SibylSat: Using SAT as an oracle to perform a greedy search on TOHTN Planning

Gaspard Quenard, Damien Pellier, Humbert Fiorino

Summary

1. Introduction to HTN planning
2. SAT-based TOHTN planners
3. SibylSAT

1. HTN Reminder

Classical planning

In AI planning, we have

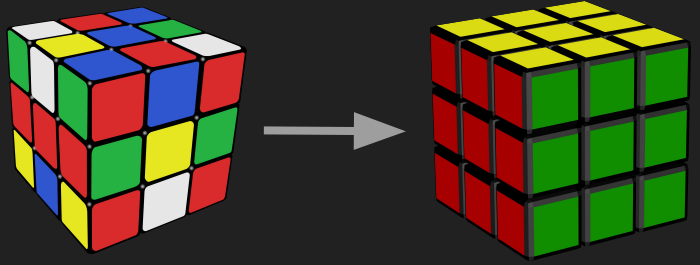
- An initial state (set of predicates)
- A goal state (set of predicates)
- Some actions, which have *preconditions* and *effects*

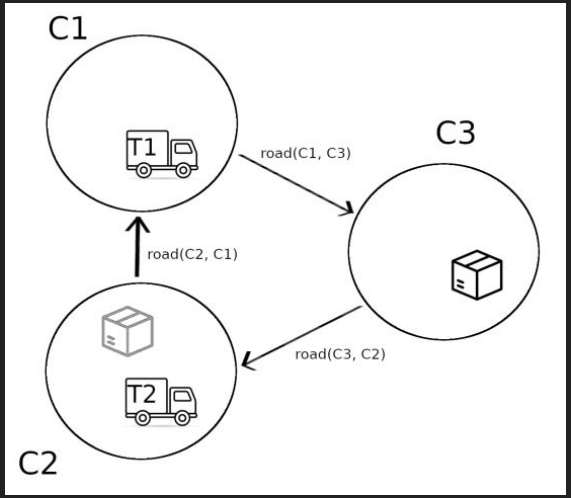
The goal of a planner if to find a *plan:*

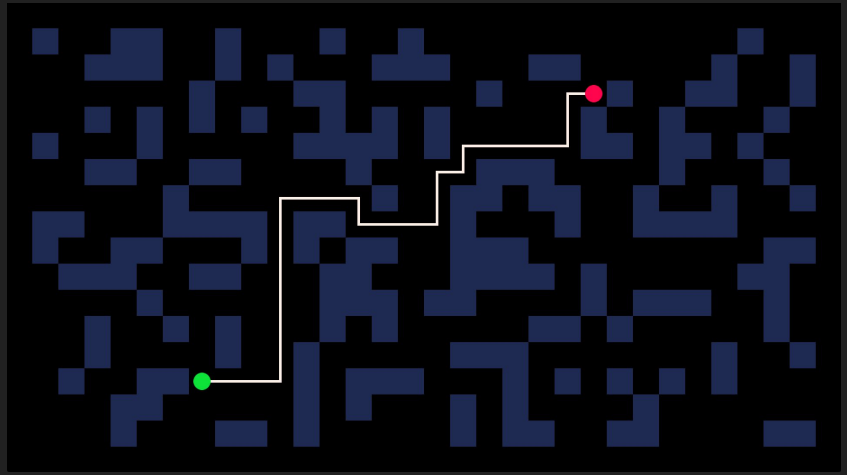
- A series of actions which leads from the initial state to the goal state

1. HTN Reminder

Classical planning

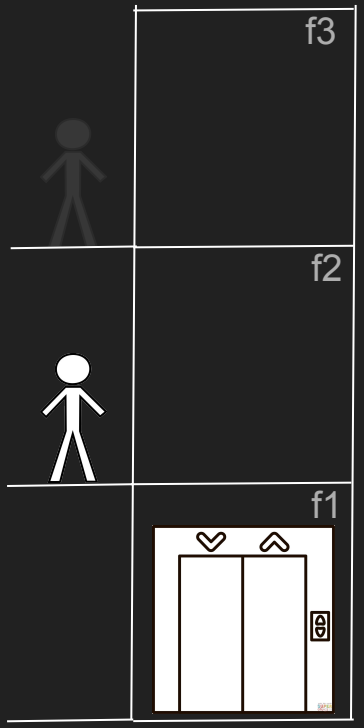






1. HTN Reminder

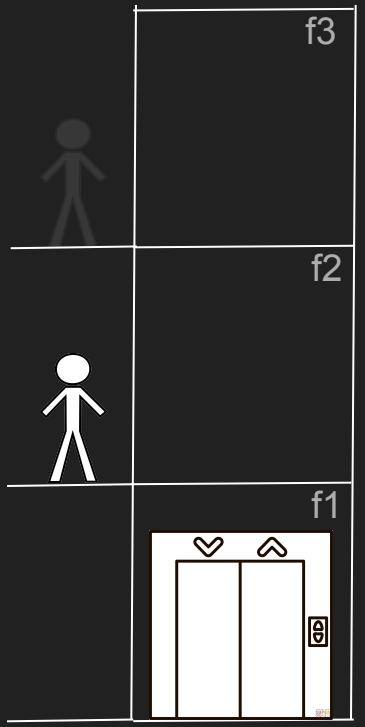
Classical planning




```
;; Init state
Lift_at f1
At p1 f2

;; Goal state
At p1 f3

;; Actions
(:action move
        :parameters (?f1 - Floor ?f2 - Floor)
        :precondition (lift_at ?f1)
        :effect (not (lift_at ?f1) and (lift_at ?f2))
)

(:action board
        :parameters (?p - Person ?f - Floor)
        :precondition ((at ?p ?f) (lift_at ?f))
        :effect (not (at ?p ?f) (boarded ?p))
)

(:action debark
        :parameters (?p - Person ?f - Floor)
        :precondition ((boarded ?p) (lift_at ?f))
        :effect (not (boarded ?p) (at ?p ?f))
        )
)
```

1. HTN Reminder

Classical planning

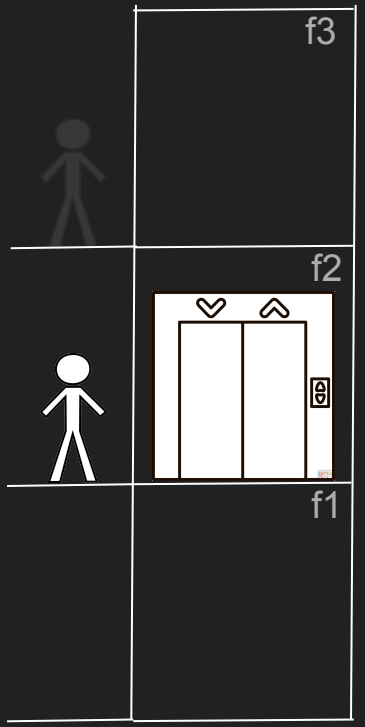


Plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f3)
4. debark(p, f3)

1. HTN Reminder

Classical planning

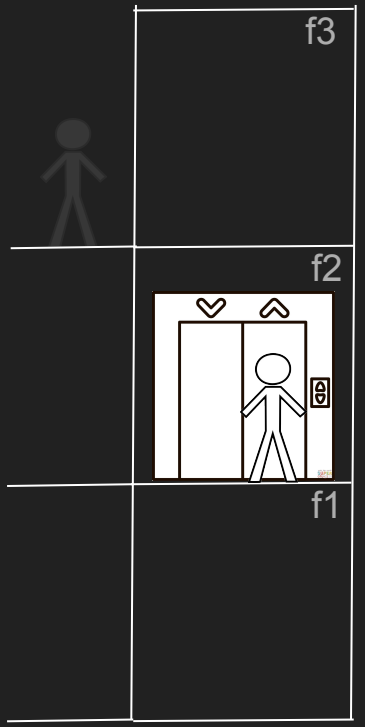


Plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f3)
4. debark(p, f3)

1. HTN Reminder

Classical planning

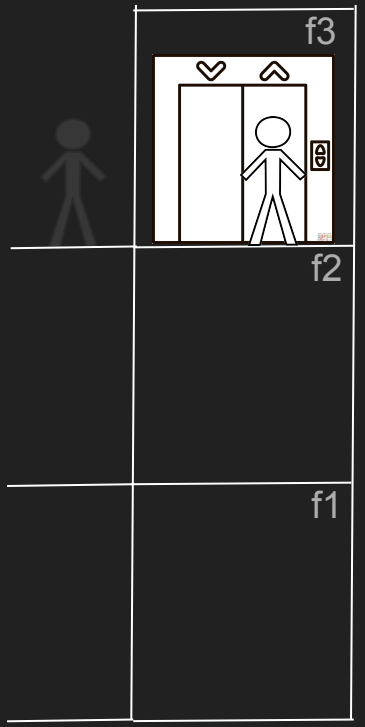


Plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f3)
4. debark(p, f3)

1. HTN Reminder

Classical planning

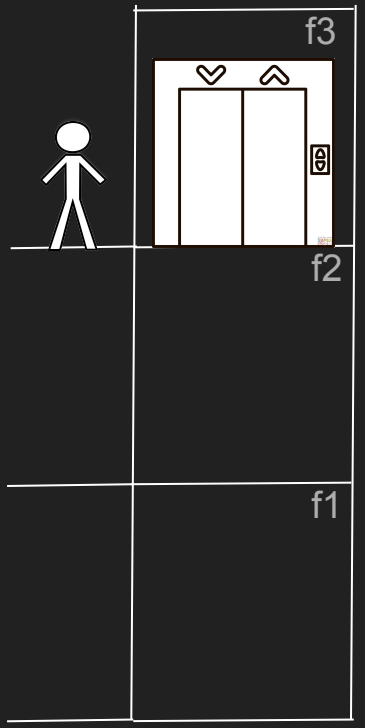


Plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f3)
4. debark(p, f3)

1. HTN Reminder

Classical planning



Plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f3)
4. debark(p, f3)

1. HTN Reminder

Classical planning

Issues with classical AI planning:

1. Inefficiency for large problems: combinatorial explosion of states
2. Lack of intuitiveness in general plans

1. HTN Reminder

Hierarchical Task network (HTN)

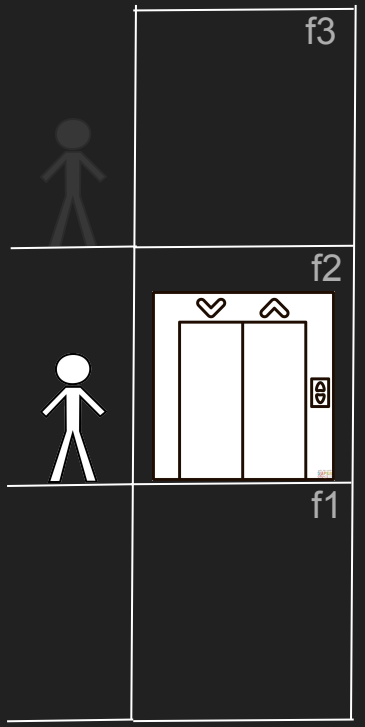
HTN: creating plans by decomposing tasks into smaller, more manageable subtasks.

