Reasoning about Control and Delegation

Wiebe van der Hoek[1]

[1] based on work with Michael Wooldridge and Dirk Walther

Overview

Cooperation and Propositional Control

- We need to reason about issues related to cooperation in multi-agent systems
- “Coalition”: group of agents (or “players”)
- Examples:
  - Coalitional power: what can a coalition achieve, independently of other agents in the system?
  - Coalition formation: which coalitions will form?
  - Outcomes: how will a coalition act?
- Our concern: representation of, and reasoning about, aspects of cooperation by using formal logic
- Most commonly known logics of cooperation: *Alternating-time Temporal Logic* (ATL) and *Coalition Logic* (CL)

Overview

Aims of this presentation:

- CL-PC: cooperation logic of propositional control;
- DCL-PC: extends CL-PC with delegation operators;
- add partial information to CL-PC

Cooperation Logics

- Two important developments in logical foundations of MAS:
  - Pauly's Coalition Logic (2000);
  - temporal cooperation logic – ATL of Alur, Henzinger, and Kupferman (1997-2002).
- Basic idea in both systems: cooperation modality:

$$\langle\langle C \rangle\rangle \varphi$$

meaning

coalition C can cooperate to ensure that φ

Semantics – the intuition

- $\langle\langle C\rangle\rangle\varphi$ means C have winning strategy for φ.
- Semantics:

$$\langle\langle C\rangle\rangle\varphi \qquad \text{iff} \qquad \exists\sigma_C : \forall\sigma_{\bar{C}} : \mathit{out}(\sigma_C, \sigma_{\bar{C}}) \models \varphi$$

- Notice $\exists\forall$ pattern of quantifiers...
  $\langle\langle\cdot\rangle\rangle$ is not a conventional modality
- This is α-ability
- Counter-intuitive readings of $\langle\langle\cdot\rangle\rangle$ when negated.

CL-PC

- CL and ATL give no answer to the question of where an agent's powers come from.
- In CL-PC, we gave one answer to this question.
- Assume every agent *i* has unique, complete control over a set $\mathbb{A}_i$ of propositional (Boolean) variables.
- Choices/powers available to agents correspond to all valuations possible to these variables.
- Basic language construct

$$\Diamond_C \varphi$$

means there exists a choice available to C s.t. (if nothing else changes) then φ.

Semantics

$$M = \langle Ag, \mathbb{A}, \mathbb{A}_1, \ldots, \mathbb{A}_n, \theta \rangle$$

where:

- $Ag = \{1, \ldots, n\}$ is a finite, non-empty set of *agents*;
- $\mathbb{A} = \{p, q, \ldots\}$ is a finite, non-empty set of *propositional variables*;
- $\mathbb{A}_1 \ldots, \mathbb{A}_n$ is a partition of $\mathbb{A}$ among the members of Ag, with the intended interpretation that $\mathbb{A}_i$ is the subset of $\mathbb{A}$ representing those variables under the control of agent $i \in Ag$; and finally,
- $\theta : \mathbb{A} \to \{\mathbf{true}, \mathbf{false}\}$ is a propositional valuation function, which determines the initial truth value of every propositional variable.

Truth

$M = \langle Ag, \mathbb{A}, \mathbb{A}_1, \ldots, \mathbb{A}_n, \theta \rangle$ is given.

$$M \models^{\mathrm{d}} \Diamond_C \varphi \text{ iff } \exists \theta_C (\theta_C = \theta \mod C \text{ and } M \dagger \theta_C \models^{\mathrm{d}} \varphi)$$

Truth

$M = \langle Ag, \mathbb{A}, \mathbb{A}_1, \ldots, \mathbb{A}_n, \theta \rangle$ is given.

$$M \models^{\mathrm{d}} \Diamond_C \varphi \text{ iff } \exists \theta_C (\theta_C = \theta \mod C \text{ and } M \dagger \theta_C \models^{\mathrm{d}} \varphi)$$

$$\Box_C \varphi \triangleq \neg \Diamond_C \neg \varphi.$$

Example

$$M = \langle \{1, 2\}, \mathbb{A}, \mathbb{A}_1, \mathbb{A}_2, \theta \rangle$$

and $\mathbb{A} = \{p, q\}$, $\mathbb{A}_1 = \emptyset$, $\mathbb{A}_2 = \{p, q\}$ and $\theta(x) = \textbf{true}$ for all x

Example

$$M = \langle \{1, 2\}, \mathbb{A}, \mathbb{A}_1, \mathbb{A}_2, \theta \rangle$$

and $\mathbb{A} = \{p, q\}$, $\mathbb{A}_1 = \emptyset$, $\mathbb{A}_2 = \{p, q\}$ and $\theta(x) =$ **true** for all x

There is no 1-valuation θ_1 such that $M \dagger \theta_1 \models^{\mathrm{d}} \neg p$.
That is, $M \models^{\mathrm{d}} \Box_1 p$

