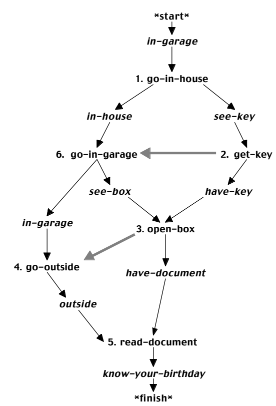
Examples of the OSCAR Planner in Operation

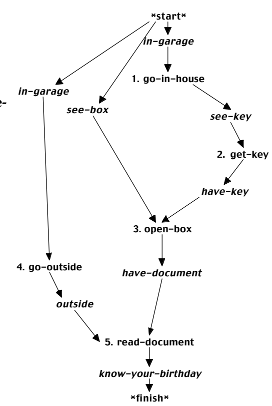
```
OSCAR_3.31        10/28/1999        12:25:57
              Non-Linear-Planner-43
Goal-state:
        know-your-birthdate
Given:
        in-garage
        see-box
        ( (in-garage & go-in-house) =>
                (see-key & (in-house & (~in-garage & (~outside & ~see-box)))))
        ( (in-garage & go-outside) =>
                (outside & (~in-house & (~in-garage & (~see-box & ~see-key)))))
        ( (see-key & get-key) => have-key)
        ( (in-house & go-to-garage) =>
                 (in-garage & (~in-house & (~outside & (see-box & ~see-key)))))
        ( ((have-key & see-box) & open-box) => have-document)
        ( ((have-document & outside) & read-document) => know-your-birthdate)

============================================================
Elapsed time = 0.36 sec
Cumulative size of arguments = 94
Size of inference-graph = 133 of which 0 were unused suppositions.
70% of the inference-graph was used in the argument.
198 interests were adopted.
58 interests were discharged by nodes used in the solution.
29% of the interests were used directly in finding the solution.
The branching factor = 1.09
149 interest-schemes were constructed.
68 instantiated-premises were constructed.
256 cycles of reasoning occurred.
39 plans were constructed.
```

```
Plan #39
  PLAN-STEPS:
    (1) go-in-house
        causal-links:
          0 --in-garage--> 1
    (2) get-key
        causal-links:
          1 --see-key--> 2
        ordering-constraints:
          2 > 1
    (6) go-to-garage
        causal-links:
          1 --in-house--> 6
        ordering-constraints:
          6 > 2
    (3) open-box
        causal-links:
          2 --have-key--> 3
          6 --see-box--> 3
        ordering-constraints:
          3 > 6
    (4) go-outside
        causal-links:
          6 --in-garage--> 4
        ordering-constraints:
          4 > 3
    (5) read-document
        causal-links:
          3 --have-document--> 5
          4 --outside--> 5
        ordering-constraints:
          5 > 4
    GOAL: know-your-birthdate
        established by:
          5 --> know-your-birthdate
```

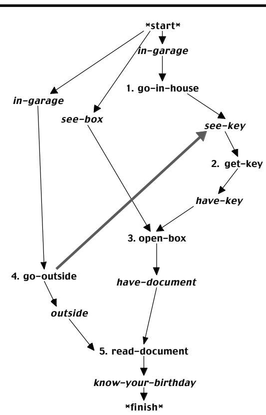
Plan 39
The solution



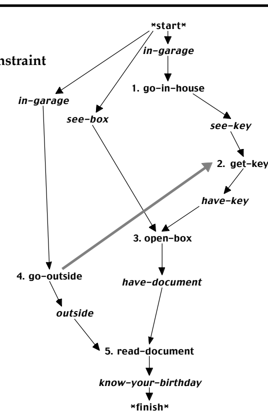
Plan 12
constructed by goal-regression, null-plan, and split-conjunctive-goal.

