An Operational Semantics for a Fragment of PRS

IJCAI 2018
Stockholm, Sweden

Lavindra de Silva [1] **Felipe Meneguzzi**[2] Brian Logan[3]

[1]University of Nottingham, UK
`lavindra.desilva@nottingham.ac.uk`
[2]Pontifical Catholic University of Rio Grande do Sul, Brazil
`felipe.meneguzzi@pucrs.br`
[3]University of Nottingham, Nottingham, UK
`brian.logan@nottingham.ac.uk`

July 18, 2018

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

Key Contribution

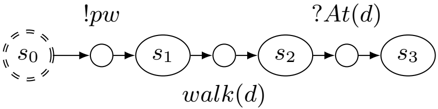
- We formalise a significant fragment of PRS
    - graph-based plan bodies;
    - language constructs to wait for and preserve maintenance goals,
    - reasoning rules to operationalise such constructs, including:
        - adopt, suspend, resume, and abort possibly nested goals
- We use the formalisation 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 (Λ) containing actions:
  - $act(\vec{v}){:}\psi \ \leftarrow \Phi^+; \Phi^-$
  - STRIPS style action-rules with precondition and positive/negative effects
- Plan-library (Π) containing plan-rules
  - $e(\vec{t}){:}\varphi \, ; \psi \ \leftarrow G$
  - Plan-rules contain three key parts:
    - an *event-goal* $e(\vec{t})$ – stating when the plan is relevant
      an optional goal-condition φ – describing what the plan achieves
    - a *context condition* ψ – 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 ($!\mathbf{ev}$, or $!\phi$)
    - wait ($\textsc{wt}(\phi)$)
    - passive or active preserve ($\textsc{pr}_p(!\textbf{ev}, \phi)$, or $\textsc{pr}_a(!\textbf{ev}, \phi)$)
  - a directed bipartite graph split into:
    - state nodes
    - transition nodes (labelled with programs)

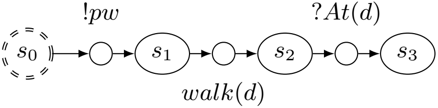


Example Graphs

G_{walk}

$travelTo(dest)$:
$At(x) \wedge WalkDist(x, dest) \leftarrow G_{walk}$

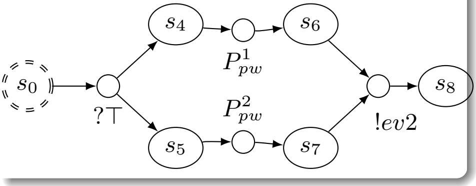
G_{walk}



G_{pw}

Within G_{walk}, event $!pw$ leads to executing the following rule:
$!pw : \top \leftarrow G_{pw}$

G_{pw}



Running Example

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 $dest$:

$$travelTo(dest) : At(x) \wedge WalkDist(x, dest) \leftarrow G_{walk}$$
$$travelTo(dest) : At(x) \wedge \exists y(InCity(x, y) \wedge InCity(dest, y)) \leftarrow G_{city}$$
$$travelTo(dest) : At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(dest, y)) \leftarrow G_{far}$$
$$travelTo(dest) : \top \leftarrow G_{home}$$

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:

$$\begin{aligned}\psi_1 =& At(x) \wedge WalkDist(x, Uni)\\ \psi_2 =& At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y))\\ \psi_3 =& At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))\end{aligned}$$

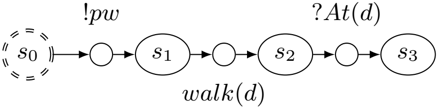
Semantics of Plan-Body Graphs

Example

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

$$\begin{aligned}\psi_1 =& At(x) \wedge WalkDist(x, Uni)\\ \psi_2 =& At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y))\\ \psi_3 =& At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))\end{aligned}$$

G_{walk} – as stored in the Plan Library



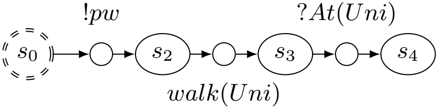
Semantics of Plan-Body Graphs

Example

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

$$\begin{aligned}\psi_1 =& At(x) \wedge WalkDist(x, Uni)\\ \psi_2 =& At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y))\\ \psi_3 =& At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))\end{aligned}$$

G_{walk} – when $\mathcal{B} \vdash \psi_1$



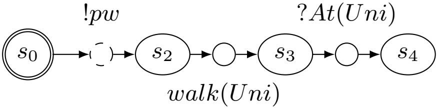
Semantics of Plan-Body Graphs

Example

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

$$\begin{aligned}\psi_1 =& At(x) \wedge WalkDist(x, Uni)\\ \psi_2 =& At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y))\\ \psi_3 =& At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))\end{aligned}$$

G_{walk} – transitioning to the $!pw$ node



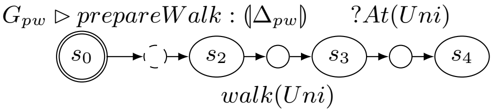
Semantics of Plan-Body Graphs

Example

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

$$\begin{aligned}
\psi_1 =& At(x) \wedge WalkDist(x, Uni) \\
\psi_2 =& At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y)) \\
\psi_3 =& At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))
\end{aligned}$$

G_{walk} – executing sub-graph G_{pw}



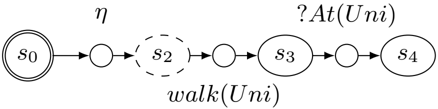
Semantics of Plan-Body Graphs

Example

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

$$\psi_1 = At(x) \wedge WalkDist(x, Uni)$$
$$\psi_2 = At(x) \wedge \exists y(InCity(x, y) \wedge InCity(Uni, y))$$
$$\psi_3 = At(x) \wedge \neg\exists y(InCity(x, y) \wedge InCity(Uni, y))$$

G_{walk} – after executing G_{pw}



Soundness and Completeness of the Semantics

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 Π_c^- *is a CAN library and* Λ *an action-library, there exists a PRS library* Π_p *s.t. for any event-goal* $!e$ *and beliefs* $\mathcal{B}$*:*
$\mathsf{SOL}(\Lambda, \Pi_c^-, \mathcal{B}, \{!e\}) = \mathsf{SOL}(\Lambda, \Pi_p, \mathcal{B}, \{!e\})$.

Key result: a CAN plan-library Π_c^- not mentioning Goal(ϕ_s, P, ϕ_f) programs (as there is no corresponding program in PRS) can be translated into an equivalent PRS plan-library.

Expressivity: PRS to CAN

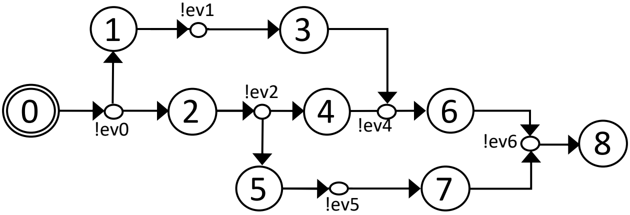
Theorem

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

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