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

Plan(a;b)

a

b

p

Natasha Alechina

Reasoning about plan revision

TIME 2012
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
one of the first agent programming languages PRS (Georgeff and Ingrand 1988), very rich. I will talk about a more modern and less rich language, 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 \( \neg p \) 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 $\phi$? tests whether a boolean belief expression $\phi$ is entailed by the agent’s beliefs, e.g.:

  $$(\text{room2 and } \neg \text{battery})?$$

  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.:
  \[
  \{\text{room1}\} \text{ moveR } \{\} \text{, } \{-\text{room1, room2}\}
  \]

- 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 $\neg p$ 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 $\pi$ is an abstract plan $\bar{\alpha}$, execution of $\pi$ blocks.
plans are sequences of basic actions and atomic plans composed by plan composition operators:

- sequence: “$\pi_1; \pi_2$” (do $\pi_1$ then $\pi_2$)
- conditional choice: “if $\phi$ then {$\pi_1$} else {$\pi_2$}”
- conditional iteration: “while $\phi$ do {$\pi$}”

e.g., the plan:

if room1 then {suck} else {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 $\kappa \leftarrow \beta | \pi$ consists of three parts:
  - $\kappa$: an (optional) *goal query* which specifies which goal(s) the plan achieves.
  - $\beta$: a *belief query* which characterises the situation(s) in which it could be a good idea to execute the plan.
  - $\pi$: a plan.

- A PG rule can be applied if $\kappa$ is entailed by the agent’s goals and $\beta$ is entailed by the agent’s beliefs.

- Applying the rule adds $\pi$ to the agent’s plans.
Example 3APL PG rules

- \[ \text{clean2} \leftarrow \text{battery} \mid \]
  - \[ \text{if room2 then } \{ \text{suck} \} \text{ else } \{ \text{moveR; suck} \} \]

states that “if the agent’s goal is to clean \text{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:

  \[ \leftarrow -\text{battery} \mid \]
  - \[ \text{if room2 then } \{ \text{charge} \} \text{ else } \{ \text{moveR; charge} \} \]

states “if the battery is low, the specified plan may be used to charge it”
Example 3APL PR rules

- a plan revision rule \( p_j = \pi_j \leftarrow \beta_j \mid \pi'_j \) can be applied if \( \pi_j \) is in the plan base, \( \beta_j \) is entailed by the agent’s beliefs and \( \pi_j \) is not executable,

- in other words the first action of \( \pi_j \) 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 \textit{moveR} fails, the agent may execute a slow but reliable version of the action, \textit{slowR}:

  
  \[
  \text{charge} \leftarrow \text{room1} \mid \{ \text{slowR}; \text{charge} \}
  \]
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 \( \sigma, \gamma \) are sets of literals representing the agent’s beliefs and goals, and \( \Pi \) 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

| |=_{cwa} (belief entailment for closed world assumption):

\[
\sigma \models_{cwa} p \iff p \in \sigma \\
\sigma \models_{cwa} \neg p \iff p \not\in \sigma \\
\sigma \models_{cwa} \phi \land \psi \iff \sigma \models_{cwa} \phi \land \sigma \models_{cwa} \psi \\
\sigma \models_{cwa} \phi \lor \psi \iff \sigma \models_{cwa} \phi \lor \sigma \models_{cwa} \psi \\
\sigma \models_{cwa} \{\phi_1, \ldots, \phi_n\} \iff \forall 1 \leq i \leq n \ \sigma \models_{cwa} \phi_i
\]

| |=_{g} (goal entailment):

\[
\gamma \models_{g} p \iff p \in \gamma \\
\gamma \models_{g} \neg p \iff \neg p \in \gamma \\
\gamma \models_{g} \phi \lor \psi \iff \gamma \models_{g} \phi \lor \gamma \models_{g} \psi
\]
Belief update function

- let $a$ be a belief update action and $\sigma$ a belief base such that $\sigma \models_{cwa} \text{prec}_j(a)$

