Reasoning about Plan Revision in Agent Programs

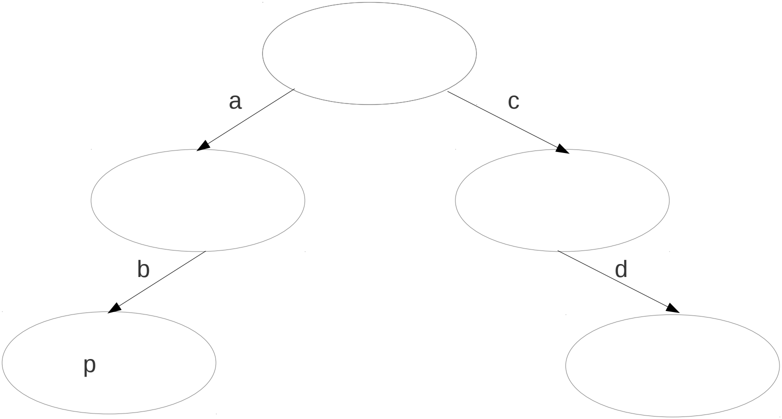
Natasha Alechina
University of Nottingham, UK

TIME 2012, Leicester 14 September 2012

What this talk is about

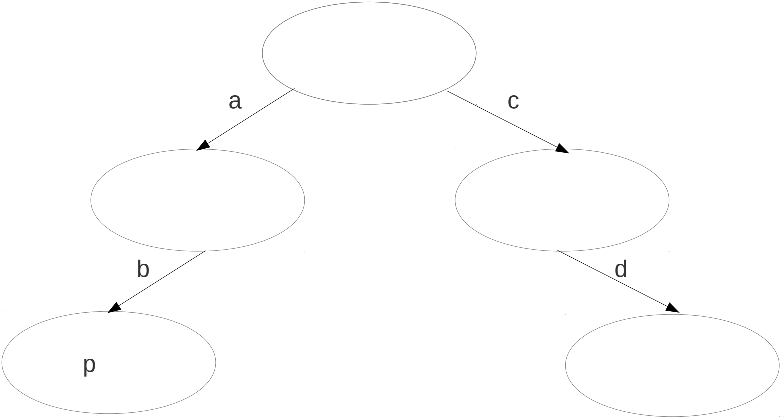
- verification (of *agent programs* with *changing plans*)
- transition systems correspond to agent program execution
- model-checking agent programs
- joint work with Brian Logan, Mehdi Dastani and John-Jules Meyer on a theorem-proving approach (using dynamic logic)
- main extension: explicit operator for ‘having a plan’

Transition systems

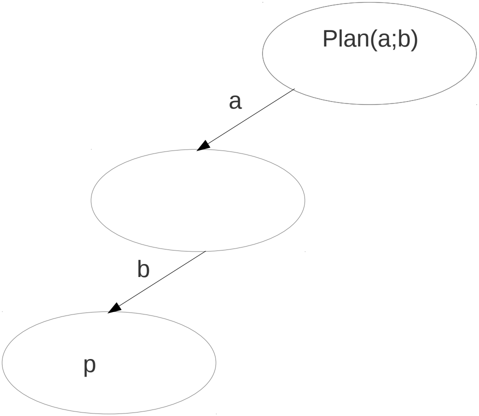


Dynamic logic

$\langle a; b \rangle p$, $\langle c; d \rangle p$



Having and executing a plan



What is an agent?

- many definitions of ‘agent’ in the literature — key ideas include:
- **autonomy:** an agent operates without the direct intervention of humans or other agents
- **situatedness:** an agent interacts with its environment (which may contain other agents)
- **reactivity:** an agent responds in a timely fashion to changes in its environment
- **proactivity:** an agent exhibits goal-directed behaviour

What I will mean by an agent

- a computational system whose behaviour can be usefully characterised in terms of propositional attitudes such as beliefs and goals
- and which is programmed in an agent programming language which makes explicit use of propositional attitudes

What is an agent programming language?

