General Dynamic Dynamic Logic

Patrick Girard Jeremy Seligman

Department of Philosophy
University of Auckland

Fenrong Liu

Department of Philosophy
Tsinghua University

Abstract

Dynamic epistemic logic (DEL) extends purely modal epistemic logic (S5) by adding dynamic operators that change the model structure. Propositional dynamic logic (PDL) extends basic modal logic with programs that allow the definition of complex modalities. We provide a common generalisation: a logic that is 'dynamic' in both senses, and one that is not limited to S5 as its modal base. It also incorporates, and significantly generalises, all the features of existing extensions of DEL such as BMS [3] and LCC [21]. Our dynamic operators work in two steps. First, they provide a multiplicity of transformations of the original model, one for each 'action' in a purely syntactic 'action structure' (in the style of BMS). Second, they specify how to combine these multiple copies to produce a new model. In each step, we use the generality of PDL to specify the transformations. The main technical contribution of the paper is to provide an axiomatisation of this 'general dynamic dynamic logic' (GDDL). This is done by providing a computable translation of GDDL formulas to equivalent PDL formulas, thus reducing the logic to PDL, which is decidable. The proof involves switching between representing programs as terms and as automata. We also show that both BMS and LCC are special cases of GDDL, and that there are interesting applications that require the additional generality of GDDL, namely the modelling of private belief update. More recent extensions and variations of BMS and LCC are also discussed.

Keywords: Dynamic logic, BMS, LCC, PDL, belief change.

Recent research in epistemic logic extends the classical S5-analysis of knowledge with dynamic operators that model the epistemically relevant changes brought about by various acts of communication. These are represented as extensions of the basic epistemic language with expression of the form $[a]\varphi$ interpreted as 'after action a is performed, φ is the case'. The primary example of such an action is the 'public announcement' of a proposition ψ, written $!\psi$, which achieves the right effect by simply removing the $\neg\varphi$-states (those states of the model in which ψ is false), so that everyone subsequently knows that these possibilities are no longer open. [1] A rich array of dynamic operators have been introduced to deal with private communications of various sorts, and also actions that affect more than just the epistemic states of agents, the so-called 'real world changes'. [2]

Meanwhile, interest has grown in applying similar techniques to other branches of modal logic, such as doxastic logic (the logic of belief) [4,18] and preference logic [7,11,12,20]. A significant difference from the epistemic setting is the need to describe dynamic operators that change the relational structure of the underlying model, not just the size of its domain (announcement) or the propositional valuations (real-world change). For example, if one models the doxastic state of an agent by a plausibility relation between epistemically possible state, 'upgrading' a proposition φ, so that it is believed, may be modelled by an operator that transforms the plausibility relation by removing links from φ-states to $\neg\varphi$-states and adding links from $\neg\varphi$-states to φ-states. This ensures that every possible state in which φ is true becomes more plausible (for the given agent) than every possibility in which φ is false. Currently, however, there is no way of adapting the technology of BMS to model the doxastic effect on a multiplicity of agents of one or more of those agents *privately* upgrading their beliefs as a response to a less-than-public communication.

We solve this problem by providing a more general framework, inspired by Theorem 4.11 in [12], first noted in [20], which states that any dynamic operator whose effect on a model can be described in PDL (without Kleene's iteration operator $*$) can be reduced to the underlying modal logic using essentially only the standard axioms of PDL. We show how this idea can be used to extend BMS (and LCC), so that a vast range of dynamic operators can be modelled in a way that allows for private changes and real-world changes in epistemic logic, doxastic logic, preference logic, and any other normal modal logic. [3] We also extend it by adding the Kleene star, so that certain desirable frame conditions (such as transitivity) can be imposed. [4]

Section 1 introduces the concept of a 'PDL-transformation', which is a general

[1] Notoriously, it is not guaranteed that the announced proposition is subsequently known because its very announcement may change its truth value, e.g. announcing 'the sun is shining but you don't know it' results in your knowing that the sun is shining and so making the announcement false.

[2] See for example, the textbooks [23] and [19]. The initial paper on public announcement was [15] and the most significant advance came with the eponymously acronymed BMS [3]. A recent extension of BMS, incorporating real-world changes, is LCC [21], the 'Logic of Communication and Change'.

[3] In particular, we can have dynamic operators over epistemic logics weaker than S5, so catering for those who wish to avoid the controversial properties of positive or negative introspection.

[4] There are various ways in which the Kleene star can be used in dynamics. Our sense will become clear but, for example, it is *not* the sense of [13], which is known to give an undecidable logic.

way of changing models using PDL terms. These transformation are used extensively in Section 2, in which GDDL is defined semantically and then axiomatised, using a technique that exploits the possibility of representing programs both by PDL-terms and by finite state automata. We illustrate GDDL by showing how it can be used to model private belief change. Finally, Section 3 shows how BMS and LCC are special cases of GDDL, and then goes on to discuss other recent variations and extensions such as [2,9,10,22,24].

1 Preliminaries

A *Kripke signature* is a pair $\langle P, R\rangle$ of sets of symbols. The elements of P are *propositional variables* and those of R are *relation symbols.* A model of this signature, $M = \langle W, V\rangle$ consists of a set W (of *states*) and a *valuation* function V mapping each $p \in P$ to $V(p) \subseteq W$ and each $r \in R$ to $V(r) \subseteq W^2$. To describe such structures, we define $T(P, R)$ to be the set of *programs* π and $L(P, R)$ to be the set of *formulas* φ in the usual way:

$$\begin{array}{lcl} \pi & ::= & r \mid \mathsf{E} \mid \varphi? \mid (\pi;\pi) \mid (\pi \cup \pi) \mid \pi^* \\ \varphi & ::= & p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle\pi\rangle\varphi \end{array}$$

where $r \in R$ and $p \in P$. Here, E, is the universal program that jumps between arbitrary states. In each model M, semantic values $[\![\varphi]\!]^M \subseteq W$ and $[\![\pi]\!]^M \subseteq W^2$ are given by:

$$\begin{array}{lcl} [\![p]\!]^M & = & V(p) \\ [\![\neg\varphi]\!]^M & = & W \setminus [\![\varphi]\!]^M \\ [\![(\varphi \wedge \psi)]\!]^M & = & [\![\varphi]\!]^M \cap [\![\psi]\!]^M \\ [\![\langle\pi\rangle\varphi]\!]^M & = & \{u \in W \mid u[\![\pi]\!]^M v \text{ and } v \in [\![\varphi]\!]^M, \text{ for some } v \in W\} \\ & & \\ [\![r]\!]^M & = & V(r) \\ [\![\mathsf{E}]\!]^M & = & W^2 \\ [\![\varphi?]\!]^M & = & \{\langle u, u\rangle \mid u \in [\![\varphi]\!]^M\} \\ [\![\pi_1;\pi_2]\!]^M & = & \{\langle u, v\rangle \mid u[\![\pi_1]\!]^M w \text{ and } w[\![\pi_2]\!]^M v, \text{ for some } w \in W\} \\ [\![\pi_1 \cup \pi_2]\!]^M & = & [\![\pi_1]\!]^M \cup [\![\pi_2]\!]^M \\ [\![\pi^*]\!]^M & = & \begin{array}{l}\{\langle u, v\rangle | u = v \text{ or } u_i[\![\pi]\!]^M u_{i+1} \text{ for some } n \geq 0, u_0, \ldots, u_n \in W \\ \quad \text{such that } u_0 = u \text{ and } u_n = v\}\end{array} \end{array}$$

As usual, we also write $u[\![\pi]\!]^M v$ for $\langle u, v\rangle \in [\![\pi]\!]^M$ and $M, u \models \varphi$ for $u \in [\![\varphi]\!]^M$.

PDL-transformations We will use expressions from our language to describe changes to models. Given a model M of signature $\langle P, R\rangle$, we will show how to obtain a model ΛM of a possibly different signature $\langle Q, S\rangle$ in such a way that we retain some control over which formulas are satisfied in the new model. Specifically, we will also define a (computable) translation φ^Λ of each formula $\varphi \in L(Q, S)$ such that

$$M, u \models \varphi^\Lambda \text{ iff } \Lambda M, u \models \varphi$$

This is the content of Lemma 1.1, below. Specifically, we say that a PDL-*transformation* Λ from signature $\langle P, R\rangle$ to signature $\langle Q, S\rangle$ consists of

(i) a formula $|\Lambda| \in L(P, R)$,

(ii) an algorithm[5] for calculating $\Lambda(q) \in L(P, R)$ for each $q \in Q$ and

(iii) an algorithm for calculating a term $\Lambda(s) \in T(P, R)$ for each $s \in S$.

Now, given a model $M = \langle W, R\rangle$ of signature $\langle P, R\rangle$, we define the model ΛM of signature $\langle Q, S\rangle$ to be $\langle \Lambda W, \Lambda V\rangle$, where

$$\begin{array}{lll} \Lambda W & = & [\![|\Lambda|]\!]^M \\ \Lambda V(q) & = & [\![\Lambda(q)]\!]^M \cap \Lambda W \text{ for each } q \in Q \\ \Lambda V(s) & = & [\![\Lambda(s)]\!]^M \cap \Lambda W^2 \text{ for each } s \in S \end{array}$$

In other words, the domain of the new model is simply a restriction of the domain of the old model (defined by $|\Lambda|$) and the interpretation of the symbols of $\langle Q, S\rangle$ are given by evaluating the corresponding PDL-expressions provided by Λ, and then restricting them to the new domain.