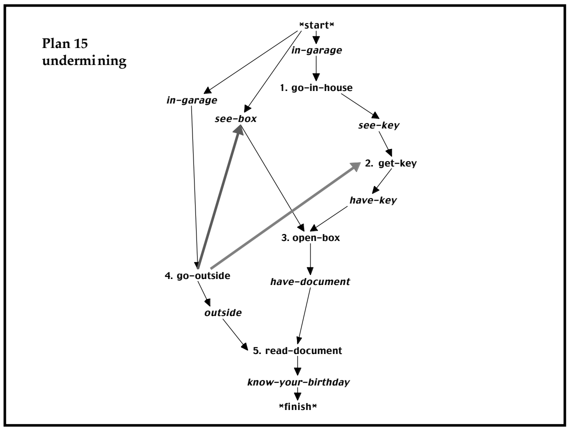


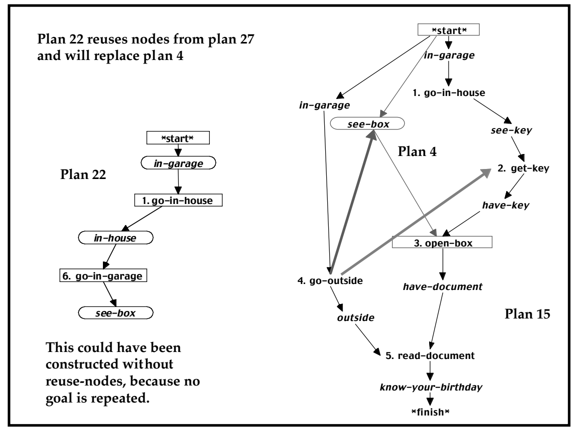
Plan 12
undermining

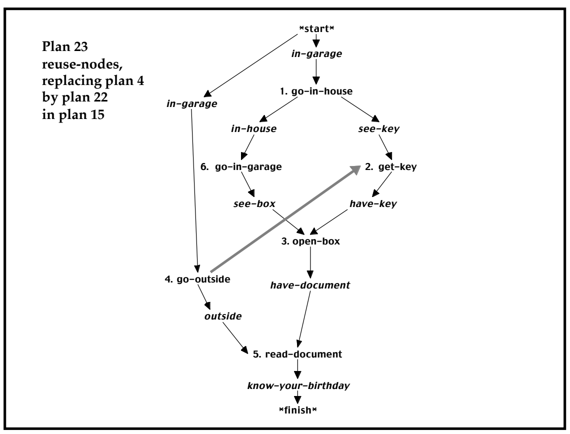


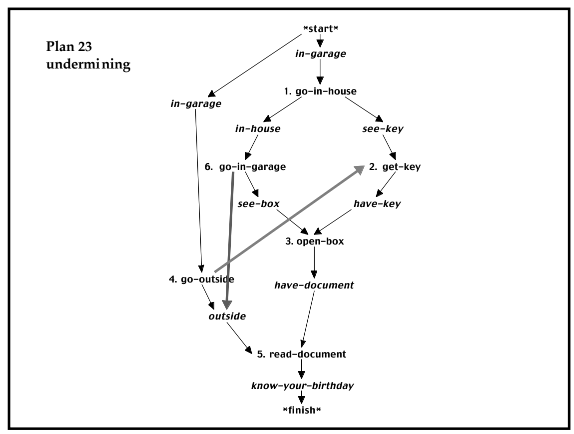
Plan 15
add-ordering-constraint to plan 12

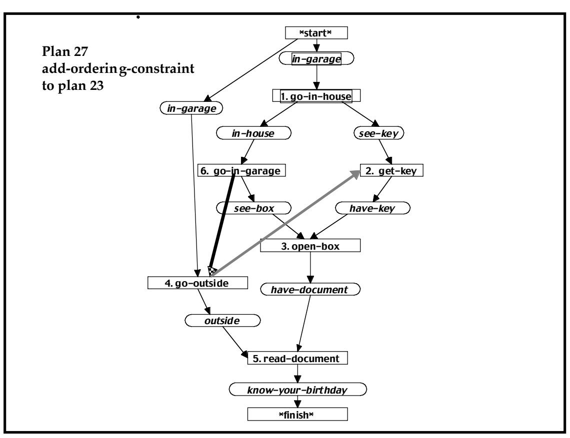


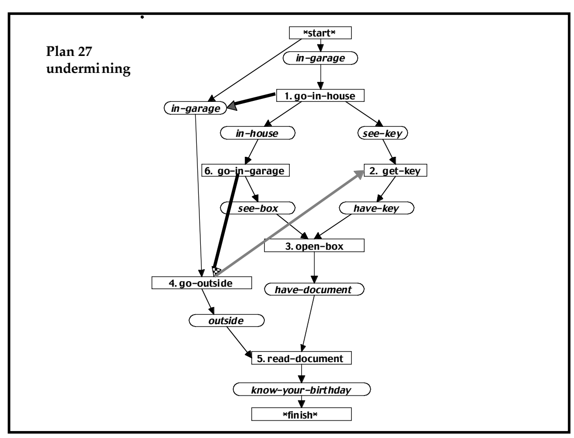

Plan 15
undermining
start
in-garage
1. go-in-house
in-garage
see-box
see-key
2. get-key
have-key
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

