Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing

Ron Alford
MITRE; McClean, VA

Vikas Shivashankar
Knexus Research Corporation; National Harbor, MD

Mark Roberts
NRC-NRL Postdoctoral Fellow; Washington, DC

Jeremy Frank
NASA Ames Research Center; Moffett Field, CA

David W. Aha
Naval Research Laboratory; Washington, DC

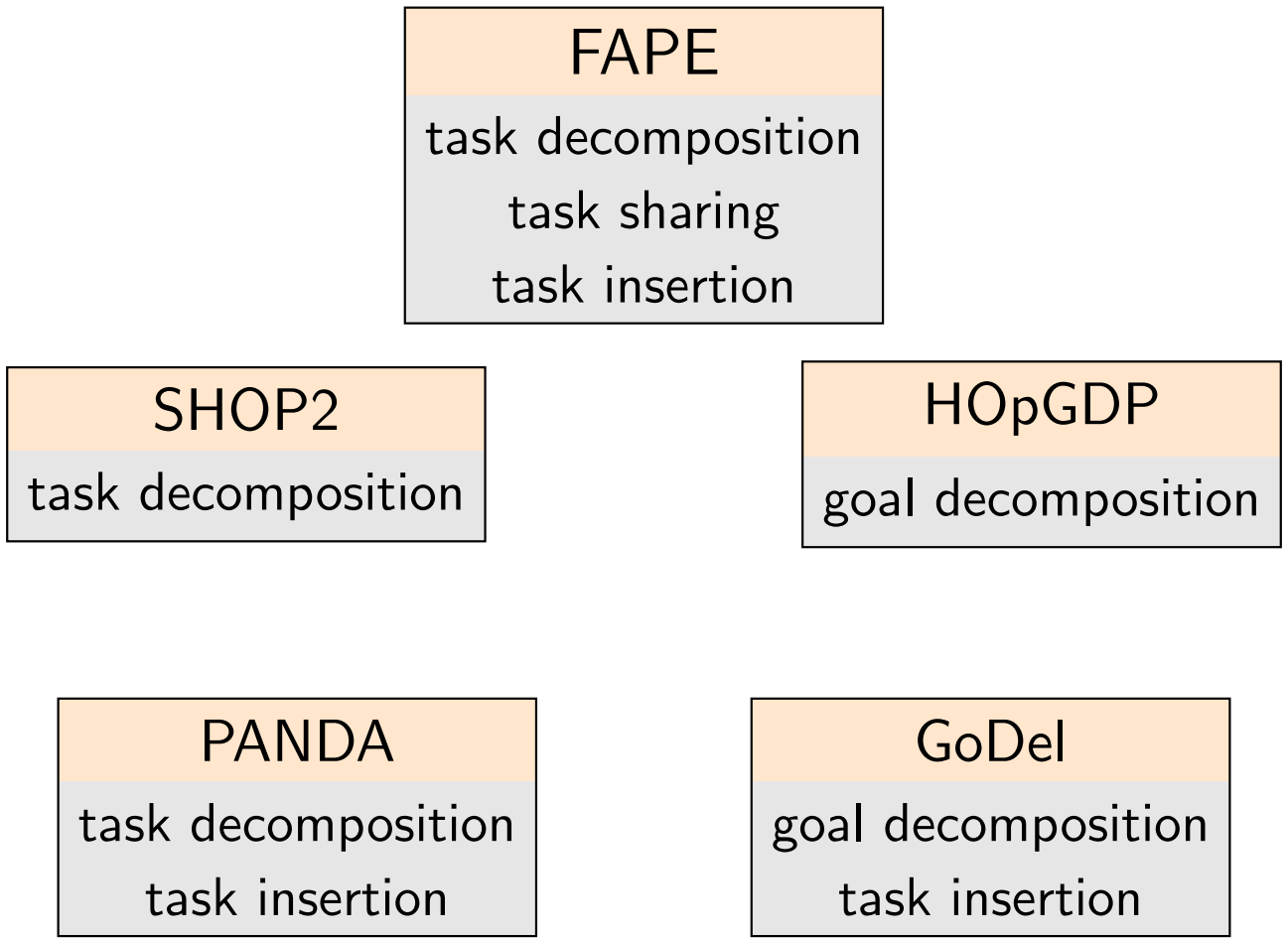
Hierarchical Planning

Hierarchical planning is the problem of decomposing an initial task or goal into a sequence of executable steps.

- Encoding control knowledge for planning
  - Agent planning in robotics and simulation: ARTUE (2010), Johnny (2012)
  - Planning and mission generation for games: KILLZONE 2, Elder Scrolls, Armed Assault
- Representing and planning with processes and hierarchies
  - Composing web services (Sirin, 2004; Tang 2013; others)
  - Program configuration (Soltani, 2012)

Source: Fraunhofer

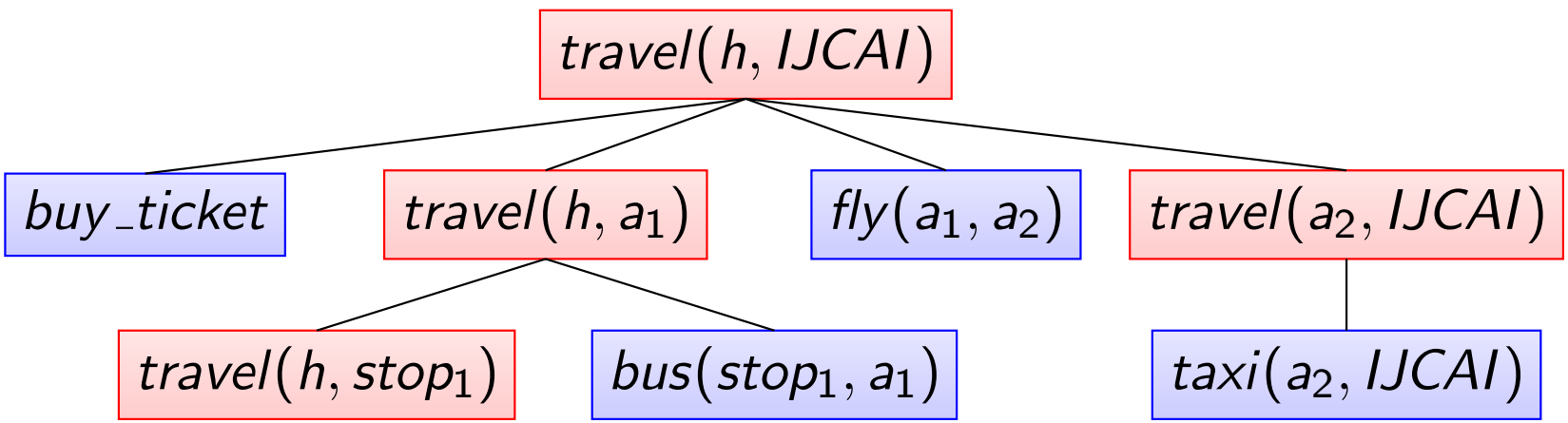
A Proliferation of Hierarchical Planners (and Formalisms)



