"Add Another Blue Stack of the Same Height!": ASP Based Planning and Planning Failure Analysis

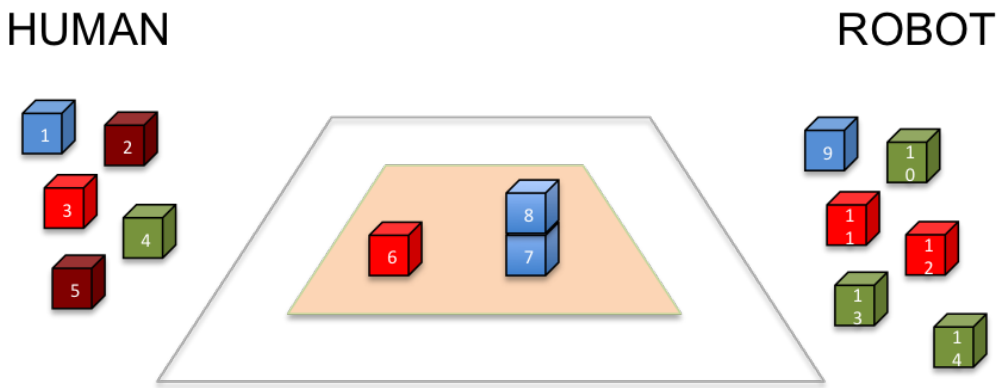
Chitta Baral[1] Tran Cao Son[2]

[1]Computer Science and Engineering, Arizona State University, Tempe, AZ
[2]Computer Science, New Mexico State University, Las Cruces, NM