Example

$$M = \langle \{1, 2\}, \mathbb{A}, \mathbb{A}_1, \mathbb{A}_2, \theta \rangle$$

and $\mathbb{A} = \{p, q\}$, $\mathbb{A}_1 = \emptyset$, $\mathbb{A}_2 = \{p, q\}$ and $\theta(x) =$ **true** for all x

Similarly, agent 1 cannot avoid $p \wedge q$, i.e., $M \models^{\mathrm{d}} \Box_1(p \wedge q)$

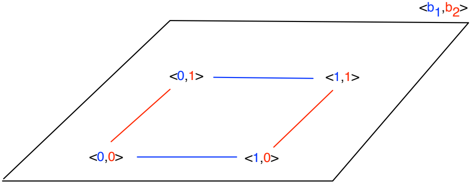
Example

$$M = \langle \{1, 2\}, \mathbb{A}, \mathbb{A}_1, \mathbb{A}_2, \theta \rangle$$

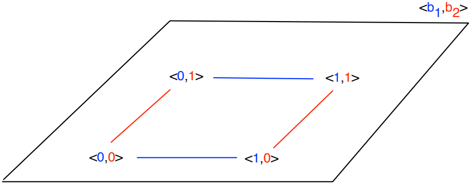
and $\mathbb{A} = \{p, q\}$, $\mathbb{A}_1 = \emptyset$, $\mathbb{A}_2 = \{p, q\}$ and $\theta(x) = \textbf{true}$ for all x

Similarly, agent 1 cannot avoid $p \wedge q$, i.e., $M \models^{\mathrm{d}} \Box_1(p \wedge q)$
But this does not mean $p \wedge q$ is inevitable: it depends on the choice that 2 makes. Since 2 controls both p and q, we have $M \models^{\mathrm{d}} \Diamond_2 \neg p$, $M \models^{\mathrm{d}} \Diamond_2 \neg q$, and $M \models^{\mathrm{d}} \Diamond_2 \neg (p \vee q)$

Kripke Models

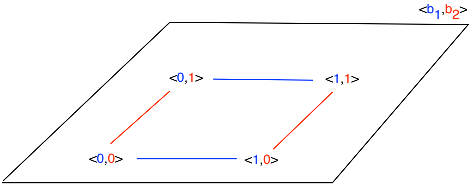


Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^k (\neg b_1 \wedge \neg b_2)$)

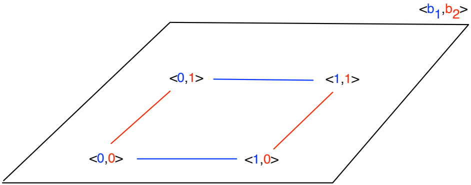
Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^k (\neg b_1 \wedge \neg b_2)$)

- no agent can force them both going to Bach $(\neg\Diamond_1(b_1 \wedge b_2) \wedge \neg\Diamond_2(b_1 \wedge b_2))$

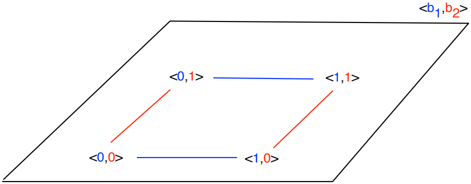
Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^{k} (\neg b_1 \wedge \neg b_2)$)

- whereas in full cooperation they can share a Bach evening ($\Diamond_{1,2}(b_1 \wedge b_2)$).

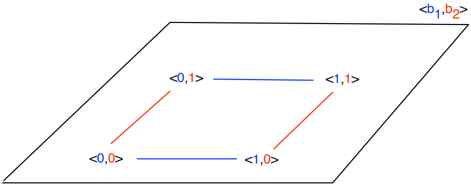
Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^{k} (\neg b_1 \wedge \neg b_2)$)

- On a global level, neither agent can establish an evening out together ($\mathcal{K}_I \not\models^{k} \Diamond_i(b_1 \wedge b_2)$ and $\mathcal{K}_I \not\models^{k} \Diamond_i(\neg b_1 \wedge \neg b_2)$, $i \leq 2$),

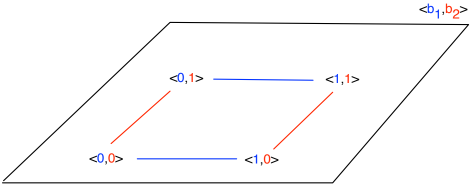
Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^k (\neg b_1 \wedge \neg b_2)$)

- but fortunately, they can cooperate to have an evening out together $\mathcal{K}_I \models^k \Diamond_{1,2}(b_1 \wedge b_2) \wedge \Diamond_{1,2}(\neg b_1 \wedge \neg b_2)$,

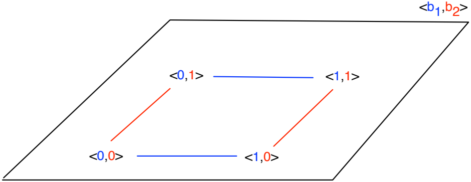
Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^k (\neg b_1 \wedge \neg b_2)$)

- but this still involves a choice $\mathcal{K}_I \models^k \neg\Box_{1,2}(b_1 \vee b_2) \wedge \neg\Box_{1,2}(\neg b_1 \wedge \neg b_2)$.

