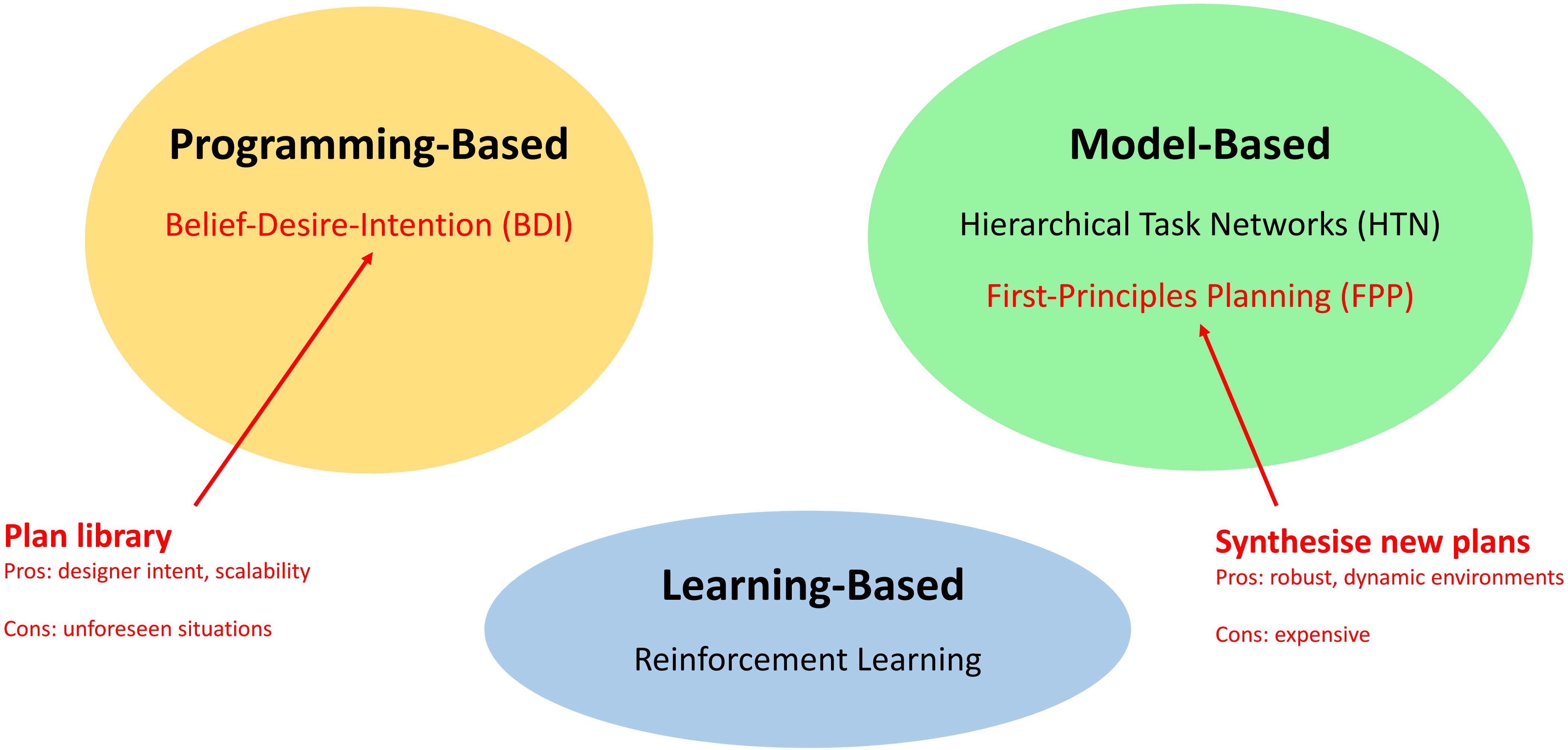
A Formal Approach to Embedding First-Principles Planning in BDI Agent Systems

