

Hierarchical Planning in BDI Agent Programming Languages

Sebastian Sardina
RMIT University
ssardina@cs.rmit.edu.au

Lavindra de Silva
RMIT University
ldesilva@cs.rmit.edu.au

Lin Padgham
RMIT University
linpa@cs.rmit.edu.au

MOTIVATION & OBJECTIVES

- What is the problem?**
No **lookahead deliberation** in available BDI Agent Programming languages and systems:
(a) only *reactive* planning;
(b) implemented systems with no formal semantics.
- What is the aim?**
Accommodate **planning capabilities** into BDI Agent Programming language (e.g., AGENTSPEAK) with:
(a) **formal semantics**;
(b) **direct implementation**.
- What is our solution/approach?**
Extend the BDI agent language CAN with a new language construct $\text{Plan}(P)$: **on-demand planning on P !**
- What is the outcome?**
CANPLAN = CAN + **HTN planning**
- What do we leave out?**
Plan monitoring & re-planning, goal reasoning, and much more...

BDI-PROGRAMMING

A very popular **agent-oriented** programming methodology.

Key features of BDI agent-oriented systems:

- Beliefs:** information about the world (set of facts).
 - Events:** goals/desires to resolve; internal or external.
 - Plan library:** recipes for handling goals-events: $e : \psi \leftarrow P$.
 - Intentions:** partially instantiated programs; commitment.
- **A Personal Scheduler Manager example:**
- Events:** $reqMeeting(p, t)$ (external); $!freeSlot(t)$ (internal).
 - Plan library:** plans for handling a meeting request with different priorities depending on the person p .
- 2 Intentions:** a partial plan to make slot $Wed(3pm)$ free; a plan to inform $John$ that a meeting is not possible.

HTN PLANNING

Properties:

- Goals-to-do instead of goals-to-be: $goToLocation(YYZ)$
- Decomposition of high-level *tasks*.
- User provides (procedural) domain knowledge.
- Some similarities with "programming".
- Subsumes first-principle "STRIP" planning.
- Well understood semantics and implementations.

Components of HTN:

- State:** set of atoms + CWA.
- Task:** a common goal; primitive or compound.
- Task Network:** set of tasks T + order/state constraints ϕ .
- Method:** a task network to solve high-level task e .

HTN solution: a plan σ that stands for a **full decomposition** of a top-level task. Operational and model-theoretic semantics were given in [Erol, Hendler & Nau 94]:

$$\Rightarrow sol(d, \mathcal{B}, \mathcal{D}): \text{set of all plans that solve } d.$$

