General Dynamic Dynamic Logic

Patrick Girard[1]
Jeremy Seligman[2]
Fenrong Liu[3]

Department of Philosophy[1,2]
The University of Auckland
Auckland, New Zealand

Department of Philosophy[3]
Tsinghua University
Beijing, China

Advances in Modal Logic, Copenhagen, 2012

General Dynamic Dynamic Logic

Patrick Girard[1]
Jeremy Seligman[2]
Fenrong Liu[3]

Department of Philosophy[1,2]
The University of Auckland
Auckland, New Zealand

Department of Philosophy[3]
Tsinghua University
Beijing, China

Advances in Modal Logic, Copenhagen, 2012

Dynamic Dynamic?

Dynamic - as in Propositional Dynamic Logic (PDL)

Dynamic - as in Dynamic Epistemic Logic (DEL)

Dynamic Dynamic?

Dynamic - as in Propositional Dynamic Logic (PDL)

Dynamic - as in Dynamic Epistemic Logic (DEL)

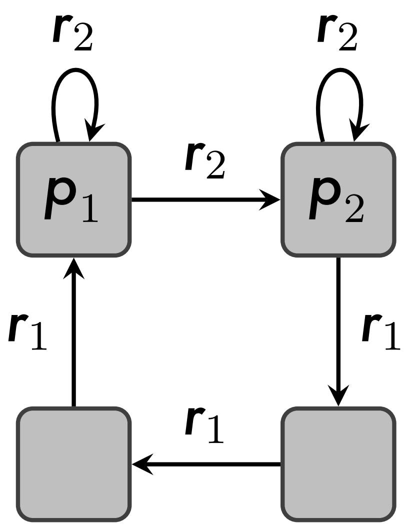
van Benthem, J., J. van Eijck and B. Kooi, Logics of communication and change, *Information and computation* 204 (2006), pp. 1620–1662.

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

Defining Relations with PDL

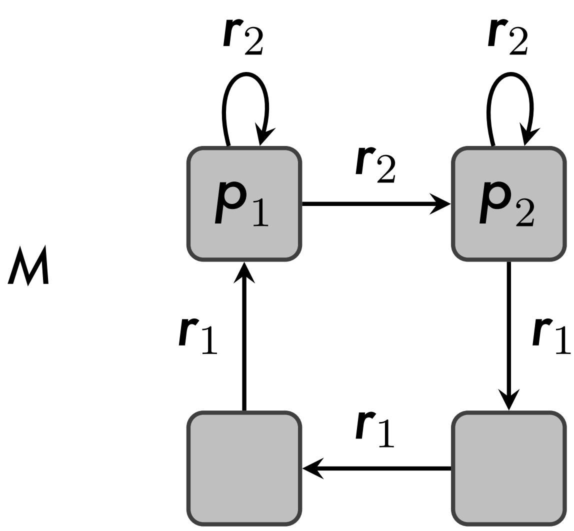
Within this model, we can define new relations using PDL.

M



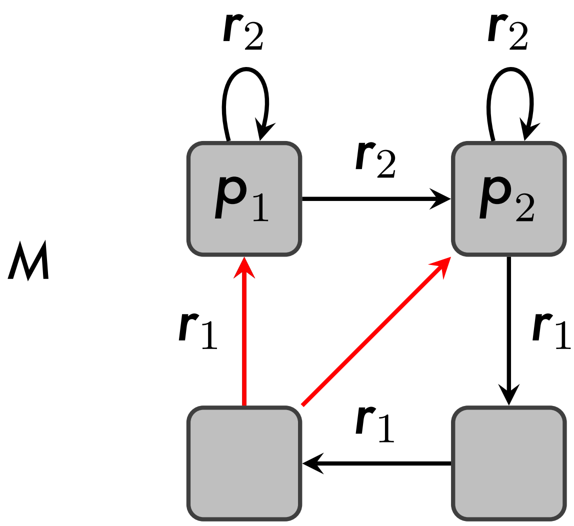
Defining Relations with PDL: Composition

Composition: $r_1 ; r_2$



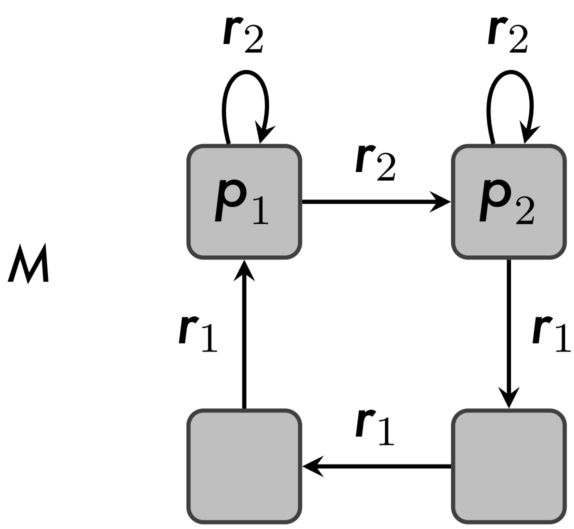
Defining Relations with PDL: Composition

