Multi-Tier Planning with $\text{LTL}_f/\text{PLTL}_f$ Goals

Ramon Fraga Pereira

Reasoning Agents Seminar

Sapienza University of Rome
DIAG

April, 2021

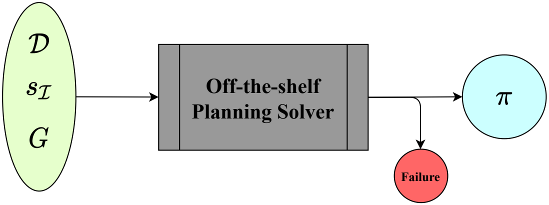
Outline

- Introduction and Motivation;
- Recap on FOND Planning;
- Multi-Tier Automated Planning;
- $\mathrm{LTL}_f/\mathrm{PLTL}_f$ to FOND Planning; and
- Project Ideas;

Introduction: AI Planning

- In **AI Planning**, a *plan* π is synthesized from a *domain model* $\mathcal{D}$, an *initial state* $s_{\mathcal{I}}$ and a *goal (condition) state* G;

- A *domain model* $\mathcal{D}$ describes how *actions* change the world, via the specification of their *preconditions* and *effects*.

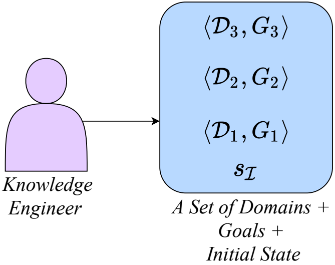


Motivation

- A *domain model* $\mathcal{D}$ is never complete, and sometimes it needs to make **different assumptions** on the environment's dynamics.

- By allowing the specification of just **one** *domain model* $\mathcal{D}$, the knowledge engineer is only able to make **one set of assumptions**, and to specify a **single** goal.

- Ciolek et al.[1] propose a Multi-tier Planning framework.


$\langle \mathcal{D}_3, G_3 \rangle$
$\langle \mathcal{D}_2, G_2 \rangle$
$\langle \mathcal{D}_1, G_1 \rangle$
$s_{\mathcal{I}}$
Knowledge Engineer
A Set of Domains + Goals + Initial State

[1] Ciolek et al., Multi-tier Automated Planning for Adaptive Behavior, ICAPS, 2020.

Recap on FOND Planning

FOND Planning

Fully Observable Non-Deterministic (FOND) Planning

Definition (**FOND Planning Domain**)

A FOND planning *domain model* is a tuple $\mathcal{D} = \langle 2^{\mathcal{F}}, \mathcal{A} \rangle$, where:

- $\mathcal{F}$ is the set of **fluents**, and $2^{\mathcal{F}}$ is a set of **states**;
- $\mathcal{A}$ is a set of **non-deterministic actions**.
  - Every action $a \in \mathcal{A}$ has:
    - A set of **preconditions** Pre_a; and
    - A set of **non-deterministic effects** $Eff_a = \langle e_0, ..., e_n \rangle$ with $n \geq 1$;

Action: `pick-up-block`

- *Pre*: `(and (hand-empty) (block-on-table))`

- Eff_0: `()` - Eff_1*:* `((and (holding-block) (not (hand-empty)) (not (block-on-table)))`

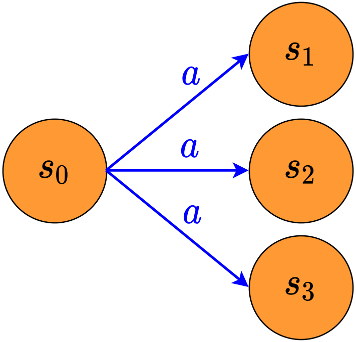
FOND Planning Problem

Definition (**FOND Planning Problem**)

A FOND planning *problem* is tuple $P = \langle \mathcal{D}, s_{\mathcal{I}}, G \rangle$, $\mathcal{D}$ is a FOND planning domain model, $s_{\mathcal{I}}$ is the **initial state**, and G is the **goal state**;

FOND Planning: Who controls what?