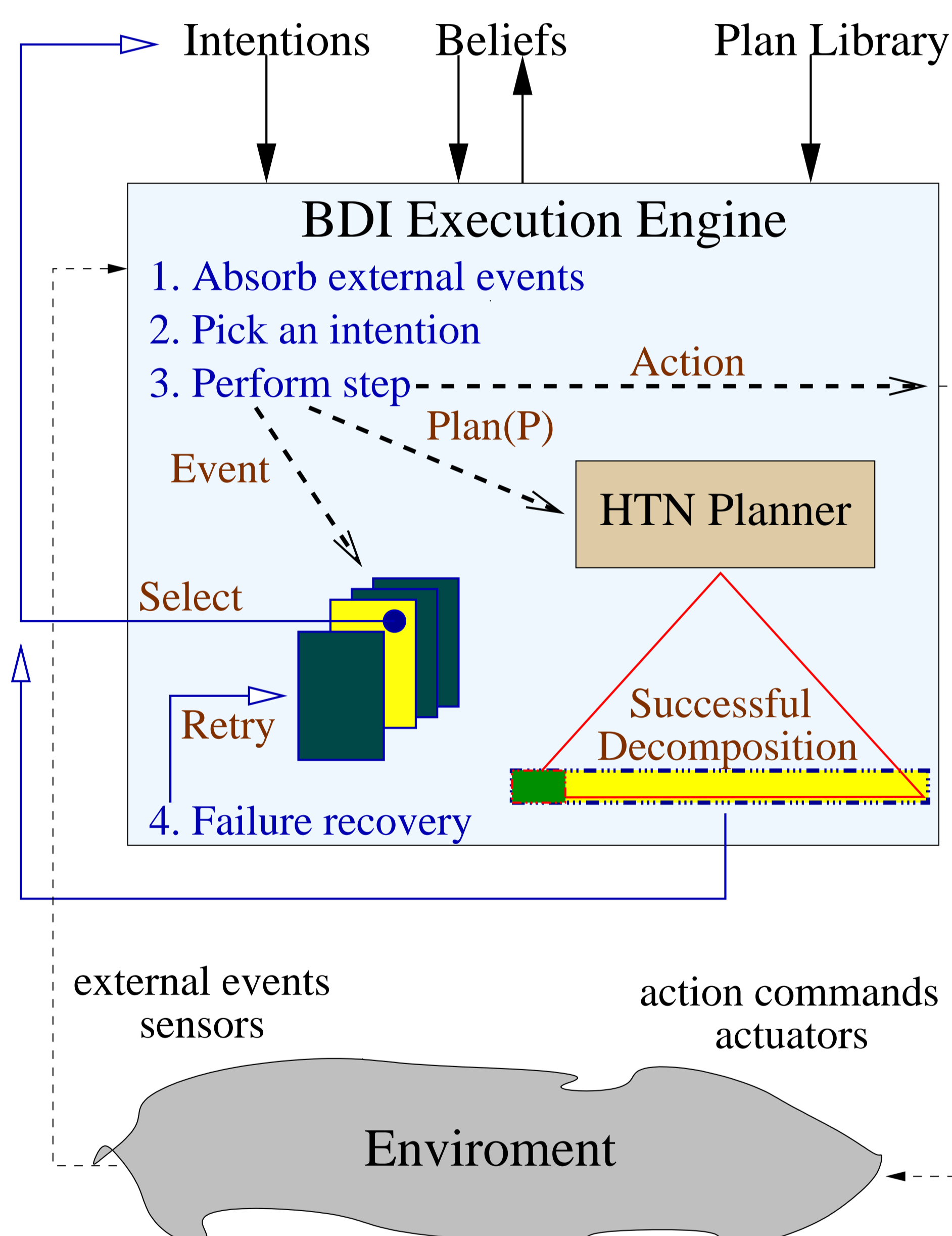
BDI AND HTN: SIMILARITIES

BDI SYSTEMS	HTN SYSTEMS
belief base	state
plan library	method library
event	high-level task
action	primitive task
plan-body/program	network task
plan rule	method
plan rule context	method precondition
test $?\phi$ in plan-body	state constraints
sequence in plan-body	ordering constraint \prec
parallelism in plan-body	no ordering constraint

THE CAN LANGUAGE [WPHT02]

- A CAN agent is defined as $\mathcal{A} = \langle \mathcal{N}, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, where:
 - \mathcal{N} is the **agent name**.
 - Π is a **plan library** containing plan rules $e : \psi \leftarrow P$
 e is the *triggering event* and ψ is the *context condition*;
 P is the *plan-body* from the following language:
 $P ::= act \mid ?\phi \mid +b \mid -b \mid \underline{e} \mid P_1; P_2 \mid P_1 \parallel P_2 \mid \text{Goal}(\phi_s, P_1, \phi_f)$.
 - \mathcal{B} is the **belief base**: current agent's knowledge.
 - \mathcal{A} is the sequence of **executed actions**.
 - Γ is the **intention base**: partially instantiated plan-bodies.
- Special features in CAN:
 - Includes **failure-handling** when a sub-goal fails:
 \Rightarrow retries alternative plans for the sub-goal.
 - Construct **Goal**(ϕ_s, P, ϕ_f): declarative+procedural goal
 \Rightarrow achieve goal ϕ_s by using plan P ; failing if ϕ_f .

CANPLAN = CAN + Plan



FORMALIZATION OF CANPLAN

- Two basic transitions: $\langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{\text{bdi/plan}} \langle \mathcal{B}', \mathcal{A}', P' \rangle$
 - Agent level transition: $\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle \Rightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}', \mathcal{A}', \Gamma' \rangle$
- $$\frac{P \in \Gamma \quad \langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{\text{bdi}} \langle \mathcal{B}', \mathcal{A}', P' \rangle}{\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle \Rightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}', \mathcal{A}', (\Gamma \setminus \{P\}) \cup \{P'\} \rangle} A_{step}$$
- $$\frac{!e \rightarrow (\psi_1 : P_1, \dots, \psi_n : P_n) \rightarrow}{P_i \triangleright (\psi_1 : P_1, \dots, \psi_{i-1} : P_{i-1}, \psi_{i+1} : P_{i+1}, \dots, \psi_n : P_n) \rightarrow \dots} \rightarrow$$
- $$\frac{\Delta = \{\psi_i \theta : P_i \theta \mid e' : \psi_i \leftarrow P_i \in \Pi \wedge \theta = \text{mgu}(e, e')\}}{\langle \mathcal{B}, \mathcal{A}, !e \rangle \rightarrow \langle \mathcal{B}, \mathcal{A}, (\Delta) \rangle} Event$$
- $$\frac{\psi_i : P_i \in \Delta \quad \mathcal{B} \models \psi_i \theta}{\langle \mathcal{B}, \mathcal{A}, (\Delta) \rangle \rightarrow \langle \mathcal{B}, \mathcal{A}, P_i \theta \triangleright (\Delta \setminus P_i) \rangle} Sel$$
- $$\frac{\langle \mathcal{B}, \mathcal{A}, P_1 \rangle \xrightarrow{\text{bdi}} \langle \mathcal{B}, \mathcal{A}, P_1 \triangleright P_2 \rangle}{\langle \mathcal{B}, \mathcal{A}, nil \triangleright P_2 \rangle \rightarrow \langle \mathcal{B}, \mathcal{A}, nil \rangle} \triangleright_f$$
- $$\frac{\langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{\text{plan}} \langle \mathcal{B}', \mathcal{A}', P' \rangle \quad \langle \mathcal{B}', \mathcal{A}', P' \rangle \xrightarrow{\text{plan}} \langle \mathcal{B}_f, \mathcal{A}_f, nil \rangle}{\langle \mathcal{B}, \mathcal{A}, \text{Plan}(P) \rangle \xrightarrow{\text{bdi}} \langle \mathcal{B}', \mathcal{A}', \text{Plan}(P') \rangle} Plan$$