Composition: $r_1 ; r_2$



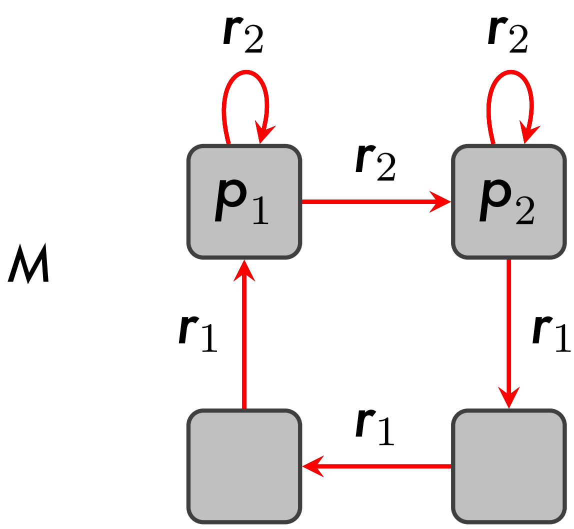
Defining Relations with PDL: Choice

Choice: $r_1 \cup r_2$



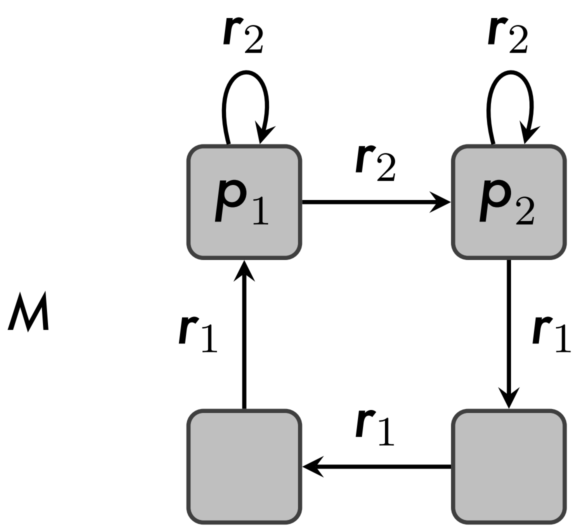
Defining Relations with PDL: Choice

Choice: $r_1 \cup r_2$



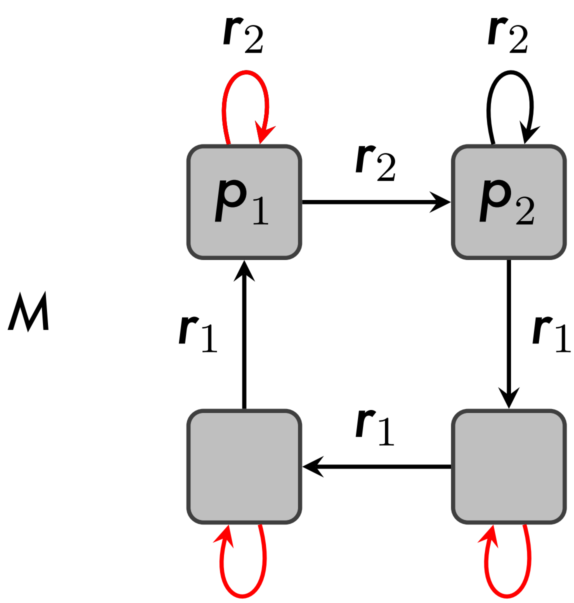
Defining Relations with PDL: Test

Test: $\neg p_2?$



Defining Relations with PDL: Test

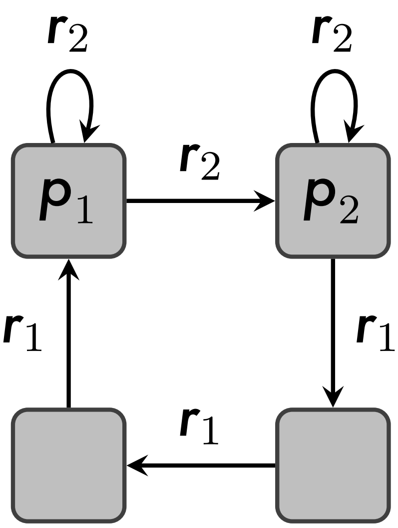
Test: $\neg p_2?$



Defining Relations with PDL: Iteration

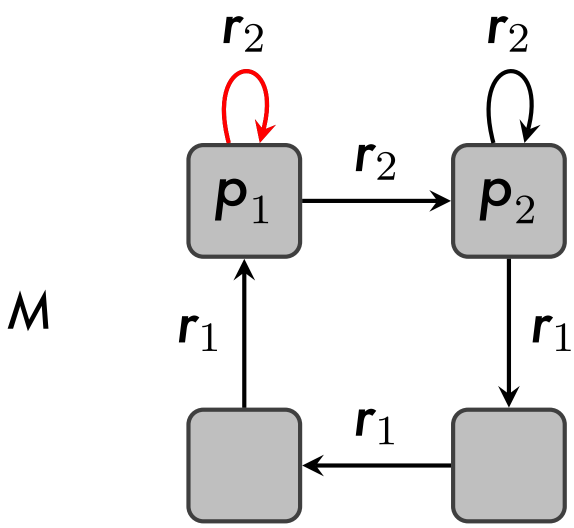
Iteration: r_1^*

M



Defining Relations with PDL: Iteration

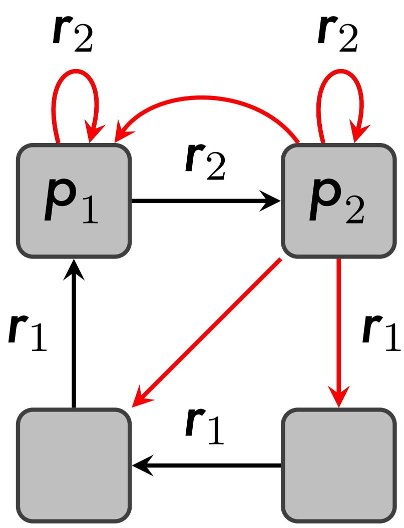
Iteration: r_1^*



Defining Relations with PDL: Iteration

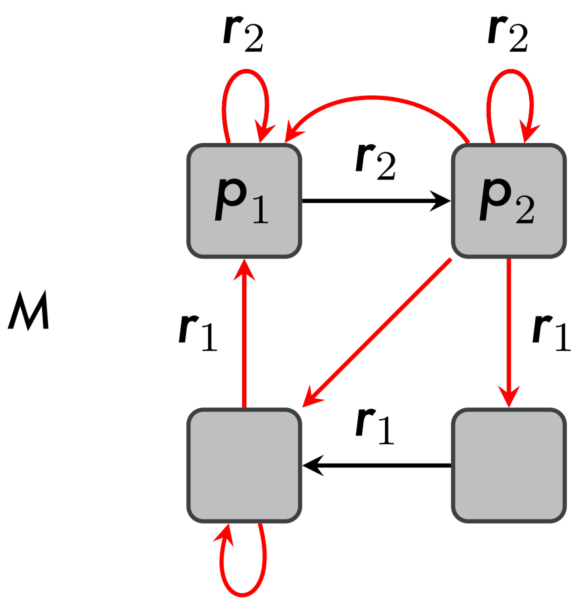
Iteration: r_1^*

M



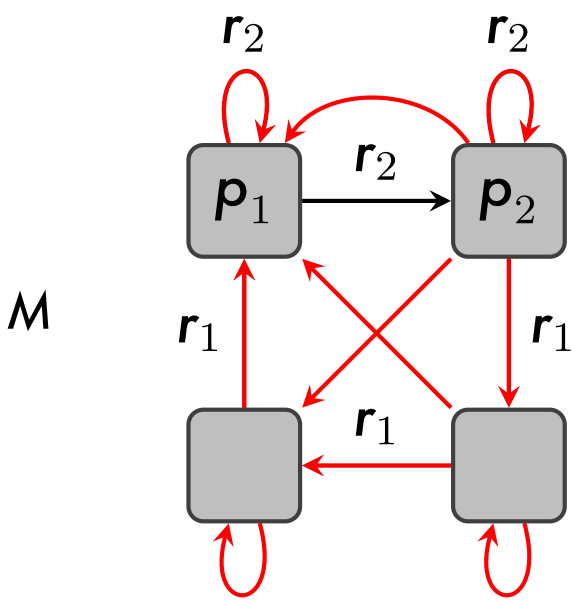
Defining Relations with PDL: Iteration

Iteration: r_1^*



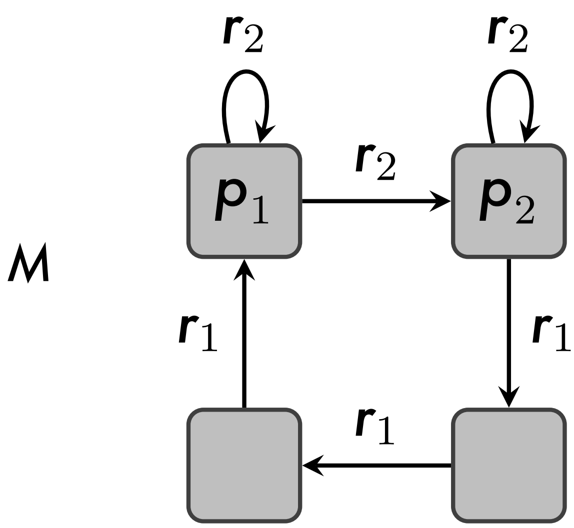
Defining Relations with PDL: Iteration

Iteration: r_1^*



Defining Relations with PDL: Programs

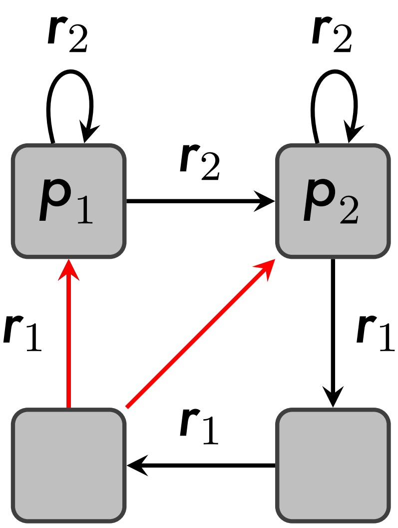
