UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning

authors

Kutluhan Erol James Hendler Dana S. Nau

What is HTN?

What is HTN?

The world and basic actions are represented similar to STRIPS.

What is HTN?

The world and basic actions are represented similar to STRIPS.

HTN-planning replaces STRIPS-style goals with tasks and task networks.

Types of Tasks

Types of Tasks

Goal Tasks 

Types of Tasks

Goal Tasks 

Primitive Tasks 

Types of Tasks

Goal Tasks 

Primitive Tasks 

Compound Tasks 

Types of Tasks

Goal Tasks 

Similar to STRIP-style goals, which includes properties that we wish to make true.

Primitive Tasks 

Compound Tasks 

Types of Tasks

Goal Tasks 

Similar to STRIP-style goals, which includes properties that we wish to make true.

Primitive Tasks 

Tasks that we can directly achieve by executing corresponding action.

Compound Tasks 

Types of Tasks

Goal Tasks 

Similar to STRIP-style goals, which includes properties that we wish to make true.

Primitive Tasks 

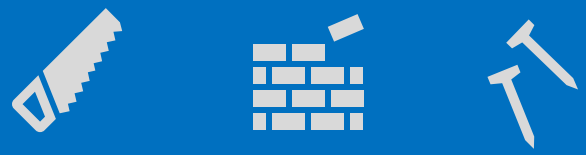
Tasks that we can directly achieve by executing corresponding action.

Compound Tasks 

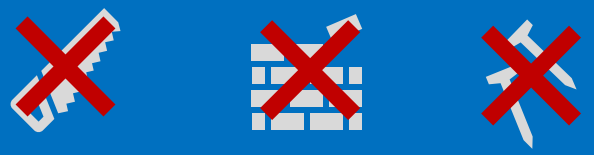
Tasks with desired changes that involve several goal tasks and primitive tasks.

Compound Tasks

Building a house



Having a house



Compound Tasks

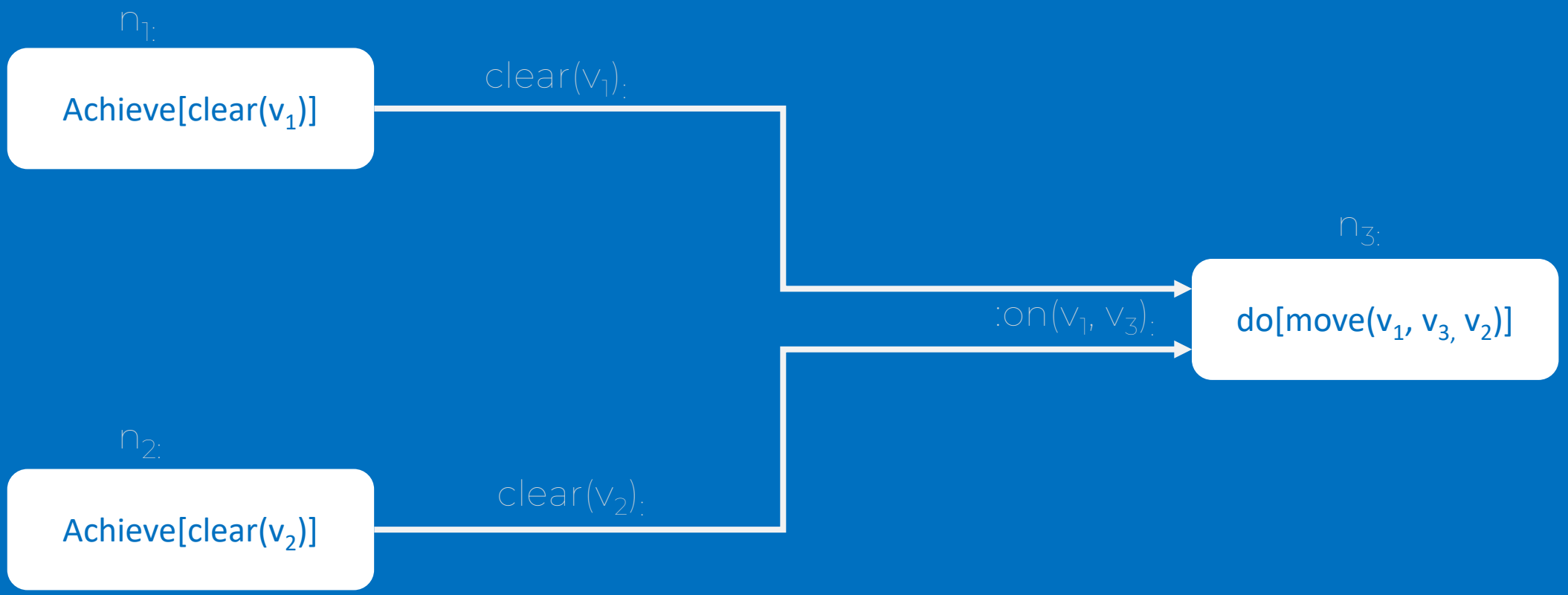
Compound Tasks

Task Networks

Task Networks

Tasks are connected together in HTN planning via the use of task networks.

