mathbf{A}}_{ijn}(\pi) \end{array}$$

where it is assumed that the action model $\mathbf{A}$ has n states, and the states are numbered $0, \ldots, n-1$. $K^{\mathbf{A}}_{ijn}$ is the Kleene path transformer, as in [9].

The proof system for PDL$^+$ consists of all axioms and rules of LCC except the reduction axiom:

$$[\mathbf{A}, e_i][\pi]\phi \leftrightarrow \bigwedge_{j=0}^{n-1} [T^{\mathbf{A}}_{ij}(\pi)][\mathbf{A}, e_j]\phi.$$

In addition, PDL$^+$ has the axioms for converse atomic programs as in section 2, and reduction axioms of the form:

$$[\mathbf{A}, e_i][\pi]\phi \leftrightarrow \bigwedge_{j=0}^{n-1} [\underline{T}^{\mathbf{A}}_{ij}(\pi)][\mathbf{A}, e_j]\phi.$$

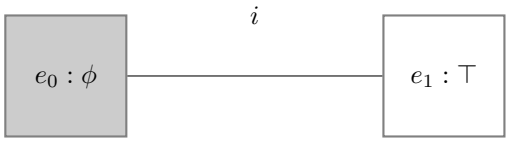
This is the patch we need to prove the completeness result (Theorem 2).

Appendix 2: Restricted Announcements Versus Restricted Belief Changes

A restricted announcement of ϕ is an announcement of ϕ that is not delivered to one of the agents i. Notation $!\phi^{-i}$. The action model for $!\phi^{-i}$ has event set $\{e_0, e_1\}$, with e_0 the actual event, where e_0 has precondition ϕ and e_1 precondition $\top$, and with the preference relation given by

$$P_i = \{(e_0, e_0), (e_1, e_1), (e_0, e_1), (e_1, e_0)\},$$

and $P_j = \{(e_0, e_0), (e_1, e_1)\}$ for all $j \neq i$.



A protocol for restricted announcements, for epistemic situation M, is a set of finite sequences of formula-agent pairs, such that each sequence

$$(\phi_0, i_0), \ldots, (\phi_n, i_n)$$

has the following property:

$$\forall k \in \mathbb{N} : 0 \leq k < n \rightarrow \exists i \in Ag : \mathbf{M}, w \models [!\phi_0^{-i_0}], \ldots, [!\phi_{k-1}^{-i_{k-1}}][\sim_i]\phi_k.$$

Intuitively, at every stage in the sequence of restricted announcements, some agent has to possess the required knowledge to make the next announcement in the sequence. We can now prove that such protocols can never establish common knowledge of purely propositional facts.

Theorem 4. *Let C express common knowledge among set of agents Ag. Let* $\mathbf{M}$ *be an epistemic model with actual world w such that* $\mathbf{M}, w \models \neg C\phi$*, with ϕ purely propositional. Then there is no protocol with*

$$\mathbf{M}, w \models [!\phi_0^{-i_0}], \ldots, [!\phi_n^{-i_n}]C\phi.$$

for any sequence $(\phi_0, i_0), \ldots, (\phi_n, i_n)$ in the protocol.

Proof. We show that $\neg C\phi$ is an invariant of any restricted announcement.

Assume $\mathbf{M}, w \models \neg C\phi$. Let $(\mathbf{A}, e)$ be an action model for announcement $!\psi^{-i}$, the announcement of ψ, restricted to $Ag - \{i\}$. Then $\mathbf{A}$ has events e and e', with $\mathbf{pre}(e) = \psi$ and $\mathbf{pre}(e') = \top$. If $\mathbf{M}, w \models \neg\psi$ then the update does not succeed, and there is nothing to prove. Suppose therefore that $\mathbf{M}, w \models \psi$. Since $\mathbf{pre}(e') = \top$, the model $\mathbf{M} \otimes \mathbf{A}$ restricted to domain $D = \{(w, e') \mid w \in W_{\mathbf{M}}\}$ is a copy of the original model $\mathbf{M}$. Thus, it follows from $\mathbf{M}, w \models \neg C\phi$ that

$$\mathbf{M} \otimes A \restriction D, (w, e') \models \neg C\phi.$$

Thus, there is an C-accessible world-event pair (w', e'') in D with

$$\mathbf{M} \otimes A \restriction D, (w', e'') \models \neg\phi.$$

Since ϕ is purely propositional, we get from this that:

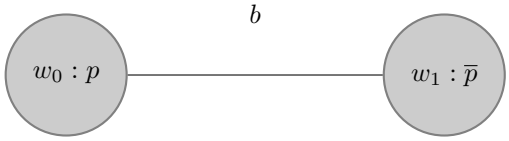
$$\mathbf{M} \otimes A, (w', e'') \models \neg\phi.$$

Observe that since common knowledge is preserved under model restriction, absence of common knowledge is preserved under model extension. The C-accessible world-event pair (w', e'') in $\mathbf{M} \otimes A \restriction D$ will still be C-accessible in $\mathbf{M} \otimes A$. Therefore, it follows that $\mathbf{M} \otimes A, (w, e') \models \neg C\phi$. By the construction of $\mathbf{M} \otimes A$, we get from this that $\mathbf{M} \otimes A, (w, e) \models \langle i\rangle\neg C\phi$, and therefore $\mathbf{M} \otimes A, (w, e) \models \neg C\phi$, by the definition of common knowledge.

It follows immediately that no protocol built from restricted announcements can create common knowledge of propositional facts.

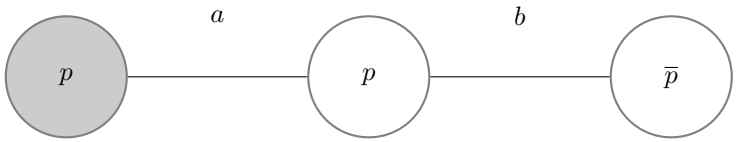
The case of the two generals planning a coordinated attack on the enemy, but failing to achieve common knowledge about it [16,17] can be viewed as a special case of this theorem.

If there are just two agents i, j, the only way for agent i to send a restricted message is by allowing uncertainty about the delivery. If i, j are the only agents, and i knows ϕ then the restricted message $!\phi^{-j}$ conveys no information, so the only reasonable restricted announcement of ϕ is $!\phi^{-i}$. The upshot of this announcement is that the message gets delivered to j, but i remains uncertain about this. According to the theorem, such messages cannot create common knowledge. Initial situation:

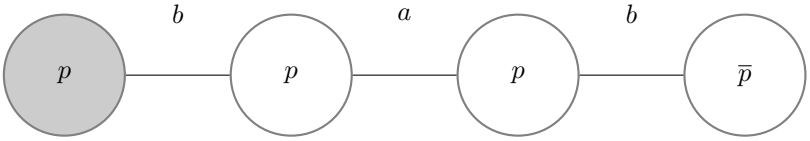


Update action for general a (left) and general b (right):

Situation after first message from general a:

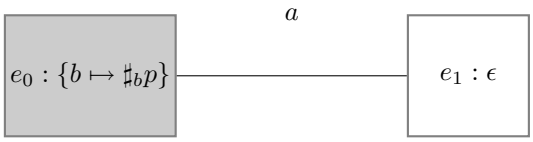


Situation after update by a followed by update by b:

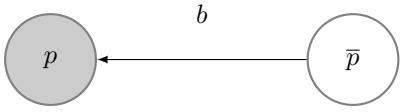


And so on ...

Now look at the case where restricted announcements are replaced by non-public belief revisions. Then the power of restricted belief change turns up in the following example. We start out from the initial situation again, and we update using the action model for non-public belief change:



Here is the update result (after minimalisation under bisimulation):



The example shows that it is possible to achieve common safe belief in p in a single step, by means of a non-public belief change.

DEMO — A Demo of Epistemic Modelling

Jan van Eijck*
CWI, Kruislaan 413, 1098 SJ Amsterdam

May 22, 2007

Abstract

This paper introduces and documents *DEMO*, a Dynamic Epistemic Modelling tool. *DEMO* allows modelling epistemic updates, graphical display of update results, graphical display of action models, formula evaluation in epistemic models, translation of dynamic epistemic formulas to PDL formulas. Also, DEMO implements the reduction of dynamic epistemic logic [22, 2, 3, 1] to PDL given in [17] and presented in the context of generic languages for communication and change in [6]. Epistemic models are minimized under bisimulation, and update action models are simplified under action emulation (an appropriate structural notion for having the same update effect, cf. [19]). The paper is an exemplar of tool building for epistemic update logic. It contains the essential code of an implementation in Haskell [27], in 'literate programming' style [28], of *DEMO*.

Keywords: Knowledge representation, epistemic updates, dynamic epistemic modelling, action models. information change, logic of communication.

ACM Classification (1998) E 4, F 4.1, H 1.1.

Introduction

In this introduction we will demonstrate how *DEMO*, which is short for *Dynamic Epistemic MOdelling*,[1] can be used to check semantic intuitions about what goes on in epistemic update situations.[2] For didactic purposes, the initial examples have been kept extremely simple. Although the situation of message passing about just two basic propositions with just three epistemic agents already reveals many subtleties, the reader should bear in mind that *DEMO* is capable of modelling much more complex situations.

In a situation where you and I know nothing about a particular aspect of the state of the world (about whether p and q hold, say), our state of knowledge is modelled by a Kripke model where the worlds are the four different possibilities for the truth of p and q ($\emptyset$, p, q, pq), your epistemic accessibility relation $\sim_a$ is the total relation on these four possibilities, and mine $\sim_b$ is the total relation on these four possibilities as well. There is also c, who like the two of us, is completely ignorant about p and q. This initial model is generated by *DEMO* as follows.

```
DEMO> showM (initE [P 0,Q 0] [a,b,c])
```

*Other affiliations: Uil-OTS, Janskerkhof 13, 3512 BL Utrecht and NIAS, Meijboomlaan 1, 2242 PR Wassenaar

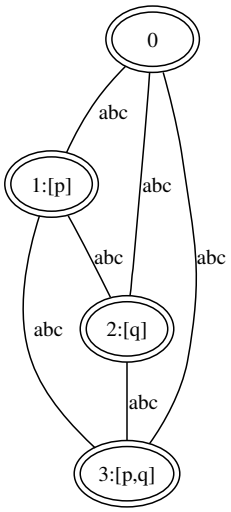
[1] Or short for *DEMO of Epistemic MOdelling*, for those who prefer co-recursive acronyms.

[2] The program source code is available from `http://www.cwi.nl/~jve/demo/`.

```
==> [0,1,2,3]
[0,1,2,3]
(0,[])(1,[p])(2,[q])(3,[p,q])
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
(c,[[0,1,2,3]])
```

Here `initE` generates an initial epistemic model, and `showM` shows that model in an appropriate form, in this case in the partition format that is made possible by the fact that the epistemic relations are all equivalences.

As an example of a different kind of representation, let us look at the picture that can be generated with *dot* [30] from the file produced by the DEMO command `writeP "filename" (initE [P 0,Q 0])`.



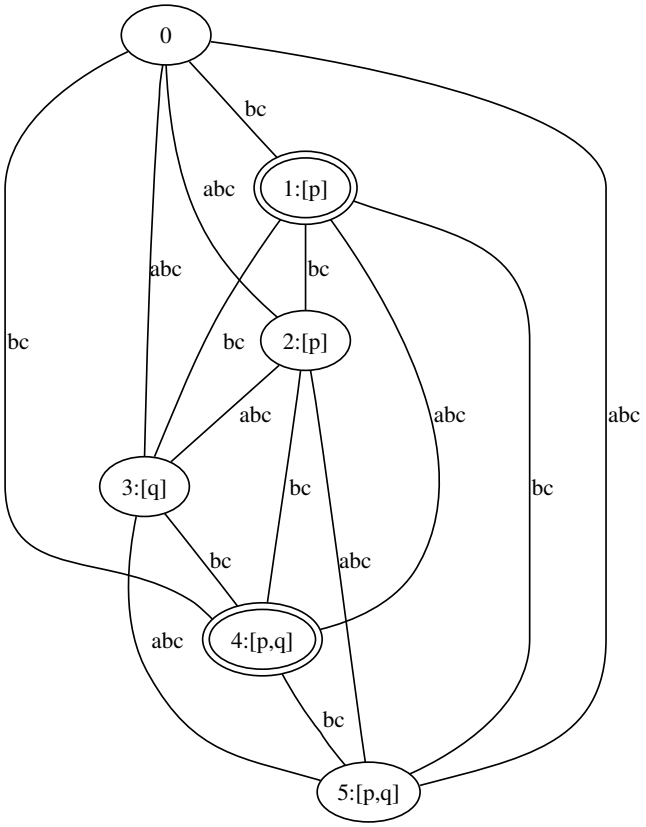
This is a model where none of the three agents a, b or c can distinguish between the four possibilities about p and q. *DEMO* shows the partitions generated by the accessibility relations $\sim_a, \sim_b, \sim_c$. Since these three relations are total, the three partitions each consist of a single block. Call this model `e0`.

Now suppose a wants to know whether p is the case. She asks whether p and receives a truthful answer from somebody who is in a position to know. This answer is conveyed to a in a message. b and c have heard a's question, and so are aware of the fact that an answer may have reached a. b and c have seen *that* an answer was delivered, but they don't know which answer. This is not a secret communication, for b and c know that a has inquired about p. The situation now changes as follows:

```
DEMO> showM (upd e0 (message a p))
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[[0,2,3,5],[1,4]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
```

Note that `upd` is a function for updating an epistic model with (a representation of) a communicative action. In this case, the result is again a model where the three accessibility relations are equivalences,

but one in which a has restricted her range of possibilities to $1, 4$ (these are worlds where p is the case), while for b and c all possibilities are still open. Note that this epistemic model has two 'actual worlds': this means that there are two possibilities that are compatible with 'how things really are'. In graphical display format these 'actual worlds' show up as double ovals:



DEMO also allows us to display the action models corresponding to the epistemic updates. For the present example (we have to indicate that we want the action model for the case where $\{a, b, c\}$ is the set of relevant agents):

```
showM ((message a p) [a,b,c])
==> [0]
[0,1]
(0,p)(1,T)
(a,[[0],[1]])
(b,[[0,1]])
(c,[[0,1]])
```

Notice that in the result of updating the initial situation with this message, some subtle things have changed for b and c as well. Before the arrival of the message, $\Box_b(\neg\Box_a p \wedge \neg\Box_a\neg p)$ was true, for b knew that a did not know about p. But now b has heard a's question about p, and is aware of the fact that an answer has reached a. So in the new situation b knows that a knows about p. In other words, $\Box_b(\Box_a p \vee \Box_a\neg p)$ has become true. On the other hand it is still the case that b knows that a knows nothing about q: $\Box_b\neg\Box_a q$ is still true in the new situation. The situation for c is similar to that for b. These things can be checked in *DEMO* as follows:

```
DEMO> isTrue (upd e0 (message a p)) (K b (Neg (K a q)))
True
DEMO> isTrue (upd e0 (message a p)) (K b (Neg (K a p)))
False
```

If you receive the same message about p twice, the second time the message gets delivered has no further effect. Note the use of `upds` for a sequence of updates.

```
DEMO> showM (upds e0 [message a p, message a p])
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[[0,2,3,5],[1,4]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
```

Now suppose that the second action is a message informing b about p:

```
DEMO> showM (upds e0 [message a p, message b p])
==> [1,6]
[0,1,2,3,4,5,6,7,8,9]
(0,[])(1,[p])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[p,q])(7,[p,q])(8,[p,q])(9,[p,q])

(a,[[0,3,4,5,8,9],[1,2,6,7]])
(b,[[0,2,4,5,7,9],[1,3,6,8]])
(c,[[0,1,2,3,4,5,6,7,8,9]])
```

The graphical representation of this model is slightly more difficult to fathom at a glance. See Figure 1. In this model a and b both know about p, but they do not know about each other's knowledge about p. c still knows nothing, and both a and b know that c knows nothing. Both $\Box_a\Box_b p$ and $\Box_b\Box_a p$ are false in this model. $\Box_a\neg\Box_b p$ and $\Box_b\neg\Box_a p$ are false as well, but $\Box_a\neg\Box_c p$ and $\Box_b\neg\Box_c p$ are true.

```
DEMO> isTrue (upds e0 [message a p, message b p]) (K a (K b p))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (K a p))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (Neg (K b p)))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (Neg (K c p)))
True
```

