Conformant Planners: Approximations vs. Representation

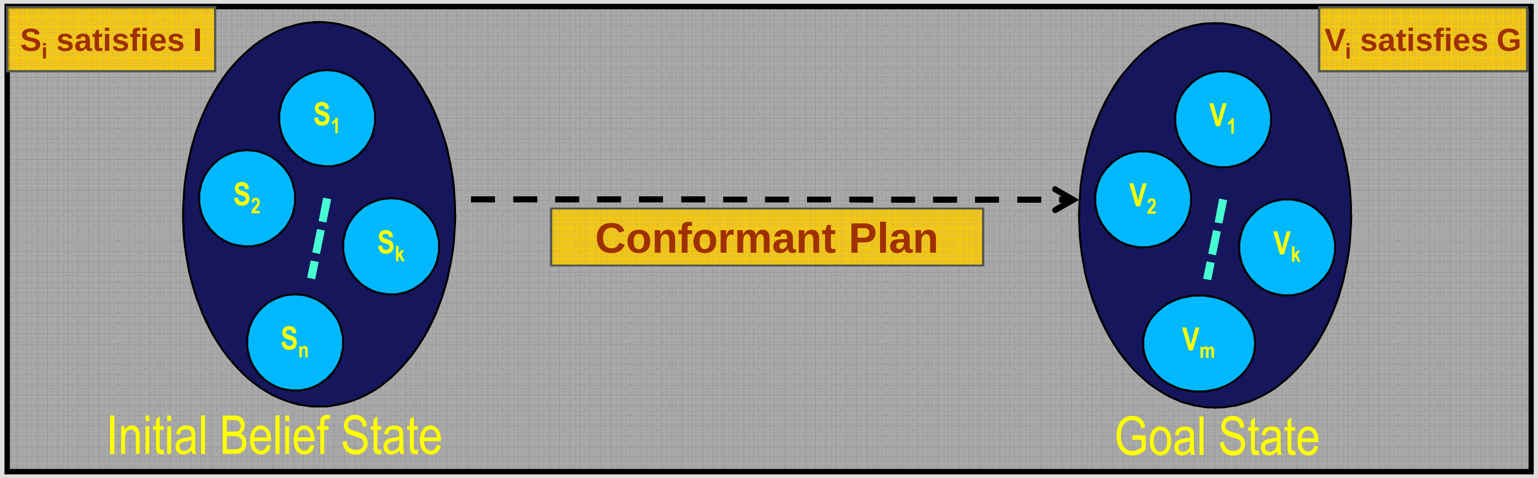
Son Thanh To, Vien Dang Tran, Khoi Hoang Nguyen,
Tran Cao Son, Enrico Pontelli

Computer Science Department
New Mexico State University, Las Cruces, NM 88003

POSTER TEMPLATE BY:
www.PosterPresentations.com

Conformant Planning Problem

- Given: planning problem $P = \langle F, O, I, G \rangle$ where
  - F is a set of propositions
  - O is a set of operators
  - I is the initial state – often incomplete
  - G is the goal
- Problem: Computing a plan that achieves G from all possible initial states of the world satisfying I



www.PosterPresentations.com

Goal, Motivated Questions, and Facts

- **Goal**: develop state-of-the-art conformant planners
- **Motivated questions**:
  - How does the definition of a progression function influence the performance of a conformant planner?
  - How does the representation of belief states influence the performance of a conformant planner?
- **Motivated facts**:
  - CpA^{PH}, an *approximation-based conformant planner*, uses an incomplete progression function & a compact belief state representation performs very well in its first implementation
  - CpA^{PH} differs from all of its counterparts when it was introduced
  - CpA^{PH} needs complete initial belief state in benchmark problems with disjunctive information about the initial state