- intuitively, $\sigma \models_{cwa} \text{prec}_j(a)$ if it contains all positive literals in $\text{prec}_j(a)$ and does not contain the negative ones

- the result of executing belief update action $a$ with respect to $\sigma$ (assuming $\text{prec}_j(a)$ holds and the action results in the $\text{post}_{j,i}$ becoming true) is defined as:

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

- intuitively, the result of the update satisfies (entails under $\models_{cwa}$) the corresponding postcondition $\text{post}_{j,i}(a)$
Transitions: belief test actions

belief test actions

\[ \sigma \models \text{cwa} \beta \]

\[ \langle \sigma, \gamma, \{ \beta?; \pi \triangleright \kappa \} \rangle \rightarrow \langle \sigma, \gamma, \{ \pi \triangleright \kappa \} \rangle \]
Transitions: belief update actions

belief update actions when the corresponding goal not achieved yet:

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

belief update actions when the corresponding goal is achieved:

\[
\sigma \models_{\text{cwa prec}} \alpha \quad T_{i,j}(\alpha, \sigma) = \sigma' \quad \gamma' = \gamma \setminus \{ \phi \mid \sigma' \models_{\text{cwa}} \phi \} \quad \sigma' \models_{\text{cwa}} \kappa \\
\langle \sigma, \gamma, \{ \alpha; \pi \triangleright \kappa \} \rangle \quad \longrightarrow \quad \langle \sigma', \gamma', \{ \} \rangle
\]
Transitions: plans

- **conditional choice**

  \[\sigma \models_{cwa} \phi\]

  \[
  \langle \sigma, \gamma, \{(\text{if } \phi \text{ then } \pi_1 \text{ else } \pi_2); \pi \triangleright \kappa\} \rangle \longrightarrow \langle \sigma, \gamma, \{\pi_1; \pi \triangleright \kappa\} \rangle
  \]

  \[\sigma \not\models_{cwa} \phi\]

  \[
  \langle \sigma, \gamma, \{(\text{if } \phi \text{ then } \pi_1 \text{ else } \pi_2); \pi \triangleright \kappa\} \rangle \longrightarrow \langle \sigma, \gamma, \{\pi_2; \pi \triangleright \kappa\} \rangle
  \]

- **conditional iteration**

  \[\sigma \models_{cwa} \phi\]

  \[
  \langle \sigma, \gamma, \{(\text{while } \phi \text{ do } \pi_1); \pi \triangleright \kappa\} \rangle \longrightarrow \langle \sigma, \gamma, \{\pi_1; (\text{while } \phi \text{ do } \pi_1); \pi \triangleright \kappa\} \rangle
  \]

  \[\sigma \not\models_{cwa} \phi\]

  \[
  \langle \sigma, \gamma, \{(\text{while } \phi \text{ do } \pi_1 \triangleright \kappa); \pi\} \rangle \longrightarrow \langle \sigma, \gamma, \{\pi \triangleright \kappa\} \rangle
  \]
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 $p_j = \pi_j \leftarrow \beta_j \mid \pi'_j$

\[
\forall i \sigma \not\models_{cwa} \text{prec}_i(\alpha) \quad \sigma \models_{cwa} \beta_j
\]

\[
\langle \sigma, \gamma, \{ \pi_j = \alpha; \pi \triangleright \kappa \} \rangle \rightarrow \langle \sigma, \gamma, \{ \pi'_j \triangleright \kappa \} \rangle
\]

\[
\sigma \not\models_{cwa} \beta \quad \sigma \models_{cwa} \beta_j
\]

\[
\langle \sigma, \gamma, \{ \pi_j = \beta?; \pi \triangleright \kappa \} \rangle \rightarrow \langle \sigma, \gamma, \{ \pi'_j \triangleright \kappa \} \rangle
\]

\[
\sigma \models_{cwa} \beta_j
\]

\[
\langle \sigma, \gamma, \{ \pi_j = \bar{\alpha}; \pi \triangleright \kappa \} \rangle \rightarrow \langle \sigma, \gamma, \{ \pi'_j; \pi \triangleright \kappa \} \rangle
\]

