Reviving Partial-Order Causal-Link Planning:
Is It Possible? Is It Worth It?

Scott Howsam, Harrison Oates, Pascal Bercher

School of Computing
The Australian National University

HSDIP 2026



Australian
National
University

Why Study POCL Planning?

Partial-Order Causal-Link (POCL) planning was dominant through the early 1990s.

Structural advantages:

- *Least commitment:* only order actions when forced
- *Explicit causal links:* direct pruning of threats
- *Compact representations:* one node = many total orders

Still relevant today:

- HTN planning
- Temporal / concurrent planning
- Plan explanation & repair

Why Study POCL Planning?

Partial-Order Causal-Link (POCL) planning was dominant through the early 1990s.

Structural advantages:

- *Least commitment:* only order actions when forced
- *Explicit causal links:* direct pruning of threats
- *Compact representations:* one node = many total orders

Still relevant today:

- HTN planning
- Temporal / concurrent planning
- Plan explanation & repair

Despite these advantages, POCL planners have been consistently outperformed by state-based planners since the late 1990s.

This paper asks: can we finally understand *why*, and chart a path forward?

Planning Formalism

A **planning domain** $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$:

- $\mathcal{V}$: finite set of state variables (fluents); a state $s \in 2^{\mathcal{V}}$
- $\mathcal{A}$: finite set of actions; each $a = \langle pre(a), add(a), del(a) \rangle$
- a applicable in s iff $pre(a) \subseteq s$; generates $(s \setminus del(a)) \cup add(a)$

Running example — Making Breakfast:

- **boil:** $Water \rightarrow HotWater, \neg Water$
- **grind:** $CoffeeBean \rightarrow Grounds, \neg CoffeeBean$
- **make_coffee:**
  $HotWater, Ground \rightarrow Cofee, \neg HotWater, \neg Ground$
- **toast:** $Bread \rightarrow Toast, \neg Bread$

Initial state: $\{Water, CofeeBean, Bread\}$

Goal: $\{Coffee, Toast\}$

Partial Plans

A **(partial) POCL plan** $P = (PS, \prec, CL)$:

- PS Set of *plan steps* (labelled action instances) including ps_{init} and ps_{goal}
- $\prec$ Partial order on PS ("a must execute before b")
- CL Set of *causal links* (ps_p, v, ps_c): ps_p provides v for ps_c, protecting it from deletion

A plan P is a *solution* when:

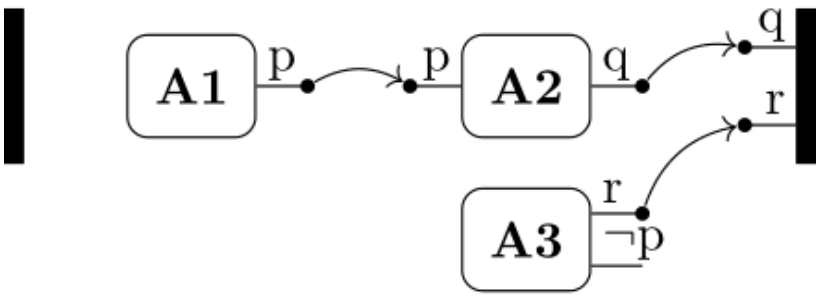
- Every precondition of every step is supported by some causal link
- There are no *causal threats*

Search starts from the initial plan P_{init} containing only ps_{init} and ps_{goal}.

Flaws: Open Preconditions & Threats

Open precondition:
A precondition of some *ps* not yet supported by a causal link.

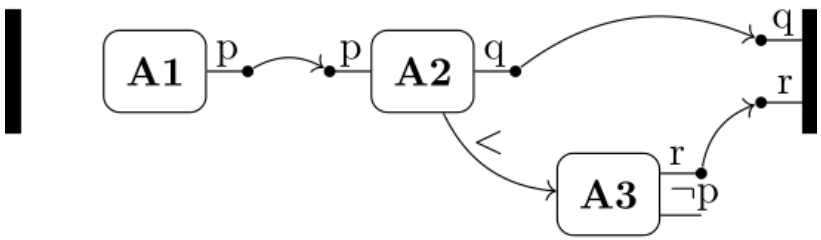
Resolver: add an action *a* (or reuse existing step) and a causal link.



Unresolved threat

Causal threat:
Step ps_t threatens link (ps_p, v, ps_c) if $v \in del(ps_t)$ and ps_t is unordered w.r.t. the link.