Task Networks



Non-primitive task of achieve[on(v1,v2)]

Task Networks

(

$(n_1: \text{achieve}[\text{clear}(v_1)])$ $(n_2: \text{achieve}[\text{clear}(v_1)])$ $(n_3: \text{do}[\text{move}(v_1, v_2, v_3)])$

$(n_1 < n_3) \wedge (n_2 < n_3) \wedge (n1,\text{clear}(v_1),n_3)$

$\wedge(n_2, \text{clear}(v_2), n_3) \wedge (\text{on}(v_1, v_3), n_3)$

$\wedge \neg(v_1 = v_2) \wedge \neg(v_1 = v_3) \wedge \neg(v_2 = v_3)$

)

HTN Planning Procedure

HTN Planning Procedure

1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

UMCP Planning Procedure

1. Input a planning problem P = ⟨d, I, D⟩ .
2. if d is primitive, then
    1. If comp(d, I, D) 6 = ∅, return a member of it.
    2. Otherwise return FAILURE.
3. Pick a non-primitive task node (n : α) in d.
4. Nondeterministically choose a method m for α.
5. Set d := reduce(d, n, m).
6. Set Γ := τ (d, I, D).
7. Nondeterministically set d := some element of Γ.
8. Go to step 2.

HTN Planning Procedure

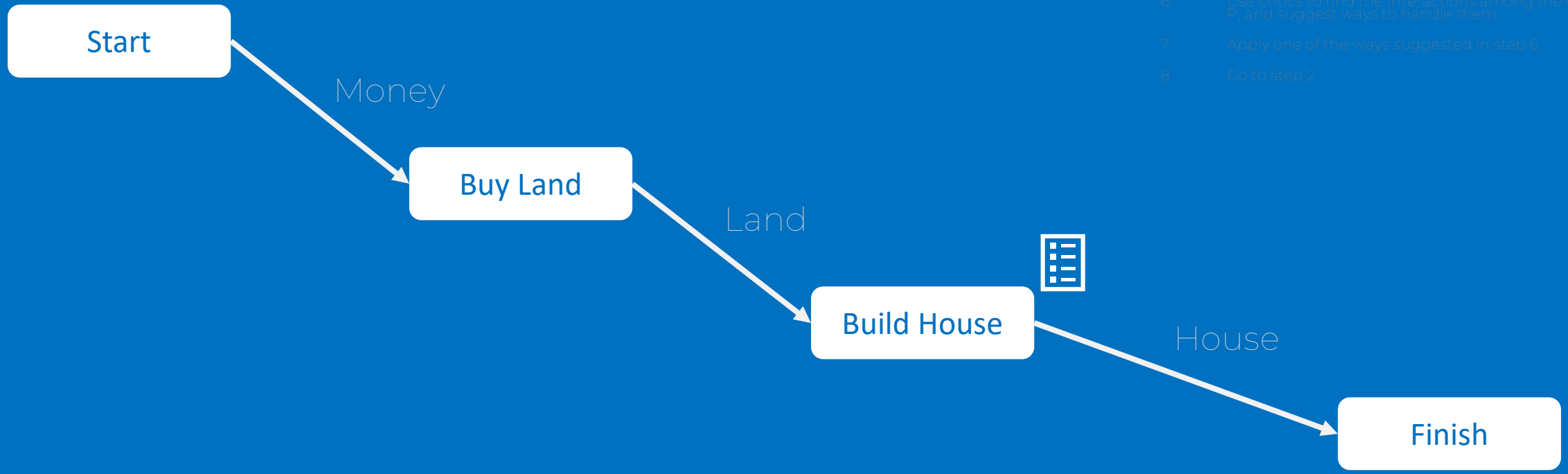
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure

1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

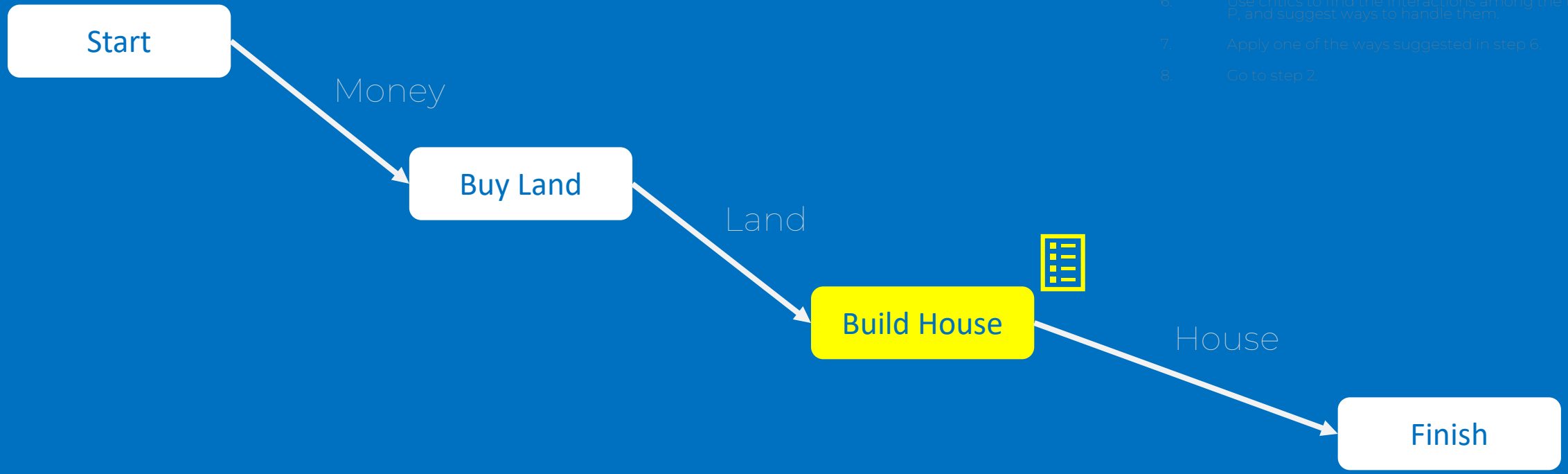
HTN Planning Procedure

1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.