Program: $(r_1; r_2) \cup (p_1?; r_2)$



Defining Relations with PDL: Programs

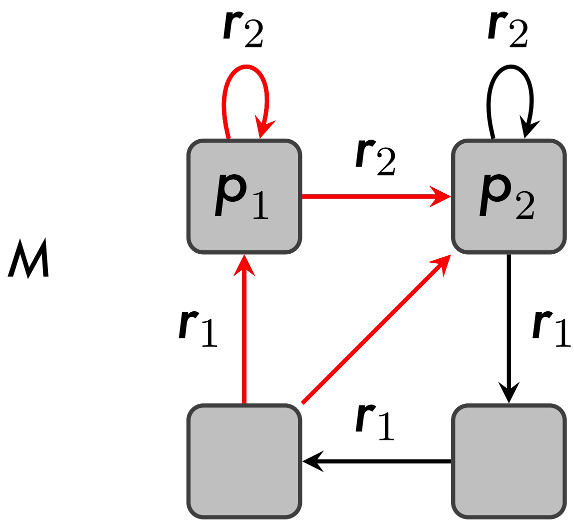
Program: $(r_1; r_2) \cup (p_1?; r_2)$

M



Defining Relations with PDL: Programs

Program: $(r_1; r_2) \cup (p_1?; r_2)$

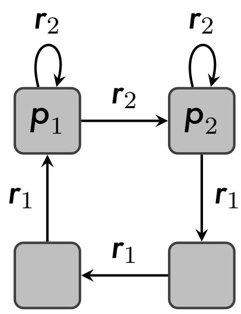
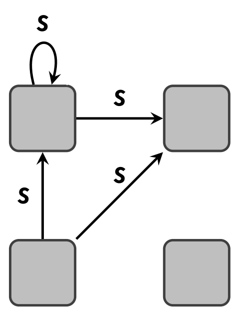


PDL Transformations

$$\Lambda(\boldsymbol{s}) \qquad = (\boldsymbol{r}_1; \boldsymbol{r}_2) \cup (\boldsymbol{p}_1?; \boldsymbol{r}_2)$$

PDL Transformations

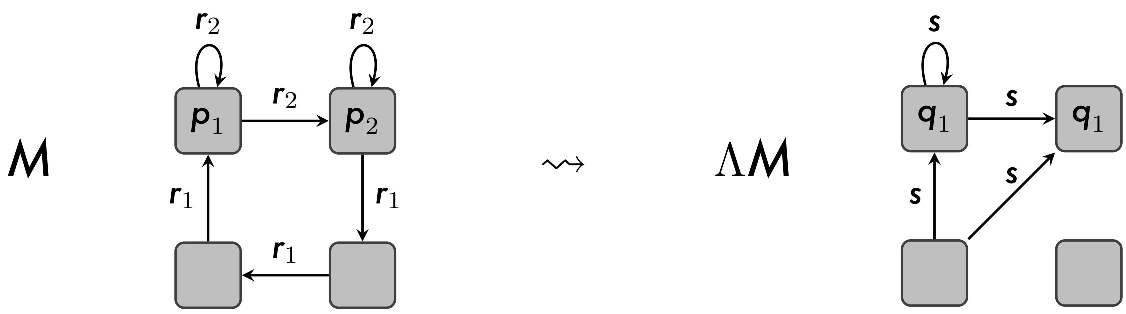
$$\Lambda(s) \quad = (r_1; r_2) \cup (p_1?; r_2)$$

M  $\rightsquigarrow$ ΛM 

PDL Transformations

$$\Lambda(q_1) = \langle r_2 \rangle \neg p_1$$

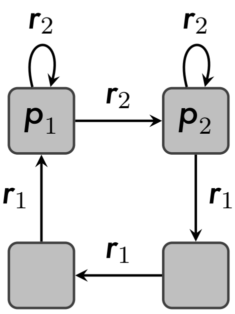
$$\Lambda(s) = (r_1; r_2) \cup (p_1?; r_2)$$



PDL Transformations

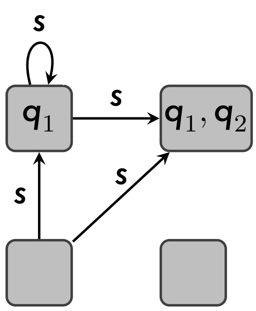
$$
\begin{aligned}
\Lambda(q_1) &= \langle r_2 \rangle \neg p_1 \\
\Lambda(q_2) &= \langle p_2?; r_1 \rangle \neg p_2 \\
\Lambda(s) &= (r_1; r_2) \cup (p_1?; r_2)
\end{aligned}
$$