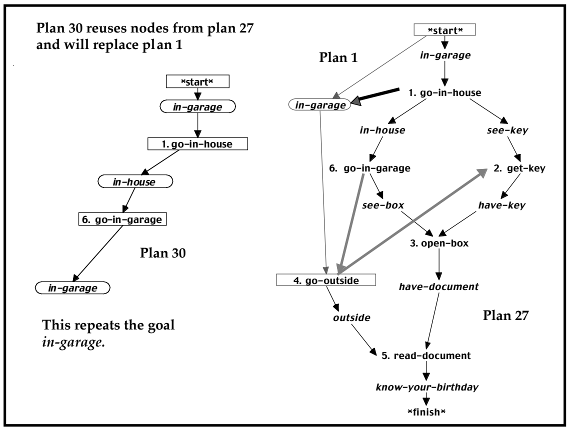

Plan 22 reuses nodes from plan 27
and will replace plan 4
Plan 22
start
in-garage
1. go-in-house
in-house
6. go-in-garage
see-box
This could have been
constructed without
reuse-nodes, because no
goal is repeated.
start
in-garage
1. go-in-house
in-garage
see-box
see-key
Plan 4
2. get-key
have-key
3. open-box
4. go-outside
have-document
outside
Plan 15
5. read-document
know-your-birthday
finish

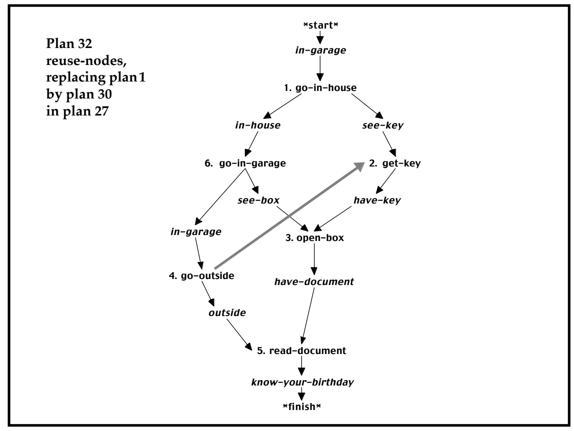

Plan 23
reuse-nodes,
replacing plan 4
by plan 22
in plan 15
start
in-garage
1. go-in-house
in-garage
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

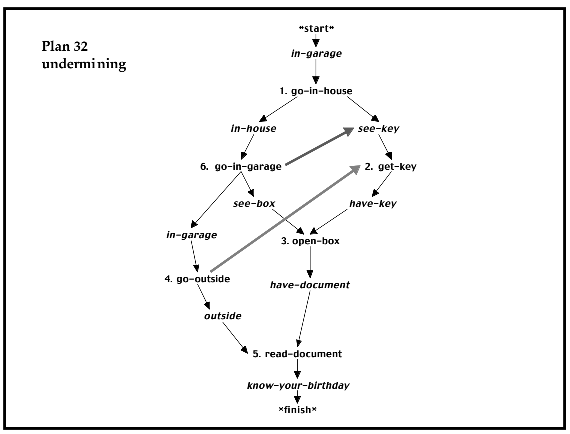

Plan 23
undermining
start
in-garage
1. go-in-house
in-garage
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