- **Abstract Task**: A high-level goal that needs to be achieved. It's not directly executable but provides a broad objective.
- **Methods**: These are ways to decompose abstract tasks into subtasks.
- **Primitive Task**: They are directly executable actions.

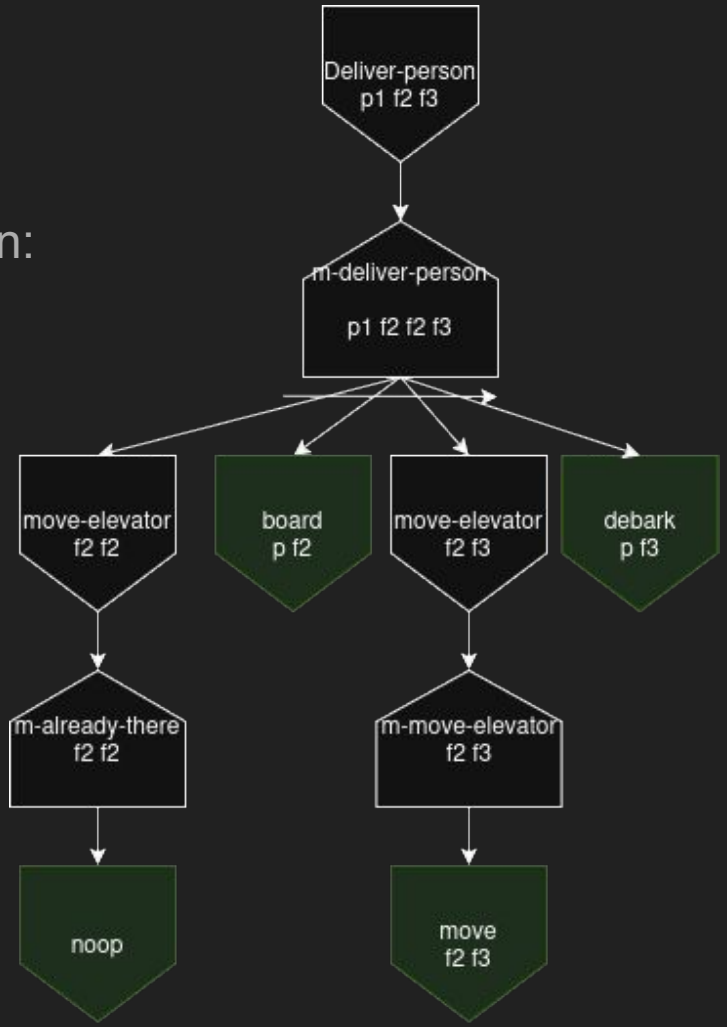
```
;; Task to move a person from its initial floor to its target floor
task (deliver-person ?p - person ?f1 - floor ?f2 - floor)
task (move-elevator ?f0 - floor ?f1 - floor)

: method method-already-destination
: parameters ( ?p - person ?f1 - floor ?f2 )
: task ( deliver-person ?p ?f1 ?f2 )
: precondition (?f1 == ?f2)
: ordered-subtasks ()

: method method-deliver-person
: parameters ( ?p - person ?f0 - floor ?f1 - floor ?f2 - floor )
: task ( deliver-person ?p ?f1 ?f2 )
: precondition ( lift-at ?f0 and  ?f1 != ?f2))
: ordered-subtasks        ;; Each subtask of a method can be an action or a task
(move-elevator ?f0 ?f1)  ;; Move the elevator to the person floor
( board ?p ?f1 )              ;; Board the person
(move-elevator ?f0 ?f1)  ;; Move the elevator to the person target floor
( debark ?p ?f2 )             ;; Debark the person
```

1. HTN Reminder

Hierarchical Task network (HTN)



Plan:



2. SAT-based HTN planners

Introduction

Encoding-based solvers are popular tools for solving AI planning problems.

Process:

- Translate a planning problem into an other formalism
  - SAT
  - CSP
  - SMT
- Use a solver highly optimized for this formalism
- If a model is found, translate it into a plan

2. Introduction to SAT solvers

What are SAT solvers ?

Boolean formula:

$(A \vee B) \wedge (\neg A \vee C)$

SAT solver

SAT!

A = false
B = true
C = true

$A \wedge \neg A$

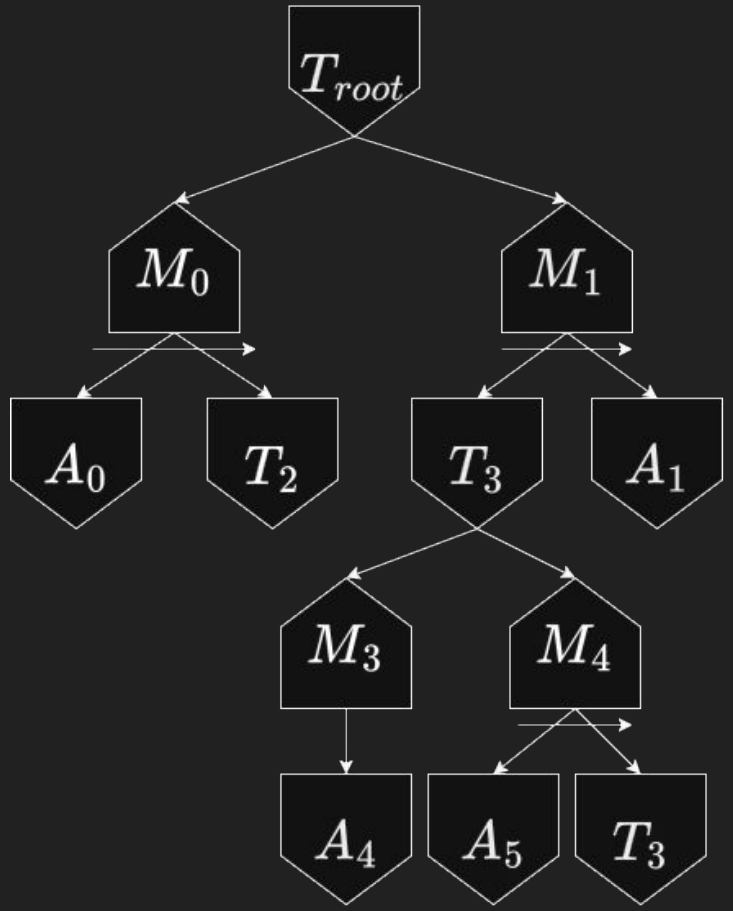
UNSAT!

2. SAT-based HTN planners

Classical approach

Search space used by SAT based planner: Path decomposition Tree

- Structure can be expanded infinitely for recursive domain
- Search space is a subtree of the infinite structure



2. SAT-based HTN planners

Classical approach

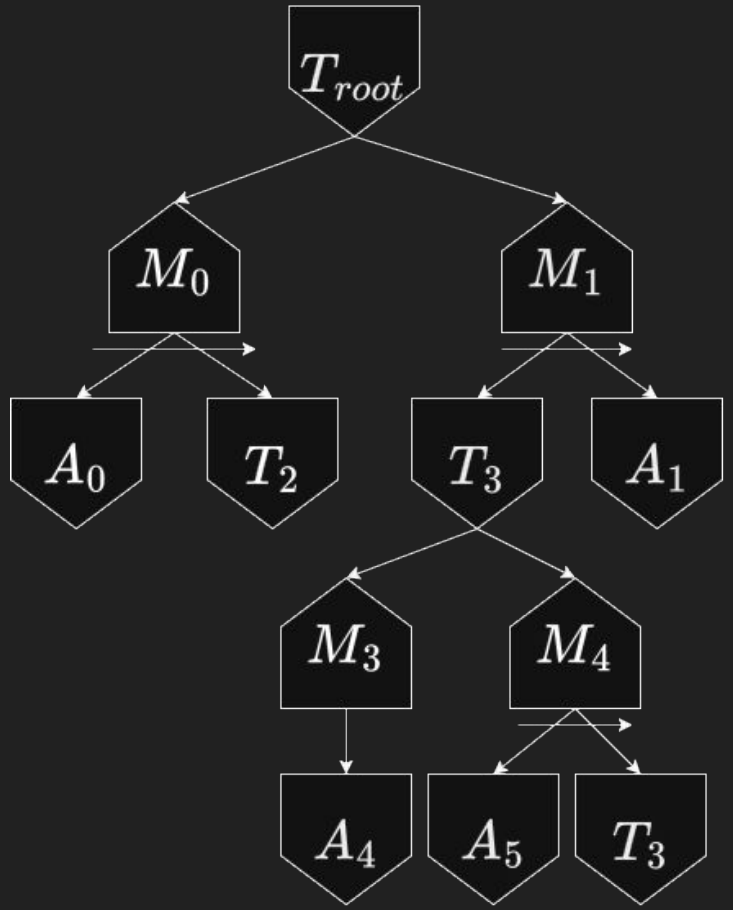
Encoding Path decomposition tree into formula:

1st: Add timestamp for all tasks/methods.

Then:

- Initial abstract task must be true
- Initial state at timestamp 1
- Methods at timestamp t:
  - precondition at time step t
- Actions at timestep t:
  - precondition at time step t
  - effects at time step t+1
- Hierarchy must be encoded
- Leafs must only consist of actions

…

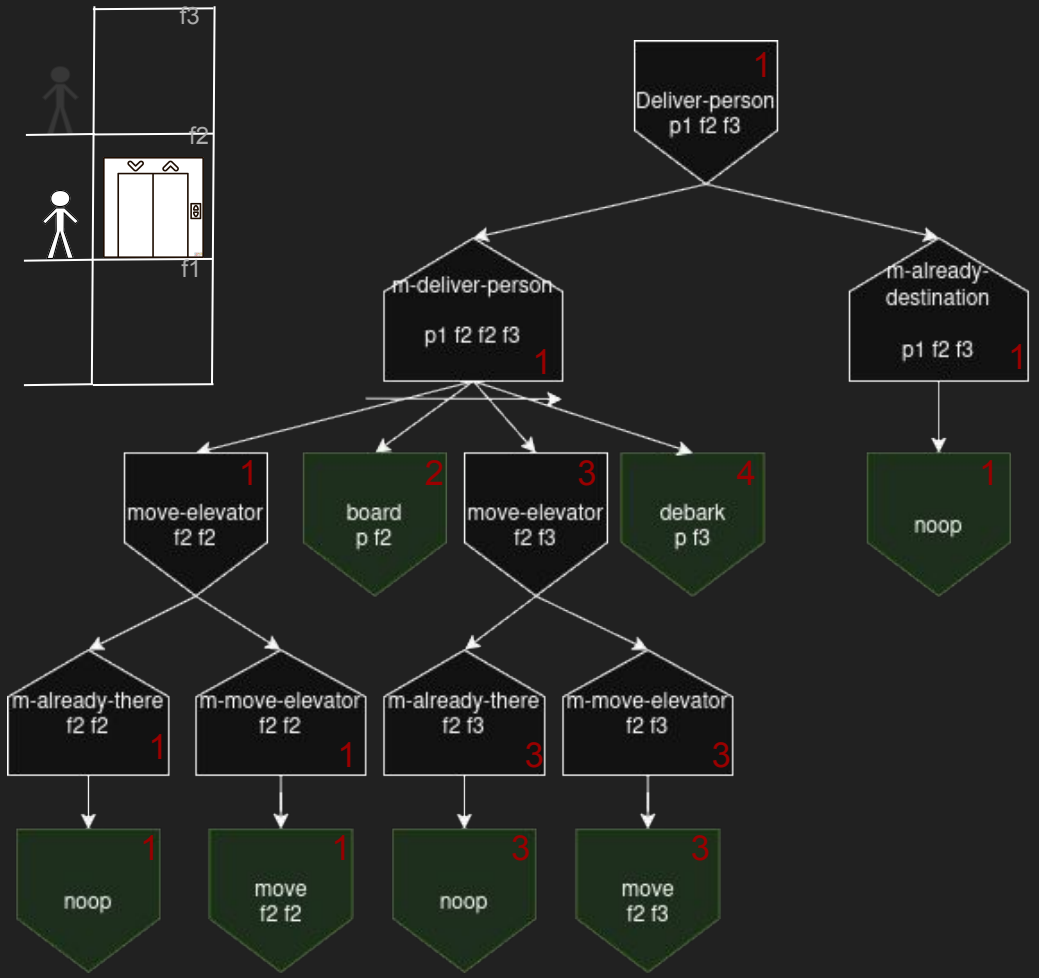


2. SAT-based HTN planners

Classical approach

Encoding Path decomposition tree into formula:

;; Initial abstract task must be true:
Deliver_person_p1_f2_f3__1 = true

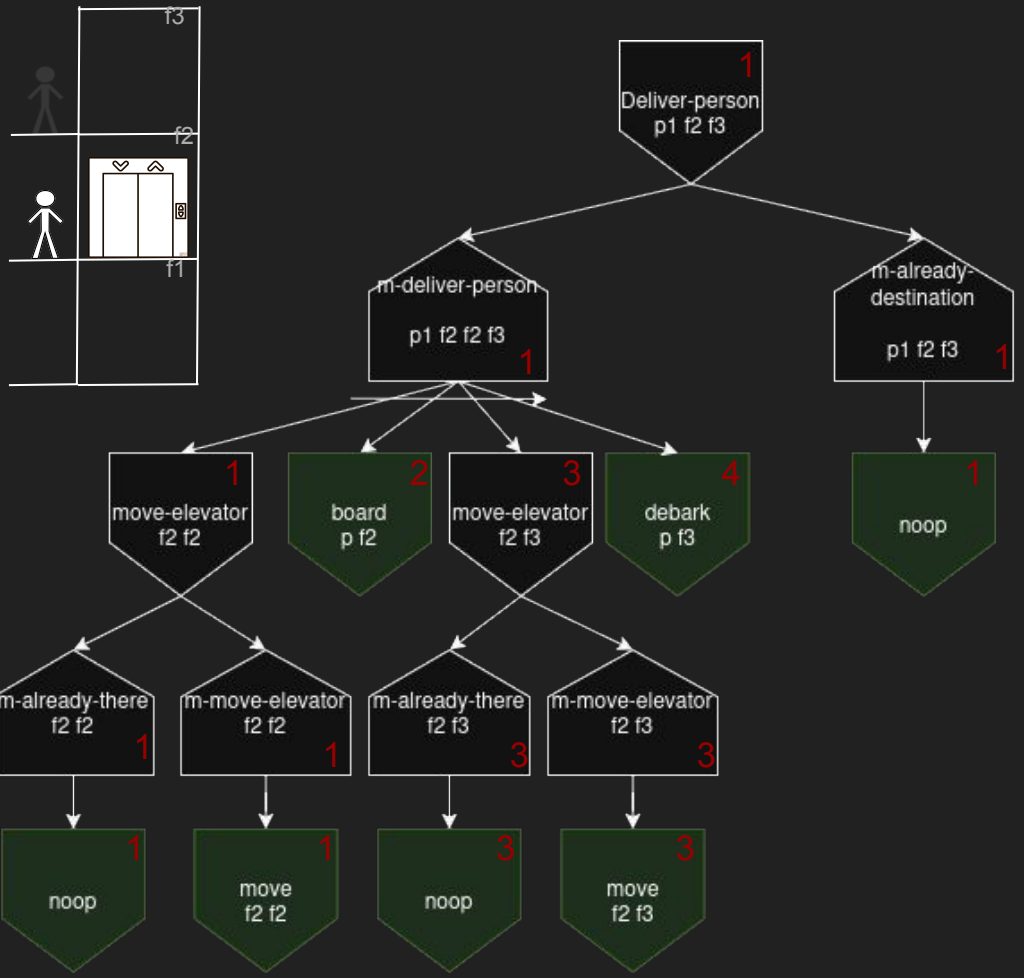


2. SAT-based HTN planners

Classical approach

Encoding Path decomposition tree into formula:

;; Initial abstract task must be true:
Deliver_person_p1_f2_f3__1 = true
;; Initial state
Lift_at_f2__1 = true
At_p_f2__1 = true
…

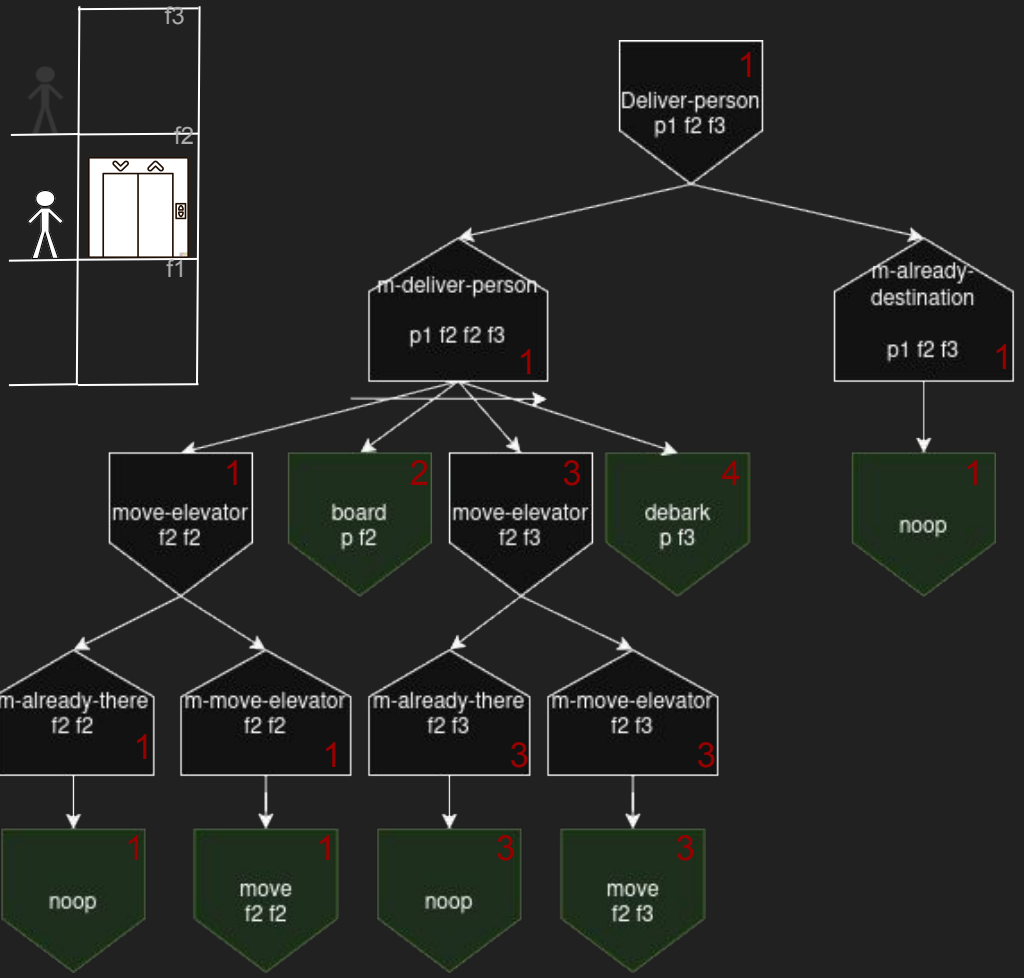


2. SAT-based HTN planners

Classical approach

Encoding Path decomposition tree into formula:

```
;; Initial abstract task must be true:
Deliver_person_p1_f2_f3__1 = true
;; Initial state
Lift_at_f2__1 = true
At_p_f2__1 = true
…
;; Encode hierarchy
Deliver_person_p1_f2_f3__1 =>
        M-deliver-person_p1_f2_f2_f3__1 XOR
        M-already-destination_p1_f2_f3__1
M-deliver-person_p1_f2_f2_f3__1 =>
        Move-elevator_f2_f2__1  AND
        Board_p_f2__2           AND
        Move-elevator_f2_f3__3  AND
        Debark_p_3__4
…
```