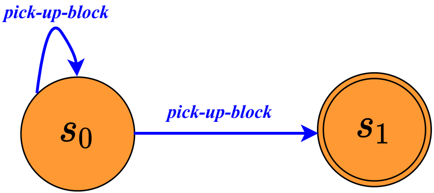
- Fluents are controlled by the Environment.
- Actions are controlled by the Agent.



Solution to FOND Planning Problems

Definition (Solution to a FOND Planning Problem)

A **solution** to a FOND planning problem $\mathcal{P}$ is a *policy* π, which is formally defined as a partial function $\pi : (2^{\mathcal{F}})^{+} \rightarrow \mathcal{A}$ that maps *states* into *applicable actions* that eventually achieve the goal G;



Types of Solutions to FOND Planning

Definition (**Weak Solution**)

A **weak solution** is a policy π that achieves the goal state G under at least one selection of action outcomes.

Definition (**Strong Solution**)

A **strong solution** is a policy π that is guaranteed to achieve the goal state G regardless of the environment's non-determinism.

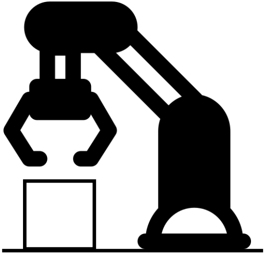
Definition (**Strong-Cyclic Solution**)

A **strong-cyclic solution** is a policy π that guarantees to achieve the goal state G only under the assumption of fairness[a].

[a] The fairness assumption defines that every outcome of a non-deterministic action will occur infinitely often if the action is executed infinitely often.

FOND Planning Example

- $\mathcal{A}$ = `pick-up-block` with **two possible outcomes**: it may **succeed**, or it may **fail** by leaving the block on the table.
- $s_{\mathcal{I}}$ = `(hand-empty) (block-on-table)`
- G = `(holding-block)`

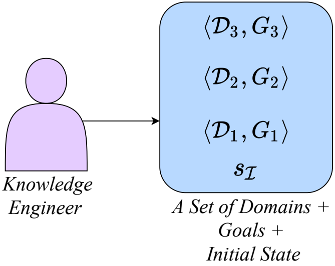


- Two types of solutions:
  - **Weak solution**: A single execution `pick-up-block` may **fail** or **succeed**;
  - **Strong-cyclic solution**: Execute `pick-up-block` until **succeed**;

Multi-Tier Automated Planning

Multi-Tier Automated Planning

- **Multi-Tier Planning** is a framework that allows the specification of *different sets of domain models* with *different corresponding goals*.
- The framework aims to support the *synthesis of adaptive behavior* so as to mitigate the intrinsic risk in any planning modeling task.


$\langle \mathcal{D}_3, G_3 \rangle$
$\langle \mathcal{D}_2, G_2 \rangle$
$\langle \mathcal{D}_1, G_1 \rangle$
$s_{\mathcal{I}}$
Knowledge Engineer
A Set of Domains + Goals + Initial State

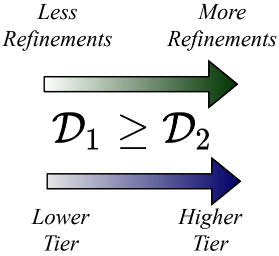
Multi-Tier Planning Domain

Definition (**Multi-Tier Planning Domain**)

A *multi-tier planning domain* is a tuple $\langle \Omega, \leq \rangle$, where:

- Ω *is a set of FOND planning domain models* $\mathcal{D}$ over the same fluents $\mathcal{F}$ and action operators signature[a].
- $\leq$ is a *partial-order relation* over Ω. For instance, $\mathcal{D}_1 \leq \mathcal{D}_2$ defines that $\mathcal{D}_2$ a **higher tier** model that refines a **lower tier** model $\mathcal{D}_1$.

