An Operational Semantics for a Fragment of PRS

IJCAI 2018
Stockholm, Sweden

Lavindra de Silva¹  Felipe Meneguzzi²  Brian Logan³

¹University of Nottingham, UK
lavindra.desilva@nottingham.ac.uk

²Pontifical Catholic University of Rio Grande do Sul, Brazil
felipe.meneguzzi@pucrs.br

³University of Nottingham, Nottingham, UK
brian.logan@nottingham.ac.uk

July 18, 2018
Motivation

Syntax

Semantics

Key Properties

Discussion
Motivation

- PRS is a seminal reasoning system:
  - it is one of the first practical implementations of BDI systems;
  - it is widely used in robotics today;
  - it influenced most subsequent agent programming languages;
    - agents community generally believe it to be more expressive; yet
    - no precise formalisation of the language.

- We aim to fill these gaps to allow comparison of PRS with its successors

  CAN AgentSpeak Golog X-BDI JACK dMARS
We formalise a significant fragment of PRS graph-based plan bodies; language constructs to wait for and preserve maintenance goals, reasoning rules to operationalise such constructs, including:

- adopt, suspend, resume, and abort possibly nested goals

We use the formalisation to prove key properties of PRS most importantly:

- CAN style plan-rules can be directly translated to PRS graph notation
- PRS plan-body graphs cannot be directly translated to CAN
Agent Structure

- Belief base ($\mathcal{B}$)
- Action-library ($\Lambda$) containing actions:
  - $\text{act}(\vec{v}) : \psi \leftarrow \Phi^+ ; \Phi^-$
  - STRIPS style action-rules with precondition and positive/negative effects
- Plan-library ($\Pi$) containing plan-rules
  - $e(t) : \varphi ; \psi \leftarrow G$
  - Plan-rules contain three key parts:
    - an event-goal $e(t)$ – stating when the plan is relevant
    - an optional goal-condition $\varphi$ – describing what the plan achieves
    - a context condition $\psi$ – describing when the plan is applicable
    - a plan-body graph $G$ – what the agent executes
Plan-body graphs

Plan body-graphs comprise two key structures:

- **User programs**, including:
  - Actions (from the action-library)
  - Belief addition/removal ($+b, -b$)
  - Tests ($?\phi$)
  - Event-goal or goal-condition programs ($!ev$, or $!\phi$)
  - Wait ($WT(\phi)$)
  - Passive or active preserve ($PR_p(!ev, \phi)$, or $PR_a(!ev, \phi)$)

- A directed bipartite graph split into:
  - State nodes
  - Transition nodes (labelled with programs)

![Diagram of a directed bipartite graph](walk(d))
Example Graphs

\[ G_{\text{walk}} \]
\[ \text{travelTo(dest)} : \text{At}(x) \land \text{WalkDist}(x, \text{dest}) \leftarrow G_{\text{walk}} \]

\[ G_{\text{pw}} \]

Within \( G_{\text{walk}} \), event \( !pw \) leads to executing the following rule:
\[ !pw : \top \leftarrow G_{\text{pw}} \]
Example

The agent has the following plan rules used to address the subgoal `travelTo(dest)` to go from the current location to the destination location `dest`:

\[
\begin{align*}
\text{travelTo}(\text{dest}) & : \text{At}(x) \land \text{WalkDist}(x, \text{dest}) \leftarrow G_{\text{walk}} \\
\text{travelTo}(\text{dest}) & : \text{At}(x) \land \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{dest}, y)) \leftarrow G_{\text{city}} \\
\text{travelTo}(\text{dest}) & : \text{At}(x) \land \neg \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{dest}, y)) \leftarrow G_{\text{far}} \\
\text{travelTo}(\text{dest}) & : \top \leftarrow G_{\text{home}}
\end{align*}
\]
Semantics of Plan-Body Graphs

Example

- Agent receives event: !travelTo(Uni)
Semantics of Plan-Body Graphs

Example

- Agent receives event: $!travelTo(Uni)$
- Current Plan: $!travelTo(Uni) : (\psi_1 : G_{walk}, \psi_2 : G_{city}, \psi_3 : G_{far})$, where:

  $\psi_1 = At(x) \land WalkDist(x, Uni)$
  $\psi_2 = At(x) \land \exists y (InCity(x, y) \land InCity(Uni, y))$
  $\psi_3 = At(x) \land \neg \exists y (InCity(x, y) \land InCity(Uni, y))$
Semantics of Plan-Body Graphs

Example

- Agent receives event: \(!\text{travelTo}(\text{Uni})\)
- Current Plan: \(!\text{travelTo}(\text{Uni}) : (\psi_1 : G_{\text{walk}}, \psi_2 : G_{\text{city}}, \psi_3 : G_{\text{far}})\), where:

\[
\begin{align*}
\psi_1 &= \text{At}(x) \land \text{WalkDist}(x, \text{Uni}) \\
\psi_2 &= \text{At}(x) \land \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y)) \\
\psi_3 &= \text{At}(x) \land \neg \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y))
\end{align*}
\]

\(G_{\text{walk}}\) – as stored in the Plan Library

![Diagram of plan-body graph]

\(walk(d)\)
Semantics of Plan-Body Graphs

Example

- Agent receives event: \( \text{travelTo}(\text{Uni}) \)
- Current Plan: \( \text{travelTo}(\text{Uni}) : (\psi_1 : G_{walk}, \psi_2 : G_{city}, \psi_3 : G_{far}) \),

where:

\[
\begin{align*}
\psi_1 &= \text{At}(x) \land \text{WalkDist}(x, \text{Uni}) \\
\psi_2 &= \text{At}(x) \land \exists y (\text{InCity}(x,y) \land \text{InCity}(\text{Uni},y)) \\
\psi_3 &= \text{At}(x) \land \neg \exists y (\text{InCity}(x,y) \land \text{InCity}(\text{Uni},y))
\end{align*}
\]