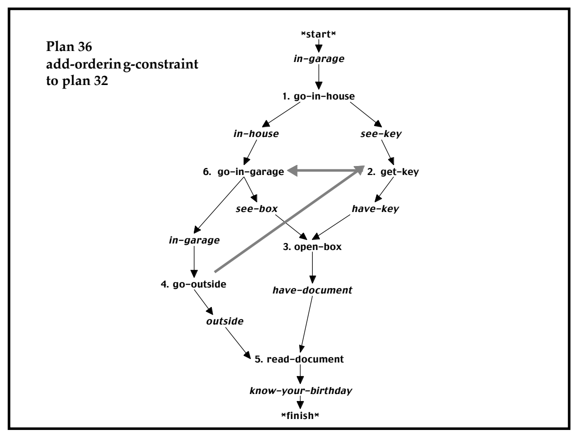

Plan 27
add-ordering-constraint
to plan 23
start
in-garage
1. go-in-house
in-garage
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

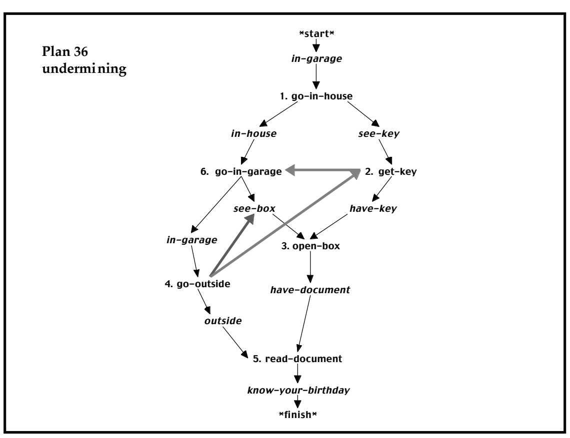

Plan 27
undermining
start
in-garage
1. go-in-house
in-garage
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

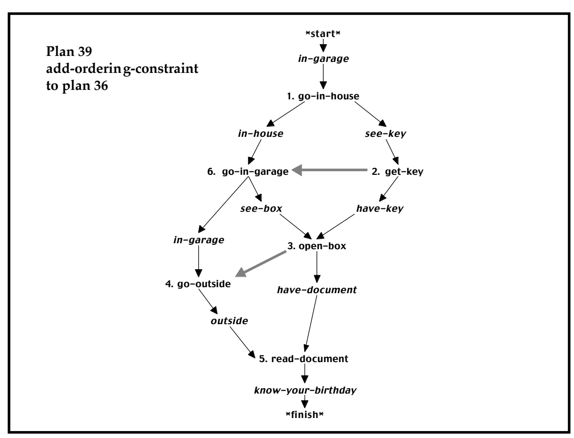

Plan 30 reuses nodes from plan 27
and will replace plan 1
start
in-garage
1. go-in-house
in-house
6. go-in-garage
in-garage
Plan 30
This repeats the goal
in-garage.
Plan 1
start
in-garage
1. go-in-house
in-garage
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
3. open-box
4. go-outside
have-document
outside
Plan 27
5. read-document
know-your-birthday
finish


Plan 32
reuse-nodes,
replacing plan 1
by plan 30
in plan 27
start
in-garage
1. go-in-house
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
in-garage
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish


Plan 32
undermining
start
in-garage
1. go-in-house
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
in-garage
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish


Plan 36
add-ordering-constraint
to plan 32
start
in-garage
1. go-in-house
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
in-garage
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish


Plan 36
undermining
start
in-garage
1. go-in-house
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
in-garage
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish


Plan 39
add-ordering-constraint
to plan 36
start
in-garage
1. go-in-house
in-house
see-key
6. go-in-garage
2. get-key
see-box
have-key
in-garage
3. open-box
4. go-outside
have-document
outside
5. read-document
know-your-birthday
finish

