PLANNING WITH GRAPHPLAN APPROACH

Ivan Bratko
FRI, University of Ljubljana

2021/22

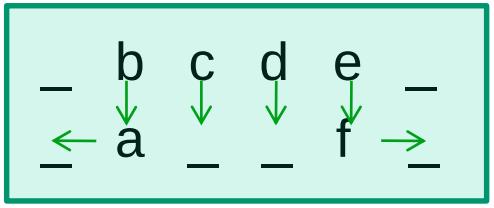
THE GRAPHPLAN APPROACH

Planning graphs enable compact representation of a large set of plans

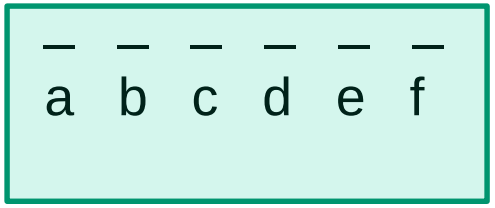
PLANNING GRAPH ENABLES ECONOMICAL REPRESENTATION OF MANY PLANS

- Example (This is equivalent to Russell&Norvig (3rd ed) exercise 11.12):

Start state:



Goal:

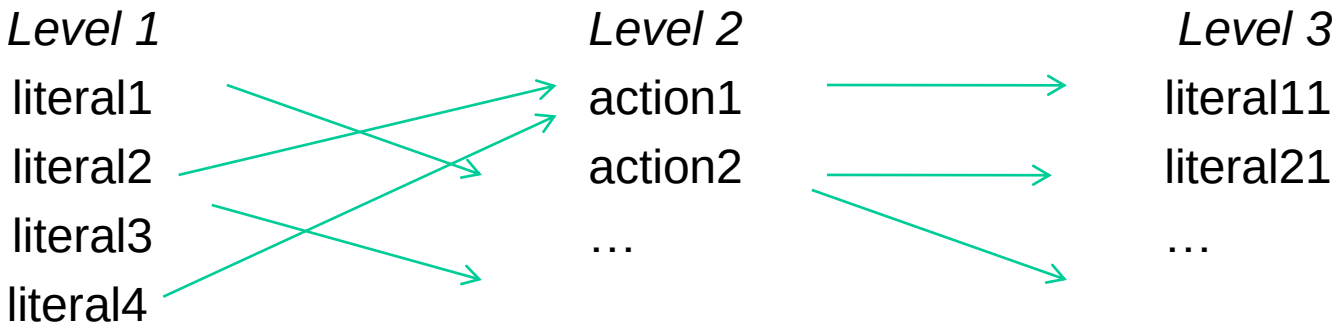


- 180 linearisations of min. POP
  (permutations of 6 actions, constr. “a” before “b”, “f” before “e”:
  6! / (2*2) = 180)
- 4 GRAPHPLAN plans

- 4 GRAPHPLAN plans
  - [a,f], [b,c,d,e]
  - [a,c,f], [b,d,e]
  - [a,d,f], [b,c,e]
  - [a,c,d,f], [b,e]
- These 4 (parallel) plans represent 180 linearisations (totally ordered) plans

PLANNING GRAPH

- Consists of interchanging levels of literals and levels of actions
- Each level = (set of literals, set of actions)

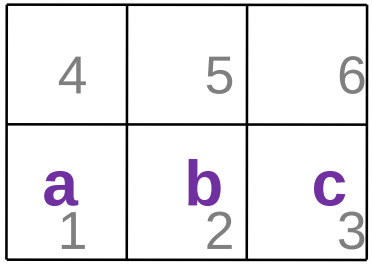


- Literal → Action means: Literal is a precondition for Action
- Action → Literal means: Literal is an effect of Action

- "Literal" in logic means predicate expression like: at(a,1), ~clear(3)