Kripke Models



Let b_1 be the atom denoting that agent 1 chooses for Bach, and similarly for b_2 and agent 2. Assume that initially no agent is going to Bach (i.e., w and Θ are such that $\mathcal{K}, w \models^{k} (\neg b_1 \wedge \neg b_2)$)

- The two semantics are equivalent

Axiomatisation for CL-PC

Prop	φ,	where φ is any propositional tautology
K(i)	$\Box_i(\varphi \to \psi) \to (\Box_i\varphi \to \Box_i\psi)$	
T(i)	$\Box_i\varphi \to \varphi$	
B(i)	$\varphi \to \Box_i\Diamond_i\varphi$	
empty	$\Box_\emptyset\varphi \leftrightarrow \varphi$	
at_least	$\ell(p) \to \bigvee_{i \in Ag} \Diamond_i \neg\ell(p)$	
at_most	$\ell(p) \to (\Diamond_i\neg\ell(p) \to \Box_j\ell(p))$	where $i \neq j$
eff(i)	$(\psi \wedge \ell(p)) \to \Diamond_i(\psi \wedge \neg\ell(p))$	$\left\{ \begin{array}{l} p \in \mathbb{A}_i, \\ p \notin \mathbb{A}(\psi), \text{ and} \\ \psi \text{ is objective} \end{array} \right.$
no_eff(i)	$\Diamond_i\ell(p) \to \Box_i\ell(p)$	where $p \notin \mathbb{A}_i$
Comp	$\Box_{C_1}\Box_{C_2}\varphi \leftrightarrow \Box_{C_1 \cup C_2}\varphi$	

CL-PC Results

- $\Diamond_C \varphi$ is a “true modal diamond”.

CL-PC Results

- $\Diamond_C \varphi$ is a "true modal diamond".
- Complete axiomatization for CL-PC
  Based on a normal form

CL-PC Results

- $\Diamond_C \varphi$ is a "true modal diamond".
- Complete axiomatization for CL-PC
- Model checking + satisfiability are PSPACE-complete.

CL-PC Results

- $\Diamond_C \varphi$ is a "true modal diamond".
- Complete axiomatization for CL-PC
- Model checking + satisfiability are PSPACE-complete.
- Characterisation of control:

Control

Characterisation of control:

$$controls(C, \varphi) \mathrel{\hat{=}} \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$

Control

Characterisation of control:

$$controls(C, \varphi) \hat{=} \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$

Control

Characterisation of control:

$$controls(C, \varphi) \mathrel{\hat{=}} \Diamond_C \varphi \wedge \Diamond_C \neg \varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$
3. $controls(i, p) \wedge controls(i, q) \rightarrow controls(i, controls(p \wedge q))$

Control

Characterisation of control:

$$controls(C, \varphi) \triangleq \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$
3. $controls(i, p) \wedge controls(i, q) \rightarrow controls(i, controls(p \wedge q))$
4. $$controls(i, p \wedge q) \quad \leftrightarrow$$

Control

Characterisation of control:

$$controls(C, \varphi) \triangleq \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$
3. $controls(i, p) \wedge controls(i, q) \rightarrow controls(i, controls(p \wedge q))$
4. $$\begin{array}{rcl} controls(i, p \wedge q) & \leftrightarrow & \\ p \wedge q & \wedge & (controls(i, p) \vee controls(i, q)) \quad \vee \end{array}$$

Control

Characterisation of control:

$$controls(C, \varphi) \triangleq \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$
3. $controls(i, p) \wedge controls(i, q) \rightarrow controls(i, controls(p \wedge q))$
4. $$\begin{array}{rcll} controls(i, p \wedge q) & \leftrightarrow & & \\ p \wedge q & \wedge & (controls(i, p) \vee controls(i, q)) & \vee \\ p \wedge \neg q & \wedge & controls(i, q)) & \vee \end{array}$$

Control

Characterisation of control:

$$controls(C, \varphi) \mathrel{\hat{=}} \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i, p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j, p))$
3. $controls(i, p) \wedge controls(i, q) \rightarrow controls(i, controls(p \wedge q))$
4. $$\begin{array}{rcll} controls(i, p \wedge q) & \leftrightarrow & & \\ p \wedge q & \wedge & (controls(i, p) \vee controls(i, q)) & \vee \\ p \wedge \neg q & \wedge & controls(i, q)) & \vee \\ \neg p \wedge q & \wedge & controls(i, p) & \vee \end{array}$$

Control

Characterisation of control:

$$controls(C,\varphi) \triangleq \Diamond_C \varphi \wedge \Diamond_C \neg\varphi$$