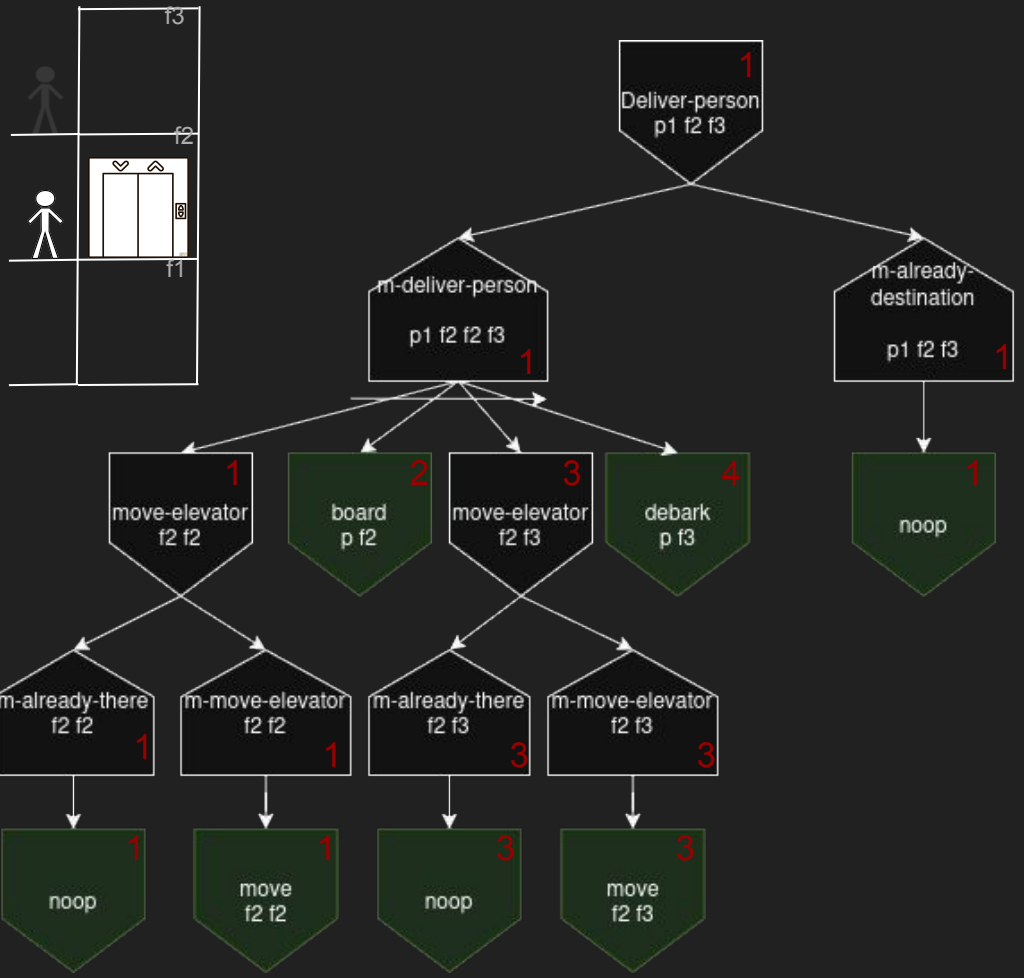


2. SAT-based HTN planners

Classical approach

Encoding Path decomposition tree into formula:

```
;; Initial abstract task must be true:
Deliver_person_p1_f2_f3__1 = true
;; Initial state
Lift_at_f2__1 = true
At_p_f2__1 = true
…
;; Encode hierarchy
Deliver_person_p1_f2_f3__1 =>
        M-deliver-person_p1_f2_f2_f3__1 XOR
        M-already-destination_p1_f2_f3__1
M-deliver-person_p1_f2_f2_f3__1 =>
        Move-elevator_f2_f2__1  AND
        Board_p_f2__2           AND
        Move-elevator_f2_f3__3  AND
        Debark_p_3__4
…
;; Precondition and effects of actions
Move_f2_f3__3 => Lift_at_f2__3
Move_f2_f3__3 => ⌐Lift_at_f2__4 AND Lift_at_f3__4
```

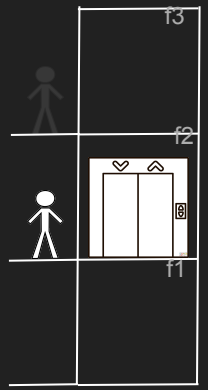


2. SAT-based HTN planners

Classical approach

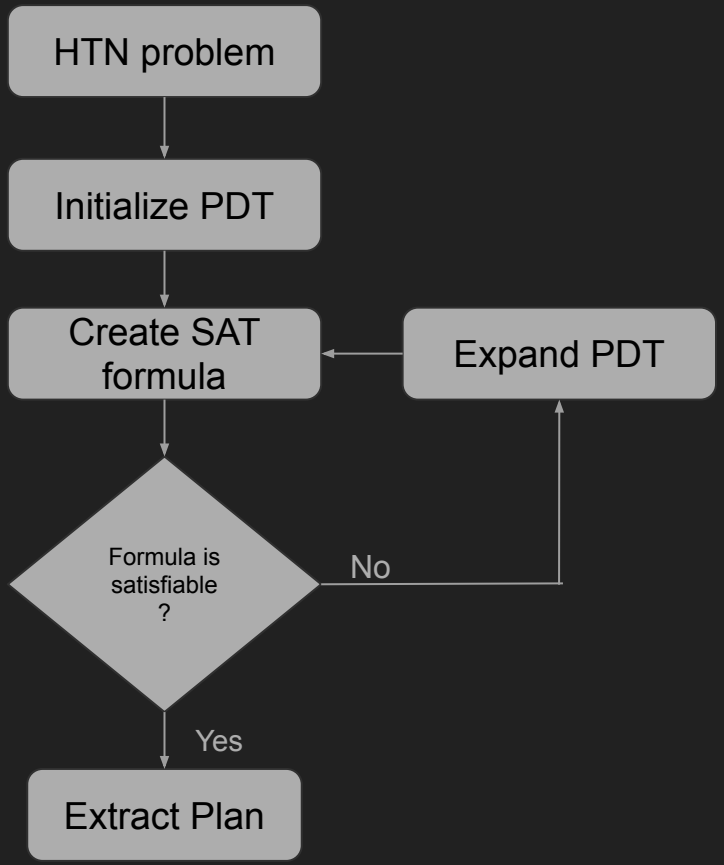
Boolean formula

SAT solver



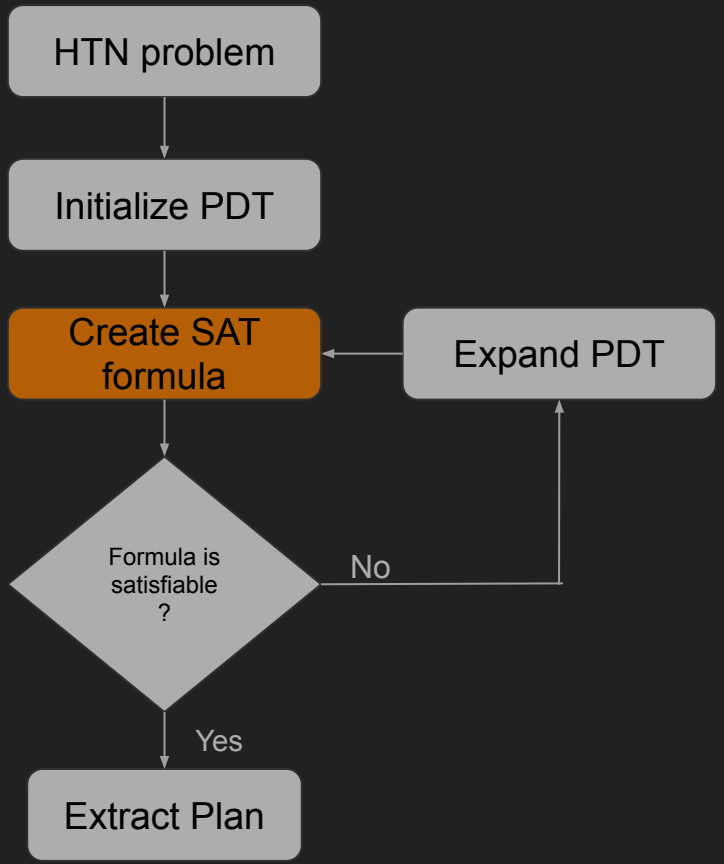
2. SAT-based HTN planners

Classical approach



2. SAT-based HTN planners

Classical approach

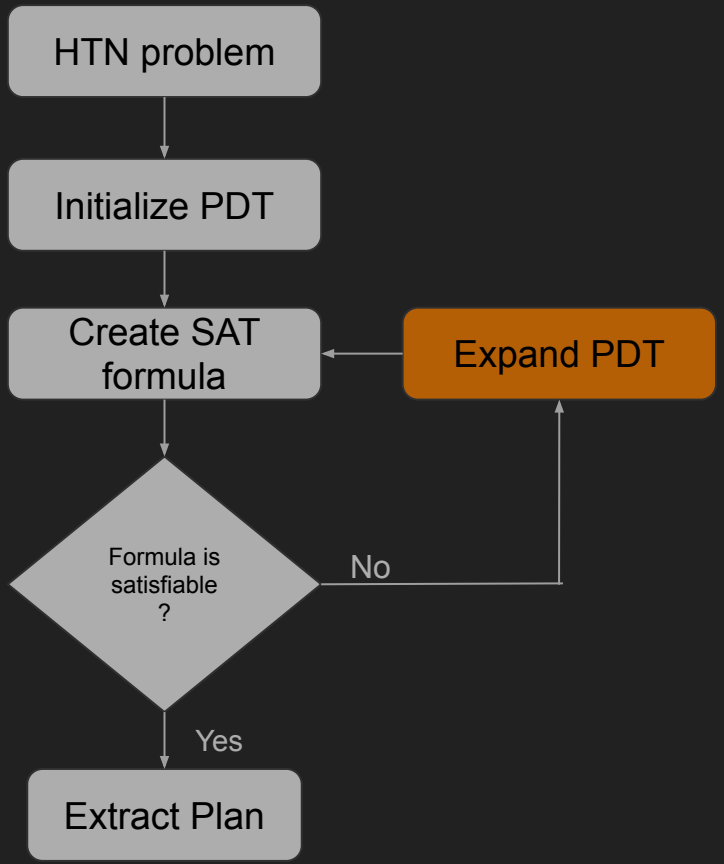


Difference between
SAT-based HTN planners ?

