On Trustworthy, Explainable, and Verifiable High-Level Autonomy via Hierarchical Planning

Roman Barták

Charles University, Faculty of Mathematics and Physics
Czech Republic

Context

Hierarchical Planning

- finding a plan (sequence of actions) by decomposing tasks into subtasks

Plan Validation

- validate if a given sequence sequence is a correct hierarchical plan

Plan Recognition

- complete an observed plan prefix to find out which task is being executed

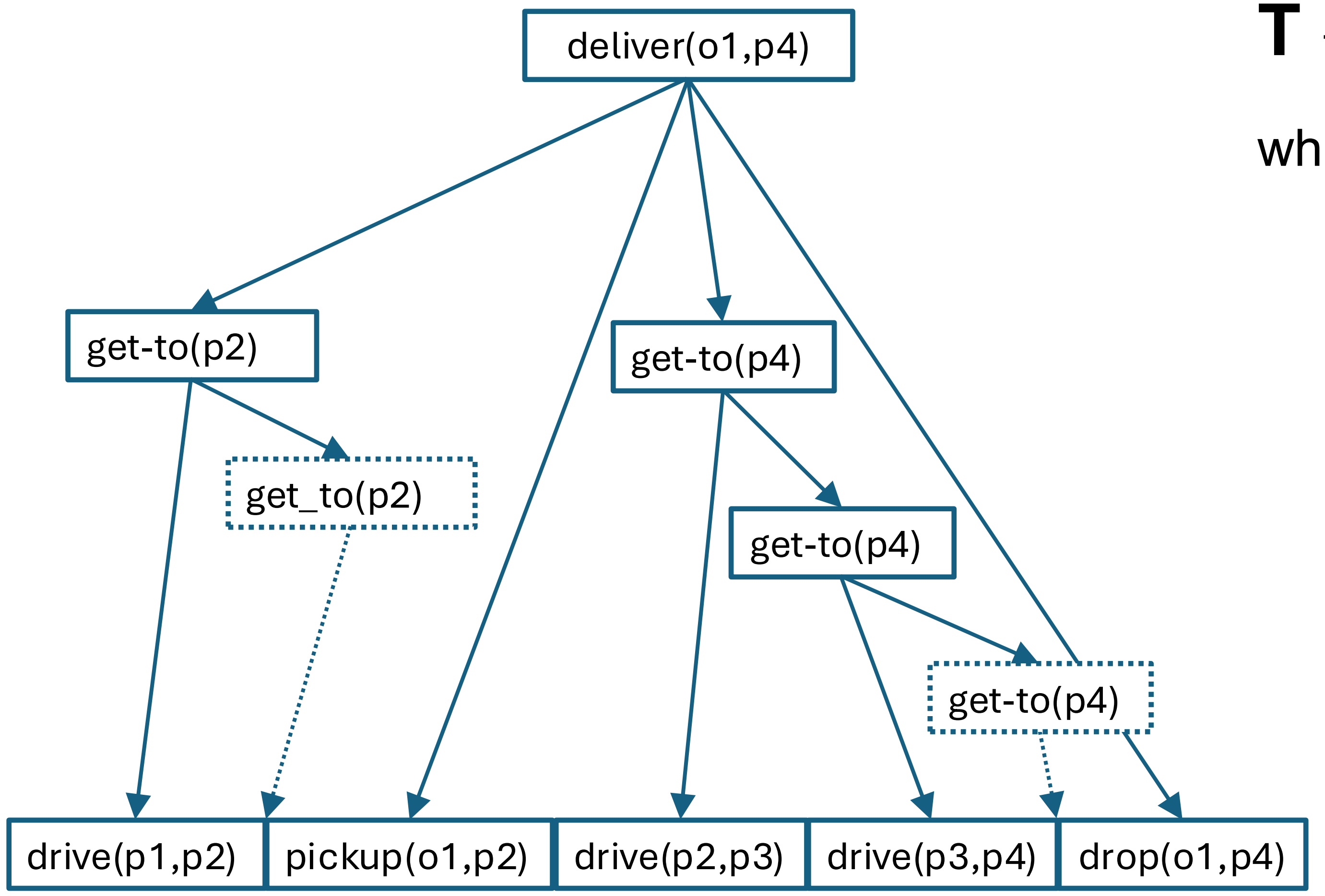
Plan Correction

- make any action sequence a correct hierarchical plan by a minimal number of modifications