HTN Planning (Overview)

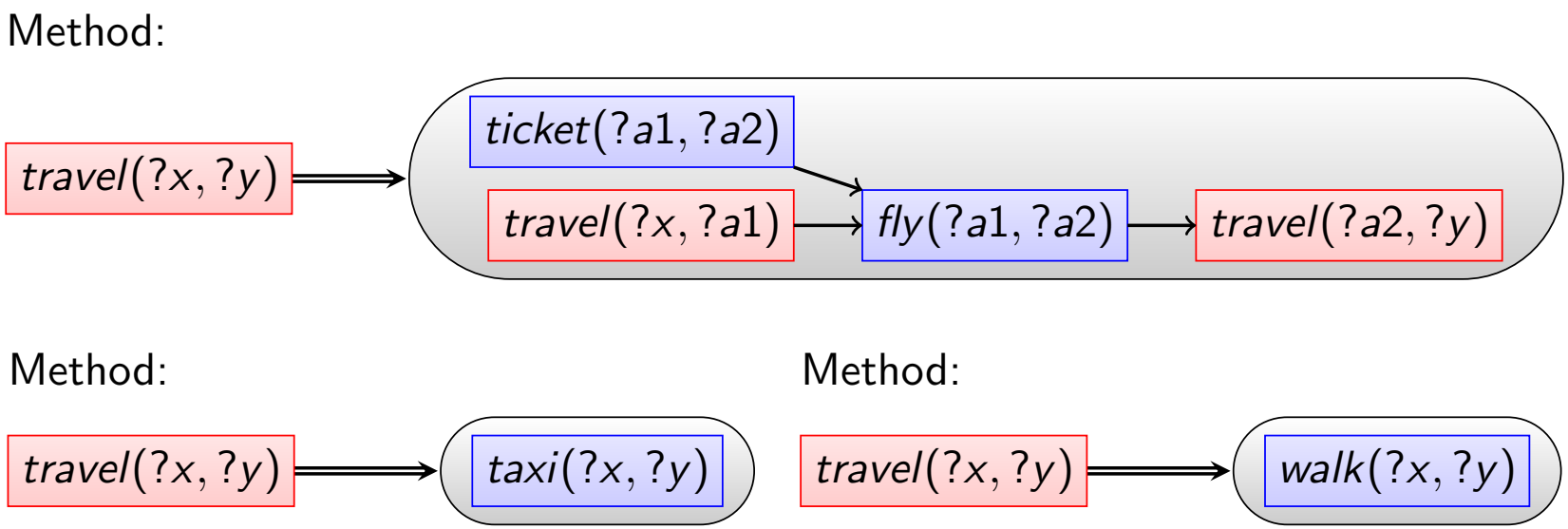
The purpose of HTN planning is to complete a task. Tasks are either:

- Primitive, which corresponds to some concrete action we know how to perform, e.g., $walk(room, hall)$, or $drink(coffee)$
- Non-primitive, which is an abstract task. E.g. $travel(home, IJCAI)$
- Must recursively decompose non-primitive tasks until we get primitive tasks we know how to execute directly
- We are given a set of methods, which are recipes on how to accomplish abstract tasks. E.g., to travel from *home* to *IJCAI*, we might decompose as follows:

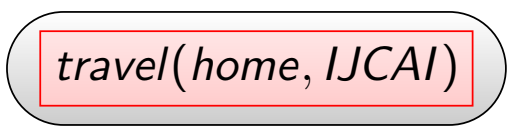


Methods and Decomposition

- A method (t, tn) is a non-primitive task t paired with a network tn



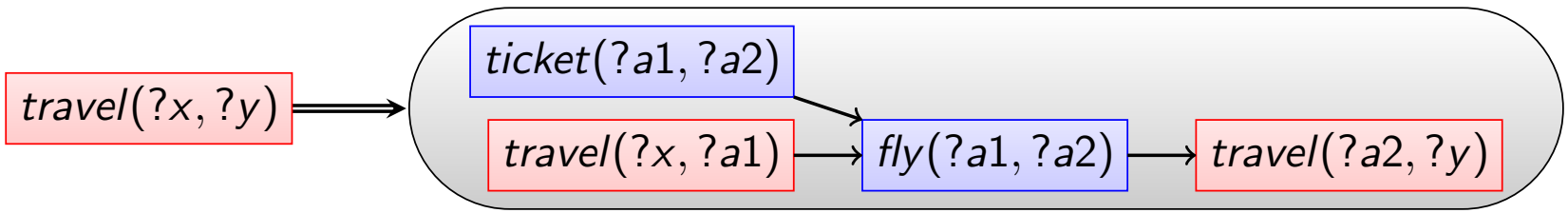
- We decompose a task network by replacing a node in the network with a corresponding method's network.