POSTER TEMPLATE BY:
www.PosterPresentations.com

Considerations in Conformant Planners

- How to encode a belief state? Many possibilities
  - OBDD
  - DNF
  - CNF
  - …

  each might have its own desirable properties (e.g. minimal)
- How to progress? By a function Φ
  - Given an action *a* and a belief state *S* in the corresponding representation, compute the belief state *U* resulting from executing *a* in *S*, written as $U = \Phi(a, S)$
  - Certain operations on a representation might lead to a formula which no longer satisfies the desirable properties and require some overhead after the computation (e.g., updating minimal CNF might not result in a minimal CNF)

POSTER TEMPLATE BY: www.PosterPresentations.com

Main Characteristics of CpA

- Approximation-based progression function
- Encoding of belief state enable easy computation of successor belief state
- Search for plan in the space of 3^n partial states instead of the space of 2^{2^n} belief states as most other conformant planners (for problems with conjunction of literals as initial state)
- Maintain completeness through special reasoning technique
  - CpA incurs significant overhead in the computation of the representation of the initial belief state
  - CpA uses DNF-formulae to encode belief states and can potentially require a lot of memory
- CpA uses a combination of the cardinality and the number of satisfied subgoals heuristic as its heuristic function

POSTER TEMPLATE BY:
www.PosterPresentations.com

Main Characteristics of DNF

- ❑ A middle-ground between approximation and complete reasoning
- ❑ Search for plan in the space of 2^{2^n} belief states
- ❑ Use minimal DNF-formulae to represent belief states, also enable easy computation of successor belief state
- ❑ Progression function defined over minimal DNF-formulae
  - ❑ DNF incurs overhead for the transformation of successor belief state into minimal DNF-formulae
- ❑ DNF uses a combination of the cardinality, the number of satisfied subgoals, and the square distance to the goal heuristic as its heuristic function