Model Correction

- modify a task decomposition model to “cover” a given plan

Hierarchical Planning

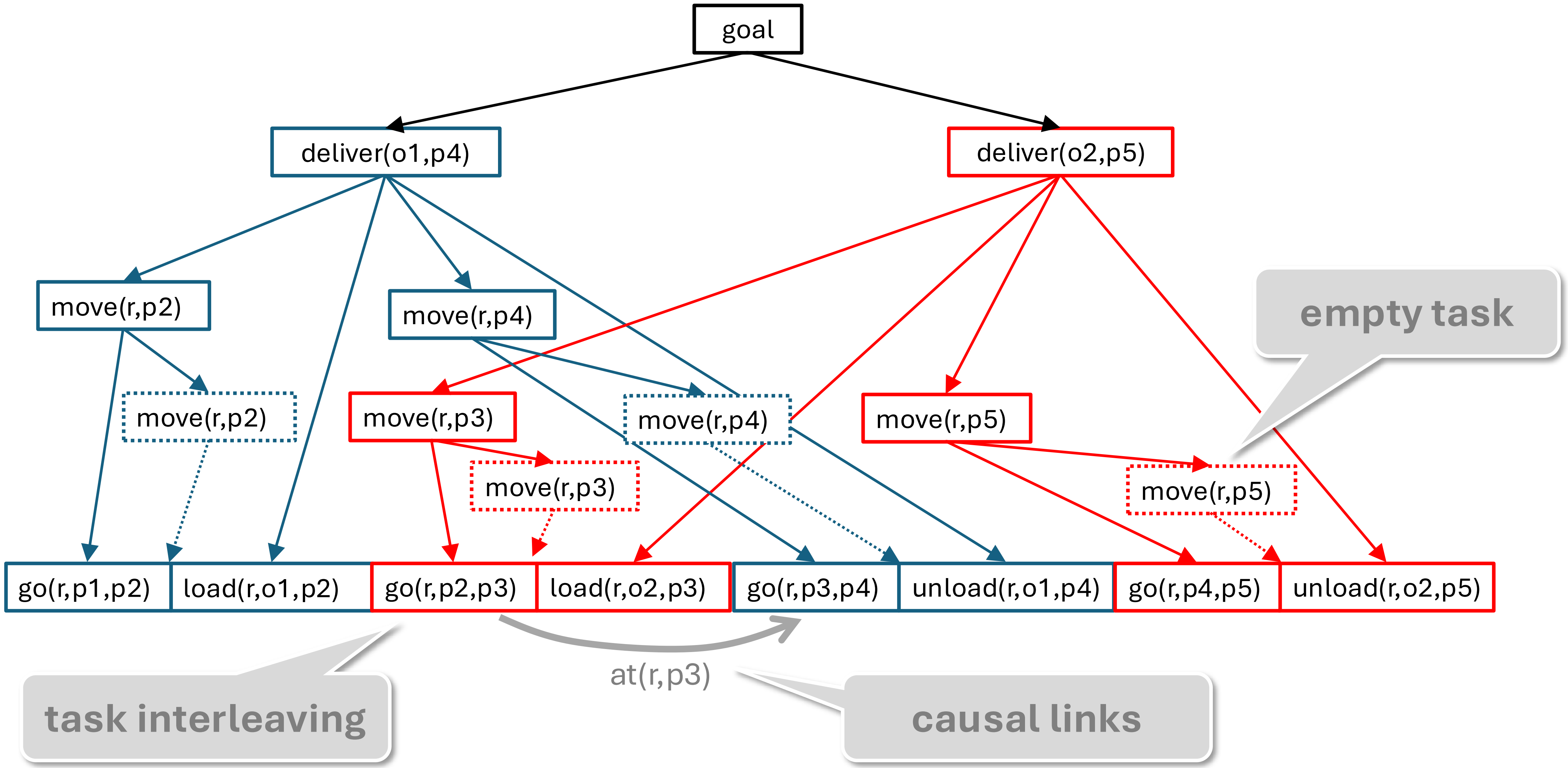


$$T \rightarrow T_1 \ldots T_k \ [C]$$

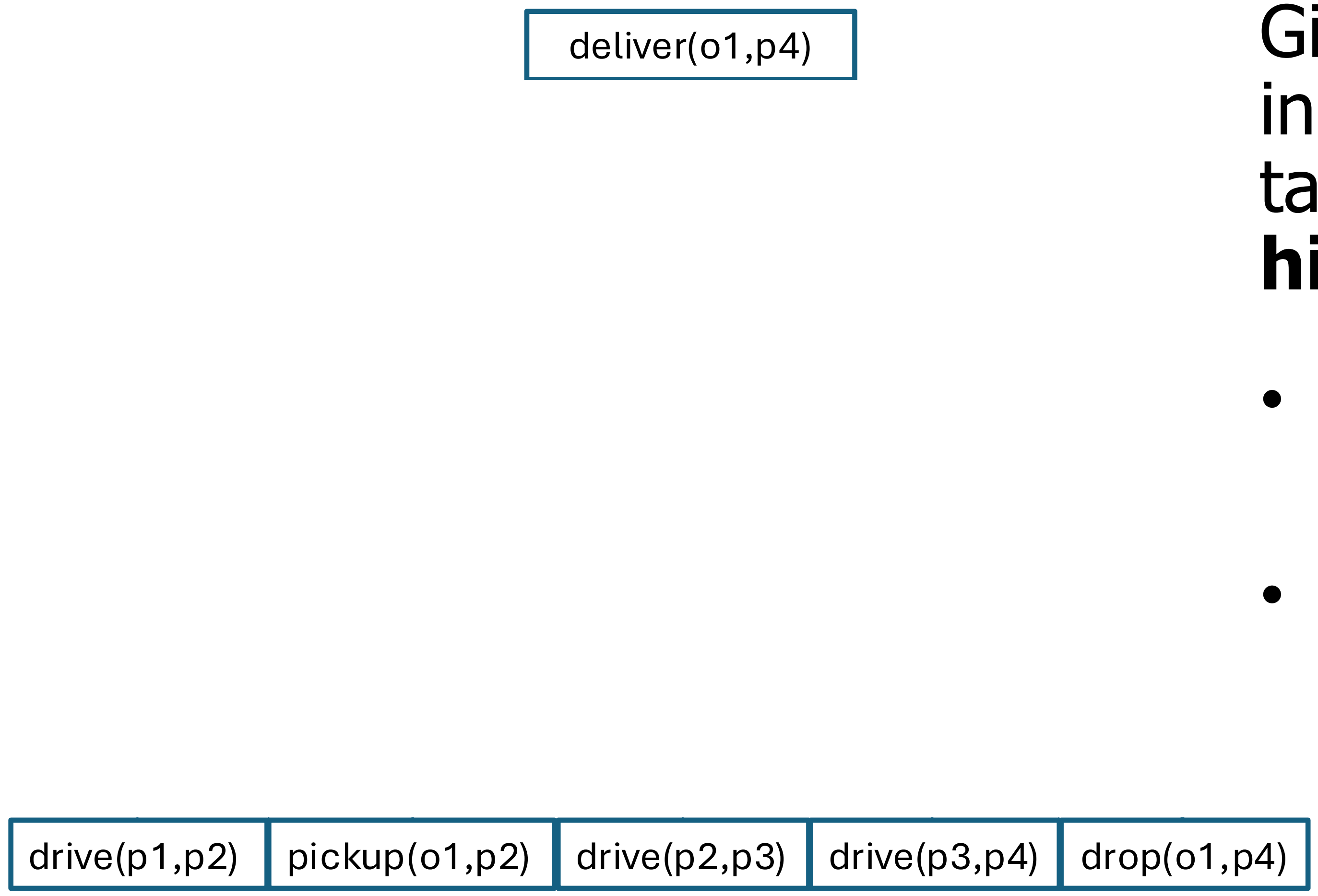
where **C** are decomposition constraints:

- $T_i < T_j$: ordering of tasks
- **before(U,p)**: a precondition constraint (**p** is true right before the first task from **U**)
- **after(U,p)**: a postcondition constraint (**p** is true right after the last task from **U**), not an effect!!
- **between(U,p,V)**: prevailing constraints (**p** is true between the last task of **U** and the first task of **V**)