For the translation, we can inductively compute formulas φ^Λ and terms π^Λ of signature $\langle P, R\rangle$ from each formula φ and term π of signature $\langle Q, S\rangle$, as follows:

$$\begin{array}{llllll} q^\Lambda & = & \Lambda(q) & s^\Lambda & = & \Lambda(s); |\Lambda|? \\ (\neg\varphi)^\Lambda & = & \neg\varphi^\Lambda & \mathsf{E}^\Lambda & = & \mathsf{E}; |\Lambda|? \\ (\varphi \wedge \psi)^\Lambda & = & (\varphi^\Lambda \wedge \psi^\Lambda) & (\varphi?)^\Lambda & = & (\varphi^\Lambda)? \\ (\langle\pi\rangle\varphi)^\Lambda & = & \langle\pi^\Lambda\rangle\varphi^\Lambda & (\pi_1; \pi_2)^\Lambda & = & \pi_1^\Lambda; \pi_2^\Lambda \\ & & & (\pi_1 \cup \pi_2)^\Lambda & = & \pi_1^\Lambda \cup \pi_2^\Lambda \\ & & & (\pi^*)^\Lambda & = & (\pi^\Lambda)^* \end{array}$$

Most of the clauses in this definition are fairly obviously what is required. Note, however, the role of formula $|\Lambda|$, which acts as a restriction on the quantifier $\langle\pi\rangle$ in s^Λ, as can be seen by expanding the semantic definition of $\langle s^\Lambda\rangle\varphi$, since

$$M, u \models \langle s^\Lambda\rangle\varphi \quad \text{iff} \quad \exists v : M, v \models |\Lambda|, u[\![s^\Lambda]\!]^M v \,\&\, M, v \models \varphi.$$

As remarked above, the definition is designed precisely so that the following result holds:

Lemma 1.1 *For each state u of ΛM and v of M, and for each formula $\varphi \in L(Q, S)$,* $M, u \models \varphi^\Lambda$ *iff* $\Lambda M, u \models \varphi$, *and*
$u[\![\pi^\Lambda]\!]^M v$ *iff* $v \in \Lambda W$ *and* $u[\![\pi]\!]^{\Lambda M} v$.

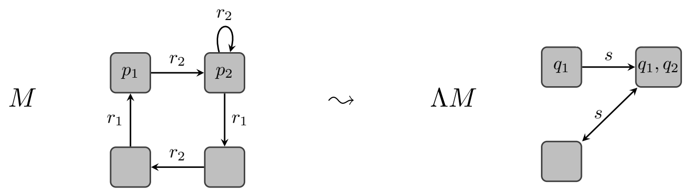
[5] We refer to 'algorithms' here in an informal way, which could be made precise, but doing so would require us to be boringly pedantic about the way the symbols of the signature, for example, are presented, and to choose arbitrarily between many equally good ways of representing these algorithms. Besides, in most cases of interest, the signature $\langle Q, S\rangle$ is finite, and in this case, it is enough merely to list the various components of Λ.

Proof: See Appendix. ☺

Example Let $P = \{p_1, p_2\}$, $R = \{r_1, r_2\}$, $Q = \{q_1, q_2\}$, and $S = \{s\}$. Let Λ be the transformation from $\langle P, R\rangle$ to $\langle Q, S\rangle$ given by

$$\begin{aligned}
&|\Lambda| = \langle r_1\rangle p_1 \vee \langle r_2\rangle p_2\\
&\Lambda(q_1) = \langle r_2\rangle\neg p_1 \quad \text{and} \quad \Lambda(q_2) = \langle p_2?; r_1\rangle\neg p_2\\
&\Lambda(s) = (r_1; r_2) \cup (p_1?; r_2)
\end{aligned}$$

Then with model M as shown below, we get ΛM as follows:



As a simple example of Lemma 1.1 in action, let φ be the formula $\langle s\rangle(q_1 \wedge q_2)$. Then φ^Λ is $\langle((r_1; r_2) \cup (p_1?; r_2)); (\langle r_1\rangle p_1 \vee \langle r_2\rangle p_2)?\rangle(\langle r_2\rangle\neg p_1 \wedge \langle p_2?; r_1\rangle\neg p_2)$. A bit of checking will confirm that $[\![\varphi]\!]^{\Lambda M}$ and $[\![\varphi^\Lambda]\!]^M$ are both equal to the set of states depicted in the left columns of these diagrams.

In what follows we will use a concise notation for PDL transformations, which we illustrate by rewriting Λ, from the above example, as

$$\langle|\langle r_1\rangle p_1 \vee \langle r_2\rangle p_2|, q_1 := \langle r_2\rangle\neg p_1, q_2 := \langle p_2?; r_1\rangle\neg p_2, s := (r_1; r_2) \cup (p_1?; r_2)\rangle$$

To simplify notation further, we will omit trivial domain restriction ($\top$) and those parts of a PDL-transformation that do not change anything. For example, the action of 'public announcement of φ' is just $\langle|\varphi|\rangle$. Finally, the identity transformation is written as I. Some PDL-transformations found in the literature [12,18,20,26] are listed below:

Add $\neg p \to p$ links	=	$\langle r := (r \cup (\neg p?; \mathsf{E}; p?))^*\rangle$ ($*$ used to preserve transitivity)
Delete $p \to \neg p$ links ('suggestion')	=	$\langle r := ((\neg p?; r; \neg p?) \cup (p?; r; p?)$ $\cup(\neg p?; r; p?)\rangle$
Delete $p \to \neg p$ links Add $\neg p \to p$ links ('radical upgrade')	=	$\langle r := ((\neg p?; r; \neg p?) \cup (p?; r; p?)$ $\cup(\neg p?; \mathsf{E}; p?)\rangle$

2 General Dynamic Dynamics

Given a signature $\langle P, R\rangle$, we will define a class of dynamic operators to add to PDL to produce our dynamic dynamic logic, GDDL. Just as with the 'action structures' of BMS, we think of these operators as syntactic objects, albeit somewhat complex ones. A GDDL *dynamic operator* $[A, G, H, a]$ consists of four components:

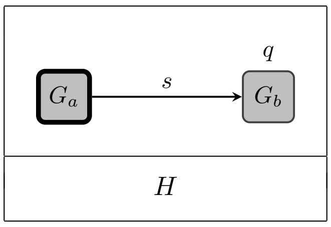
(i) a finite structure $A = \langle D, U\rangle$ of some finite signature $\langle Q, S\rangle$ (distinct from $\langle P, R\rangle$), whose nodes $d \in D$ are called *program nodes*,

(ii) a PDL-transformation G_d from $\langle P, R\rangle$ to $\langle P, R\rangle$ for each $d \in D$,

(iii) a PDL-transformation H from $\langle P \cup Q, R \cup S\rangle$ to $\langle P, R\rangle$, and

(iv) a distinguished element $a \in D$.

We can represent the dynamic operator $[A, G, H, a]$ in the following diagrammatic style:



The upper section contains a representation of the structure A, with each node containing its associated PDL-transformation. Relations $U(s)$ are indicated by arrows labelled with 's', a symbol of S, and $U(q)$ is shown by labelling nodes with 'q' from Q. (The above labels and arrows are merely illustrative.) The distinguished node a is highlighted with darker edges. The lower section contains the PDL-transformation H.

The language of GDDL is given by:

$$\begin{array}{lcl} \pi & ::= & r \mid \mathsf{E} \mid \varphi? \mid (\pi;\pi) \mid (\pi \cup \pi) \mid \pi^* \\ \varphi & ::= & p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle\pi\rangle\varphi \mid [A, G, H, a]\varphi \end{array}$$

where, again, $r \in R$, $p \in P$, and $[A, G, H, a]$ is a GDDL dynamic operator. Let $T^+(P, R)$ and $L^+(P, R)$ be the set of GDDL terms and formulas so defined. Notice, in particular, how the two senses in which the language is 'dynamic' are captured by $\langle\pi\rangle$ and $[A, G, H, a]$. $L(P, R)$ is already dynamic in the first sense but not in the second.