Stuart Russell's Flat Tire Problem

Given:

```
( :type wheel1 wheel)
( :type wheel2 wheel)
( :type hub isa-hub)
( :type nuts are-nuts)
( :type boot container)
( intact wheel2)
( in jack boot)
( in pump boot)
( in wheel2 boot)
( in wrench boot)
( on wheel1 hub)
( on-ground hub)
( tight nuts hub)
~( locked boot)
~( is-open boot)
~( inflated wheel2)
~( unfastened hub)
(all x :type container)( ((~( locked x) & ~( is-open x)) & ( open-up x)) => ( is-open x))
(all x :type container)( (( is-open x) & ( close x)) => ~( is-open x))
(all x)(all y :type container)( ((( in x y) & ( is-open y)) & ( fetch x y)) => (( have x) & ~( in x y)))
(all x)(all y :type container)( ((( have x) & ( is-open y)) & ( put-away x y)) => (~( have x) & ( in x y)))
(all x :type are-nuts)(all y :type isa-hub)( ((( have wrench) & (( tight x y) & ( on-ground y))) & ( loosen x y))
          => (( loose x y) & ~( tight x y)))
(all x :type are-nuts)(all y :type isa-hub)( ((( have wrench) & (( loose x y) & ( on-ground y))) & ( tighten x y))
          => (( tight x y) & ~( loose x y)))
(all x :type isa-hub)( ((( on-ground x) & ( have jack)) & ( jack-up x)) => (~( on-ground x) & ~( have jack)))
(all x :type isa-hub)( (~( on-ground x) & ( jack-down x)) => (( on-ground x) & ( have jack)))
(all x :type are-nuts)(all y :type isa-hub)( ((~( on-ground y) & (~( unfastened y) & (( have wrench) & ( loose x y)))) & ( undo x y))
          => (( have x) & (( unfastened y) & (~( on x y) & ~( loose x y)))))
(all x :type are-nuts)(all y :type isa-hub)( ((~( on-ground y) & (( unfastened y) & (( have wrench) & ( have x)))) & ( do-up x y))
          => (( loose x y) & (~( unfastened y) & ~( have x))))
(all x :type wheel)(all y :type isa-hub)( ((~( on-ground y) & (( on x y) & ( unfastened y))) & ( remove-wheel x y))
          => (( have x) & (( wheeless y) & ~( on x y))))
(all x :type wheel)(all y :type isa-hub)( ((( have x) & (( wheeless y) & (( unfastened y) & ~( on-ground y)))) & ( put-on-wheel x y))
          => (( on x y) & (~( have x) & ~( wheeless y))))
(all x :type wheel)( ((( have pump) & (~( inflated x) & ( intact x))) & ( inflate x)) => ( inflated x))
```

Goal-state:

```
~( is-open boot)
( in jack boot)
( in pump boot)
( in wheel1 boot)
( in wrench boot)
( tight nuts hub)
( inflated wheel2)
( on wheel2 hub)
```