The order in which a and b are informed does not matter:

```
DEMO> showM (upds e0 [message b p, message a p])
==> [1,6]
[0,1,2,3,4,5,6,7,8,9]
(0,[])(1,[p])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[p,q])(7,[p,q])(8,[p,q])(9,[p,q])

(a,[[0,2,4,5,7,9],[1,3,6,8]])
(b,[[0,3,4,5,8,9],[1,2,6,7]])
(c,[[0,1,2,3,4,5,6,7,8,9]])
```

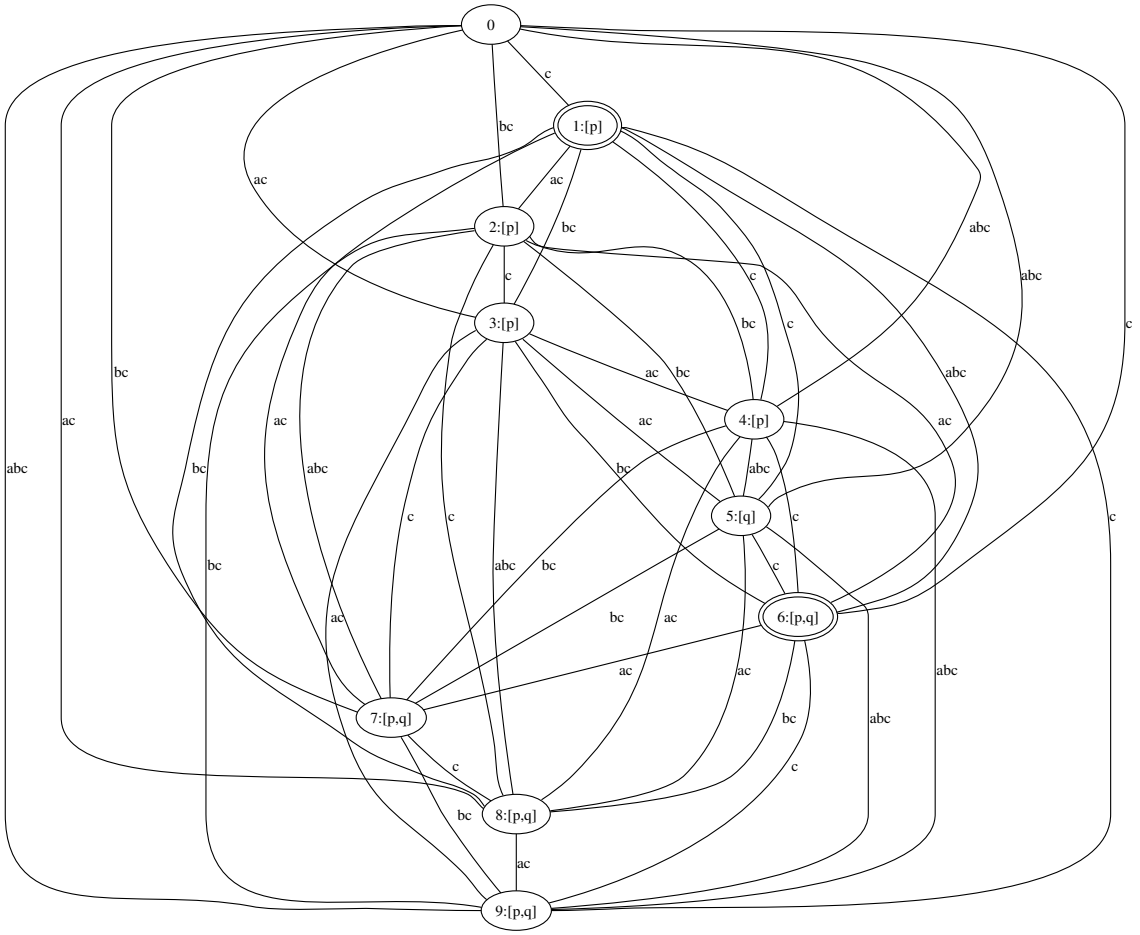


Figure 1: Situation after second message

Modulo renaming this is the same as the earlier result. The example shows that the epistemic effects of distributed message passing are quite different from those of a public announcement or a group message.

```
DEMO> showM (upd e0 (public p))
==> [0,1]
[0,1]
(0,[p])(1,[p,q])
(a,[[0,1]])
(b,[[0,1]])
(c,[[0,1]])
```

The result of the public announcement that p is that a, b and c are informed that p and about each other's knowledge about p.

DEMO allows to compare the action models for public announcement and individual message passing:

```
DEMO> showM ((public p) [a,b,c])
```

```
==> [0]
[0]
(0,p)
(a,[[0]])
(b,[[0]])
(c,[[0]])

DEMO> showM ((cmp [message a p, message b p, message c p]) [a,b,c])
==> [0]
[0,1,2,3,4,5,6,7]
(0,p)(1,p)(2,p)(3,p)(4,p)
(5,p)(6,p)(7,T)
(a,[[0,1,2,3],[4,5,6,7]])
(b,[[0,1,4,5],[2,3,6,7]])
(c,[[0,2,4,6],[1,3,5,7]])
```

Here `cmp` gives the sequential composition of a list of communicative actions. This involves, among other things, computation of the appropriate preconditions for the combined action model.

More subtly, the situation is also different from a situation where a, b receive the same message that p, with a being aware of the fact that b receives the message and vice versa. Such group messages create common knowledge.

```
DEMO> showM (groupM [a,b] p [a,b,c])
==> [0]
[0,1]
(0,p)(1,T)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
```

The difference with the case of the two separate messages is that now a and b are aware of each other's knowledge that p:

```
DEMO> isTrue (upd e0 (groupM [a,b] p)) (K a (K b p))
True
DEMO> isTrue (upd e0 (groupM [a,b] p)) (K b (K a p))
True
```

In fact, this awareness goes on, for arbitrary nestings of $\Box_a$ and $\Box_b$, which is what common knowledge means. Common knowledge can be checked directly, as follows:

```
DEMO> isTrue (upd e0 (groupM [a,b] p)) (CK [a,b] p)
True
```

It is also easily checked in DEMO that in the case of the separate messages no common knowledge is achieved.

Next, look at the case where two separate messages reach a and b, one informing a that p and the other informing b that $\neg q$:

```
DEMO> showM (upds e0 [message a p, message b (Neg q)])
==> [2]
```

```
[0,1,2,3,4,5,6,7,8]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[p])(6,[q])(7,[p,q])(8,[p,q])
(a,[[0,1,4,5,6,8],[2,3,7]])
(b,[[0,2,4],[1,3,5,6,7,8]])
(c,[[0,1,2,3,4,5,6,7,8]])
```

Again the order in which these messages are delivered is immaterial for the end result, as you should expect:

```
DEMO> showM (upds e0 [message b (Neg q), message a p])
==> [2]
[0,1,2,3,4,5,6,7,8]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[p])(6,[q])(7,[p,q])(8,[p,q])
(a,[[0,1,3,5,6,8],[2,4,7]])
(b,[[0,2,3],[1,4,5,6,7,8]])
(c,[[0,1,2,3,4,5,6,7,8]])
```

Modulo a renaming of worlds, this is the same as the previous result.

The logic of public announcements and private messages is related to the logic of knowledge, with [24] as the pioneer publication. This logic satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a\varphi \Rightarrow \Box_a\psi)$ (if a knows that φ implies ψ, and she knows φ, then she also knows ψ),
- positive introspection $\Box_a\varphi \Rightarrow \Box_a\Box_a\varphi$ (if a knows φ, then a knows that she knows φ),
- negative introspection $\neg\Box_a\varphi \Rightarrow \Box_a\neg\Box_a\varphi$ (if a does not know φ, then she knows that she does not know),
- truthfulness $\Box_a\varphi \Rightarrow \varphi$ (if a knows φ then φ is true).

As is well known, the first of these is valid on all Kripke frames, the second is valid on precisely the transitive Kripke frames, the third is valid on precisely the euclidean Kripke frames (a relation R is euclidean if it satisfies $\forall x\forall y\forall z((xRy \wedge xRz) \Rightarrow yRz)$), and the fourth is valid on precisely the reflexive Kripke frames. A frame satisfies transitivity, euclideanness and reflexivity iff it is an equivalence relation, hence the logic of knowledge is the logic of the so-called S5 Kripke frames: the Kripke frames with an equivalence $\sim_a$ as epistemic accessibility relation. Multi-agent epistemic logic extends this to multi-S5, with an equivalence $\sim_b$ for every $b \in B$, where b is the set of epistemic agents.

Now suppose that instead of open messages, we use *secret* messages. If a secret message is passed to a, b and c are not even aware that any communication is going on. This is the result when a receives a secret message that p in the initial situation:

```
DEMO> showM (upd e0 (secret [a] p))
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[([],[0,2,3,5]),([],[1,4])])
(b,[([1,4],[0,2,3,5])])
(c,[([1,4],[0,2,3,5])])
```

This is not an S5 model anymore. The accessibility for a is still an equivalence, but the accessibility for b is lacking the property of reflexivity. The worlds $1, 4$ that make up a's conceptual space (for these are the worlds accessible for a from the actual worlds $1, 4$) are precisely the worlds where the b and c arrows are *not* reflexive. b enters his conceptual space from the vantage points 1 and 4, but b does not see these vantage points itself. Similarly for c. In the *DEMO* representation, the list `([1,4],[0,2,3,5])` gives the entry points `[1,4]` into conceptual space `[0,2,3,5]`.

The secret message has no effect on what b and c believe about the facts of the world, but it has effected b's and c's beliefs about the beliefs of a in a disastrous way. These beliefs have become inaccurate. For instance, b now believes that a does *not* know that p, but he is mistaken! The formula $\Box_b \neg \Box_a p$ is true in the actual worlds, but $\neg \Box_a p$ is false in the actual worlds, for a *does* know that p, because of the secret message. Here is what *DEMO* says about the situation (`isTrue` evaluates a formula in all of the actual worlds of an epistemic model):

```
DEMO> isTrue (upd e0 (secret [a] p)) (K b (Neg (K a p)))
True
DEMO> isTrue (upd e0 (secret [a] p)) (Neg (K a p))
False
```

This example illustrates a regress from the world of knowledge to the world of consistent belief: the result of the update with a secret propositional message does not satisfy the postulate of truthfulness anymore.

The logic of consistent belief satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a \varphi \Rightarrow \Box_a \psi)$,
- positive introspection $\Box_a \varphi \Rightarrow \Box_a \Box_a \varphi$,
- negative introspection $\neg \Box_a \varphi \Rightarrow \Box_a \neg \Box_a \varphi$,
- consistency $\Box_a \varphi \Rightarrow \Diamond_a \varphi$ (if a believes that φ then there is a world where φ is true, i.e., φ is consistent).

Consistent belief is like knowledge, except for the fact that it replaces the postulate of truthfulness $\Box_a \varphi \Rightarrow \varphi$ by the weaker postulate of consistency.

Since the postulate of consistency determines the serial Kripke frames (a relation R is serial if $\forall x \exists y\ xRy$), the principles of consistent belief determine the Kripke frames that are transitive, euclidean and serial, the so-called KD45 frames.

In the conceptual world of secrecy, inconsistent beliefs are not far away. Suppose that a, after having received a secret message informing her about p, sends a message to b to the effect that $\Box_a p$. The trouble is that this is *inconsistent* with what b believes.

```
DEMO> showM (upds e0 [secret [a] p, message b (K a p)])
==> [1,5]
[0,1,2,3,4,5,6,7]
(0,[])(1,[p])(2,[p])(3,[p])(4,[q])
(5,[p,q])(6,[p,q])(7,[p,q])
(a,([],[([],[0,3,4,7]),([],[1,2,5,6])]))
(b,([1,5],[([2,6],[0,3,4,7])]))
(c,([],[([1,2,5,6],[0,3,4,7])]))
```

This is not a KD45 model anymore, for it lacks the property of seriality for b's belief relation. b's belief contains two isolated worlds $1, 5$. Since 1 is the actual world, this means that b's belief state has become inconsistent: from now on, b will believe *anything*.

So we have arrived at a still weaker logic. The logic of possibly inconsistent belief satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a\varphi \Rightarrow \Box_a\psi)$,
- positive introspection $\Box_a\varphi \Rightarrow \Box_a\Box_a\varphi$,
- negative introspection $\neg\Box_a\varphi \Rightarrow \Box_a\neg\Box_a\varphi$.

This is the logic of K45 frames: frames that are transitive and euclidean.

In [16] some results and a list of questions are given about the possible deterioration of knowledge and belief caused by different kind of message passing. E.g., the result of updating an S5 model with a public announcement or a non-secret message, if defined, is again S5. The result of updating an S5 model with a secret message to some of the agents, if defined, need not even be KD45. One can prove that the result is KD45 iff the model we start out with satisfies certain epistemic conditions. The update result always is K45. Such observations illustrate why S5, KD45 and K45 are ubiquitous in epistemic modelling. See [7, 23] for general background on modal logic, and [9, 20] for specific background on these systems.

If this introduction has convinced the reader that the logic of public announcements, private messages and secret communications is rich and subtle enough to justify the building of the conceptual modelling tools to be presented in the rest of the report, then it has served its purpose.

In the rest of the report, we first fix a formal version of epistemic update logic as an implementation goal. After that, we are ready for the implementation.

Further information on various aspects of dynamic epistemic logic is provided in [1, 2, 4, 5, 12, 20, 21, 29].

Design

DEMO is written in a high level functional programming language Haskell [27]. Haskell is a non-strict, purely-functional programming language named after Haskell B. Curry. The design is modular. Operations on lists and characters are taken from the standard Haskell `List` and `Char` modules. The following modules are part of DEMO:

Models The module that defines general models over a number of agents. In the present implemenation these are A through E. It turns out that more than five agents are seldom needed in epistemic modelling. *General models* have variables for their states and their state adornments. By letting the state adornments be valuations we get *Kripke models*, by letting them be formulas we get *update models*.

MinBis The module for minimizing models under bisimulation by means of partition refinement.

Display The module for displaying models in various formats. Not discussed in this paper.

ActEpist The module that specializes general models to action models and epistemic models. Formulas may contain action models as operators. Action models contain formulas. The definition of formulas is therefore also part of this module.

DPLL Implementation of Davis, Putnam, Logemann, Loveland (DPLL) theorem proving [10, 11] for propositional logic. The implementation uses discrimination trees or *tries*, following [33]. This is used for formula simplification. Not discussed in this paper.

Semantics Implementation of the key semantic notions of epistemic update logic. It handles the mapping from communicative actions to action models.

DEMO Main module.

Main Module

1 Main Module Declaration

```
module DEMO
    (
     module List,
     module Char,
     module Models,
     module Display,
     module MinBis,
     module ActEpist,
     module DPLL,
     module Semantics
    )
    where

import List
import Char
import Models
import Display
import MinBis
import ActEpist
import DPLL
import Semantics
```

2 Version

The first version of *DEMO* was written in March 2004. This version was extended in May 2004 with an implementation of automata and a translation function from epistemic update logic to Automata PDL. In September 2004, I discovered a direct reduction of epistemic update logic to PDL [17]. This motivated a switch to a PDL-like language, with extra modalities for action update and automata update. I decided to leave in the automata for the time being, for nostalgic reasons.

In Summer 2005, several example modules with DEMO programs for epistemic puzzles (some of them contributed by Ji Ruan) and for checking of security protocols (with contributions by Simona Orzan) were added, and the program was rewritten in a modular fashion.

In Spring 2006, automata update was removed, and in Autumn 2006 the code was refactored for the present report.

```
version :: String
version = "DEMO 1.06, Autumn 2006"
```

Definitions

3 Models and Updates

In this section we formalize the version of dynamic epistemic logic that we are going to implement.

Let p range over a set of basic propositions P and let a range over a set of agents Ag. Then the language of PDL over P, Ag is given by:

$$\begin{array}{rcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

Employ the usual abbreviations: $\bot$ is shorthand for $\neg\top$, $\varphi_1 \vee \varphi_2$ is shorthand for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2$ is shorthand for $\neg(\varphi_1 \wedge \varphi_2)$, $\varphi_1 \leftrightarrow \varphi_2$ is shorthand for $(\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, and $\langle\pi\rangle\varphi$ is shorthand for $\neg[\pi]\neg\varphi$. Also, if $B \subseteq Ag$ and B is finite, use B as shorthand for $b_1 \cup b_2 \cup \cdots$. Under this convention, formulas for expressing general knowledge $E_B\varphi$ take the shape $[B]\varphi$, while formulas for expressing common knowledge $C_B\varphi$ appear as $[B^*]\varphi$, i.e., $[B]\varphi$ expresses that it is general knowledge among agents B that φ, and $[B^*]\varphi$ expresses that it is common knowledge among agents B that φ. In the special case where $B = \emptyset$, B turns out equivalent to $?\bot$, the program that always fails.