Resolver: add ordering $ps_t \prec ps_p$ or $ps_c \prec ps_t$.



Threat resolved by ordering

The POCL Search Algorithm

1. **Select** a partial plan P from the fringe
2. If P has **no flaws** $\Rightarrow$ **return** P (solution!)
3. **Select** a flaw f from P
4. For each **resolver** r of f:
   - $P' \leftarrow P \oplus r$ (add action / causal link / ordering)
   - If P' is consistent, push P' onto the fringe
5. **Repeat** from step 1

The POCL Search Algorithm

1. **Select** a partial plan P from the fringe
2. If P has **no flaws** $\Rightarrow$ **return** P (solution!)
3. **Select** a flaw f from P
4. For each **resolver** r of f:
   - $P' \leftarrow P \oplus r$ (add action / causal link / ordering)
   - If P' is consistent, push P' onto the fringe
5. **Repeat** from step 1

Key design choices:

- *Plan selection:* best-first requires a good heuristic $h(P)$
- *Flaw selection:* typically "fewest resolvers first"

Both important, but we focus on plan selection. Flaw selection has been comprehensively investigated by Younes and Simmons (JAIR 2003) and others.

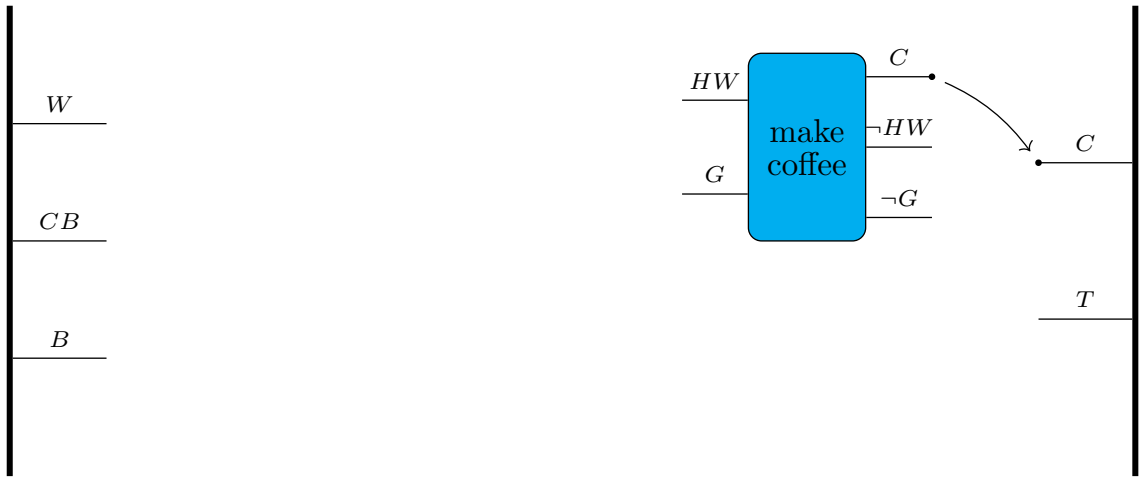
Plan Refinement: Breakfast Example



Initial plan: ps_{init} (produces W, CB, B) and ps_{goal} (needs C, T).

Flaws: ps_{goal} needs C; ps_{goal} needs T.