POSTER TEMPLATE BY:
www.PosterPresentations.com

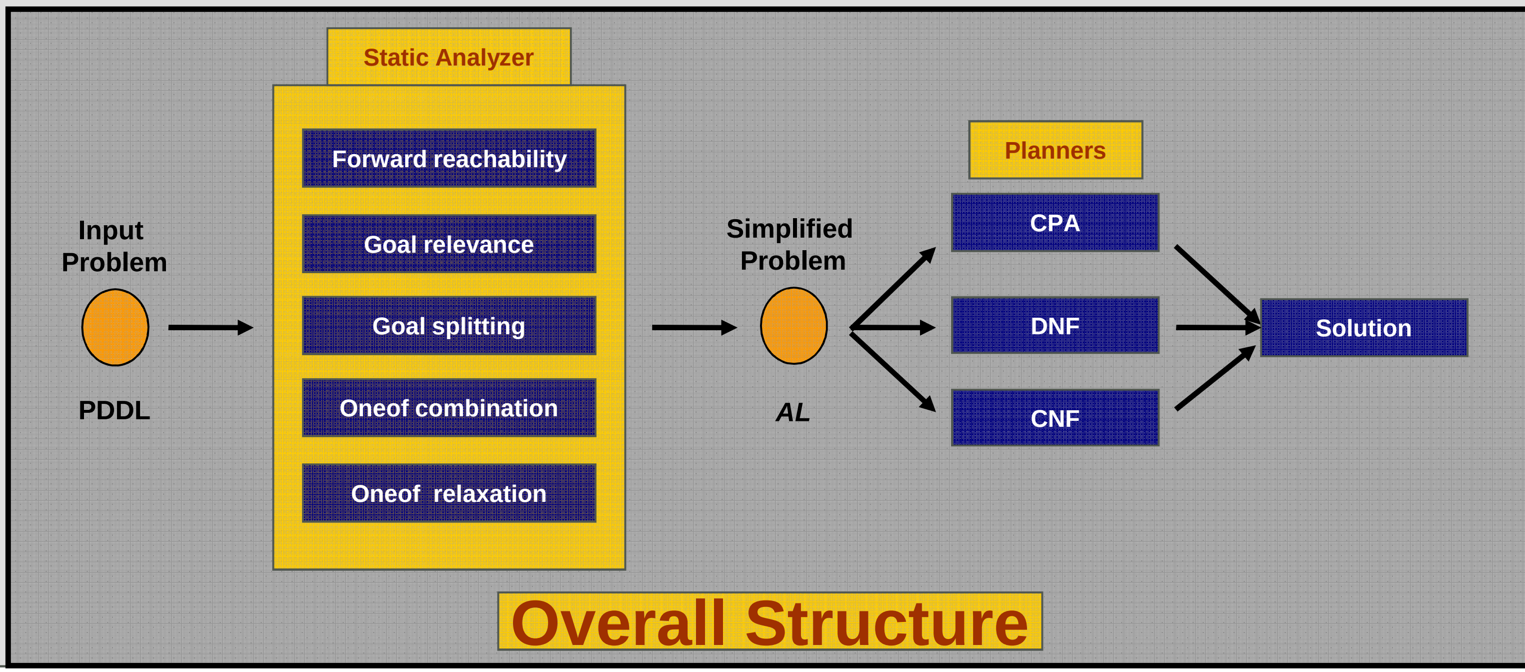
Main Characteristics of CNF

- Search for plan in the space of 2^{2^n} belief states
- Use minimal CNF-formulae to represent belief states, a departure of easy computation of successor belief state
- Progression function defined over minimal CNF-formulae
  - CNF also incurs overhead for the transformation of successor belief state into minimal CNF-formulae
- CNF uses the number of satisfied subgoals as its heuristic function

POSTER TEMPLATE BY:
www.PosterPresentations.com

Simplification Techniques for Scalability and Performance

- Forward reachability: eliminating redundant actions and propositions
- Goal relevance: identifying necessary information in the initial belief state to guarantee completeness
- Goal splitting: divide-and-conquer using subgoals
- Oneof-combination: reducing the size of the initial belief state
- Oneof-relaxation: replacing mutual exclusive or by disjunctive or

