Concurrency, Convertibility, and Complexity in POCL Planning

Harrison Oates

Supervisor: Pascal Bercher

School of Computing, Australian National University

Introduction

Non-sequential plan representations, especially partial order causal link (POCL) plans are useful for modelling concurrency, but their computational properties are often misunderstood. *Makespan* is the minimum time required to execute a plan under parallelism.

The below three figures are plans for the task of making breakfast. Figure 1 is a parallel plan, which comprises of action sets, Figure 2 is a partial order (PO) plan, and Figure 3 is a POCL plan. Each of these solve the breakfast problem with makespan 2.

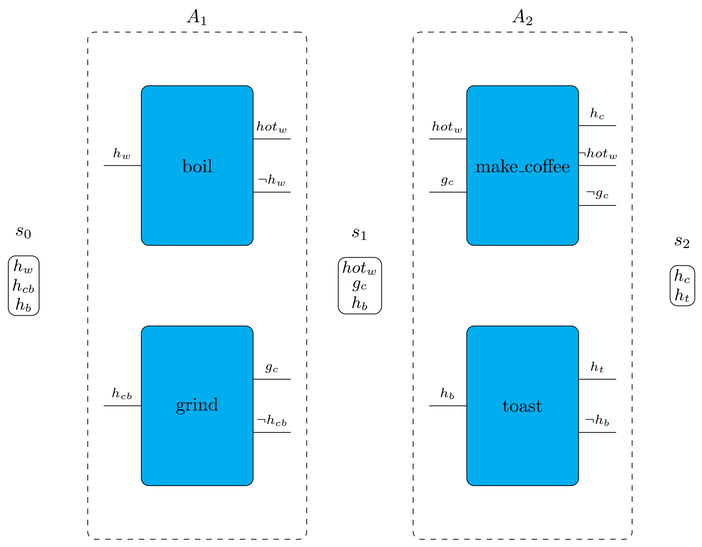

Figure 1. A parallel plan for the breakfast problem

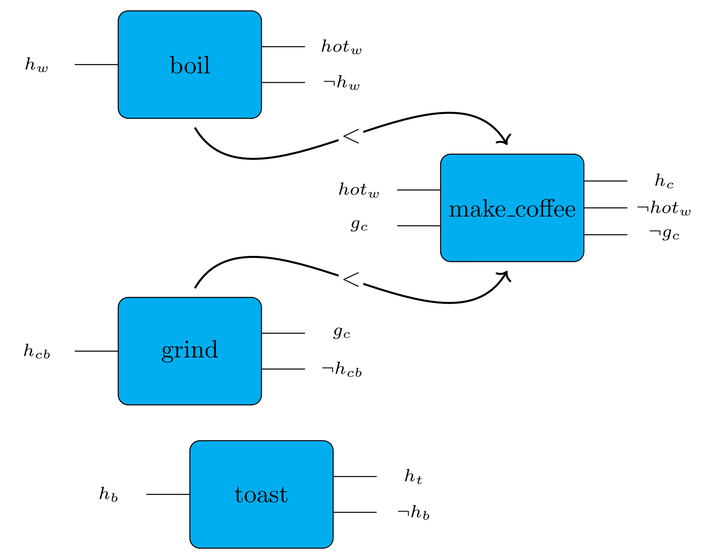

Figure 2. A PO plan for the breakfast problem. The arrows represent ordering constraints in $\prec$ necessary to ensure every linearisation is executable.