2. SAT-based HTN planners

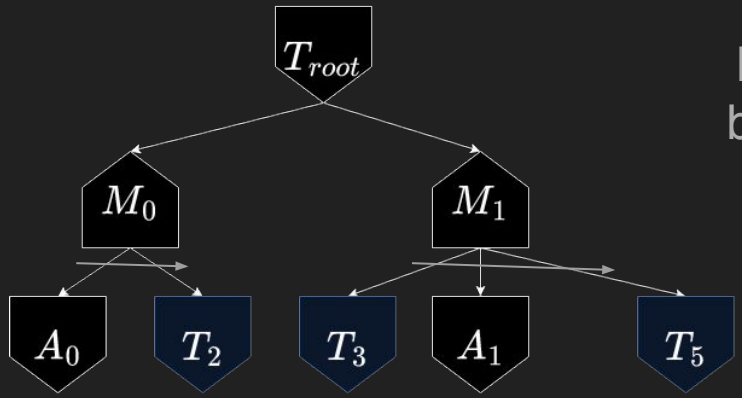
Classical approach

Expansion PDT: Breadth first search along the depth of the hierarchy

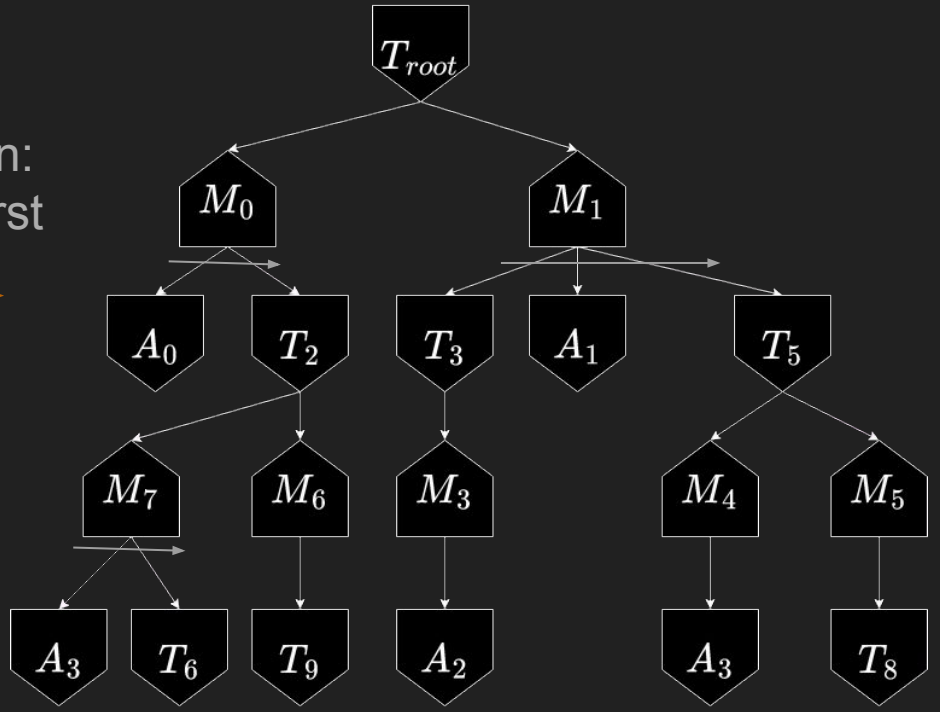


2. SAT-based HTN planners

Classical approach



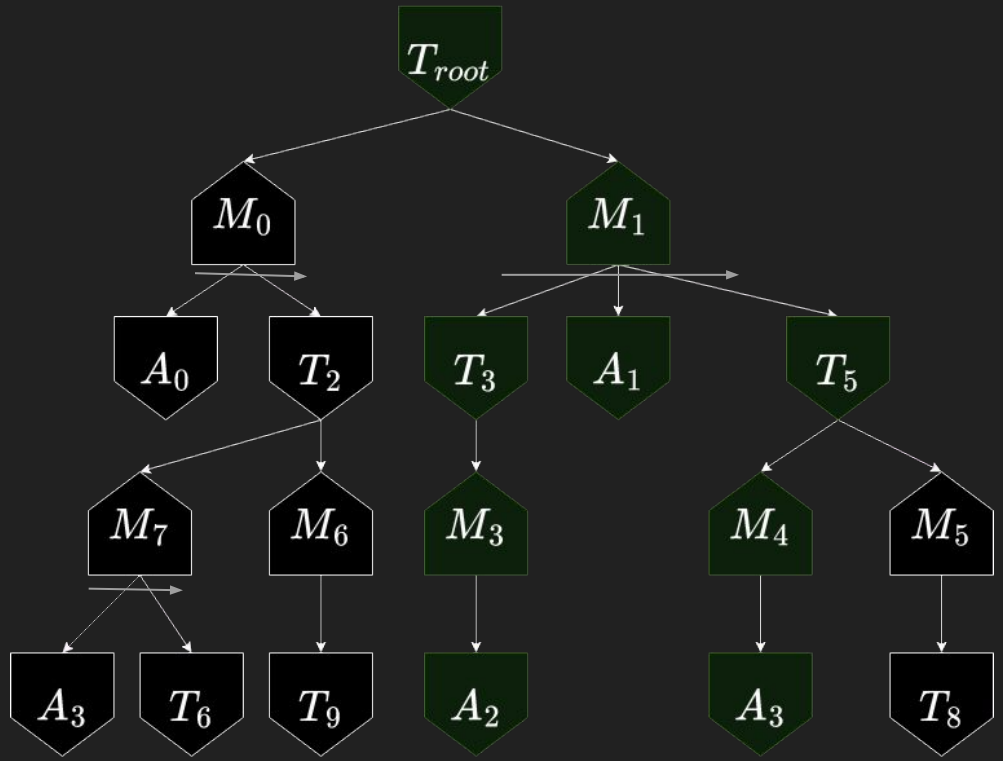
Expansion:
breadth-first



2. SAT-based HTN planners

Classical approach

Plan solution found:



2. SAT-based HTN planners

Classical approach

Problem of current SAT-based HTN planners:

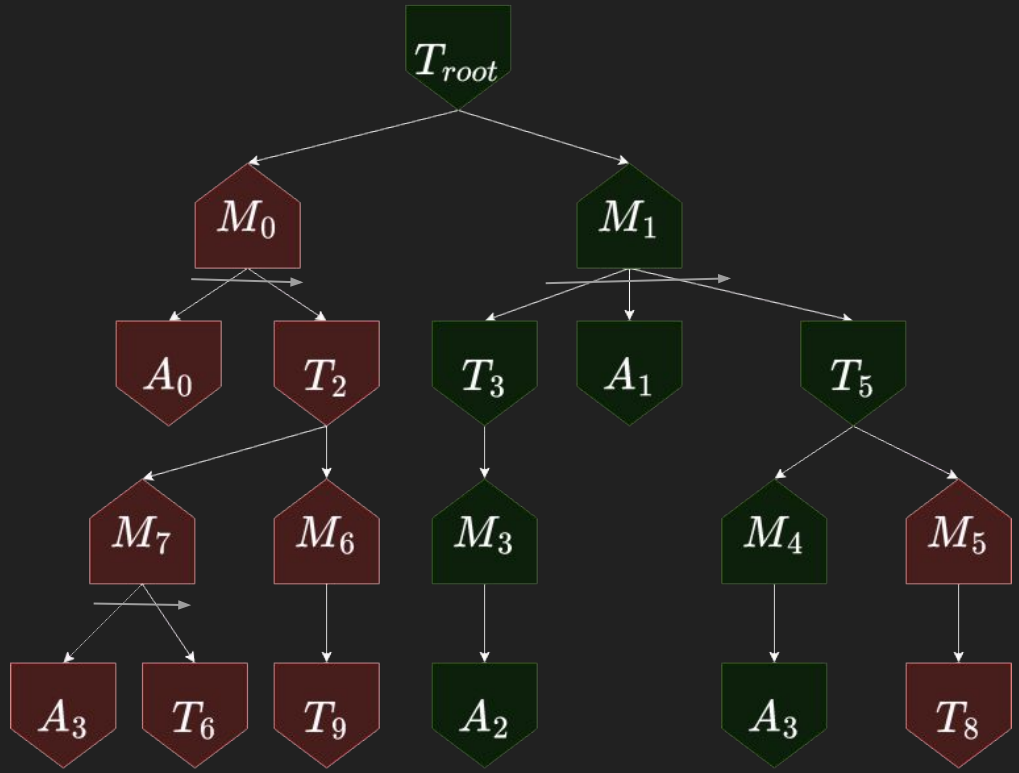
- Search is uninformed
  - Can be inefficient when dealing with large search spaces

2. SAT-based HTN planners

Classical approach

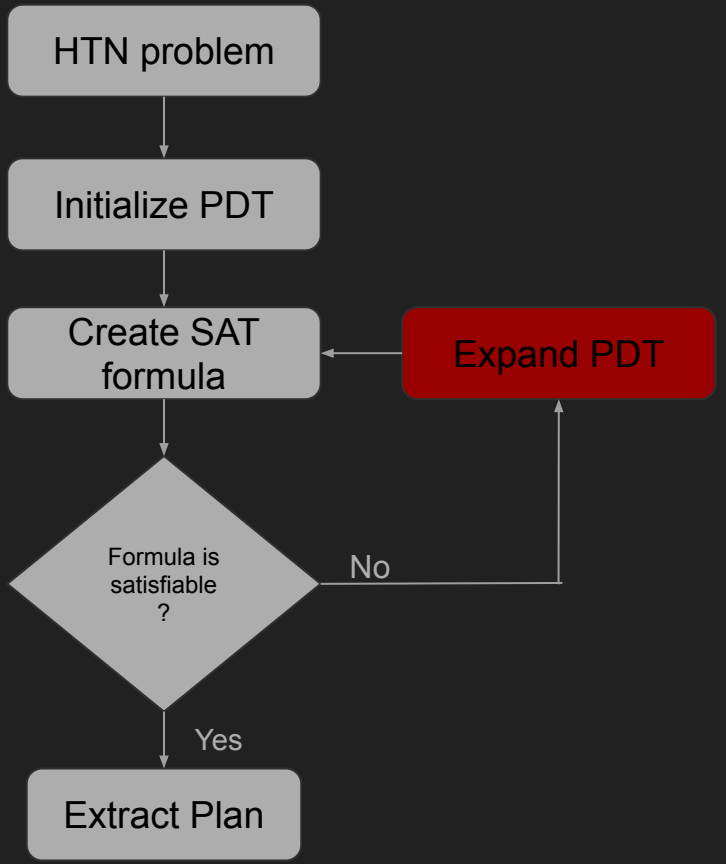
50% of search space developed is not used…

More time spend by the solver to search for a solution in a bigger search space



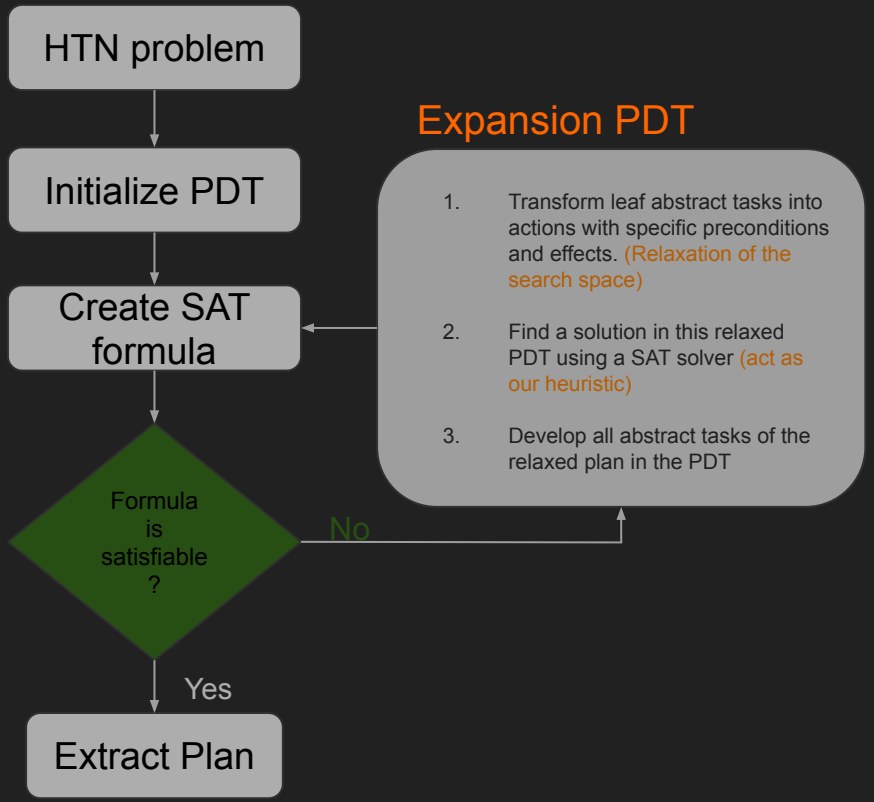
2. Encoding based solvers

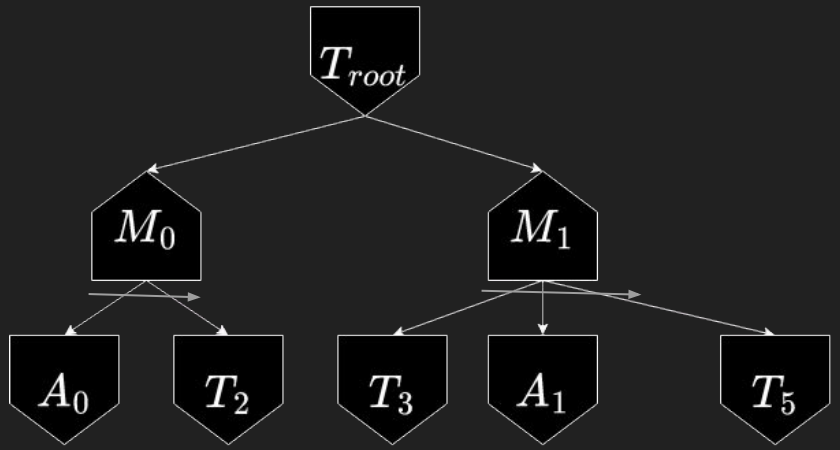
Question: Is it possible to guide the search in SAT-based planner using heuristic ?



