Arbitrary public announcement logic with mental programs

Tristan Charrier   François Schwarzentruber

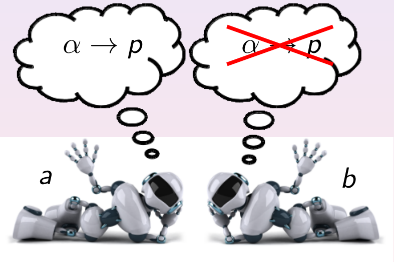
ENS Rennes

13 april 2015

Outline

Arbitrary public announcement logic: example

(α is true)

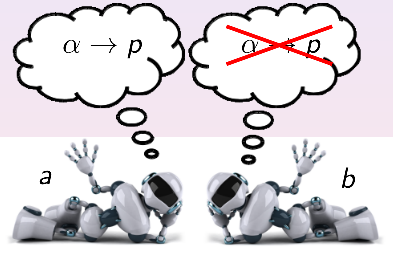


Could we publicly announce a true formula so that:

- a knows p;
- b does not know p?

Arbitrary public announcement logic: example

(α is true)



Could we publicly announce a true formula so that:

- a knows p;
- b does not know p?

Yes!  $\boxed{\alpha}$

Arbitrary public announcement logic: definition

Syntax

$$\varphi, \psi, \ldots \quad ::= \quad p \mid \neg\varphi \mid \varphi \vee \psi \mid K_a\varphi \mid \langle\psi!\rangle\varphi \mid \Diamond\varphi$$

- $K_a\varphi$: agent a knows φ
- $\langle\psi!\rangle\varphi$: ψ is true and after publicly announcing ψ, φ holds.
- $\Diamond\varphi$: there exists a ($\Diamond$-free) formula ψ such that $\langle\psi!\rangle\varphi$ holds.

Semantics

Kripke models. Public announcement = restrictions.

Theorem

The satisfiability problem in arbitrary public announcement logic is ***undecidable****.*

Summary of our contribution

Variant of arbitrary public announcement logic: DLPA-APAL

- Possible worlds are *valuations*;
- Epistemic relations are defined by *mental programs*

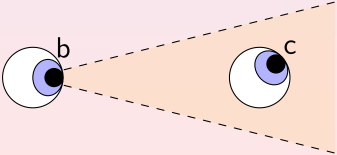
  expressed in Dynamic Logic of Propositional Assignements.

- **decidable**

  Model checking and satisfiability problem are A_{poly}EXPTIME-complete.

- **expressive**

  Proof-of-concept: epistemic logic for autonomous cameras in the plane.



Outline

Outline

Syntax of DLPA-APAL

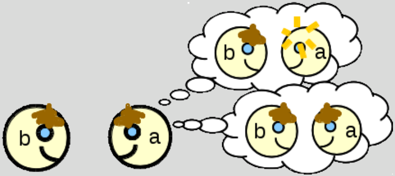
Syntax

$$\varphi, \psi, \ldots \quad ::= \quad p \mid \neg\varphi \mid \varphi \vee \psi \mid \cancel{K_a \varphi} \mid \langle \psi ! \rangle \varphi \mid \Diamond \varphi$$

$K_\pi \varphi$ instead

where π is a **mental program**

Example (mental program for child *a*)



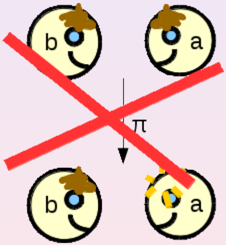
program π:

non-deterministic choice:

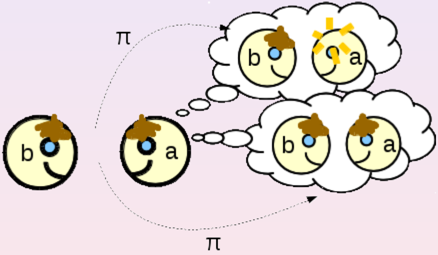
1. **either** clean *a*'s forehead
2. **or** make *a*'s forehead dirty

Mental programs...

are not performed...



but they describe epistemic relations.



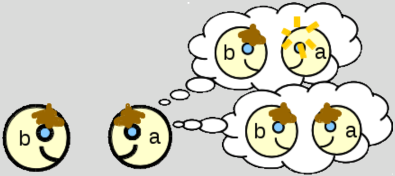
Syntax for mental programs

Syntax (Dynamic logic of propositional assignments)

$$\pi \quad ::= \quad \underbrace{p \leftarrow \bot \mid p \leftarrow \top}_{\text{assignements}} \mid \beta? \mid \pi ; \pi \mid \pi \cup \pi$$

- β?: test whether the Boolean formula β is true;
- sequence;
- non-deterministic choice.