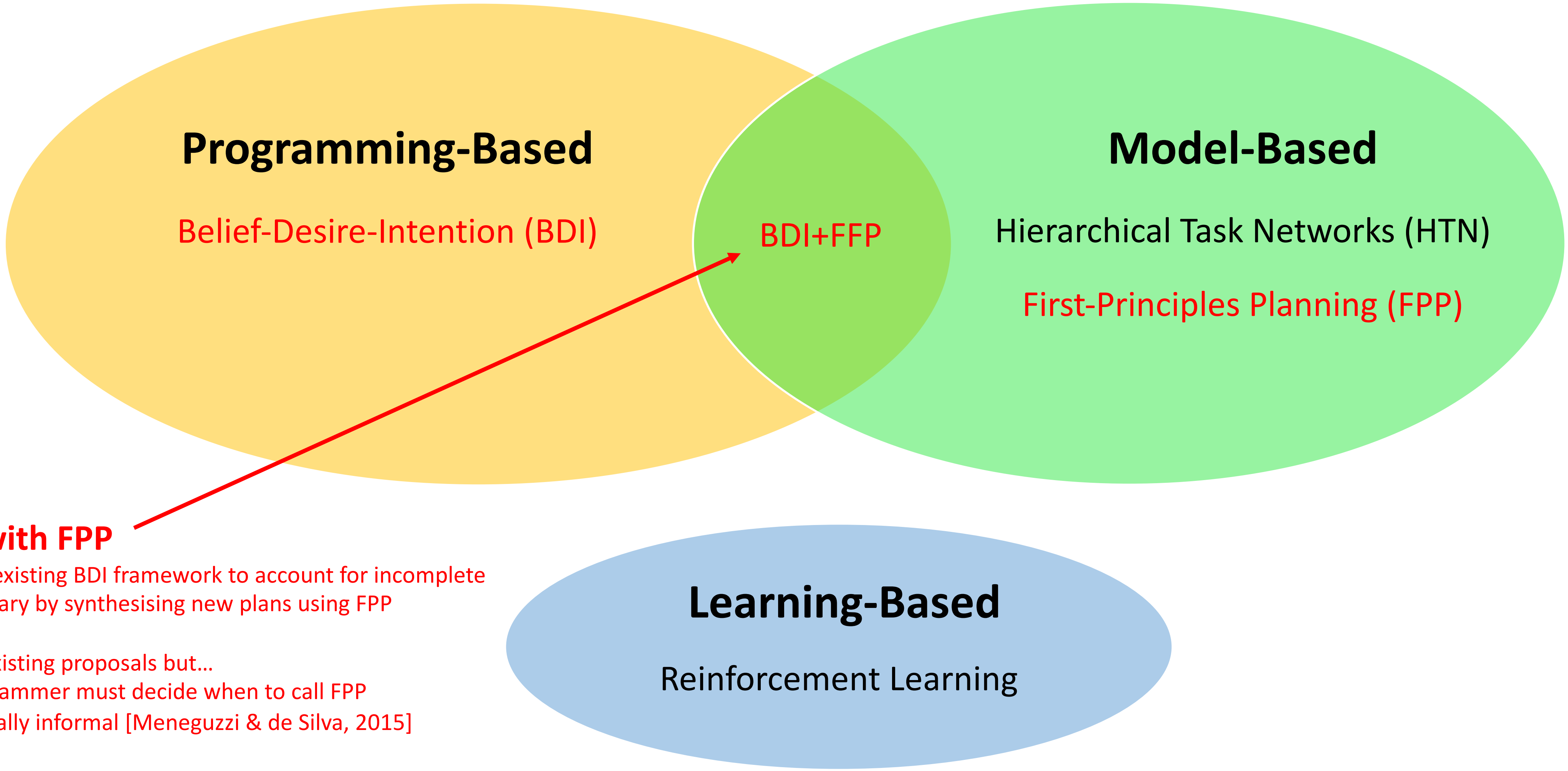
Mengwei Xu, Kim Bauters, **Kevin McAreavey**, Weiru Liu