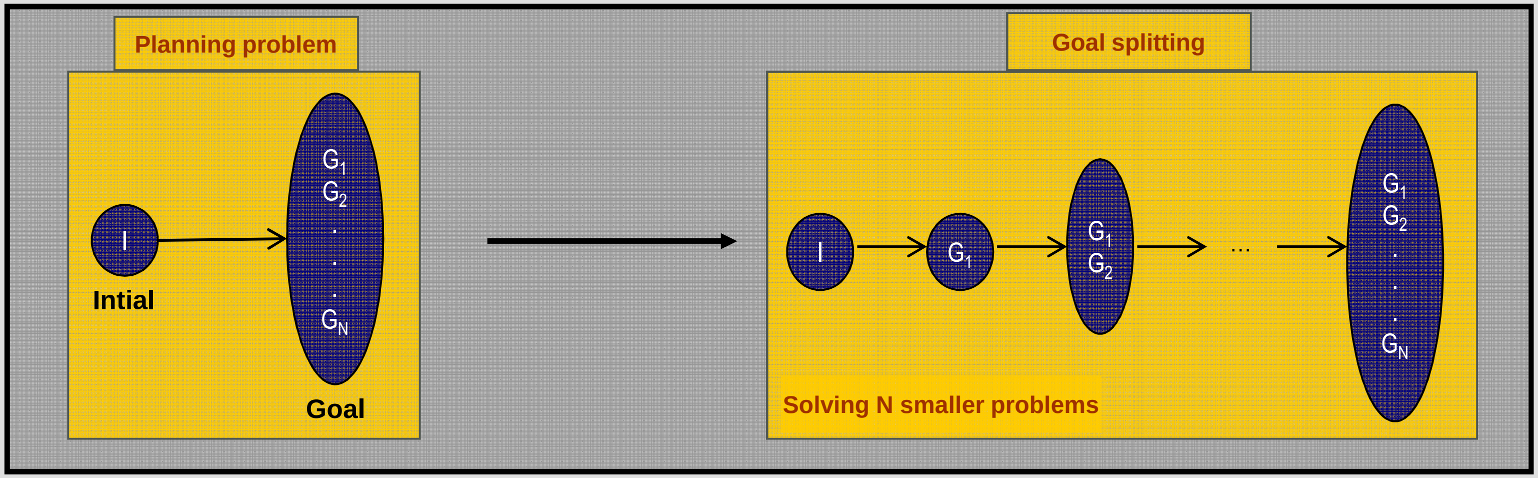


Overall Structure

www.PosterPresentations.com

Simplification Techniques: Goal Splitting

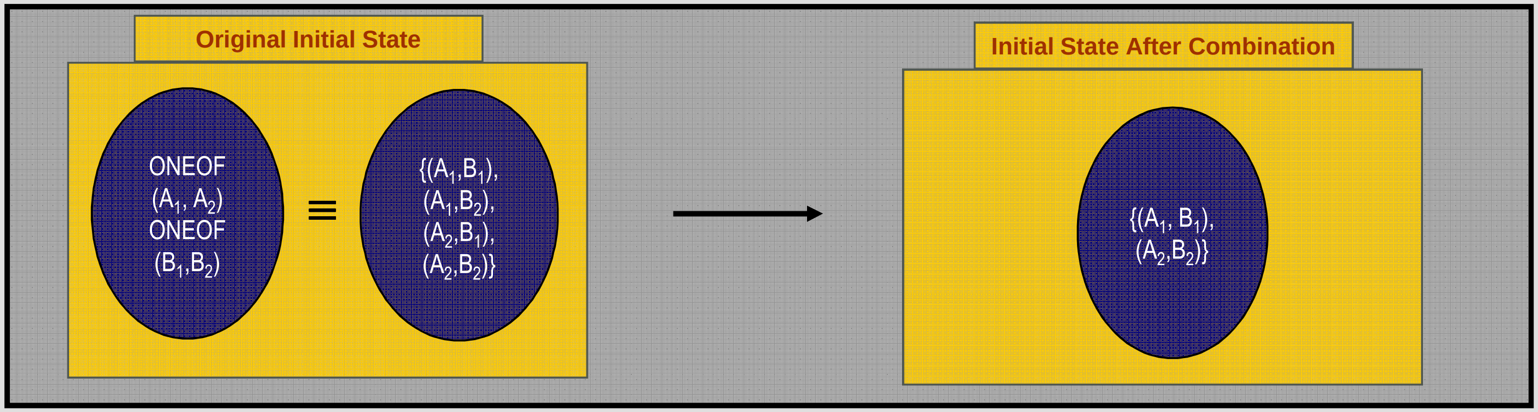
- If a problem *P* contains a subgoal whose truth value cannot be negated by the actions used to reach the other goals, then the problem can be decomposed into a sequence of smaller problems
- Improve scalability