1. $controls(i,p)$ iff $p \in \mathbb{A}_i$
2. $\neg controls(i, controls(j,p))$
3. $controls(i,p) \wedge controls(i,q) \rightarrow controls(i, controls(p \wedge q))$
4. $$\begin{array}{ccl} controls(i, p \wedge q) & \leftrightarrow & \\ p \wedge q & \wedge & (controls(i,p) \vee controls(i,q)) \quad \vee \\ p \wedge \neg q & \wedge & controls(i,q)) \quad \vee \\ \neg p \wedge q & \wedge & controls(i,p) \quad \vee \\ \neg p \wedge \neg q & \wedge & (controls(i,p) \wedge controls(i,q)) \end{array}$$

Ability

- Definition of α ability:

$$\langle\langle C \rangle\rangle_\alpha \varphi \quad \leftrightarrow \quad \Diamond_C \Box_{\bar{C}} \varphi$$

Ability

- Definition of α ability:

$$\langle\langle C \rangle\rangle_\alpha \varphi \quad \leftrightarrow \quad \Diamond_C \Box_{\bar{C}} \varphi$$

- Definition of β ability:

$$\langle\langle C \rangle\rangle_\beta \varphi \quad \leftrightarrow \quad \Box_{\bar{C}} \Diamond_C \varphi$$

DCL-PC

- In CL-PC, allocation of variables to agents is fixed.
- In DCL-PC, we extend with delegation programs for transferring propositions around.
  (Builds on propositional dynamic logic.)
- An atomic delegation program has form:

$$i \rightsquigarrow_p j$$

  meaning

  i gives control of proposition p to j.
- These are combined with ;, ?, $\cup$, * as in PDL

Examples

- $\neg\varphi \wedge [i \leadsto_p j]\Diamond_j\varphi$
  φ is not currently true, but if *i* gives *j* control of *p*, then *j* will have the ability to achieve φ
- $\neg\Diamond_j p \wedge [i \leadsto_p j]\Diamond_j p$
  j does not currently have ability to make φ true, but if *i* gives *j* control of *p*, then *j* will have the ability to achieve φ
- $[(i \leadsto_p j) \cup (i \leadsto_q j)]\Diamond_j\varphi$
  if agent *i* gives either *p* or *q* to *j*, then *j* will be able to achieve φ.
- A bigger example:

$$\langle \textbf{while } \neg\Diamond_j\varphi \textbf{ do } \bigcup_{p\in\mathbb{A}_i} i \leadsto_p j\rangle\top$$

Examples

- $\neg\varphi \wedge [i \rightsquigarrow_p j]\Diamond_j\varpi$
  φ is not currently true, but if i gives j control of p, then j will have the ability to achieve φ
- $\neg\Diamond_j p \wedge [i \rightsquigarrow_p j]\Diamond_j p$
  j does not currently have ability to make φ true, but if i gives j control of p, then j will have the ability to achieve φ
- $[(i \rightsquigarrow_p j) \cup (i \rightsquigarrow_q j)]\Diamond_j\varphi$
  if agent i gives either p or q to j, then j will be able to achieve φ.
- A bigger example:

$$\langle \textbf{while } \neg\Diamond_j\varphi \textbf{ do } \bigcup_{p\in\mathbb{A}_i} i \rightsquigarrow_p j\rangle\top$$

Examples

- $\neg\varphi \wedge [i \rightsquigarrow_p j]\Diamond_j \varpi$
  φ is not currently true, but if *i* gives *j* control of *p*, then *j* will have the ability to achieve φ
- $\neg\Diamond_j p \wedge [i \rightsquigarrow_p j]\Diamond_j p$
  j does not currently have ability to make φ true, but if *i* gives *j* control of *p*, then *j* will have the ability to achieve φ
- $[(i \rightsquigarrow_p j) \cup (i \rightsquigarrow_q j)]\Diamond_j \varphi$
  if agent *i* gives either *p* or *q* to *j*, then *j* will be able to achieve φ.
- A bigger example:

$$\langle \textbf{while } \neg\Diamond_j \varphi \textbf{ do } \bigcup_{p \in \mathbb{A}_i} i \rightsquigarrow_p j \rangle \top$$

Examples

- $\neg\varphi \wedge [i \rightsquigarrow_p j]\Diamond_j\varpi$
  φ is not currently true, but if i gives j control of p, then j will have the ability to achieve φ
- $\neg\Diamond_j p \wedge [i \rightsquigarrow_p j]\Diamond_j p$
  j does not currently have ability to make φ true, but if i gives j control of p, then j will have the ability to achieve φ
- $[(i \rightsquigarrow_p j) \cup (i \rightsquigarrow_q j)]\Diamond_j\varphi$
  if agent i gives either p or q to j, then j will be able to achieve φ.
- A bigger example:

$$\langle \textbf{while } \neg\Diamond_j\varphi \textbf{ do } \bigcup_{p\in\mathbb{A}_i} i \rightsquigarrow_p j\rangle\top$$

Example

Two clients c_1 and c_2, and a server s. The server always has control over one of the propositional variables p_1 and p_2, in particular s wants to guarantee that those variables are never true simultaneously. At the same time, c_1 and c_2 want to ensure that at least one of the variables p_i ($i = 1, 2$) is true, where variable p_i belongs to client c_i.