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)

- 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 $R = \langle \mathcal{P}, \text{PG}, \text{PR}, \text{Ac}, \tilde{\text{Ac}}, \text{Act}, \text{Plan} \rangle$

- $\mathcal{P}$ 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
- $\tilde{\text{Ac}}$ 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 $\kappa$ is one of the agent's goals and $\pi$ is a plan occurring in PG and PR rules or a suffix of such a plan
Language of PDL-3APL

- program expressions:
  \[ \rho ::= \alpha \in \text{Ac} \mid t(\phi) \mid \bar{a} \in \bar{\text{Ac}} \mid \delta_{ri} \mid \delta_{pj} \mid \rho_1; \rho_2 \mid \rho_1 \cup \rho_2 \mid \rho^* \]

- formula:
  \[ \psi ::= Bp \mid Gp \mid G \neg p \mid x \mid P^x \pi \mid P \epsilon \mid \neg \psi \mid \psi_1 \land \psi_2 \mid \langle \rho \rangle \psi \]
Models of PDL-3APL

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

$$M = (W, V, R_\alpha, R_t(\phi), R\bar{\alpha}, R_{\delta_i}, R_{\delta_p})$$

where

- $W$ is a non-empty set of states.
- $V = (V_b, V_g, V_c, V_p)$ such that for every $s \in W$:
  - $V_b(s) = \{p_1, \ldots, p_m : p_i \in P\}$ is the set of the agent’s beliefs in $s$;
  - $V_g(s) = \{(-)u_1, \ldots, (-)u_n : u_i \in P\}$ 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 $\pi$ is the agent’s plan in $s$ and $\kappa$ is the goal(s) achieved by this plan.
- $R_\alpha, R_t(\phi), R\bar{\alpha}, R_{\delta_i}, R_{\delta_p}$ are binary relations on $W$.
Conditions on models

- **C1** \( V_g(s) \cap V_b(s) = \emptyset \) and \( \{ p : \neg p \in V_g(s) \} \subseteq V_b(s) \)

- **C2** If \( V_p(s) = \{ \alpha; \pi \triangleright \kappa \} \), \( V_b(s) \models_{cwa} prec_i(\alpha) \) and \( x \notin V_c(s) \), then there is an \( R_\alpha \) transition to a state \( s' \) where \( V_b(s') = T_{i,j}(\alpha, V_b(s)) \), \( V_g(s') = V_g(s) \setminus (\{ p : p \in V_b(s') \} \cup \{ \neg p : p \notin V_b(s') \}) \) 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 $V_p(s) = \{\alpha; \pi \triangleright \kappa\}$, $V_b(s) \not\models_{\text{cwa}} \text{pre}_c(\alpha)$, and $x \notin V_c(s)$, then there is an $R_\alpha$ 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 $R_\alpha$, $R_{\bar{\alpha}}$ and $R_t(\phi)$ transitions from state $s$ to itself.

- **Condition for PR rules:** if $x \in V_c(s)$, $V_p(s) = \{\pi_j \triangleright \kappa\}$, $V_b(s) \models_{\text{cwa}} \beta_j$, 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 $p_j = \pi_j \leftarrow \beta_j | \pi'_j$).
Satisfaction

- \( M, s \models Bp \iff p \in V_b(s) \)
- \( M, s \models Gp \iff p \in V_g(s) \)
- \( M, s \models G\neg p \iff \neg p \in V_g(s) \)
- \( M, s \models x \iff x \in V_c(s) \)
- \( M, s \models P^\kappa \pi \iff V_p(s) = \{ \pi \triangleright \kappa \} \)
- \( M, s \models P\epsilon \iff V_p(s) = \{ \} \)
- \( M, s \models \neg \psi \iff M, s \not\models \psi \)
- \( M, s \models \psi_1 \land \psi_2 \iff M, s \models \psi_1 \) 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

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

A1 \( Bp \rightarrow \neg Gp \)

A2 \( G \neg p \rightarrow Bp \)