M



$\rightsquigarrow$

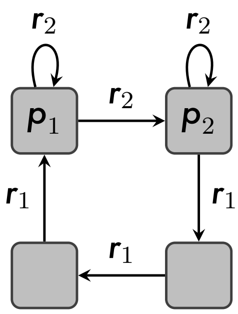
ΛM



PDL Transformations

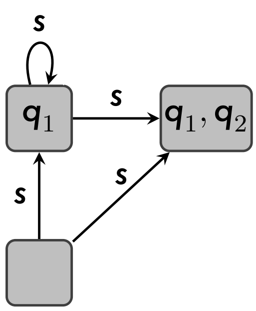
$$
\begin{aligned}
|\Lambda| &= \langle r_1 \rangle p_1 \vee \langle r_2 \rangle p_2 \\
\Lambda(q_1) &= \langle r_2 \rangle \neg p_1 \\
\Lambda(q_2) &= \langle p_2?; r_1 \rangle \neg p_2 \\
\Lambda(s) &= (r_1; r_2) \cup (p_1?; r_2)
\end{aligned}
$$

M



$\rightsquigarrow$

ΛM



PDL Transformations: Signatures

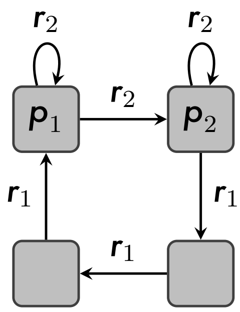
The Λ of our example changed the signature of the model.

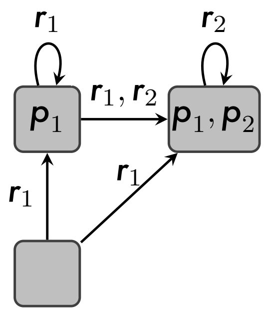
$$\begin{array}{ll} M & \text{has signature } \{p_1, p_2, r_1, r_2\} \\ \Lambda M & \text{has signature } \{q_1, q_2, s\} \end{array}$$

But we can also define signature-preserving transformations...

PDL Transformations: Signature preserving

$$
\begin{aligned}
|\Lambda'| &= \langle r_1\rangle p_1 \vee \langle r_2\rangle p_2\\
\Lambda'(p_1) &= \langle r_2\rangle \neg p_1\\
\Lambda'(p_2) &= \langle p_2?; r_1\rangle \neg p_2\\
\Lambda'(r_1) &= (r_1; r_2) \cup (p_1?; r_2)\\
\Lambda'(r_2) &= r_2
\end{aligned}
$$

M 

$\rightsquigarrow$ $\Lambda' M$ 

PDL Dynamic Operators

For $u \in |\Lambda|$,

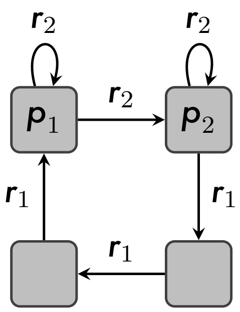
$$M, u \models [\Lambda]\varphi \quad \textbf{iff} \quad \Lambda M, u \models \varphi$$

PDL Dynamic Operators

For $u \in |\Lambda|$,

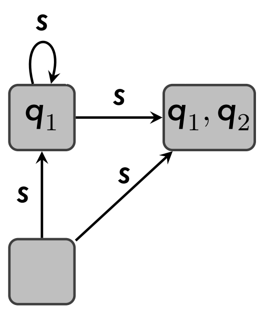
$$M, u \models [\Lambda]\varphi \quad \text{iff} \quad \Lambda M, u \models \varphi$$

M



$\rightsquigarrow$

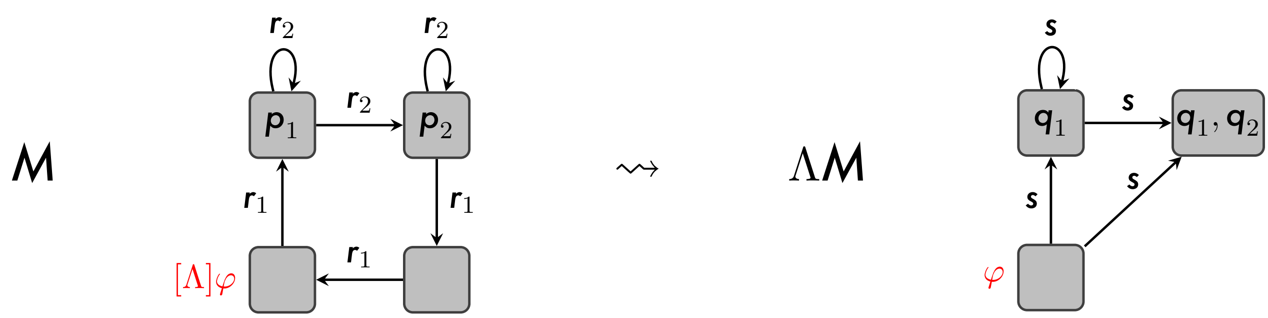
ΛM



PDL Dynamic Operators

For $u \in |\Lambda|$,

$$M, u \models [\Lambda]\varphi \quad \text{iff} \quad \Lambda M, u \models \varphi$$

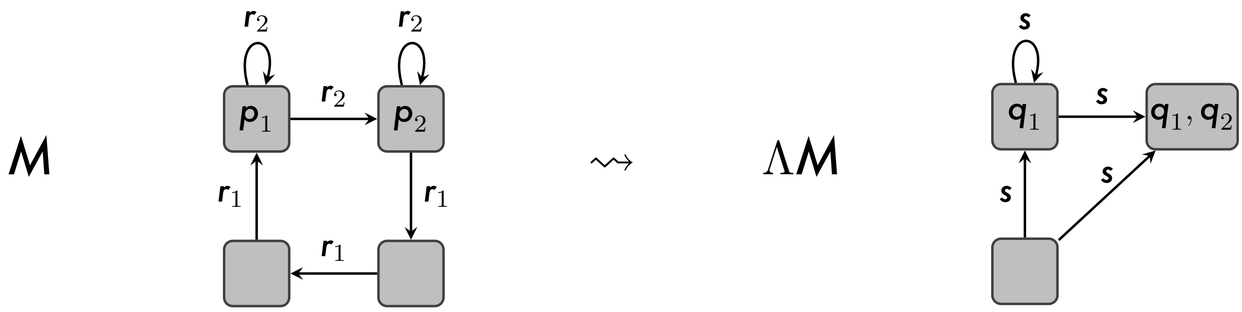


Note that φ is in the signature of ΛM.