Example (mental program for child a)

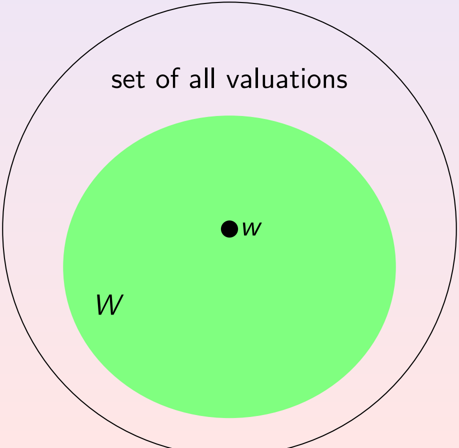


$$mud_a \leftarrow \bot \cup mud_a \leftarrow \top$$

Outline

Semantics

$[[\varphi]]_W$: subset of valuations satisfying φ when already made announcements restricted possible worlds to W.



Semantics

$$
\begin{aligned}
[\![\top]\!]_W &= W; \\
[\![p]\!]_W &= \{w \in W \mid p \in w\}; \\
[\![\neg\varphi]\!]_W &= W \setminus [\![\varphi]\!]_W; \\
[\![\varphi \vee \psi]\!]_W &= [\![\varphi]\!]_W \cup [\![\psi]\!]_W; \\
[\![\hat{K}_\pi\varphi]\!]_W &= \left\{ w \in W \mid \begin{array}{l}\text{there exists } u \in W,\ (w,u) \in [\![\pi]\!] \\ \text{and } u \in [\![\varphi]\!]_W\end{array} \right\}; \\
[\![\langle\psi!\rangle\varphi]\!]_W &= [\![\psi]\!]_W \cap [\![\varphi]\!]_{W \cap [\![\psi]\!]_W}; \\
[\![\Diamond\varphi]\!]_W &= \left\{ u \in W \mid \begin{array}{l}\text{there exists a } {\scriptstyle(\Diamond\text{-free})} \text{ formula } \psi \\ \text{such that } u \in [\![\langle\psi!\rangle\varphi]\!]_W\end{array} \right\};
\end{aligned}
$$

$$
\begin{aligned}
[\![p\leftarrow\bot]\!] &= \left\{(w,u) \in W^2_{all} \mid u = w \setminus \{p\}\right\}; \\
[\![p\leftarrow\top]\!] &= \left\{(w,u) \in W^2_{all} \mid u = w \cup \{p\}\right\}; \\
[\![\pi;\pi']\!] &= [\![\pi]\!] \circ [\![\pi']\!]; \\
[\![\pi \cup \pi']\!] &= [\![\pi]\!] \cup [\![\pi']\!]; \\
[\![\beta?]\!] &= \left\{(w,w) \in W^2_{all} \mid w \models_{PL} \beta\right\}
\end{aligned}
$$

Outline

Outline

1. Introduction
2. Definition of our variant DLPA-APAL
3. Decidability and complexity results
    - Decision problems
    - Recall of A_{poly}EXPTIME
    - Upper bound
    - Lower bound
4. Proof-of-concept: cameras
5. Future work

Decision problems

Model checking

- input: a valuation w, a formula φ;
- output: yes iff $W_{\text{all valuations}}, w \models \varphi$.

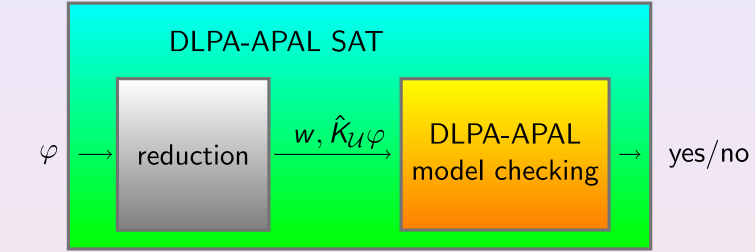
Satisfiability problem

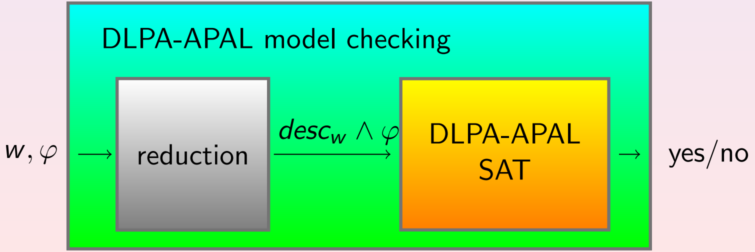
- input: a formula φ;
- output: yes iff there exists a valuation w such that $W_{\text{all valuations}}, w \models \varphi$.

Theorem

Both decision problems are A_{poly}EXPTIME-complete.

Decision problems are interreducible

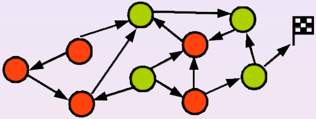




Outline

Alternating Turing Machines

- -states and -states



- Two players:

 plays in -states

 plays in -states

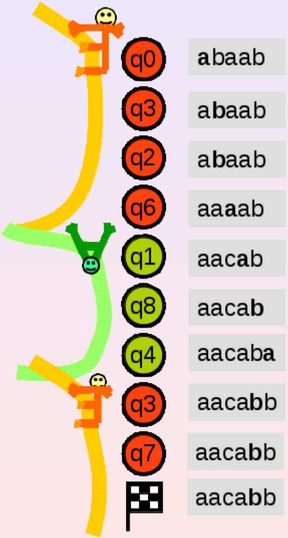
- The initial input word is accepted iff  has a strategy to end in the accepting state .