Example

Two clients c_1 and c_2, and a server s.

$$Inv \triangleq \bigvee_{i=1,2} controls(s, p_i) \ \wedge \ \bigvee_{i=1,2} controls(c_i, p_i)$$

$$\begin{aligned}\beta \triangleq \quad & \Big((controls(s, p_1)? \ ; \ s \leadsto_{p_1} c_1 \ ; \ c_2 \leadsto_{p_2} s) \\ & \cup (controls(s, p_2)? \ ; \ s \leadsto_{p_2} c_2 \ ; \ c_1 \leadsto_{p_1} s)\Big)^*.\end{aligned}$$

This says that an arbitrary number of times one variable p_i is passed from the server to the client c_i, and another variable p_j ($i \neq j$) from the client c_j to the server.

Example

Two clients c_1 and c_2, and a server s.

$$Inv \triangleq \bigvee_{i=1,2} controls(s, p_i) \wedge \bigvee_{i=1,2} controls(c_i, p_i)$$

Using Inv and β, we can describe the whole scenario as follows:

$$Inv \rightarrow [\beta] Inv$$
$$Inv \rightarrow [\beta]\Big(\Diamond_s \neg(p_1 \wedge p_2) \wedge \Diamond_{\{c_1,c_2\}}(p_1 \vee p_2)\Big)$$

Syntax

DCL-PC formulas:

DCL	$::=$	$\top$	/* truth constant */
	$\mid$	p	/* propositional variables */
	$\mid$	$\neg$DCL	/* negation */
	$\mid$	DCL $\vee$ DCL	/* disjunction */
	$\mid$	$\Diamond_C$DCL	/* contingent cooperative ability */
	$\mid$	$\langle\delta\rangle$DCL	/* existential dynamic operator */

Syntax

Delegation programs:

$$
\begin{array}{lcll}
\delta & ::= & i \leadsto_p j & \text{/* } i \text{ gives } p \text{ to } j \text{ */} \\
& | & \delta ; \delta & \text{/* sequential composition */} \\
& | & \delta \cup \delta & \text{/* non-deterministic choice */} \\
& | & \delta^* & \text{/* iteration */} \\
& | & \textsc{dcl}? & \text{/* test */}
\end{array}
$$

Models

A model for DCL-PC is a structure:

$$M = \langle Ag, \mathbb{A}, \xi_0, \theta \rangle$$

- $Ag = \{1, \ldots, n\} \neq \emptyset$ is a finite set of agents;
- $\mathbb{A} = \{p, \ldots, r\} \neq \emptyset$ is a finite set of propositional variables;
- $\xi_0 = \langle \mathbb{A}_1 \ldots, \mathbb{A}_n \rangle$ is the initial allocation of $\mathbb{A}$ to Ag;
- $\theta : \mathbb{A} \to \{\textbf{true}, \textbf{false}\}$ is a propositional valuation function, which determines the initial truth value of every propositional variable.

Semantics

$M \models \top$

$M \models p$ iff $\theta(p) = \mathbf{true}$ (where $p \in \mathbb{A}$);

$M \models \neg\varphi$ iff $M \not\models \varphi$;

$M \models \varphi \vee \psi$ iff $M \models \varphi$ or $M \models \psi$;

$M \models \Diamond_C \varphi$ iff there exists a valuation θ_C for C's propositions such that $M \dagger \theta_C \models \varphi$.

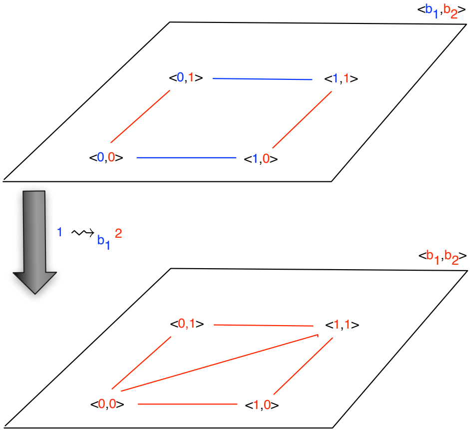
$M \models \langle\delta\rangle\varphi$ iff there exists a model M' such that $(M, M') \in R_\delta$ and $M' \models \varphi$.

We need to define R_δ.

