How Much Does a Household Robot Need to Know in Order to Tidy up?

AAAI Workshop on *Intelligent Robotic Systems*

*Bernhard Nebel, Christian Dornhege, Andreas Hertle*

Department of Computer Science
Foundations of Artificial Intelligence

Albert-Ludwigs-Universität Freiburg
Outline

- Motivation
- Expanding universes
- Limited uncertainty
- Continual replanning vs. conditional planning
  - Soundness, completeness, complexity
- Conclusions & Outlook
Motivation: Tidy up

- Use plan-based agents
  - to anticipate the future
  - to compose behaviors / motor programs into complex action sets: **plans**
  - in order to achieve goals
- What should they *know* in order to generate and execute a plan?
- What kind of *planning technique* should they use?

**Classical planning**
- is well researched
- there are fast planning systems (TFD/M)
Historic Perspective

Planning problem classes

- **Domain:** closed or open
- **Effects:** deterministic, non-deterministic, probabilistic
- **Observability of the environment:** complete, partial, not observable
- **Horizon & Objective:** …

- **Classical Planning:** closed domain, deterministic actions, complete observability (in the beginning), …
Household situations

- It is *not known* how many objects exist in the household (> 10000)
  - but the set of types of objects is fixed
- It is *not known* what states the objects are in
  - but the state can be observed when the robot is close to the object
- The outcomes of actions can vary (non-determinism)

▸ Classical planning is not adequate
What do we lose?

- There is no planning system for open domain, non-deterministic, partially observable planning.

- Even if we do we away with open domains, CltAltAlt, POND or MBP could be used.
  - However, they are slow compared with e.g. TFD.

- If we simplify the problem and use a classical planner:
  - What kind of reasoning do we lose?
  - Can we guarantee completeness / soundness under some conditions?
  - How hard is it to check such conditions?
A note about notation

- We will not use any particular planning formalism or language in this talk.

- Think of PDDL extended by non-determinism and branching: \textit{NPPDL}.

- All pre-conditions (and goal conditions) are implicitly in the scope of a modal knowledge operator (most of the time)
  - single-agent logic!
In principle, we want a planning language with an open domain = countable number of objects of each type

Current planners use propositionalization (grounding to propositional logic) in order to be efficient

Planning with open domains is undecidable [Erol et al 95]
- Turing machine with an unbounded number of tape cells could be simulated
Expanding universes

- Instead of an open domain, consider only the objects you know about

- If you detect a new object
  - add it to the domain
  - and replan with the new domain

- Seems reasonable, because our household robot is not supposed to simulate a Turing machine.
Completeness?

- What does completeness mean in this context?
  - If there is a plan under the open domain semantics, then there should be a sequence of plans generated by replanning over expanding universes such that the final one is successful.

- Clearly unachievable because of undecidability

- Unclear, how to formalize a guarantee for which we can achieve completeness
Soundness?

- **Universal quantification** in pre-conditions with open domain semantics can be problematic.

- However, universally quantified conditions make only sense if we quantify over known objects!

- **Note:**
  - You should know about all your tools!
  - Formulation of goal can be non-trivial, e.g.
    - *remove all known objects from all known tables*
Uncertainty (logical)

- Uncertainty is **produced** by
  - the initial state description
    - *The door is open or closed*
  - non-deterministic effects of actions
    - *Opening the door can be successful or not*

- Uncertainty is **reduced** by
  - observations / sensing actions
    - *Determine the state of the door*
  - (deterministic) action (possibly conditional) effects
    - *Closing the door*
Representing logical uncertainty

- Usually: *set of possible worlds*
- Drawback: *exponential blowup* wrt. single model (STRIPS) case

- Alternative: Use *three-valued logic*, where the third value means *unknown*
- You cannot represent anymore
  - Know(A ∨ B)
- but only
  - Know(A) ∨ Know(B)
What do we lose?