University of BRISTOL

Agent Frameworks



Extending BDI with FPP



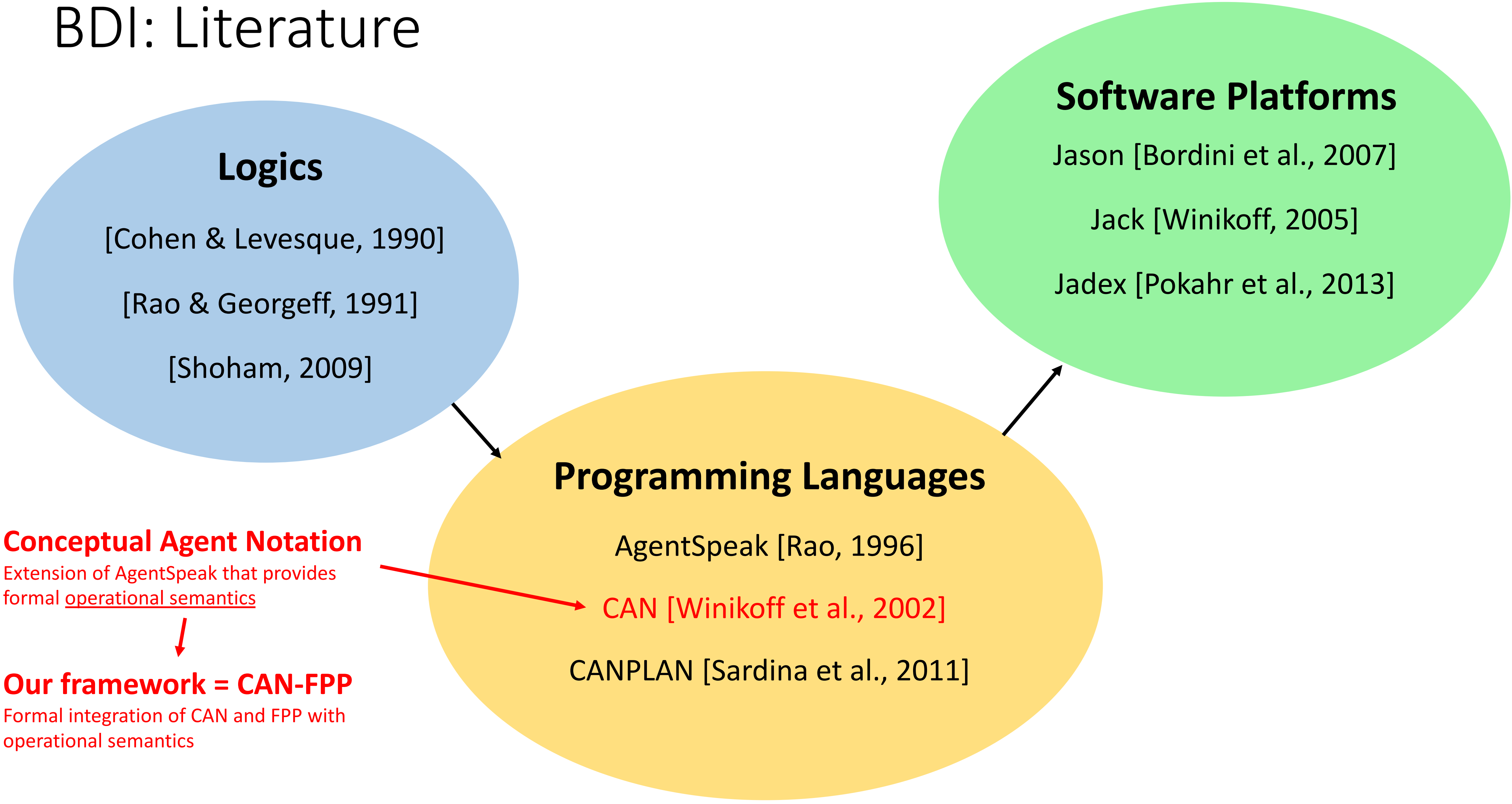
BDI with FPP

Extend existing BDI framework to account for incomplete plan library by synthesising new plans using FPP

Some existing proposals but…

- Programmer must decide when to call FPP
- Typically informal [Meneguzzi & de Silva, 2015]

BDI: Literature



CAN: Agent $(\mathcal{B}, \Lambda, \Pi)$

Initial belief base

Belief base specifying agent's initial beliefs

Plan library

Set of plan rules

Action description library

Set of STRIPS-style action descriptions

CAN: Agent $(\mathcal{B}, \Lambda, \Pi)$

Initial belief base

Belief base specifying agent's initial beliefs

Belief base $\mathcal{B} \subseteq \mathcal{L}$

Set of formulas from **logical language** $\mathcal{L}$

$\mathcal{B}$ must support:

- $\mathcal{B} \vDash \varphi$ (Entailment)
- $\mathcal{B} \cup \{\varphi\}$ (Addition)
- $\mathcal{B} \setminus \{\varphi\}$ (Deletion)

<u>Assume $\mathcal{B}$ is a set of atoms</u>

CAN: Agent $(\mathcal{B}, \Lambda, \Pi)$

Action description library

Set of STRIPS-style action descriptions

Action description $\mathrm{act} : \varphi \leftarrow \mathcal{B}^- ; \mathcal{B}^+$

Primitive action symbol

Precondition $\varphi \in \mathcal{L}$

Set of "delete" atoms $\mathcal{B}^- \subseteq \mathcal{L}$

Set of "add" atoms $\mathcal{B}^+ \subseteq \mathcal{L}$

CAN: Agent $(\mathcal{B}, \Lambda, \Pi)$

Plan library

Set of plan rules

Triggering event e

e.g. belief update, new (sub)goal

Context $\varphi \in \mathcal{L}$

Formula from $\mathcal{L}$

Plan rule $e : \varphi \leftarrow P$

Body (program) $P ::= \mathrm{nil} \mid \mathrm{act} \mid ?\varphi \mid +b \mid -b \mid !e \mid P_1 ; P_2 \mid P_1 \rhd P_2 \mid P_1 \parallel P_2 \mid (\!| \varphi_1 : P_1, \ldots, \varphi_n : P_n |\!) \mid \mathrm{goal}(\varphi_s, P, \varphi_f)$

Empty program

Primitive action

Entailment

Belief addition

Belief deletion

New (sub)goal

Sequencing

Conditional

Concurrency

Relevant plans

CAN declarative goal

CAN: Operational Semantics

Static entities that will be omitted

Basic configuration $(\mathcal{B}, \Lambda, \Pi, \mathcal{A}, P)$

Current belief base
Belief base specifying agent's current beliefs

Current intention
Remainder of a plan body to execute

Execution history $\mathcal{A} = (\mathrm{act}_1, \mathrm{act}_2, \ldots, \mathrm{act}_n)$
Sequence of primitive actions executed so far

Current intention set
Set of (partially executed) plan bodies

Agent configuration $(\mathcal{B}, \Lambda, \Pi, \mathcal{A}, \Gamma)$

CAN: Operational Semantics (Cont.)

Transition $\mathcal{C} \longrightarrow \mathcal{C}'$

Transition between (basic or agent) configurations

Derivation Rule $\dfrac{p_1 \quad p_2 \quad \dots \quad p_n}{c}\ l$

Specifies when/how an agent transitions to new configuration

$$\frac{}{(\mathcal{B}, \mathcal{A}, +b) \longrightarrow (\mathcal{B} \cup \{b\}, \mathcal{A}, \text{nil})}\ +b$$

$$\frac{(a : \varphi \leftarrow \mathcal{B}^- ;\ \mathcal{B}^+) \in \Lambda \quad a\theta = \text{act} \quad \mathcal{B} \vDash \varphi\theta}{(\mathcal{B}, \mathcal{A}, \text{act}) \longrightarrow ([\mathcal{B} \setminus \mathcal{B}^-\theta] \cup \mathcal{B}^+\theta, \mathcal{A} \cdot \text{act}, \text{nil})}\ \text{act}$$

Substitution $\varphi\theta$

Substitution of variables in φ with values from unifier θ

Our Framework: CAN-FPP

1. Adds notion of “pure” declarative goals, in addition to “CAN” declarative goals
    - “Active” vs. “inactive” pure declarative goals
    - ”Procedural” vs. “declarative” intention sets

2. Adds instrinsic support for automatic calls to FPP
    - Direct call (i.e. programmer specified)
    - Belief-driven call (via motivation library)
    - Recover-aid call (i.e. following plan failure)

3. Extends operational semantics
    - Adopting pure declarative goals
    - Planning for pure declarative goals
    - Managing (dropping, reactivating) pure declarative goals

FPP Problem $(\Lambda, \mathcal{B}, \varphi_g)$

Action description library

Set of STRIPS-style action descriptions

Initial state

Set of atoms $\mathcal{B} \subseteq \mathcal{L}$

Goal

Formula from $\varphi_g \in \mathcal{L}$

Offline solution $\mathrm{sol}^{off}(\Lambda, \mathcal{B}, \varphi_g) = \mathrm{act}_1 \, ; \mathrm{act}_2 \, ; \ldots ; \mathrm{act}_n$