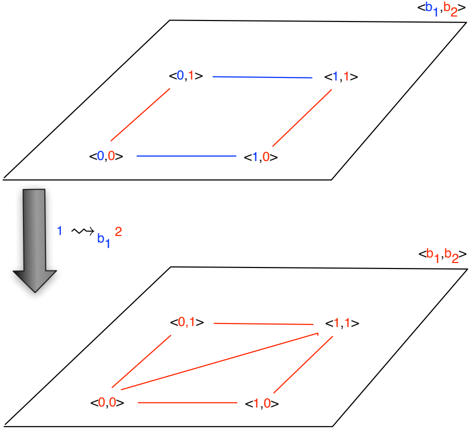
Program Relations

- For each program δ, we need to define a binary relation R_δ over models for the logic, so $(M, M') \in R_\delta$ iff M' could result from executing δ from M.
- For composite programs ($*$, $;$, $?$, $\cup$) definition is as in PDL.
- For atomic delegation programs $i \leadsto_p j$: $(M, M') \in R_{i \leadsto_p j}$ iff either $i = j$, $p \in \mathbb{A}_i$ and $M = M'$, or else:
  1. $p \in \mathbb{A}_i$
  2. $\mathbb{A}'_i = \mathbb{A}_i \setminus \{p\}$
  3. $\mathbb{A}'_j = \mathbb{A}_j \cup \{p\}$
  4. all other components of M' are as in M.

Kripke Models



Kripke Models



The two semantics are equivalent

Axioms

Propositional Control Component

$CL-PC$ $\quad \varphi$ where φ is a CL-PC tautology

Dynamic Component

$K(\delta)$	$[\delta](\varphi \rightarrow \psi) \rightarrow ([\delta]\varphi \rightarrow [\delta]\psi)$
$union(\delta)$	$[\delta \cup \delta']\varphi \leftrightarrow ([\delta]\varphi \wedge [\delta']\varphi)$
$comp(\delta)$	$[\delta; \delta']\varphi \leftrightarrow ([\delta][\delta']\varphi)$
$test(\delta)$	$[\varphi?]\psi \leftrightarrow (\varphi \rightarrow \psi)$
$mix(\delta)$	$(\varphi \wedge [\delta][\delta*]\varphi) \leftrightarrow ([\delta*]\varphi)$
$ind(\delta)$	$(\varphi \wedge [\delta*](\varphi \rightarrow [\delta]\varphi)) \rightarrow ([\delta*]\varphi)$

Delegation and Control Axioms

atomic permanence	$\langle\delta\rangle\top \rightarrow (p \leftrightarrow [\delta]p)$
persistence$_1$(control)	$(controls(i,p) \rightarrow \Box_j controls(i,p))$
persistence$_2$(control)	$controls(i,p) \rightarrow [j \rightsquigarrow_q h]controls(i,p)$ $i \neq j$ or $p \neq q$
precondition(delegation)	$\langle i \rightsquigarrow_p j\rangle\top \rightarrow controls(i,p)$
delegation	$controls(i,p) \rightarrow \langle i \rightsquigarrow_p j\rangle controls(j,p)$
func	$controls(i,p) \rightarrow (\langle i \rightsquigarrow_p j\rangle\varphi \leftrightarrow [i \rightsquigarrow_p j]\varphi)$

<u>Rules of Inference</u>

ModusPonens	$\vdash \varphi, \vdash (\varphi \rightarrow \psi) \Rightarrow \vdash \psi$	
Necessitation	$\vdash \varphi \Rightarrow \vdash \Box\varphi$	$\Box = [\delta], [i \rightsquigarrow_p j]$, or $\Box_i$

Some Derived Theorems

inverse :
$controls(i, p) \rightarrow (\varphi \leftrightarrow [i \rightsquigarrow_p j; j \rightsquigarrow_p i]\varphi)$

reverse :
$([i \rightsquigarrow_p j][k \rightsquigarrow_q h]\varphi) \leftrightarrow ([k \rightsquigarrow_q h][i \rightsquigarrow_p j]\varphi)$
where $(j \neq k$ and $h \neq i)$ or $p \neq q$

objectivepermanence :
$\langle\delta\rangle\top \rightarrow (\varphi \leftrightarrow [\delta]\varphi)$ where φ is objective.

Some Results

- The axiomization is sound + complete.
- For DCL-PC, the model checking and satisfiability problems are PSPACE-complete (wrt semantics presented here).
- Why is model checking so hard?!
  Because we have very succinct models.
- Closest fragment of PDL is deterministic PDL, which is EXPTIME-complete.

Knowledge and Control

- partial observability of the state of the world
  - agents may be uncertain about the values of the variables
  - if agent i's goal is $p \leftrightarrow \neg q$ he can achieve this if (1) he controls at least one of the variables, and (2) if he controls one of them, he knows the value of the other

Knowledge through Observation

$$F = \langle N, \mathbb{A}_1, \ldots, \mathbb{A}_n, V_1, \ldots, V_n \rangle, \text{ where}$$

- $N = \{1, 2, \ldots, n\}$ is a (finite, nonempty) set of agents.
- The sets $\mathbb{A}_i$ form a partition of $\mathbb{A}$.
- $V_i \subseteq \mathbb{A}$ is the set of atoms whose values are visible to i.