The semantics of PDL over P, Ag is given relative to labelled transition systems $\mathbf{M} = (W, V, R)$, where W is a set of worlds (or states), $V : W \to \mathcal{P}(P)$ is a valuation function, and $R = \{\xrightarrow{a} \subseteq W \times W \mid a \in Ag\}$ is a set of labelled transitions, i.e., binary relations on W, one for each label a. In what follows, we will take the labeled transitions for a to represent the epistemic alternatives of an agent a.

The formulae of PDL are interpreted as subsets of $W_{\mathbf{M}}$ (the state set of $\mathbf{M}$), the actions of PDL as binary relations on $W_{\mathbf{M}}$, as follows:

$$\begin{array}{rcl} [\![\top]\!]^{\mathbf{M}} & = & W_{\mathbf{M}} \\ [\![p]\!]^{\mathbf{M}} & = & \{w \in W_{\mathbf{M}} \mid p \in V_{\mathbf{M}}(w)\} \\ [\![\neg\varphi]\!]^{\mathbf{M}} & = & W_{\mathbf{M}} - [\![\varphi]\!]^{\mathbf{M}} \\ [\![\varphi_1 \wedge \varphi_2]\!]^{\mathbf{M}} & = & [\![\varphi_1]\!]^{\mathbf{M}} \cap [\![\varphi_2]\!]^{\mathbf{M}} \\ [\![[\pi]\varphi]\!]^{\mathbf{M}} & = & \{w \in W_{\mathbf{M}} \mid \forall v(\text{ if } (w,v) \in [\![\pi]\!]^{\mathbf{M}} \text{ then } v \in [\![\varphi]\!]^{\mathbf{M}})\} \\ \\ [\![a]\!]^{\mathbf{M}} & = & \xrightarrow{a}_{\mathbf{M}} \\ [\![?\varphi]\!]^{\mathbf{M}} & = & \{(w,w) \in W_{\mathbf{M}} \times W_{\mathbf{M}} \mid w \in [\![\varphi]\!]^{\mathbf{M}}\} \\ [\![\pi_1;\pi_2]\!]^{\mathbf{M}} & = & [\![\pi_1]\!]^{\mathbf{M}} \circ [\![\pi_2]\!]^{\mathbf{M}} \\ [\![\pi_1 \cup \pi_2]\!]^{\mathbf{M}} & = & [\![\pi_1]\!]^{\mathbf{M}} \cup [\![\pi_2]\!]^{\mathbf{M}} \\ [\![\pi^*]\!]^{\mathbf{M}} & = & ([\![\pi]\!]^{\mathbf{M}})^* \end{array}$$

If $w \in W_{\mathbf{M}}$ then we use $\mathbf{M} \models_w \varphi$ for $w \in [\![\varphi]\!]^{\mathbf{M}}$.

[3] proposes to model epistemic actions as epistemic models, with valuations replaced by preconditions. See also: [4, 5, 12, 17, 20, 21, 29, 32].

Action models for a given language $\mathcal{L}$ Let a set of agents Ag and an epistemic language $\mathcal{L}$ be given. An action model for $\mathcal{L}$ is a triple $A = ([s_0, \ldots, s_{n-1}], \mathrm{pre}, T)$ where $[s_0, \ldots, s_{n-1}]$ is a finite list of action states, $\mathrm{pre} : \{s_0, \ldots, s_{n-1}\} \to \mathcal{L}$ assigns a precondition to each action state, and $T : Ag \to \mathcal{P}(\{s_0, \ldots, s_{n-1}\}^2)$ assigns an accessibility relation $\stackrel{a}{\to}$ to each agent $a \in Ag$.

A pair $\mathbf{A} = (A, s)$ with $s \in \{s_0, \ldots, s_{n-1}\}$ is a pointed action model, where s is the action that actually takes place.

The list ordering of the action states in an action model will play an important role in the definition of the program transformations associated with the action models.

In the definition of action models, $\mathcal{L}$ can be any language that can be interpreted in PDL models. Actions can be executed in PDL models by means of the following product construction:

Action Update Let a PDL model $\mathbf{M} = (W, V, R)$, a world $w \in W$, and a pointed action model (A, s), with $A = ([s_0, \ldots, s_{n-1}], \mathrm{pre}, T)$, be given. Suppose $w \in [\![\mathrm{pre}(s)]\!]^{\mathbf{M}}$. Then the result of executing (A, s) in $(\mathbf{M}, w)$ is the model $(\mathbf{M} \otimes A, (w, s))$, with $\mathbf{M} \otimes A = (W', V', R')$, where

$$\begin{aligned} W' &= \{(w, s) \mid s \in \{s_0, \ldots, s_{n-1}\}, w \in [\![\mathrm{pre}(s)]\!]^{\mathbf{M}}\} \\ V'(w, s) &= V(w) \\ R'(a) &= \{((w, s), (w', s')) \mid (w, w') \in R(a), (s, s') \in T(a)\}. \end{aligned}$$

In case there is a set of actual worlds and a set of actual actions, the definition is similar: those world/action pairs survive where the world satisfies the preconditions of the action. See below.

The language of $\mathrm{PDL}^{\mathrm{DEL}}$ (update PDL) is given by extending the PDL language with update constructions $[A, s]\varphi$, where (A, s) is a pointed action model. The interpretation of $[A, s]\varphi$ in $\mathbf{M}$ is given by:

$$[\![[A, s]\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \text{if } \mathbf{M} \models_w \mathrm{pre}(s) \text{ then } (w, s) \in [\![\varphi]\!]^{\mathbf{M} \otimes A}\}.$$

Using $\langle A, s\rangle\varphi$ as shorthand for $\neg[A, s]\neg\varphi$, we see that the interpretation for $\langle A, s\rangle\varphi$ turns out as:

$$[\![\langle A, s\rangle\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \mathbf{M} \models_w \mathrm{pre}(s) \text{ and } (w, s) \in [\![\varphi]\!]^{\mathbf{M} \otimes A}\}.$$

Updating with multiple pointed update actions is also possible. A multiple pointed action is a pair (A, S), with A an action model, and S a subset of the state set of A. Extend the language with updates $[A, S]\varphi$, and interpret this as follows:

$$[\![[A, S]\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \forall s \in S(\text{ if } \mathbf{M} \models_w \mathrm{pre}(s) \text{ then } \mathbf{M} \otimes A \models_{(w,s)} \varphi)\}.$$

In [17] it is shown how dynamic epistemic logic can be reduced to PDL by program transformation. Each action model $\mathbf{A}$ has associated program transformers $T^{\mathbf{A}}_{ij}$ for all states s_i, s_j in the action model, such that the following hold:

Lemma 1 (Program Transformation, Van Eijck [17]) *Assume A has n states $s_0, \ldots, s_{n-1}$. Then:*

$$\mathbf{M} \models_w [A, s_i][\pi]\varphi \text{ iff } \mathbf{M} \models_w \bigwedge_{j=0}^{n-1} [T^A_{ij}(\pi)][A, s_j]\varphi.$$

This lemma allows a reduction of dynamic epistemic logic to PDL, a reduction that we will implement in the code below.

4 Operations on Action Models

Sequential Composition If $(\mathbf{A}, S)$ and $(\mathbf{B}, T)$ are multiple pointed action models, their sequential composition $(\mathbf{A}, S) \odot (\mathbf{B}, T)$ is given by:

$$(\mathbf{A}, S) \odot (\mathbf{B}, T) := ((W, \mathrm{pre}, R), S \times T),$$

where

- $W = W_{\mathbf{A}} \times W_{\mathbf{B}}$,
- $\mathrm{pre}(s, t) = \mathrm{pre}(s) \wedge \langle \mathbf{A}, S \rangle \mathrm{pre}(t)$,
- R is given by: $(s, t) \xrightarrow{a} (s', t') \in R$ iff $s \xrightarrow{a} s' \in R_{\mathbf{A}}$ and $t \xrightarrow{a} t' \in R_{\mathbf{B}}$.

The unit element for this operation is the action model

$$\mathbf{1} = ((\{0\}, 0 \mapsto \top, \{0 \xrightarrow{a} 0 \mid a \in Ag\}), \{0\}).$$

Updating an arbitary epistemic model $\mathbf{M}$ with $\mathbf{1}$ changes nothing.

Non-deterministic Sum The non-deterministic sum $\oplus$ of multiple-pointed action models $(\mathbf{A}, S)$ and $(\mathbf{B}, T)$ is the action model $(\mathbf{A}, S) \oplus (\mathbf{B}, T)$ is given by:

$$(\mathbf{A}, S) \oplus (\mathbf{B}, T) := ((W, \mathrm{pre}, R), S \uplus T),$$

where $\uplus$ denotes disjoint union, and where

- $W = W_{\mathbf{A}} \uplus W_{\mathbf{B}}$,
- $\mathrm{pre} = \mathrm{pre}_{\mathbf{A}} \uplus \mathrm{pre}_{\mathbf{B}}$,
- $R = R_{\mathbf{A}} \uplus R_{\mathbf{B}}$.

The unit element for this operation is called $\mathbf{0}$: the multiple pointed action model given by $((\emptyset, \emptyset, \emptyset), \emptyset)$.

5 Logics for Communication

Here are some specific action models that can be used to define various languages of communication.

Public announcement of φ: action model $(\mathbf{S}, \{0\})$, with

$$S_{\mathbf{S}} = \{0\}, p_{\mathbf{S}} = 0 \mapsto \varphi, R_{\mathbf{S}} = \{0 \xrightarrow{a} 0 \mid a \in A\}.$$

Individual message to b **that** φ: action model $(\mathbf{S}, \{0\})$, with

$$S_{\mathbf{S}} = \{0, 1\}, p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top, R_{\mathbf{S}} = \{0 \xrightarrow{b} 0, 1 \xrightarrow{b} 1\} \cup \{0 \sim_a 1 \mid a \in A - \{b\}\}$$

Group message to B **that** φ: action model $(\mathbf{S}, \{0\})$, with

$$S_{\mathbf{S}} = \{0, 1\}, p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top, R_{\mathbf{S}} = \{0 \sim_a 1 \mid a \in A - B\}.$$

Secret individual communication to b that φ: action model $(\mathbf{S}, \{0\})$, with

$$\begin{array}{lcl} S_{\mathbf{S}} & = & \{0,1\}, \\ p_{\mathbf{S}} & = & 0 \mapsto \varphi, 1 \mapsto \top, \\ R_{\mathbf{S}} & = & \{0 \stackrel{b}{\to} 0\} \cup \{0 \stackrel{a}{\to} 1 \mid a \in A - \{b\}\} \cup \{1 \stackrel{a}{\to} 1 \mid a \in A\}. \end{array}$$

Secret group communication to B that φ: action model $(\mathbf{S}, \{0\})$, with

$$\begin{array}{lcl} S_{\mathbf{S}} & = & \{0,1\}, \\ p_{\mathbf{S}} & = & 0 \mapsto \varphi, 1 \mapsto \top, \\ R_{\mathbf{S}} & = & \{0 \stackrel{b}{\to} 0 \mid b \in B\} \cup \{0 \stackrel{a}{\to} 1 \mid a \in A - B\} \cup \{1 \stackrel{a}{\to} 1 \mid a \in A\}. \end{array}$$

Test of φ: action model $(\mathbf{S}, \{0\})$, with

$$S_{\mathbf{S}} = \{0,1\}, p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top, R_{\mathbf{S}} = \{0 \stackrel{a}{\to} 1 \mid a \in A\} \cup \{1 \stackrel{a}{\to} 1 \mid a \in A\}.$$

Individual revelation to b of a choice from $\{\varphi_1, \ldots, \varphi_n\}$: action model $(\mathbf{S}, \{1, \ldots, n\})$, with

$$\begin{array}{lcl} S_{\mathbf{S}} & = & \{1, \ldots, n\}, \\ p_{\mathbf{S}} & = & 1 \mapsto \varphi_1, \ldots, n \mapsto \varphi_n, \\ R_{\mathbf{S}} & = & \{s \stackrel{b}{\to} s \mid s \in S_{\mathbf{S}}\} \cup \{s \stackrel{a}{\to} s' \mid s, s' \in S_{\mathbf{S}}, a \in A - \{b\}\}. \end{array}$$

Group revelation to B of a choice from $\{\varphi_1, \ldots, \varphi_n\}$: action model $(\mathbf{S}, \{1, \ldots, n\})$, with

$$\begin{array}{lcl} S_{\mathbf{S}} & = & \{1, \ldots, n\}, \\ p_{\mathbf{S}} & = & 1 \mapsto \varphi_1, \ldots, n \mapsto \varphi_n, \\ R_{\mathbf{S}} & = & \{s \stackrel{b}{\to} s \mid s \in S_{\mathbf{S}}, b \in B\} \cup \{s \stackrel{a}{\to} s' \mid s, s' \in S_{\mathbf{S}}, a \in A - B\}. \end{array}$$

Transparant informedness of B about φ: action model $(\mathbf{S}, \{0,1\})$, with

$$\begin{array}{lcl} S_{\mathbf{S}} & = & \{0,1\}, \\ p_{\mathbf{S}} & = & 0 \mapsto \varphi, 1 \mapsto \neg\varphi, \\ R_{\mathbf{S}} & = & \{0 \stackrel{a}{\to} 0 \mid a \in A\} \cup \{0 \stackrel{a}{\to} 1 \mid a \in A - B\} \cup \{1 \stackrel{a}{\to} 0 \mid a \in A - B\} \cup \{1 \stackrel{a}{\to} 1 \mid a \in A\}. \end{array}$$

Transparant informedness of B about φ is the special case of a group revelation ot B of a choice from $\{\varphi, \neg\varphi\}$. Note that all but the revelation action models and the transparant informedness action models are single pointed (their sets of actual states are singletons).

The language for the logic of group announcements:

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \bigwedge[\varphi_1, \ldots, \varphi_n] \mid \bigvee[\varphi_1, \ldots, \varphi_n] \mid \Box_a\varphi \mid E_B\varphi \mid C_B\varphi \mid \ [\pi]\varphi$$

$$\pi \quad ::= \quad \mathbf{1} \mid \mathbf{0} \mid \text{public } B\ \varphi \mid \odot[\pi_1, \ldots, \pi_n] \mid \oplus[\pi_1, \ldots, \pi_n]$$

Semantics for this: use the semantics of **1**, **0**, **public** B φ, and the operations on multiple pointed action models from Section 4.

The logic of tests and group announcements: as above, but now also allowing tests $?\varphi$ as basic programs. Semantics: add the semantics of $?\varphi$ to the above repertoire.

The logic of individual messages: as above, but now the basic actions are messages to individual agents. Semantics: start out from the semantics of **message** a φ.

The logic of tests, group announcements, and group revelations as above, but now also allowing revelations from alternatives. Semantics: use the semantics of **reveal** B $\{\varphi_1, \ldots, \varphi_n\}$.