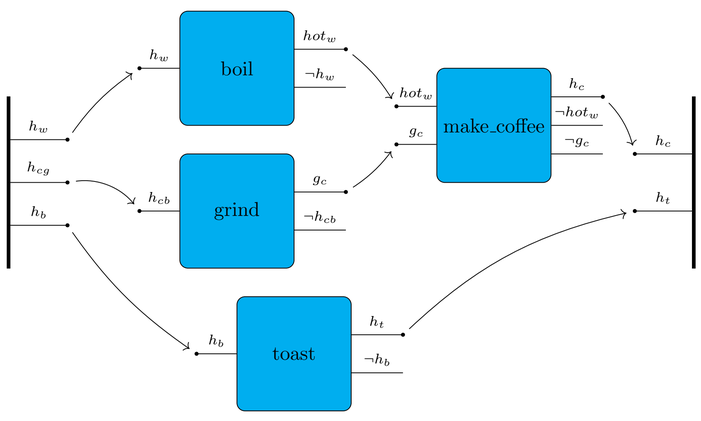

Figure 3. A POCL plan for the breakfast problem. Arrows depict causal links and implied ordering constraints are not shown.

Complexity Classes at a Glance

- **P**: Problems solvable in polynomial time (deterministic).
- **NP**: Solutions verifiable in polynomial time; finding them may require exponential time.
- **PSPACE**: Problems solvable using polynomial space (time may be exponential).
- **EXPTIME**: Problems requiring exponential time in the worst case.
- **NEXPTIME**: Nondeterministic exponential time; contains all problems in EXPTIME.
- **EXPSPACE**: Problems requiring exponential space; contains all previous classes.

Makespan Convertibility Does Not Hold In General

In the ordering

$$\text{Sequential} \rightarrow \text{Parallel} \rightarrow \text{PO} \rightarrow \text{POCL}$$

We can preserve makespan going from left to right, but right to left only works when converting from a POCL plan to a PO plan.

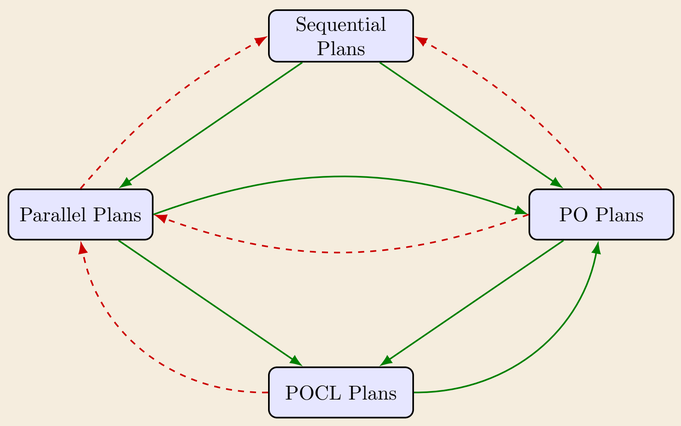

Figure 4. Makespan convertibility between plan representations. Solid green arrows indicate makespan can always be preserved. Dashed red arrows indicate makespan is not generally preserved.

The Counterexample

In the below plan, while a_1 and a_2 are unordered and thus share the same optimal release time, they have inconsistent effects, which violates the non-interference criteria for parallel plans. Therefore, a_1 and a_2 *must* be serialized in any valid parallel plan, even though the POCL structure allows them to be concurrent.

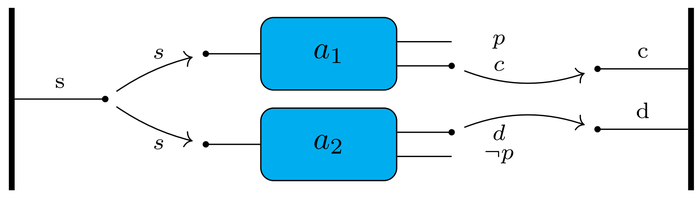

Figure 5. A POCL plan where unordered actions a_1 and a_2 have inconsistent effects.

Consequences Of Makespan Non-Convertibility

- **We refute Theorem 1** of Pecora et al. (2006), which claims that planning graph planners maximise PO concurrency.
- **Heuristics admissible for makespan-optimal parallel planning are inadmissible for makespan-optimal PO and POCL planning.** An example is h^m_p by Haslum and Geffner (2000).
- 'Optimal' POCL planners like CPT (Vidal and Geffner, 2006) that rely on these heuristics guarantee optimality *only within the space of POCL plans that can be represented as parallel plans*, not within the general space of POCL plans.