Knowledge through Observation

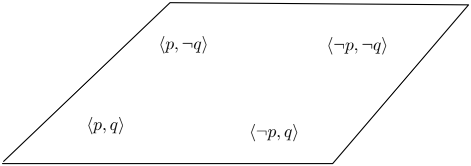
$$F = \langle N, \mathbb{A}_1, \ldots, \mathbb{A}_n, V_1, \ldots, V_n \rangle, \text{ where}$$

- $N = \{1, 2, \ldots, n\}$ is a (finite, nonempty) set of agents.
- The sets $\mathbb{A}_i$ form a partition of $\mathbb{A}$.
- $V_i \subseteq \mathbb{A}$ is the set of atoms whose values are visible to i.

- It seems natural to assume $\mathbb{A}_i \subseteq V_i$
- Write $K_i\varphi$ for 'i knows φ,
  i.e., φ is true in all states that look the same for i as the current state.

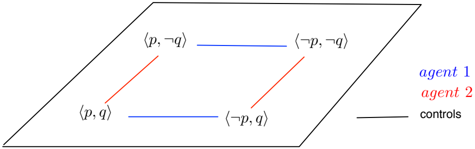
Example

Suppose $N = \{1,2\}$ and $\mathbb{A} = \{p,q\}$



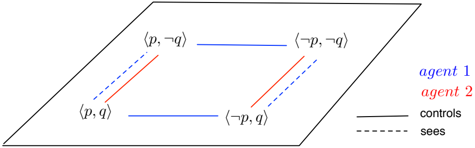
Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$



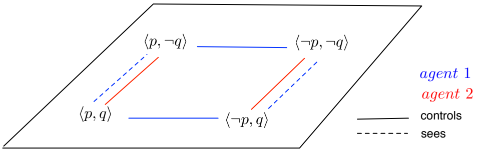
Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with
$\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$.



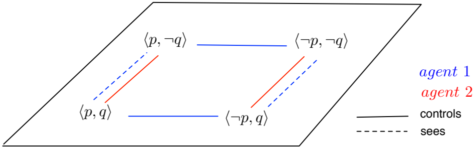
Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true.**



Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true**.

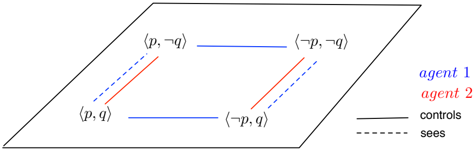


$F, \theta \models^d \Diamond_1 (p \leftrightarrow \neg q)$

Agent 1 can set his variable p in such a way that p and q have different values.

Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true**.

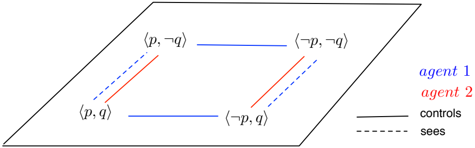


$F, \theta \models^d \neg K_1 q \wedge \neg K_1 \neg q \wedge K_1(K_2 q \vee K_2 \neg q)$

Agent 1 does not know the value of variable q, but he does know that 2 knows the value of q.

Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true**.

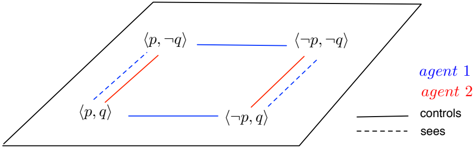


$F, \theta \models^d K_1 \Diamond_1 (p \leftrightarrow \neg q) \wedge \neg \Diamond_1 K_1 (p \leftrightarrow \neg q)$

Agent 1 knows that he can make p and q take on different values (because he controls p, and hence can make it different to q in any given state). However, agent 1 cannot choose values for the variables he controls in such a way that he knows that p and q take on different values.

Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true**.

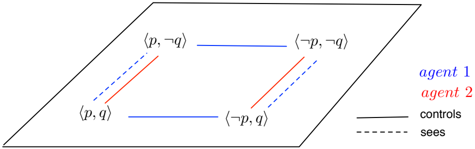


$F, \theta \models^d K_2 \Box_1((K_2 p \vee K_2 \neg p) \wedge (K_2 q \vee K_2 \neg q))$

Agent 2 knows that whatever truth values 1 chooses for her variables, 2 will know the value of p and of q.

Example

Suppose $N = \{1, 2\}$ and $\mathbb{A} = \{p, q\}$, with $\mathbb{A}_1 = \{p\}$, $\mathbb{A}_2 = \{q\}$ while $V_1 = \{p\}$ and $V_2 = \{p, q\}$. Let $\theta(p) = \theta(q) =$ **true**.