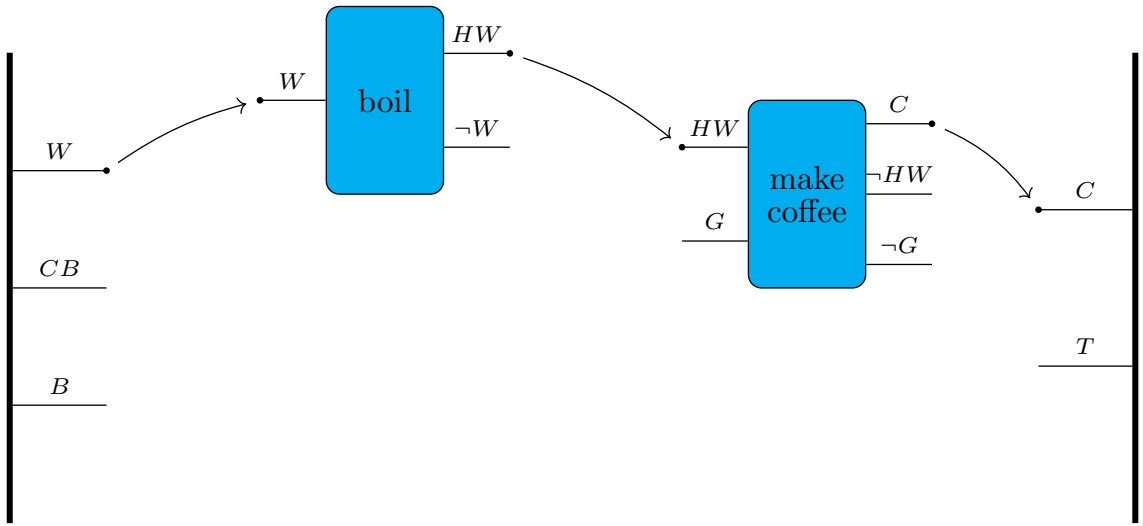
Plan Refinement: Breakfast Example



Added `make_coffee`; causal link `make_coffee`: $C \rightarrow$ `goal`: C.

Remaining: `make_coffee` needs HW, G; `goal` needs T.

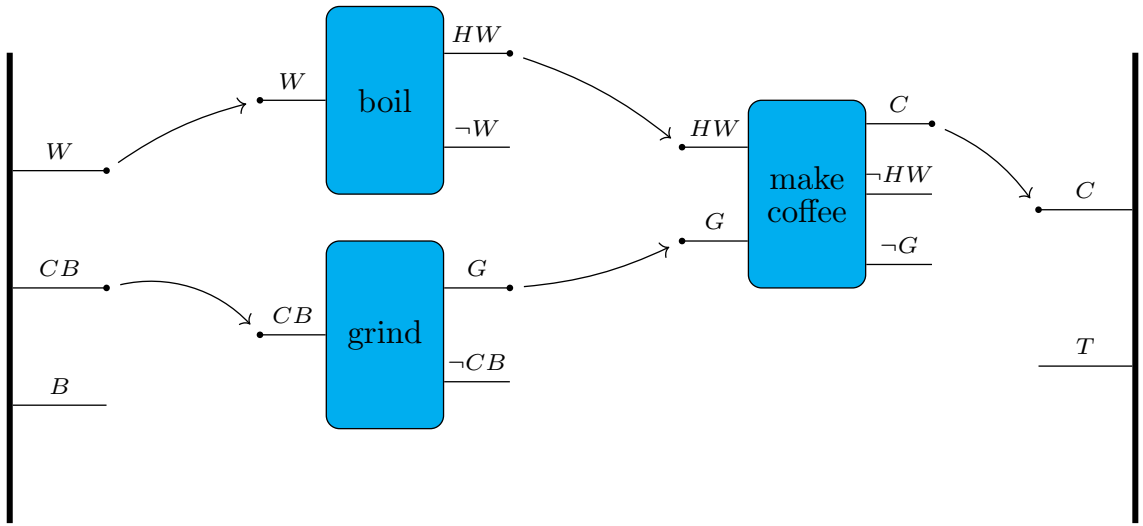
Plan Refinement: Breakfast Example



Added `boil`; links `init`: $W \rightarrow$ `boil` and `boil`: $HW \rightarrow$ `make_coffee`.

Remaining: `make_coffee` needs G; `goal` needs T.