Kripke Models

6 Module Declaration

```
module Models where

import List
```

7 Agents

```
data Agent = A | B | C | D | E deriving (Eq,Ord,Enum,Bounded)
```

Give the agents appropriate names:

```
a, alice, b, bob, c, carol, d, dave, e, ernie  :: Agent
a = A; alice = A
b = B; bob   = B
c = C; carol = C
d = D; dave  = D
e = E; ernie = E
```

Make agents showable in an appropriate way:

```
instance Show Agent where
  show A = "a"; show B = "b"; show C = "c"; show D = "d" ; show E = "e"
```

8 Model Datatype

It will prove useful to generalize over states. We first define general models, and then specialize to action models and epistemic models. In the following definition, `state` and `formula` are variables over types. We assume that each model carries a list of distinguished states.

```
data Model state formula = Mo
                           [state]
                           [(state,formula)]
                           [Agent]
                           [(Agent,state,state)]
                           [state]
                           deriving (Eq,Ord,Show)
```

Decomposing a pointed model into a list of single-pointed models:

```
decompose ::  Model state formula -> [Model state formula]
decompose (Mo states pre agents rel points) =
  [ Mo states pre agents rel [point] | point <- points ]
```

It is useful to be able to map the precondition table to a function. Here is general tool for that. Note that the resulting function is partial; if the function argument does not occur in the table, the value is undefined.

```
table2fct :: Eq a => [(a,b)] -> a -> b
table2fct t = \ x -> maybe undefined id (lookup x t)
```

Another useful utility is a function that creates a partition out of an equivalence relation:

```
rel2part :: (Eq a) => [a] -> (a -> a -> Bool) -> [[a]]
rel2part [] r = []
rel2part (x:xs) r = xblock : rel2part rest r
  where
  (xblock,rest) = partition (\ y -> r x y) (x:xs)
```

The *domain* of a model is its list of states:

```
domain :: Model state formula -> [state]
domain (Mo states _ _ _ _) = states
```

The *eval* of a model is its list of state/formula pairs:

```
eval :: Model state formula -> [(state,formula)]
eval (Mo _ pre _ _ _) = pre
```

The *agentList* of a model is its list of agents:

```
agentList :: Model state formula -> [Agent]
agentList (Mo _ _ ags _ _) = ags
```

The *access* of a model is its labelled transition component:

```
access :: Model state formula -> [(Agent,state,state)]
access (Mo _ _ _ rel _) = rel
```

The distinguished points of a model:

```
points :: Model state formula -> [state]
points (Mo _ _ _ _ pnts) = pnts
```

When we are looking at models, we are only interested in generated submodels, with as their domain the distinguished state(s) plus everything that is reachable by an accessibility path.

```
gsm :: Ord state => Model state formula ->  Model state formula
gsm (Mo states pre ags rel points) = (Mo states' pre' ags rel' points)
   where
   states' = closure rel ags points
   pre'    = [(s,f)     | (s,f)     <- pre,
                           elem s states'                  ]
   rel'    = [(ag,s,s') | (ag,s,s') <- rel,
                           elem s states',
                           elem s' states'                 ]
```

The closure of a state list, given a relation and a list of agents:

```
closure ::  Ord state =>
             [(Agent,state,state)] -> [Agent] -> [state] -> [state]
closure rel agents xs
  | xs' == xs = xs
  | otherwise = closure rel agents xs'
     where
     xs' = (nub . sort) (xs ++ (expand rel agents xs))
```

The expansion of a relation R given a state set S and a set of agents B is given by $\{t \mid s \xrightarrow{b} t \in R, s \in S, b \in B\}$. Implementation:

```
expand :: Ord state =>
           [(Agent,state,state)] -> [Agent] -> [state] -> [state]
expand rel agnts ys =
      (nub . sort . concat)
         [ alternatives rel ag state | ag    <- agnts,
                                       state <- ys       ]
```

The epistemic alternatives for agent a in state s are the states in sR_a (the states reachable through R_a from s):

```
alternatives :: Eq state =>
                [(Agent,state,state)] -> Agent -> state -> [state]
alternatives rel ag current =
  [ s' | (a,s,s') <- rel, a == ag, s == current ]
```

Model Minimization under Bisimulation

9 Module Declaration

```
module MinBis where

import List
import Models
```

10 Partition Refinement

Any Kripke model can be simplified by replacing each state s by its bisimulation class $[s]$. The problem of finding the smallest Kripke model modulo bisimulation is similar to the problem of minimizing the number of states in a finite automaton [26]. We will use partition refinement, in the spirit of [31]. Here is the algorithm:

- Start out with a partition of the state set where all states with the same precondition function are in the same class. The equality relation to be used to evaluate the precondition function is given as a parameter to the algorithm.
- Given a partition Π, for each block b in Π, partition b into sub-blocks such that two states s, t of b are in the same sub-block iff for all agents a it holds that s and t have $\xrightarrow{a}$ transitions to states in the same block of Π. Update Π to Π' by replacing each b in Π by the newly found set of sub-blocks for b.
- Halt as soon as $\Pi = \Pi'$.

Looking up and checking of two formulas against a given equivalence relation:

```
lookupFs :: (Eq a,Eq b) => a -> a -> [(a,b)] -> (b -> b -> Bool) -> Bool
lookupFs i j table r = case lookup i table of
  Nothing -> lookup j table == Nothing
  Just f1 -> case lookup j table of
      Nothing -> False
      Just f2 -> r f1 f2
```

Computing the initial partition, using a particular relation for equivalence of formulas:

```
initPartition :: (Eq a, Eq b) => Model a b -> (b -> b -> Bool) -> [[a]]
initPartition (Mo states pre ags rel points) r =
  rel2part states (\ x y -> lookupFs x y pre r)
```

Refining a partition:

```
refinePartition :: (Eq a, Eq b) =>
                     Model a b -> [[a]] -> [[a]]
refinePartition m p = refineP m p p
  where
  refineP :: (Eq a, Eq b) => Model a b -> [[a]] -> [[a]] -> [[a]]
  refineP m part [] = []
  refineP m part (block:blocks) =
     newblocks ++ (refineP m part blocks)
       where
         newblocks =
           rel2part block (\ x y -> sameAccBlocks m part x y)
```

Function that checks whether two states have the same accessible blocks under a partition:

```
sameAccBlocks :: (Eq a, Eq b) =>
         Model a b -> [[a]] -> a -> a -> Bool
sameAccBlocks m@(Mo states pre ags rel points) part s t =
    and [ accBlocks m part s ag == accBlocks m part t  ag |
                                                ag <- ags ]
```

The accessible blocks for an agent from a given state, given a model and a partition:

```
accBlocks :: (Eq a, Eq b) =>
               Model a b -> [[a]] -> a -> Agent -> [[a]]
accBlocks m@(Mo states pre ags rel points) part s ag =
    nub [ bl part y | (ag',x,y) <- rel, ag' == ag, x == s ]
```

The block of an object in a partition:

```
bl :: Eq a => [[a]] -> a -> [a]
bl part x = head (filter (elem x) part)
```

Initializing and refining a partition:

```
initRefine :: (Eq a, Eq b) =>
                Model a b -> (b -> b -> Bool) -> [[a]]
initRefine m r = refine m (initPartition m r)
```

The refining process:

```
refine :: (Eq a, Eq b) => Model a b -> [[a]] -> [[a]]
refine m part = if rpart == part
                      then part
                      else refine m rpart
  where rpart = refinePartition m part
```

11 Minimization

Use this to construct the minimal model. Notice the dependence on relational parameter `r`.

```
minimalModel :: (Eq a, Ord a, Eq b, Ord b) =>
                (b -> b -> Bool) -> Model a b -> Model [a] b
minimalModel r m@(Mo states pre ags rel points) =
  (Mo states' pre' ags rel' points')
     where
     partition = initRefine m r
     states'   = partition
     f         = bl partition
     rel'      = (nub.sort) (map (\ (x,y,z) -> (x, f y, f z)) rel)
     pre'      = (nub.sort) (map (\ (x,y)   -> (f x, y))      pre)
     points'   =  map f points
```

Converting `a`'s into integers, using their position in a given list of `a`'s.

```
convert :: (Eq a, Show a) => [a] -> a  -> Integer
convert = convrt 0
  where
  convrt :: (Eq a, Show a) => Integer -> [a] -> a -> Integer
  convrt n []     x = error (show x ++ " not in list")
  convrt n (y:ys) x | x == y    = n
                    | otherwise = convrt (n+1) ys x
```

Converting an object of type `Model a b` into an object of type `Model Integer b`:

```
conv ::  (Eq a, Show a) =>
           Model a b -> Model Integer b
conv  (Mo worlds val ags acc points) =
      (Mo (map f worlds)
          (map (\ (x,y)   -> (f x, y)) val)
           ags
          (map (\ (x,y,z) -> (x, f y, f z)) acc))
          (map f points)
  where f = convert worlds
```

Use this to rename the blocks into integers:

```
bisim ::  (Eq a, Ord a, Show a, Eq b, Ord b) =>
           (b -> b -> Bool) -> Model a b -> Model Integer b
bisim r = conv . (minimalModel r)
```

Formulas, Action Models and Epistemic Models

12 Module Declaration

```
module ActEpist
where

import List
import Models
import MinBis
import DPLL
```

Module `List` is a standard Haskell module. Module `Models` is described in Chapter 5, and Module `MinBis` in Chapter 8. Module `DPLL` refers to an implementation of Davis, Putnam, Logemann, Loveland (DPLL) theorem proving (not included in this document, but available at `www.cwi.nl/~jve/demo`).

13 Formulas

Basic propositions:

```
data Prop = P Int | Q Int | R Int deriving (Eq,Ord)
```

Show these in the standard way, in lower case, with index 0 omitted.

```
instance Show Prop where
  show (P 0) = "p"; show (P i) = "p" ++ show i
  show (Q 0) = "q"; show (Q i) = "q" ++ show i
  show (R 0) = "r"; show (R i) = "r" ++ show i
```

Formulas, according to the definition:

$$\begin{array}{lcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \bigwedge[\varphi_1,\ldots,\varphi_n] \mid \bigvee[\varphi_1,\ldots,\varphi_n] \mid [\pi]\varphi \mid \ [\mathbf{A}]\varphi \\ \pi & ::= & a \mid B \mid ?\varphi \mid \bigcirc[\pi_1,\ldots,\pi_n] \mid \bigcup[\pi_1,\ldots,\pi_n] \mid \pi^* \end{array}$$

Here, p ranges over basic propositions, a ranges over agents, B ranges over non-empty sets of agents, and $\mathbf{A}$ is a multiple pointed action model (see below) $\bigcirc$ denotes sequential composition of a list of programs. We will often write $\bigcirc[\pi_1,\pi_2]$ as $\pi_1;\pi_2$, and $\bigcup[\pi_1,\pi_2]$ as $\pi_1 \cup \pi_2$.

Note that general knowledge among agents B that φ is expressed in this language as $[B]\varphi$, and common knowledge among agents B that φ as $[B^*]\varphi$. Thus, $[B]\varphi$ can be viewed as shorthand for $[\bigcup_{b\in B} b]\varphi$. In case $B = \emptyset$, $[B]\varphi$ turns out to be equivalent to $[?\bot]\varphi$.

For convenience, we have also left in the more traditional way of expressing individual knowledge $\Box_a\varphi$, general knowledge $E_B\varphi$ and common knowledge $C_B\varphi$.

```
data Form = Top
          | Prop Prop
          | Neg  Form
          | Conj [Form]
          | Disj [Form]
          | Pr Program Form
          | K Agent Form
          | EK [Agent] Form
          | CK [Agent] Form
          | Up AM Form
          deriving (Eq,Ord)
```

```
data Program = Ag Agent
             | Ags [Agent]
             | Test Form
             | Conc [Program]
             | Sum  [Program]
             | Star Program
             deriving (Eq,Ord)
```

Some useful abbreviations:

```
impl :: Form -> Form -> Form
impl form1 form2 = Disj [Neg form1, form2]

equiv :: Form -> Form -> Form
equiv form1 form2 = Conj [form1 `impl` form2, form2 `impl` form1]

xor :: Form -> Form -> Form
xor x y = Disj [ Conj [x, Neg y], Conj [Neg x, y]]
```

The negation of a formula:

```
negation :: Form -> Form
negation (Neg form) = form
negation form       = Neg form
```

Show formulas in the standard way:

```
instance Show Form where
  show Top = "T" ; show (Prop p) = show p; show (Neg f) = '-':(show f);
  show (Conj fs)     = '&': show fs
  show (Disj fs)     = 'v': show fs
  show (Pr p f)      = '[': show p ++ "]" ++ show f
  show (K agent f)   = '[': show agent ++ "]" ++ show f
  show (EK agents f) = 'E': show agents ++ show f
  show (CK agents f) = 'C': show agents ++ show f
  show (Up pam f)    = 'A': show (points pam) ++ show f
```

Show programs in a standard way:

```
instance Show Program where
  show (Ag a)       = show a
  show (Ags as)     = show as
  show (Test f)     = '?': show f
  show (Conc ps)    = 'C': show ps
  show (Sum ps)     = 'U': show ps
  show (Star p)     = '(': show p ++ ")*"
```

Programs can get very unwieldy very quickly. As is well known, there is no normalisation procedure for regular expressions. Still, here are some rewriting steps for simplification of programs:

$$
\begin{aligned}
\emptyset &\rightarrow ?\bot \\
?\varphi_1 \cup ?\varphi_2 &\rightarrow ?(\varphi_1 \vee \varphi_2) \\
?\bot \cup \pi &\rightarrow \pi \\
\pi \cup ?\bot &\rightarrow \pi \\
\bigcup[\pi_1, \ldots, \pi_k, \bigcup[\pi_{k+1}, \ldots, \pi_{k+m}], \pi_{k+m+1}, \ldots, \pi_{k+m+n}] &\rightarrow \bigcup[\pi_1, \ldots, \pi_{k+m+n}] \\
\bigcup[] &\rightarrow ?\bot \\
\bigcup[\pi] &\rightarrow \pi \\
?\varphi_1; ?\varphi_2 &\rightarrow ?(\varphi_1 \wedge \varphi_2) \\
?\top; \pi &\rightarrow \pi \\
\pi; ?\top &\rightarrow \pi \\
?\bot; \pi &\rightarrow ?\bot \\
\pi; ?\bot &\rightarrow ?\bot \\
\bigcirc[\pi_1, \ldots, \pi_k, \bigcirc[\pi_{k+1}, \ldots, \pi_{k+m}], \pi_{k+m+1}, \ldots, \pi_{k+m+n}] &\rightarrow \bigcirc[\pi_1, \ldots, \pi_{k+m+n}] \\
\bigcirc[] &\rightarrow ?\top \\
\bigcirc[\pi] &\rightarrow \pi \\
(?\varphi)^* &\rightarrow ?\top \\
(?\varphi \cup \pi)^* &\rightarrow \pi^* \\
(\pi \cup ?\varphi)^* &\rightarrow \pi^* \\
\pi^{**} &\rightarrow \pi^*
\end{aligned}
$$

Simplifying unions by splitting up in test part, accessibility part and rest:

```
splitU :: [Program] -> ([Form],[Agent],[Program])
splitU [] = ([],[],[])
splitU (Test f: ps) = (f:fs,ags,prs)
  where (fs,ags,prs) = splitU ps
splitU (Ag x: ps) = (fs,union [x] ags,prs)
  where (fs,ags,prs) = splitU ps
splitU (Ags xs: ps) = (fs,union xs ags,prs)
  where (fs,ags,prs) = splitU ps
splitU (Sum ps: ps') = splitU (union ps ps')
splitU (p:ps)        = (fs,ags,p:prs)
  where (fs,ags,prs) = splitU ps
```

Simplifying compositions:

```
comprC :: [Program] -> [Program]
comprC [] = []
comprC (Test Top: ps)          = comprC ps
comprC (Test (Neg Top): ps)    = [Test (Neg Top)]
comprC (Test f: Test f': rest) = comprC (Test (canonF (Conj [f,f'])): rest)
comprC (Conc ps : ps')         = comprC (ps ++ ps')
comprC (p:ps)                  = let ps' = comprC ps
                                 in
                                   if ps' == [Test (Neg Top)]
                                     then [Test (Neg Top)]
                                     else p: ps'
```

Use this in the code for program simplification:

```
simpl :: Program -> Program
simpl (Ag x)                    = Ag x
simpl (Ags [])                  = Test (Neg Top)
simpl (Ags [x])                 = Ag x
simpl (Ags xs)                  = Ags xs
simpl (Test f)                  = Test (canonF f)
```

Simplifying unions:

```
simpl (Sum prs) =
  let (fs,xs,rest) = splitU (map simpl prs)
      f            = canonF (Disj fs)
  in
    if xs == [] && rest == []
      then Test f
    else if xs == [] && f == Neg Top && length rest == 1
      then (head rest)
    else if xs == [] && f == Neg Top
      then Sum rest
    else if xs == []
      then Sum (Test f: rest)
    else if length xs == 1  && f == Neg Top
      then Sum (Ag (head xs): rest)
    else if length xs == 1
      then Sum (Test f: Ag (head xs): rest)
    else if f == Neg Top
      then Sum (Ags xs: rest)
    else Sum (Test f: Ags xs: rest)
```

Simplifying sequential compositions:

```
simpl (Conc prs) =
    let prs' = comprC (map simpl prs)
    in
      if prs'== []                  then Test Top
      else if length prs' == 1      then head prs'
      else if head prs' == Test Top then Conc (tail prs')
      else                               Conc prs'
```

Simplifying stars:

```
simpl (Star pr) = case simpl pr of
    Test f             -> Test Top
    Sum [Test f, pr']  -> Star pr'
    Sum (Test f: prs') -> Star (Sum prs')
    Star pr'           -> Star pr'
    pr'                -> Star pr'
```

Property of being a purely propositional formula:

```
pureProp ::  Form -> Bool
pureProp Top       = True
pureProp (Prop _)  = True
pureProp (Neg f)   = pureProp f
pureProp (Conj fs) = and (map pureProp fs)
pureProp (Disj fs) = and (map pureProp fs)
pureProp  _        = False
```

Some example formulas and formula-forming operators:

```
bot, p0, p, p1, p2, p3, p4, p5, p6 :: Form
bot = Neg Top
p0 = Prop (P 0); p = p0; p1  = Prop (P 1); p2 = Prop (P 2)
p3 = Prop (P 3); p4 = Prop (P 4); p5 = Prop (P 5); p6 = Prop (P 6)

q0, q, q1, q2, q3, q4, q5, q6 :: Form
q0 = Prop (Q 0); q = q0; q1  = Prop (Q 1); q2 = Prop (Q 2);
q3 = Prop (Q 3); q4 = Prop (Q 4); q5 = Prop (Q 5); q6 = Prop (Q 6)

r0, r, r1, r2, r3, r4, r5, r6:: Form
r0 = Prop (R 0); r = r0; r1  = Prop (R 1); r2 = Prop (R 2)
r3 = Prop (R 3); r4 = Prop (R 4); r5 = Prop (R 5); r6 = Prop (R 6)

u   = Up ::  AM -> Form -> Form

nkap = Neg (K a p)
nkanp = Neg (K a (Neg p))
nka_p = Conj [nkap,nkanp]
```

14 Reducing Formulas to Canonical Form

For computing bisimulations, it is useful to have some notion of equivalence (however crude) for the logical language. For this, we reduce formulas to a canonical form. We will derive canonical forms that are unique up to propositional equivalence, employing a propositional reasoning engine. This is still rather crude, for any modal formula will be treated as a propositional literal.

