Systematic Nonlinear Planning

Written by: David McAllester and David Rosenblitt
Published: December 1991
Presented by: Tyler Ferry

Introduction

- The paper presents a simple, sound, complete, and systematic algorithm for domain-independent STRIPS planning.
- A ground version of Tate's NONLIN procedure from 1976
- Planning procedures have used three basic techniques to optimize the required search process:
  1. Use of "Lifting"
  2. Nonlinear construction
  3. Use "abstraction spaces" where planning is done at a higher level first. Also called "least commitment planners"

STRIPS Planning Overview

- Stanford Research Institute Problem Solver
- Introduced by Fikes & Nilsson in 1971 as a formal model for “common-sense” planning.
- Later proved in 1985 by Canny that formal STRIPS plans are PSPACE-complete but can be optimized with various techniques.
- Uses operators with preconditions, add lists, and delete lists to help solve problems.
- Debatably ineffective for larger problems

STRIPS Operator

STRIPS operator consists of:

- Operator name
- A prerequisite list
- An add list
- A delete list

Example: MOVE(A, B, C)
Move block A from block B to block C

- Prerequisites: CLEAR(A), ON(A,B), CLEAR(C)
- Add effects: ON(A, C), CLEAR(B)
- Delete effects: ON(A, B), CLEAR(C)

STRIPS Solution

A STRIPS planning problem is a triple <O, Σ, Ω>

- O = set of STRIPS operators
- Σ = set of initial propositions
- Ω = set of goal propositions

A solution to problem <O, Σ, Ω> - a sequence of operations $\alpha \in O$, s.t. the result of consecutively applying the operations in α starting with the initial state Σ results in a set that contains the goal set Γ

Other key concepts

Nonlinear planning - partial order plan instead of total order

- A plan consisting of a symbol table, set of causal links, and safety conditions

Symbol table - a mapping from a finite set of step names to operators

- Must contain both START (an operator with no prereqs or deletes) and FINISH (empty add and delete list) steps
- Doesn't impose ordering

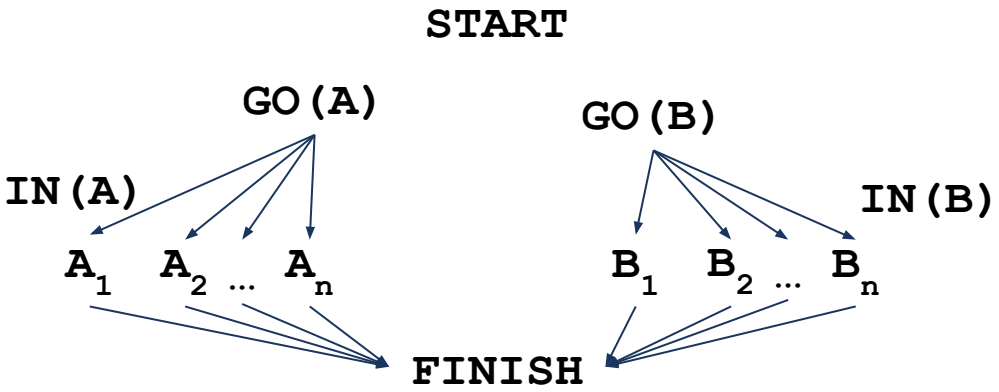
Causal link - a triple <s, P, w>

- s is step name with P in its add list
- P is a propositional symbol
- w is a step name with P as a prerequisite
- Written as $s \xrightarrow{P} w$

Causal link

Causal link $s \xrightarrow{P} w$ requires that:

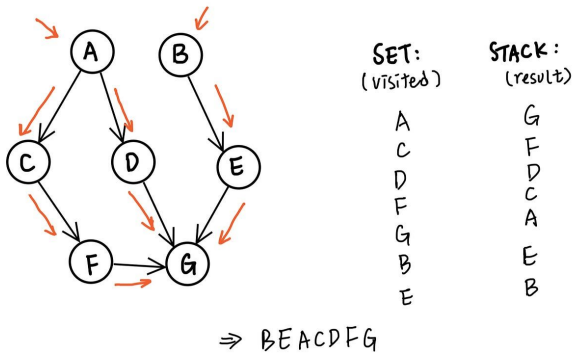
- Step *s* precedes step *w (s<w or w>s)*
- No step between s and w either adds or deletes P
  - A step that adds or deletes P is considered a **threat**
  - A **safety condition** is an ordering of steps such that s<w
- Example:
  - A robot must perform tasks that must be done in a specific room. The name of the task indicates the room that the robot needs to be in
  - Below shows a plan of causal links
  - Causal link $\mathtt{GO(A)} \xrightarrow{\mathtt{IN(A)}} \mathtt{A_1}$ says task A_1 has the prerequisite of IN(A) which is given by GO(A)



Topological Sort

- A topological sort is a linear sequence of all the step names in the symbol table such that:
  - The first step is START
  - The last step is FINISH
  - For each causal link $s \xrightarrow{P} w$ in the plan, step s precedes w
  - For each safety constraint u < v (or v > u) in the plan, the step u precedes step v

- A topological sort of a nonlinear plan is a **solution**
- A nonlinear plan with no topological sort is considered **order inconsistent**
- Topological sort ensures no redundant steps



Algorithm and Completeness

Algorithm overview

1. Define initial state and goals
2. Create causal links to track dependencies
3. Detect threats
4. Apply safety conditions
5. Search to eliminate redundant plans
6. Run FIND-COMPLETION on plan Z with cost bound c:

IF order inconsistent
Fail
IF Z is complete
RETURN Z
ELSE IF there is a causal link between steps s and w **AND** threat v exists
RETURN either (nondeterministic):
FIND-COMPLETION(Z+(v<s), c) or
FIND-COMPLETION(Z+(v>w),c)
ELSE
//There now must be a step w and Prereq P s.t. there is no causal link
//Find step or operator named s that adds P
RETURN FIND-COMPLETION(Z+ $s \xrightarrow{P} w$, c)

Lifting

- Technique invented in 1965 that is now standard for algorithm writing
- Any ground expressions can be made into symbolic “lifted” expressions.
  - Can be formed with substitution
- Example:
  - Block problem with n blocks has n^3 possibilities for MOVE(A,B,C)
  - These possibilities are just different instances of MOVE(x,y,z)
  - Rather than create n^3 new symbols in the symbol table, the algorithm only needs to create one using copies of fresh variables
- Treated as a separate optimization
  - Decreases algorithm complexity

References

- McAllester, David; Rosenblitt, David. Systematic Nonlinear Planning. December 1991
- Topological search image (slide 7): Medium, Course Schedule and Topological Sorting