- Belief, Desire and Intentions (BDI) framework, (Bratman 1987)
- *BDI agent programming languages* are designed to facilitate the implementation of BDI agents:
  - *programming constructs* corresponding to beliefs, desires and intentions
  - agent *architecture* or *interpreter* enforces relationships between beliefs, desires and intentions and which causes the agent to choose actions to achieve its goals based on its beliefs

3APL

- one of the first agent programming languages PRS (Georgeff and Ingrand 1988), very rich. I will talk about a more modern and less rich langauge, 3APL
- 3APL is a BDI agent programming language proposed in (Dastani et al. 2003)
- I present a cut-down version of 3APL (mostly regarding the language for beliefs, but also distinction between external and internal actions, not considering messages etc.)

3APL beliefs

- the beliefs of a 3APL agent represent its information about its environment and itself
- beliefs are represented by a set of positive literals
- the initial beliefs of an agent are specified by its program
- e.g., the agent may initially believe that it's in *room1* and its battery is charged:

```
Beliefs:
  room1, battery
```

3APL goals

- the agent's goals represent situations the agent wants to realise (not necessarily all at once)
- goals are represented by a set of arbitrary literals
- the initial goals of an agent are specified by its program
- e.g., the agent may initially want to achieve a situation in which both *room1* and *room2* are clean

```
Goals:
  clean1, clean2
```

Declarative goals

- the beliefs and goals of an agent are related to each other
  - if an agent believes p, then it will not pursue p as a goal
  - if an agent does not believe that p, it will not have $-p$ as a goal
- these relationships are enforced by the agent architecture

3APL basic actions

- *basic actions* specify the capabilities of the agent (what it can do independent of any particular agent program)
- 2 types of basic actions:
  - belief test actions: test whether the agent has a given belief
  - belief update actions: “external” actions which change the agent’s beliefs

Belief test actions

- a *belief test action* ϕ? tests whether a boolean belief expression ϕ is entailed by the agent's beliefs, e.g.:

  ```
  (room2 and -battery)?
  ```

  tests whether the agent believes it is in *room2* and its battery is not charged

Belief update actions

- *belief update actions* change the beliefs (and goals) of the agent
- a belief update action is specified in terms of its pre- and postconditions (sets of literals), e.g.:

  ```
  {room1} moveR {  }, {-room1, room2}
  ```

- an action can be executed if one of its pre-conditions is entailed by the agent's current beliefs
- executing the action updates the agent's beliefs to make one of the postconditions entailed by the agent's beliefs (actions non-deterministic)

Belief entailment

- a belief query (a belief test action or an action precondition) is entailed by the agent's belief base if
  - all positive literals in the query are contained in the agent's belief base, and
  - for every negative literal $-p$ in the query, p is not in the belief base
  - i.e., we use entailment under the closed world assumption
- goal entailment corresponds to a formula being classically entailed by *one* of the goals in the goal base

Belief update

- executing a belief update action
  - adds all positive literals in the corresponding postcondition to the belief base, and
  - for every negative literal $-p$ in the postcondition, p is removed from the agent's belief base
- goals which are achieved by the postcondition of an action are dropped
- for simplicity, we assume that the agent's beliefs about its environment are always correct and its actions in the environment are always successful

Abstract plans

- unlike basic actions, *abstract plans* cannot be directly executed by the agent.
- abstract plans provide an abstraction mechanism (similar to procedures in imperative programming) which are expanded into basic actions using plan revision rules
- if the first step of a plan π is an abstract plan $\bar{\alpha}$, execution of π blocks.