A_{poly}EXPTIME

Definition

A_{poly}EXPTIME is the class of problems solvable:

- by an **A**lternating Turing machine
- in **exp**onential **time**
- but the number of alternations is **poly**nomial.



Outline

Theorem

The model checking in DLPA-APAL is in $A_{poly}EXPTIME$.

Proof.

procedure MC(w, φ)
  We compute W_{all}
  call $mc_{yes}(W_{all}, w, \varphi)$

□

procedure $mc_{yes}(W, w, \varphi)$
- **match** φ **with**
  - **case** $\varphi = p$: **if** $p \notin w$ **then reject**
  - **case** $\varphi = \neg\psi$: $mc_{no}(W, w, \psi)$
  - **case** $\varphi = \psi_1 \vee \psi_2$:
    - $(\exists)$ choose $i \in \{1, 2\}$
    - $mc_{yes}(W, w, \psi_i)$
  - **case** $\varphi = \hat{K}_\pi\psi$:
    - $(\exists)$ choose $u \in W$
    - $ispath_{yes}(w, u, \pi)$
    - $mc_{yes}(W, u, \psi)$
  - **case** $\varphi = \langle\psi!\rangle\chi$:
    - $mc_{yes}(W, w, \psi)$
    - $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
    - $W'' = W' \cup \{w\}$
    - $(\forall)$ choose $u \in W''$
    - $(\forall)$ choose $v \in W \backslash W''$
    - $mc_{yes}(W, u, \psi)$
    - $mc_{no}(W, v, \psi)$
    - $mc_{yes}(W'', w, \chi)$
  - **case** $\varphi = \Diamond\chi$:
    - $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
    - $W'' = W' \cup \{w\}$
    - $mc_{yes}(W'', w, \chi)$

procedure $mc_{no}(W, w, \varphi)$
- **match** φ **with**
  - **case** $\varphi = p$: **if** $p \in w$ **then reject**
  - **case** $\varphi = \neg\psi$: $mc_{yes}(W, w, \psi)$
  - **case** $\varphi = \psi_1 \vee \psi_2$:
    - $(\forall)$ choose $i \in \{1, 2\}$
    - $mc_{no}(W, w, \psi_i)$
  - **case** $\varphi = \hat{K}_\pi\psi$:
    - $(\forall)$ choose $u \in W$
    - $(\exists)$ choose $i \in \{0, 1\}$
    - **if** $i = 0$ **then**
      - $ispath_{no}(w, u, \pi)$
    - **else**
      - $mc_{no}(W, u, \psi)$
  - **case** $\varphi = \langle\psi!\rangle\chi$:
    - $(\exists)$ choose $i \in \{0, 1\}$
    - **if** $i = 0$ **then**
      - $mc_{no}(W, w, \psi)$
    - **else**
      - $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
      - $W'' = W' \cup \{w\}$
      - $(\forall)$ choose $u \in W''$
      - $(\forall)$ choose $v \in W \backslash W''$
      - $mc_{yes}(W, u, \psi)$
      - $mc_{no}(W, v, \psi)$

DUAL

procedure $ispath_{yes}(w, u, \pi)$
match π **with**
case $\pi = p{\leftarrow}\bot$:
if $u \neq w \setminus \{p\}$ **then reject**
case $\pi = p{\leftarrow}\top$:
if $u \neq w \cup \{p\}$ **then reject**
case $\pi = \pi_1 ; \pi_2$:
($\exists$) choose a valuation v.
$ispath_{yes}(w, v, \pi_1)$
$ispath_{yes}(v, u, \pi_2)$
case $\pi = \pi_1 \cup \pi_2$:
($\exists$) choose $i \in \{1, 2\}$
$ispath_{yes}(w, u, \pi_i)$
case $\pi = \beta?$:
if $w \neq u$ or $w \not\models_{PL} \beta$
then reject

procedure $ispath_{no}(w, u, \pi)$
match π **with**
case $\pi = p{\leftarrow}\bot$:
if $u = w \setminus \{p\}$ **then reject**
case $\pi = p{\leftarrow}\top$:
if $u = w \cup \{p\}$ **then reject**
case $\pi = \pi_1 ; \pi_2$:
($\forall$) choose a valuation v.
($\exists$) choose $i \in \{1, 2\}$
if $i = 1$
then $ispath_{no}(w, v, \pi_1)$
else $ispath_{no}(v, u, \pi_2)$
case $\pi = \pi_1 \cup \pi_2$:
($\forall$) choose $i \in \{1, 2\}$
$ispath_{no}(w, u, \pi_i)$
case $\pi = \beta?$:
if $w = u$ and $w \models_{PL} \beta$
then reject