Sequence of actions from Λ that is guaranteed to reach state $\mathcal{B}'$ from $\mathcal{B}$ such that $\mathcal{B}' \vDash \varphi_g$

Online solution $\mathrm{sol}^{on}(\Lambda, \mathcal{B}, \varphi_g) = \mathrm{act}_1$

Next best action from Λ so as to reach state $\mathcal{B}'$ from $\mathcal{B}'$ such that $\mathcal{B} \vDash \varphi_g$

<u>Assumes classical planning</u>

Could be extended to e.g. conformant planning

CAN-FPP: Agent $(\mathcal{B}, \Pi, \Lambda, \mathcal{M})$

Extended plan library
Set of extended plan rules

Motivation library
Set of motivation rules [à la van Riemsdijk, 2004]

Motivation rule $\varphi \rightsquigarrow \text{goal}(\varphi_s, \varphi_f)$

Triggering condition $\varphi \in \mathcal{L}$

Pure declarative goal

Extended plan rule $e : \varphi \leftarrow P'$

Extended body (program) $P' ::= P \mid \text{goal}(\varphi_s, \varphi_f) \mid \text{activate}\left(\text{goal}(\varphi_s, \varphi_f)\right)$

Pure declarative goal

Reactivation program

CAN-FPP: Agent Configuration $(\mathcal{B}, \Pi, \Lambda, \mathcal{M}, \mathcal{A}, \Gamma)$

Motivation library $\mathcal{M}$ is a static entity and will be omitted

Extended intention set $\Gamma = \Gamma_{pr} \cup \Gamma_{de}^{+} \cup \Gamma_{de}^{-}$

$\Gamma_{pr} \cap (\Gamma_{de}^{+} \cup \Gamma_{de}^{-}) = \emptyset$

$\Gamma_{de}^{+} \cap \Gamma_{de}^{-} = \emptyset$

Procedural intention set Γ_{pr}

Standard CAN intention set

Active declarative intention set Γ_{de}^{+}

Set of “active” pure declarative goals

Inactive declarative intention set Γ_{de}^{-}

Set of “inactive” pure declarative goals

Declarative intention set $\Gamma_{de} = \Gamma_{de}^{+} \cup \Gamma_{de}^{-}$

Set of all pure declarative goals

CAN-FPP: Adopting Pure Declarative Goals

$$\frac{P \in \Gamma_{pr} \quad P = \text{goal}(\varphi_s, \text{nil}, \varphi_f) \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f)}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{goal} (\mathcal{B}, \mathcal{A}, [\Gamma_{pr} \setminus \{P\}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\}])} A_{goal}^{1}$$

Direct

Translate a CAN declarative goal to a pure declarative goal if it has no procedure

$$\frac{(\varphi \rightsquigarrow P^{\uparrow}) \in \mathcal{M} \quad \mathcal{B} \vDash \varphi\theta \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f)}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{goal} (\mathcal{B}, \mathcal{A}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\theta\})} A_{goal}^{2}$$

Belief-driven

Trigger a motivation rule

$$\frac{P \in \Gamma_{pr} \quad P = \text{goal}(\varphi_s, P', \varphi_f) \quad \mathcal{B} \vDash \varphi_f \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f)}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{goal} (\mathcal{B}, \mathcal{A}, [\Gamma_{pr} \setminus \{P\}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\}])} A_{goal}^{3}$$

Recovery-aid

Translate a CAN declarative goal to a pure declarative goal if the procedure is blocked

CAN-FPP: Planning for Pure Declarative Goals

$$\frac{P^{\uparrow} \in \Gamma_{de}^{+} \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f) \quad \text{sol}^{off}(\Lambda, \mathcal{B}, \varphi_s) = P \quad P = \text{act}_1 ; \ldots ; \text{act}_n}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{bdi} (\mathcal{B}, \mathcal{A}, [\Gamma_{de} \setminus \{P^{\uparrow}\}, \Gamma_{pr} \cup \{\text{goal}(\varphi_s, P, \varphi_f)\}])} P_{\mathcal{F}^{off}}$$