EXAMPLE PLANNING PROBLEM

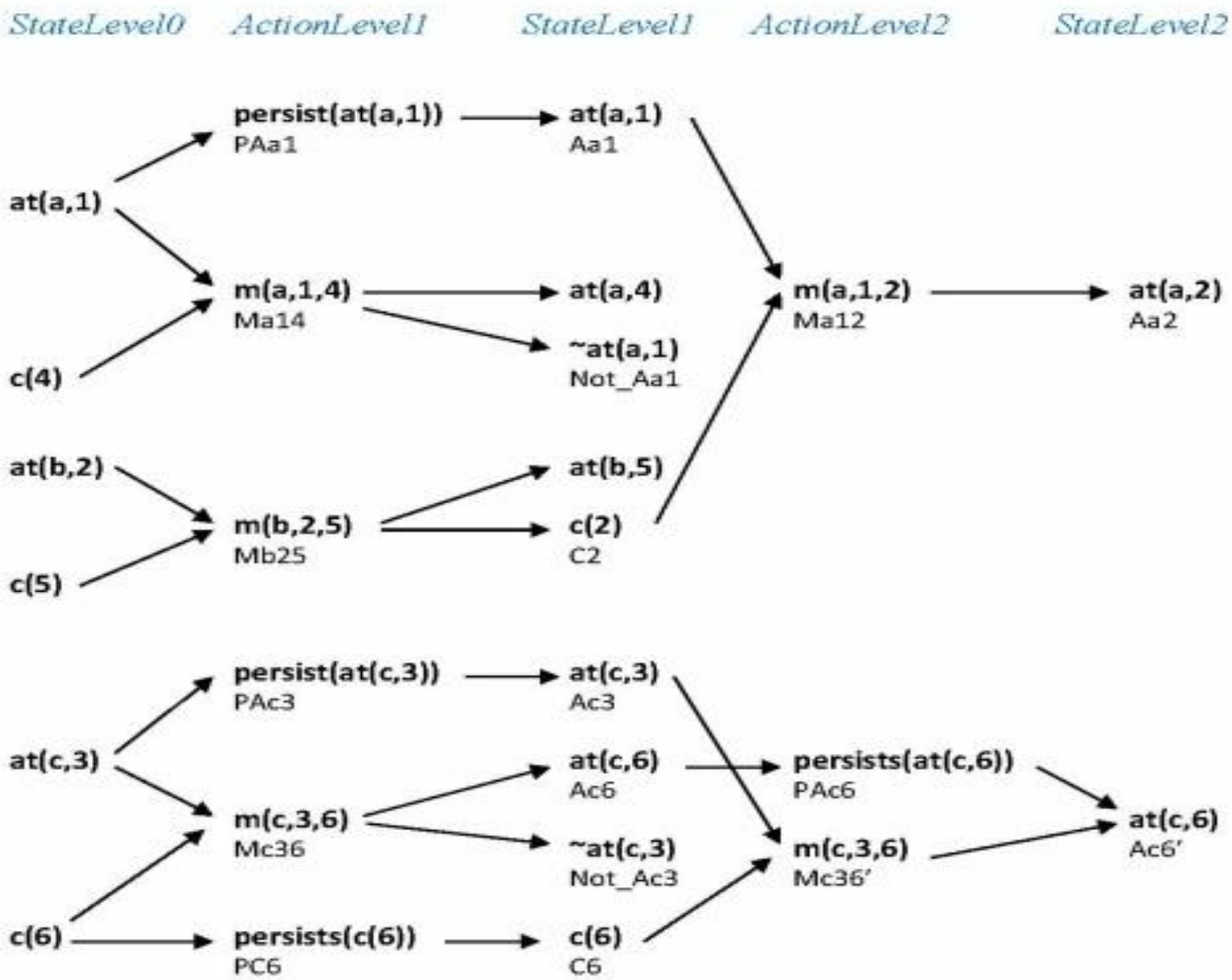
Start state:



Goals: at(a,2), at(c,6)