3. SibylSAT

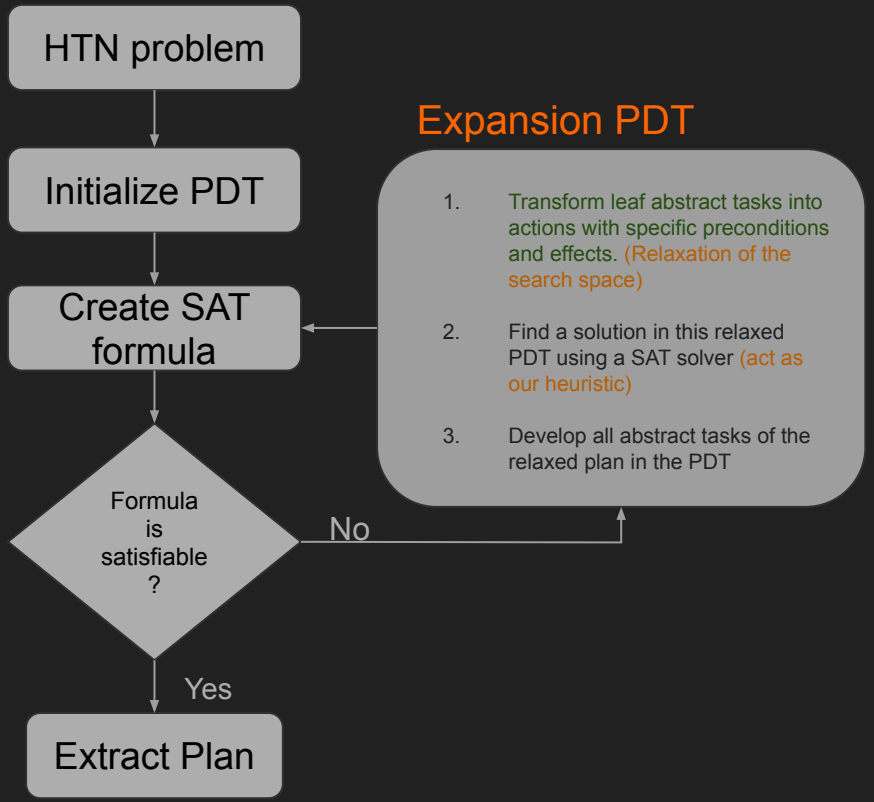
Using SAT as an oracle to perform greedy search

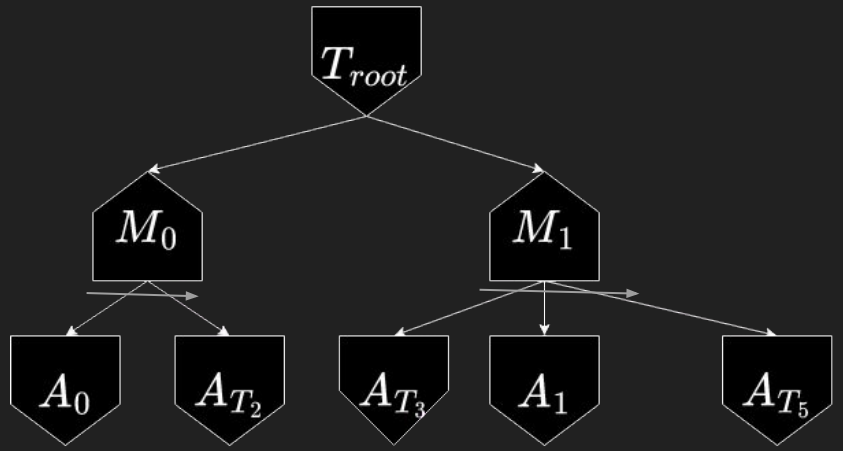




3. SibylSAT

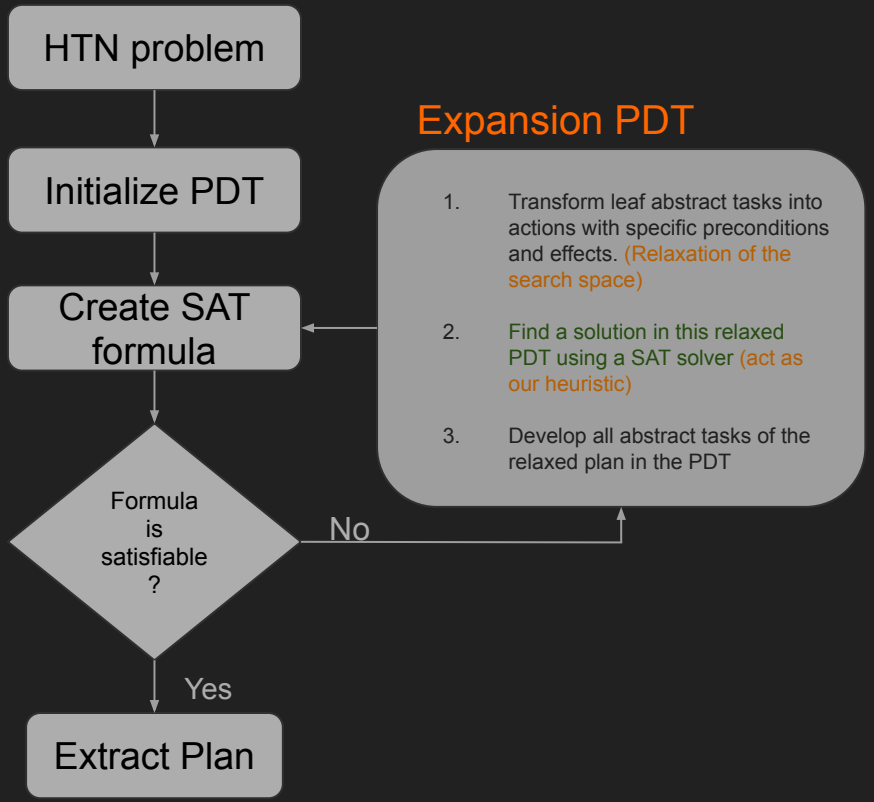
Using SAT as an oracle to perform greedy search

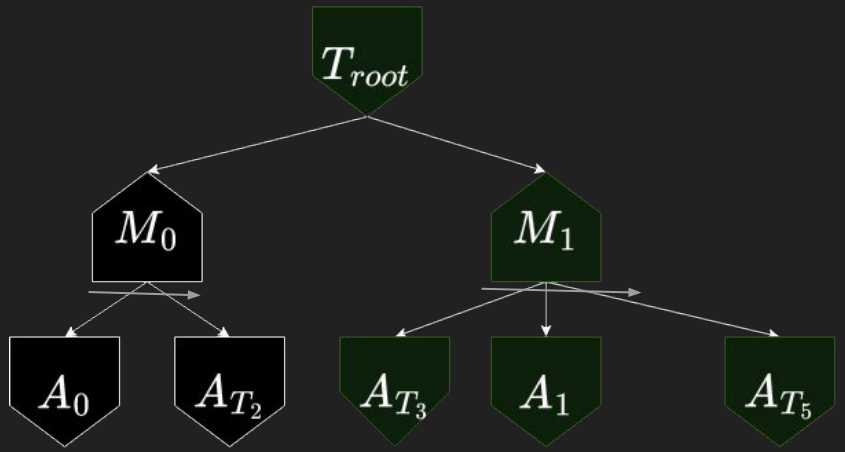




3. SibylSAT

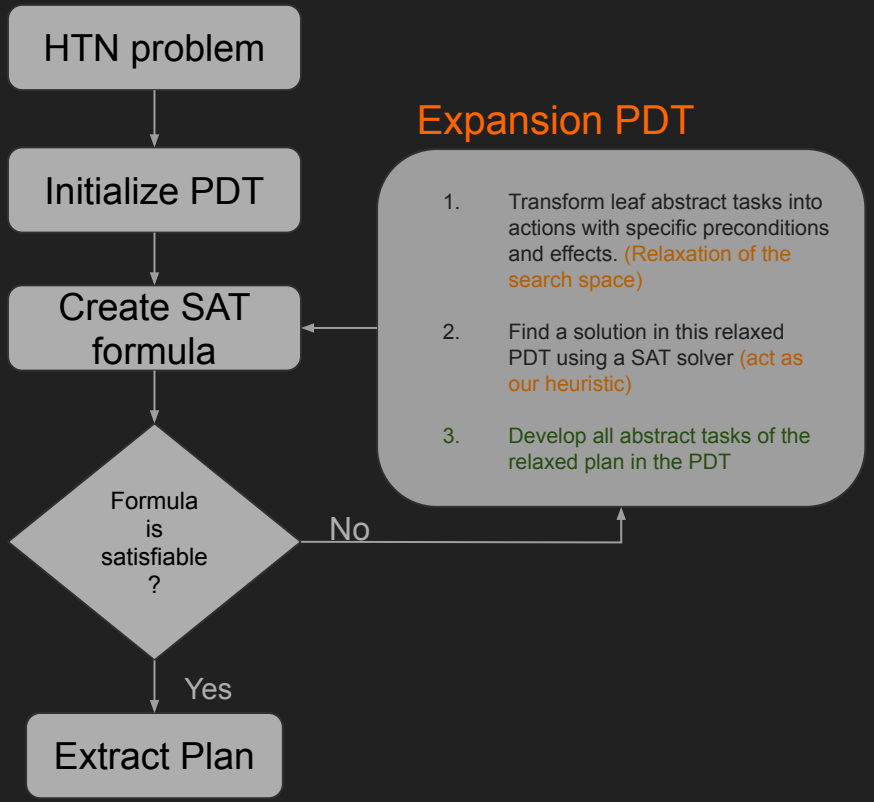
Using SAT as an oracle to perform greedy search