PDL Dynamic Operators: Translation

Given Λ and φ (in sig ΛM), find φ^{Λ} (in sig M) such that

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

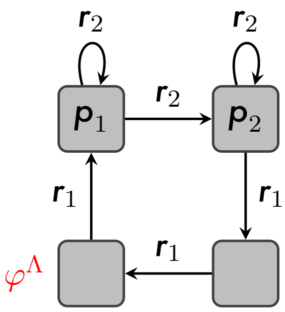


PDL Dynamic Operators: Translation

Given Λ and φ (in sig ΛM), find φ^Λ (in sig M) such that

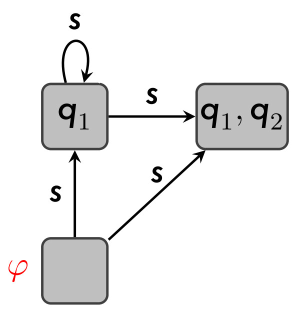
$$M, u \models \varphi^\Lambda \quad \text{iff} \quad \Lambda M, u \models \varphi$$

M



$\rightsquigarrow$

ΛM



PDL Dynamic Operators: Translation Example

We want to calculate φ^{Λ} from φ.

$$
\begin{aligned}
|\Lambda| &= \langle \mathbf{r}_1 \rangle \mathbf{p}_1 \vee \langle \mathbf{r}_2 \rangle \mathbf{p}_2 \\
\Lambda(\mathbf{q}_1) &= \langle \mathbf{r}_2 \rangle \neg \mathbf{p}_1 \\
\Lambda(\mathbf{q}_2) &= \langle \mathbf{p}_2 ?; \mathbf{r}_1 \rangle \neg \mathbf{p}_2 \\
\Lambda(\mathbf{s}) &= (\mathbf{r}_1 ; \mathbf{r}_2) \cup (\mathbf{p}_1 ?; \mathbf{r}_2)
\end{aligned}
$$

PDL Dynamic Operators: Translation Example

We want to calculate φ^Λ from φ.

$$\begin{aligned} |\Lambda| &= \langle \mathbf{r}_1\rangle \mathbf{p}_1 \vee \langle \mathbf{r}_2\rangle \mathbf{p}_2 \\ \Lambda(\mathbf{q}_1) &= \langle \mathbf{r}_2\rangle \neg \mathbf{p}_1 \\ \Lambda(\mathbf{q}_2) &= \langle \mathbf{p}_2?; \mathbf{r}_1\rangle \neg \mathbf{p}_2 \\ \Lambda(\mathbf{s}) &= (\mathbf{r}_1; \mathbf{r}_2) \cup (\mathbf{p}_1?; \mathbf{r}_2) \end{aligned}$$

$$\langle \mathbf{s}; \mathbf{s}\rangle \mathbf{q}_1 \quad \mapsto \quad \langle \Lambda(\mathbf{s}); |\Lambda|?; \Lambda(\mathbf{s}); |\Lambda|?\rangle \Lambda(\mathbf{q}_1)$$

PDL Dynamic Operators: Translation Example

We want to calculate φ^{Λ} from φ.

$$\begin{aligned}
|\Lambda| &= \langle \mathbf{r}_1 \rangle \mathbf{p}_1 \vee \langle \mathbf{r}_2 \rangle \mathbf{p}_2 \\
\Lambda(\mathbf{q}_1) &= \langle \mathbf{r}_2 \rangle \neg \mathbf{p}_1 \\
\Lambda(\mathbf{q}_2) &= \langle \mathbf{p}_2 ?; \mathbf{r}_1 \rangle \neg \mathbf{p}_2 \\
\Lambda(\mathbf{s}) &= (\mathbf{r}_1; \mathbf{r}_2) \cup (\mathbf{p}_1 ?; \mathbf{r}_2)
\end{aligned}$$

$$\langle \mathbf{s}; \mathbf{s} \rangle \mathbf{q}_1 \quad \mapsto \quad \langle \Lambda(\mathbf{s}); |\Lambda|?; \Lambda(\mathbf{s}); |\Lambda|? \rangle \Lambda(\mathbf{q}_1)$$

Expands to:

$$\langle (\mathbf{r}_1; \mathbf{r}_2) \cup (\mathbf{p}_1 ?; \mathbf{r}_2); \langle \mathbf{r}_1 \rangle \mathbf{p}_1 \vee \langle \mathbf{r}_2 \rangle \mathbf{p}_2 ?; (\mathbf{r}_1; \mathbf{r}_2) \cup (\mathbf{p}_1 ?; \mathbf{r}_2); \langle \mathbf{r}_1 \rangle \mathbf{p}_1 \vee \langle \mathbf{r}_2 \rangle \mathbf{p}_2 ? \rangle \langle \mathbf{r}_2 \rangle \neg \mathbf{p}_1$$

PDL Dynamic Operators: Axiomatisation

Extend the standard axioms and rules of PDL with the schema:

$$[\Lambda]\varphi \leftrightarrow \varphi^{\Lambda}$$

PDL Dynamic Operators: Axiomatisation

Extend the standard axioms and rules of PDL with the schema:

$$[\Lambda]\varphi \leftrightarrow \varphi^{\Lambda}$$

This is complete and decidable.

Decidable Extensions of PDL

- relational converse $\boldsymbol{r}^{\circ}$
- the universal relation $\boldsymbol{E}$
- relational complement $-\boldsymbol{r}$
- intersection $\boldsymbol{r}_1 \cap \boldsymbol{r}_2$

Walther, D. (2004) *Propositional Dynamic Logic with Negation on Atomic Programs*, Msc Thesis, Dresden University of Technology.

Lutz, C. (2005). PDL with Intersection and Converse Is Decidable. In L. Ong (Ed.), *Computer Science Logic*, Springer LNCS 3634, pp. 413–427.

PDL Dynamic Operators: Applications

Announcement of φ:	$\lvert\Lambda\rvert$	$= \varphi$
Upgrade with belief that φ:	$\Lambda(\leq)$	$= (\varphi?; \leq; \varphi?) \cup (\neg\varphi?; \leq; \neg\varphi?)$ $\cup(\neg\varphi?; \boldsymbol{E}; \varphi?)$
Upgrade with preference for φ:	$\Lambda(\preceq)$	$= (\preceq \cup(\neg\varphi?; \boldsymbol{E}; \varphi?))^*$

$\leq$ is a *plausibility* ordering;
$\preceq$ is a *preference* ordering and $\top$ is the universal relation.

PDL Dynamic Operators: Applications

Announcement of φ:	$\lvert\Lambda\rvert$	$= \varphi$
Upgrade with belief that φ:	$\Lambda(\leq)$	$= (\varphi?; \leq; \varphi?) \cup (\neg\varphi?; \leq; \neg\varphi?)$ $\cup(\neg\varphi?; \boldsymbol{E}; \varphi?)$
Upgrade with preference for φ:	$\Lambda(\preceq)$	$= (\preceq \cup(\neg\varphi?; \boldsymbol{E}; \varphi?))^*$

$\leq$ is a *plausibility* ordering;
$\preceq$ is a *preference* ordering and $\top$ is the universal relation.

C.f.

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

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

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

PDL Dynamic Operators: Example from Epistemic Doxastic Logic

$\leq_1$ and $\leq_2$ are preorders (plausibility for agents 1 and 2);
$\sim_1$ and $\sim_2$ are equivalences (indistinguishability for agents 1 and 2).