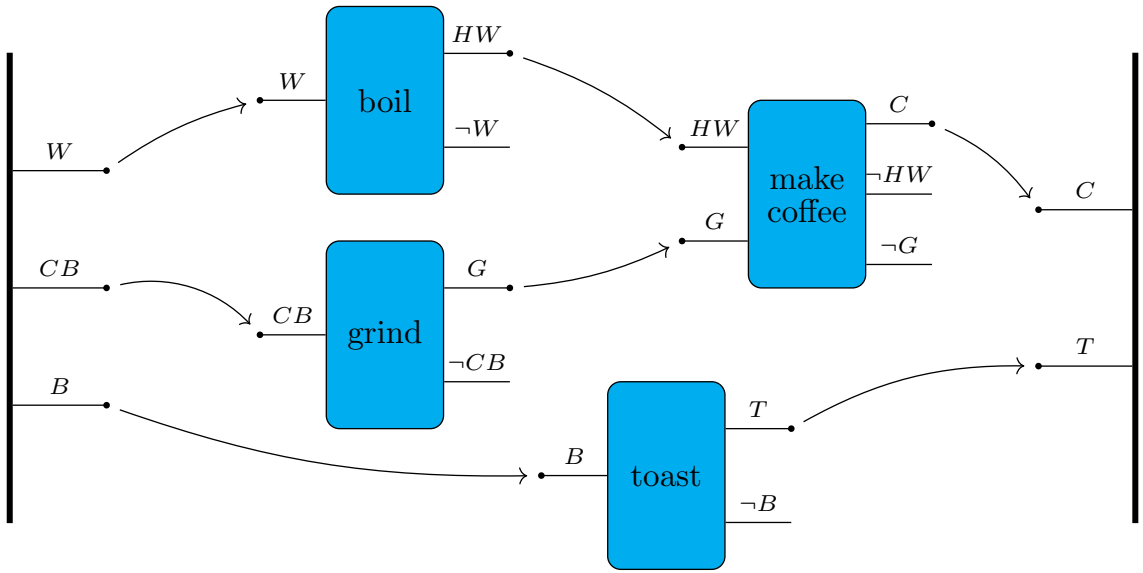
Plan Refinement: Breakfast Example



Added `grind`; links `init`: $CB \rightarrow$ `grind` and `grind`: $G \rightarrow$ `make_coffee`.

Remaining: `goal` needs T.

Plan Refinement: Breakfast Example



Added `toast`; links `init`: $B \rightarrow$ `toast` and `toast`: $T \rightarrow$ `goal`.

No more flaws — solution found!

Three Decades of Revival Attempts

Heuristic	**Key Idea**
Relax (RePOP)	Adapts h^{FF} to partial plans
Additive (VHPOP)	Adapts h^{add} to partial plans
Sample-FF	Sample linearisations of partial plan and then calculate h^{FF} between them
Lplan	LP relaxation of plan existence
State-based encoding	Compile node $\rightarrow$ classical problem, apply state-based heuristic to resulting state

We give a polynomial-time compilation of causal links into classical planning problems to improve Lplan and state-based heuristics.

Pattern: No approach achieves competitive performance with contemporary state-based methods.

Three Decades of Revival Attempts

Heuristic	Key Idea
Relax (RePOP)	Adapts h^{FF} to partial plans
Additive (VHPOP)	Adapts h^{add} to partial plans
Sample-FF	Sample linearisations of partial plan and then calculate h^{FF} between them
Lplan	LP relaxation of plan existence
State-based encoding	Compile node $\rightarrow$ classical problem, apply state-based heuristic to resulting state

We give a polynomial-time compilation of causal links into classical planning problems to improve Lplan and state-based heuristics.

Pattern: No approach achieves competitive performance with contemporary state-based methods.

Why? Two fundamental, intertwined challenges.

Challenge 1: The NP-Hardness Barrier

Theorem 1 (Bercher 2021)

Delete-relaxed POCL planning is solvable in *polynomial time* if **either**:

(a) $\prec$ is a *total* ordering, **or**

(b) The initial partial plan is delete-relaxed *and* causal links are not respected.

Otherwise, it is **NP-complete**.

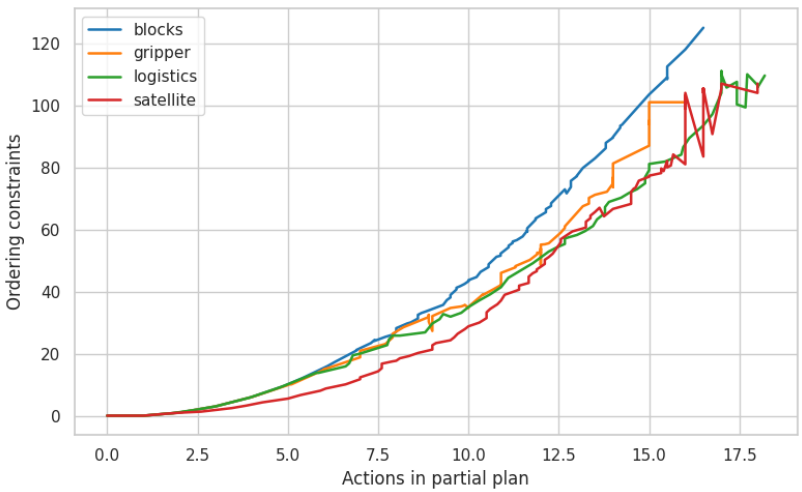
Implication: any polynomial heuristic must abandon either partial orders *or* causal/delete information from the initial partial plan

- Doing this is analogous to ignoring information from the initial state in state-based planning