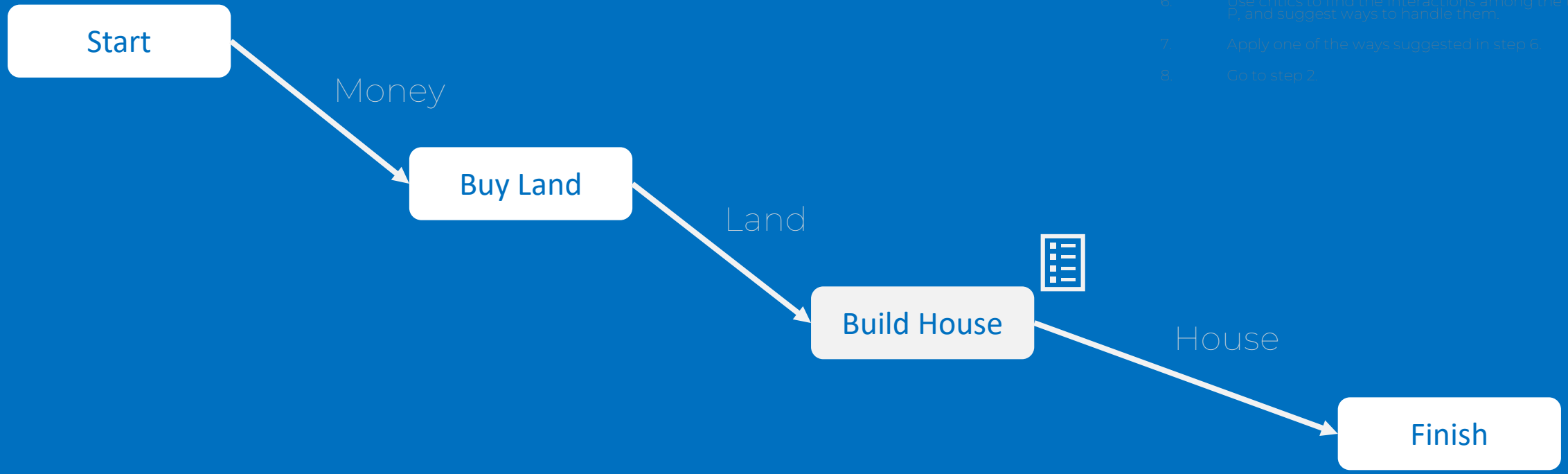
HTN Planning Procedure

1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

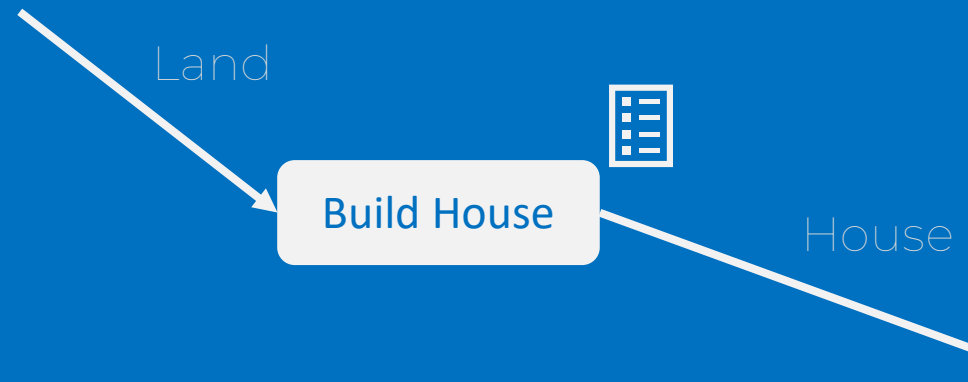


HTN Planning Procedure

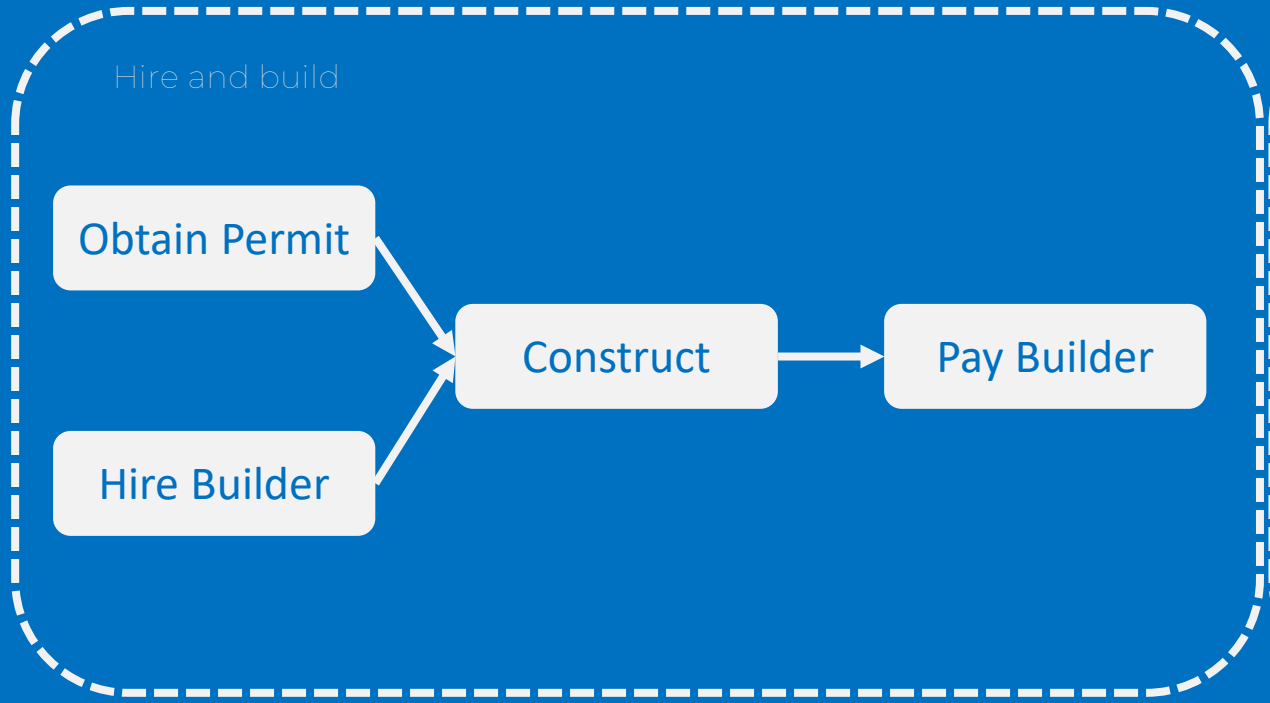
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