Agent i *knows* that φ in state w iff φ holds in every $v \sim_i w$.
Agent i *believes* that φ in state w iff φ holds in every $\leq_i$-maximal state $v \sim_i w$.

PDL Dynamic Operators: Example from Epistemic Doxastic Logic

$\leq_1$ and $\leq_2$ are preorders (plausibility for agents 1 and 2);
$\sim_1$ and $\sim_2$ are equivalences (indistinguishability for agents 1 and 2).

Agent $\boldsymbol{i}$ *knows* that φ in state $\boldsymbol{w}$ iff φ holds in every $\boldsymbol{v} \sim_i \boldsymbol{w}$.
Agent $\boldsymbol{i}$ *believes* that φ in state $\boldsymbol{w}$ iff φ holds in every $\leq_i$-maximal state $\boldsymbol{v} \sim_i \boldsymbol{w}$.

$$\Uparrow_1 \boldsymbol{r}(\leq_1) \quad = (\boldsymbol{r}?; \leq_1; \boldsymbol{r}?) \cup (\neg\boldsymbol{r}?; \leq_1; \neg\boldsymbol{r}?) \cup (\neg\boldsymbol{r}?; \sim_1; \boldsymbol{r}?)$$

($[\Uparrow_1 \boldsymbol{r}]\boldsymbol{M}$ is the result of agent 1 upgrading with a belief that $\boldsymbol{r}$.)

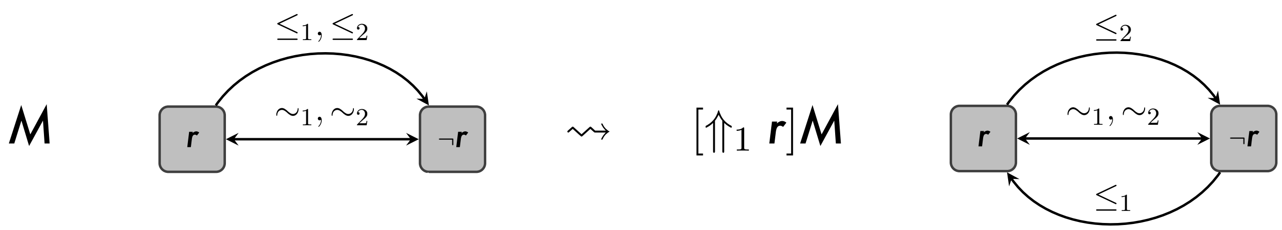
PDL Dynamic Operators: Example from Epistemic Doxastic Logic

$\leq_1$ and $\leq_2$ are preorders (plausibility for agents 1 and 2);
$\sim_1$ and $\sim_2$ are equivalences (indistinguishability for agents 1 and 2).

Agent $\boldsymbol{i}$ *knows* that φ in state $\boldsymbol{w}$ iff φ holds in every $\boldsymbol{v} \sim_i \boldsymbol{w}$.
Agent $\boldsymbol{i}$ *believes* that φ in state $\boldsymbol{w}$ iff φ holds in every $\leq_i$-maximal state $\boldsymbol{v} \sim_i \boldsymbol{w}$.

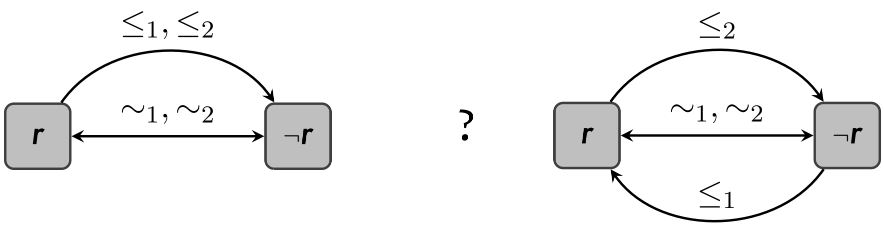
$$\Uparrow_1 \boldsymbol{r}(\leq_1) \quad = (\boldsymbol{r}?; \leq_1; \boldsymbol{r}?) \cup (\neg\boldsymbol{r}?; \leq_1; \neg\boldsymbol{r}?) \cup (\neg\boldsymbol{r}?; \sim_1; \boldsymbol{r}?)$$

($[\Uparrow_1 \boldsymbol{r}]M$ is the result of agent 1 upgrading with a belief that $\boldsymbol{r}$.)



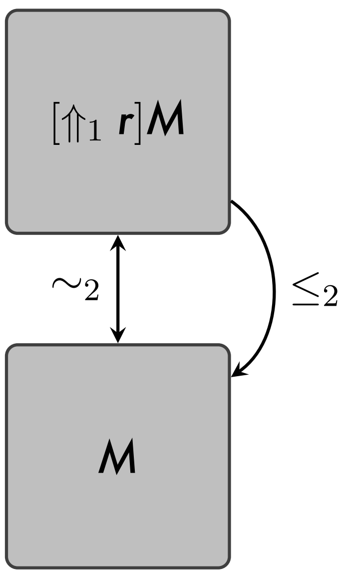
Private Belief Change: The Problem

If the change to agent 1's beliefs is *private* then agent 2 should not know whether she is in M or in $[\Uparrow_1 r]M$.



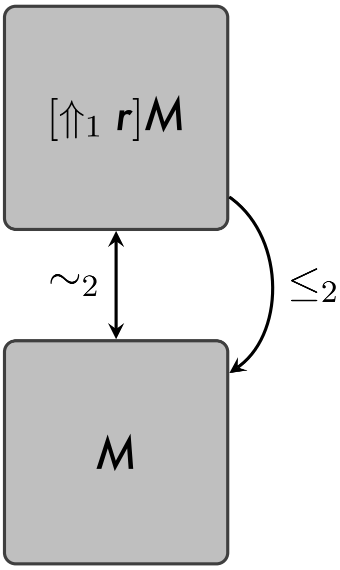
Private Belief Change: The Solution

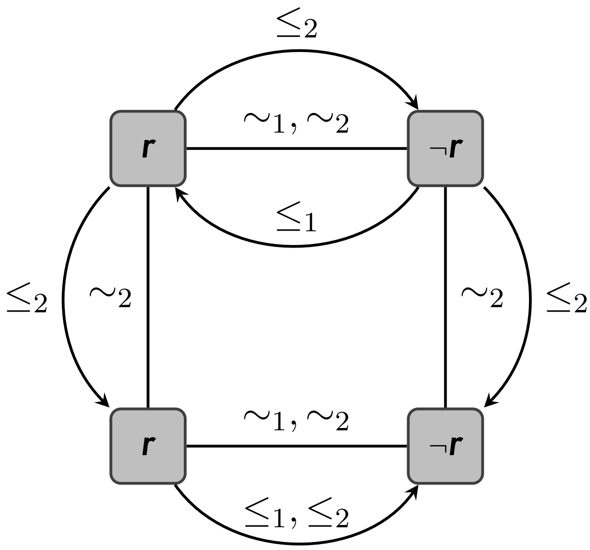
If the change to agent 1's beliefs is *private* then represent agent 2's ignorance within the model:



Private Belief Change: The Solution

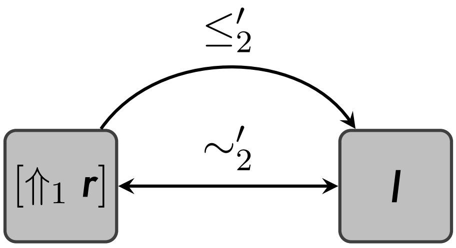
If the change to agent 1's beliefs is *private* then represent agent 2's ignorance within the model:





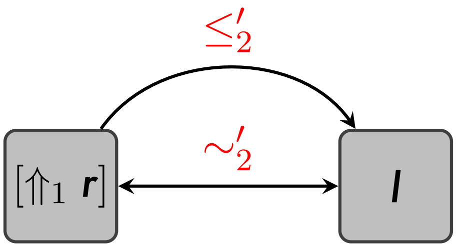
General Dynamic Operators: The Core Idea

Represent dynamic operators as (finite) models:



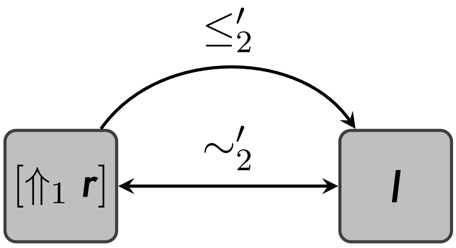
General Dynamic Operators: The Core Idea

Represent dynamic operators as (finite) models:



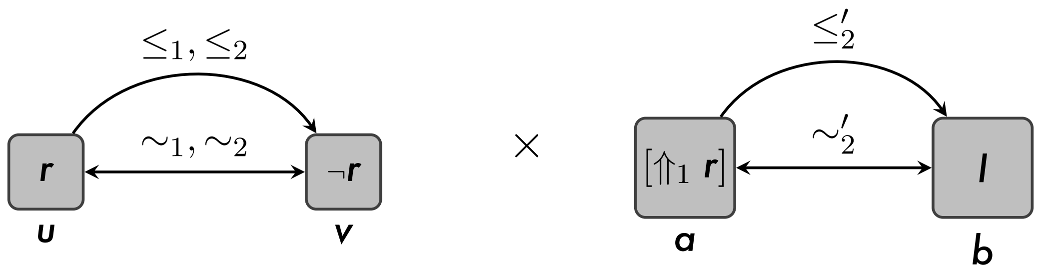
General Dynamic Operators: The Core Idea

Represent dynamic operators as (finite) models:

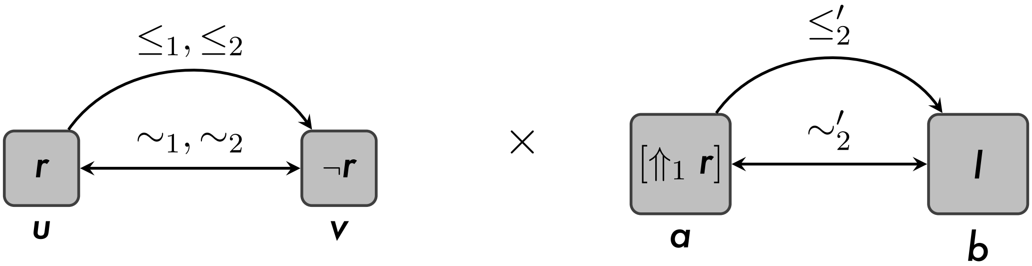


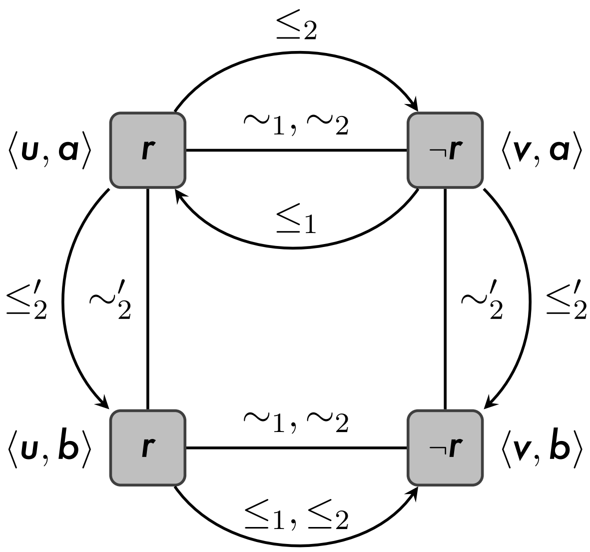
C.f. [BMS] Baltag, A., L. S. Moss and S. Solecki, *The logic of public announcements, common knowledge and private suspicious*, Technical Report SEN-R9922, CWI, Amsterdam (1999)