\( G_{walk} \) – when \( B \models \psi_1 \)

\( \text{walk}(\text{Uni}) \)

\( !pw \)  \( \text{At}(\text{Uni}) \)  \( ?s_4 \)
Semantics of Plan-Body Graphs

Example

- Agent receives event: ![travelTo](Uni)
- Current Plan: ![travelTo](Uni) : \{ψ₁ : G_{walk}, ψ₂ : G_{city}, ψ₃ : G_{far}\}

where:

\begin{align*}
ψ₁ &= At(x) ∧ WalkDist(x, Uni) \\
ψ₂ &= At(x) ∧ ∃y(InCity(x, y) ∧ InCity(Uni, y)) \\
ψ₃ &= At(x) ∧ ¬∃y(InCity(x, y) ∧ InCity(Uni, y))
\end{align*}

\[G_{walk} – transitioning to the !pw node\]

\[\text{walk}(Uni)\]
Semantics of Plan-Body Graphs

Example

- Agent receives event: \( !\text{travelTo}(\text{Uni}) \)
- Current Plan: \( !\text{travelTo}(\text{Uni}) : \{ \psi_1 : G_{\text{walk}}, \psi_2 : G_{\text{city}}, \psi_3 : G_{\text{far}} \} \), where:

\[
\psi_1 = \text{At}(x) \land \text{WalkDist}(x, \text{Uni}) \\
\psi_2 = \text{At}(x) \land \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y)) \\
\psi_3 = \text{At}(x) \land \neg \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y))
\]

- \( G_{\text{walk}} \) – executing sub-graph \( G_{\text{pw}} \)

\[
G_{\text{pw}} \triangleright \text{prepareWalk} : (\Delta_{\text{pw}}) \quad ?\text{At}(\text{Uni})
\]

\[
\begin{align*}
&s_0 \quad \xrightarrow{\ldots} \quad s_2 \quad \xrightarrow{\text{walk}(\text{Uni})} \quad s_3 \quad \xrightarrow{\ldots} \quad s_4
\end{align*}
\]
Semantics of Plan-Body Graphs

Example

- Agent receives event: $!\text{travelTo}(\text{Uni})$
- Current Plan: $!\text{travelTo}(\text{Uni}) : (\langle \psi_1 : G_{walk}, \psi_2 : G_{city}, \psi_3 : G_{far} \rangle)$, where:

$$
\psi_1 = \text{At}(x) \land \text{WalkDist}(x, \text{Uni})
$$

$$
\psi_2 = \text{At}(x) \land \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y))
$$

$$
\psi_3 = \text{At}(x) \land \neg \exists y (\text{InCity}(x, y) \land \text{InCity}(\text{Uni}, y))
$$

$\text{G}_{walk} – \text{after executing G}_{pw}$

$\eta$

$?\text{At}(\text{Uni})$

$s_0 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4$

$\text{walk( Uni)}$
Theorems 1-4 ensure that our fragment of PRS works, in summary:

- The semantics is sound: all valid transitions from valid states result in valid states
- Wait and preserve programs are complete:
  - They are only removed under the right conditions
Expressivity: CAN to PRS

Theorem

*If* $\Pi_c^-$ *is a CAN library and* $\Lambda$ *an action-library, there exists a PRS library* $\Pi_p$ *s.t. for any event-goal* $!e$ *and beliefs* $B$:

$$\text{SOL}(\Lambda, \Pi_c^-, B, \{!e\}) = \text{SOL}(\Lambda, \Pi_p, B, \{!e\}).$$

**Key result:** a CAN plan-library $\Pi_c^-$ not mentioning $\text{Goal}(\phi_s, P, \phi_f)$ programs (as there is no corresponding program in PRS) can be translated into an equivalent PRS plan-library.
Theorem

There exists a PRS library $\Pi_p^-$, an action-library $\Lambda$, and event-goal $!e$, s.t. for any CAN library $\Pi_c \in \text{CAN}(\Pi_p^-)$ and beliefs $B$:

$$\text{SOL}(\Lambda, \Pi_p^-, B, \{!e\}) \neq \text{SOL}(\Lambda, \Pi_c, B, \{!e\}).$$

**Key result:** the converse does not hold: even if we ignore programs that have no counterparts in CAN, some PRS plan-libraries cannot be ‘directly mapped’ to CAN libraries.
Example of unconvertible PRS Plan

The following non-series-parallel plan-body graph cannot be converted into a single CAN plan-body graph:

\[
ev0^1 \rightarrow ev0^2 \rightarrow ev1^1 \rightarrow ev2^1 \rightarrow ev2^2 \rightarrow ev5^1 \rightarrow ev1^2 \rightarrow ev4^1 \rightarrow ev4^2 \rightarrow ev5^2 \rightarrow ev6^1 \rightarrow ev6^2
\]
Future Work

- Translations of constructs from related work into PRS
  - van Riemsdijk et al. 2009
  - Dastani et al. 2011
  - Thangarajah et al. 2014

- Proofs to account for translating graph plan-bodies to sets of CAN or AgentSpeak plan-rules

- Extend the semantics to account for further PRS features:
  - Meta-level reasoning
  - Overlapping plan steps