Decomposition tree



Hierarchical Plan Explanation

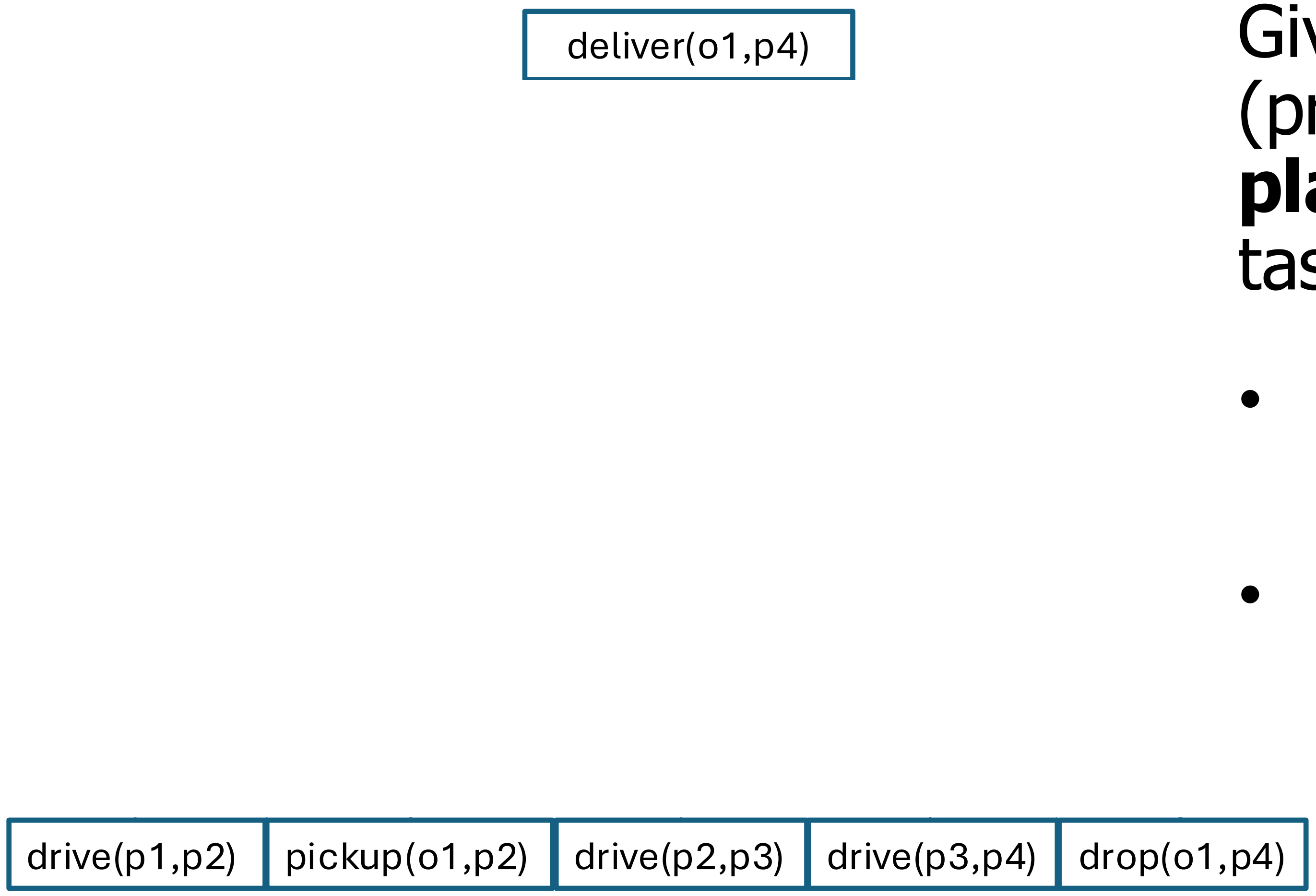


Given a sequence of actions and initial state (and perhaps a goal task), **validate that it is a valid hierarchical plan**:

- the plan is executable in the initial state;
- the plan is obtained by decomposition of a (given) task from the domain model.