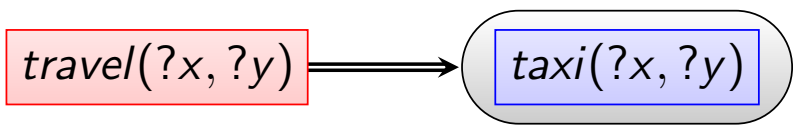
Methods and Decomposition

- A method (t, tn) is a non-primitive task t paired with a network tn

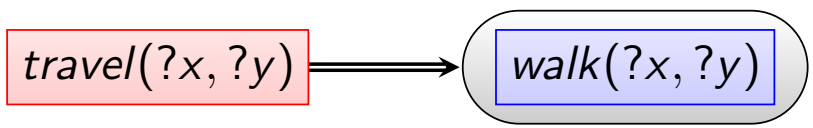
Method:



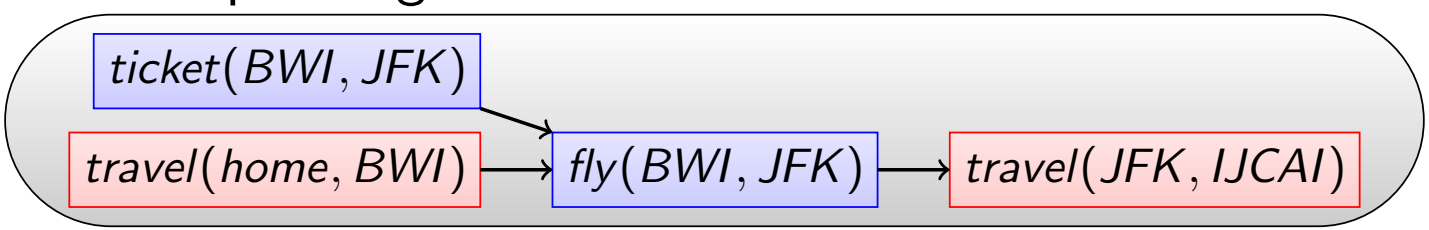
Method:



Method:



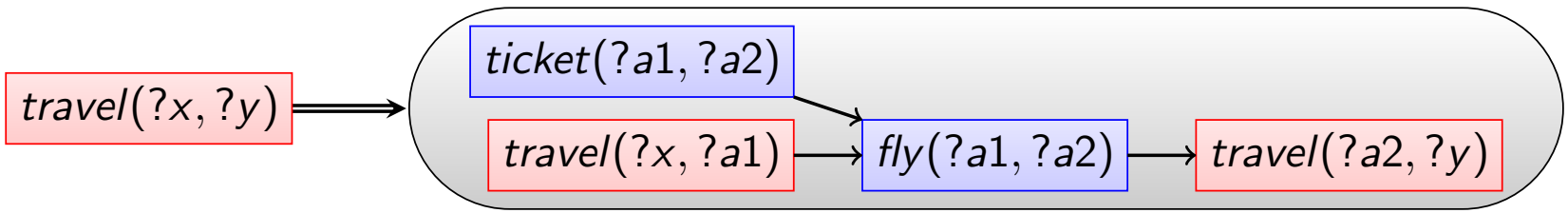
- We decompose a task network by replacing a node in the network with a corresponding method's network.



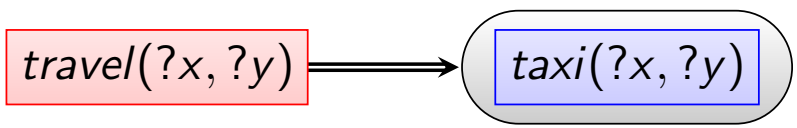
Methods and Decomposition

- A method (t, tn) is a non-primitive task t paired with a network tn

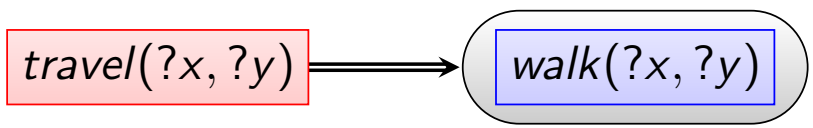
Method:



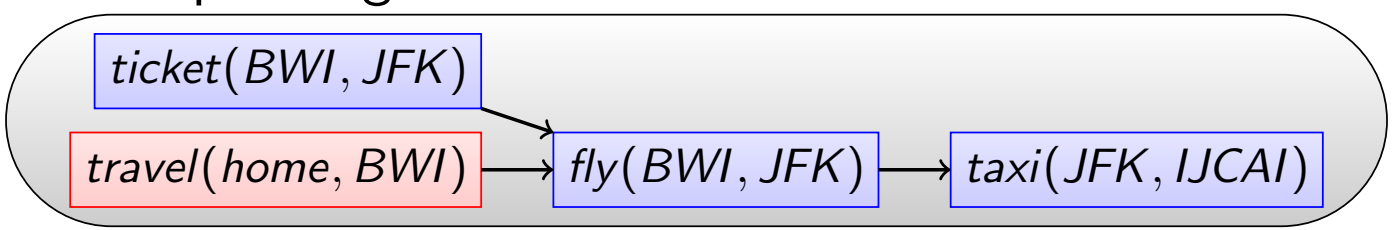
Method:



Method:



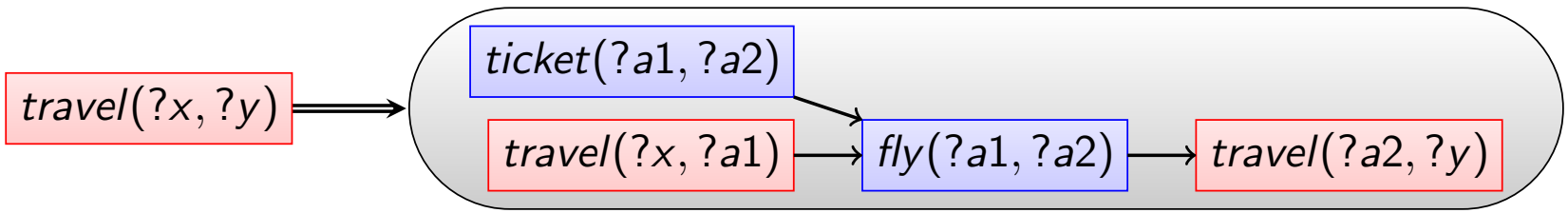
- We decompose a task network by replacing a node in the network with a corresponding method's network.