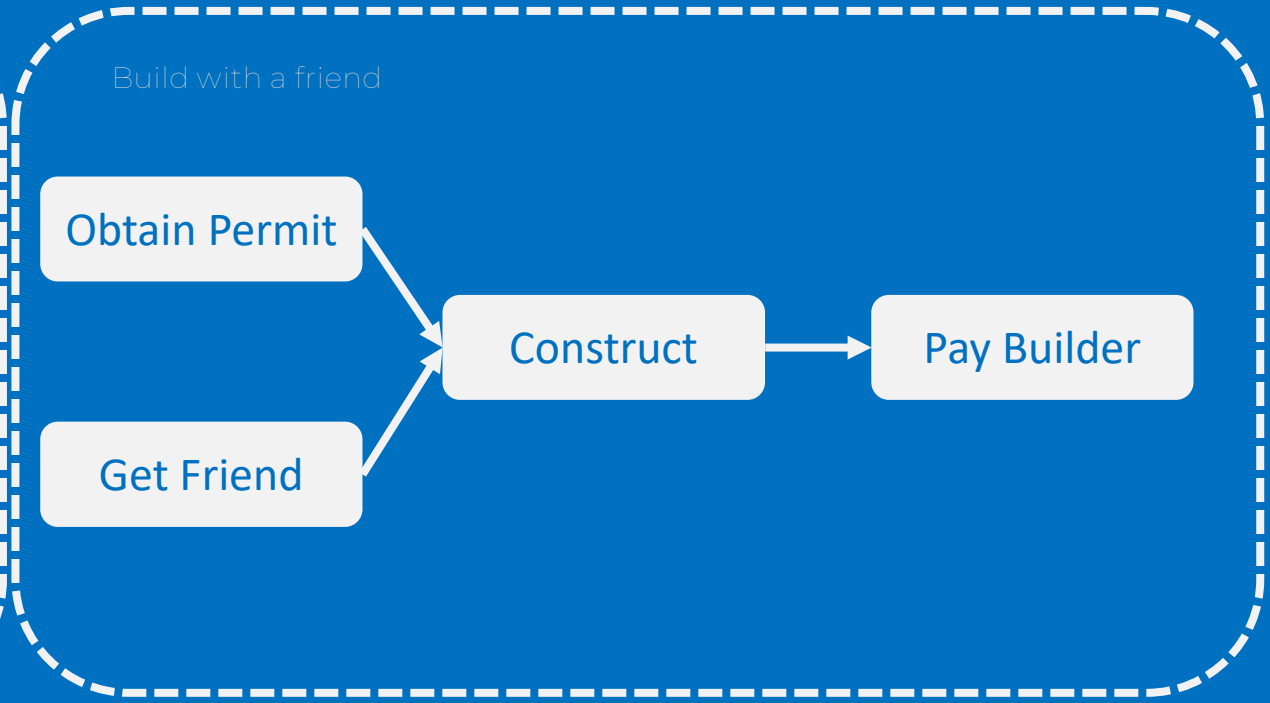


HTN Planning Procedure



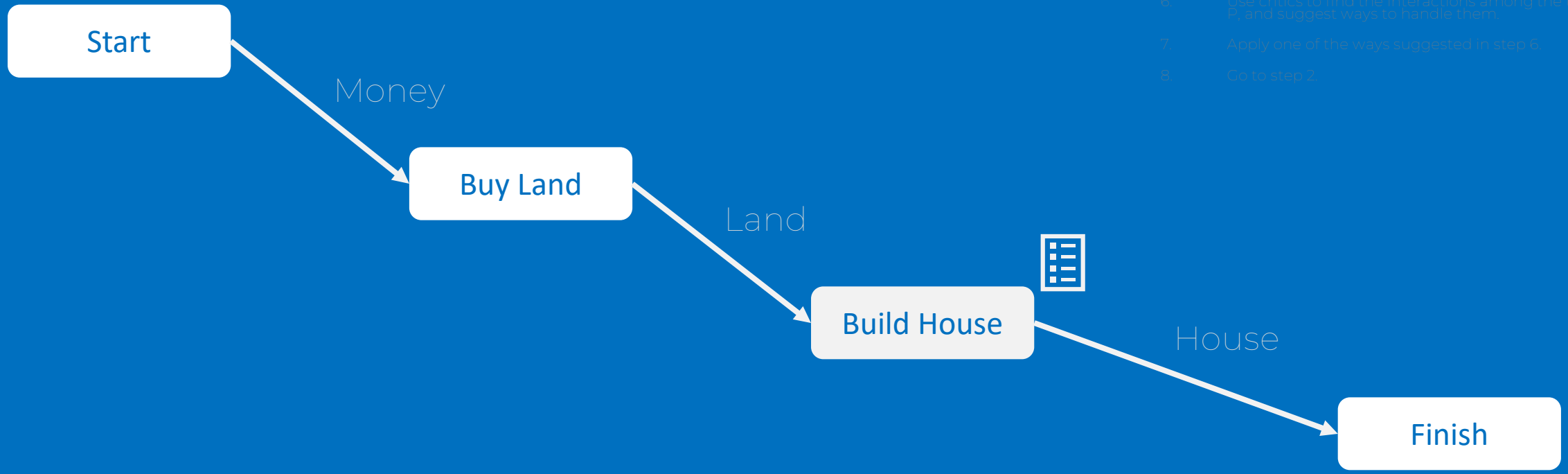
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.