We think of each element d of D as representing a possible action whose effect on M is to transform it to $G_d M$. This could be an announcement, a belief or preference change, or something far more complex, depending on the application. The particular element a is the one that actually occurs. The only restriction is that the transformation is definable by PDL expressions.[6]

[6] In BMS elements of action structures are associated with formulas, called 'pre-conditions' which act to restrict the domain but which have no effect on the relational structure of the

We represent the interaction between $A = \langle D, U\rangle$ and $M = \langle W, V\rangle$ by constructing the model $GM = \langle GW, GV\rangle$ of combined signature $\langle P \cup Q, R \cup S\rangle$, as follows:

$$GW = \{\langle u, d\rangle \mid u \in [\![|G_d|]\!]^M\}$$

Then, for $\langle u, d\rangle$ and $\langle v, e\rangle$ in GW:

$$\begin{array}{lll}
\langle u,d\rangle \in GV(p) & \text{iff } u \in [\![p]\!]^{G_dM} & \text{for each } p \in P \\
\langle u,d\rangle \in GV(q) & \text{iff } d \in U(q) & \text{for each } q \in Q \\
\langle u,d\rangle\, GV(r)\, \langle v,e\rangle & \text{iff } d = e \text{ and } u\, [\![r]\!]^{G_dM}\, v & \text{for each } r \in R \\
\langle u,d\rangle\, GV(s)\, \langle v,e\rangle & \text{iff } u = v \text{ and } d\, U(s)\, e & \text{for each } s \in S
\end{array}$$

We can think of GM as resulting from the process of replacing each program node $d \in D$ by the transformed model G_dM that results from applying the PDL-transformation G_d to M. The structure of A remains, linking these transformed models together.[7]

Finally, we use the transformation H to recover a model of signature $\langle P, R\rangle$ from GM, defining

$$[A, G, H]M \quad = \quad HGM$$

H encodes the way in which the structure of A coordinates the different programs represented by the elements of D. Again, there is great generality here. All that is required is that this means of coordination is, in some sense, PDL-definable.[8] Then, we can specify the semantics for our new dynamic operators in a standard way:[9]

$$M, u \models [A, G, H, a]\varphi \quad \text{iff} \quad [A, G, H]M, \langle u, a\rangle \models \varphi$$

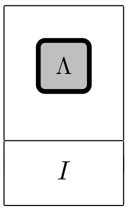
Because of the generality of the approach, it is useful to consider the special case in which a GDDL-operator $[A, G, H, a]$ is defined by a single PDL-transformation Λ, defining

model. See Section 3 for details.

[7] Another useful metaphor for visualising GM is that it is a two-dimensional model in which the S links run in a horizontal direction and the R links run in a vertical direction. Whereas the S links are merely copies of their projection on to D, the R links vary. In the dth place in the horizontal direction the vertical R links form a copy of those in G_dM.

[8] In BMS, the coordination is built into the details of the model construction, not a parameter of the dynamic operators. Also, the signature used for A can be taken to be the same as that for M, since both are simply families of equivalence relations. Again, see Section 3 for details.

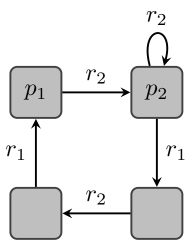
[9] Although a is not relevant to computing $[A, G, H]M$, we define $[A, G, H, a]M = [A, G, H]M$, for uniformity of notation when we consider arbitrary operators.

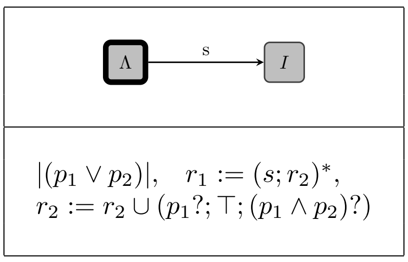
$[\Lambda] =$ 

Example 2: For a purely abstract example, illustrating various aspects of the definition, take Λ to be the PDL-transformation

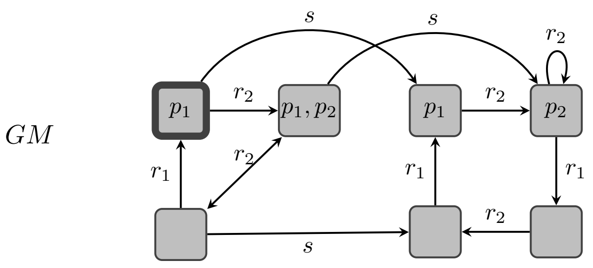
$$\langle |\langle r_1\rangle p_1 \vee \langle r_2\rangle p_2|, p_1 := \langle r_2\rangle\neg p_1, p_2 := \langle p_2?; r_1\rangle\neg p_2, r_2 := (r_1; r_2) \cup (p_1?; r_2)\rangle$$

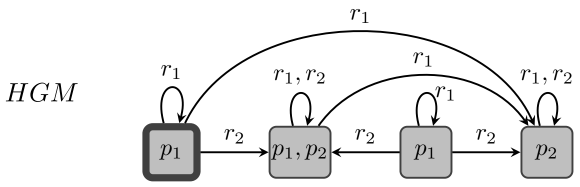
Take the model M of example 1 (shown left) and the dynamic operator $[A, G, H, a]$ (shown right):




Λ
s
I
$|(p_1 \vee p_2)|, \quad r_1 := (s; r_2)^*,$
$r_2 := r_2 \cup (p_1?; \top; (p_1 \wedge p_2)?)$

We first compute GM, then HGM:





2.1 Private Belief Change

As an example of what can be done with GDDL, and a concrete illustration of the definitions in action, we will consider an application to doxastic logic. For the basic logic of belief change, we follow van Benthem's [18] account, in which the effect of a belief update is to change an agent's judgements about the relative plausibility ($\leq$) of different epistemic possibilities. The basic idea is that when updating with p, an agent should judge any p-state to be strictly more plausible than any $\neg p$-state, while retaining her earlier judgements about relative plausibility among p-states and among $\neg p$-states. Belief is defined as truth in maximally plausible states. We extend this to a multi-agent context in which belief changes are private. There are many options here, but we choose to add a separate relation ($\sim$) modelling knowledge (epistemic indistinguishability). In this example, we will assume that it is an equivalence relation, as is standard, but it should be clear that many variations are possible within the present framework.

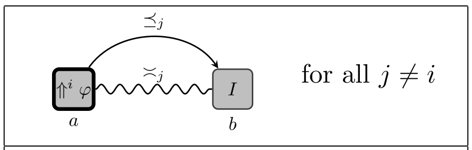
Given a finite sets of agents I, consider a Kripke signature $\langle P, R\rangle$ with $R = \{\sim_i, \leq_i, \beta_i \mid i \in I\}$, the class $\mathcal{D}$ of models $M = \langle W, V\rangle$ of this signature for which, for each agent i, $V(\sim_i)$ is a equivalence relation, $V(\leq_i) \subseteq V(\sim_i)$ is a preorder, and

$$u\beta_i v \text{ iff } u \sim_i v \text{ and } w \leq_i v \text{ for each } w \in W \text{ such that } v \leq_i w.$$

In other words, a state v is β_i-accessible from u iff it is maximally plausible among those states that are epistemically indistinguishable from u. In this way, the formula $[\beta_i]\varphi$ states that φ is believed by i, according to the analysis of [18].[10] Updating agent i's beliefs with φ results in a new plausibility relation defined by[11] $(\varphi?; \leq_i; \varphi?) \cup (\neg\varphi?; \leq_i; \neg\varphi?) \cup (\neg\varphi?; \sim_i; \varphi?)$.

The PDL-transformation $\Uparrow^i \varphi$ ('update i's beliefs with φ') maps $\leq_i$ to this PDL term, keeping everything else the same. The fact that agent i believes ψ after upgrading her beliefs with φ should then expressed in GDDL as $[\Uparrow^i \varphi][\beta_i]\psi$.

But there is a problem. Not only is $[\Uparrow^i p][\beta_i]p$ valid (as expected), but so too is $[\Uparrow^i p][\beta_j][\beta_i]p$, for any other agent j. In other words, it is logically true that after i doxastically updates with p, not only does she believe that p but everyone else believes that she believes p. In this way, $[\Uparrow^i \varphi]$ is not really *private* at all. This situation is familiar from BMS, and we adopt a similar solution, but one that exploits the relation-change potential of PDL-transformations, defining the operator $(\uparrow^i \varphi) = [A, G, H, a]$ as follows:



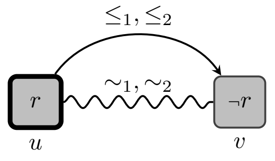
$\sim_j := (\sim_j \cup \asymp_j)^*$ $\quad$ $\leq_j := (\leq_j \cup \preceq_j)^*$

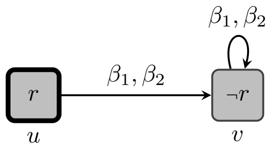
[10] Note that although β_i is in some sense redundant, in that it is fully determined by $\sim_i$ and $\leq_i$, the operator $[\beta_i]$ is not modally definable from the operators $[\sim_i]$ and $[\leq_i]$; nonetheless, the relationship between the three operators can easily be acclimatised.

[11] This differs slightly from [18] because of the restriction to the $\sim_i$-equivalence class discussed above.