DUAL

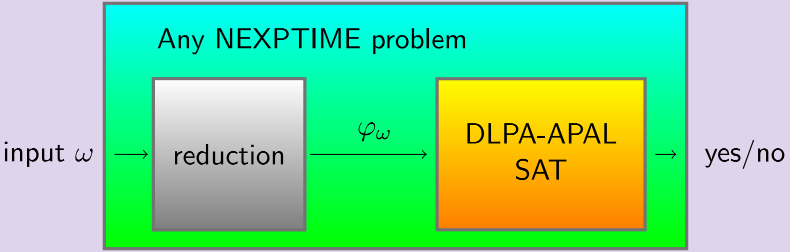
Outline

First step: NEXPTIME-hardness

Theorem

The satisfiability problem in DLPA-APAL is NEXPTIME-hard.

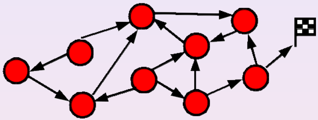
Proof.



□

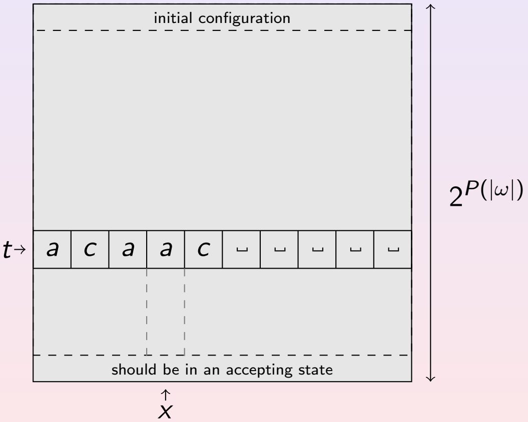
Proof

Let M the Turing machine associated to the NEXPTIME problem.



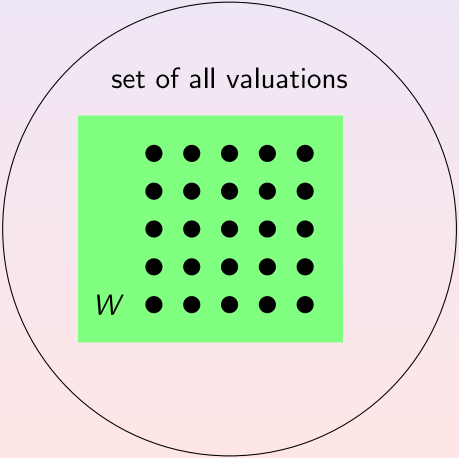
ω is a positive instance
iff
there exists an accepting execution of M starting with ω
iff
φ_ω (to be defined) is DLPA-APAL-satisfiable

Representing an accepting execution as a grid



Main idea

$$\varphi_\omega := \Diamond \left(\text{a formula stating that the remaining set of valuations } W \text{ represents an accepting execution from } \omega \right)$$



A valuation in W

$=$ a cell at a given time

Example of a valuation

$\neg$ cursor is here

$\neg x_0, \neg x_1, x_2, \neg x_3$

$(x = 4)$

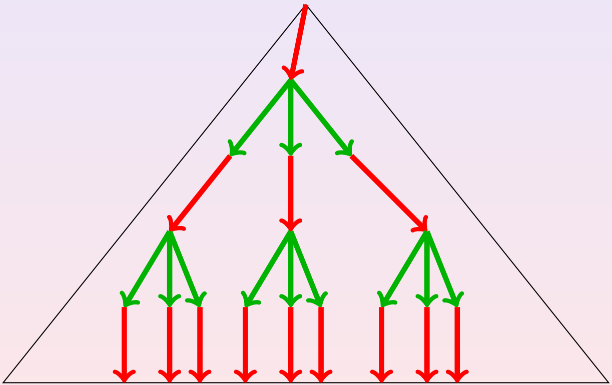
$\neg t_0, t_1, \neg t_2, \neg t_3$

$(t = 2)$

current state q_{13}

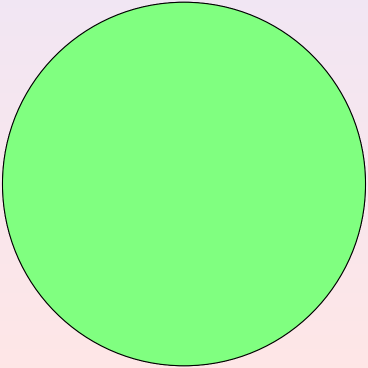
a is in the cell

A_{poly}EXPTIME-hard



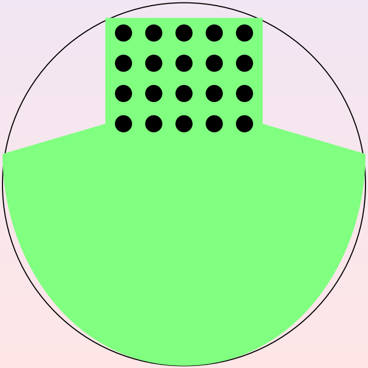
Main idea (revisited)

$$\varphi_\omega := \Diamond \left(\boxed{\text{the first interval of time is an execution starting from } \omega \text{ and the rest is unconstrained}} \wedge \Box \left(\boxed{\text{the second interval of time is the continuation of the execution and the rest is unconstrained}} \rightarrow \Diamond \boxed{\text{the last interval of time is the end of the execution and it is accepting}} \right) \right)$$