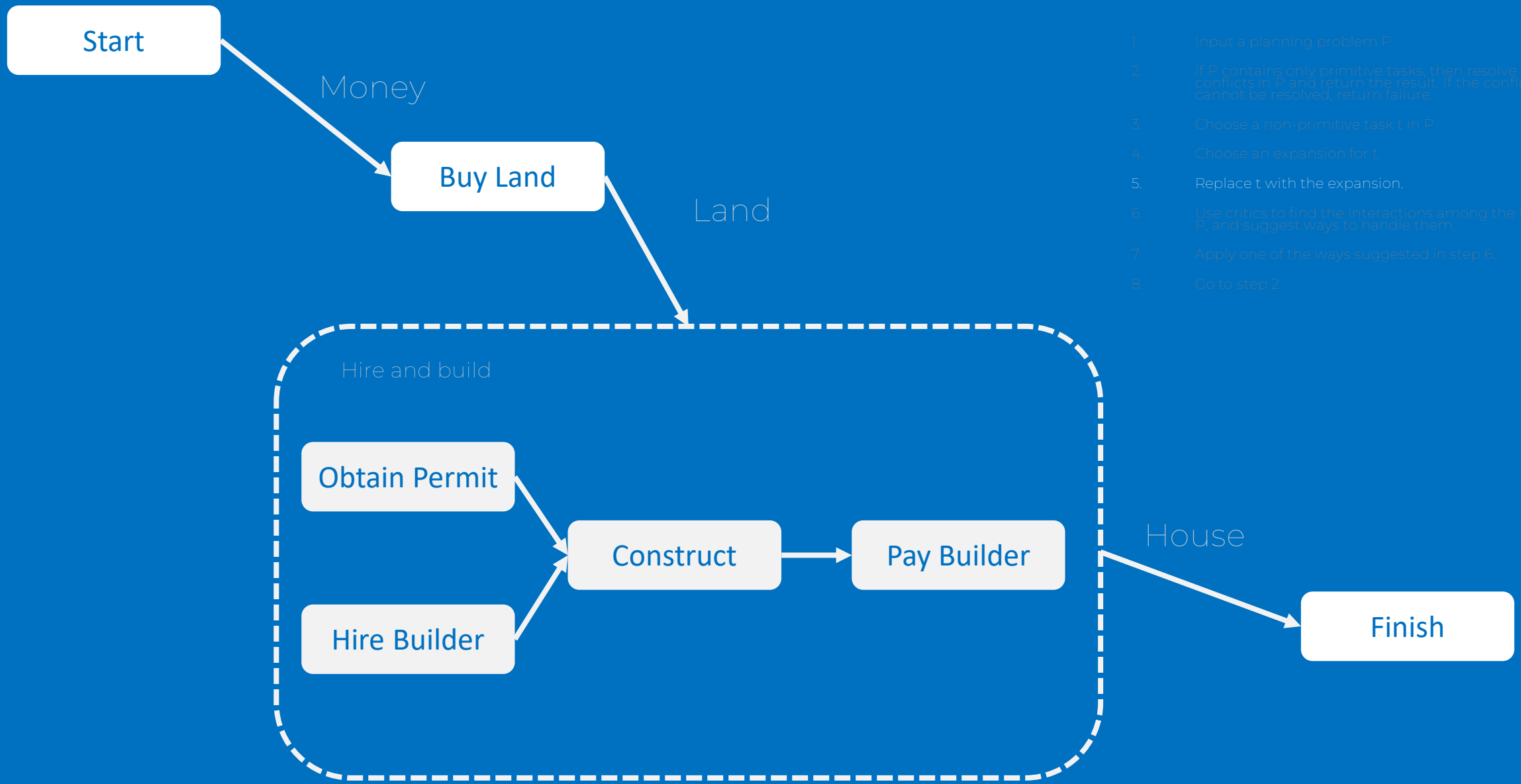


HTN Planning Procedure

1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

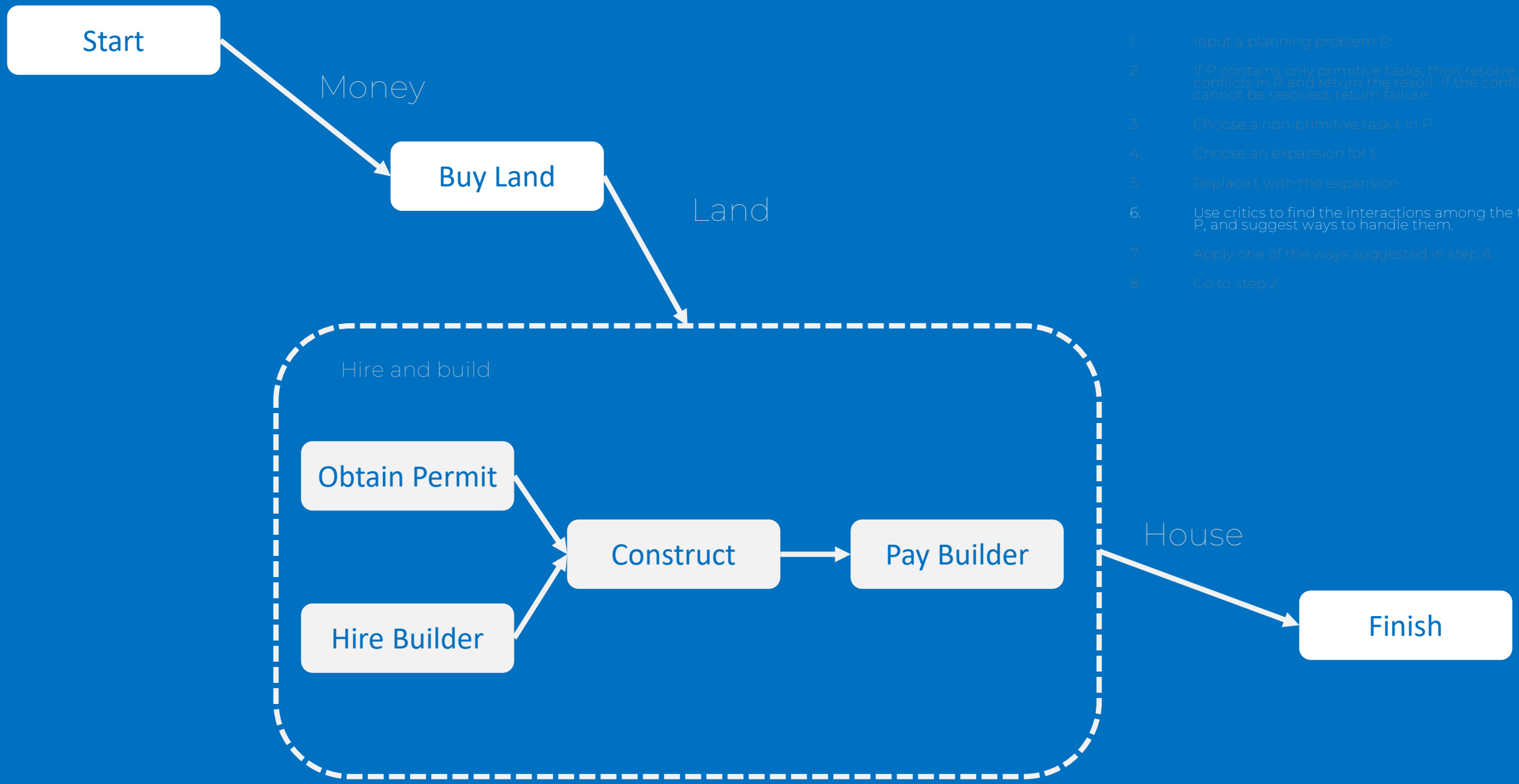