Geib, Maraist, Goldman; ICAPS 2008

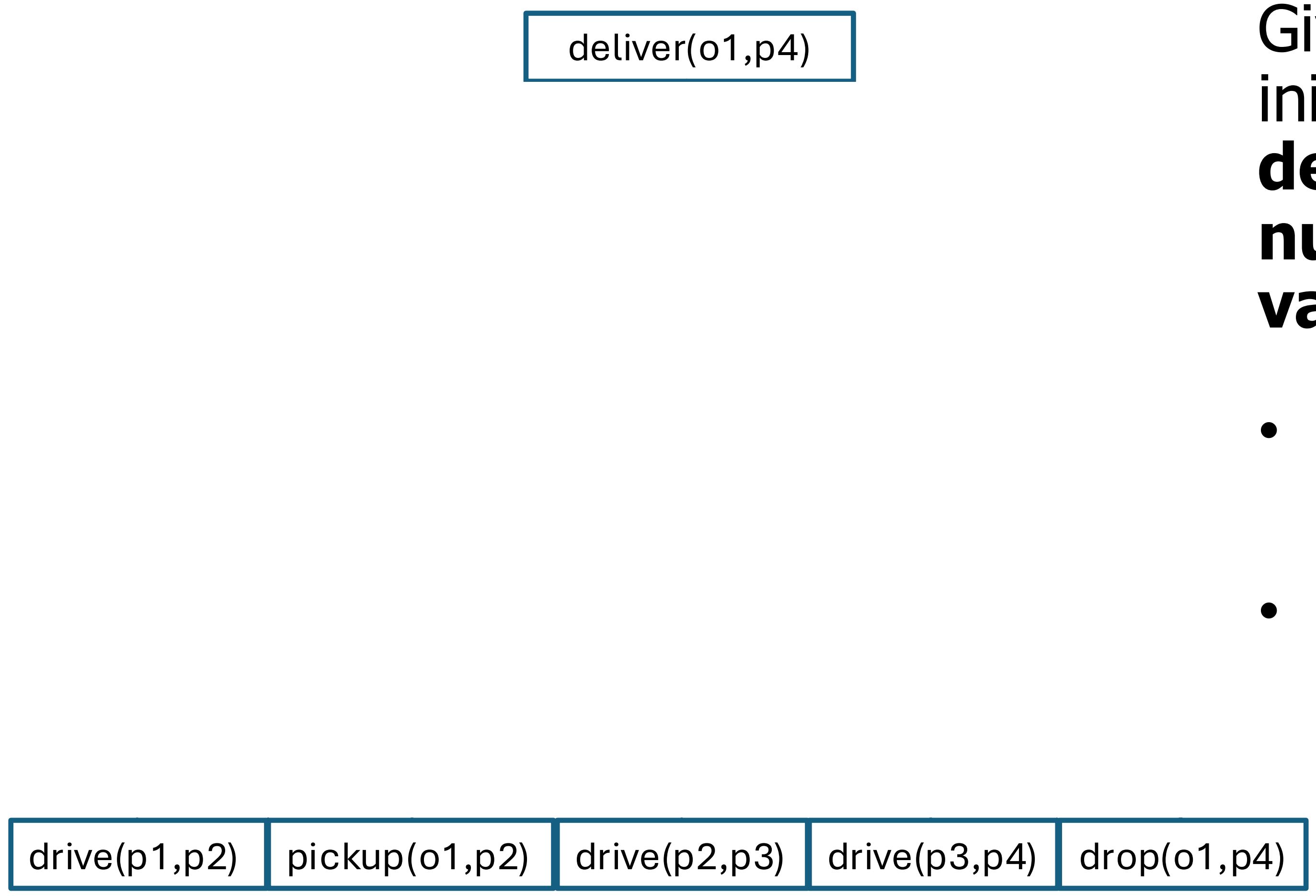
Hierarchical Plan Recognition



Given a sequence of actions (prefix) and initial state, **find a plan suffix** (and possibly a goal task) such that:

- the complete plan is executable in the initial state;
- the complete plan is obtained by decomposition of a (given) task from the domain model.