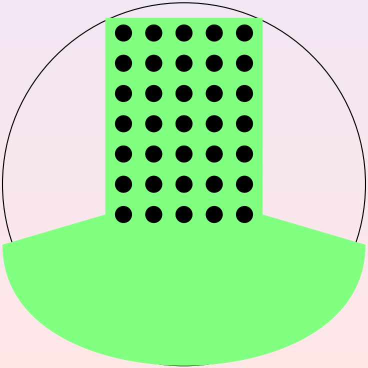
Main idea (revisited)

$$\varphi_\omega := \Diamond \left(\text{the first interval of time is an execution starting from } \omega \text{ and the rest is unconstrained} \wedge \Box \left(\text{the second interval of time is the continuation of the execution and the rest is unconstrained} \rightarrow \Diamond \text{the last interval of time is the end of the execution and it is accepting} \right) \right)$$



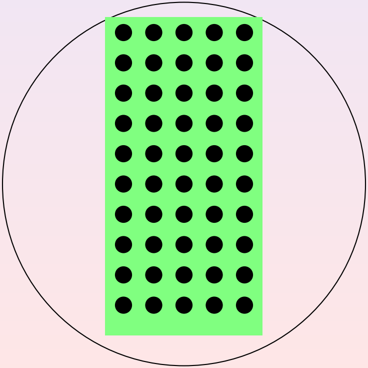
Main idea (revisited)

$$\varphi_\omega := \Diamond\left(\text{the first interval of time is an execution starting from } \omega \text{ and the rest is unconstrained} \wedge \Box\left(\text{the second interval of time is the continuation of the execution and the rest is unconstrained} \rightarrow \Diamond\,\text{the last interval of time is the end of the execution and it is accepting}\right)\right)$$



Main idea (revisited)

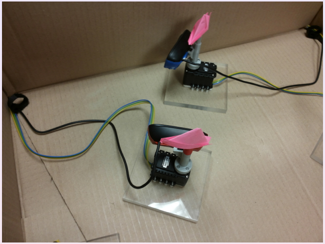
$$\varphi_\omega := \Diamond\left(\text{the first interval of time is an execution starting from } \omega \text{ and the rest is unconstrained} \wedge \Box\left(\text{the second interval of time is the continuation of the execution and the rest is unconstrained} \rightarrow \Diamond\, \text{the last interval of time is the end of the execution and it is accepting}\right)\right)$$



Outline

Outline

Agents are cameras



Cameras

- Can turn;
- Can not move.

(simplification of a real multi-robot environment)

Common knowledge

- of the positions of agents;
- of the abilities of perception.

Syntax

Syntax

$$\varphi, \psi, \ldots \quad ::= \quad a \text{ sees } b \mid \neg\varphi \mid \varphi \vee \psi \mid K_a\varphi \mid \langle\varphi!\rangle\psi \mid \Diamond\psi$$

agent a knows φ

φ is true and after publicly announcing ψ, φ holds.

there exists a ($\Diamond$-free) formula ψ such that $\langle\psi!\rangle\varphi$ holds.

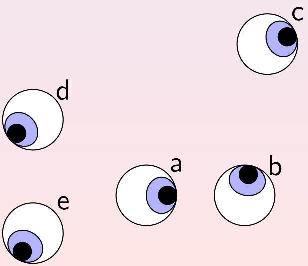
Semantics

Assumption

Common knowledge of the positions.

Set of worlds

Given a fixed $pos' : AGENTS \to \mathbb{R}^2$,
worlds are $w = (pos, dir)$ s. th. $pos = pos'$



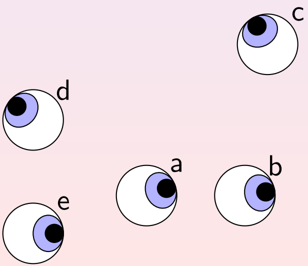
Semantics

Assumption

Common knowledge of the positions.

Set of worlds

Given a fixed $pos' : AGENTS \to \mathbb{R}^2$,
worlds are $w = (pos, dir)$ s. th. $pos = pos'$



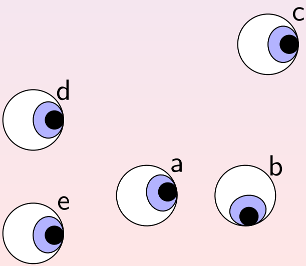
Semantics

Assumption

Common knowledge of the positions.