www.PosterPresentations.com

Simplification Techniques: oneof-combination

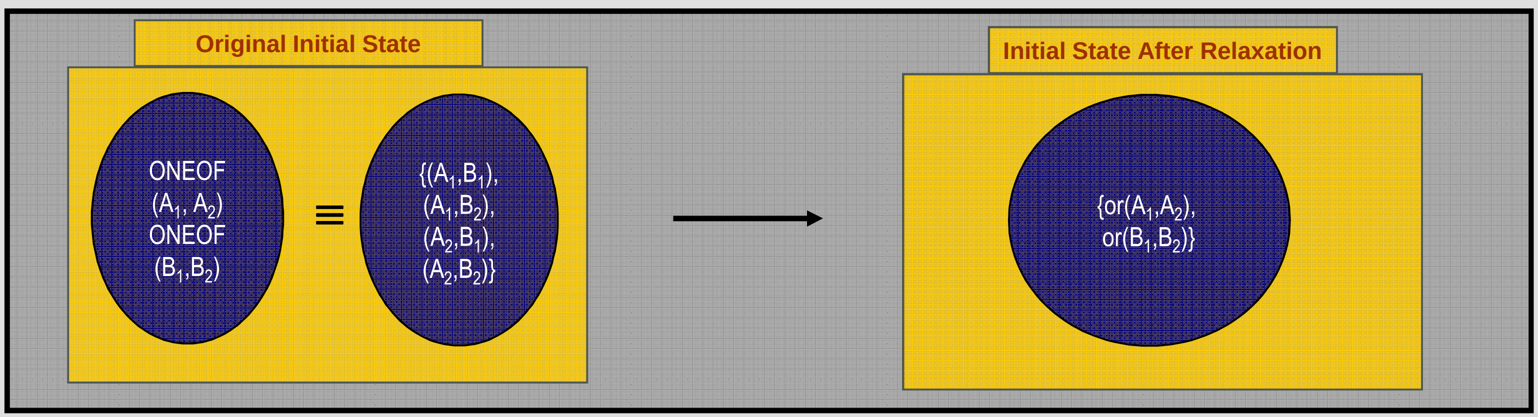
- If actions and propositions in different **oneof**'s have no interaction then we do not need to consider all possible permutations of the **oneof**'s.
- Reducing the size of the initial belief state
- Improve scalability
- Suitable for DNF and CpA



www.PosterPresentations.com

Simplification Techniques: `oneof`-relaxation

- If actions and propositions in an **`oneof`**–clause satisfy certain properties then an **`oneof`**–clause can be replaced by an **`or`**–clause
- Increasing the size of the initial belief state
- Improve scalability
- Suitable for CNF