A3a \( P^{\kappa \pi} \rightarrow \neg P^{\kappa' \pi'} \) where \( \pi' \neq \pi \) or \( \kappa' \neq \kappa \)

A3b \( P_{\epsilon} \lor \sqrt{\pi \kappa \in \text{Plan}} P^{\kappa \pi} \)

BA1 \( \neg x \land P^{\kappa}(\alpha; \pi) \land f_b(prec_i(\alpha)) \land \psi \land \psi' \rightarrow \langle \alpha \rangle \left( f_b(post_{ij}(\alpha)) \land \neg f_b(\kappa) \land P^{\kappa \pi} \land \psi \lor (f_b(post_{ij}(\alpha)) \land f_b(\kappa) \land x \land P_{\epsilon} \land \psi') \right) \)

where \( \psi, \psi' \) are any formulas not containing plan expressions or literals in \( f_b(post_{ij}(\alpha)) \), and in addition \( \psi' \) does not contain \( x \)

BA2a \( \neg x \land P^{\kappa \pi} \rightarrow [u] \bot \) where \( \pi \neq u; \pi' \) and \( u \in Ac \cup \bar{Ac} \)

BA2b \( \neg x \land P^{\kappa \pi} \rightarrow [t(\phi)] \bot \) if \( \pi \) does not start with a belief test action \( \phi \) or a conditional plan test on \( \psi \) where \( \phi = \psi \) or \( \phi = \neg \psi \)
Axioms continued

BA3 \( \neg x \land P^\kappa(\alpha; \pi) \land f_b(\text{prec}_i(\alpha)) \land \bigwedge_j \psi_j \land \bigwedge_j \psi'_j \rightarrow [\alpha](\bigvee_j (f_b(\text{post}_{ij}(\alpha)) \land \neg f_b(\kappa) \land P^\kappa\pi \land \psi_j)) \lor \bigvee_j (f_b(\text{post}_{ij}(\alpha)) \land f_b(\kappa) \land x \land P\epsilon \land \psi'_j)) \)

where \( \psi_j \) and \( \psi'_j \) are any formulas not containing plan expressions or literals in \( f_b(\text{post}_{ij}(\alpha)) \), and in addition \( \psi'_j \) does not contain \( x \)

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

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

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

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

BA8 \( x \land \psi \rightarrow \langle [u] \rangle \psi \) where \( u \) is \( \alpha \), \( t(\phi) \) or \( \bar{\alpha} \)
Axioms continued

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

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

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

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

CP5  \[ \neg x \land (P^\kappa \pi_{if} \lor P^\kappa \pi_{wh}) \land \neg f_b(\phi) \rightarrow [t(\phi)] \perp \text{ where } \pi_{if} \text{ and } \pi_{wh} \text{ are as above} \]

PG1  \[ P_{\epsilon} \land f_g(\kappa_i) \land f_b(\beta_i) \land \psi_{np} \rightarrow \langle [\delta_{ri}] \rangle (\neg x \land P^{\kappa_i} \pi_i \land \psi_{np}) \]

PG2  \[ \neg P_{\epsilon} \lor \neg f_g(\kappa_i) \lor \neg f_b(\beta_i) \rightarrow [\delta_{ri}] \perp \]

PR1  \[ x \land P^\kappa \pi_j \land f_b(\beta_j) \land \psi_{np} \rightarrow \langle [\delta_{pj}] \rangle (\neg x \land P^\kappa \pi_j' \land \psi_{np}) \]

PR2  \[ \neg x \lor \neg P^\kappa \pi_j \lor \neg f_b(\beta_j) \rightarrow [\delta_{pj}] \perp \]
Translation of the program

- \( \text{tr}(R) = (\bigcup_i (\delta_{ri}; f_p(\pi_i)) \cup \bigcup_j (\delta_{pj}; f_p(\pi'_j)))^+ \)

- **Theorem:** \( \text{tr}(R) \) 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 \( \langle \text{tr}(R) \rangle \phi \) and \( [\text{tr}(R)]\phi \) 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 exploit.
- 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’).