Set of worlds

Given a fixed $pos' : AGENTS \to \mathbb{R}^2$,
worlds are $w = (pos, dir)$ s. th. $pos = pos'$



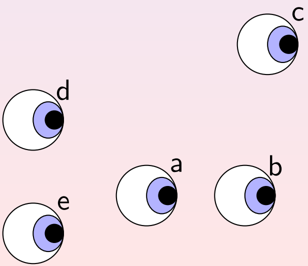
Semantics

Assumption

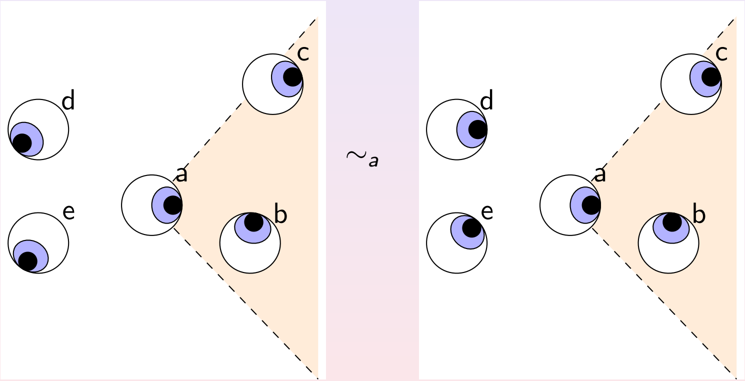
Common knowledge of the positions.

Set of worlds

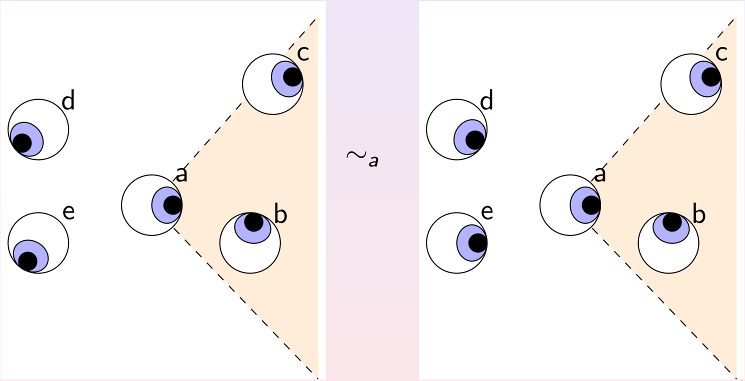
Given a fixed $pos' : AGENTS \to \mathbb{R}^2$,
worlds are $w = (pos, dir)$ s. th. $pos = pos'$



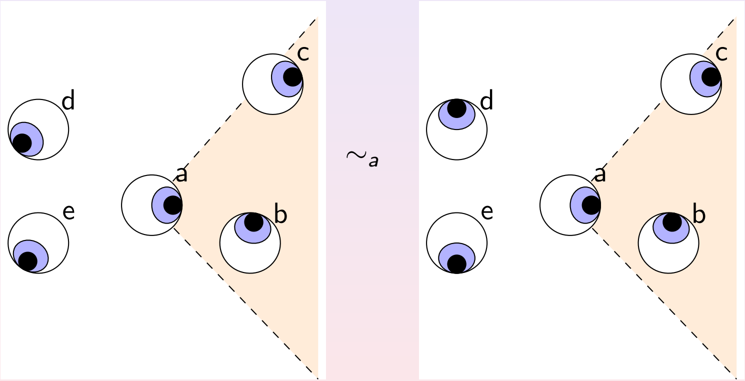
Semantics: $\mathcal{M}^{pos'}_{\text{cameras}}$



Semantics: $\mathcal{M}^{pos'}_{\text{cameras}}$



Semantics: $\mathcal{M}^{pos'}_{\text{cameras}}$



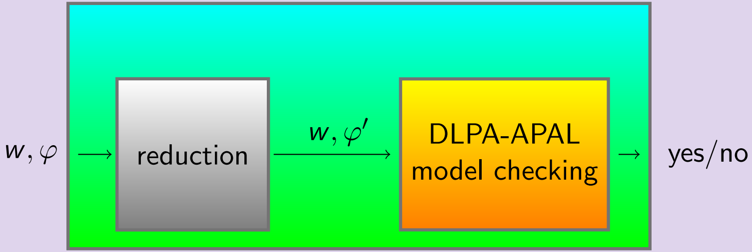
Outline

Spoiler

Theorem

Model checking of a formula containing arbitrary public announcement operator in a setting of cameras is in $A_{poly}EXPTIME$.

Proof.