General Dynamic Operators: Computing the Product

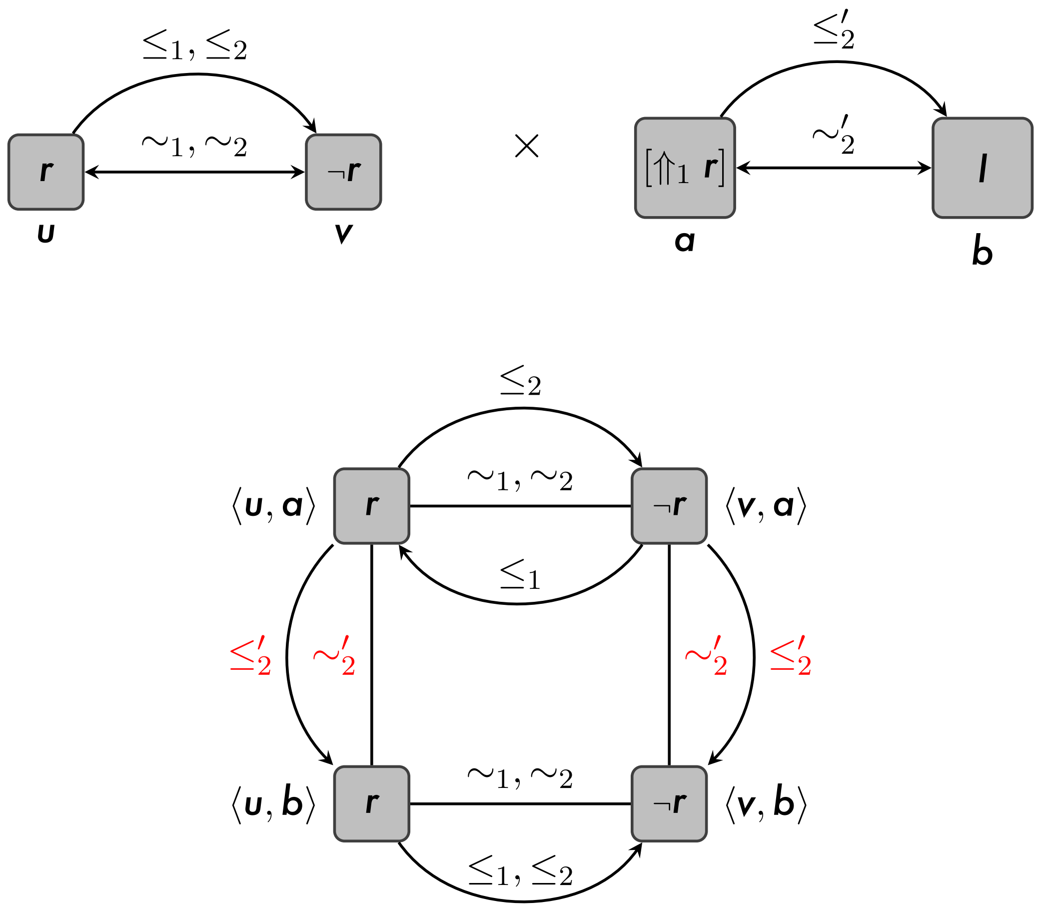


General Dynamic Operators: Computing the Product





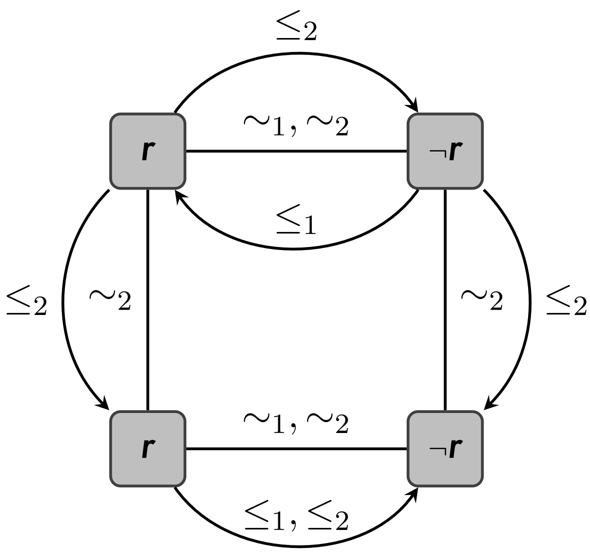
General Dynamic Operators: Computing the Product



General Dynamic Operators: Integrating

Use PDL again to compute the final model:

$$\sim_i := (\sim_i ; \sim_i')^* \quad \text{and} \quad \leq_i := (\leq_i ; \leq_i')^*$$



General Dynamic Operators: Applications

1. BMS, the original system of dynamic epistemic logic with private change
2. LCC, the 'logic of communication and change'
3. Priority Update: variant of BMS to cope with belief revision
4. Facebook Logic: reasoning about changes to social networks

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

[Priority Update] Baltag, A., and Smets, S. (2008). A qualitative theory of dynamic interactive Belief Revision. In Bonanno, G., van der Hoek, W., and Wooldridge, M. (Eds.), Logic and the Foundations of Game and Decision Theory, Texts in Logic and Games (Vol. 3). Amsterdam University Press.

General Dynamic Operators: GDDL

A GDDL *dynamic operator* $[A, G, H, a]$ on models of signature $\langle P, R \rangle$ consists of four things:

1. a finite model $A = \langle D, U \rangle$ of some finite signature $\langle Q, S \rangle$,
2. a PDL-transformation G_d from $\langle P, R \rangle$ to $\langle P, R \rangle$ for each $d \in D$,
3. a PDL-transformation H from $\langle P \cup Q, R \cup S \rangle$ to $\langle P, R \rangle$, and
4. a distinguished element $a \in D$.

($\langle Q, S \rangle$ must be distinct from $\langle P, R \rangle$.)

General Dynamic Operators: GDDL

A GDDL *dynamic operator* $[A, G, H, a]$ on models of signature $\langle P, R\rangle$ consists of four things:

1. a finite model $A = \langle D, U\rangle$ of some finite signature $\langle Q, S\rangle$,
2. a PDL-transformation G_d from $\langle P, R\rangle$ to $\langle P, R\rangle$ for each $d \in D$,
3. a PDL-transformation H from $\langle P \cup Q, R \cup S\rangle$ to $\langle P, R\rangle$, and
4. a distinguished element $a \in D$.

($\langle Q, S\rangle$ must be distinct from $\langle P, R\rangle$.)

The operator transforms a model M (of signature $\langle P, R\rangle$) to the model $[A, G, H, a]M$ (also of signature $\langle P, R\rangle$) by the method illustrated in the previous example.

General Dynamic Operators: Axiomatisation

It is possible to define a translation from φ to $\varphi^{[A,G,H,a]}$ such that

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

General Dynamic Operators: Axiomatisation

It is possible to define a translation from φ to $\varphi^{[A,G,H,a]}$ such that

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

And so we can extend the standard axioms and rules of PDL with the schema

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

to obtain a complete and decidable axiomatisation.

General Dynamic Operators: Axiomatisation

It is possible to define a translation from φ to $\varphi^{[A,G,H,a]}$ such that

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

And so we can extend the standard axioms and rules of PDL with the schema

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

to obtain a complete and decidable axiomatisation.

The proof is automata-theoretic, making use of the correspondence between PDL programs and automata, and the relative ease of product-like constructions with automata.

Update = merge + integration

The effect an change makes on agent attitudes depends on the attitudes to that change.

1. First compute the effect of transparent changes (those know to all) and merge.
2. Then integrate, according to some algorithm, depending on the application.

Update = merge + integration

The effect an change makes on agent attitudes depends on the attitudes to that change.

1. First compute the effect of transparent changes (those know to all) and merge.
2. Then integrate, according to some algorithm, depending on the application.

Different rules for integration express different theories of interaction.

Update = merge + integration

The effect an change makes on agent attitudes depends on the attitudes to that change.

1. First compute the effect of transparent changes (those know to all) and merge.
2. Then integrate, according to some algorithm, depending on the application.

Different rules for integration express different theories of interaction.

(Thanks to Johan van Benthem for this pithy slogan.)

Some directions

1. an algebraic approach: how to understand this product in the category of PDL-transformations
2. a better axiomatisation: the one we give is somewhat indirect, involving automata theory. Is there a more direct translation?
3. higher-order / lower-order interaction: weakness of will, permissible law changes, changing what is possible to do.