[a] Actions share the same preconditions across all FOND domains in Ω.



Less Refinements

More Refinements

$$\mathcal{D}_1 \geq \mathcal{D}_2$$

Lower Tier

Higher Tier

Multi-Tier Planning Problem

Definition (**Multi-Tier Planning Problem**)

A *multi-tier planning problem* is a tuple $\mathcal{M} = \langle\langle\Omega, \leq\rangle, s_{\mathcal{I}}, \mathcal{G}\rangle$, where:

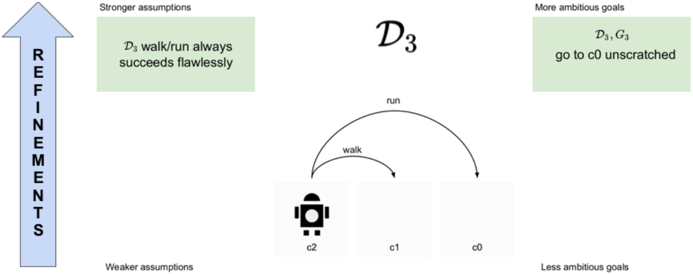
- $\langle\Omega, \leq\rangle$ is a *multi-tier planning domain*;
- $s_{\mathcal{I}}$ is the *initial state*; and
- $\mathcal{G}$ is a function that maps each domain $\mathcal{D}$ in Ω to its corresponding goal G, i.e., $\mathcal{G}(\mathcal{D})$.

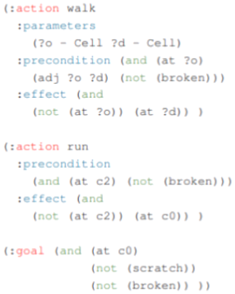
Multi-Tier Controller

Definition (**Multi-Tier Controller**)

A *multi-tier controller* is a function $\mathcal{C}$ that maps each domain $\mathcal{D} \in \Omega$ to a specific policy $\mathcal{C}(\mathcal{D})$.

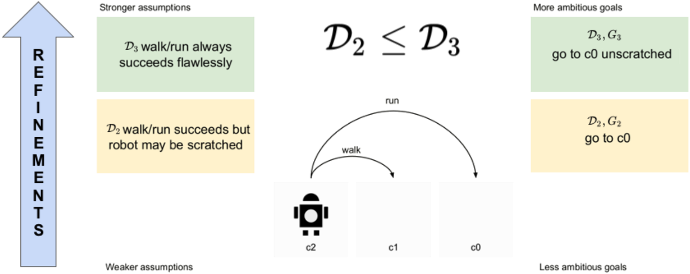
Multi-Tier Planning Example (1 of 3)

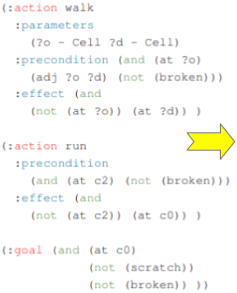




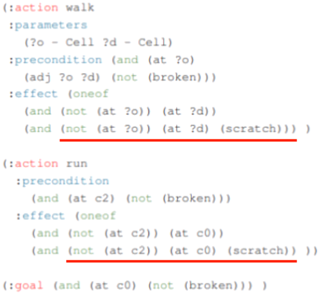
```
(:action walk
  :parameters
    (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (and
    (not (at ?o)) (at ?d)) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (and
    (not (at c2)) (at c0)) )

(:goal (and (at c0)
            (not (scratch))
            (not (broken)) ))
```

Multi-Tier Planning Example (2 of 3)


Stronger assumptions
$\mathcal{D}_3$ walk/run always succeeds flawlessly
$\mathcal{D}_2$ walk/run succeeds but robot may be scratched
Weaker assumptions
REFINEMENTS
$\mathcal{D}_2 \leq \mathcal{D}_3$
run
walk
c2
c1
c0
More ambitious goals
$\mathcal{D}_3, G_3$ go to c0 unscratched
$\mathcal{D}_2, G_2$ go to c0
Less ambitious goals