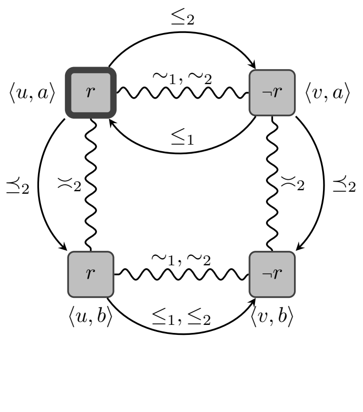
Here, A is a two-state structure with domain $D = \{a, b\}$. It has signature $\langle Q, S\rangle$, where Q is empty and $S = \{\asymp_j, \preceq_j \mid j \in I\}$, and interprets $\asymp_j$ as an equivalence relation and $\preceq_j$ as a preorder, although not all of the links are shown in the diagram. The distinguished node a contains the PDL-transformation $\Uparrow^i \varphi$, representing the action of updating i's belief's with φ, and the other node, b, contains the identity transformation I, representing the action of doing nothing. Only agent i knows which of the two possible actions were performed, so $a \asymp_j b$ for all $j \neq i$. Likewise, we will assume (although there is room for more subtlety here) that each of these other agents regards it as more plausible that i's beliefs have not changed. This is captured by making $a \preceq_j b$ and not $b \preceq_j a$ for all those j. Finally, the integrating transformation H is defined by taking the (reflexive, transitive closure of) the unions of the two epistemic and the two doxastic relations.

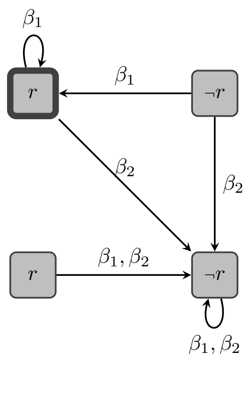
To see how this works, we will consider an application of $(\uparrow^i \varphi)$ to the model M displayed below:[12]





We will think of M as representing a scenario with two agents, called 1 and 2, who both falsely believe $\neg r$: 'it hasn't rained today'. Now, the operator $\uparrow^1 r$ should represent the action of agent 1 privately upgrading her belief in r, in a way that is unobserved by agent 2; perhaps she takes a furtive glance out of the window and sees someone closing an umbrella. The resulting model $(\uparrow^1 r)$ is shown below (with β_i displayed separately):





[12] Throughout this section reflexive loops and compositions of arrows for $\sim$ and $\leq$ are suppressed in diagrams, and the relations β_1 and β_2, which are defined in terms of the others, are shown separately.

Here, the designated state satisfies $[\beta_1]r$ (agent 1 believes it to be raining), $[\beta_2]\neg r$ (agent 2 still believes it is not raining) and $[\beta_2][\beta_1]\neg r$ (agent 2 also believes, falsely, that agent 1 still believes it not to be raining).[13]

2.2 Axiomatisation

The key to understanding the logic of GDDL is to find a computable translation $\varphi^{[A,G,H,a]}$ of each formula φ in $L(P,R)$ such that

$$[A,G,H,a]\varphi \leftrightarrow \varphi^{[A,G,H,a]}$$

is valid as a formula of GDDL. From this (and the replacement of logical equivalents) it follows that every formula of $L^+(P,R)$ is equivalent to a formula of $L(P,R)$, and can be proved to be so using these equivalences as axioms. We can reduce the two senses of 'dynamic' in dynamic dynamic logic to one.

How, then, to define $\varphi^{[A,G,H,a]}$? Our approach will be to define a formula $\varphi^{[A,G,d]}$ of $L(P,R)$ for each φ of $L(P \cup Q, R \cup S)$ and a program $\pi^{[A,G,d,e]}$ of $T(P,R)$ for each π of $T(P \cup Q, R \cup S)$ such that for any model M of signature $\langle P,R\rangle$, the following result holds:

Lemma 2.1 *For each $\langle u,d\rangle, \langle v,e\rangle \in GW$,*

(i) *$GM, \langle u,d\rangle \models \varphi$ iff $M,u \models \varphi^{[A,G,d]}$, and*
(ii) *$\langle u,d\rangle[\![\pi]\!]^{GM}\langle v,e\rangle$ iff $u[\![\pi^{[A,G,d,e]}]\!]^{M}v$*

This will be proved below. We can then define $\varphi^{[A,G,H,d]} = \varphi^{H\,[A,G,d]}$ so that

Lemma 2.2 *For each operator $[A,G,H,a]$ of* GDDL *and each formula φ of $L(P,R)$, the following is valid:*

$$[A,G,H,a]\varphi \leftrightarrow \varphi^{[A,G,H,a]}$$

Proof: From Lemmas 2.1 and 1.1. (Note that, since GM is of signature $\langle P \cup Q, R \cup S\rangle$ and $[A,G,H,d]M = HGM$ is of signature $\langle P,R\rangle$, the formula φ^H is in $L(P \cup Q, R \cup S)$, as required by Lemma 2.1.) ☺

To define $\pi^{[A,G,d,e]}$ we need a small excursion into automata theory.[14] For

[13] Although the definition of a private belief update operator is only an example to show what can be done in GDDL, it illustrates the need for some constraints in getting sensible results for epistemic logic. In particular, it is important that the epistemic relation $\asymp_i$ in the operator constrains the definition of the various transformations G_d so that each agent knows that actions affecting her own psychological state have occurred, namely, that $d \asymp_i e$ implies both $G_d(\asymp_i) = G_e(\asymp_i)$ and $G_d(\leq_i) = G_e(\leq_i)$.

[14] This excursion into automata theory is solely for the purpose of producing the reduction axioms, in a recursive way. Once the axioms are produced, however, no essential use of automata remains in the logic. We find the technique useful and illuminating but recognise that there may be an alternative approach that provides reduction axioms in a more direct way. We leave this as an open problem.

each signature, say that σ is a *basic program* of that signature if it is either a relation symbol, E, or a test. Then for each model N and a program π, define $\Sigma(N, u, v, \pi)$ to be the set of strings $\sigma_1 \dots \sigma_n$ of basic subprograms of π such that $u_i [\![\sigma_i]\!]^N u_{i+1}$ for some sequence $u_0, \dots, u_n$ of states in N with $u_0 = u$ and $u_n = v$. Say that a finite state automaton A over an alphabet of basic subprograms of a program π *represents* π iff for all models N and states u and v,

$$u[\![\pi]\!]^N v \text{ iff some word of } \Sigma(N, u, v, \pi) \text{ is accepted by } A.$$

It has been well known since [16] that every PDL program is represented by some automaton and every automaton represents some PDL program.[15] Moreover, each can be computed from the other.[16] Now, given φ in $L(P \cup Q, R \cup S)$ and π in $T(P \cup Q, R \cup S)$ and states $d, e \in D$, we will define $\varphi^{[A,G,d]}$ and $\pi^{[A,G,d,e]}$ by mutual induction.

The definition of $\varphi^{[A,G,d]}$ is straightforwardly inductive:

$$\begin{array}{ll}
p^{[A,G,d]} & = G_d(p) \\
q^{[A,G,d]} & = \begin{cases} \top \text{ if } d \in U(q) \\ \bot \text{ otherwise} \end{cases} \\
(\neg\varphi)^{[A,G,d]} & = \neg\varphi^{[A,G,d]} \\
(\varphi \wedge \psi)^{[A,G,d]} & = (\varphi^{[A,G,d]} \wedge \psi^{[A,G,d]}) \\
(\langle\pi\rangle\varphi)^{[A,G,d]} & = \bigvee_{e \in D} \langle \pi^{[A,G,d,e]} \rangle (|G_e| \wedge \varphi^{[A,G,e]})
\end{array}$$

The program $\pi^{[A,G,d,e]}$ is obtained by constructing a corresponding automaton, which will refer to $\psi^{[A,G,d]}$ for subprograms $\psi?$ of π, which is inductively legitimate. This will take the next couple of paragraphs.

First, consider an automaton A_π which represents π. Let A_π have states X, of which $X_0 \subseteq X$ are initial states, $X_1 \subseteq X$ are accepting states, and for each σ (a basic subprogram of π), $T(\sigma) \subseteq X^2$ is such that there is a transition from x_1 to x_2 labelled by σ iff $\langle x_1, x_2 \rangle \in T(\sigma)$.

Now for each symbol σ in the alphabet of A_π (a basic subprogram of π) and each $c_1, c_2 \in D$, define σ^{c_1,c_2} as follows:

$$\sigma^{c_1,c_2} = \begin{cases}
\psi^{[A,G,c]}? & \text{if } \sigma = \psi? \text{ and } c_1 = c_2 = c \\
G_c(\sigma) & \text{if } \sigma \in R \text{ and } c_1 = c_2 = c \\
|G_{c_1}|?; \mathsf{E}; |G_{c_2}|? & \text{if } \sigma = \mathsf{E} \\
\top? & \text{if } \sigma \in S \text{ and } \langle c_1, c_2 \rangle \in U(\sigma) \\
\bot? & \text{otherwise}
\end{cases}$$

[15] The representation depends only on the compositional structure of the program not on the particular choice of basic programs, so additions to PDL such as tests and E are not a problem.

[16] The complexity of translating between the two representations has been investigated in [8].

Construct a new automaton $B^{d,e}_{\pi}$, whose alphabet consists of the basic programs σ^{c_1,c_2} where σ is in the alphabet of A_π, with states $X' = X \times D$, initial states $X'_0 = X_0 \times \{d\}$, accepting states $X'_1 = X_1 \times \{e\}$, and transition function T' defined by

$$T'(\tau) \quad = \quad \{\langle\langle x_1, c_1\rangle, \langle x_2, c_2\rangle\rangle \mid \text{for some } \sigma,\ \langle x_1, x_2\rangle \in T(\sigma) \text{ and } \sigma^{c_1,c_2} = \tau\}$$

Now, let $\pi^{[A,G,d,e]}$ be the program of $T(P,R)$ represented by $B^{d,e}_{\pi}$.