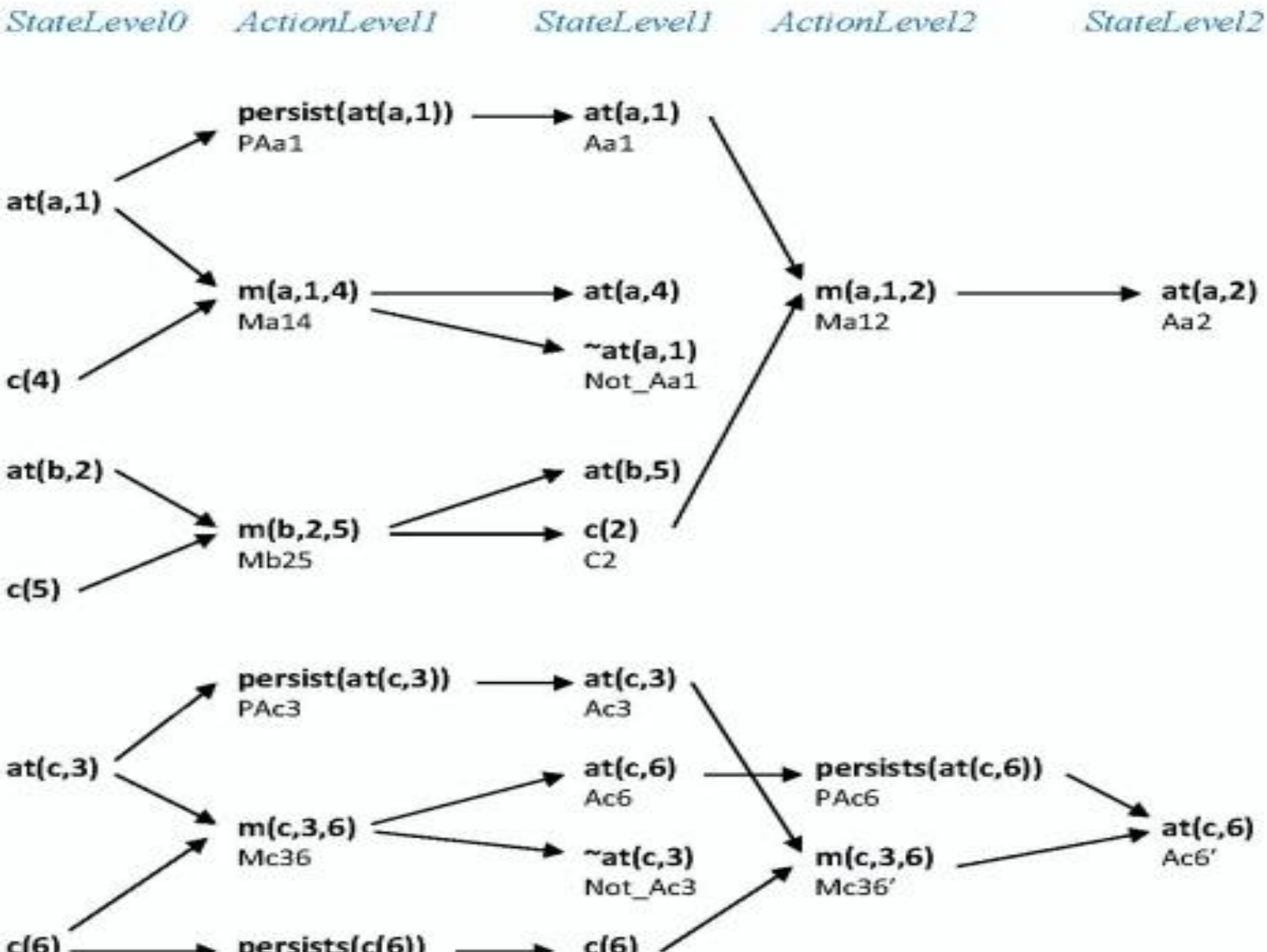
Planning graph for this problem – next slide

Example planning graph, goals are at(a,2) and at(c,6) (from Bratko, Prolog Programming for AI, 2012)



Note: This is only part of planning graph

Deriving a plan from this planning graph by logical reasoning
Find values of Boolean variables Aa2, Ma12 ...