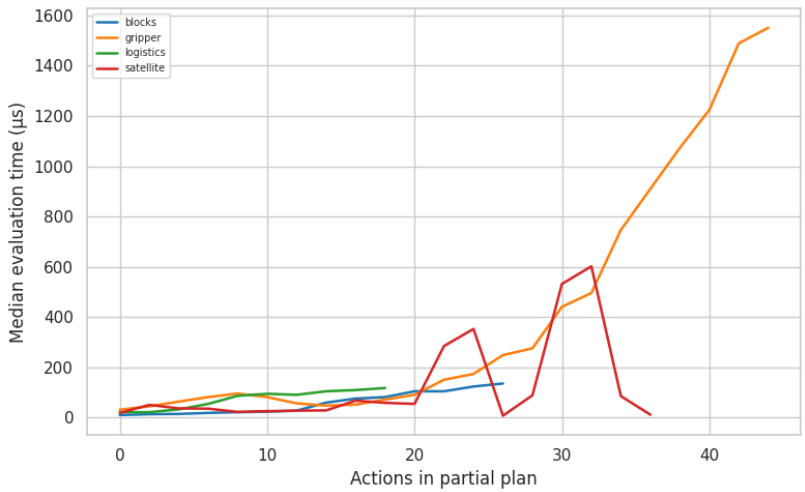
Challenge 2: Quadratic Node Growth

Theorem 2

There exist planning instances with n steps such that any partial plan satisfying Bäckström's least-commitment optimality criteria must contain $\Omega(n^2)$ ordering constraints. Partial-plan size grows *quadratically* with search depth.



Ordering constraints vs. plan size ($R^2 \geq 0.98$)



Heuristic eval. time vs. plan size

The Informativeness-Tractability Tradeoff

The two challenges are intertwined:

- Only way to achieve tractable heuristics known is to ignore information that makes POCL planning attractive...

The Informativeness-Tractability Tradeoff

The two challenges are intertwined:

- Only way to achieve tractable heuristics known is to ignore information that makes POCL planning attractive...
- But the remaining information compounds quadratically with depth.

The Informativeness-Tractability Tradeoff

The two challenges are intertwined:

- Only way to achieve tractable heuristics known is to ignore information that makes POCL planning attractive...
- But the remaining information compounds quadratically with depth.

Every previous revival attempt lands in one of the “failed” regions of this design space.

Unexplored Design Space

Several promising directions have never been tried for POCL planning:

Incremental computation Update the heuristic *across refinements* rather than recomputing from scratch at each node (analogous to incremental SAT solving)

Width-based search Measure *novelty* of new atom tuples rather than goal distance—sidesteps NP-hardness entirely

Pattern databases Precompute goal distances on abstract problems; constant-time lookup at search time

Operator counting LP framework combining constraints from landmarks + abstractions

Hybrid approaches Exploit POCL advantages early, switch representation when node size explodes

Total-order plan-space Keep causal links but fix the ordering—avoids NP-hardness without losing all POCL benefits

Most Promising: Incremental & Width-Based

Incremental computation

Each refinement adds a step, a CL, or an ordering. Instead of recomputing *h* from scratch, *update* it:

- Compatible with best-first search
- Could amortise quadratic cost to near-constant per refinement
- Analogous to incremental SAT / watched literals

Width-based search

Measure the *novelty* of a new partial plan: has this combination of fluents / causal links / ordering atoms been seen before?

- Scales with novelty threshold, not node size
- No NP-hard sub-problem
- Can be applied to many features of POCL, e.g. flaw novelty, CL novelty, ordering novelty

Hybrid Approaches

Core idea: exploit the least-commitment advantages of POCL early in search, but avoid the quadratic cost explosion as plans grow. Some concrete examples we give (non-exhaustive):

Option A — Size threshold: once $|PS|$ exceeds a bound k, compile to a classical problem and switch to state-based search

Option B — Step fusion: compress the partial plan by merging step pairs that are always consecutive, reducing $|PS|$

Option C — Sub-problems: spawn fresh subproblems from current partial plan to make sub-plans which can be subsequently stitched together

Each option targets the node-growth challenge.

Conclusion

Is revival *possible*?

- Probably not via previous paths (delete-relaxation)
- A huge unexplored design space remains
- The community may have abandoned POCL prematurely

Is it *worth it*?

- Yes—especially beyond classical planning
- HTN and temporal planners already build on POCL structures but lack heuristic guidance
- Explicit causal structure enables shorter makespans and richer plan explanations