www.PosterPresentations.com

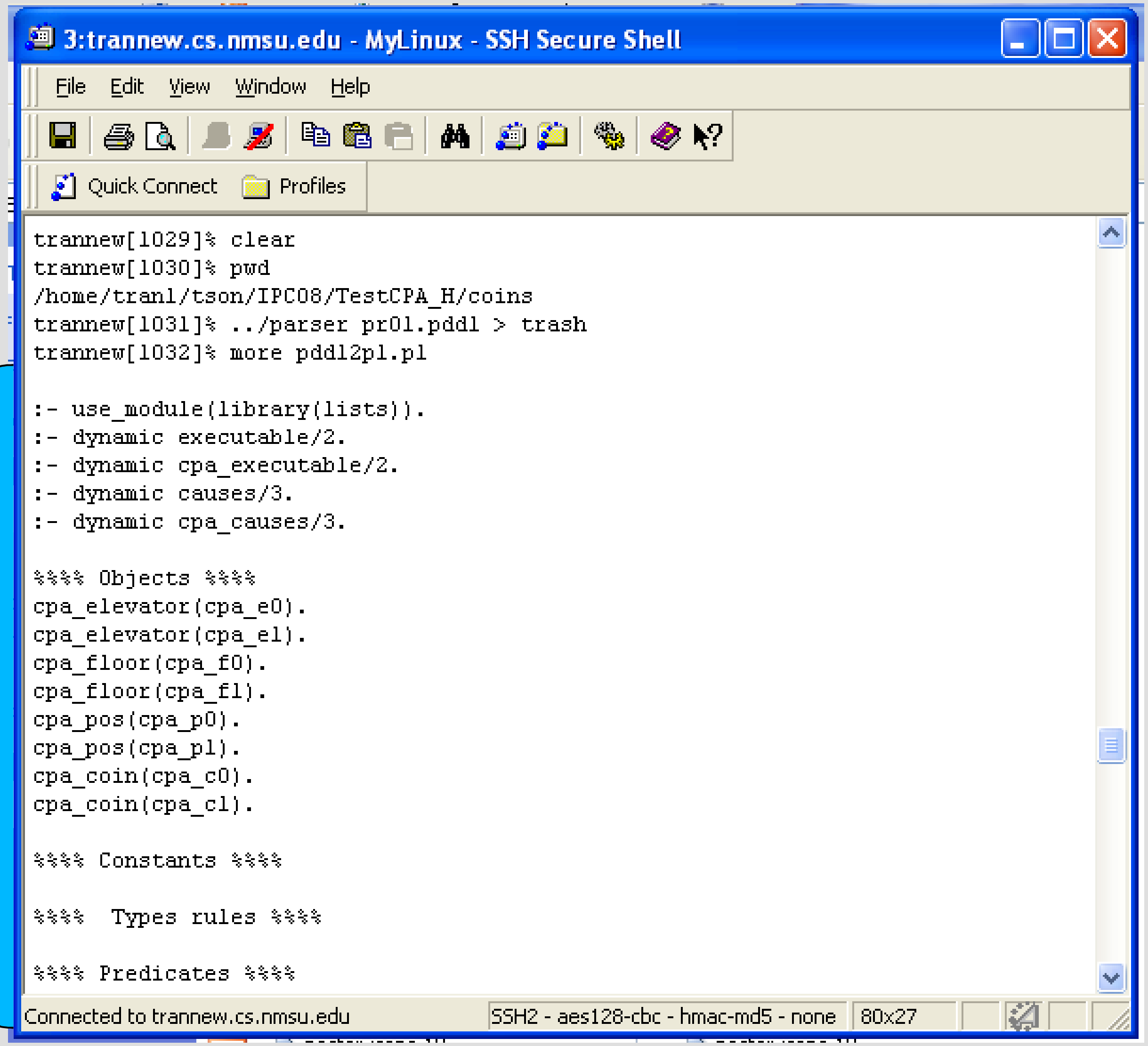
Conclusions

- Presentation of three conformant planners: CpA, DNF, and CNF
- There exists no “one size fits all” representation for all domains
- The choice of belief state representation impacts
  - performance of conformant planner
  - choice of simplification techniques
  - algorithm for computing successor belief state

POSTER TEMPLATE BY:
www.PosterPresentations.com

A Sample Run – CpA - Preprocessor

Translating from PDDL to Prolog



Prolog Representation of PDDL (segment)