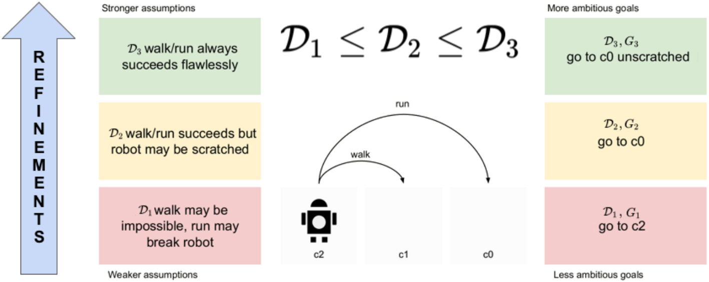


```
(:action walk
  :parameters
    (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (and
    (not (at ?o)) (at ?d)) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (and
    (not (at c2)) (at c0)) )

(:goal (and (at c0)
             (not (scratch))
             (not (broken)) ))
```



```
(:action walk
 :parameters
   (?o - Cell ?d - Cell)
 :precondition (and (at ?o)
   (adj ?o ?d) (not (broken)))
 :effect (oneof
   (and (not (at ?o)) (at ?d))
   (and (not (at ?o)) (at ?d) (scratch))) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (oneof
    (and (not (at c2)) (at c0))
    (and (not (at c2)) (at c0) (scratch)) ))

(:goal (and (at c0) (not (broken))) )
```

Multi-Tier Planning Example (3 of 3)



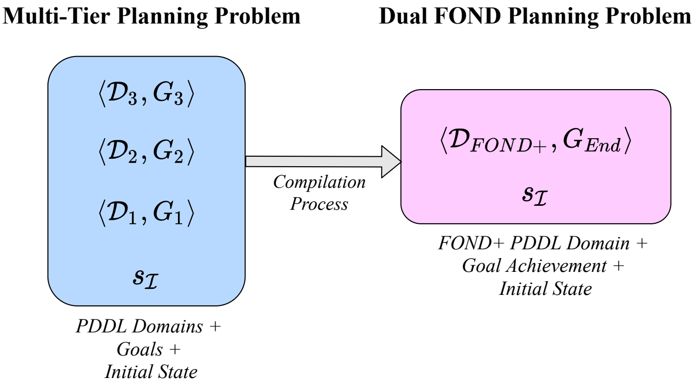
REFINEMENTS
Stronger assumptions
$\mathcal{D}_3$ walk/run always succeeds flawlessly
$\mathcal{D}_2$ walk/run succeeds but robot may be scratched
$\mathcal{D}_1$ walk may be impossible, run may break robot
Weaker assumptions
$\mathcal{D}_1 \leq \mathcal{D}_2 \leq \mathcal{D}_3$
run
walk
c2
c1
c0
More ambitious goals
$\mathcal{D}_3, G_3$ go to c0 unscratched
$\mathcal{D}_2, G_2$ go to c0
$\mathcal{D}_1$, G_1 go to c2
Less ambitious goals

```
(:action walk
  :parameters
    (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (and
    (not (at ?o)) (at ?d)) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (and
    (not (at c2)) (at c0)) )

(:goal (and (at c0)
            (not (scratch))
            (not (broken)) ))
```

```
(:action walk
 :parameters
   (?o - Cell ?d - Cell)
 :precondition (and (at ?o)
   (adj ?o ?d) (not (broken)))
 :effect (oneof
   (and (not (at ?o)) (at ?d))
   (and (not (at ?o)) (at ?d) (scratch))) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (oneof
    (and (not (at c2)) (at c0))
    (and (not (at c2)) (at c0) (scratch)) ))

(:goal (and (at c0) (not (broken))) )
```

```
(:action walk
  :parameters (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (oneof
    (and (not (at ?o)) (at ?d))
    (and (not (at ?o)) (at ?d) (scratch))
    (scratch) ))

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (oneof
    (and (not (at c2)) (at c0))
    (and (not (at c2)) (at c0) (scratch))
    (broken) ))

(:goal (and (at c2) (not broken)) )
```