LPNMR 2015

Outline

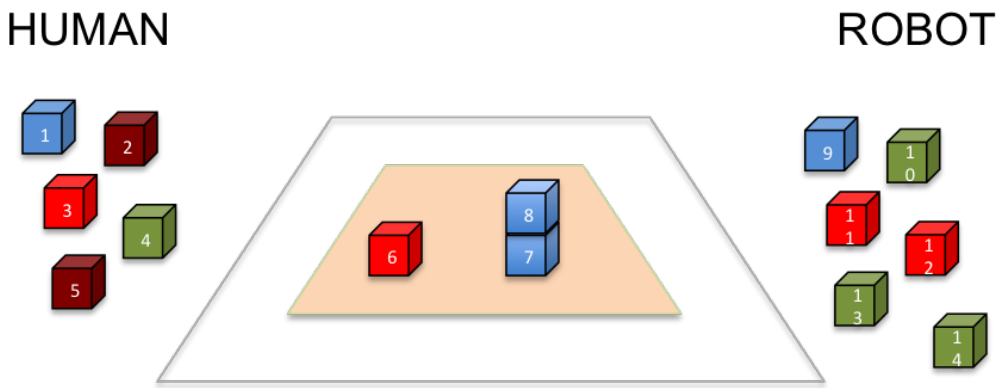
Human-Robot Interaction



Human

Robot

Human-Robot Interaction

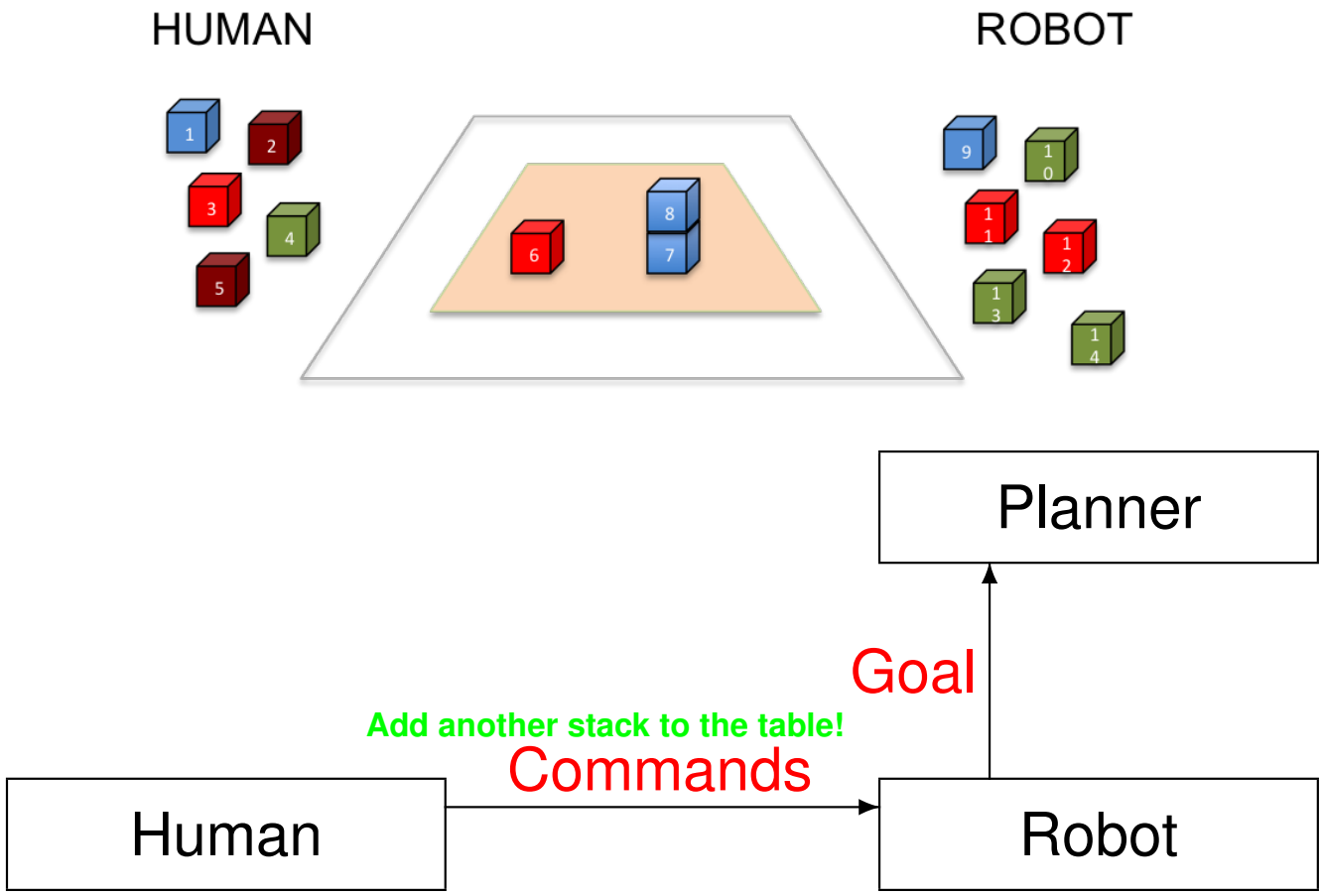


Add another stack to the table!