The DPLL (Davis, Putnam, Logemann, Loveland) engine expects clauses represented as lists of integers, so we first have to translate to this format. This translation should start with computing a mapping from positive literals to integers.

For the non-propositional operators we use a little bootstrapping, by putting the formula inside the operator in canonical form, using the function `canonF` to be defined below. Also, since the non-propositional operators all behave as Box modalities, we can reduce $\Box\top$ to $\top$.

```
mapping :: Form -> [(Form,Integer)]
mapping f = zip lits [1..k]
  where
  lits = (sort . nub . collect) f
  k    = toInteger (length lits)
  collect :: Form -> [Form]
  collect Top         = []
  collect (Prop p)    = [Prop p]
  collect (Neg f)     = collect f
  collect (Conj fs)   = concat (map collect fs)
  collect (Disj fs)   = concat (map collect fs)
  collect (Pr pr f)   = if canonF f == Top then [] else [Pr pr (canonF f)]
  collect (K ag f)    = if canonF f == Top then [] else [K ag (canonF f)]
  collect (EK ags f)  = if canonF f == Top then [] else [EK ags (canonF f)]
  collect (CK ags f)  = if canonF f == Top then [] else [CK ags (canonF f)]
  collect (Up pam f)  = if canonF f == Top then [] else [Up pam (canonF f)]
```

Putting in clausal form, given a mapping for the literals, and using bootstrapping for formulas in the scope of a non-propositional operator. Note that $\Box\top$ is reduced to $\top$, and $\neg\Box\top$ to $\bot$.

```
cf :: (Form -> Integer) -> Form -> [[Integer]]
cf g (Top)            = []
cf g (Prop p)         = [[g (Prop p)]]
cf g (Pr pr f)        = if canonF f == Top then []
                        else [[g (Pr pr (canonF f))]]
cf g (K ag f)         = if canonF f == Top then []
                        else [[g (K ag (canonF f))]]
cf g (EK ags f)       = if canonF f == Top then []
                        else [[g (EK ags (canonF f))]]
cf g (CK ags f)       = if canonF f == Top then []
                        else [[g (CK ags (canonF f))]]
cf g (Up am f)        = if canonF f == Top then []
                        else [[g (Up am (canonF f))]]
cf g (Conj fs)        = concat (map (cf g) fs)
cf g (Disj fs)        = deMorgan (map (cf g) fs)
```

Negated formulas:

```
cf g (Neg Top)        = [[]]
cf g (Neg (Prop p))   = [[- g (Prop p)]]
cf g (Neg (Pr pr f))  = if canonF f == Top then [[]]
                        else [[- g (Pr pr (canonF f))]]
cf g (Neg (K ag f))   = if canonF f == Top then [[]]
                        else [[- g (K ag (canonF f))]]
cf g (Neg (EK ags f)) = if canonF f == Top then [[]]
                        else [[- g (EK ags (canonF f))]]
cf g (Neg (CK ags f)) = if canonF f == Top then [[]]
                        else [[- g (CK ags (canonF f))]]
cf g (Neg (Up am f))  = if canonF f == Top then [[]]
                        else [[- g (Up am (canonF f))]]
cf g (Neg (Conj fs))  = deMorgan (map (\ f -> cf g (Neg f)) fs)
cf g (Neg (Disj fs))  = concat   (map (\ f -> cf g (Neg f)) fs)
cf g (Neg (Neg f))    = cf g f
```

De Morgan's disjunction distribution:

$$\varphi \vee (\psi_1 \wedge \cdots \wedge \psi_n) \leftrightarrow (\varphi \vee \psi_1) \wedge \cdots \wedge (\varphi \vee \psi_n).$$

De Morgan's disjunction distribution, for the case of a disjunction of a list of clause sets.

```
deMorgan :: [[[Integer]]] -> [[Integer]]
deMorgan [] = [[]]
deMorgan [cls] = cls
deMorgan (cls:clss) = deMorg cls (deMorgan clss)
  where
  deMorg :: [[Integer]] -> [[Integer]] -> [[Integer]]
  deMorg cls1 cls2 = (nub . concat) [ deM cl cls2 | cl <- cls1 ]
  deM :: [Integer] -> [[Integer]]  -> [[Integer]]
  deM cl cls = map (fuseLists cl) cls
```

Function `fuseLists` keeps the literals in the clauses ordered.

```
fuseLists :: [Integer] -> [Integer] -> [Integer]
fuseLists [] ys = ys
fuseLists xs [] = xs
fuseLists (x:xs) (y:ys) | abs x < abs y  = x:(fuseLists xs (y:ys))
                        | abs x == abs y = if x == y
                                           then x:(fuseLists xs ys)
                                           else if x > y
                                             then x:y:(fuseLists xs ys)
                                             else y:x:(fuseLists xs ys)
                        | abs x > abs y  = y:(fuseLists (x:xs) ys)
```

Given a mapping for the positive literals, the satisfying valuations of a formula can be collected from the output of the DPLL process. Here `dp` is the function imported from the module `DPLL`.

```
satVals :: [(Form,Integer)] -> Form -> [[Integer]]
satVals t f = (map fst . dp) (cf (table2fct t) f)
```

Two formulas are propositionally equivalent if they have the same sets of satisfying valuations, computed on the basis of a literal mapping for their conjunction:

```
propEquiv :: Form -> Form -> Bool
propEquiv f1 f2 = satVals g f1 == satVals g f2
  where g = mapping (Conj [f1,f2])
```

A formula is a (propositional) contradiction if it is propositionally equivalent to `Neg Top`, or equivalently, to `Disj []`:

```
contrad :: Form -> Bool
contrad f = propEquiv f (Disj [])
```

A formula is (propositionally) consistent if it is not a propositional contradiction:

```
consistent :: Form -> Bool
consistent = not . contrad
```

Use the set of satisfying valuations to derive a canonical form:

```
canonF :: Form -> Form
canonF f = if (contrad (Neg f))
             then Top
             else if fs == []
             then Neg Top
             else if length fs == 1
             then head fs
             else Disj fs
  where g   = mapping f
        nss = satVals g f
        g'  = \ i -> head [ form | (form,j) <- g, i == j ]
        h   = \ i -> if i < 0 then Neg (g' (abs i)) else g' i
        h'  = \ xs -> map h xs
        k   = \ xs -> if xs == []
                      then Top
                      else if length xs == 1
                              then head xs
                              else Conj xs
        fs  = map k (map h' nss)
```

This gives:

```
ActEpist> canonF p
p
ActEpist> canonF (Conj [p,Top])
p
ActEpist> canonF (Conj [p,q,Neg r])
&[p,q,-r]
ActEpist> canonF (Neg (Disj [p,(Neg p)]))
-T
ActEpist> canonF (Disj [p,q,Neg r])
v[p,&[-p,q],&[-p,-q,-r]]
ActEpist> canonF (K a (Disj [p,q,Neg r]))
[a]v[p,&[-p,q],&[-p,-q,-r]]
ActEpist> canonF (Conj  [p, Conj [q,Neg r]])
&[p,q,-r]
ActEpist> canonF (Conj  [p, Disj [q,Neg (K a (Disj []))]])
v[&[p,q],&[p,-q,-[a]-T]]
ActEpist> canonF (Conj  [p, Disj [q,Neg (K a (Conj []))]])
&[p,q]
```

15 Action Models and Epistemic Models

Action models and epistemic models are built from states. We assume states are represented by integers:

```
type State = Integer
```

Epistemic models are models where the states are of type `State`, and the precondition function assigns lists of basic propositions (this specializes the precondition function to a valuation).

```
type EM = Model State [Prop]
```

Find the valuation of an epistemic model:

```
valuation  :: EM -> [(State,[Prop])]
valuation = eval
```

Action models are models where the states are of type `State`, and the precondition function assigns objects of type `Form`. The only difference between an action model and a static model is in the fact that action models have a precondition function that assigns a formula instead of a set of basic propositions.

```
type AM = Model State Form
```

The preconditions of an action model:

```
preconditions :: AM -> [Form]
preconditions (Mo states pre ags acc points) =
   map (table2fct pre) points
```

Sometimes we need a single precondition:

```
precondition :: AM -> Form
precondition am = canonF (Conj (preconditions am))
```

The zero action model $\mathbf{0}$:

```
zero :: [Agent] -> AM
zero ags = (Mo [] [] ags [] [])
```

The purpose of action models is to define relations on the class of all static models. States with precondition $\bot$ can be pruned from an action model. For this we define a specialized version of the `gsm` function:

```
gsmAM :: AM -> AM
gsmAM (Mo states pre ags acc points) =
  let
    points' = [ p | p <- points, consistent (table2fct pre p) ]
    states' = [ s | s <- states, consistent (table2fct pre s) ]
    pre'    = filter (\ (x,_) -> elem x states') pre
    f       = \ (_,s,t) -> elem s states' && elem t states'
    acc'    = filter f acc
  in
  if points' == []
     then zero ags
     else gsm (Mo states' pre' ags acc' points')
```

16 Program Transformation

For every action model A with states $s_0, \ldots, s_{n-1}$ we define a set of n^2 program transformers $T^A_{i,j}$ ($0 \leq i < n, 0 \leq j < n$), as follows [17]:

:

$$
\begin{array}{rcl}
T^A_{ij}(a) & = & \begin{cases} ?\mathrm{pre}(s_i); a & \text{if } s_i \xrightarrow{a} s_j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(?\varphi) & = & \begin{cases} ?(\mathrm{pre}(s_i) \wedge [A, s_i]\varphi) & \text{if } i = j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(\pi_1; \pi_2) & = & \bigcup_{k=0}^{n-1} (T^A_{ik}(\pi_1); T^A_{kj}(\pi_2)) \\
T^A_{ij}(\pi_1 \cup \pi_2) & = & T^A_{ij}(\pi_1) \cup T^A_{ij}(\pi_2) \\
T^A_{ij}(\pi^*) & = & K^A_{ijn}(\pi)
\end{array}
$$

where $K^A_{ijk}(\pi)$ is a (transformed) program for all the π^* paths from s_i to s_j that can be traced through A while avoiding a pass through intermediate states s_k and higher. Thus, $K^A_{ijn}(\pi)$ is a program for all the π^* paths from s_i to s_j that can be traced through A, period.

$K^A_{ijk}(\pi)$ is defined by recursing on k, as follows:

$$K^A_{ij0}(\pi) \;=\; \begin{cases} ?\top \cup T^A_{ij}(\pi) & \text{if } i = j, \\ T^A_{ij}(\pi) & \text{otherwise} \end{cases}$$

$$K^A_{ij(k+1)}(\pi) \;=\; \begin{cases} (K^A_{kkk}(\pi))^* & \text{if } i = k = j, \\ (K^A_{kkk}(\pi))^*; K^A_{kjk}(\pi) & \text{if } i = k \neq j, \\ K^A_{ikk}(\pi); (K^A_{kkk}(\pi))^* & \text{if } i \neq k = j, \\ K^A_{ijk}(\pi) \cup (K^A_{ikk}(\pi); (K^A_{kkk}(\pi))^*; K^A_{kjk}(\pi)) & \text{otherwise } (i \neq k \neq j). \end{cases}$$

Lemma 2 (Kleene Path) *Suppose* $(w, w') \in [\![T^A_{ij}(\pi)]\!]^{\mathbf{M}}$ *iff there is a* π *path from* (w, s_i) *to* (w', s_j) *in* $\mathbf{M} \otimes A$. *Then* $(w, w') \in [\![K^A_{ijn}(\pi)]\!]^{\mathbf{M}}$ *iff there is a* π^* *path from* (w, s_i) *to* (w', s_j) *in* $\mathbf{M} \otimes A$.

The Kleene path lemma is the key ingredient in the proof of the following program transformation lemma.

Lemma 3 (Program Transformation) *Assume* A *has* n *states* $s_0, \ldots, s_{n-1}$. *Then:*

$$\mathbf{M} \models_w [A, s_i][\pi]\varphi \text{ iff } \mathbf{M} \models_w \bigwedge_{j=0}^{n-1} [T^A_{ij}(\pi)][A, s_j]\varphi.$$

The implementation of the program transformation functions is given here:

```
transf :: AM -> Integer -> Integer -> Program -> Program
transf am@(Mo states pre allAgs acc points) i j (Ag ag) =
   let
     f = table2fct pre i
   in
   if elem (ag,i,j) acc && f == Top          then Ag ag
   else if elem (ag,i,j) acc && f /= Neg Top then Conc [Test f, Ag ag]
   else Test (Neg Top)
transf am@(Mo states pre allAgs acc points) i j (Ags ags) =
   let ags' = nub [ a | (a,k,m) <- acc, elem a ags, k == i, m == j ]
       ags1 = intersect ags ags'
       f    = table2fct pre i
   in
     if ags1 == [] || f == Neg Top        then Test (Neg Top)
     else if f == Top && length ags1 == 1 then Ag (head ags1)
     else if f == Top                     then Ags ags1
     else Conc [Test f, Ags ags1]
transf am@(Mo states pre allAgs acc points) i j (Test f) =
   let
     g = table2fct pre i
   in
   if i == j
      then Test (Conj [g,(Up am f)])
      else Test (Neg Top)
transf am@(Mo states pre allAgs acc points) i j (Conc [])  =
  transf am i j (Test Top)
transf am@(Mo states pre allAgs acc points) i j (Conc [p]) = transf am i j p
transf am@(Mo states pre allAgs acc points) i j (Conc (p:ps)) =
  Sum [ Conc [transf am i k p, transf am k j (Conc ps)] | k <- [0..n] ]
    where n = toInteger (length states - 1)
transf am@(Mo states pre allAgs acc points) i j (Sum [])  =
  transf am i j (Test (Neg Top))
transf am@(Mo states pre allAgs acc points) i j (Sum [p]) = transf am i j p
transf am@(Mo states pre allAgs acc points) i j (Sum ps)  =
  Sum [ transf am i j p | p <- ps ]
transf am@(Mo states pre allAgs acc points) i j (Star p) = kleene am i j n p
  where n = toInteger (length states)
```

Implementation of $K^{\mathbf{A}}_{ijk}$:

```
kleene ::  AM -> Integer -> Integer -> Integer -> Program -> Program
kleene am i j 0 pr =
  if i == j
    then Sum [Test Top, transf am i j pr]
    else transf am i j pr
kleene am i j k pr
  | i == j && j == pred k = Star (kleene am i i i pr)
  | i == pred k           =
    Conc [Star (kleene am i i i pr), kleene am i j i pr]
  | j == pred k           =
    Conc [kleene am i j j pr, Star (kleene am j j j pr)]
  | otherwise             =
      Sum [kleene am i j k' pr,
           Conc [kleene am i k' k' pr,
                 Star (kleene am k' k' k' pr), kleene am k' j k' pr]]
      where k' = pred k
```

Transformation plus simplification:

```
tfm ::  AM -> Integer -> Integer -> Program -> Program
tfm am i j pr = simpl (transf am i j pr)
```

The program transformations can be used to translate Update PDL to PDL, as follows:

$$
\begin{aligned}
t(\top) &= \top \\
t(p) &= p \\
t(\neg\varphi) &= \neg t(\varphi) \\
t(\varphi_1 \wedge \varphi_2) &= t(\varphi_1) \wedge t(\varphi_2) \\
t([\pi]\varphi) &= [r(\pi)]t(\varphi) \\
t([A,s]\top) &= \top \\
t([A,s]p) &= t(\mathrm{pre}(s)) \rightarrow p \\
t([A,s]\neg\varphi) &= t(\mathrm{pre}(s)) \rightarrow \neg t([A,s]\varphi) \\
t([A,s](\varphi_1 \wedge \varphi_2)) &= t([A,s]\varphi_1) \wedge t([A,s]\varphi_2) \\
t([A,s_i][\pi]\varphi) &= \bigwedge_{j=0}^{n-1} [T^A_{ij}(r(\pi))]t([A,s_j]\varphi) \\
t([A,s][A',s']\varphi) &= t([A,s]t([A',s']\varphi)) \\
t([A,S]\varphi) &= \bigwedge_{s\in S} t[A,s]\varphi)
\end{aligned}
$$

$$
\begin{aligned}
r(a) &= a \\
r(B) &= B \\
r(?\varphi) &= ?t(\varphi) \\
r(\pi_1;\pi_2) &= r(\pi_1);r(\pi_2) \\
r(\pi_1 \cup \pi_2) &= r(\pi_1) \cup r(\pi_2) \\
r(\pi^*) &= (r(\pi))^*.
\end{aligned}
$$

The correctness of this translation follows from direct semantic inspection, using the program transformation lemma for the translation of $[A, s_i][\pi]\varphi$ formulas.