Makespan Complexity Results

- It is NP–complete to transform a POCL plan into a makespan-optimal parallel plan. Reduce from graph-colouring. We also establish tight bounds on the makespan!
- It is PSPACE–complete to decide if a planning problem has a parallel, PO or POCL plan with makespan $\leq k$ for **binary** k.
- It is NP–complete to decide if a planning problem has a parallel, PO or POCL plan with makespan $\leq k$ for **unary** k.

Deciding Lifted POCL Planning Problems

- We investigate POCL problems in the *lifted* setting, where action schemas use first-order terms rather than propositions in their preconditions and effects.
- Lifted reasoning avoids the combinatorial explosion of grounding all actions and predicates.
- **We introduce a novel framework that characterises delete relaxation across two orthogonal dimensions**: the scope of relaxation (which plan steps are relaxed) and whether causal links are respected.

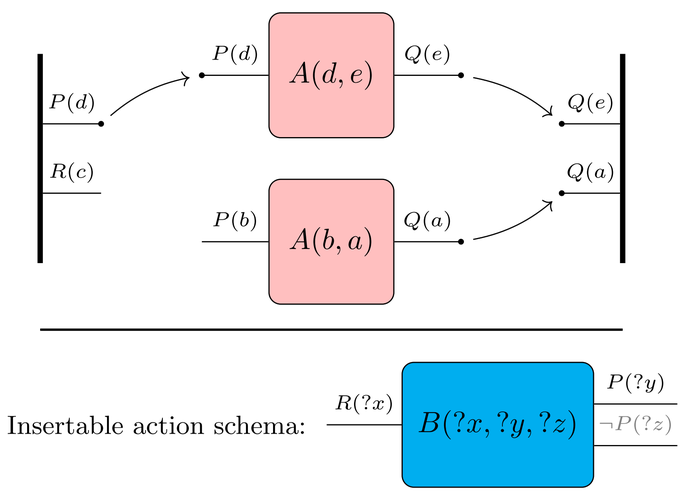

Figure 6. Inserting $B(c,b,d)$ into a plan with a causal link on $P(d)$. Under A-relaxation, the step is delete-free and poses no threat. Under A-relaxation respecting causal links, the unrelaxed action deletes $P(d)$, so a threat is detected, even though $P(d)$ won't actually be deleted in execution.

Lifted POCL Plan Existence Complexity Results

Relaxation Type	Initial Plan	New Action	Respects Links?	Complexity (Complete For)
None (Unrelaxed)	Original	Original	N/A	EXPSPACE
P-relaxed	Relaxed	Original	No	EXPSPACE
A-relaxed	Original	Relaxed	No	EXPTIME
AP-relaxed	Relaxed	Relaxed	No	EXPTIME
A-relaxed (RL)	Original	Relaxed	Yes	EXPTIME
AP-relaxed (RL)	Relaxed	Relaxed	Yes	EXPTIME

Table 1. Complexity of plan existence in lifted POCL planning under different delete relaxations. 'RL' variants respect causal links.

Complexity Results for Bounded POCL Problems

Bound Type	Complexity
Length-bounded (All)	NEXPTIME–complete
Makespan-bounded (Unrelaxed)	NEXPTIME–complete
Makespan-bounded (Relaxed)	in NEXPTIME

Table 2. Complexity of bounded plan refinement in lifted POCL planning. 'Relaxed' refers to all delete relaxation variants.

No Gain from Further Relaxation

Unlike in the ground setting, changing the scope of delete relaxation does not fundamentally reduce the complexity of lifted POCL planning.

- For both **A-relaxation** (new actions are delete-free) and **AP-relaxation** (all actions are delete-free), plan existence remains **EXPTIME–complete**.
- This holds even if the initial plan is totally ordered or if pre-existing causal links are protected.
- **Insight:** The hardness is rooted in reasoning about variable assignments and the exponential number of potential groundings, which is not simplified by ignoring delete effects.

If we bound the length or makespan of the POCL plan, delete relaxation does not make the problem easier at all!

Australian National University