The two automata are designed to be synchronised in the sense given by the following technical lemma:

Lemma 2.3 *Assume that for each test $\psi?$ occurring in π, and each $\langle w, c\rangle \in GW$, $GM, \langle w, c\rangle \models \psi$ iff $M, w \models \psi^{[A,G,c]}$. Then, given $x_1, x_2 \in X$ and $\langle u, c_1\rangle, \langle v, c_2\rangle \in GW$, consider the following properties of the labels of the automata A_π and $B^{d,e}_{\pi}$:*

$$\gamma(\sigma): \quad \begin{array}{c}\langle x_1, x_2\rangle \in T(\sigma)\\ \text{and}\\ \langle u, c_1\rangle [\![\sigma]\!]^{GM} \langle v, c_2\rangle\end{array} \qquad\qquad \gamma'(\tau): \ \begin{array}{c}\langle\langle x_1, c_1\rangle, \langle x_2, c_2\rangle\rangle \in T'(\tau)\\ \text{and}\\ u[\![\tau]\!]^{M} v\end{array}$$

Then for each symbol τ of the alphabet of $B^{d,e}_{\pi}$,

$$\gamma'(\tau) \text{ iff } \gamma(\sigma) \text{ and } \sigma^{c_1,c_2} = \tau \text{ for some } \sigma \text{ in the alphabet of } A_\pi$$

Proof: See Appendix. ☺

We are now ready to prove Lemma 2.1.

Proof of Lemma 2.1: The *rank* of a formula or program is defined as follows. Formulas of rank n are not of rank $n-1$ but contain no programs of rank n, and programs of rank n not of rank $n-1$ but contain no test formulas of rank n. (In particular, formulas of rank 0 contain no programs and programs of rank 0 contain no test formulas; formulas of rank 1 contain at least one program of rank 0 but none of rank 1 and programs of rank 1 contain at least one test formula of rank 0 but none of rank 1, etc.) To prove the lemma we show, by induction on the rank n of a formula φ of $L(P \cup Q, R \cup S)$, that for $\langle u, d\rangle, \langle v, e\rangle \in W'$,

(i) $GM, u, d \models \varphi$ iff $M, u \models \varphi^{[A,G,d]}$, and

(ii) for any program π of rank $\leq n$, $\langle u, d\rangle [\![\pi]\!]^{GM} \langle v, e\rangle$ iff $u[\![\pi^{[A,G,d,e]}]\!]^{M} v$

We prove part 1 by induction on the structure of φ.

For propositional variables:
$GM, u, d \models p$ iff $u \in V_d(p)$ iff $u \in [\![G_d(p)]\!]^{M}$ iff $M, u \models G_d(p)$ iff $M, u \models p^{[A,G,d]}$
$GM, u, d \models q$ iff $d \in U(q)$ iff $q^{[A,G,d]} = \top$ iff $M, u \models q^{[A,G,d]}$ (given that $q^{[A,G,d]} \in \{\top, \bot\}$)

For Booleans ($\neg$ and $\wedge$), the proof is straightforwardly inductive.

For formulas of the form $\langle\pi\rangle\psi$, note that π must be of rank $< n$ and so for each $\langle w, f\rangle \in W'$

$$\langle u, d\rangle [\![\pi]\!]^{GM} \langle w, f\rangle \text{ iff } u[\![\pi^{[A,G,d,f]}]\!]^M v$$

and by the (inner, structural) inductive hypothesis,

$$GM, w, f \models \psi \text{ iff } M, w, \models \psi^{[A,G,f]}$$

But then the following are equivalent:

$GM, u, d \models \langle\pi\rangle\psi$
$\langle u, d\rangle [\![\pi]\!]^{GM} \langle w, f\rangle$ and $GM, w, f \models \psi$ for some $\langle w, f\rangle \in W'$
$u[\![\pi^{[A,G,d,f]}]\!]^M w$ and $M, w, \models \psi^{[A,G,f]}$ for some $\langle w, f\rangle \in W'$
$M, w \models |G_f|$, $u[\![\pi^{[A,G,d,f]}]\!]^M w$ and $M, w, \models \psi^{[A,G,f]}$ for some $f \in D, w \in W$
$M, u \models \bigvee_{f\in D} \langle\pi^{[A,G,d,f]}\rangle(|G_f| \wedge \psi^{[A,G,f]})$
$M, u \models \langle\pi\rangle\psi^{[A,G,d]}$

For part 2, we know that any test formula in π is of rank $< n$. So suppose $\langle u, d\rangle [\![\pi]\!]^{GM} \langle v, e\rangle$. Then by choice of A_π we have that some word $\sigma_1 \ldots \sigma_n$ in $\Sigma(GM, \langle u, d\rangle, \langle v, e\rangle, \pi)$ is accepted by A_π. This implies that

(i) there are $u_0, \ldots, u_n$ and $c_0, \ldots, c_n \in D$ such that $u_0 = u$, $c_0 = d$, $u_n = v$ $c_n = e$ and $\langle u_i, c_i\rangle [\![\sigma_i]\!]^{GM} \langle u_{i+1}, c_{i+1}\rangle$ for $0 \le i < n$, and

(ii) there are $x_0, \ldots, x_n \in X$ and such that $x_0 \in X_0$, $x_n \in X_1$, and $\langle x_i, x_{i+1}\rangle \in T(\sigma_i)$ for $0 \le i < n$.

Now for each i we have $\gamma_i(\sigma_i)$:

$$\langle x_i, x_{i+1}\rangle \in T(\sigma_i) \quad \text{and} \quad \langle u_i, c_i\rangle [\![\sigma_i]\!]^{GM} \langle u_{i+1}, c_{i+1}\rangle$$

So, by Lemma 2.3, for $\tau_i = \sigma_i^{c_i, c_{i+1}}$, we have $\gamma'_i(\tau_i)$:

$$\langle\langle x_i, c_i\rangle, \langle x_{i+1}, c_{i+1}\rangle\rangle \in T'(\tau_i) \quad \text{and} \quad u_i[\![\tau_i]\!]^M u_{i+1}$$

Also, since $c_0 = d$, $c_n = e$, $x_0 \in X_0$, $x_n \in X_1$, we have that $\langle x_0, c_0\rangle \in X'_0$ and $\langle x_n, c_n\rangle \in X'_1$. Thus:

(i) there are $u_0, \ldots, u_n$ such that $u_0 = u$, $u_n = v$ and $u_i[\![\tau_i]\!]^M u_{i+1}$ for $0 \le i < n$, and

(ii) there are $x_0, \ldots, x_n \in X$ and $c_0, \ldots, c_n \in D$ such that $\langle x_0, c_0\rangle \in X_0$, $\langle x_n, c_n\rangle \in X_1$, and $\langle\langle x_i, c_i\rangle, \langle x_{i+1}, c_{i+1}\rangle\rangle \in T'(\tau_i)$ for $0 \le i < n$.

This is precisely what is required for $\tau_1 \ldots \tau_n$ to be accepted by $B_\pi^{d,e}$. Then by definition of $\pi^{[A,G,d,e]}$, we have that $u[\![\pi^{[A,G,d,e]}]\!]^M v$, as required. The converse is proved similarly. ☺

Theorem 2.4 *The logic of* GDDL *is completely axiomatised by the axioms and rules of* PDL *(see Definition 4.78 in [5]) and the schema*

$$\vdash [A, G, H, a]\varphi \leftrightarrow \varphi^{[A,G,H,a]}$$

Corollary 2.5 GDDL *is decidable.*

Proof: We have a computable reduction of GDDL to its PDL fragment, which is itself decidable. ☺

3 Applications

In the remainder of the paper, we show how two well-known systems for dynamic epistemic logic (BMS and LCC) are special cases of GDDL and make further connections to more recent developments.