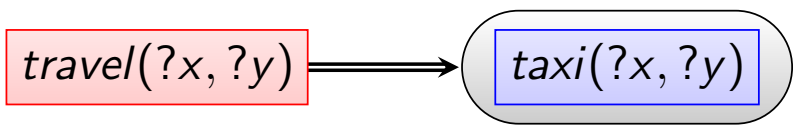
Methods and Decomposition

- A method (t, tn) is a non-primitive task t paired with a network tn

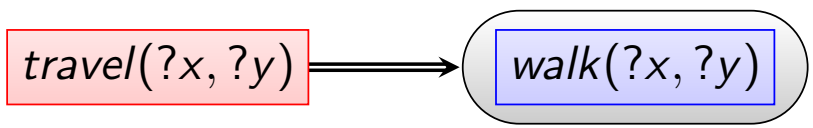
Method:



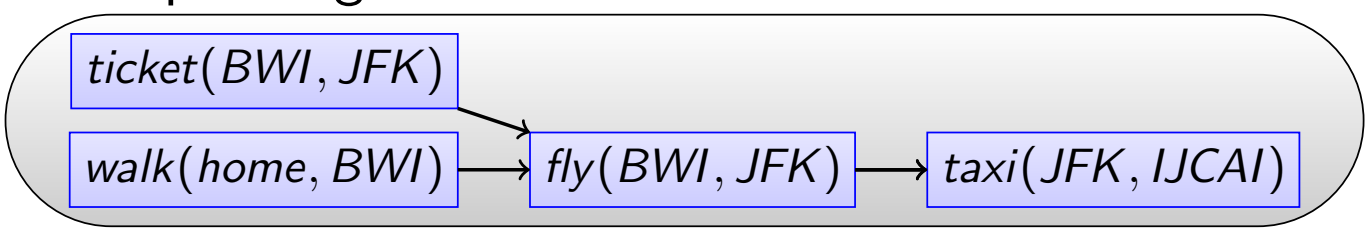
Method:



Method:



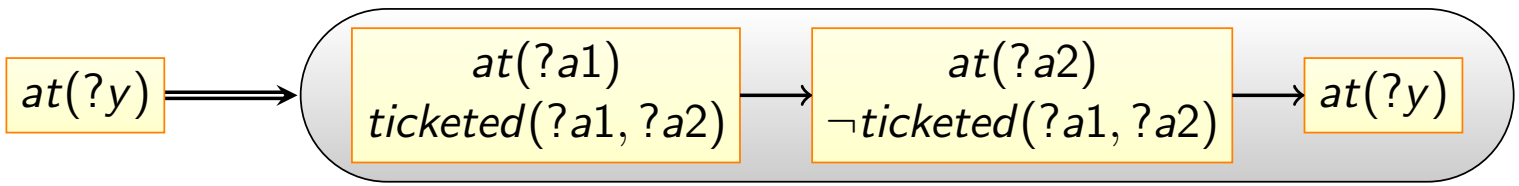
- We decompose a task network by replacing a node in the network with a corresponding method's network.



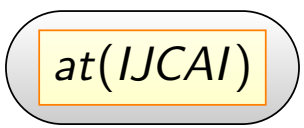
Goal Decomposition

- A goal method (g, tn) is goal paired with a network tn

Method:



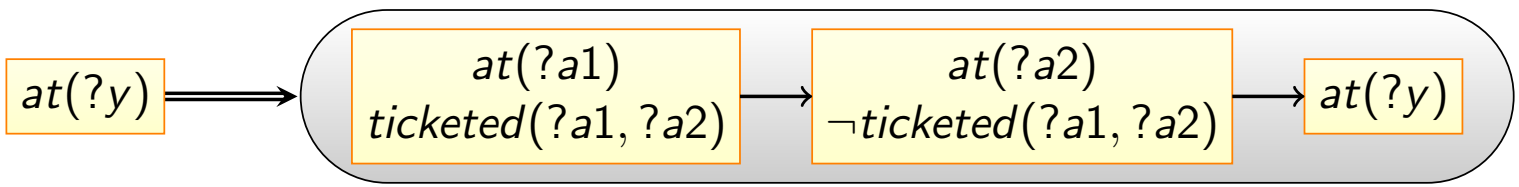
- We decompose a goal by <u>prepending</u> a node in the network with a relevant method's network.



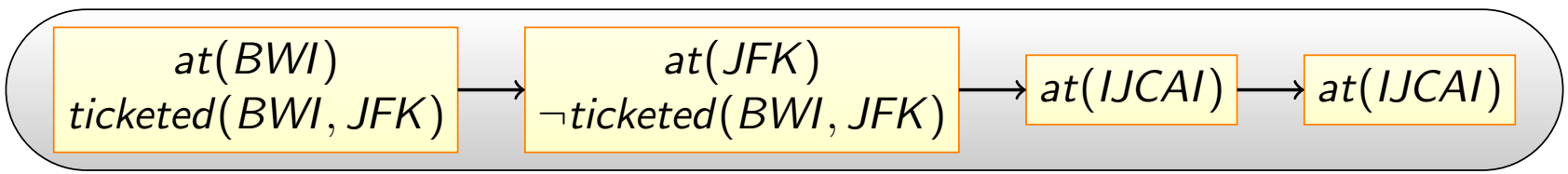
Goal Decomposition

- A goal method (g, tn) is goal paired with a network tn

Method:

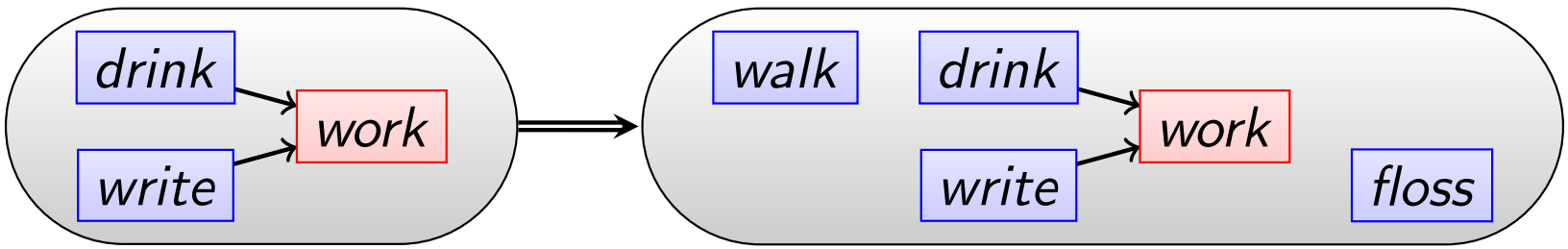


- We decompose a goal by <u>prepending</u> a node in the network with a relevant method's network.



Task Insertion

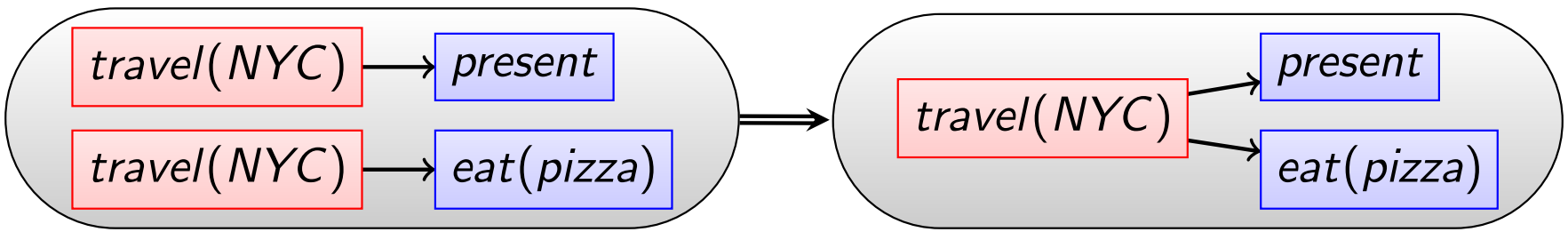
- An alternate set of semantics, HTN Planning with Task Insertion (TIHTN Planning) allows the insertion of tasks without a method.



- Developed by Kambhampati (1998) and Biundo (2002)
- Used as a model for replanning and search with partial HTN knowledge

Task Sharing

- Task sharing allows unconstrained identical tasks to be merged

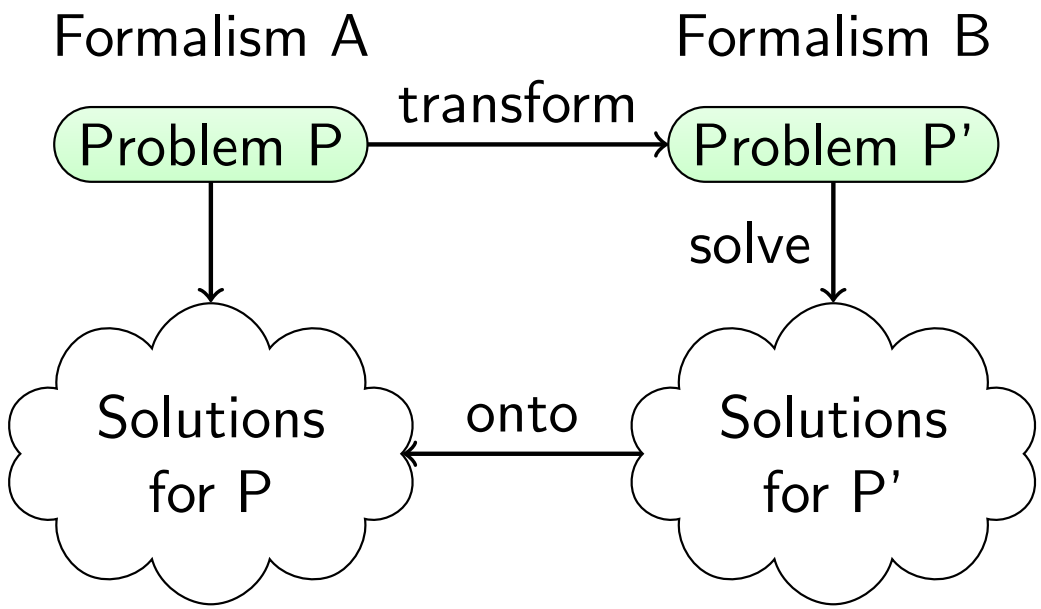


- The ANML language implicitly defines task sharing
- FAPE and other ANML implementations support sharing.

Relating Hierarchical Planning Formalisms

Reductions

- Preserves solvabilty
- Planners for B can technically be used to solve for problems in A.