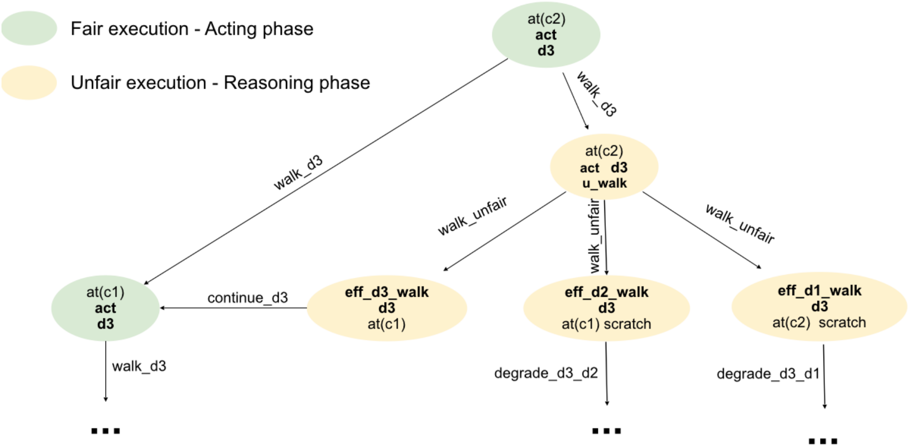
Compilation to Dual FOND Planning


Multi-Tier Planning Problem
Dual FOND Planning Problem
$\langle \mathcal{D}_3, G_3 \rangle$
$\langle \mathcal{D}_2, G_2 \rangle$
$\langle \mathcal{D}_1, G_1 \rangle$
$s_{\mathcal{I}}$
PDDL Domains + Goals + Initial State
Compilation Process
$\langle \mathcal{D}_{FOND+}, G_{End} \rangle$
$s_{\mathcal{I}}$
FOND+ PDDL Domain + Goal Achievement + Initial State

- A **Dual FOND Problem** has **fair** and **unfair** actions;
- FOND-SAT[2] is the only planner that extracts Dual-FOND solutions.

[2] Geffner and Geffner, Compact Policies for Fully Observable Non-Deterministic Planning as SAT, ICAPS, 2018.

Dual FOND Policy for the Multi-Tier Planning Example

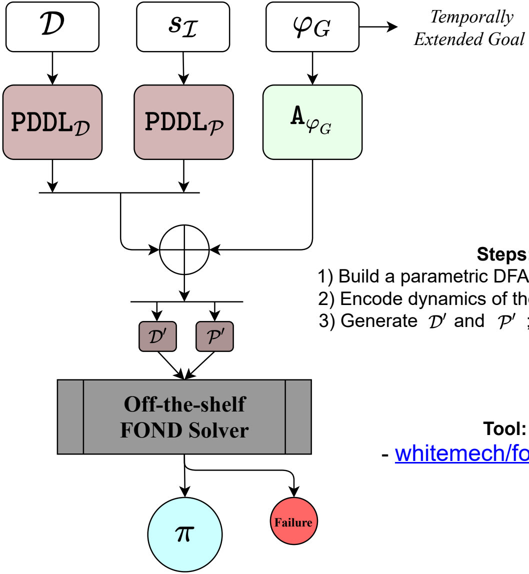


$\mathrm{LTL}_f/\mathrm{PLTL}_f$ to FOND Planning

LTL_f and PLTL_f to FOND Planning

- LTL_f (Linear Temporal Logic over Finite Traces);
- PLTL_f (Past Linear Temporal Logic over Finite Traces);
- Why use Temporally Extended Goals in FOND planning?
  - A different and richer class of policies;
  - Restrict the way the goal is achieved.