$F, \theta \models^d K_2((p \wedge q) \wedge \Diamond_1(\neg p \wedge \Diamond_2(\neg p \wedge \neg q)))$

Agent 2 knows that $(p \wedge q)$ and that 1 can bring about that $\neg p$ which 2 can further narrow down to $(\neg p \wedge \neg q)$.

Theoretical Results

Theorem

- *We have a sound and complete axiomatisation for this logic with control and knowledge;*
- *The model checking and satisfiability problems for the logic are both PSPACE-complete.*

Knowledge and Control

- partial observability of the state of the world
  - agents may be uncertain about the values of the variables
  - if agent i's goal is $p \leftrightarrow \neg q$ he can achieve this if
    (1) he controls at least one of the variables, and
    (2) if he controls one of them, he knows the value of the other

Knowledge and Control

- partial observability of the state of the world
  - agents may be uncertain about the values of the variables
  - if agent i's goal is $p \leftrightarrow \neg q$ he can achieve this if (1) he controls at least one of the variables, and (2) if he controls one of them, he knows the value of the other
- there may also be uncertainty about who controls what
  - agents may be uncertain about what they control themselves;
  - agents may be uncertain who to join coalitions with

Knowledge and Control

- partial observability of the state of the world
  - agents may be uncertain about the values of the variables
  - if agent i's goal is $p \leftrightarrow \neg q$ he can achieve this if (1) he controls at least one of the variables, and (2) if he controls one of them, he knows the value of the other
- there may also be uncertainty about who controls what
  - agents may be uncertain about what they control themselves;
  - agents may be uncertain who to join coalitions with
- there may be uncertainty about the above: higher order issues

Example: Voting

Assume agents either desire something (p_i) or not. They can reveal their preference through q_i: if $p_i \leftrightarrow q_i$, agent i is truthful, otherwise he lies.

Example: Voting

Assume agents either desire something (p_i) or not. They can reveal their preference through q_i: if $p_i \leftrightarrow q_i$, agent i is truthful, otherwise he lies.
Here, $\mathbb{A}_i = \{q_i\}$, $\mathbb{A}_{Nature} = \{p_1, \ldots, p_N\}$
Assume i knows who controls q_j. Moreover, $V_i = \{p_i\} \cup \{q_j \mid j \in N\}$. In other words, we assume agents cannot control what they prefer, although what they can do is choose their vote. They are aware of their own desire and other's revealed desires.

Example: Voting

Assume agents either desire something (p_i) or not. They can reveal their preference through q_i: if $p_i \leftrightarrow q_i$, agent i is truthful, otherwise he lies.
Here, $\mathbb{A}_i = \{q_i\}$, $\mathbb{A}_{Nature} = \{p_1, \ldots, p_N\}$
Assume i knows who controls q_j. Moreover, $V_i = \{p_i\} \cup \{q_j \mid j \in N\}$. In other words, we assume agents cannot control what they prefer, although what they can do is choose their vote. They are aware of their own desire and other's revealed desires.

$$K_i(\ell(p_i) \rightarrow (\Diamond_j(\ell(p_j) \wedge q_j) \wedge \Diamond_j(\ell(p_j) \wedge \neg q_j)))$$

i.e., i knows that j can vote truthfully but it can also lie.

Example: Voting

Assume agents either desire something (p_i) or not. They can reveal their preference through q_i: if $p_i \leftrightarrow q_i$, agent i is truthful, otherwise he lies.
Here, $\mathbb{A}_i = \{q_i\}$, $\mathbb{A}_{Nature} = \{p_1, \ldots, p_N\}$
Assume i knows who controls q_j. Moreover, $V_i = \{p_i\} \cup \{q_j \mid j \in N\}$. In other words, we assume agents cannot control what they prefer, although what they can do is choose their vote. They are aware of their own desire and other's revealed desires.

$$K_i(\ell(p_i) \to (\Diamond_j(\ell(p_j) \land q_j) \land \Diamond_j(\ell(p_j) \land \neg q_j)))$$

i.e., i knows that j can vote truthfully but it can also lie. We also get $K_i q_j \to \neg(K_i p_j \lor K_i \neg p_j)$: even if i knows j's vote, it does not know j's real preference.

Conclusion

Starting point: Coalition Logic for Propositional Control

- when building software agents, this is a natural way to think about their ability
- in some implemented systems (MOCHA) this is how powers of agents are specified

Conclusion

Starting point: Coalition Logic for Propositional Control

- CL-PC has been extended:
  - CL-PC was generalised to model partial and shared control ([Gerbrandy, 2006]): the $\mathbb{A}_i$ need not be a partition;
  - DCL-PC [vdH, Walther and Wooldridge, 2010]: a logic for control and delegation: The $\mathbb{A}_i$ are not fixed;
  - This can be given a dynamic logic twist, with basic programs $q \leftarrow \top$ (q is given the value *true*), $q \leftarrow \bot$, $i \overset{q}{\hookrightarrow}$ (i loses control over q) and $i \overset{q}{\hookleftarrow}$ [Herzig and Troquard, 2010]