3APL plans

- *plans* are sequences of basic actions and atomic plans composed by plan composition operators:
  - sequence: "$\pi_1 ; \pi_2$" (do π_1 then π_2)
  - conditional choice: "$\texttt{if } \phi \texttt{ then } \{\pi_1\} \texttt{ else } \{\pi_2\}$"
  - conditional iteration: "$\texttt{while } \phi \texttt{ do } \{\pi\}$"
- e.g., the plan:

  $\texttt{if room1 then } \{\texttt{suck}\} \texttt{ else } \{\texttt{moveL; suck}\}$

  causes the agent to clean *room1* if it's currently in *room1*, otherwise it first moves (left) to *room1* and then cleans it

3APL PG rules

- *planning goal rules* are used for plan selection based on the agent's current goals and beliefs
- a planning goal rule $\kappa \leftarrow \beta \mid \pi$ consists of three parts:
  - κ: an (optional) *goal query* which specifies which goal(s) the plan achieves
  - β: a *belief query* which characterises the situation(s) in which it could be a good idea to execute the plan
  - π: a plan
- a PG rule can be applied if κ is entailed by the agent's goals and β is entailed by the agent's beliefs
- applying the rule adds π to the agent's plans

Example 3APL PG rules

- ```
  clean2 <- battery |
    if room2 then {suck} else {moveR; suck}
  ```

  states that "if the agent's goal is to clean *room2* and its battery is charged, then the specified plan may be used to clean the room"

- an agent can generate a plan based only on its current beliefs (reactive invocation), e.g., the rule:

  ```
  <- -battery |
    if room2 then {charge} else {moveR; charge}
  ```

  states "if the battery is low, the specified plan may be used to charge it"
  ```

Example 3APL PR rules

- a plan revision rule $p_j = \pi_j \leftarrow \beta_j \,|\, \pi'_j$ can be applied if π_j is in the plan base, β_j is entailed by the agent's beliefs and π_j is not executable,
- in other words the first action of π_j is either a belief update or belief test action which is not executable in the current belief state, or an abstract plan
- for example, if *moveR* fails, the agent may execute a slow but reliable version of the action, *slowR*:

```
charge <- room1 |
  {slowR; charge}
```

Operational semantics

- we define the operational semantics of 3APL in terms of a transition system
- states are *agent configurations* $\langle \sigma, \gamma, \Pi \rangle$ where σ, γ are sets of literals representing the agent's beliefs and goals, and Π is a set of plan entries representing the agent's current active plans (annotated by the goals which they were adopted to achieve)
- each transition corresponds to a single step in the execution of the agent
- different execution strategies give rise to different semantics
- for simplicity we focus on non-interleaved execution—i.e., the agent executes a single plan to completion before choosing another plan

Formal entailment definitions

- $\models_{cwa}$ (belief entailment for closed world assumption):

  $\sigma \models_{cwa} p$ iff $p \in \sigma$

  $\sigma \models_{cwa} -p$ iff $p \notin \sigma$

  $\sigma \models_{cwa} \phi \texttt{ and } \psi$ iff $\sigma \models_{cwa} \phi$ and $\sigma \models_{cwa} \psi$

  $\sigma \models_{cwa} \phi \texttt{ or } \psi$ iff $\sigma \models_{cwa} \phi$ or $\sigma \models_{cwa} \psi$

  $\sigma \models_{cwa} \{\phi_1, \ldots, \phi_n\}$ iff $\forall 1 \leq i \leq n \;\; \sigma \models_{cwa} \phi_i$

- $\models_g$ (goal entailment):

  $\gamma \models_g p$ iff $p \in \gamma$

  $\gamma \models_g -p$ iff $-p \in \gamma$

  $\gamma \models_g \phi \texttt{ or } \psi$ iff $\gamma \models_g \phi$ or $\gamma \models_g \psi$

Belief update function

- let a be a belief update action and σ a belief base such that $\sigma \models_{cwa} \mathtt{prec}_{\mathtt{j}}(a)$
- intuitively, $\sigma \models_{cwa} \mathtt{prec}_{\mathtt{j}}(a)$ if it contains all positive literals in $\mathtt{prec}_{\mathtt{j}}(a)$ and does not contain the negative ones
- the result of executing belief update action a with respect to σ (assuming $\mathtt{prec}_{\mathtt{j}}(a)$ holds and the action results in the $\mathtt{post}_{\mathtt{j,i}}$ becoming true) is defined as:

$$T_{j,i}(a,\sigma) = (\sigma \cup \{p : p \in \mathtt{post}_{\mathtt{j,i}}(\mathtt{a})\}) \setminus \{p : -p \in \mathtt{post}_{\mathtt{j,i}}(\mathtt{a})\}$$

- intuitively, the result of the update satisfies (entails under $\models_{cwa}$) the corresponding postcondition $\mathtt{post}_{\mathtt{j,i}}(\mathtt{a})$

Transitions: belief test actions

- belief test actions

$$\frac{\sigma \models_{\mathit{cwa}} \beta}{\langle \sigma, \gamma, \{\beta?; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi \triangleright \kappa\}\rangle}$$

Transitions: belief update actions

- belief update actions when the corresponding goal not achieved yet:

$$\frac{\sigma \models_{cwa} \texttt{prec}_i(\alpha) \;\; T_{i,j}(\alpha,\sigma) = \sigma' \quad \gamma' = \gamma \setminus \{\phi \mid \sigma' \models_{cwa} \phi\} \;\; \sigma' \not\models_{cwa} \kappa}{\langle \sigma, \gamma, \{\alpha; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma', \gamma', \{\pi \triangleright \kappa\}\rangle}$$

- belief update actions when the corresponding goal is achieved:

$$\frac{\sigma \models_{cwa} \texttt{prec}_i(\alpha) \;\; T_{i,j}(\alpha,\sigma) = \sigma' \quad \gamma' = \gamma \setminus \{\phi \mid \sigma' \models_{cwa} \phi\} \;\; \sigma' \models_{cwa} \kappa}{\langle \sigma, \gamma, \{\alpha; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma', \gamma', \{\ \}\rangle}$$

Transitions: plans