Offline planning

Generate plan for an active declarative goal and add plan to procedural intention set

$$\frac{P^{\uparrow} \in \Gamma_{de}^{+} \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f) \quad \text{sol}^{on}(\Lambda, \mathcal{B}, \varphi_s) = \text{act} \quad P = \text{act} ; \text{activate}(P^{\uparrow})}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{bdi} (\mathcal{B}, \mathcal{A}, [\Gamma_{de}^{-} \cup \{P^{\uparrow}\}, \Gamma_{pr} \cup \{\text{goal}(\varphi_s, P, \varphi_f)\}])} P_{\mathcal{F}^{on}}$$

Online planning

Generate a single action for pure declarative goal, deactivate that goal, create plan to first execute that action then to reactive the goal, and add plan to procedural intention set

$$\frac{P^{\uparrow} \in \Gamma_{de}^{+} \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f) \quad \text{sol}(\Lambda, \mathcal{B}, \varphi_s) = \bot}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{bdi} (\mathcal{B}, \mathcal{A}, \Gamma_{de} \setminus \{P^{\uparrow}\})} P_{\mathcal{F}^{\bot}}$$

Planning failure

Drop pure declarative goal if planning fails (alternatively: add goal to inactive declarative intention set)

CAN-FPP: Managing Pure Declarative Goals

$$\frac{P^{\uparrow} \in \Gamma_{de} \quad P^{\uparrow} = \text{goal}(\varphi_s, \varphi_f) \quad \mathcal{B} \vDash \varphi_s \vee \varphi_f}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{drop} (\mathcal{B}, \mathcal{A}, \Gamma_{de} \setminus \{P^{\uparrow}\})} \; G_{drop}$$

Drop pure declarative goal

Drop declarative goal (whether active or inactive) if success or fail condition is met

$$\frac{P \in \Gamma_{pr} \quad P = \text{activate}(P^{\uparrow}) \quad P^{\uparrow} \in \Gamma_{de}^{-}}{(\mathcal{B}, \mathcal{A}, \Gamma) \xrightarrow{goal} (\mathcal{B}, \mathcal{A}, [\Gamma_{pr} \setminus \{P\}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\}])} \; A_{re}$$

Reactivate pure declarative goal

Activate a pure declarative goal that is inactive

CAN-FPP: Theoretical Results

Offline FPP

Goal state

$$\frac{P^{\uparrow} = \mathrm{goal}(\varphi_s, \varphi_f) \quad \mathrm{sol}^{off}(\Lambda, \mathcal{B}, \varphi_s) = P \quad (\mathcal{B}, \mathcal{A}, P) \xrightarrow{bdi} \ldots \xrightarrow{bdi} (\mathcal{B}', \mathcal{A} \cdot P, \mathrm{nil}) \quad \mathcal{B}' \vDash \varphi_s}{(\mathcal{B}, \mathcal{A}, P^{\uparrow}) \xrightarrow{bdi} \ldots \xrightarrow{bdi} (\mathcal{B}', \mathcal{A} \cdot P, \mathrm{nil})} \text{ Theorem 1 (i)}$$