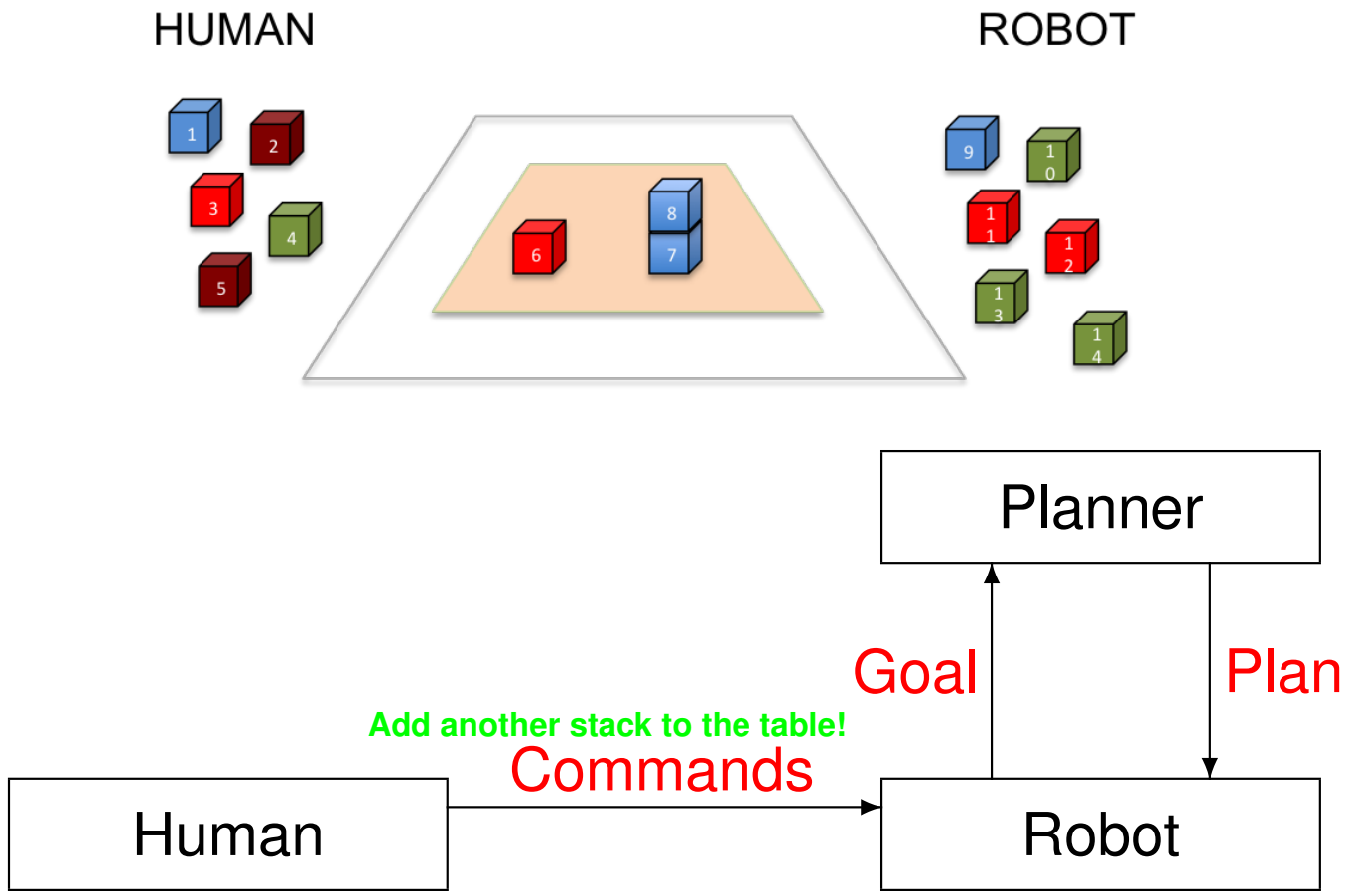
Commands

Human → Robot

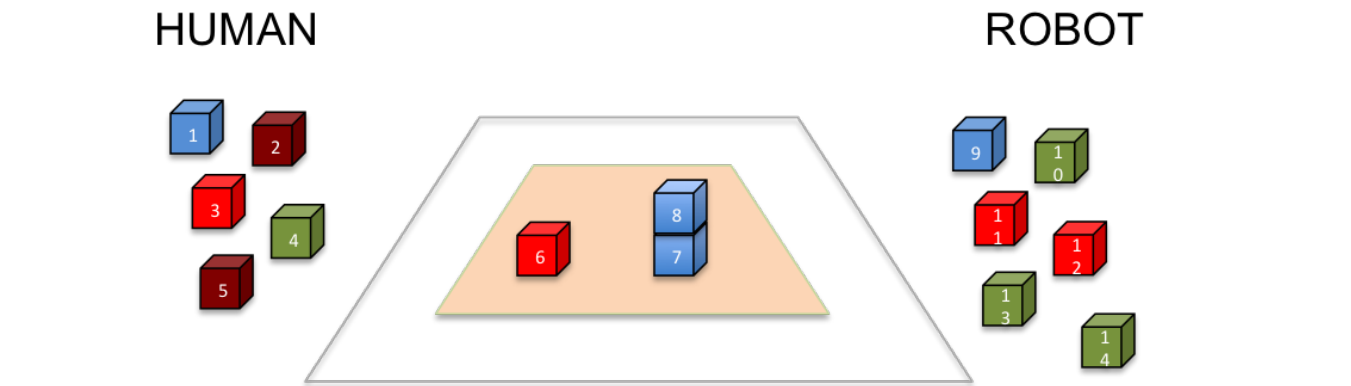