- conditional choice

$$\frac{\sigma \models_{cwa} \phi}{\langle \sigma, \gamma, \{(\texttt{if}\ \phi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2); \pi \rhd \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi_1; \pi \rhd \kappa\}\rangle}$$

$$\frac{\sigma \not\models_{cwa} \phi}{\langle \sigma, \gamma, \{(\texttt{if}\ \phi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2); \pi \rhd \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi_2; \pi \rhd \kappa\}\rangle}$$

- conditional iteration

$$\frac{\sigma \models_{cwa} \phi}{\langle \sigma, \gamma, \{(\texttt{while}\ \phi\ \texttt{do}\ \pi_1); \pi \rhd \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi_1; (\texttt{while}\ \phi\ \texttt{do}\ \pi_1); \pi \rhd \kappa}$$

$$\frac{\sigma \not\models_{cwa} \phi}{\langle \sigma, \gamma, \{(\texttt{while}\ \phi\ \texttt{do}\ \pi_1 \rhd \kappa); \pi\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi \rhd \kappa\}\rangle}$$

Transitions: PG rules

- planning goal rules $\kappa \leftarrow \beta \mid \pi$

$$\frac{\gamma \models_{g} \kappa \quad \sigma_{cwa} \models \beta}{\langle \sigma, \gamma, \{\} \rangle \longrightarrow \langle \sigma, \gamma, \{\pi \triangleright \kappa\} \rangle}$$

Transitions: PR rules

- plan revision rules $p_j = \pi_j \leftarrow \beta_j \,|\, \pi'_j$

$$\frac{\forall i \ \sigma \not\models_{cwa} \mathtt{prec}_i(\alpha) \qquad \sigma \models_{cwa} \beta_j}{\langle \sigma, \gamma, \{\pi_j = \alpha; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi'_j \triangleright \kappa\}\rangle}$$

$$\frac{\sigma \not\models_{cwa} \beta \qquad\qquad \sigma \models_{cwa} \beta_j}{\langle \sigma, \gamma, \{\pi_j = \beta?; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi'_j \triangleright \kappa\}\rangle}$$

$$\frac{\sigma \models_{cwa} \beta_j}{\langle \sigma, \gamma, \{\pi_j = \bar{\alpha}; \pi \triangleright \kappa\}\rangle \longrightarrow \langle \sigma, \gamma, \{\pi'_j; \pi \triangleright \kappa\}\rangle}$$

where $\bar{\alpha}$ is the name of an abstract plan.

State of the art in model-checking agent programs

- Model-checking AgentSpeak (Promela, Spin)
  Rafael H. Bordini, Michael Fisher, Carmen Pardavila, Michael Wooldridge: Model checking AgentSpeak. AAMAS 2003:409-416
- General platform for model-checking BDI agents (AIL and AJPF)
  Louise A. Dennis, Michael Fisher, Matthew P. Webster, Rafael H. Bordini: Model checking agent programming languages. Autom. Softw. Eng. 19(1): 5-63 (2012)
- Work with Goal, 3/2APL,...

Challenges

- In common with general model-checking: scalability issues
- In common with general (software) model-checking: hard to deal with an infinite number of possible inputs/events, first-order properties
- I think there is still no system specification language at the right level of abstraction
- Beliefs, goals, plans, etc. are treated as just ordinary data structures: same as lists of strings or some other ‘dumb’ values
- However, they do have some logical structure (e.g. closure under the agent’s reasoning rules) and connections to each other, which should be used, in a transparent fashion (use something more like Maude?)
- The most interesting logical challenge here I think is the logic of having committed to a set of intentions

What does having a set of intentions mean

- If an agent's set of intentions is $\{a; b; c, \ d; e; f\}$ then it is easy to figure out what the possible actions by the agent are (a and d); for more general plans it is more complicated, but also well defined