HTN Planning Procedure



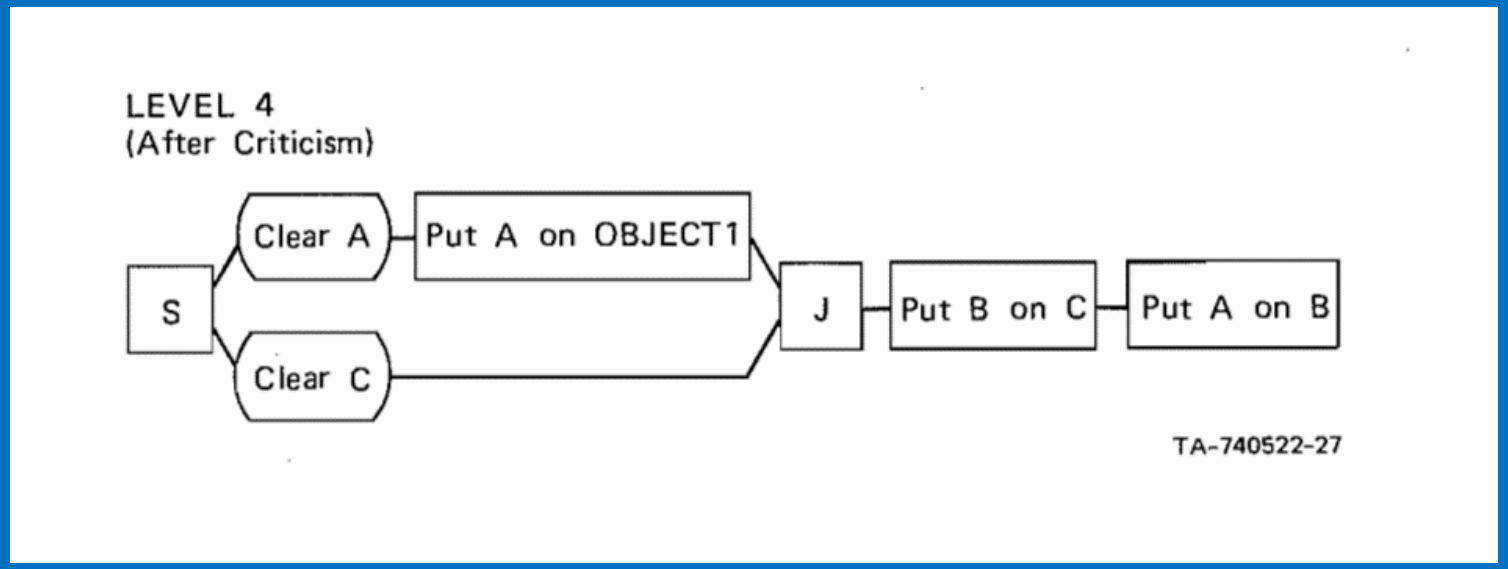
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure



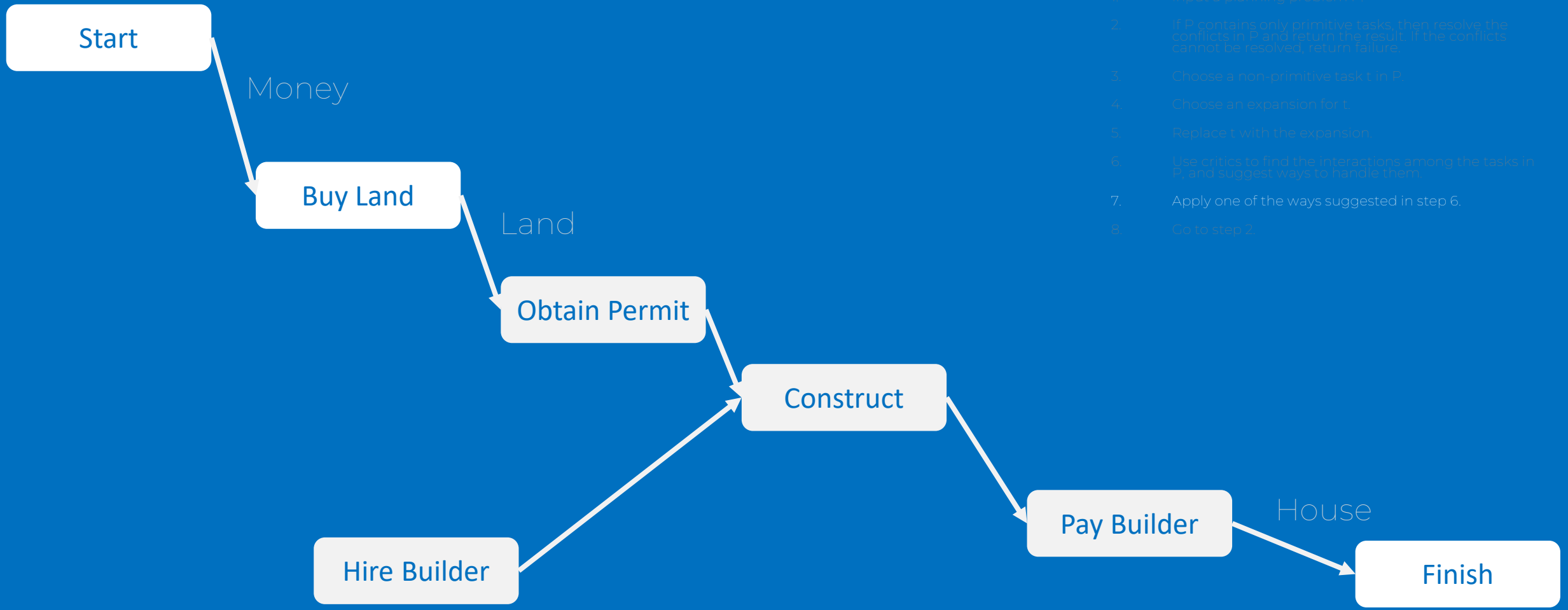
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure



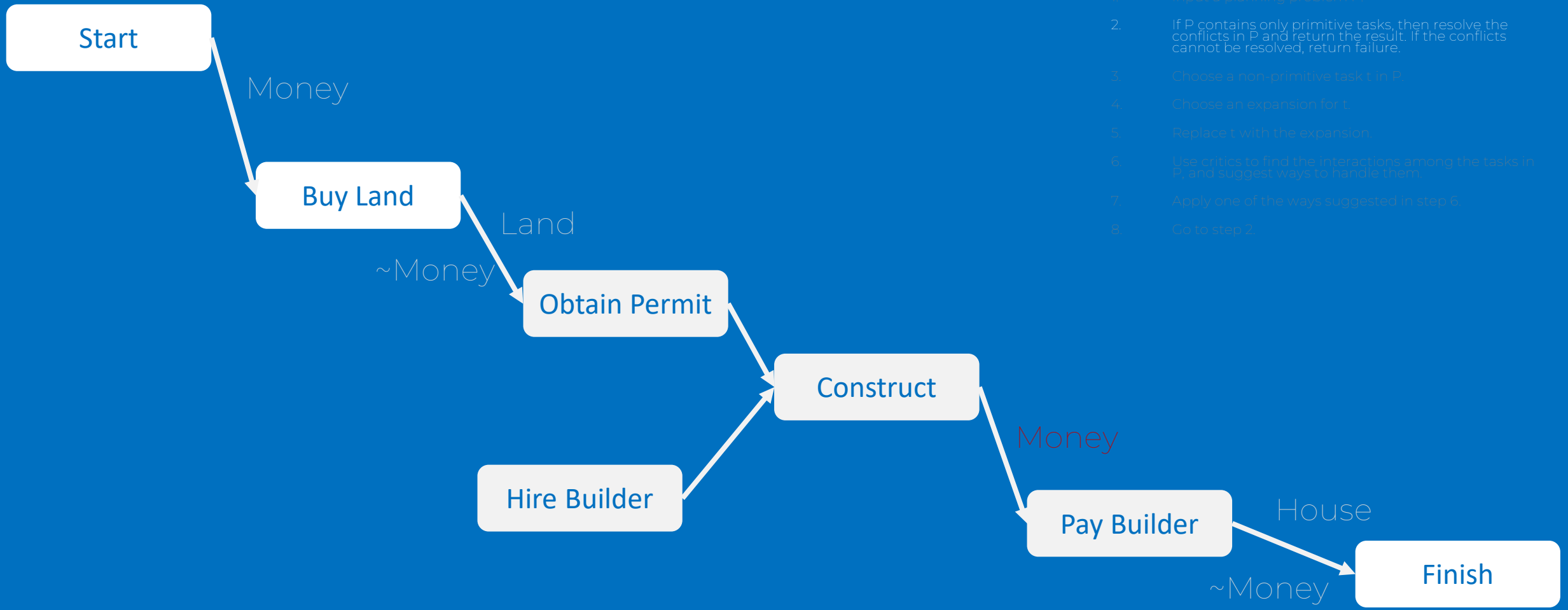
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure



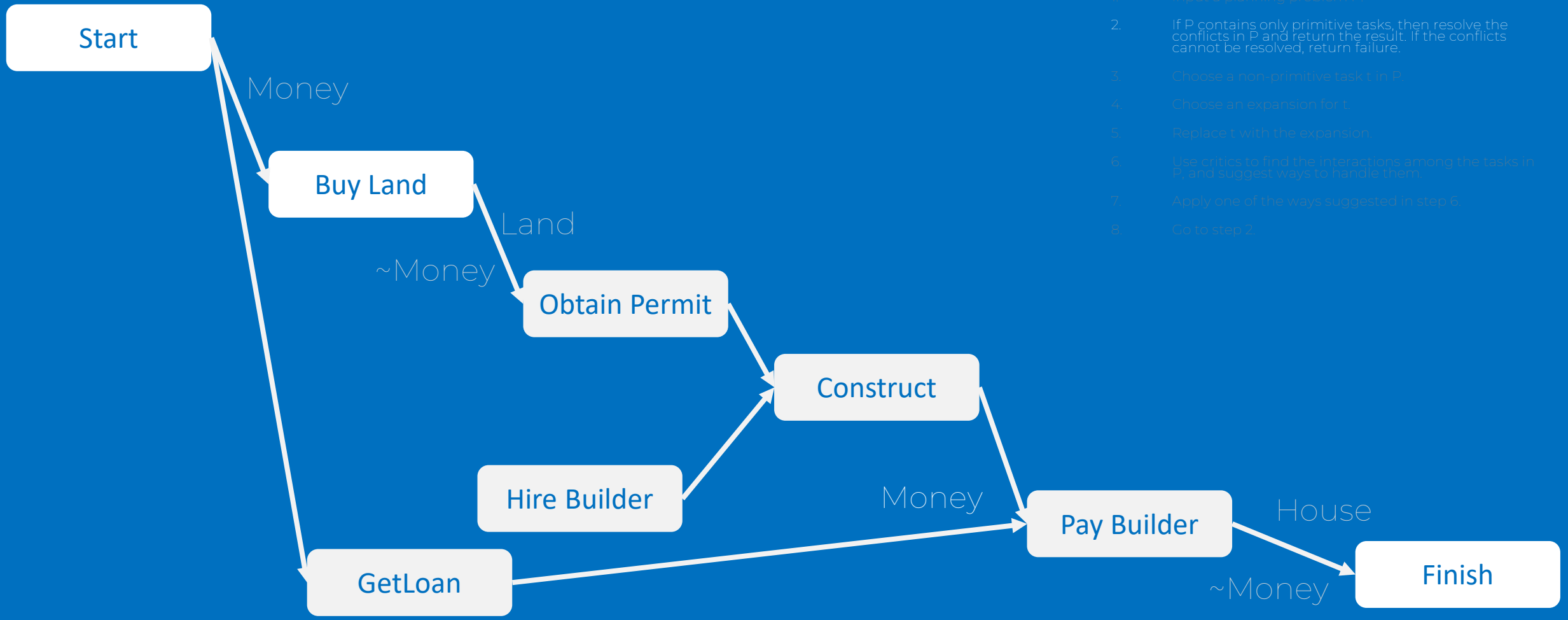
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure



1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

HTN Planning Procedure



1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P and return the result. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

Soundness of UMCP

Whenever UMCP returns a plan, it achieves the input task network at the initial state with respect to all the models that satisfy the methods and the operators

Completeness of UMCP

Whenever UMCP fails to find a plan, there is no plan that achieves the input task network at the initial state with respect to all the models that satisfy the methods and the operators.

Knowledge Representation

Knowledge Representation

Classical Planning

The world state and actions are typically represented using logical expressions, like in the STRIPS, where actions are defined by preconditions and effects.

Knowledge Representation

Classical Planning

The world state and actions are typically represented using logical expressions, like in the STRIPS, where actions are defined by preconditions and effects.

HTN Planning

Represents actions as a hierarchy of tasks. Tasks can be decomposed into subtasks, allowing for a more structured representation of complex actions.

This hierarchical representation enables the decomposition of high-level goals into simpler subgoals.

Problem-Solving Approach

Problem-Solving Approach

Classical Planning

These algorithms focus on finding a sequence of primitive actions to achieve a goal state, often using search algorithms like A* or heuristic search.

Problem-Solving Approach

Classical Planning

These algorithms focus on finding a sequence of primitive actions to achieve a goal state, often using search algorithms like A* or heuristic search.

HTN Planning

HTN planning works by decomposing complex tasks into simpler subtasks, iteratively expanding and resolving conflicts until a conflict-free with primitive tasks is found.

This approach allows for more structured and goal-directed problem-solving.


XBOX 360
ROCKSTAR GAMES PRESENTS
MAX PAYNE 3
MATURE 17+
M
CONTENT RATED BY ESRB


PS3
PlayStation Network
DARK SOULS
MATURE 17+
JEUNES ADULTES
M
CONTENT RATED BY ESRB
BANDAI


TRANSFORMERS
FALL OF CYBERTRON
TEEN
T
CONTENT RATED BY ESRB
Online Interactions Not Rated by the ESRB
ACTIVISION