Human-Robot Interaction

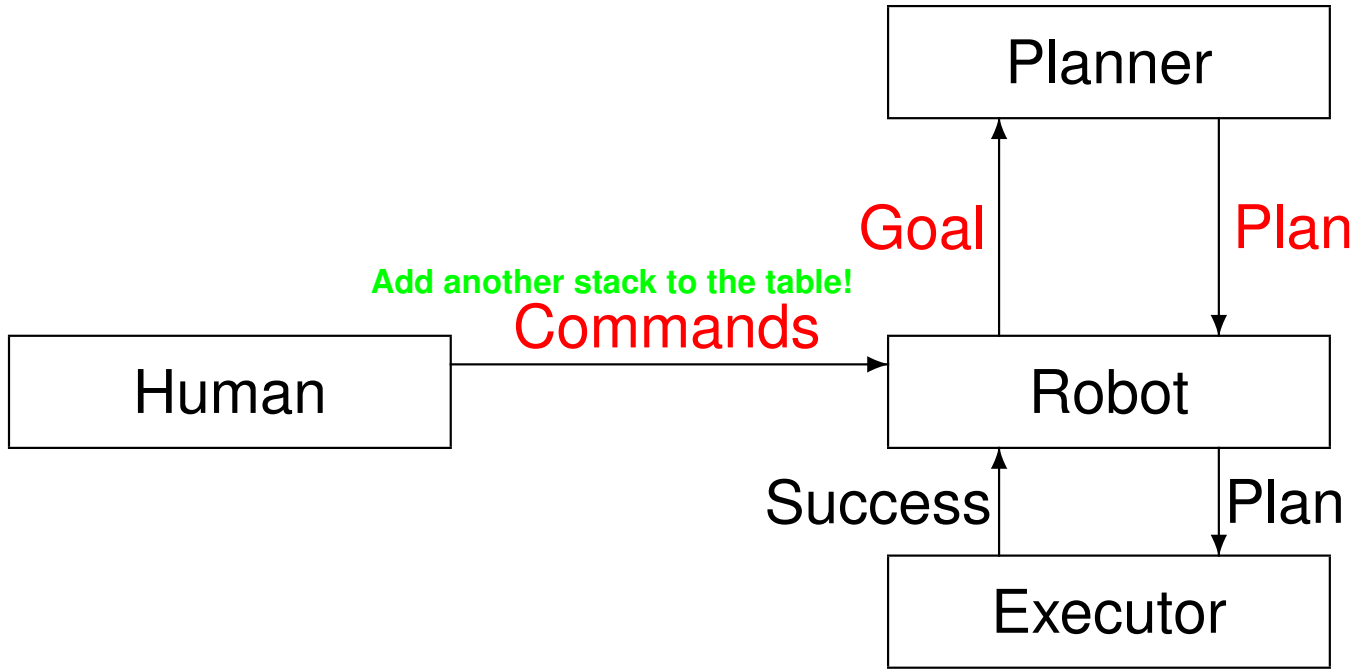