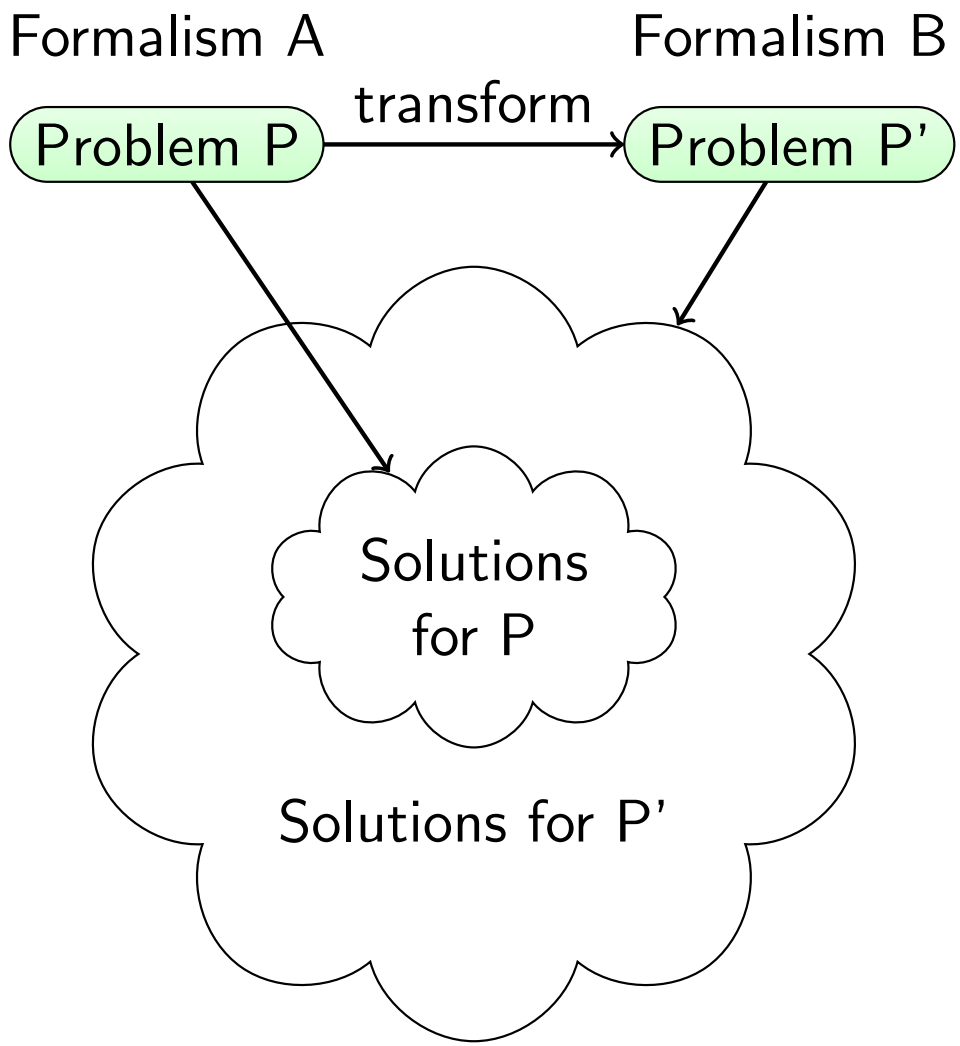
Relating Hierarchical Planning Formalisms

Reductions

- Preserves solvabilty
- Planners for B can technically be used to solve for problems in A.

Relaxation

- P solvable implies P' solvable (not vice versa)
- B's heuristics can be used for A



Relating Hierarchical Planning Formalisms

Reductions

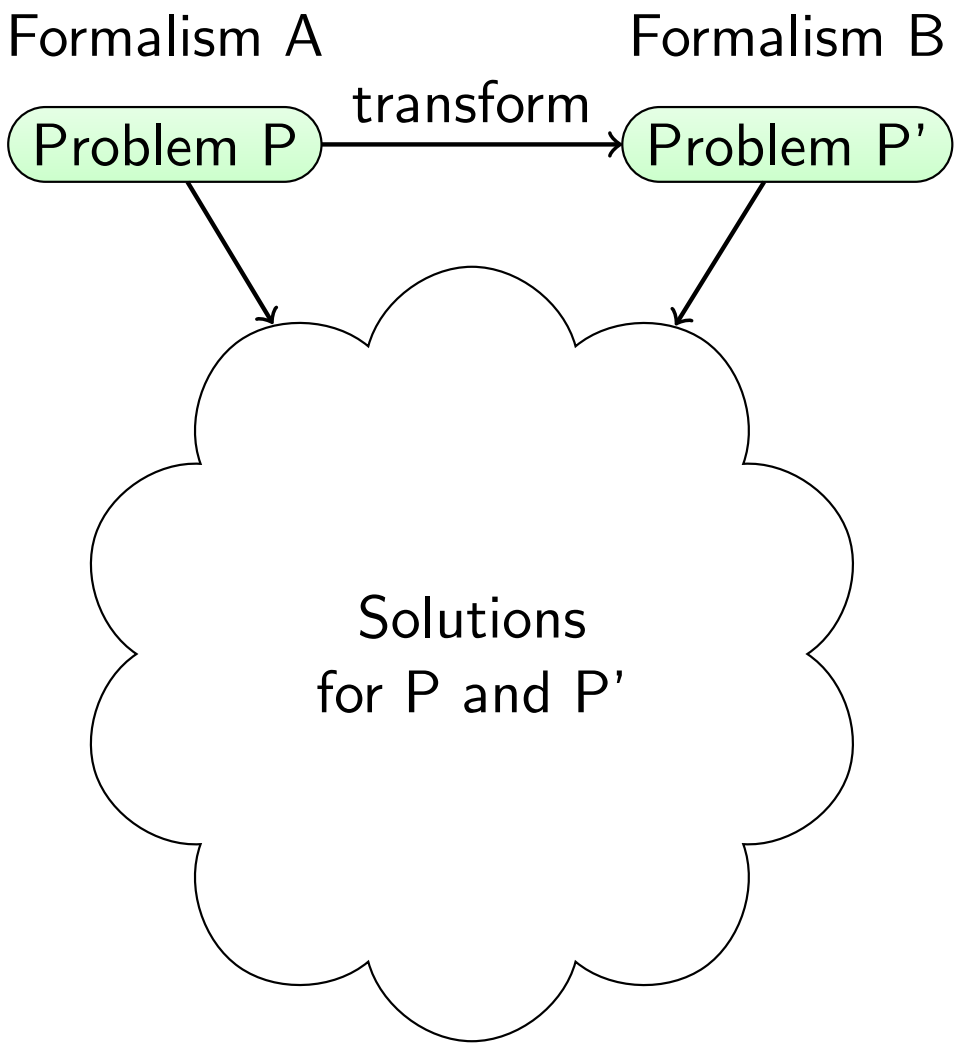
- Preserves solvabilty
- Planners for B can technically be used to solve for problems in A.