- no logic with explicit adopted plans (in the logical language), apart from TCS11 (for single agent/single plan) and a paper in informal proceedings of DALT 2009.
- there are logics with explicit strategies (Simon and Ramanujam 2008,2009), but strategies and plans are not exactly the same and logics have no 'he *has adopted* this strategy' operator

Verification by theorem proving

- State properties of the system as axioms (completely axiomatise the operational semantics)
- Prove that the desired property logically follows from them
- This is a more complex problem than model-checking, but it is easier to deal with first-order, infinite domains, etc.

Signature of an agent program

- The signature of an agent program R is defined as $R = \langle \mathcal{P}$, PG, PR, Ac, $\bar{Ac}$, Act, $Plan\rangle$
  - $\mathcal{P}$ is a set of belief and goal atoms
  - PG is a set of planning goal rules, $r_i = \kappa_i \leftarrow \beta_i \mid \pi_i$
  - PR is a set of plan revision rules, $p_j = \pi_j \leftarrow \beta_j \mid \pi'_j$
  - *Ac* is a set of belief update actions occurring in the plans of PG and PR rules
  - $\bar{Ac}$ is a set of abstract plans occurring in the plans of PG and PR rules
  - Act is the set of specifications for belief update actions *Ac*
  - *Plan* is the set of all possible $\pi \triangleright \kappa$ pairs where κ is one of the agent's goals and π is a plan occurring in PG and PR rules or a suffix of such a plan

Language of PDL-3APL

- program expressions:

$$\rho ::= \alpha \in Ac \mid t(\phi) \mid \bar{a} \in \bar{A}c \mid \delta_{ri} \mid \delta_{p_j} \mid \rho_1 ; \rho_2 \mid \rho_1 \cup \rho_2 \mid \rho^*$$

- formula:

$$\psi ::= Bp \mid Gp \mid G{-}p \mid \; x \mid P^{\kappa}\pi \mid P\epsilon \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \langle\rho\rangle\psi$$

Models of PDL-3APL

Let $R = \langle \mathcal{P}, \text{PG}, \text{PR}, \text{Ac}, \bar{A}c, \text{Act}, Plan \rangle$ be the signature of an agent program. A PDL-3APL model M relative to R is defined as

$$M = (W, V, \mathcal{R}_\alpha, \mathcal{R}_{t(\phi)}, \mathcal{R}_{\bar{\alpha}}, \mathcal{R}_{\delta_{r_i}}, \mathcal{R}_{\delta_{p_j}})$$

where

- W is a non-empty set of states.
- $V = (V_b, V_g, V_c, V_p)$ such that for every $s \in W$:
  - $V_b(s) = \{p_1, \ldots, p_m : p_i \in \mathcal{P}\}$ is the set of the agent's beliefs in s;
  - $V_g(s) = \{(-)u_1, \ldots, (-)u_n : u_i \in \mathcal{P}\}$ is the set of the agent's goals in s (note that V_g assigns literals rather than propositional variables);
  - $V_c(s)$ is either an empty set or $\{x\}$;
  - $V_p(s)$ is either the empty set or a singleton set $\{\pi \triangleright \kappa\}$, where π is the agent's plan in s and κ is the goal(s) achieved by this plan
- $\mathcal{R}_\alpha$, $\mathcal{R}_{t(\phi)}$, $\mathcal{R}_{\bar{\alpha}}$, $\mathcal{R}_{\delta_{r_i}}$, $\mathcal{R}_{\delta_{p_i}}$ are binary relations on W

Conditions on models

- **C1** $V_g(s) \cap V_b(s) = \emptyset$ and $\{p : -p \in V_g(s)\} \subseteq V_b(s)$
- **C2** If $V_p(s) = \{\alpha; \pi \triangleright \kappa\}$, $V_b(s) \models_{cwa} \texttt{prec}_i(\alpha)$ and $x \notin V_c(s)$, then there is an R_α transition to a state s' where $V_b(s') = T_{i,j}(\alpha, V_b(s))$, $V_g(s') = V_g(s) \setminus (\{p : p \in V_b(s')\} \cup \{-p : p \notin V_b(s')\})$ and if $V_b(s') \not\models_{cwa} \kappa$, $V_p(s') = \{\pi \triangleright \kappa\}$.