Successful execution

Offline FPP

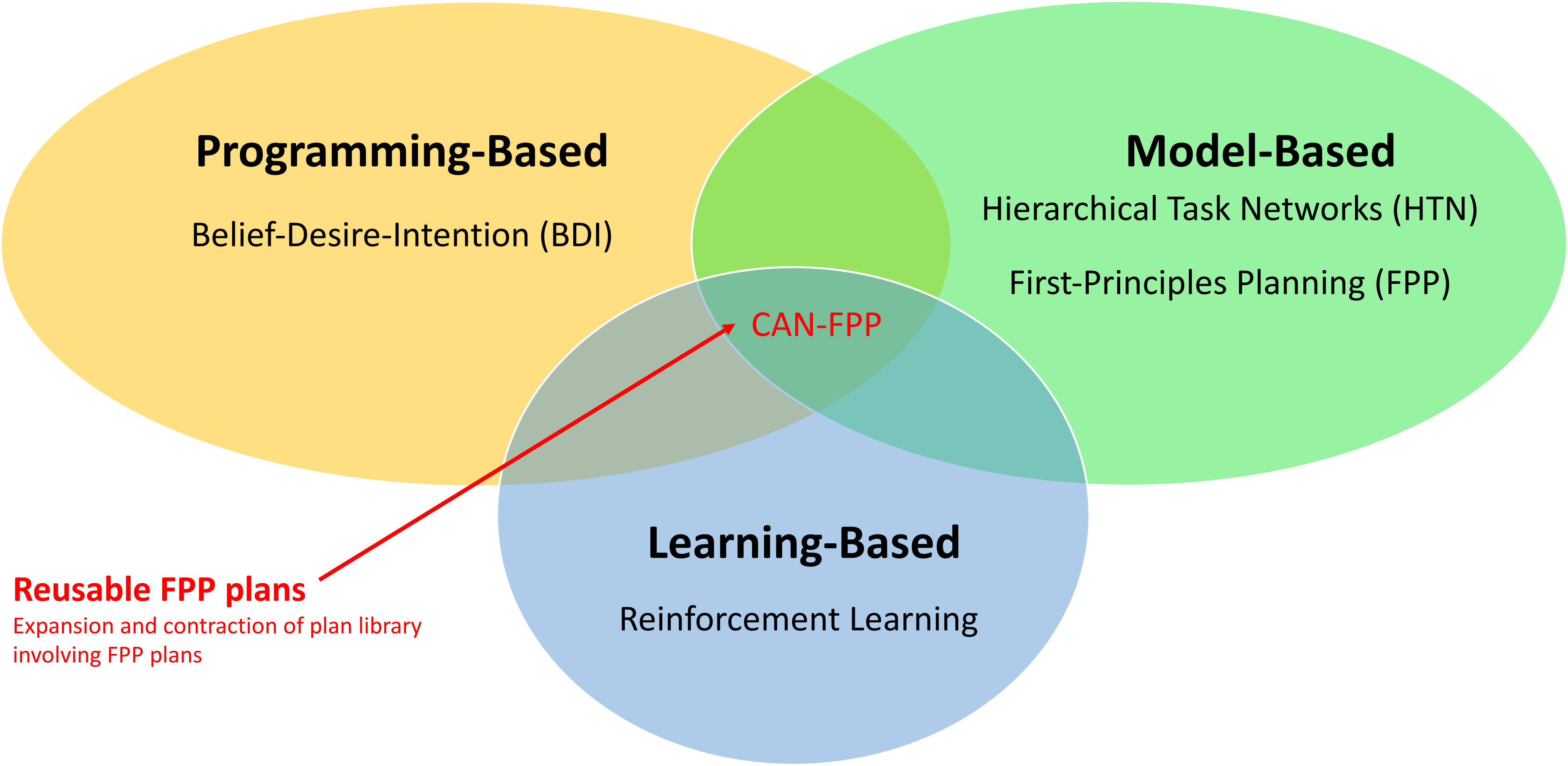
Goal state

$$\frac{\begin{array}{c} P^{\uparrow} = \mathrm{goal}(\varphi_s, \varphi_f) \quad \mathrm{sol}^{on}(\Lambda, \mathcal{B}, \varphi_s) = \mathrm{act}_1 \quad P = \mathrm{act}_1 \,; \mathrm{activate}(P^{\uparrow}) \\ (\mathcal{B}, \mathcal{A}, P) \xrightarrow{bdi} \ldots \xrightarrow{bdi} (\mathcal{B}', \mathcal{A} \cdot \mathrm{act}_1 \cdot \ldots \cdot \mathrm{act}_n, \mathrm{nil}) \quad \mathcal{B}' \vDash \varphi_s \end{array}}{(\mathcal{B}, \mathcal{A}, P^{\uparrow}) \xrightarrow{bdi} \ldots \xrightarrow{bdi} (\mathcal{B}', \mathcal{A} \cdot \mathrm{act}_1 \cdot \ldots \cdot \mathrm{act}_n, \mathrm{nil})} \text{ Theorem 1 (iii)}$$

Successful execution

Future Work

Mengwei Xu, Kim Bauters, Kevin McAreavey, and Weiru Liu. **A framework for plan library evolution in BDI agent systems.** In *Proceedings of ICTAI'18*, to appear.



Reusable FPP plans

Expansion and contraction of plan library involving FPP plans

Questions?

Thank you