Relaxation

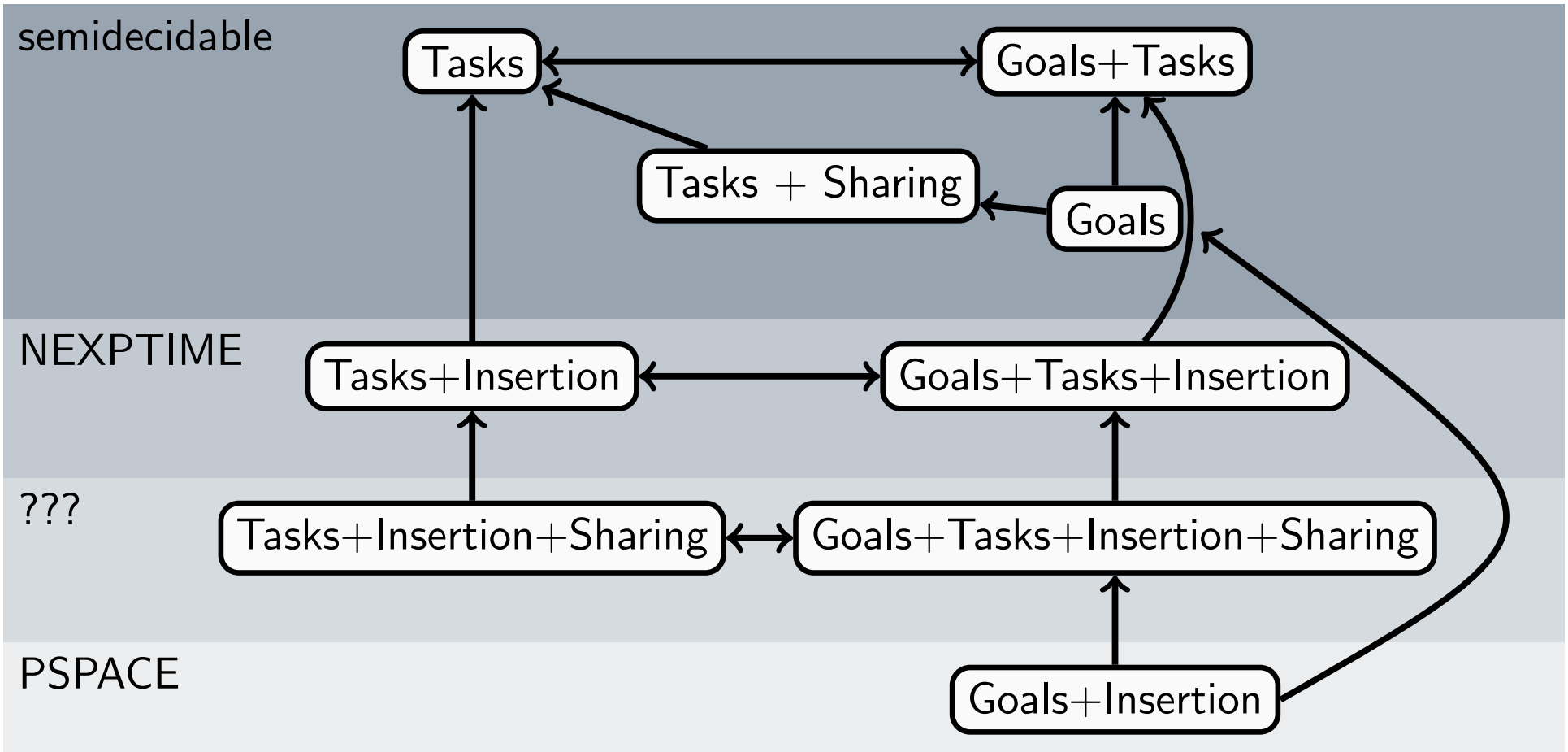
- P solvable implies P' solvable (not vice versa)
- B's heuristics can be used for A

Plan-Preserving Transformations

- Same solutions for P and P'
- Preserves optimality



Plan-Preserving Transformations



Going against the grain

Using goal-decomposition planners
for task planning:

Going against the grain

Task-decomposition planner

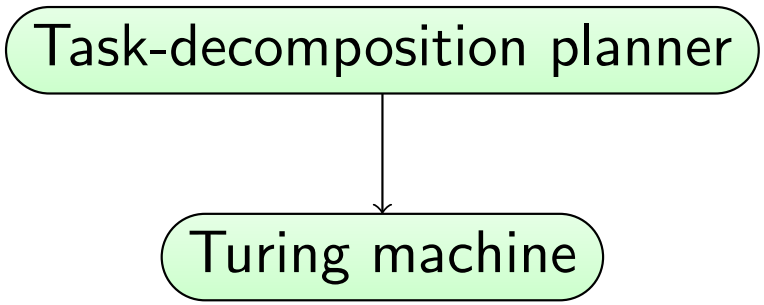
Using goal-decomposition planners for task planning:

- First, write a task planner

Going against the grain

Using goal-decomposition planners for task planning:

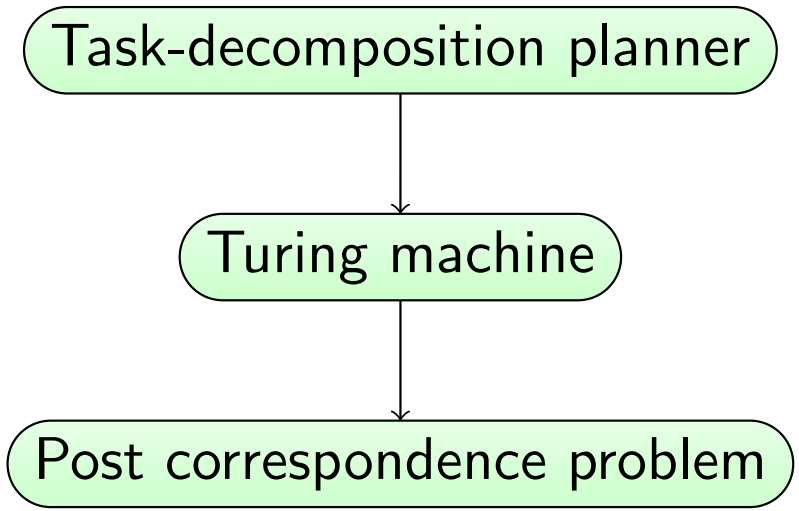
- First, write a task planner
- Compile planner into Turing machine (TM)