3.1 BMS

For BMS [3], we will be working with a signature $\langle P, R\rangle$ for which P is a (countably infinite) set of propositional variables and $R = \{K_i \mid i \in I\}$ is a set of epistemic relations, one for each agent $i \in I$, with I finite. The BMS system is not dynamic in the first (PDL) sense and so we will refer to basic modal language (in which the only terms are the atoms K_i) as $L^-(P, R)$. A model $M = \langle W, V\rangle$ of this signature is an *epistemic* model iff all the relations $V(K_i)$ are equivalence relations. So far, this is all just standard epistemic logic. The innovation was to define an *action* structure to be a structure of the form $\langle D, U, \mathsf{pre}\rangle$ for which $\langle D, U\rangle$ is an epistemic structure and $\mathsf{pre}\colon D \to L^-(P, R)$ assigns a formula to each element of D that expresses the 'precondition' of performing the action it represents. For example, if d represents the announcement of φ, then it is usually assumed that, as a precondition, the announcement must be true, and so $\mathsf{pre}(d) = \varphi$. Action structures are finite and so can be added to the syntax as dynamic operators. The full language of BMS is thus

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle K_i\rangle\varphi \mid [D, U, \mathsf{pre}, a]\varphi$$

where $\langle D, U, \mathsf{pre}\rangle$ is an action structure and $a \in D$ is the designated action. Given action structure $\Delta = \langle D, U, \mathsf{pre}\rangle$ and epistemic model $M = \langle W, V\rangle$, the *product* model ΔM is defined to be $\langle \Delta W, \Delta V\rangle$, where

$$\begin{array}{ll} \Delta W & = \{\langle u, d\rangle \mid M, u \models \mathsf{pre}(d)\} \\ \Delta V(p) & = \{\langle u, d\rangle \in \Delta W \mid u \in V(p)\} \text{ for each } p \in P \\ \Delta V(K_i) & = \{\langle\langle u, d\rangle, \langle v, e\rangle\rangle \in (\Delta W)^2 \mid \langle u, v\rangle \in V(K_i) \text{ and } \langle d, e\rangle \in U(K_i)\} \end{array}$$

Finally, the semantics of the BMS operator $[D, U, \mathsf{pre}, a]$ is given by

$$M, u \models [D, U, \mathsf{pre}, a]\varphi \quad \text{iff} \quad \langle D, U, \mathsf{pre}\rangle M, \langle u, a\rangle \models \varphi$$

This can be seen as a special case of our general construction. First, we take a copy K'_i of each symbol K_i, because we need to keep the signature of the epistemic model distinct from that of the action structure. Then we define the structure $A = \langle D, U' \rangle$ of signature $\langle Q, S \rangle$, with $Q = \emptyset$ and $S = \{K'_i \mid i \in I\}$, such that $U'(K'_i) = U(K_i)$, for each $i \in I$. For each $d \in D$ we define the transformation G_d by $\langle |G_d| = \mathsf{pre}(d) \rangle$. This captures the idea of a precondition. Finally, we take H to be the PDL-transformation given by $\langle K_i := K_i; K'_i \rangle$. To show that $[D, U, \mathsf{pre}, a]\varphi$ is logically equivalent to $[A, G, H, a]\varphi$, the following theorem is sufficient.

Theorem 3.1 *With $\Delta = \langle D, U, \mathsf{pre} \rangle$ and A, G and H defined as above,*

$$\Delta M = [A, G, H]M$$

Proof: See Appendix. ☺

It follows that every function on the class of models of a given signature that is definable by a BMS operator is also definable by a GDDL operator.

3.2 LCC

LCC [21], the Logic of Communication and Change, extends BMS in two ways: by expanding the base language to include PDL modalities, and by introducing 'real-world' change. The first extension is relatively straightforward. It just amounts to moving from $L^-(P, R)$ to $L(P, R)$, and the argument that the resulting system is a fragment of GDDL goes through as above. The second extension, to model 'real-world' change, is achieved using 'propositional substitutions', which are functions $\sigma : P \to L(P, R)$ with a finite base, meaning that σ is the identity function on all but a finite number of propositional variables. An action that changes something other than just the psychological states of agents can thus be represented by a propositional substitution σ such that, after the change, p is true of state u iff $\sigma(p)$ were true of it before the change.[17]

A LCC *action structure* $\Delta = \langle D, U, \mathsf{pre}, \mathsf{sub} \rangle$ consists of a BMS-like[18] action structure $\langle D, U, \mathsf{pre} \rangle$ and a propositional substitution function sub_d for each $d \in D$. Given an epistemic model $M = \langle W, V \rangle$, the LCC *product* model ΔM is defined to be $\langle \Delta W, \Delta V \rangle$ as for BMS, except that

$$\Delta V(p) \quad = \{\langle u, d \rangle \in \Delta W \mid u \in V(\mathsf{sub}_d(p))\} \text{ for each } p \in P$$

To extend our earlier representation of BMS operators in GDDL requires only one small change: the PDL-transformation G_d is now defined by $\langle \mathsf{pre}(d), p :=$

[17] The restriction to σ of finite base requires the changes to be, in some sense, local. However, the embedding of LCC in GDDL shows that what is important here is only that there is some finite representation of σ, on the basis of which σ can be recovered algorithmically.

[18] The only difference is that $\mathsf{pre}(d)$ is not restricted to $L^-(P, R)$; it may be any formula of $L(P, R)$.

$\mathsf{sub}_d(p)\rangle$. With A and H defined as for BMS, we have the required result:

Theorem 3.2 *With $\Delta = \langle D, U, \mathsf{pre}, \mathsf{sub}\rangle$ and A, G and H defined as above,*

$$\Delta M = [A, G, H]M$$

Proof: See Appendix. ☺

It follows that every function on the class of models of a given signature that is definable by a BMS operator is also definable by a GDDL operator.

3.3 Other approaches

LCC was extended in [24] by basing it on PDL with relational converse (which is still decidable). While GDDL does not include relational converse in the base language, we can think of no reason why it could not be added, in a similar manner to our addition of E over the vanilla PDL. We conjecture that other decidable extensions of PDL could be taken as the base logic. The combination of converse with nominals (which is decidable from [14,1], p.12, Theorem 3.5) should present no further difficulty. Another particularly useful decidable addition would be atomic negation on programs (see [25]) on the basis of which E and the 'window' operators are definable.

There have been several attempts to extend dynamic logic to cope with relation-changing dynamic operators. Firstly, [22] shows how, for given finite epistemic models M_1 and M_2, it is possible to define a BMS operator $[D, U, \mathsf{pre}, a]\varphi$ of BMS such that $[D, U, \mathsf{pre}, a]\varphi M_1 = M_2$ but this is far from what is needed to represent relation-changing operators that have a uniform action on all models.

Next, [2] presents an interesting extension of modal logic with 'graph-modification' operators for models of arbitrary signature (as with GDDL). There are three kinds of operator: 'label modifications' $p + \varphi$ and $p - \varphi$, which increase (and respectively decrease) $V(p)$ by $\{u \mid M, u \models \varphi\}$; 'edge label modification' operators $r + (\varphi, \psi)$ and $r - (\varphi, \psi)$, which increase (resp. decrease) $V(r)$ by the set $\{\langle u, v\rangle \mid M, u \models \varphi \text{ and } M, u \models \psi\}$; and 'new state' operators nw and $\overrightarrow{nw}$, which add a new isolated state to the domain (and shifts the point of evaluation to it, in the case of $\overrightarrow{nw}$). Of these, the label modification and edge label modification operators define PDL-transformations and so can be represented in GDDL. The new state operators are more tricky. Instead, in GDDL, we can represent the addition of a set of (isolated) states, either with or without a shift in evaluation point, but we know of no way to ensure that this set is a singleton.[19] Importantly, GDDL extends the approach of [2] by

[19] For the analogue of nw, use the operator $[A, G, H, a]$ in which A has domain $\{a, b\}$ with an empty signature, G_a is the identity, G_b maps all propositional variables to $\bot$ and all relation symbols to $\bot?$, and H is also the identity. Then if $M = \langle W, V\rangle$, $[A, G, H, a]M$ is isomorphic to $\langle W \cup W', V\rangle$ with W' disjoint from W. For $\overrightarrow{nw}$, use $[A, G, H, b]$. The size of W' can be diminished using a domain restriction component to G_b but without formulas that are

including all PDL-transformations and products.[20]

Finally, [9] and [10] propose a system of 'arrow update' operators with a novel syntax in which new transitions for a relation are introduced by specifying their pre- and post-conditions. They show that the resulting system is dynamically equivalent to BMS, and so can be seen as a way of giving an explicit syntax for the part of BMS that allows a limited form of relation-change. GDDL is a genuine extension of both, as can be seen from the example of private belief change in Section 2.1.[21]

4 Conclusion

GDDL achieves our objective of generalising existing approaches to dynamic epistemic logic to allow relational change and opens up a number of possibilities for further work. Firstly, it would be interesting to look at fragments that can be expressed in a more restricted syntax. Even the syntactically promiscuous logics of BMS and LCC exploit only a small part of the generality of GDDL operators, suggesting that other restrictions may be equally interesting in their own right. A proper study of these would require a better understanding of the class of model-transforming functions defined by GDDL. A good place to start with this investigation is the category of PDL-transformations of models of arbitrary signatures.

Secondly, we propose that many applications would profit from the ability to code appropriate GDDL operators. For example, the study of the relationship between first and higher-order psychological attitudes requires the interplay between levels available in GDDL. This arises in preference logic when trying to account for weakness of will, one analysis of which is the preference for having different preferences: I may prefer an action in which my preference for smoking is downgraded to one in which it is not. Moving from the personal setting, similar level-distinctions occur when reflecting on normative systems. Certain changes to the law, for example, may be regarded as permissible, while others are not. And with a multi-agent perspective, one could try to devise GDDL operators to model changes to conflicting normative systems, and so provide a logic for reasoning about the effect of those changes, potentially giving a new approach to reasoning about conflict resolution. When the relations in a

guaranteed to be true at only one state (such as the nominals of hybrid logic), we cannot ensure that it is a singleton.