Logical relations between truth of literals entail partial order plans

- Aa2 ^ Ac6’ % Goals of plan, so Aa2 and Ac6 are true
- Ma12 ⇔ Aa2 % only m(a,1,2) achieves at(a,2)

  So Ma12 must be true
- Ma12 → Aa1 ^ C2 % Preconds. for m(a,1,2)

  So Aa1 and C2 must be true
- PAa1 ⇔ Aa1 % only persist(at(a,1)) achieves at(a,1)

  So PAa1 must be true
- Also, Mb25 must be true to ensure C2
- And similar for bottom part of planning graph:

  to ensure Ac6’ we need either PAc3 and Mc36’,

  or Mc36 and PAc6’
- So a partial order plan is [[m(b,2,5)] , [m(a,1,2), m(c,3,6)]]

“MUTEX” LITERALS AND ACTIONS

- Literals may be **mutually exclusive** (abbreviated as mutex)
- E.g.: at(a,1) and ~at(a,1) So: Aa1 XOR Not_Aa1
- Also actions may be mutex,
  e.g. persist(at(a,1)) and m(a,1,2), because they have mutex effects

TRUTH OF LITERALS

- Literal appearing at a level intuitively means: “Literal *could* be true at this level, but we cannot be sure”.
- Literal not appearing at a level means: “Literal *cannot* be true at this level”
- Action appearing at a level could be possible to execute (but we cannot be sure)
- Action not appearing at a level means: “This action cannot be executed at this level”
- *Persistence actions* are virtual actions that represent persistence of literals through inactivity. For each literal L at a literals level, there is a persistence action whose precondition is L, and effect is also L.

PLANNING GRAPH

- In this sense, planning graph is **complete and not sound**. This means: it represents all potential solutions, but some of them may not be possible
- Planning graph only takes into account some of the negative interactions among actions, thus only detecting some things that are not possible
- Planning graph is **optimistic** in judging what literals are true
- Planning graph technique only available for propositional descriptions (no variables in literals and actions!)

COMPLEXITY OF CONSTRUCTING PLANNING GRAPH

- **Low order polynomial** in the number of actions and literals
- In practice, this means much better efficiency than other planning approaches (basic means-ends, goal regression, POP planning)
- But **extracting** a plan from planning graph is **exponential**

WHY IS PLANNING GRAPH ECONOMICAL COMPARED TO STATE SPACE?

- Consider 32 robots on 8x8 grid

- # states (positions of all robots): 64 * 63 * 62 * ... * 33 = $4.8 * 10^{53}$

- # all literals: ~ 4200
  at(Robot, Cell) 32 x 64 x 2 (#robots * #cells * +/- literal)
  clear(Cell) 64 * 2
- # Actions: m(Robot, Cell1, Cell2): 32 * 64 * 4 ~ 8000

- Size of planning graph: (#literals + #actions) * #levels
  E.g. 10 action levels, assume ALL possible at each level:
  ~ 120000 (compared to $4.8 * 10^{53}$)
- Where did exponential complexity disappear? Constraints!
- However, **solving constraints** still has **exponential complexity**

MUTEX LINKS BETWEEN ACTIONS

- “Mutex” = mutually exclusive
- Mutex links may connect two objects of the same level
- A mutex link between two actions A1 and A2 means: A1 and A2 cannot be both executed at that level
- No mutex link between A1 and A2 means: both A1 and A2 *could* be executed at this level in any order or in parallel (but we cannot be sure)

MUTEX LINKS BETWEEN LITERALS

- Analogically defined for mutual exclusion of literals
- A mutex link between two literals means: the two literals cannot be both true
- No mutex link betwen two literals: both literals *could* both be true