Going against the grain

Using goal-decomposition planners for task planning:

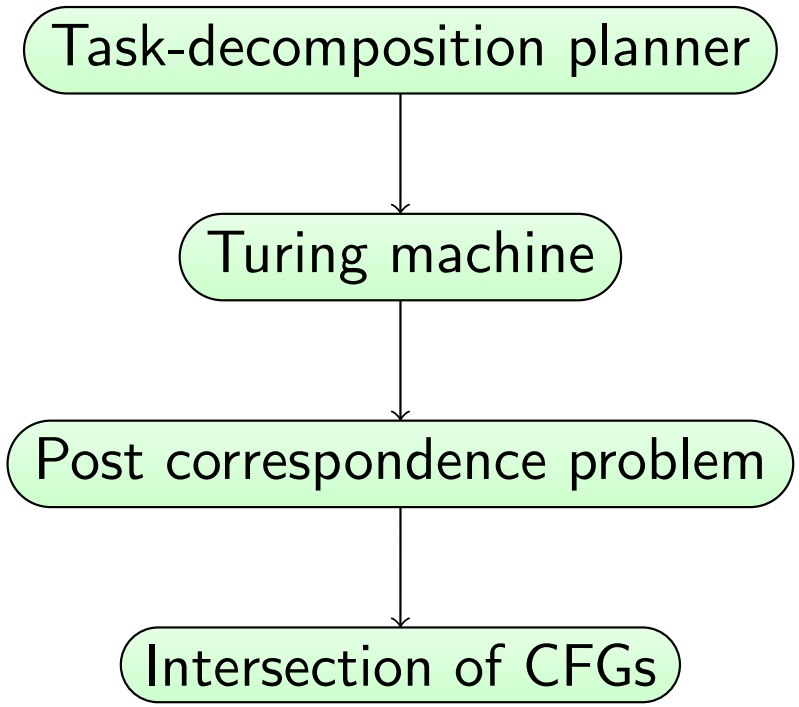
- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)



Going against the grain

Using goal-decomposition planners for task planning:

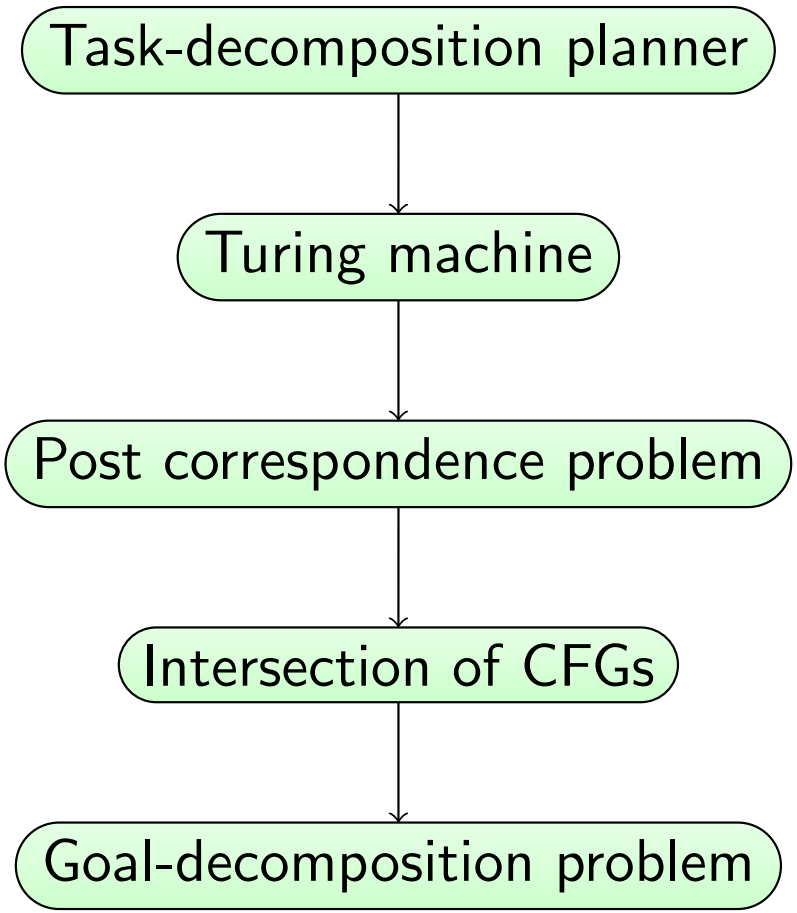
- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)
- Translate PCP into intersection of CFGs



Going against the grain

Using goal-decomposition planners for task planning:

- First, write a task planner
- Compile planner into Turing machine (TM)
- Translate TM into a Post correspondence problem (PCP)
- Translate PCP into intersection of CFGs
- Encode CFGs as goal methods



Conclusions

Contributions:

- Combined formalism for goal and task decomposition
- Formal semantics for task sharing
- Formal relationships between sets of semantics

Conclusion: The semantics of HTN planning are both simple and general. Why use anything else?

- Classical heuristics can be easily adapted for goal-decomposition
- Task insertion allows the use of more efficient (terminating) algorithms
- Task sharing may allow even more efficient algorithms when combined with task insertion (future work)

Future work: Practice still has to catch up to theory (more admissible heuristics, optimal and suboptimal planning implementations, etc)