[20] [2] also discusses 'local' graph modification operators, which significantly extend the expressive power of their system, sufficient to express the hybrid binder $\downarrow x$, and so leads to undecidability ([6]). Augmentation of GDDL with a distinguished nominal for the point of evaluation should have a similar effect.

[21] [9] only allows a uniform update for all agents (the 'common update policy') but hints at the possibility of providing distinct updates for each agent 'privately'. This is done in [10], but these updates are not really 'private', just as our $\Uparrow_i$ operator (Section 2.1) is not private. To achieve the genuine privacy of $\uparrow_i$, some (as yet unformulated) hybrid of BMS and arrow update logic would be needed.

model are understood as state-transitions of some process (as in the standard computational interpretation of PDL), GDDL operators encode operations for changing what is possible to do, and so give a basis for reasoning about design.

Thirdly, from a technical point of view, the interaction between levels raises interesting questions about the framing of general constraints on those interactions so as to ensure sensible results. We saw an example of this in our brief exploration of private belief change (Footnote 13) but as yet we have no idea about how to frame a general theory of such constraints.

These suggestions are of course very speculative but they give a sense of how GDDL could be used to open up a new area for applications. Our motivations for developing the system arose from technical considerations in an ongoing project called 'logic in the community' [17], which aims at studying the consequences of social relationships for our understanding of rational procedures. We expect there to be many further uses for GDDL in that project.

References

[1] Areces, C., P. Blackburn and M. Marx, *The computational complexity of hybrid temporal logics*, Logic Journal of the IGPL **8** (2000), pp. 653–679.

[2] Aucher, G., P. Balbiani, L. F. del Cerro and A. Herzig, *Global and local graph modifiers*, in: *Proceedings of the 5th workshop on methods for modalities (MAM5 2007)*, Electronic notes in theoretical computer science **231** (2009), pp. 293–307.

[3] Baltag, A., L. S. Moss and S. Solecki, *The logic of public announcements, common knowledge and private suspicious*, Technical Report SEN-R9922, CWI, Amsterdam (1999).

[4] Baltag, A. and S. Smets, *The logic of conditional actions*, in: R. van Rooij and K. Apt, editors, *New perspective on games and interaction*, Texts in logic and games **4**, Amsterdam University Press, 2008 pp. 9–31.

[5] Blackburn, P., M. de Rijke and Y. Venema, "Modal Logic," Cambridge University Press, Cambridge, Mass., 2001.

[6] Blackburn, P. and J. Seligman, *What are hybrid languages?*, in: M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, editors, *Advances in Modal Logic*, **1**, CSLI Publications, Stanford University, 1998 pp. 41–62.

[7] Girard, P., "Modal Logic for Belief and Preference Change," Ph.D. thesis, Stanford University (2008).

[8] Harel, D. and R. Sherman, *Propositional dynamic logic of flowcharts*, Information and Control **64** (1985), pp. 119–135.

[9] Kooi, B. and B. Renne, *Arrow update logic*, Review of Symbolic Logic **4** (2011), pp. 536–559.

[10] Kooi, B. and B. Renne, *Generalized arrow update logic*, in: *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK XIII (2011), pp. 205–211.

[11] Liu, F., "Changing for the Better: Preference Dynamics and Agent Diversity," Ph.D. thesis, Institute for logic, Language and Computation, Universiteit van Amsterdam, Amsterdam, The Netherlands (2008), ILLC Dissertation series DS-2008-02.

[12] Liu, F., "Reasoning about Preference Dynamics," Synthese Library **354**, Springer, 2011.

[13] Miller, J. S. and L. S. Moss, *The undecidability of iterated modal relativization*, Studia Logica **68** (2001), pp. 1–37.

[14] Passy, S. and T. Tinchev, *An essay in combinatory dynamic logic*, Information and Computation **93** (1991), pp. 263–332.

[15] Plaza, J., *Logics of public communications*, Synthese **158** (2007), pp. 165–179.

[16] Pratt, V. R., *Using graphs to understand* PDL, in: *Logic of Programs, Workshop* (1982), pp. 387–396.

[17] Seligman, J., F. Liu and P. Girard, *Logic in the community*, in: M. Banerjee and A. Seth, editors, *ICLA*, Lecture Notes in Computer Science **6521**, 2011, pp. 178–188.

[18] van Benthem, J., *Dynamic logic for belief revision*, Journal of Applied Non-classical Logic **17** (2007), pp. 129–155.

[19] Van Benthem, J., "Modal Logic for Open Minds," CSLI lecture notes, Center for the Study of Language and Information, 2010.

[20] van Benthem, J. and F. Liu, *The dynamics of preference upgrade*, Journal of Applied Non-Classical Logics **17** (2007), pp. 157–182.

[21] van Benthem, J., J. van Eijck and B. Kooi, *Logics of communication and change*, Information and computation **204** (2006), pp. 1620–1662.

[22] van Ditmarsch, H. and B. Kooi, *Semantic results for ontic and epistemic change*, in: G. Bonanno, W. van der Hoek and M. Wooldridge, editors, *Logic and the foundations of game and decision theory (LOFT 7)*, Texts in logic and games (2008), pp. 87–117.

[23] van Ditmarsch, H., W. van der Hoek and B. Kooi, "Dynamic Epistemic Logic," Berlin: Springer, 2007.

[24] van Eijck, J. and Y. Wang, *Propositional dynamic logic as a logic of belief revision*, in: W. Hodges and R. de Queiroz, editors, *WoLLIC 2008* (2008), pp. 136–148.

[25] Walther, D., "Propositional Dynamic Logic with Negation on Atomic Programs," Master's thesis, Dresden University of Technology (2004).

[26] Zhen, L. and J. Seligman, *A logical model of the dynamics of peer pressure*, Electronic Notes in Theoretical Computer Science **278** (2011), pp. 275–288.

Appendix

Proof of Lemma 1.1: We prove the two claims simultaneously by induction. Assume that $u \in \Lambda W$.

$$\begin{array}{llll} M, u \models q^\Lambda & \text{iff} & M, u \models \Lambda(q) & (\text{definition of } q^\Lambda) \\ & \text{iff} & u \in [\![\Lambda(q)]\!]^M \cap \Lambda W & (u \in \Lambda W) \\ & \text{iff} & u \in \Lambda V(q) & (\text{definition } \Lambda V(q)) \\ & \text{iff} & \Lambda M, u \models q & (\text{definition } \Lambda M) \end{array}$$

Negation and conjunction are straightforward.

$$\begin{array}{llll} M, u \models (\langle \pi \rangle \psi)^\Lambda & \text{iff} & M, u \models \langle \pi^\Lambda \rangle \psi^\Lambda & (\text{def. } (\langle \pi \rangle \psi)^\Lambda) \\ & \text{iff} & u[\![\pi^\Lambda]\!]^M v \text{ and } M, v \models \psi^\Lambda \text{ for some } v \in W & (\text{def.}) \\ & \text{iff} & u[\![\pi]\!]^{\Lambda M} v \text{ and } \Lambda M, v \models \psi \text{ for some } v \in \Lambda W & (\text{IH}) \\ & \text{iff} & \Lambda M, u \models \langle \pi \rangle \psi & (\text{def.}) \end{array}$$

$$\begin{array}{llll} u[\![s^\Lambda]\!]^M v & \text{iff} & u[\![\Lambda(s); |\Lambda|?]\!]^M v & (\text{definition of } s^\Lambda) \\ & \text{iff} & \exists w : u[\![\Lambda(s)]\!]^M w \,\&\, w[\![|\Lambda|?]\!]^M v & (\text{definition of } [\![\Lambda(s); |\Lambda|?]\!]^M) \\ & \text{iff} & u[\![\Lambda(s)]\!]^M v \,\&\, v[\![|\Lambda|?]\!]^M v & (w[\![|\Lambda|?]\!]^M v \Rightarrow w = v) \\ & \text{iff} & u[\![\Lambda(s)]\!]^M v \,\&\, v \in [\![|\Lambda|]\!]^M & (\text{definition of } [\![|\Lambda|?]\!]^M) \\ & \text{iff} & u[\![\Lambda(s)]\!]^M v \,\&\, v \in \Lambda W & (\text{definition of } \Lambda W) \\ & \text{iff} & u[\![s]\!]^{\Lambda M} v \,\&\, v \in \Lambda W & (\text{definition of } [\![\Lambda(s)]\!]^M) \end{array}$$

$$\begin{array}{llll}
u[\![(\psi?)^{\Lambda}]\!]^M v & \text{iff} & u = v \text{ and } v[\![(\psi^{\Lambda})?]\!]^M v & (u[\![(\psi?)^{\Lambda}]\!]^M v \Rightarrow u = v) \\
& \text{iff} & u = v, v \in [\![\psi^{\Lambda}]\!]^M & (\text{definition of } [\![(\psi^{\Lambda})?]\!]^M) \\
& \text{iff} & u = v, v \in \Lambda W \text{ and } v \in [\![\psi]\!]^{\Lambda M} & (\text{IH, and } u \in \Lambda W) \\
& \text{iff} & u = v, v \in \Lambda W \text{ and } v[\![\psi?]\!]^{\Lambda M} v & (\text{definition of } [\![\psi?]\!]^{\Lambda M}) \\
& \text{iff} & v \in \Lambda W \,\&\, u[\![\psi?]\!]^{\Lambda M} v & (u[\![\psi?]\!]^{\Lambda M} v \Rightarrow u = v)
\end{array}$$

The remaining cases are straightforward.