Hierarchical Plan Correction



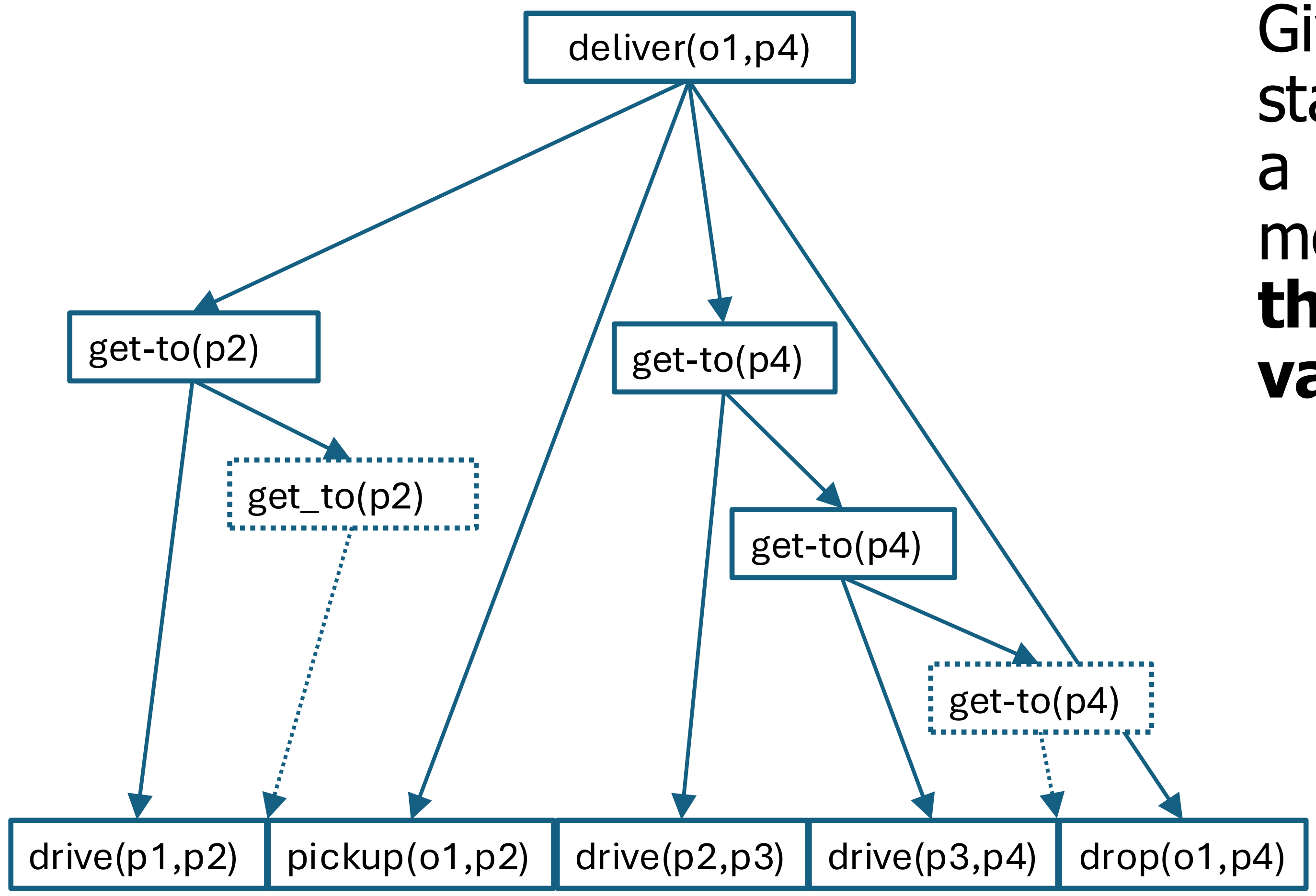
Given a sequence of actions and initial state (and perhaps a goal task), **delete or insert a minimum number of actions to obtain a valid hierarchical plan**:

- the plan is executable in the initial state;
- the plan is obtained by decomposition of a (given) task from the domain model.

Applications of Plan Correction

- **HTN plan validation** (when neither action deletion nor action insertion is enabled);
- **correcting an HTN plan** for a known goal (when the possible top-level task is given);
- **HTN plan recognition with full observability** (when action deletion is disabled, and action insertion is enabled only after the observed plan prefix);
- **HTN plan recognition with partial observability** (when action deletion is disabled, and action insertion is enabled anywhere in the observed sequence);
- **HTN plan recognition with full or partial observability and with noise** (when action deletion is enabled);
- **HTN planning** (when action insertion is enabled, and the input plan is empty).
- **HTN plan repair** (when action insertion is enabled after the modified state modelled as a dummy action)

Model Correction



Given a sequence of actions, initial state (and perhaps a goal task), and a domain model (decomposition methods), **update the model such that the action sequence is a valid hierarchical plan**.

deliver(O,P) → *pickup*(O,P'), *drop*(O,P)

deliver(O,P) → get-to(P'), *pickup*(O,P'),
get_to(P), *drop*(O,P)

get_to(P) → [before(get_to, at(P))]
get_to(P) → *drive*(P',P''), get_to(P)

Complete vision

Autonomous, adaptable agent with explainable and verifiable behavior

- start with some **initial** (possibly empty) hierarchical and action **models**
- **plan** for a given goal task, for example using TIHTN (i.e. using also classical planning)
- if the plan does not work, then re-plan, **repair the plan**
- if the plan is correct, **update the model** accordingly (learn new tasks and actions)

In multi-agent environment:

- use the model to predict actions of other agents
- update the model based on observations of other agents (learning by observation)