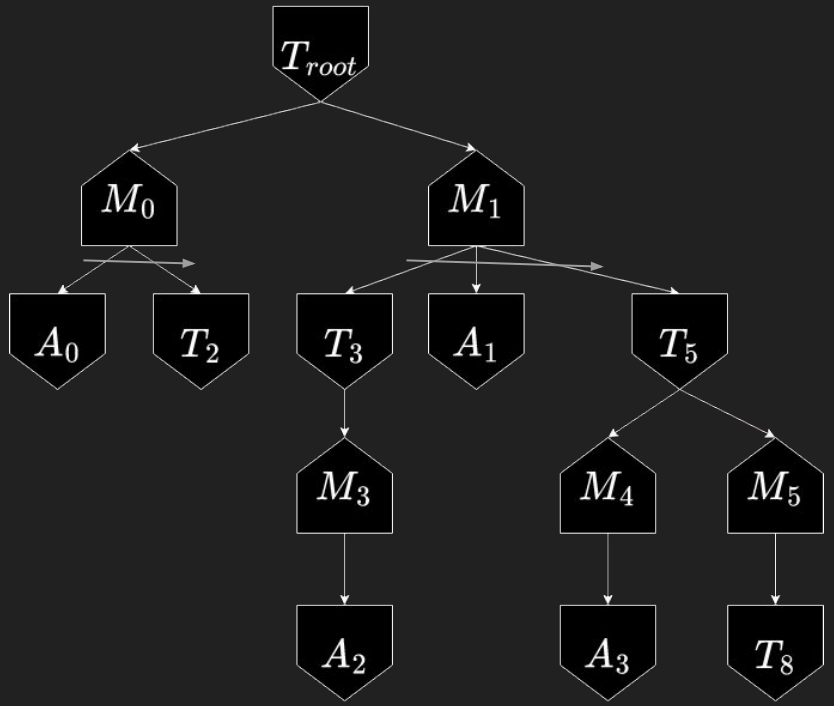




3. SibylSAT

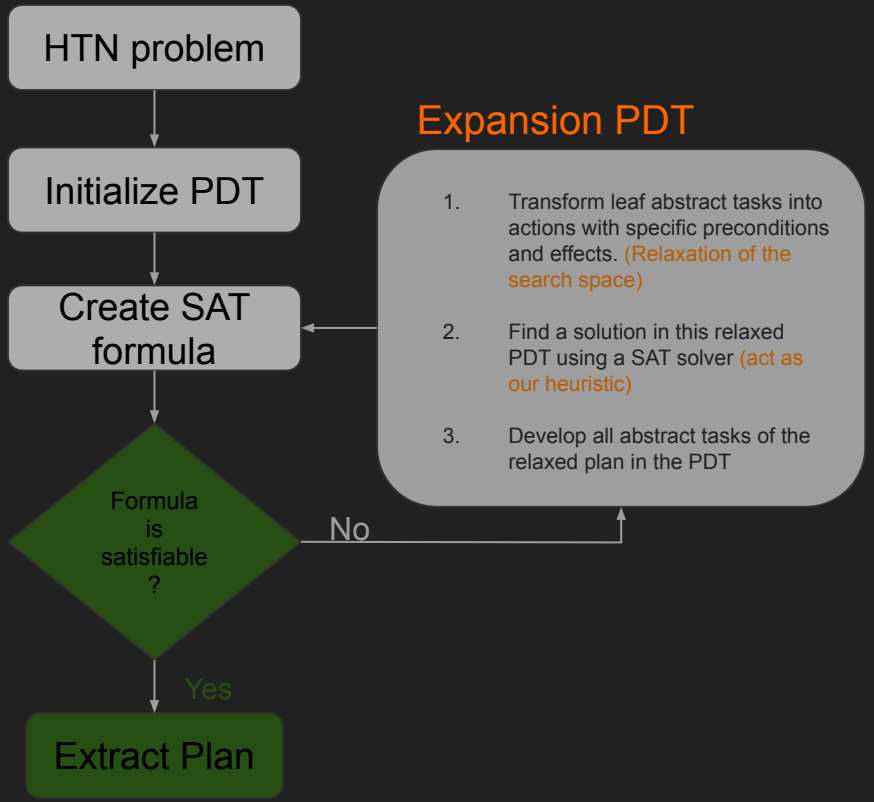
Using SAT as an oracle to perform greedy search

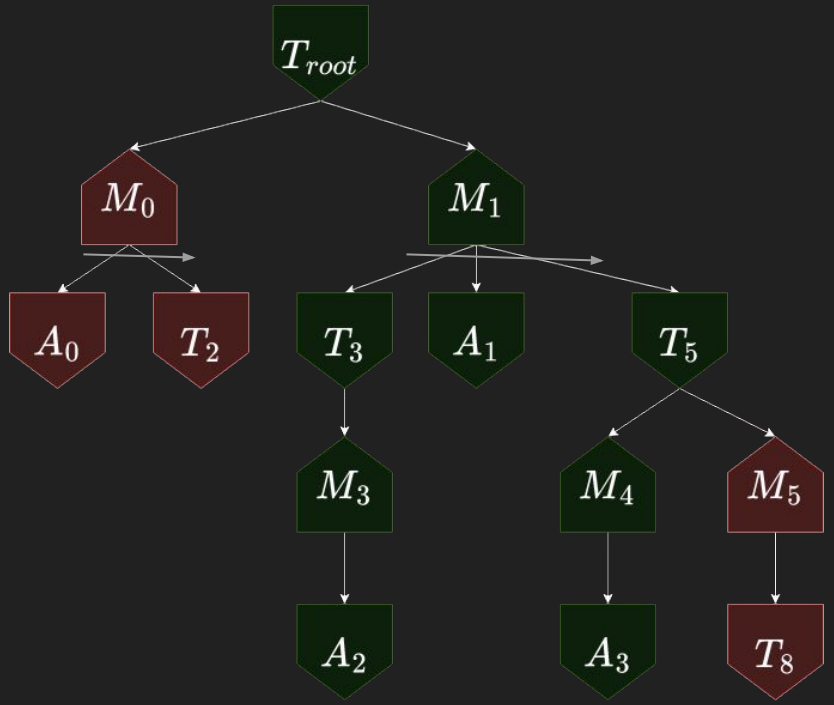




3. SibylSAT

Using SAT as an oracle to perform greedy search



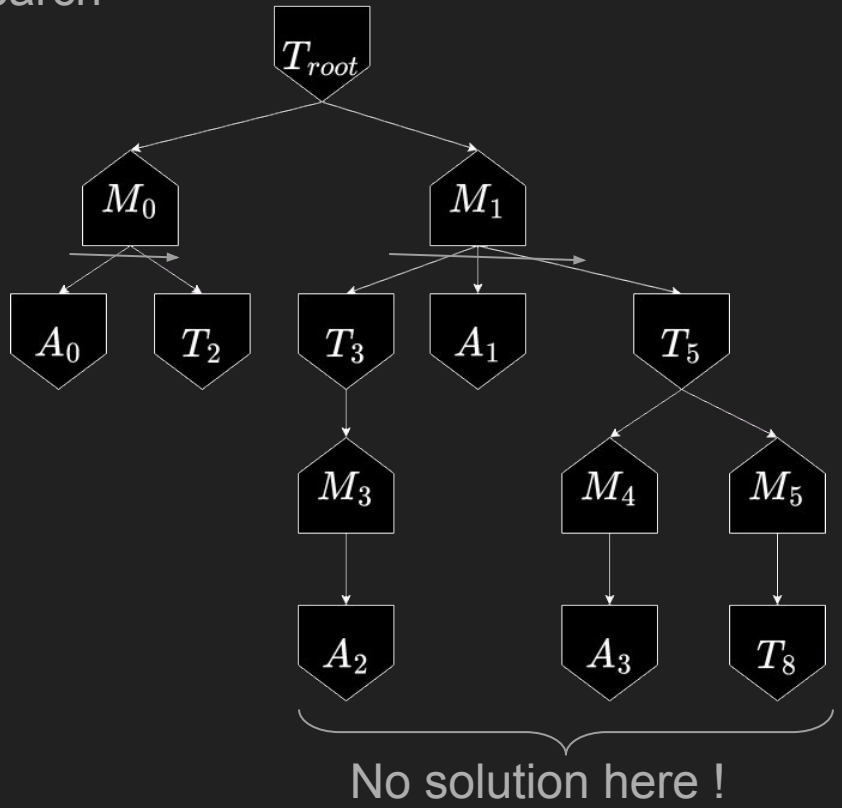


3. SibylSAT

Using SAT as an oracle to perform greedy search

1. Often multiple relaxed solutions available…

2. Relaxed solution chosen by the solver does not necessarily lead to a solution…



3. SibylSAT

Determinate preconditions and effects of abstract tasks

A lot of research on the subject:

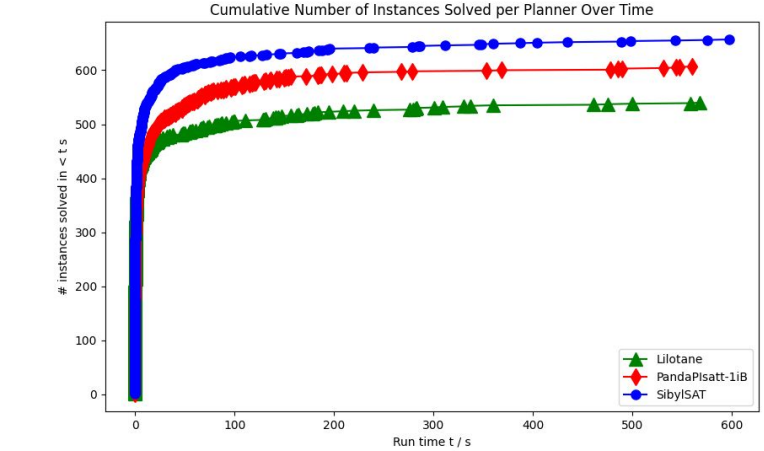
- Planners:
  - Lilotane
  - HyperTensionN
- Papers
  - Revealing Hidden Preconditions and Effects of Compound HTN Planning Tasks — A Complexity Analysis
  - A Look-Ahead Technique for Search-Based HTN Planning: Reducing the Branching Factor by Identifying Inevitable Task Refinements

3. SibylSAT

Results

Planners: state of the art SAT-based planners

- PandaPisatt-1iB (Block compression and invariant pruning for sat-based totally-ordered htn planning)
- Lilotane (Lilotane: A lifted sat-based approach to hierarchical planning)