The crucial clauses in this translation procedure are those for formulas of the forms $[A, S]\varphi$ and $[A, s]\varphi$, and more in particular the one for formulas of the form $[A, s][\pi]\varphi$. It makes sense to give separate functions for the steps that pull the update model through program π given formula φ.

```
step0, step1 :: AM -> Program -> Form -> Form
step0 am@(Mo states pre allAgs acc []) pr f = Top
step0 am@(Mo states pre allAgs acc [i]) pr f = step1 am pr f
step0 am@(Mo states pre allAgs acc  is) pr f =
  Conj [ step1 (Mo states pre allAgs acc [i]) pr f | i <- is ]
step1 am@(Mo states pre allAgs acc [i]) pr f =
   Conj [ Pr (transf am i j (rpr pr))
              (Up (Mo states pre allAgs acc [j]) f) | j <- states ]
```

Perform a single step, and put in canonical form:

```
step :: AM -> Program -> Form -> Form
step am pr f = canonF (step0 am pr f)
```

```
t :: Form -> Form
t Top = Top
t (Prop p) = Prop p
t (Neg f) = Neg (t f)
t (Conj fs) = Conj (map t fs)
t (Disj fs) = Disj (map t fs)
t (Pr pr f) = Pr (rpr pr) (t f)
t (K x f)   = Pr (Ag x) (t f)
t (EK xs f)  = Pr (Ags xs) (t f)
t (CK xs f)  = Pr (Star (Ags xs)) (t f)
```

Translations of formulas starting with an action model update:

```
t (Up am@(Mo states pre allAgs acc [i]) f) = t' am f
t (Up am@(Mo states pre allAgs acc  is) f)  =
   Conj [ t' (Mo states pre allAgs acc [i]) f | i <- is ]
```

Translations of formulas starting with a single pointed action model update are performed by `t'`:

```
t' :: AM -> Form -> Form
t' am Top            = Top
t' am (Prop p)       = impl (precondition am) (Prop p)
t' am (Neg f)        =  Neg (t' am f)
t' am (Conj fs)      = Conj (map (t' am) fs)
t' am (Disj fs)      = Disj (map (t' am) fs)
t' am (K x f)        = t' am (Pr (Ag x) f)
t' am (EK xs f)      = t' am (Pr (Ags xs) f)
t' am (CK xs f)      = t' am (Pr (Star (Ags xs)) f)
t' am (Up am' f)     = t' am (t (Up am' f))
```

The crucial case: update action having scope over a program. We may assume that the update action is single pointed.

```
t' am@(Mo states pre allAgs acc [i]) (Pr pr f) =
   Conj [ Pr (transf am i j (rpr pr))
              (t' (Mo states pre allAgs acc [j]) f) | j <- states ]
t' am@(Mo states pre allAgs acc is) (Pr pr f) =
   error "action model not single pointed"
```

Translations for programs:

```
rpr :: Program -> Program
rpr (Ag x)       = Ag x
rpr (Ags xs)     = Ags xs
rpr (Test f)     = Test (t f)
rpr (Conc ps)    = Conc (map rpr ps)
rpr (Sum  ps)    = Sum  (map rpr ps)
rpr (Star p)     = Star (rpr p)
```

Translating and putting in canonical form:

```
tr :: Form -> Form
tr = canonF . t
```

Some example translations:

```
ActEpist> tr (Up (public p) (Pr (Star (Ags [b,c])) p))
T
ActEpist> tr (Up (public (Disj [p,q])) (Pr (Star (Ags [b,c])) p))
[(U[?T,C[?v[p,q],[b,c]]])*]v[p,&[-p,-q]]
ActEpist> tr (Up (groupM [a,b] p) (Pr (Star (Ags [b,c])) p))
[C[C[(U[?T,C[?p,[b,c]]])*,C[?p,[c]]],(U[U[?T,[b,c]],C[c,(U[?T,C[?p,[b,c]]])*,C[?p,[c]]]])*]]p
ActEpist> tr (Up (secret [a,b] p) (Pr (Star (Ags [b,c])) p))
[C[C[(U[?T,C[?p,[b]]])*,C[?p,[c]]],(U[U[?T,[b,c]],C[?-T,(U[?T,C[?p,[b]]])*,C[?p,[c]]]])*]]p
```

Semantics

We now turn to the implementation of the semantics module.

17 Semantics Module Declaration

```
module Semantics
where

import List
import Char
import Models
import Display
import MinBis
import ActEpist
import DPLL
```

18 Semantics Implementation

The group alternatives of group of agents a are the states that are reachable through $\bigcup_{a \in A} R_a$.

```
groupAlts :: [(Agent,State,State)] -> [Agent] -> State -> [State]
groupAlts rel agents current =
  (nub . sort . concat) [ alternatives rel a current | a <- agents ]
```

The common knowledge alternatives of group of agents a are the states that are reachable through a finite number of R_a links, for $a \in A$.

```
commonAlts :: [(Agent,State,State)] -> [Agent] -> State -> [State]
commonAlts rel agents current =
  closure rel agents (groupAlts rel agents current)
```

The model update function takes a static model and and action model and returns an object of type `Model (State,State) [Prop]`. The up function takes an epistemic model and an AM and returns an EM. Its states are the `(State,State)` pairs that result from the cartesian product construction described in [2]. Note that the update function uses the truth definition (given below as `isTrueAt`).

We will set up matters in such way that updates with action models get their list of agents from the epistemic model that gets updated. For this, we define:

```
type FAM = [Agent] -> AM
```

```
up :: EM -> FAM -> Model (State,State) [Prop]
up m@(Mo worlds val ags acc points) fam =
   Mo worlds' val' ags acc' points'
   where
   am@(Mo states pre _ susp actuals) = fam ags
   worlds' = [ (w,s) | w <- worlds, s <- states,
                       formula <- maybe [] (\ x -> [x]) (lookup s pre),
                       isTrueAt w m formula                    ]
   val'    = [ ((w,s),props) | (w,props) <- val,
                               s        <- states,
                               elem (w,s) worlds'              ]
   acc'    = [ (ag1,(w1,s1),(w2,s2)) | (ag1,w1,w2) <- acc,
                                       (ag2,s1,s2) <- susp,
                                        ag1 == ag2,
                                        elem (w1,s1) worlds',
                                        elem (w2,s2) worlds'   ]
   points' = [ (p,a) | p <- points, a <- actuals,
                       elem (p,a) worlds'                      ]
```

An action model is tiny if is action list is empty or a singleton list:

```
tiny :: FAM -> Bool
tiny fam = length actions <= 1
  where actions = domain (fam [minBound..maxBound])
```

The appropriate notion of equivalence for the base case of the bisimulation for epistemic models is "having the same valuation".

```
sameVal :: [Prop] -> [Prop] -> Bool
sameVal ps qs = (nub . sort) ps ==  (nub . sort) qs
```

Bisimulation minimal version of generated submodel of update result for epistemic model and PoAM:

```
upd :: EM -> FAM -> EM
upd sm fam = if tiny fam then conv (up sm fam)
             else bisim (sameVal) (up sm fam)
```

Non-deterministic update with a list of PoAMs:

```
upds  :: EM -> [FAM] -> EM
upds = foldl upd
```

At last we have all ingredients for the truth definition.

```
isTrueAt :: State -> EM -> Form -> Bool
isTrueAt w m Top = True
isTrueAt w m@(Mo worlds val ags acc pts) (Prop p) =
  elem p (concat [ props | (w',props) <- val, w'==w ])
isTrueAt w m (Neg f)   = not (isTrueAt w m f)
isTrueAt w m (Conj fs) = and (map (isTrueAt w m) fs)
isTrueAt w m (Disj fs) = or  (map (isTrueAt w m) fs)
```

The clauses for individual knowledge, general knowledge and common knowledge use the functions `alternatives`, `groupAlts` and `commonAlts` to compute the relevant accessible worlds:

```
isTrueAt w m@(Mo worlds val ags acc pts) (K ag f) =
  and (map (flip ((flip isTrueAt) m) f) (alternatives acc ag w))
isTrueAt w m@(Mo worlds val ags acc pts) (EK agents f) =
  and (map (flip ((flip isTrueAt) m) f) (groupAlts acc agents w))
isTrueAt w m@(Mo worlds val ags acc pts) (CK agents f) =
  and (map (flip ((flip isTrueAt) m) f) (commonAlts acc agents w))
```

In the clause for $[\mathbf{M}]\varphi$, the result of updating the static model M with action model $\mathbf{M}$ may be undefined, but in this case the precondition $P(s_0)$ of the designated state s_0 of $\mathbf{M}$ will fail in the designated world w_0 of M. By making the clause for $[\mathbf{M}]\varphi$ check for $M \models_{w_0} P(s_0)$, truth can be defined as a total function.

```
isTrueAt w m@(Mo worlds val ags rel pts) (Up am f) =
  and [ isTrue  m' f |
         m' <- decompose (upd (Mo worlds val ags rel [w]) (\ ags -> am)) ]
```

Checking for truth in *all* the designated points of an epistemic model:

```
isTrue :: EM -> Form -> Bool
isTrue (Mo worlds val ags rel pts) form =
   and [ isTrueAt w (Mo worlds val ags rel pts) form | w <- pts ]
```

19 Tools for Constructing Epistemic Models

The following function constructs an initial epistemic model where the agents are completely ignorant about their situation, as described by a list of basic propositions. The input is a list of basic propositions used for constructing the valuations.

```
initE :: [Prop] -> [Agent] -> EM
initE allProps ags = (Mo worlds val ags accs points)
  where
    worlds = [0..(2^k - 1)]
    k      = length allProps
    val    = zip worlds (sortL (powerList allProps))
    accs   = [ (ag,st1,st2) | ag  <- ags,
                              st1 <- worlds,
                              st2 <- worlds      ]
    points = worlds
```

This uses the following utilities:

```
powerList  :: [a] -> [[a]]
powerList  [] = [[]]
powerList  (x:xs) = (powerList xs) ++ (map (x:) (powerList xs))

sortL :: Ord a => [[a]] -> [[a]]
sortL  = sortBy (\ xs ys -> if length xs < length ys then LT
                            else if length xs > length ys then GT
                            else compare xs ys)
```

Some initial models:

```
e00 :: EM
e00 = initE [P 0] [a,b]

e0 :: EM
e0 = initE [P 0,Q 0] [a,b,c]
```

20 From Communicative Actions to Action Models

Computing the update for a public announcement:

```
public :: Form -> FAM
public form ags =
   (Mo [0] [(0,form)] ags [ (a,0,0) | a <- ags ] [0])
```

Public announcements are S5 models:

```
DEMO> showM (public p [a,b,c])
==> [0]
[0]
(0,p)
(a,[[0]])
(b,[[0]])
(c,[[0]])
```

Computing the update for passing a group announcement to a list of agents: the other agents may or may not be aware of what is going on. In the limit case where the message is passed to all agents, the message is a public announcement.

```
groupM :: [Agent] -> Form -> FAM
groupM gr form agents =
  if sort gr == sort agents
    then public form agents
    else
      (Mo
         [0,1]
         [(0,form),(1,Top)]
         agents
         ([ (a,0,0) | a <- agents ]
           ++ [ (a,0,1) | a <- agents \\ gr ]
           ++ [ (a,1,0) | a <- agents \\ gr ]
           ++ [ (a,1,1) | a <- agents           ])
         [0])
```

Group announcements are S5 models:

```
Semantics> showM (groupM [a,b] p [a,b,c,d,e])
=> [0]
[0,1]
(0,p)(1,T)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
(d,[[0,1]])
(e,[[0,1]])
```

Computing the update for an individual message to b that φ:

```
message :: Agent -> Form -> FAM
message agent = groupM [agent]
```

Another special case of a group message is a test. Tests are updates that messages to the empty group:

```
test :: Form -> FAM
test = groupM []
```

Computing the update for passing a *secret* message to a list of agents: the other agents remain unaware of the fact that something goes on. In the limit case where the secret is divulged to all agents, the secret becomes a public update.

```
secret :: [Agent] -> Form -> FAM
secret agents form all_agents =
  if sort agents == sort all_agents
    then public form agents
    else
      (Mo
         [0,1]
         [(0,form),(1,Top)]
          all_agents
         ([ (a,0,0) | a <- agents ]
           ++ [ (a,0,1) | a <- all_agents \\ agents ]
           ++ [ (a,1,1) | a <- all_agents           ])
         [0])
```

Secret messages are KD45 models:

```
DEMO> showM (secret [a,b] p [a,b,c])
==> [0]
[0,1]
(0,p)(1,T)
(a,[([],[0]),([],[1])])
(b,[([],[0]),([],[1])])
(c,[([0],[1])])
```

To Do 1 *Add functions for messages with bcc.*

Here is a multiple pointed action model for the communicative action of revealing one of a number of alternatives to a list of agents, in such a way that it is common knowledge that one of the alternatives gets revealed (in [3] this is called *common knowledge of alternatives*).

```
reveal :: [Agent] -> [Form] -> FAM
reveal ags forms all_agents =
  (Mo
     states
     (zip states forms)
     all_agents
     ([ (ag,s,s) | s <- states, ag <- ags ]
       ++
      [ (ag,s,s') | s <- states, s' <- states, ag <- others ])
     states)
    where states = map fst (zip [0..] forms)
          others = all_agents \\ ags
```

Here is an action model for the communication that reveals to a one of p_1, q_1, r_1.

```
Semantics> showM (reveal [a] [p1,q1,r1] [a,b])
==> [0,1,2]
[0,1,2]
(0,p1)(1,q1)(2,r1)
(a,[[0],[1],[2]])
(b,[[0,1,2]])
```

A group of agents B gets (transparantly) informed about a formula φ if B get to know φ when φ is true, and B get to know the negation of φ otherwise. Transparancy means that all other agents are aware of the fact that B get informed about φ, i.e., the other agents learn that $(\varphi \to C_B\varphi) \wedge (\neg\varphi \to C_B\neg\varphi)$. This action model can be defined in terms of `reveal`, as follows:

```
info :: [Agent] -> Form -> FAM
info agents form =
  reveal agents [form, negation form]
```