Stuart Russell's Flat Tire Problem

```
Plan #200
AN-STEPS:
1) ( open-up boot)
  causal-links:
   0 --~( is-open boot)--> 1
   0 --~( locked boot)--> 1
2) ( fetch jack boot)
  causal-links:
   0 --( in jack boot)--> 2
   1 --( is-open boot)--> 2
  ordering-constraints:
   2 > 1
4) ( fetch wrench boot)
  causal-links:
   1 --( is-open boot)--> 4
   0 --( in wrench boot)--> 4
  ordering-constraints:
   4 > 1
5) ( loosen nuts hub)
  causal-links:
   0 --( on-ground hub)--> 5
   4 --( have wrench)--> 5
   0 --( tight nuts hub)--> 5
  ordering-constraints:
   5 > 4
3) ( jack-up hub)
  causal-links:
   0 --( on-ground hub)--> 3
   2 --( have jack)--> 3
  ordering-constraints:
   3 > 2
   3 > 5
6) ( undo nuts hub)
  causal-links:
   3 --~( on-ground hub)--> 6
   0 --~( unfastened hub)--> 6
   4 --( have wrench)--> 6
   5 --( loose nuts hub)--> 6
  ordering-constraints:
   6 > 3
7) ( remove-wheel wheel1 hub)
  causal-links:
   3 --~( on-ground hub)--> 7
   0 --( on wheel1 hub)--> 7
   6 --( unfastened hub)--> 7
  ordering-constraints:
   7 > 6
8) ( put-away wheel1 boot)
  causal-links:
   1 --( is-open boot)--> 8
   7 --( have wheel1)--> 8
  ordering-constraints:
   8 > 7
9) ( fetch pump boot)
  causal-links:
   0 --( in pump boot)--> 9
   1 --( is-open boot)--> 9
  ordering-constraints:
   9 > 1
10) ( inflate wheel2)
  causal-links:
   9 --( have pump)--> 10
   0 --~( inflated wheel2)--> 10
   0 --( intact wheel2)--> 10
  ordering-constraints:
   10 > 9
11) ( fetch wheel2 boot)
  causal-links:
   1 --( is-open boot)--> 11
   0 --( in wheel2 boot)--> 11
  ordering-constraints:
   11 > 1
12) ( put-on-wheel wheel2 hub)
  causal-links:
   3 --~( on-ground hub)--> 12
   6 --( unfastened hub)--> 12
   11 --( have wheel2)--> 12
   7 --( wheeless hub)--> 12
  ordering-constraints:
   12 > 7
   12 > 11
19) ( do-up nuts hub)
  causal-links:
   3 --~( on-ground hub)--> 19
   4 --( have wrench)--> 19
   6 --( unfastened hub)--> 19
   6 --( have nuts)--> 19
  ordering-constraints:
   19 > 12
18) ( jack-down hub)
  causal-links:
   3 --~( on-ground hub)--> 18
  ordering-constraints:
   18 > 19
14) ( put-away jack boot)
  causal-links:
   1 --( is-open boot)--> 14
   18 --( have jack)--> 14
  ordering-constraints:
   14 > 18
15) ( put-away pump boot)
  causal-links:
   1 --( is-open boot)--> 15
   9 --( have pump)--> 15
  ordering-constraints:
   15 > 10
17) ( tighten nuts hub)
  causal-links:
   4 --( have wrench)--> 17
   19 --( loose nuts hub)--> 17
   18 --( on-ground hub)--> 17
  ordering-constraints:
   17 > 18
16) ( put-away wrench boot)
  causal-links:
   1 --( is-open boot)--> 16
   4 --( have wrench)--> 16
  ordering-constraints:
   16 > 17
13) ( close boot)
  causal-links:
   1 --( is-open boot)--> 13
  ordering-constraints:
   13 > 8
   13 > 11
   13 > 14
   13 > 15
   13 > 16
GOAL:
(~( is-open boot) &
(( in jack boot) &
(( in pump boot) &
(( in wheel1 boot) &
(( in wrench boot) &
(( tight nuts hub) &
(( inflated wheel2) &
( on wheel2 hub))))))))
  established by:
   8 --> ( in wheel1 boot)
   10 --> ( inflated wheel2)
   12 --> ( on wheel2 hub)
   13 --> ~( is-open boot)
   14 --> ( in jack boot)
   15 --> ( in pump boot)
   16 --> ( in wrench boot)
   17 --> ( tight nuts hub)
```

OSCAR's Performance on Stuart Russell's Flat Tire Problem

Elapsed time = 7.39 sec
Cumulative size of arguments = 396
Size of inference-graph = 565 of which 0 were unused suppositions.
70% of the inference-graph was used in the argument.
961 interests were adopted.
288 interests were discharged by nodes used in the solution.
29% of the interests were used directly in finding the solution.
The branching factor = 1.02
679 interest-schemes were constructed.
257 instantiated-premises were constructed.
1112 cycles of reasoning occurred.
200 plans were constructed.

Planning and Searching

- Planning is generally characterized as search.
- Early planners searched the state-space, but that was immense and they could not solve hard problems.
- For a simplified version of the flat tire problem:

```
(number-of-plans *start-state* *operators*
 '((tight nuts hub) (on wheel2 hub)) 12)
```

There are 1,367,478,242 plans of length 12
for a branching factor of 5.77
There are 273 plans of length 12 establishing
((tight nuts hub) (on wheel2 hub))
The effective branching factor is 3.61

Planning and Searching

- Modern planners have been described as "searching the plan-space".
- This consists of the space of (partial) plans produced in the course of searching for the solution.
- But this space is entirely dependent on the planning algorithm, and is not characteristic of the problem itself.
- In particular, the branching factor does not tell us how hard the problem is — just how hard it is *for this planner.*
- Note that OSCAR's branching factor for the flat-tire problem is just 1.02. This hardly qualifies as search.

Planning and Searching

- My conjecture is that humans can only solve problems with very small branching factors, relative to their planning algorithm.
- OSCAR constructs plans much like human beings do.
- Most automated planners take search seriously, but this makes the problems harder than they need be.
- OSCAR is able to solve hard problems very efficiently (but also very slowly compared to other planners).