☺

Proof of Lemma 2.3: In the first direction, assume that $\langle\langle x_1, c_1\rangle, \langle x_2, c_2\rangle\rangle \in T'(\tau)$ and u$[\![\tau]\!]^M v$. Then, for some σ, $\langle x_1, x_2\rangle \in T(\sigma)$ and $\sigma^{c_1,c_2} = \tau$. Since σ is a basic program, it is either a relation symbol, E, or a test.

- If $\sigma = \psi?$ and $c_1 = c_2 = c$, then $\sigma^{c_1,c_2} = \psi^{[A,G,c]}?$. So $u[\![\psi^{[A,G,c]}?]\!]^M v$ implies that $u = v$ and $M, u \models \psi^{[A,G,c]}$. Hence, $GM, \langle u, c\rangle \models \psi$, by assumption, so $\langle u, c_1\rangle[\![\sigma]\!]^{GM}\langle v, c_2\rangle$.
- If $\sigma = r \in R, c_1 = c_2 = c$, then $\sigma^{c_1,c_2} = G_c(\sigma) = r^{G_c}$. By Lemma 1.1, $u[\![r^{G_c}]\!]^v$ implies that $u[\![\sigma]\!]^{G_c M} v$. Hence, $\langle u, c_1\rangle[\![\sigma]\!]^{GM}\langle v, c_2\rangle$, by definition.
- If $\sigma = \mathsf{E}$, then $\sigma^{c_1,c_2} = |G_{c_1}|?; \mathsf{E}; |G_{c_2}|?$. So $u[\![|G_{c_1}|?; \mathsf{E}; |G_{c_2}|?]\!]^M v$ implies that $u \in [\![|G_{c_1}|]\!]^M$ and $v \in [\![|G_{c_2}|]\!]^M$, so $\langle u, c_1\rangle \in GW$ and $\langle v, c_2\rangle \in GW$. Therefore, $\langle u, c_1\rangle[\![\mathsf{E}]\!]^{GM}\langle v, c_2\rangle$.
- If $\sigma \in S$ and $\langle c_1, c_2\rangle \in U(\sigma)$, then $\sigma^{c_1,c_2} = \top?$, so $u[\![\top?]\!]^M v$ implies that $u = v$. Thus, $\langle u, c_1\rangle[\![\sigma]\!]^{GM}\langle v, c_2\rangle$, by definition.
- In any other case, $\sigma^{c_1,c_2} = \bot?$. But this contradicts our assumption that $u[\![\sigma^{c_1,c_2}]\!]^M v$.

Therefore, $\langle x_1, x_2\rangle \in T(\sigma)$ and $\langle u, c_1\rangle[\![\sigma]\!]^{GM}\langle v, c_2\rangle$.

In the other direction, assume that $\gamma(\sigma)$ and $\sigma^{c_1,c_2} = \tau$ for some σ. But $\langle x_1, x_2\rangle \in T(\sigma)$ and $\sigma^{c_1,c_2} = \tau$ implies that $\langle\langle x_1, c_1\rangle\rangle\langle x_2, c_2\rangle \in T'(\tau)$ by definition, so we only need to show that $u[\![\sigma^{c_1,c_2}]\!]^M v$. Again, since σ is a basic program, it is either a relation symbol, E, or a test.

- If $\sigma = \psi?, c_1 = c_2 = c$, then $\sigma^{c_1,c_2} = \psi^{[A,G,c]}?$. Now, $\langle u, c\rangle[\![\psi?]\!]^{GM}\langle v, c\rangle$ implies that $u = v$ and $GM, \langle u, c\rangle \models \psi$. By assumption, $M, u \models \psi^{[A,G,c]}$, so $u[\![\sigma^{c_1,c_2}]\!]^M v$, by definition.
- If $\sigma = r \in R$, $c_1 = c_2 = c$, then $\sigma^{c_1,c_2} = G_c(\sigma)$. But $\langle u, c\rangle[\![r]\!]^{GM}\langle v, c\rangle$ implies that $\langle u, v\rangle \in V_c(r)$, by definition, so $u[\![G_c(r)]\!]^M v$. Hence, $u[\![\sigma^{c_1,c_2}]\!]^M v$.
- If $\sigma = \mathsf{E}$, then $\sigma^{c_1,c_2} = |G_{c_1}|?; \mathsf{E}; |G_{c_2}|?$. Now, $\langle u, c_1\rangle[\![\mathsf{E}]\!]^{GM}\langle v, c_2\rangle$ implies that $\langle u, c_1\rangle \in GW$ and $\langle v, c_2\rangle \in GW$, so $u \in [\![|G_{c_1}|]\!]^M$ and $v \in [\![|G_{c_2}|]\!]^M$. But $u[\![\mathsf{E}]\!]^M v$, so $u[\![|G_{c_1}|?; \mathsf{E}; |G_{c_2}|?]\!]^M v$.
- If $\sigma = s \in S$ and $\langle c_1, c_2\rangle \in U(\sigma)$, then $\sigma^{c_1,c_2} = \top?$. Thus, $\langle u, c_1\rangle[\![s]\!]^{GM}\langle v, c_2\rangle$ implies that $u = v$. But $M, u \models \top$, so $u[\![\top?]\!]^M u$. Hence, $u[\![\sigma^{c_1,c_2}]\!]^M v$.

- In any other case, $\sigma^{c_1,c_2} = \bot?$. But this contradicts our assumption that $\langle u, c_1\rangle[\![\sigma]\!]^{GM}\langle v, c_2\rangle$.

Therefore, $u[\![\sigma^{c_1,c_2}]\!]^{M}v$.

☺

Proof of Theorem 3.1: Let $\Delta M = \langle W', V''\rangle$ and $[A, G, M]M = \langle \Lambda GW, \Lambda GV\rangle$. Then,

$$\begin{array}{rcll} \Lambda GW & = & GW & (|\Lambda| = \top) \\ & = & \{\langle u, d\rangle \mid u \in [\![|G_d|]\!]\} & \text{(definition)} \\ & = & \{\langle u, d\rangle \mid u \in [\![|\mathsf{pre}(d)|]\!]\} & \text{(assumption)} \\ & = & \Delta W & \end{array}$$

$$\begin{array}{rcll} \Lambda GV(p) & = & [\![\Lambda G(p)]\!]^M \cap \Lambda GW & \text{(definition)} \\ & = & [\![p]\!]^M \cap \Delta W & (\Lambda GW = \Delta W) \\ & = & \Delta V(p) & \text{(definition)} \end{array}$$

$$\begin{array}{rcll} [\![\Lambda G_d(r_i)]\!]^{\Lambda GM} & = & [\![\Lambda(r_i)]\!]^{\Lambda GM} & (G_d(r_i) = r_i) \\ & = & [\![r_i; s_i]\!]^{GM} & \text{(assumption)} \end{array}$$

But $\langle u, d\rangle[\![r_i; s_i]\!]^{GM}\langle v, e\rangle$ iff there exists $\langle w, f\rangle \in GW$ such that $\langle u, d\rangle[\![r_i]\!]^{GM}\langle w, f\rangle$ and $\langle w, f\rangle[\![s_i]\!]^{GM}\langle v, e\rangle$, iff $d = f$ and $w = v$, by definition. Hence, $\langle u, d\rangle[\![r_i; s_i]\!]^{GM}\langle v, e\rangle$ iff $\langle u, d\rangle[\![r_i]\!]^{GM}\langle v, d\rangle$ and $\langle v, d\rangle[\![s_i]\!]^{GM}\langle v, e\rangle$ iff $u[\![G_d(r_i)]\!]^{M}v$ and $d[\![s_i]\!]^{A}e$, by definition, iff $u[\![r_i]\!]^{M}v$ and $d[\![s_i]\!]^{A}e$. Therefore, $[\![\Lambda G_d(r_i)]\!]^{\Lambda GM} = \Delta V(r)$. ☺

Proof of Theorem 3.2: The proof is the same as the previous theorem but for the propositional case:

$$\begin{array}{rcll} \Lambda G_d V(p) & = & [\![\Lambda G_d(p)]\!]^M \cap \Lambda GW & \text{(definition)} \\ & = & [\![\mathsf{sub}_d(p)]\!]^M \cap \Delta W & (\Lambda GW = \Delta W) \\ & = & \Delta V(p) & \text{(definition)} \end{array}$$

☺

Acknowledgements: We would like to thank the anonymous referees for their valuable comments. Fenrong Liu is supported by the Major Program of the National Social Science Foundation of China (NO. 11&ZD088) and the National Natural Science Foundation of China (NO.81173464). Jeremy Seligman was supported by the National Social Science Foundation Key Project of China (NO.11AZD57). He is also grateful to Auckland University for allowing a period of absence during which time the research was completed.