  If $V_b(s') \models_{cwa} \kappa$, $x \in V_c(s')$ and $V_p(s') = \{\}$.
- **C3–C10** similarly correspond to operational semantics in non-x states

Conditions for exceptional states

- Condition for non-executable actions: if $V_p(s) = \{\alpha; \pi \triangleright \kappa\}$, $V_b(s) \not\models_{cwa} \texttt{prec}_i(\alpha)$, and $x \notin V_c(s)$, then there is an R_α transition to a state s' where $x \in V_c(s')$.
- Condition for executing in exceptional states: if $x \in V_c(s)$ then there are R_α, $R_{\bar{\alpha}}$ and $R_{t(\phi)}$ transitions from state s to itself
- Condition for PR rules: if $x \in V_c(s)$, $V_p(s) = \{\pi_j \triangleright \kappa\}$, $V_b(s) \models_{cwa} \beta_j$, then there is a $R_{\delta_{p_j}}$ transition to a state s' where $V_p(s') = \{\pi'_j \triangleright \kappa\}$ and $x \notin V_c(s')$ (where $p_j = \pi_j \leftarrow \beta_j \mid \pi'_j$).

Satisfaction

- $M, s \models Bp$ iff $p \in V_b(s)$
- $M, s \models Gp$ iff $p \in V_g(s)$
- $M, s \models G{-}p$ iff $-p \in V_g(s)$
- $M, s \models x$ iff $x \in V_c(s)$
- $M, s \models P^{\kappa}\pi$ iff $V_p(s) = \{\pi \triangleright \kappa\}$
- $M, s \models P\epsilon$ iff $V_p(s) = \{\}$
- $M, s \models \neg\psi$ iff $M, s \not\models \psi$
- $M, s \models \psi_1 \wedge \psi_2$ iff $M, s \models \psi_1$ and $M, s \models \psi_2$
- $M, s \models \langle\rho\rangle\psi$ iff there exists s' such that $R_\rho(s, s')$ and $M, s' \models \psi$.

Translation into PDL

- f_b: $f_b(p) = Bp$; $f_b(\phi \texttt{ and } \psi) = f_b(\phi) \wedge f_b(\psi)$; $f_b(\phi \texttt{ or } \psi) = f_b(\phi) \vee f_b(\psi)$
- $f_g(p) = Gp$; $f_g(-p) = G-p$
- f_p:
  - $f_p(\alpha) = \alpha$
  - $f_p(\phi?) = t(\phi)$
  - $f_p(\bar{\alpha}) = \bar{\alpha}$
  - $f_p(\pi_1; \pi_2) = f_p(\pi_1); f_p(\pi_2)$
  - $f_p(\texttt{if } \phi \texttt{ then } \pi_1 \texttt{ else } \pi_2) = t(\phi); f_p(\pi_1)) \cup (t(\neg\phi); f_p(\pi_2))$
  - $f_p(\texttt{while } \phi \texttt{ do } \pi) = (t(\phi); f_p(\pi))^*; t(\neg\phi)$.

Axioms

- A1 $Bp \to \neg Gp$
- A2 $G{-}p \to Bp$
- A3a $P^{\kappa}\pi \to \neg P^{\kappa'}\pi'$ where $\pi' \neq \pi$ or $\kappa' \neq \kappa$
- A3b $P\epsilon \vee \bigvee_{\pi \rhd \kappa \in Plan} P^{\kappa}\pi$
- BA1 $\neg x \wedge P^{\kappa}(\alpha;\pi) \wedge f_b(\mathtt{prec}_i(\alpha)) \wedge \psi \wedge \psi' \to \langle\alpha\rangle($ $(f_b(\mathtt{post}_{ij}(\alpha)) \wedge \neg f_b(\kappa) \wedge P^{\kappa}\pi \wedge \psi) \vee (f_b(\mathtt{post}_{ij}(\alpha)) \wedge f_b(\kappa) \wedge x \wedge P\epsilon \wedge \psi'))$ where ψ, ψ' are any formulas not containing plan expressions or literals in $f_b(\mathtt{post}_{ij}(\alpha))$, and in addition ψ' does not contain x
- BA2a $\neg x \wedge P^{\kappa}\pi \to [u]\bot$ where $\pi \neq u;\pi'$ and $u \in Ac \cup \bar{A}c$
- BA2b $\neg x \wedge P^{\kappa}\pi \to [t(\phi)]\bot$ if π does not start with a belief test action ϕ? or a conditional plan test on ψ where $\phi = \psi$ or $\phi = \neg\psi$

Axioms continued

BA3 $\neg x \wedge P^{\kappa}(\alpha;\pi) \wedge f_b(\mathtt{prec}_i(\alpha)) \wedge \bigwedge_j \psi_j \wedge \bigwedge_j \psi'_j \rightarrow [\alpha]($
$\bigvee_j(f_b(\mathtt{post}_{ij}(\alpha)) \wedge \neg f_b(\kappa) \wedge P^{\kappa}\pi \wedge \psi_j) \vee$