An example application:

```
Semantics> showM (upd e0 (info [a,b] q))
==> [0,1,2,3]
[0,1,2,3]
(0,[])(1,[p])(2,[q])(3,[p,q])
(a,[[0,1],[2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1,2,3]])

Semantics> isTrue (upd e0 (info [a,b] q)) (CK [a,b] q)
False
Semantics> isTrue (upd e0 (groupM [a,b] q)) (CK [a,b] q)
True
```

Slightly different is informing a set of agents about what is actually the case with respect to formula φ:

```
infm :: EM -> [Agent] -> Form -> FAM
infm m ags f = if isTrue m f
                 then groupM ags f
                 else if isTrue m (Neg f)
                        then groupM ags (Neg f)
                        else one
```

And the corresponding thing for public announcement:

```
publ :: EM -> Form -> FAM
publ m f = if isTrue m f
             then public f
             else if isTrue m (Neg f)
                    then public (Neg f)
                    else one
```

21 Operations on Action Models

The trivial update action model is a special case of public announcement. Call this the `one` action model, for it behaves as 1 for the operation $\otimes$ of action model composition.

```
one :: FAM
one = public Top
```

Composition $\otimes$ of multiple pointed action models.

```
cmpP :: FAM -> FAM -> [Agent] -> Model (State,State) Form
cmpP fam1 fam2 ags =
 (Mo nstates npre ags nsusp npoints)
   where m@(Mo states pre _ susp ss) = fam1 ags
         (Mo states' pre' _ susp' ss') = fam2 ags
         npoints = [ (s,s') | s <- ss, s' <- ss' ]
         nstates = [ (s,s') | s <- states, s' <- states' ]
         npre    = [ ((s,s'), g) | (s,f)     <- pre,
                                   (s',f')   <- pre',
                                    g        <- [computePre m f f']      ]
         nsusp   = [ (ag,(s1,s1'),(s2,s2')) | (ag,s1,s2)    <- susp,
                                              (ag',s1',s2') <- susp',
                                               ag == ag'                 ]
```

Utility function for this: compute the new precondition of a state pair. If the preconditions of the two states are purely propositional, we know that the updates at the states commute and that their combined precondition is the conjunction of the two preconditions, provided this conjunction is not a contradiction. If one of the states has a precondition that is not purely propositional, we have to take the epistemic effect of the update into account in the new precondition.

```
computePre  :: AM -> Form -> Form -> Form
computePre m g g'  | pureProp conj = conj
                   | otherwise     = Conj [ g, Neg (Up m (Neg g')) ]
  where conj    = canonF (Conj [g,g'])
```

To Do 2 *Refine the precondition computation, by making more clever use of what is known about the update effect of the first action model.*

Compose pairs of multiple pointed action models, and reduce the result to its simplest possible form under action emulation.

```
cmpFAM :: FAM -> FAM -> FAM
-- cmpFAM fam fam' ags = aePmod (cmpP fam fam' ags)
cmpFAM fam fam' ags = conv (cmpP fam fam' ags)
```

Use `one` as unit for composing lists of FAMs:

```
cmp :: [FAM] -> FAM
cmp = foldl cmpFAM one
```

Here is the result of composing two messages:

```
Semantics> showM (cmp [groupM [a,b] p, groupM [b,c] q] [a,b,c])
==> [0]
[0,1,2,3]
(0,&[p,q])(1,p)(2,q)(3,T)
(a,[[0,1],[2,3]])
(b,[[0],[1],[2],[3]])
(c,[[0,2],[1,3]])
```

This gives the resulting action model. Here is the result of composing the messages in the reverse order:

```
==> [0]
[0,1,2,3]
(0,&[p,q])(1,q)(2,p)(3,T)
(a,[[0,2],[1,3]])
(b,[[0],[1],[2],[3]])
(c,[[0,1],[2,3]])
```

These two action models are bisimilar under the renaming $1 \mapsto 2, 2 \mapsto 1$.

Here is an illustration of an observation from [16].

```
m2  = initE [P 0,Q 0] [a,b,c]
psi = Disj[Neg(K b p),q]
```

```
Semantics>  showM (upds m2 [message a psi, message b p])
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[[0,1,2,3,4,5]])
(b,[[0,2,3,5],[1,4]])
(c,[[0,1,2,3,4,5]])

Semantics> showM (upds m2 [message b p, message a psi])
==> [7]
[0,1,2,3,4,5,6,7,8,9,10]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[q])(7,[p,q])(8,[p,q])(9,[p,q])
(10,[p,q])
(a,[[0,3,5,7,9],[1,2,4,6,8,10]])
(b,[[0,1,3,4,5,6,9,10],[2,7,8]])
(c,[[0,1,2,3,4,5,6,7,8,9,10]])
```

Power of action models:

```
pow :: Int -> FAM -> FAM
pow n fam = cmp (take n (repeat fam))
```

Non-deterministic sum $\oplus$ of multiple-pointed action models:

```
ndSum' :: FAM -> FAM -> FAM
ndSum' fam1 fam2 ags = (Mo states val ags acc ss)
  where
       (Mo states1 val1 _ acc1 ss1) = fam1 ags
       (Mo states2 val2 _ acc2 ss2) = fam2 ags
       f   = \ x -> toInteger (length states1) + x
       states2' = map f states2
       val2'    = map (\ (x,y)   -> (f x, y)) val2
       acc2'    = map (\ (x,y,z) -> (x, f y, f z)) acc2
       ss       = ss1 ++ map f ss2
       states   = states1 ++ states2'
       val      = val1 ++ val2'
       acc      = acc1 ++ acc2'
```

Example action models:

```
am0 = ndSum' (test p) (test (Neg p)) [a,b,c]

am1 = ndSum' (test p) (ndSum' (test q) (test r)) [a,b,c]
```

Examples of minimization for action emulation:

```
Semantics>  showM am0
==> [0,2]
[0,1,2,3]
(0,p)(1,T)(2,-p)(3,T)
(a,[([0],[1]),([2],[3])])
(b,[([0],[1]),([2],[3])])
(c,[([0],[1]),([2],[3])])

Semantics> showM (aePmod am0)
==> [0]
[0]
(0,T)
(a,[[0]])
(b,[[0]])
(c,[[0]])

Semantics>  showM am1
==> [0,2,4]
[0,1,2,3,4,5]
(0,p)(1,T)(2,q)(3,T)(4,r)
(5,T)
(a,[([0],[1]),([2],[3]),([4],[5])])
(b,[([0],[1]),([2],[3]),([4],[5])])
(c,[([0],[1]),([2],[3]),([4],[5])])

Semantics> showM (aePmod am1)
==> [0]
[0,1]
```

```
(0,v[p,&[-p,q],&[-p,-q,r]])(1,T)
(a,[([0],[1])])
(b,[([0],[1])])
(c,[([0],[1])])
```

Non-deterministic sum $\oplus$ of multiple-pointed action models, reduced for action emulation:

```
ndSum :: FAM -> FAM -> FAM
ndSum fam1 fam2 ags = (ndSum' fam1 fam2) ags
```

Notice the difference with the definition of alternative composition of Kripke models for processes given in [25, Ch 4].

The `zero` action model is the 0 for the $\oplus$ operation, so it can be used as the base case in the following list version of the $\oplus$ operation:

```
ndS :: [FAM] -> FAM
ndS = foldl ndSum zero
```

Performing a test whether φ and announcing the result:

```
testAnnounce :: Form -> FAM
testAnnounce form = ndS [ cmp [ test form, public form ],
                          cmp [ test (negation form),
                                public (negation form)] ]
```

`testAnnounce form` is equivalent to `info all_agents form`:

```
Semantics> showM (testAnnounce p [a,b,c])
==> [0,1]
[0,1]
(0,p)(1,-p)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0],[1]])

Semantics> showM (info [a,b,c] p [a,b,c])
==> [0,1]
[0,1]
(0,p)(1,-p)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0],[1]])
```

The function `testAnnounce` gives the special case of revelations where the alternatives are a formula and its negation, and where the result is publicly announced.

Note that *DEMO* correctly computes the result of the sequence and the sum of two contradictory propositional tests:

```
Semantics> showM (cmp [test p, test (Neg p)] [a,b,c])
==> []
[]

(a,[])
(b,[])
(c,[])

Semantics> showM (ndS [test p, test (Neg p)] [a,b,c])
==> [0]
[0]
(0,T)
(a,[[0]])
(b,[[0]])
(c,[[0]])
```

Examples

22 The Riddle of the Caps

Picture a situation[3] of four people a, b, c, d standing in line, with a, b, c looking to the left, and d looking to the right. a can see no-one else; b can see a; c can see a and b, and d can see no-one else. They are all wearing caps, and they cannot see their own cap. If it is common knowledge that there are two white and two black caps, then in the following situation c knows what colour cap she is wearing.

If c now announces that she knows the colour of her cap (without revealing the colour), b can infer from this that he is wearing a white cap, for b can reason as follows: "c knows her colour, so she must see two caps of the same colour. The cap I can see is white, so my own cap must be white as well." In this situation b draws a conclusion from the fact that c knows her colour.

In the following situation b can draw a conclusion from the fact that c does not know her colour.

In this case c announces that she does not know her colour, and b can infer from this that he is wearing a black cap, for b can reason as follows: "c does not know her colour, so she must see two caps of different colours in front of her. The cap I can see is white, so my own cap must be black."

To account for this kind of reasoning, we use model checking for epistemic updating, as follows. Proposition p_i expresses the fact that the i-th cap, counting from the left, is white. Thus, the facts of our first example situation are given by $p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$, and those of our second example by $p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$.

Here is the DEMO code for this example (details to be explained below):

[3] See [18].

```
module Caps
where

import DEMO

capsInfo :: Form
capsInfo = Disj [Conj [f, g, Neg h, Neg j] |
                      f <- [p1, p2, p3, p4],
                      g <- [p1, p2, p3, p4] \\ [f],
                      h <- [p1, p2, p3, p4] \\ [f,g],
                      j <- [p1, p2, p3, p4] \\ [f,g,h],
                      f < g, h < j                     ]

awarenessFirstCap  = info [b,c] p1
awarenessSecondCap = info [c]   p2

cK =  Disj [K c p3, K c (Neg p3)]
bK =  Disj [K b p2, K b (Neg p2)]

mo0  = upd (initE [P 1, P 2, P 3, P 4] [a,b,c,d]) (test capsInfo)
mo1  = upd mo0 (public capsInfo)
mo2  = upds mo1 [awarenessFirstCap, awarenessSecondCap]
mo3a = upd mo2 (public cK)
mo3b = upd mo2 (public (Neg cK))
```

An initial situation with four agents a, b, c, d and four propositions p_1, p_2, p_3, p_4, with exactly two of these true, where no-one knows anything about the truth of the propositions, and everyone is aware of the ignorance of the others, is modelled like this:

```
Caps> showM mo0
==> [5,6,7,8,9,10]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])(14,[p2,p3,p4])
(15,[p1,p2,p3,p4])
(a,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(b,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(c,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(d,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
```

The first line indicates that worlds $5, 6, 7, 8, 9, 10$ are compatible with the facts of the matter (the facts being that there are two white and two black caps). E.g., 5 is the world where a and b are wearing the white caps. The second line lists all the possible worlds; there are 2^4 of them, since every world has a different valuation. The third through sixth lines give the valuations of worlds. The last four lines represent the accessibility relations for the agents. All accessibilities are total relations, and they are represented here as the corresponding partitions on the set of worlds. Thus, the ignorance of the agents is reflected in the fact that for all of them all worlds are equivalent: none of the agents can tell any of them apart.

The information that two of the caps are white and two are black is expressed by the formula

$$\begin{array}{ll} & (p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4) \vee (p_1 \wedge p_3 \wedge \neg p_2 \wedge \neg p_4) \vee (p_1 \wedge p_4 \wedge \neg p_2 \wedge \neg p_3) \\ \vee & (p_2 \wedge p_3 \wedge \neg p_1 \wedge \neg p_4) \vee (p_2 \wedge p_4 \wedge \neg p_1 \wedge \neg p_3) \vee (p_3 \wedge p_4 \wedge \neg p_1 \wedge \neg p_2). \end{array}$$

A public announcement with this information has the following effect:

```
Caps> showM (upd mo0 (public capsInfo))
==> [0,1,2,3,4,5]
[0,1,2,3,4,5]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
(d,[[0,1,2,3,4,5]])
```

Let this model be called `mo1`. The representation above gives the partitions for all the agents, showing that nobody knows anything. A perhaps more familiar representation for this multi-agent Kripke model is given in Figure 2. In this picture, all worlds are connected for all agents, all worlds are compatible with the facts of the matter (indicated by the double ovals).

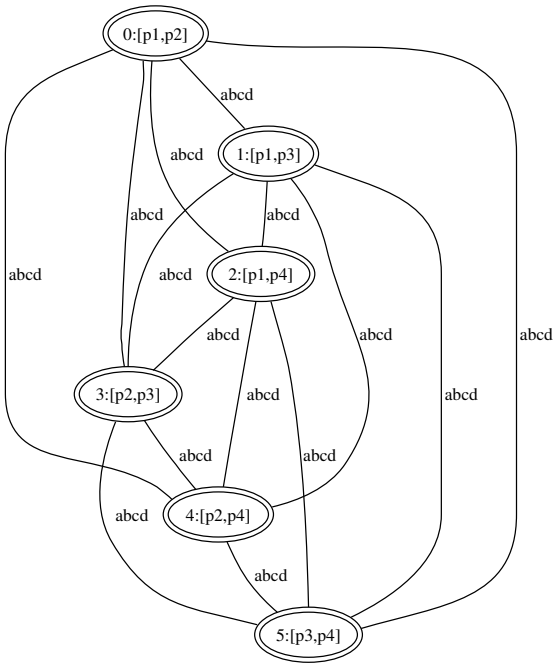


Figure 2: Caps situation where nobody knows anything about p_1, p_2, p_3, p_4.

Next, we model the fact that (everyone is aware that) b can see the first cap and that c can see the first and the second cap, as follows:

```
Caps> showM (upds mo1 [info [b,c] p1, info [c] p2])
==> [0,1,2,3,4,5]
```

```
[0,1,2,3,4,5]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2],[3,4,5]])
(c,[[0],[1,2],[3,4],[5]])
(d,[[0,1,2,3,4,5]])
```

Notice that this model reveals that in case a, b wear caps of the same colour (situations 0 and 5), c knows the colour of all the caps, and in case a, b wear caps of different colours, she does not (she confuses the cases $1, 2$ and the cases $3, 4$). Figure 3 gives a picture representation.

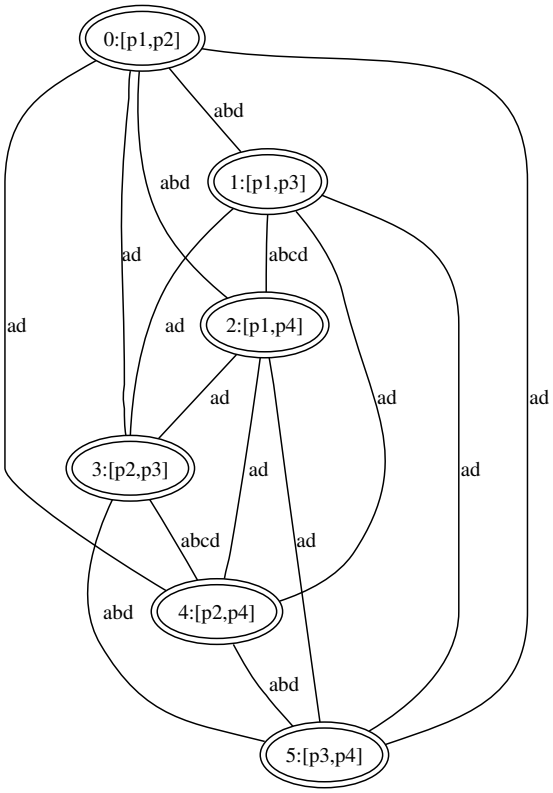


Figure 3: Caps situation after updating with awareness of what b and c can see.