Preprocessor

Calling the Preprocessor

3:trannew.cs.nmsu.edu - MyLinux - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
trannew[1036]% sicstus -l prob.pl --goal 'main,halt.' > trash
% compiling /home/tranl/tson/IPC08/TestCPA_H/coins/prob.pl...
%  loading /local/sicstus-3.12.2/lib/sicstus-3.12.2/library/lists.po...
%  module lists imported into user
%  loaded /local/sicstus-3.12.2/lib/sicstus-3.12.2/library/lists.po in module li
sts, 0 msec 13696 bytes
%  loading /local/sicstus-3.12.2/lib/sicstus-3.12.2/library/ordsets.po...
%  module ordsets imported into user
%  loaded /local/sicstus-3.12.2/lib/sicstus-3.12.2/library/ordsets.po in module
ordsets, 0 msec 13912 bytes
NAME CLASH: remove_duplicates/2 is already imported into module user from module
 lists
Do you really want to override this definition with the one in user? (y, n, p, s
, a, b, or ?) y
* [NewFluents] - singleton variables in user:create_independent_theories/3
* Approximate lines: 581-589, file: '/home/tranl/tson/IPC08/TestCPA_H/coins/prob
.pl'
* [NewFluents] - singleton variables in user:create_dependent_theories/3
* Approximate lines: 590-598, file: '/home/tranl/tson/IPC08/TestCPA_H/coins/prob
.pl'
* [NI] - singleton variables in user:compose_in/4
* Approximate lines: 946-953, file: '/home/tranl/tson/IPC08/TestCPA_H/coins/prob
.pl'
NAME CLASH: remove_duplicates/2 is already defined in module user
Do you really want to override this definition with the one in lists? (y, n, p,
s, a, b, or ?) y
% compiled /home/tranl/tson/IPC08/TestCPA_H/coins/prob.pl in module user, 200 ms
```

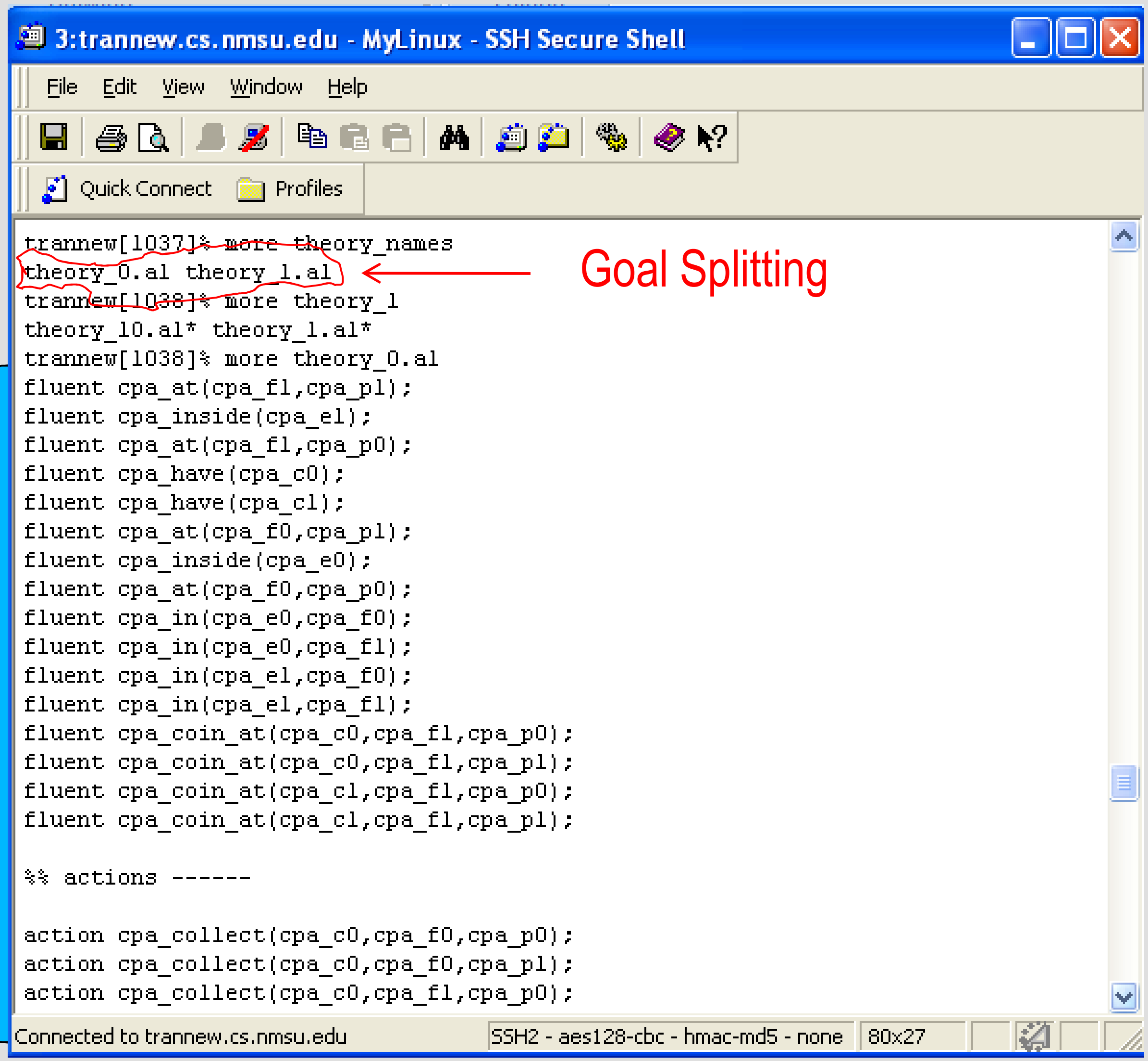
Connected to trannew.cs.nmsu.edu | SSH2 - aes128-cbc - hmac-md5 - none | 80x27

Output of Preprocessor

Output of the Preprocessor

Goal Splitting

First theory in AL