$\bigvee_j(f_b(\mathtt{post}_{ij}(\alpha)) \wedge f_b(\kappa) \wedge x \wedge P\epsilon \wedge \psi'_j))$
where ψ_j and ψ'_j are any formulas not containing plan expressions or literals in $f_b(\mathtt{post}_{ij}(\alpha))$, and in addition ψ'_j does not contain x

BA4 $\neg x \wedge P^{\kappa}(\phi?;\pi) \wedge f_b(\phi) \wedge \psi_{np} \rightarrow \langle[t(\phi)]\rangle(P^{\kappa}\pi \wedge \psi_{np})$

BA5 $\neg x \wedge P^{\kappa}(\alpha;\pi) \wedge \bigwedge_i \neg f_b(\mathtt{prec}_i(\alpha)) \wedge \psi_{nx} \rightarrow \langle[\alpha]\rangle(x \wedge \psi_{nx})$

BA6 $\neg x \wedge P^{\kappa}(\phi?;\pi) \wedge \neg f_b(\phi) \wedge \psi_{nx} \rightarrow \langle[t(\phi)]\rangle(x \wedge \psi_{nx})$

BA7 $\neg x \wedge P^{\kappa}(\bar{\alpha};\pi) \wedge \psi_{nx} \rightarrow \langle[\bar{\alpha}]\rangle(x \wedge \psi_{nx})$

BA8 $x \wedge \psi \rightarrow \langle[u]\rangle\psi$ where u is α, $t(\phi)$ or $\bar{\alpha}$

Axioms continued

CP1 $\neg x \wedge P^{\kappa}(\pi_{if};\pi) \wedge f_b(\phi) \wedge \psi_{np} \rightarrow \langle[t(\phi)]\rangle(P^{\kappa}\pi_1;\pi \wedge \psi_{np})$, where π_{if} is of the form $\texttt{if}\ \phi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2$

CP2 $\neg x \wedge P^{\kappa}(\pi_{if};\pi) \wedge \neg f_b(\phi) \wedge \psi_{np} \rightarrow \langle[t(\neg\phi)]\rangle(P^{\kappa}\pi_2;\pi \wedge \psi_{np})$, where π_{if} is as in **CP1**

CP3 $\neg x \wedge P^{\kappa}(\pi_{wh};\pi) \wedge f_b(\phi) \wedge \psi_{np} \rightarrow \langle[t(\phi)]\rangle(P^{\kappa}\pi_1;\pi_{wh};\pi \wedge \psi_{np})$, where π_{wh} is of the form $\texttt{while}\ \phi\ \texttt{do}\ \pi_1$

CP4 $\neg x \wedge P^{\kappa}(\pi_{wh};\pi) \wedge \neg f_b(\phi) \wedge \psi_{np} \rightarrow \langle[t(\neg\phi)]\rangle(P^{\kappa}\pi \wedge \psi_{np})$, where π_{wh} is as in **CP3**

CP5 $\neg x \wedge (P^{\kappa}\pi_{if} \vee P^{\kappa}\pi_{wh}) \wedge \neg f_b(\phi) \rightarrow [t(\phi)]\perp$ where π_{if} and π_{wh} are as above

PG1 $P\epsilon \wedge f_g(\kappa_i) \wedge f_b(\beta_i) \wedge \psi_{npx} \rightarrow \langle[\delta_{r\,i}]\rangle(\neg x \wedge P^{\kappa_i}\pi_i \wedge \psi_{npx})$

PG2 $\neg P\epsilon \vee \neg f_g(\kappa_i) \vee \neg f_b(\beta_i) \rightarrow [\delta_{r\,i}]\perp$

PR1 $x \wedge P^{\kappa}\pi_j \wedge f_b(\beta_j) \wedge \psi_{npx} \rightarrow \langle[\delta_{p_j}]\rangle(\neg x \wedge P^{\kappa}\pi'_j \wedge \psi_{npx})$

PR2 $\neg x \vee \neg P^{\kappa}\pi_j \vee \neg f_b(\beta_j) \rightarrow [\delta_{p_j}]\perp$

Translation of the program

- $tr(R) = (\cup_i(\delta_{ri}; f_p(\pi_i)) \bigcup \cup_j(\delta_{p_j}; f_p(\pi'_j)))^+$
- Theorem: $tr(R)$ picks out exactly those paths in a model which correspond to an execution of the program
- Can verify liveness and safety properties by checking whether $\langle tr(R) \rangle \phi$ and $[tr(R)]\phi$ are entailed by the formulas describing initial conditions
- complications: encoding plan expressions; encoding properties which hold along a path (Fahad Khan 2012, Regular Path Temporal Logic)

Conclusions

- agent programs can be verified just as ordinary programs
- however they have additional properties which it may be possible to expoit
- one of the properties is having an explicit set of plans, which seems to be an interesting logical property
- may be also of interest for game logics (being able to say ‘this player is going to play this strategy’ rather than ‘if this player plays this strategy’)