Let this model be called `mo2`. Knowledge of c about her situation is expressed by the epistemic formula $K_c p_3 \vee K_c \neg p_3$, ignorance of c about her situation by the negation of this formula. Knowledge of b about his situation is expresed by $K_b p_2 \vee K_b \neg p_2$. Let `bK`, `cK` express that b, c know about their situation. Then updating with public announcement of `cK` and with public announcement of the negation of this have different effects:

```
Caps> showM (upd mo2 (public cK))
==> [0,1]
[0,1]
(0,[p1,p2])(1,[p3,p4])
(a,[[0,1]])
```

```
(b,[[0],[1]])
(c,[[0],[1]])
(d,[[0,1]])

Caps> showM (upd mo2 (public (Neg cK)))
==> [0,1,2,3]
[0,1,2,3]
(0,[p1,p3])(1,[p1,p4])(2,[p2,p3])(3,[p2,p4])
(a,[[0,1,2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1],[2,3]])
(d,[[0,1,2,3]])
```

In both results, b knows about his situation, though:

```
Caps> isTrue (upd mo2 (public cK)) bK
True
Caps> isTrue (upd mo2 (public (Neg cK))) bK
True
```

23 Muddy Children

For this example we need four agents a, b, c, d. Four children a, b, c, d are sitting in a circle. They have been playing outside, and they may or may not have mud on their foreheads. Their father announces: "At least one child is muddy!" Suppose in the actual situation, both c and d are muddy.

a	b	c	d
○	○	●	●

Then at first, nobody knows whether he is muddy or not. After public announcement of these facts, $c(d)$ can reason as follows. "Suppose I am clean. Then $d(c)$ would have known in the first round that she was dirty. But she didn't. So I am muddy." After c, d announce that they know their state, $a(b)$ can reason as follows: "Suppose I am dirty. Then c and d would not have known in the second round that they were dirty. But they knew. So I am clean." Note that the reasoning involves awareness about *perception*.

In the actual situation where b, c, d are dirty, we get:

a	b	c	d
○	●	●	●
?	?	?	?
?	?	?	?
?	!	!	!
!	!	!	!

Reasoning of b: "Suppose I am clean. Then c and d would have known in the second round that they are dirty. But they didn't know. So I am dirty. Similarly for c and d." Reasoning of a: "Suppose I am dirty. Then b, c and d would not have known their situation in the third round. But they did know. So I am clean." And so on ... [20].

Here is the DEMO implementation of the second case of this example, with b, c, d dirty.

```
module Muddy
where

import DEMO

bcd_dirty = Conj [Neg p1, p2, p3, p4]

awareness = [info [b,c,d] p1,
             info [a,c,d] p2,
             info [a,b,d] p3,
             info [a,b,c] p4 ]

aK =  Disj [K a p1, K a (Neg p1)]
bK =  Disj [K b p2, K b (Neg p2)]
cK =  Disj [K c p3, K c (Neg p3)]
dK =  Disj [K d p4, K d (Neg p4)]

mu0 = upd (initE [P 1, P 2, P 3, P 4] [a,b,c,d]) (test bcd_dirty)
mu1 = upds mu0 awareness
mu2 = upd  mu1 (public (Disj [p1, p2, p3, p4]))
mu3 = upd  mu2 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK]))
mu4 = upd  mu3 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK]))
mu5 = upds mu4 [public (Conj[bK, cK, dK])]
```

The initial situation, where nobody knows anything, and they are all aware of the common ignorance (say, all children have their eyes closed, and they all know this) looks like this:

```
Muddy> showM mu0
==> [14]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])(14,[p2,p3,p4])
(15,[p1,p2,p3,p4])
(a,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(b,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(c,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(d,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
```

The awareness of the children about the mud on the foreheads of the others is expressed in terms of update models. Here is the update model that expresses that b, c, d can see whether a is muddy or not:

```
Muddy> showM (info [b,c,d] p1)
==> [0,1]
[0,1]
(0,p1)(1,-p1)
(a,[[0,1]])
(b,[[0],[1]])
```

```
(c,[[0],[1]])
(d,[[0],[1]])
```

Let `awareness` be the list of update models expressing what happens when they all open their eyes and see the foreheads of the others. Then updating with this has the following result:

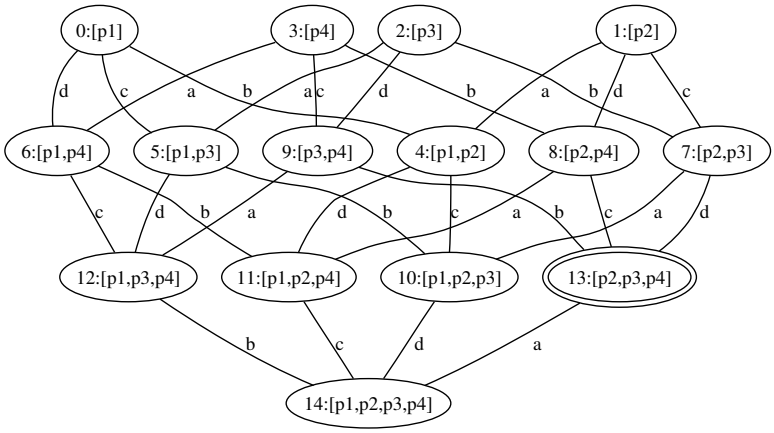
```
Muddy> showM (upds mu0 awareness)
==> [14]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])(14,[p2,p3,p4])
(15,[p1,p2,p3,p4])
(a,[[0,1],[2,5],[3,6],[4,7],[8,11],[9,12],[10,13],[14,15]])
(b,[[0,2],[1,5],[3,8],[4,9],[6,11],[7,12],[10,14],[13,15]])
(c,[[0,3],[1,6],[2,8],[4,10],[5,11],[7,13],[9,14],[12,15]])
(d,[[0,4],[1,7],[2,9],[3,10],[5,12],[6,13],[8,14],[11,15]])
```

Call the result `mu1`. An update of `mu1` with the public announcement that at least one child is muddy gives:

```
Muddy> showM (upd mu1 (public (Disj [p1, p2, p3, p4])))
==> [13]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
(0,[p1])(1,[p2])(2,[p3])(3,[p4])(4,[p1,p2])
(5,[p1,p3])(6,[p1,p4])(7,[p2,p3])(8,[p2,p4])(9,[p3,p4])
(10,[p1,p2,p3])(11,[p1,p2,p4])(12,[p1,p3,p4])(13,[p2,p3,p4])(14,[p1,p2,p3,p4])

(a,[[0],[1,4],[2,5],[3,6],[7,10],[8,11],[9,12],[13,14]])
(b,[[0,4],[1],[2,7],[3,8],[5,10],[6,11],[9,13],[12,14]])
(c,[[0,5],[1,7],[2],[3,9],[4,10],[6,12],[8,13],[11,14]])
(d,[[0,6],[1,8],[2,9],[3],[4,11],[5,12],[7,13],[10,14]])
```

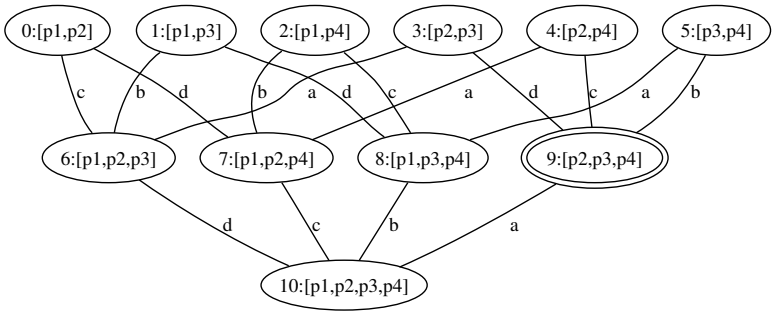
Picture representation (the double oval indicates the actual world):



Call this model `mu2`, and use `aK`, `bK`,`cK`, `dK` for the formulas expressing that a, b, c, d know whether they are muddy (see the code above). Then we get:

```
Muddy> showM (upd mu2 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK])))
==> [9]
[0,1,2,3,4,5,6,7,8,9,10]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])(6,[p1,p2,p3])(7,[p1,p2,p4])(8,[p1,p3,p4])(9,[p2,p3,p4])
(10,[p1,p2,p3,p4])
(a,[[0],[1],[2],[3,6],[4,7],[5,8],[9,10]])
(b,[[0],[1,6],[2,7],[3],[4],[5,9],[8,10]])
(c,[[0,6],[1],[2,8],[3],[4,9],[5],[7,10]])
(d,[[0,7],[1,8],[2],[3,9],[4],[5],[6,10]])
```

Picture representation:

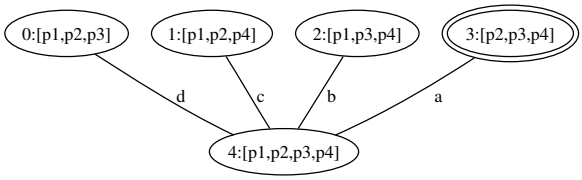


Call this model `mu3`, and update again with the same public announcement of general ignorance:

```
Muddy> showM (upd mu3 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK])))
==> [3]
[0,1,2,3,4]
(0,[p1,p2,p3])(1,[p1,p2,p4])(2,[p1,p3,p4])(3,[p2,p3,p4])(4,[p1,p2,p3,p4])

(a,[[0],[1],[2],[3,4]])
(b,[[0],[1],[2,4],[3]])
(c,[[0],[1,4],[2],[3]])
(d,[[0,4],[1],[2],[3]])
```

Picture representation:



Call this model `mu4`. In this model, b, c, d know about their situation:

```
Muddy> isTrue mu4 (Conj [bK, cK, dK])
True
```

Updating with the public announcement of this information determines everything:

```
Muddy> showM (upd mu4 (public (Conj[bK, cK, dK])))
==> [0]
[0]
(0,[p2,p3,p4])
(a,[[0]])
(b,[[0]])
(c,[[0]])
(d,[[0]])
```

24 Conclusion and Further Work

DEMO was used for solving Hans Freudenthal's Sum and Product puzzle by means of epistemic modelling in [14]. There are many variations of this. See the DEMO documentation at `http://www.cwi.nl/~jve/demo/` for descriptions and for DEMO solutions. DEMO is also good at modelling the kind of card problems described in [13], such as the Russian card problem. A DEMO solution to this was published in [15]. DEMO was used for checking a version of the Dining Cryptographers protocol [8], in [18]. All of these examples are part of the DEMO documentation.

The next step is to employ DEMO for more realistic examples, such as checking security properties of communication protocols. To develop DEMO into a tool for blackbox cryptographic analysis — where the cryptographic primitives such as one-way functions, nonces, public and private key encryption are taken as given. For this, a propositional base language is not sufficient. We should be able to express that an agent A generates a nonce n_A, and that no-one else knows the value of the nonce, without falling victim to a combinatorial explosion. If nonces are 10-digit numbers then not knowing a particular nonce means being confused between 10^{10} different worlds. Clearly, it does not make sense to represent all of these in an implementation. What could be done, however, is represent epistemic models as triples (W, R, V), where V now assigns a non-contradictory proposition to each world. Then uncertainty about the value of n_A, where the actual value is N, can be represented by means of two worlds, one where $n_a = N$ and one where $n_a \neq N$. This could be done with basic propositions of the form $e = M$ and $e \neq M$, where e ranges over cryptographic expressions, and M ranges over 'big numerals'. Implementing these ideas, and putting DEMO to the test of analysing real-life examples is planned as future work.

Acknowledgement The author is grateful to the Netherlands Institute for Advanced Studies (NIAS) for providing the opportunity to complete this paper as Fellow-in-Residence. This report and the tool that it describes were prompted by a series of questions voiced by Johan van Benthem in his talk at the annual meeting of the Dutch Association for Theoretical Computer Science, in Utrecht, on March 5, 2004. Thanks to Johan van Benthem, Hans van Ditmarsch, Barteld Kooi and Ji Ruan for valuable feedback and inspiring discussion. Two anonymous referees made suggestions for improvement, which are herewith gracefully acknowledged.

References

[1] Baltag, A. A logic for suspicious players: epistemic action and belief-updates in games. *Bulletin of Economic Research 54*, 1 (2002), 1–45.

[2] Baltag, A., Moss, L., and Solecki, S. The logic of public announcements, common knowledge, and private suspicions. Tech. Rep. SEN-R9922, CWI, Amsterdam, 1999.

[3] Baltag, A., Moss, L., and Solecki, S. The logic of public announcements, common knowledge, and private suspicions. Tech. rep., Dept of Cognitive Science, Indiana University and Dept of Computing, Oxford University, 2003.

[4] Benthem, J. v. Language, logic, and communication. In *Logic in Action*, J. van Benthem, P. Dekker, J. van Eijck, M. de Rijke, and Y. Venema, Eds. ILLC, 2001, pp. 7–25.

[5] Benthem, J. v. One is a lonely number: on the logic of communication. Tech. Rep. PP-2002-27, ILLC, Amsterdam, 2002.

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

[7] Blackburn, P., de Rijke, M., and Venema, Y. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[8] Chaum, D. The dining cryptographers problem: unconditional sender and receiver untraceability. *Journal of Cryptology 1* (1988), 65–75.

[9] Chellas, B. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[10] Davis, M., Logemann, G., and Loveland, D. A machine program for theorem proving. *Communications of the ACM 5*, 7 (1962), 394–397.

[11] Davis, M., and Putnam, H. A computing procedure for quantification theory. *Journal of the ACM 7*, 3 (1960), 201–215.

[12] Ditmarsch, H. v. *Knowledge Games*. PhD thesis, ILLC, Amsterdam, 2000.

[13] Ditmarsch, H. v. The Russian card problem. *Studia Logica 75* (2003), 31–62.

[14] Ditmarsch, H. v., Ruan, J., and Verbrugge, R. Model checking sum and product. In *AI 2005: Advances in Artificial Intelligence: 18th Australian Joint Conference on Artificial Intelligence* (2005), S. Zhang and R. Jarvis, Eds., vol. 3809 of *Lecture Notes in Computer Science*, Springer-Verlag GmbH, pp. 790–795.

[15] Ditmarsch, H. v., van der Hoek, W., van der Meyden, R., and Ruan, J. Model checking Russian cards. *Electronic Notes Theoretical Computer Science 149*, 2 (2006), 105–123.

[16] Eijck, J. v. Communicative actions. CWI, Amsterdam, 2004.

[17] Eijck, J. v. Reducing dynamic epistemic logic to PDL by program transformation. Tech. Rep. SEN-E0423, CWI, Amsterdam, December 2004. Available from `http://db.cwi.nl/rapporten/`.

[18] Eijck, J. v., and Orzan, S. Modelling the epistemics of communication with functional programming. In *Sixth Symposium on Trends in Functional Programming TFP 2005* (Tallinn, 2005), M. v. Eekelen, Ed., Institute of Cybernetics, Tallinn Technical University, pp. 44–59.

[19] Eijck, J. v., and Ruan, J. Action emulation. CWI, Amsterdam, `www.cwi.nl:~/papers/04/ae`, 2004.

[20] Fagin, R., Halpern, J., Moses, Y., and Vardi, M. *Reasoning about Knowledge*. MIT Press, 1995.

[21] Gerbrandy, J. *Bisimulations on planet Kripke*. PhD thesis, ILLC, 1999.

[22] Gerbrandy, J. Dynamic epistemic logic. In *Logic, Language and Information, Vol. 2*, L. Moss et al., Eds. CSLI Publications, Stanford, 1999.

[23] Goldblatt, R. *Logics of Time and Computation, Second Edition, Revised and Expanded*, vol. 7 of *CSLI Lecture Notes*. CSLI, Stanford, 1992 (first edition 1987). Distributed by University of Chicago Press.

[24] Hintikka, J. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca N.Y., 1962.

[25] Hollenberg, M. *Logic and Bisimulation*. PhD thesis, Utrecht University, 1998.

[26] J.E.Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, Z. Kohavi and A. Paz, Eds. Academic Press, 1971.

[27] Jones, S. P., Hughes, J., et al. Report on the programming language Haskell 98. Available from the Haskell homepage: `http://www.haskell.org`, 1999.

[28] Knuth, D. *Literate Programming*. CSLI Lecture Notes, no. 27. CSLI, Stanford, 1992.

[29] Kooi, B. P. *Knowledge, Chance, and Change*. PhD thesis, Groningen University, 2003.

[30] Koutsofios, E., and North, S. Drawing graphs with *dot*. Available from `http://www.research.att.com/~north/graphviz/`.

[31] Paige, R., and Tarjan, R. E. Three partition refinement algorithms. *SIAM J. Comput. 16*, 6 (1987), 973–989.

[32] Ruan, J. Exploring the update universe. Master's thesis, ILLC, Amsterdam, 2004.

[33] Zhang, H., and Stickel, M. E. Implementing the Davis-Putnam method. *Journal of Automated Reasoning 24*, 1/2 (2000), 277–296.