□

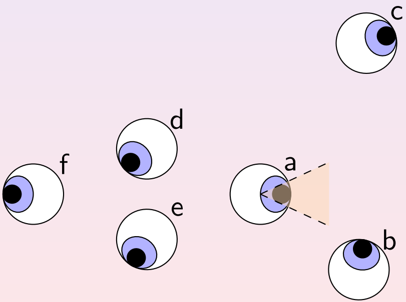
Outline

Compute vision sets

Set of vision sets of agent a

$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

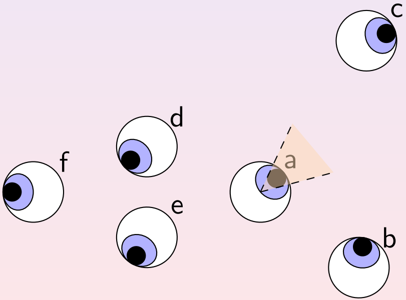


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

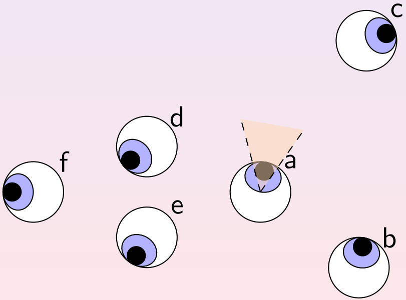


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

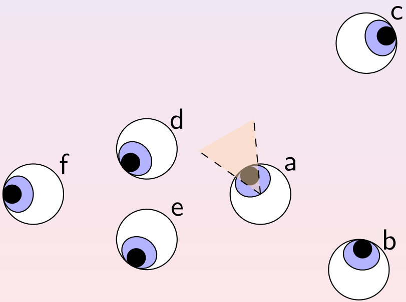


Compute vision sets

Set of vision sets of agent a

$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

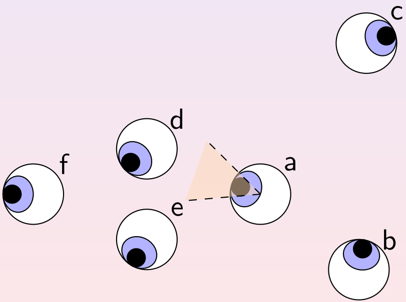


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

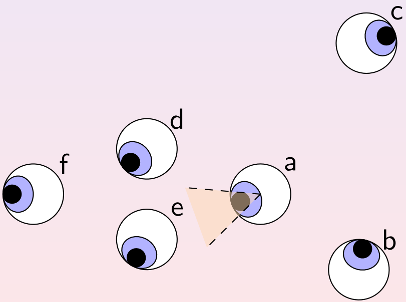


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

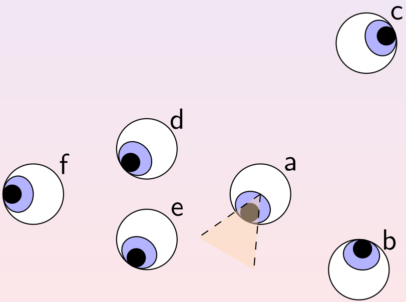


Compute vision sets

Set of vision sets of agent a

$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}$.

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

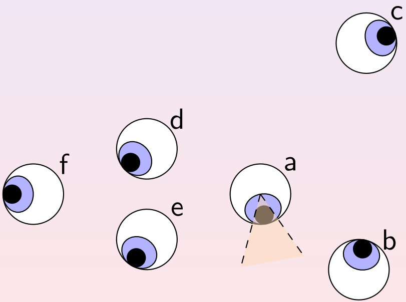


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

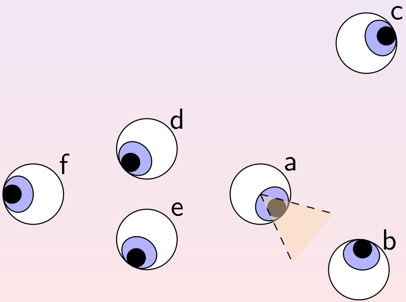


Compute vision sets

Set of vision sets of agent a

$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}$.

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$

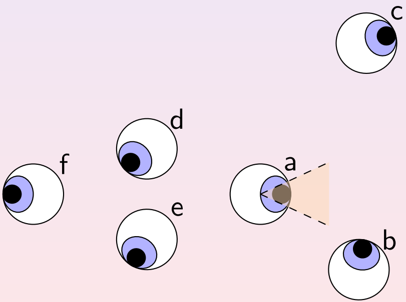


Compute vision sets

Set of vision sets of agent a

$$\mathcal{V}_a = \{\{b\}, \emptyset, \{c\}, \{d\}, \{d, f\}, \{d, f, e\}, \{f, e\}, \{e\}\}.$$

$\mathcal{V}_a$ computed in $O(\#AGT \log \#AGT)$



Announcing vision sets

Each agent sees one of its vision set:

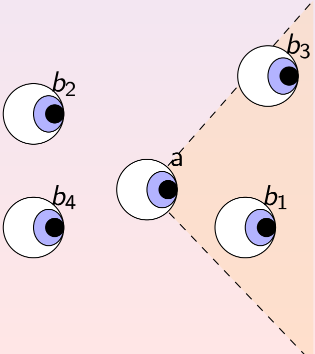
$$VisionSets := \bigwedge_{a \in AGT} \bigvee_{\Gamma \in \mathcal{V}_a} \left(\bigwedge_{b \in \Gamma} a \text{ sees } b \wedge \bigwedge_{b \notin \Gamma} \neg a \text{ sees } b \right)$$