- Possible world semantics is necessary for reasoning by case over conditional action effects:
  - Initially $\text{Know}(A \lor B)$
  - Action to make $A$ true if $B$ was true:
    - $B$ false: then $A$ must be true
    - $B$ true: then $A$ will now be true
  - Similar: Sensing $\neg B$
    
- Not possible with three-valued logic!

- Does a household robot need to have diagnostic reasoning capabilities
Completeness / Soundness

- The original problem is 2-EXP-complete, while planning with a three-valued logic and sensing is EXP-complete.

- We clearly lose completeness!
  - We cannot deal with hidden variables

- However, soundness is preserved: Any plan under the possible world semantics is a valid plan under the three-valued semantics!
Limited observability & monotonicity

- The only way to acquire knowledge about the truth value of an fluent is to sense it.
- Observability is limited: sensing actions may have preconditions, e.g., being close to the relevant object.

- By sensing we monotonically decrease uncertainty.
- Non-deterministic actions might increase it, but we can assume that by monitoring no new uncertainty is generated.
Conditional planning/Policies

- We still have sensing action outcomes that are unknown at planning time
  - Can be viewed as a special kind of non-determinism

- Plans are branching plans or policies

- What is a valid plan?
  - *Strong plan*: Cycle-free plan that guarantees success regardless of the sensing outcomes
  - *Strong cyclic plan*: Possibly cyclic plan that guarantees that the goal is always reachable
  - *Weak plan*: Sequence of actions/observations that lead to the goal
Continual planning

- Instead of planning for every contingency, generate an “optimistic” plan
- Monitor execution and replan if necessary
  - Generate policy online

- Actually, this is an approach many people have taken
- For example: Probabilistic planners are outperformed by FF-replan (IPC-04 & -06)
  - on “probabilistically uninteresting” domains
- Question: What are we losing?
Completeness & soundness

- **Completeness**: For every cyclic strong plan, for every state reachable in the strong cyclic plan:
  - the replanning approach is able to generate a successful linear plan
    - *Easy*, just create a weak plan

- **Soundness**: At each state, create only a successful plan with the correct prefix action,
  - if there is strong cyclic plan with the appropriate action.
    - *Hard*, actually as hard as non-deterministic planning, i.e. EXP-hard
Conditions for soundness

- **Invertability**: Everything can be undone
  - Very strong and unrealistic but easy to check
    - household robots might want to throw things into the garbage can

- **Strongly connected state space**: every state is reachable from every other state by weak plans
  - Weaker, but still very strong and less easy to check

- **Dead-end free**: We can never reach a state from which no goal state is reachable
  - Realistic, but hard to check
Dead-ends visually

User controlled

Environment controlled

Goal
Checking for dead ends

- Algorithm for checking for presence of dead end from initial state \( i \)
  1. Guess state \( s \)
  2. Check whether \( s \) is reachable by a weak plan from initial state \( i \)
  3. Check that there is no weak plan from \( s \)

- Step 2 can be done in \text{PSPACE} \ (for prop. planning)
- The complement of step 3 can be checked in \text{PSPACE}
- Since \text{PSPACE} is closed under \text{non-determinism} and complement, checking is in \text{PSPACE}!
- Checking for dead ends is \text{not harder} than classical planning
Conclusion

- We reduced non-deterministic, partial observable, open domain planning to classical planning, sacrificing
  - completeness, but only for puzzle mode reasoning
  - a little bit of soundness, but we can provide guarantees

- We have specified a PSPACE checkable criterion for soundness preservation

- The sacrifices all seem to preserve the functionality of a household robot
Possible improvements & challenges

- Provide **empirical justifications** for the claims about efficiency, i.e.,
  - compare nondeterministic, partially observable domain planners with continual classical replanners
- Implement checkers/provers that **prove dead-end freeness** of a given domain
- Find other characterizations of **domains preserving soundness**
- Find ways to **mediate** between classical replanning and full nondeterministic conditional planning (get inspiration from circuit diagnosis)