Most of the semantic rules are shared between the BDI cycle and the planning sub-module!

PROPERTIES OF CANPLAN

- Some remarks about Plan:
 - Plan is built-in within the BDI execution cycle.
 - Plan's semantics is defined "in terms" of the BDI cycle.
 - But failure-handling is left to the BDI cycle only.
- Technical results:
 - If $\text{Plan}(P)$ can perform a step, then $\text{Plan}(P)$ **never fails** in an environment-free agent execution.
 - All executions of $\text{Plan}(P)$ corresponds to **HTN solutions** of P w.r.t. the agent's plan-library Π .
 - P **simulates** $\text{Plan}(P)$ in every CANPLAN agent with no concurrency \parallel and no goal-programs Goal .
 $\Rightarrow \text{Plan}(\text{Plan}(P_1) \parallel P_2) = \text{Plan}(P_1 \parallel P_2) \neq \text{Plan}(P_1) \parallel P_2$.

IMPLEMENTATION OF CANPLAN

Theorem 2 is the basis for an implementation:
 $\Rightarrow \text{JACK+ JSHOP}$ [Lavindra & Padgham 05]

But:

- No concurrency \parallel and no goal-programs $\text{Goal}(\phi_s, P, \phi_f)$.
- Converts JSHOP's total-order plans into partial-order plans.
- Planner returns the relevant methods and bindings; the BDI execution engine then follows step-by-step the decomposition suggested by the planner.

PLANNING FOR DEC. GOALS

ALTERNATIVES	(A)	(B)	(C)	(D)	(E)
$\text{Plan}(P; ?\phi_s)$	✓				
$\text{Plan}(\text{Goal}(\phi_s, P, \phi_f))$	✓	✓		✓	
$\text{Goal}(\phi_s, \text{Plan}(P), \phi_f)$		✓	✓	✓	
$\text{Goal}(\phi_s, \text{Plan}(P; ?\phi_s), \phi_f)$	✓	✓	✓	✓	
$\text{Goal}(\phi_s, \text{Plan}(\text{Goal}(\phi_s, P, \phi_f)), \phi_f)$	✓	✓	✓	✓	✓

(A) P is used towards the eventual satisfaction of goal ϕ_s .

(B) Allow partial executions of P .

(C) Commitment on the goal ϕ_s .

(D) Goal dropping mechanism.

(E) P may be solved partially.

FUTURE WORK

- Planning upon failure (like PROPICE-PLAN & CPEF):
 $\Rightarrow P \triangleright \text{Plan}(P')$.
- Using planning for learning new plans
When no relevant plan is available \Rightarrow classical planning.
Add new plan to the BDI plan library Π
- Planning in the context of other intentions.
- Planning up to some level of abstraction
Leave details to the BDI execution engine.
- Use SHOP2 for implementing planning in JACK.

RELATED WORK

- Other formal BDI-style systems (with no planning):
AGENTSPEAK, 3APL, DMARS.
- Systems/architectures mixing planning with BDI execution:
 - Planners that interleave execution: IPEM; SAGE.
 - Planning always: RETSINA.
 - Planning on failure: PROPICE-PLAN; CPEF.
- INDIGOLOG [De Giacomo & Levesque 99]:
 - Execution+planning in the situation calculus.
 - Inspired semantics of $\text{Plan}(P)$ from search construct $\Sigma(\delta)$.
 - Not a typical programming language in the BDI tradition
 - GOLOG and HTN planning [Gabaldon 02].