```
trannew[1037]% more theory_names
theory_0.al theory_1.al
trannew[1038]% more theory_1
theory_10.al* theory_1.al*
trannew[1038]% more theory_0.al
fluent cpa_at(cpa_f1,cpa_p1);
fluent cpa_inside(cpa_e1);
fluent cpa_at(cpa_f1,cpa_p0);
fluent cpa_have(cpa_c0);
fluent cpa_have(cpa_c1);
fluent cpa_at(cpa_f0,cpa_p1);
fluent cpa_inside(cpa_e0);
fluent cpa_at(cpa_f0,cpa_p0);
fluent cpa_in(cpa_e0,cpa_f0);
fluent cpa_in(cpa_e0,cpa_f1);
fluent cpa_in(cpa_e1,cpa_f0);
fluent cpa_in(cpa_e1,cpa_f1);
fluent cpa_coin_at(cpa_c0,cpa_f1,cpa_p0);
fluent cpa_coin_at(cpa_c0,cpa_f1,cpa_p1);
fluent cpa_coin_at(cpa_c1,cpa_f1,cpa_p0);
fluent cpa_coin_at(cpa_c1,cpa_f1,cpa_p1);

%% actions ------

action cpa_collect(cpa_c0,cpa_f0,cpa_p0);
action cpa_collect(cpa_c0,cpa_f0,cpa_p1);
action cpa_collect(cpa_c0,cpa_f1,cpa_p0);
```

Calling the planner

3:trannew.cs.nmsu.edu - MyLinux - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
%% goal state ----------
goal  cpa_have(cpa_c0);
trannew[1039]% ../cpa
cpa*          cpa+bfs+card* cpa+bfs+rgp*  cpa.pddl2pl*
cpa+*         cpa+bfs+gc*   cpa+dfs*
trannew[1039]% ../cpa+bfs+gc theory_names
0
%%

11 cpa_go_down(cpa_e0,cpa_f1,cpa_f0) cpa_go_up(cpa_e1,cpa_f0,cpa_f1) cpa_step_in
(cpa_e0,cpa_f0,cpa_p0) cpa_go_up(cpa_e0,cpa_f0,cpa_f1) cpa_step_out(cpa_e0,cpa_f
1,cpa_p0) cpa_collect(cpa_c0,cpa_f1,cpa_p0) cpa_move_right(cpa_f1,cpa_p0,cpa_p1)
 cpa_collect(cpa_c0,cpa_f1,cpa_p1) cpa_collect(cpa_c1,cpa_f1,cpa_p1) cpa_move_le
ft(cpa_f1,cpa_p1,cpa_p0) cpa_collect(cpa_c1,cpa_f1,cpa_p0)
%%
linear 12 0 1 2 3 4 5 0 6 7 8 9 10
STATISTICS
---------------------
Total time: 0.011 (sec)
  Reading: 0.002 (sec) [17.37 %]
  Preprocessing: 0.001 (sec) [9.63 %]
  Search: 0.008 (sec) [73.01 %]
Total states allocated: 0
Total cstate(s): 0
Total cstate(s) remaining in the queue: 0
trannew[1040]%
```

Connected to trannew.cs.nmsu.edu | SSH2 - aes128-cbc - hmac-md5 - none | 80x27

Plan

Statistic