Automata Encoding of LTL$_f$ and PLTL$_f$ Goals to PDDL



Steps:

1) Build a parametric DFA of $\mathtt{A}_{\varphi_G}$ (PDFA);
2) Encode dynamics of the PDFA in PDDL;
3) Generate $\mathcal{D}'$ and $\mathcal{P}'$;

Tool:

- whitemech/fond4ltlfpltlf

Some LTL$_f$ and PLTL$_f$ Goals for the Multi-Tier Example

LTL$_f$ formula with **Until** operator $\mathcal{U}$:

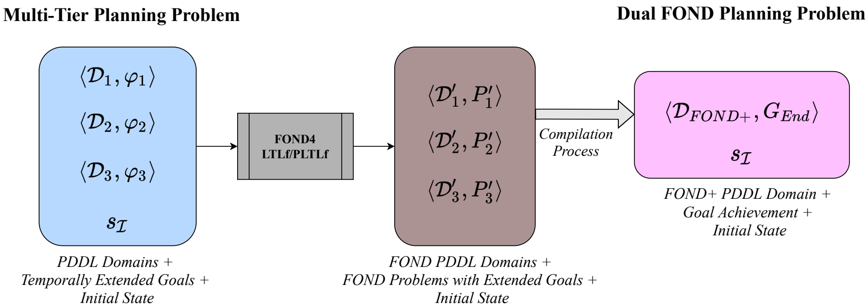
- $\phi_1 \mathcal{U} \phi_2$: $\neg$(scratch) $\mathcal{U}$ (at c0)

PLTL$_f$ formula with **Once** operator $\diamondminus$:

- $\phi_1 \wedge \diamondminus \phi_2$: (at c0) $\wedge$ $\diamondminus$ (at c1)

Project Ideas

Project Idea Overview


Multi-Tier Planning Problem
Dual FOND Planning Problem
$\langle \mathcal{D}_1, \varphi_1 \rangle$
$\langle \mathcal{D}_2, \varphi_2 \rangle$
$\langle \mathcal{D}_3, \varphi_3 \rangle$
$s_{\mathcal{I}}$
FOND4 LTLf/PLTLf
$\langle \mathcal{D}'_1, P'_1 \rangle$
$\langle \mathcal{D}'_2, P'_2 \rangle$
$\langle \mathcal{D}'_3, P'_3 \rangle$
Compilation Process
$\langle \mathcal{D}_{FOND+}, G_{End} \rangle$
$s_{\mathcal{I}}$
PDDL Domains + Temporally Extended Goals + Initial State
FOND PDDL Domains + FOND Problems with Extended Goals + Initial State
FOND+ PDDL Domain + Goal Achievement + Initial State

- **Use** FOND4LTL$_f$/PLTL$_f$[3] as part of the Multi-Tier *compilation process*;
- **Test the scalability** of FOND-SAT[4] with LTL$_f$ and PLTL$_f$ goals.

[3] http://fond4ltlfpltl.diag.uniroma1.it

[4] https://github.com/tomsons22/FOND-SAT

Other Project Ideas

- **Modify** the SAT encoding of FOND-SAT to improve the performance;
- **Modify** and **adapt** existing FOND planners to extract Dual FOND policies;
- **Develop** a new Dual FOND planner;

References and Available Tools

- Ciolek et al., Multi-tier Automated Planning for Adaptive Behavior, ICAPS, 2020;
- Fuggitti, FOND Planning for LTLf and PLTLf Goals, MSc Thesis, 2018;
- FOND4LTL$_f$/PLTL$_f$:
  - `http://fond4ltlfpltl.diag.uniroma1.it`
- Multi-Tier Planning framework:
  - `https://github.com/ssardina-planning/pypddl-translator`
- FOND-SAT planner:
  - `https://github.com/tomsons22/FOND-SAT`

Thank you!

pereira@diag.uniroma1.it