COMPLEXITY OF CONSTRUCTING PLANNING GRAPH

- Low order polynomial in the number of actions and literals
- In practice, this means (much) better efficiency than other planning approaches (basic means-ends, goal regression, partial order planning), although the extraction of a plan from a planning graph still has exponential complexity
- Specially useful: planning graph as a **source of heuristics** for heuristic search

CONSTRUCTING A PLANNING GRAPH

- Start with level S0 (conditions true in start state)
- Next level, A0: actions that have their preconditions satisfied in previous level
- Also include virtual “persistence” actions (that just preserve literals of previous level)
- Result is a *planning graph* where state levels S_i and action levels A_i are interleaved
- Ai contains all actions that are executable in S_i, and *mutex* constraints between actions
- S_i contains all literals that could result from any possible choice of actions in A_{i-1} plus mutex constraints between literals

MUTEX CONSTRAINTS

- Mutual exclusion constraints = “mutex” constraints
- E.g. move(a,1,2) and move(c,3,2) are mutex
- E.g. clear(4) and ~clear(4) are mutex

MUTEX RELATION BETWEEN LITERALS

- Mutex relation holds between two literals L1 and L2 of the same level if:
  - L1 is negation of L2, or
  - L1 and L2 are mutex due to intrinsic properties of the domain (e.g. at(robot1, loc1) and at(robot1, loc3) are mutex), or
  - Each possible pair of actions that could achieve L1 and L2 respectively, are mutually exclusive

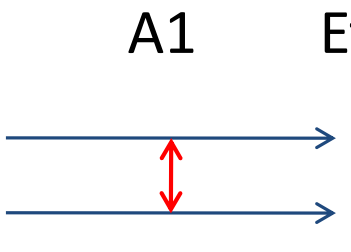
MUTEX RELATION BETWEEN ACTIONS

- Mutex relation holds between two actions A and B at the same level if:
  - An effect of A is inconsistent with an effect of B ("inconsistent effects"), or
  - An effect of A is inconsistent with a precondition of B ("interference"), or
  - A precondition of A is inconsistent with a precondition of B ("competing needs")

MUTUAL EXCLUSION BETWEEN ACTIONS: INCONSISTENT EFFECTS

Two actions A1 and A2 of the same level are mutex if

- Case 1: They have *inconsistent effects*:



inconsistent(p,q), e.g. q = ~p

Examples of inconsistent effects:

adds(A1) U deletes(A2) ≠ Ø

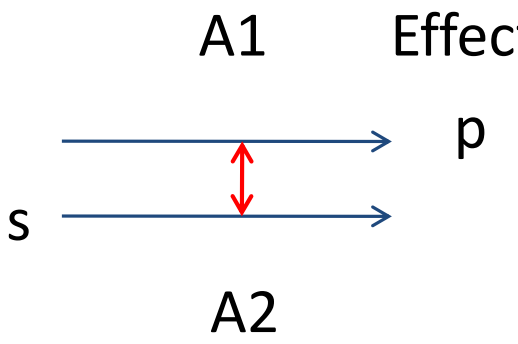
or

L1 in adds(A1) and L2 in adds(A2) and inconsistent(L1,L2)

e.g. L2 = ~L1

MUTUAL EXCLUSION BETWEEN ACTIONS: INTERFERENCE

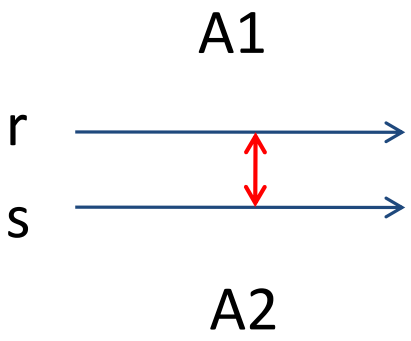
- Case 2: There is *interference* between A1 and A2: A precondition of A2 is inconsistent with an effect of A1



p in effects(A1) and s in precond(A2) and inconsistent (p,s), e.g. s = ~p

MUTUAL EXCLUSION BETWEEN ACTIONS: COMPETING NEEDS

- Case 3*: Competing needs*: A precondition of A1 is inconsistent with an effect of A2



r in precond(A1) and s in precond(A2) and inconsistent(r,s), e.g. s = ~p

MUTUAL EXCLUSION BETWEEN LITERALS

Two literals P1 and P2 of the same level are mutex if

- inconsistent(P1,P2), e.g. P2 = ~P1, e.g. P1 = ~P2

- P1 and P2 have *inconsistent support*: all pairs of actions A1 and A2 that could achieve P1 and P2 are mutex:

  (all(A1,A2):

  P1 in effects(A1) and P2 in effects(A2) and mutex(A1,A2))

  → mutex(P1,P2)

NOTES

- Two actions of the same level that are not mutex could be executed in parallel, or sequentially in any order.

- Planning graph has these properties:
  - It shows which actions may possibly be executed at a given level
  - Graph makes no guarantee that the actions will actually be executable at that level
  - However, if an action does not appear at a level then the action certainly will not be executable at that level

USING PLANNING GRAPH AS HEURISTIC

- Planning graph can be used as source for various heuristics
- E.g. a literal not appearing in the final level of a graph cannot be achieved by any plan.
- Level in which a goal literal appears can be used as an estimate of the cost (in terms of levels) of achieving that goal
- How can the (non)appearance of a literal L in level i be used for heuristic reasoning?
- If L does not appear in level i then no plan of i levels exists for L
- If L does appear, there *may* be such a plan
- Can a planning graph be used as source of heuristics for POP planning?

GRAPHPLAN ALGORITHM

- A plan can be extracted directly from a planning graph (unlike just using planning graph as source of heuristics)
- GRAPHPLAN algorithm constructs a planning graph and extracts a plan from the graph
- GRAPHPLAN iterates between two main steps:
  - Try to extract a plan from the current planning graph (if plan is extracted then stop)
  - Expand the planning graph by adding another action level and the resulting state level

GRAPHPLAN ALGORITHM

GIVEN: Goals of plan, Start state of the world

Graph := initial state level

WHILE (plan not found) and (solution still possible) DO

IF Goals all non-mutex in last level of Graph

THEN try to extract Plan from Graph so that Plan achieves Goals

IF plan successfully extracted THEN set “plan found”

ELSE

Graph := EXPAND_GRAPH(Graph)

IF Graph was not possible to expand then set “solution impossible”

EXTRACTING PLAN FROM GRAPH

- One possibility:
  - view plan extraction problem as a binary CSP problem (Constraint Satisfaction problem; variables correspond to actions in graph, domains are {in_plan, not_in_plan}

DOMAIN-SPECIFIC INCONSISTENT LITERALS

- Pairs of literals that cannot be both true at the same time, such as a literal and its negation
- This definition can be extended by domain knowledge about the properties of relations
- In the mobile robot domain, domain-specific constraints are:
  - A robot cannot be at two places at the same time
  - Two robots cannot be at the same place at a time
  - A place cannot be clear and occupied by a robot

inconsistent(c(P), at(R,P))
inconsistent(at(R,P1), at(R,P2)) if P1 \= P2.
inconsistent(at(R1,P), at(R2, P)) if R1 \= R2.

IMPLEMENTATIONS IN PROLOG

Russell&Norvig 2010 (Artificial Intelligence: A Modern Approach):
"... goal regression planning ... also used by Warren's ... WARPLAN. WARPLAN is also notable in that it was the first planner to be written in ... Prolog and was one of the best examples of the remarkable economy that can sometimes be gained by using logic programming: WARPLAN is only 100 lines of code, a small fraction of the size of comparable programs ..."

In book Prolog Programming for Artificial Intelligence, 4th edition (I. Bratko, Pearson Education 2012):

Regression planning: < 50 lines of code

Partial order planning: < 70 lines of code (using also CLP(FD))

GRAPHPLAN : < 100 lines (including CLP(FD) simulation of CLP(B))