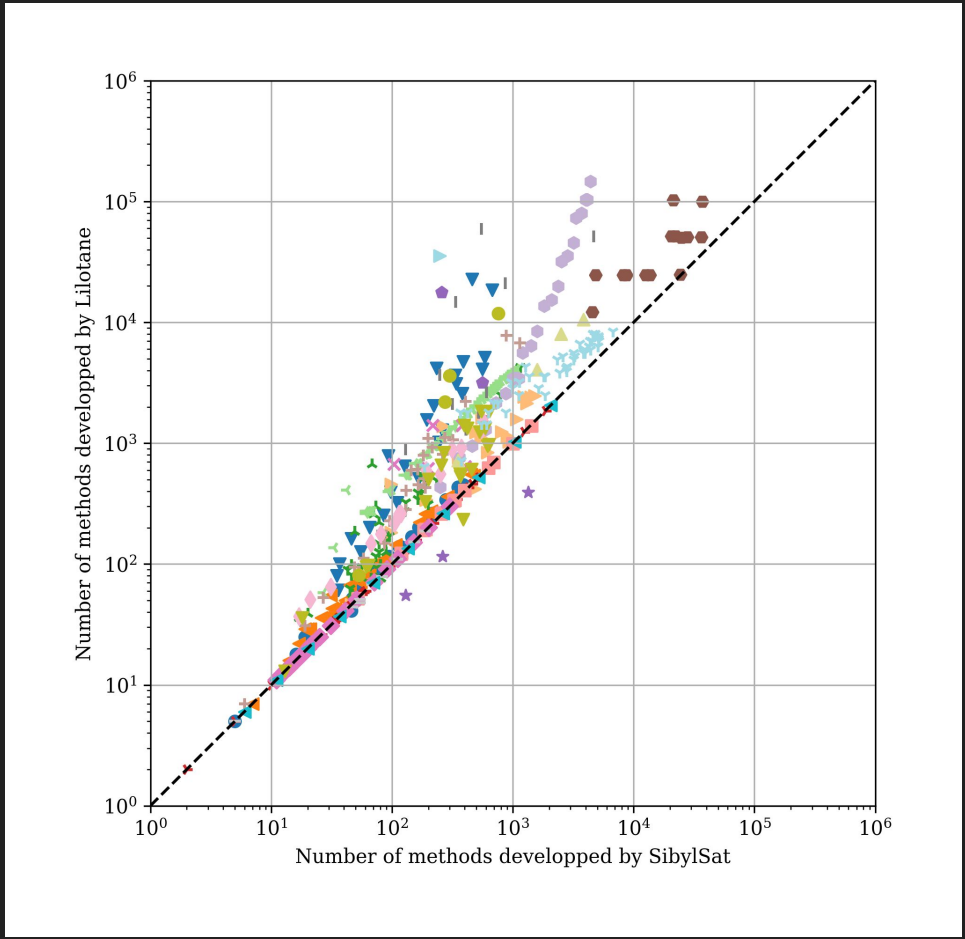


Domain		SibylSat	Lilotane	PandaPIsatt-1iB
AssemblyHierarchical	30	2.61	3.89	**4.05**
Barman-BDI	20	**16.42**	15.40	15.05
Blocksworld-GTOHP	30	**25.66**	22.56	21.11
Blocksworld-HPDDL	30	**6.74**	0.80	2.91
Childsnack	30	**27.09**	26.78	21.56
Depots	30	**25.49**	22.05	24.80
Elevator-Learned-ECAI-16	147	**146.91**	114.23	137.45
Entertainment	12	8.49	2.46	**11.53**
Factories-simple	20	**6.12**	3.77	5.99
Freecell-Learned-ECAI-16	60	**7.31**	5.26	6.25
Hiking	30	**24.87**	22.07	18.89
Lamps	30	15.40	0	**17.31**
Logistics-Learned-ECAI-16	80	**60.94**	28.48	49.00
Minecraft-Player	20	**2.66**	2.49	1.43
Minecraft-Regular	59	**32.78**	29.55	29.82
Monroe-Fully-Observable	20	18.38	**19.09**	11.56
Monroe-Partially-Observable	20	17.07	**18.21**	9.47
Multiarm-Blocksworld	74	**11.66**	1.87	9.06
Robot	20	**10.92**	10.55	10.75
Rover-GTOHP	30	**21.74**	18.05	18.54
Satellite-GTOHP	20	**14.70**	12.39	14.64
SharpSAT	21	**10.21**	8.35	8.61
Snake	20	**19.79**	19.41	16.57
Towers	20	**9.51**	8.11	5.98
Transport	40	35.05	31.18	**35.49**
Woodworking	35	28.05	**32.94**	22.01
Coverage	948	**710**	588	664
Normalized coverage	26	**19.20**	15.71	17.74
IPC score	948	**606.57**	479.94	529.83
Normalized IPC score	26	**16.23**	13.41	14.04
Quality score	948	**693.48**	525.83	606.49
Normalized quality score	26	**18.61**	14.42	16.02

3. SibylSAT

Results

Do we develop less nodes in the search space with SibylSAT ?



3. SibylSAT

https://github.com/gaspard-quenard/sibylsat

https://ebooks.iospress.nl/doi/10.3233/FAIA240987

Thank you for your attention

Questions ?

1. Background Information

Encoding-based solvers for HTN

Solver's name	Solver	Year	Technical details	Features
Encoding HTN Planning in Propositional Logic	SAT	1998	Grounded	First SAT encoding for totally ordered HTN. Not able to handle recursive problems
ASP SHOP	ASP	2003	Grounded	Based on the principles of the SHOP planner
totSAT	SAT	2018	Grounded	Based on the structure of the decomposition tree
SAT cmx	SAT	2018	Grounded	First Improvement over totSAT to handle partially ordered HTN
SAT-F	SAT	2019	Grounded	Second Improvement over totSAT to handle partially ordered HTN
TreeRex	SAT	2019	Grounded	Based on the decomposition tree. Use incremental SAT
Lilotane	SAT	2021	Lifted	Lifted version of TreeRex
PandaPIsatt	SAT	2021	Grounded	Improvement over totSAT

4. SibylSAT

Using SAT as an oracle to perform greedy search

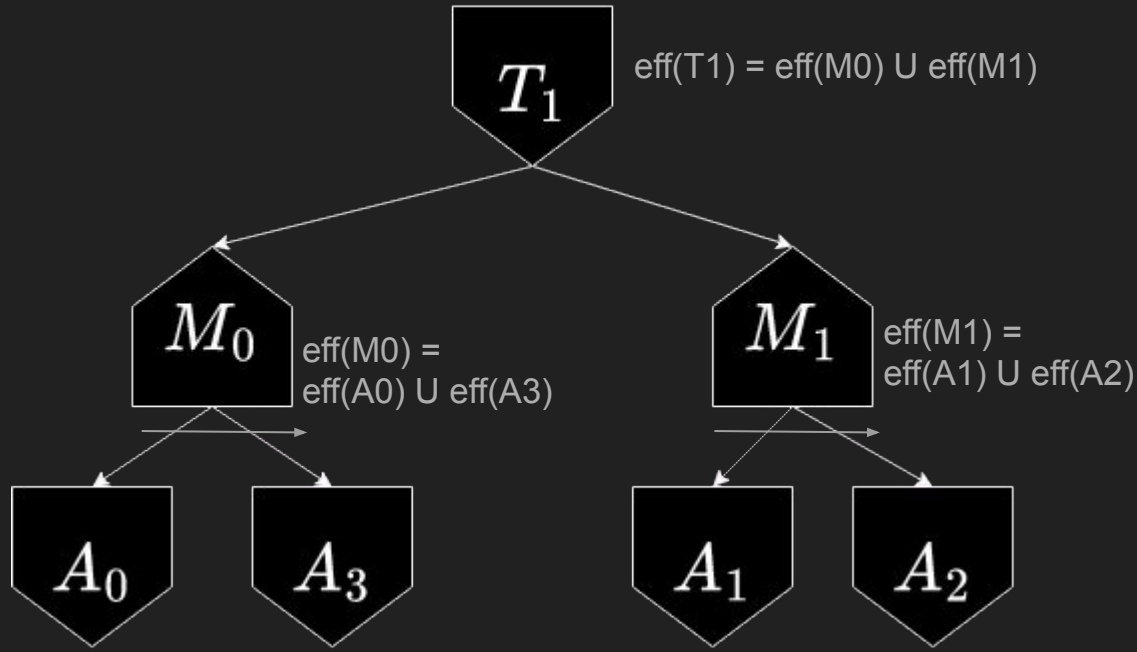
- Abstract tasks transformed into actions…
  - For all plan solution, all relaxed plans solution which can lead to this plan solution must be executable
    - Need overapproximation of effects
    - Need underapproximation of preconditions
- Relaxed solution found…
  - Does not necessarily lead to a solution…

4. SibylSAT

Determinate preconditions and effects of abstract tasks

For the effects:

- For a method: eff(m) = union effects of all subtasks
- For an abstract task: eff(t) = union effects of all methods which can accomplish task

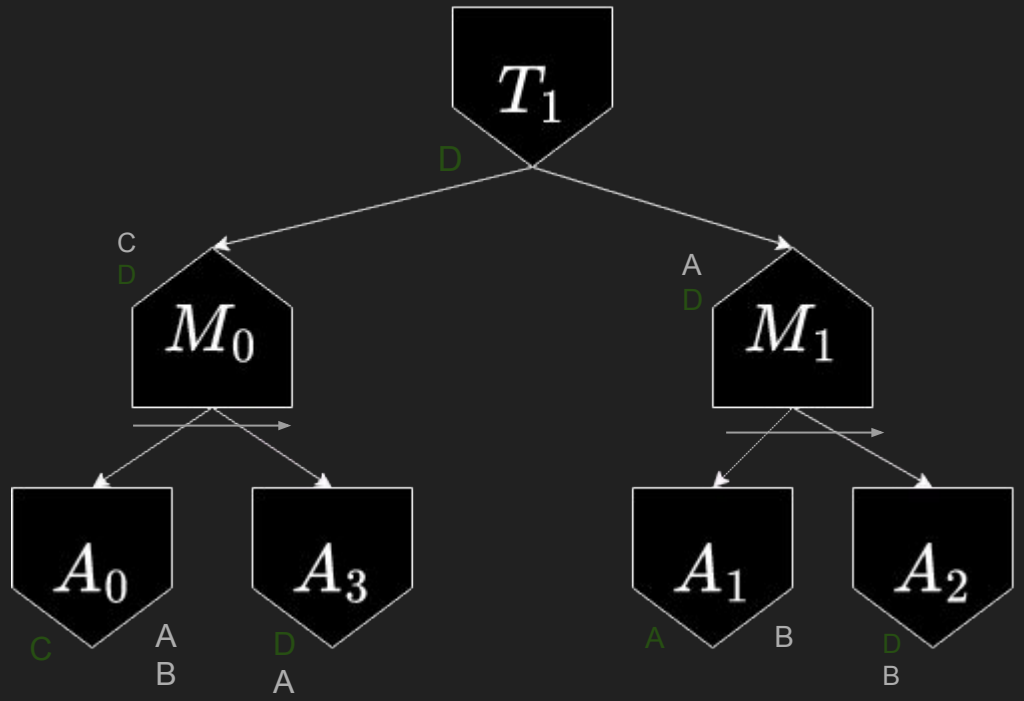


4. SibylSAT

Determinate preconditions and effects of abstract tasks

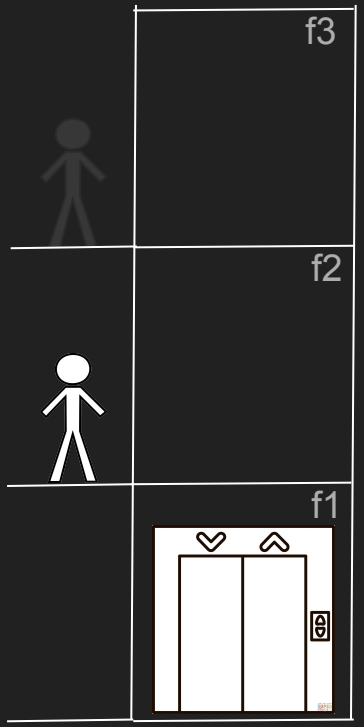
For the preconditions:

- For a method: pre(m) = All preconditions of the subtasks which cannot be accomplish by a previous subtask
- For an abstract task: Intersection of the preconditions of the methods which can accomplish the task



1. HTN Reminder

Classical planning



A non intuitive plan:

1. move(f1, f2)
2. board(p, f2)
3. move(f2, f1)
4. debark(p, f1)
5. move(f1, f3)
6. move(f3, f2)
7. move(f2, f1)
8. board(p, f1)
9. move(f1, f3)
10. debark(p, f3)