Translating epistemic operators in programs

K_a is simulated by K_{π_a} where π_a is:

$$\left[\left(a \text{ sees } b_1? \cup (a \cancel{\text{ sees }} b_1?; \overset{\curvearrowright}{b_1})\right); \ldots ; \left(a \text{ sees } b_n? \cup (a \cancel{\text{ sees }} b_n?; \overset{\curvearrowright}{b_n})\right)\right]$$

where $\overset{\curvearrowright}{c} := ((c \text{ sees } a \leftarrow \bot) \cup (c \text{ sees } a \leftarrow \top)) ; ((c \text{ sees } b_1 \leftarrow \bot) \cup (c \text{ sees } b_1 \leftarrow \top)) ; \ldots$

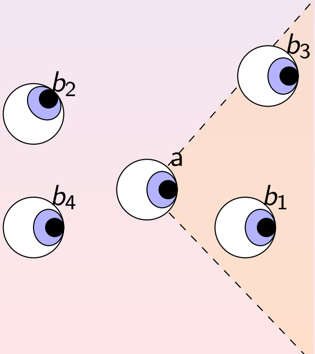


Translating epistemic operators in programs

K_a is simulated by K_{π_a} where π_a is:

$$\left[\left(a\,\text{sees}\,b_1? \cup (a\,\cancel{\text{sees}}\,b_1?;\overset{\curvearrowright}{b_1})\right);\ldots;\left(a\,\text{sees}\,b_n? \cup (a\,\cancel{\text{sees}}\,b_n?;\overset{\curvearrowright}{b_n})\right)\right]$$

where $\overset{\curvearrowright}{c} := ((c\ \text{sees}\ a \leftarrow \bot) \cup (c\ \text{sees}\ a \leftarrow \top))\,;((c\ \text{sees}\ b_1 \leftarrow \bot) \cup (c\ \text{sees}\ b_1 \leftarrow \top))\,;\ldots$

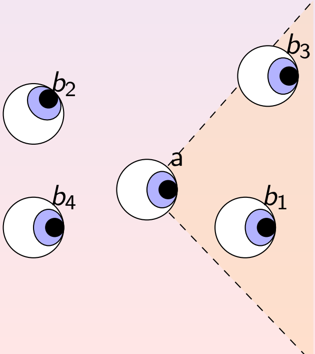


Translating epistemic operators in programs

K_a is simulated by K_{π_a} where π_a is:

$$\left[\left(a\ \text{sees}\ b_1? \cup (a\ \cancel{\text{sees}}\ b_1?; \overset{\curvearrowright}{b_1})\right); \ldots; \left(a\ \text{sees}\ b_n? \cup (a\ \cancel{\text{sees}}\ b_n?; \overset{\curvearrowright}{b_n})\right)\right]$$

where $\overset{\curvearrowright}{c} := ((c\ \text{sees}\ a \leftarrow \bot) \cup (c\ \text{sees}\ a \leftarrow \top)) ; ((c\ \text{sees}\ b_1 \leftarrow \bot) \cup (c\ \text{sees}\ b_1 \leftarrow \top)) ; \ldots$

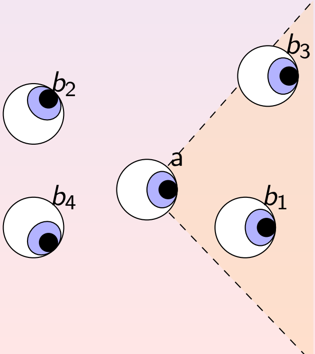


Translating epistemic operators in programs

K_a is simulated by K_{π_a} where π_a is:

$$\left[\left(a \text{ sees } b_1? \cup (\cancel{a \text{ sees}}\, b_1?; \overset{\curvearrowright}{b_1})\right); \ldots ; \left(a \text{ sees } b_n? \cup (\cancel{a \text{ sees}}\, b_n?; \overset{\curvearrowright}{b_n})\right)\right]$$

where $\overset{\curvearrowright}{c} := ((c \text{ sees } a \leftarrow \bot) \cup (c \text{ sees } a \leftarrow \top)) ; ((c \text{ sees } b_1 \leftarrow \bot) \cup (c \text{ sees } b_1 \leftarrow \top)) ; \ldots$



Reduction

Proposition

Let w a configuration of cameras. Let φ a formula.

$$w \models_{\text{epistemic logic for cameras}} \varphi \quad \text{iff} \quad w \models_{\text{DLPA-APAL}} [VisionSets!]tr(\varphi)$$

where $tr(K_a\psi) = K_{\pi_a} tr(\psi)$.

Outline

Future work

Extensions that still crack the undecidability of APAL

- Kleene-star constructions;
- Not only one-valuation models but maybe refinements;
- Other arbitrary actions, epistemic planning, etc.;

Practical

- Generate ‘the formula to announce’;
- First order theories;
- Find more tractable fragments of DLPA-APAL;

In particular, is model checking in APAL with cameras A_{poly}EXPTIME-hard?

- Implementation;
- Build a demo with real cameras.

Thank you for your attention.