Human-Robot Interaction

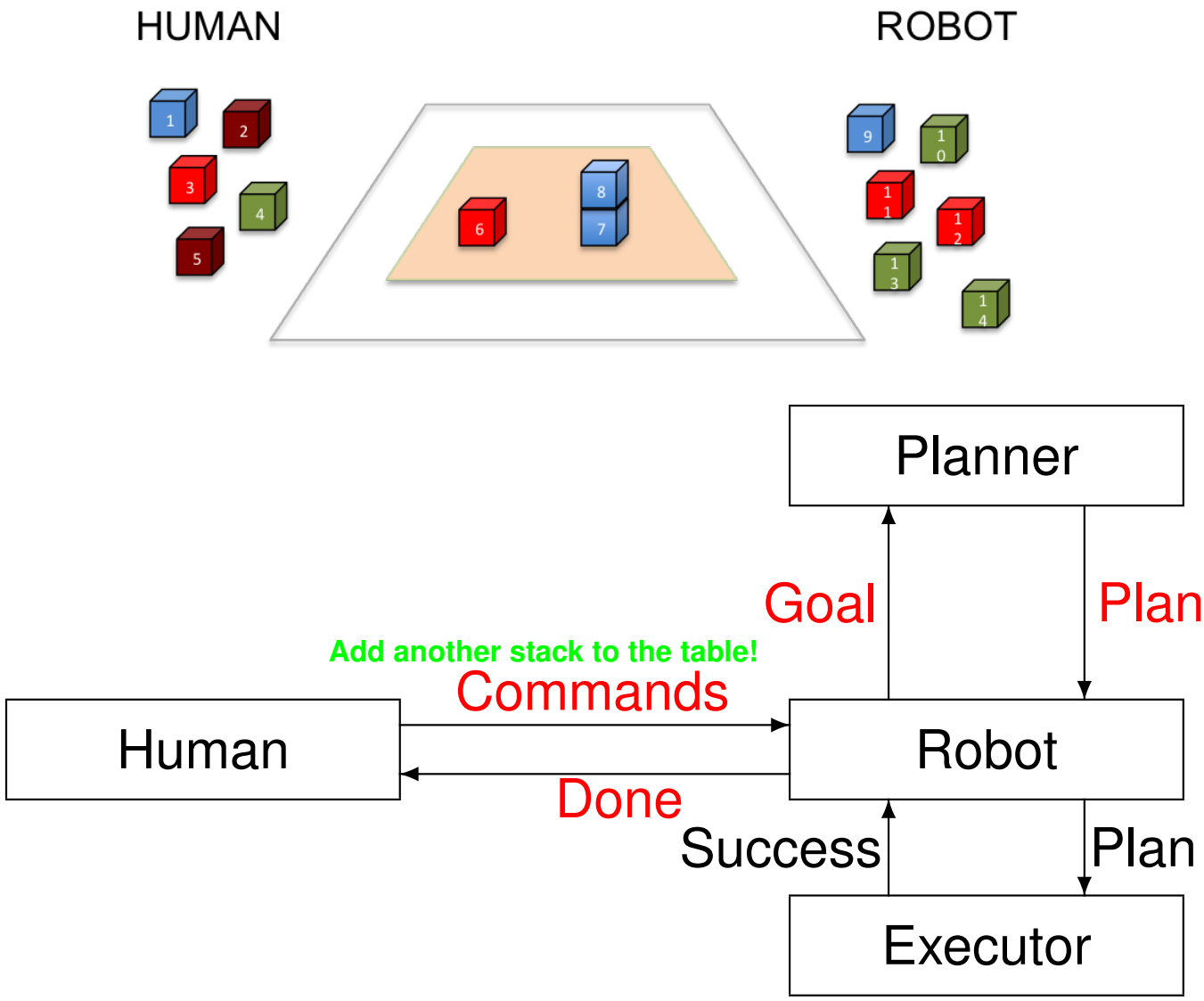


Human-Robot Interaction

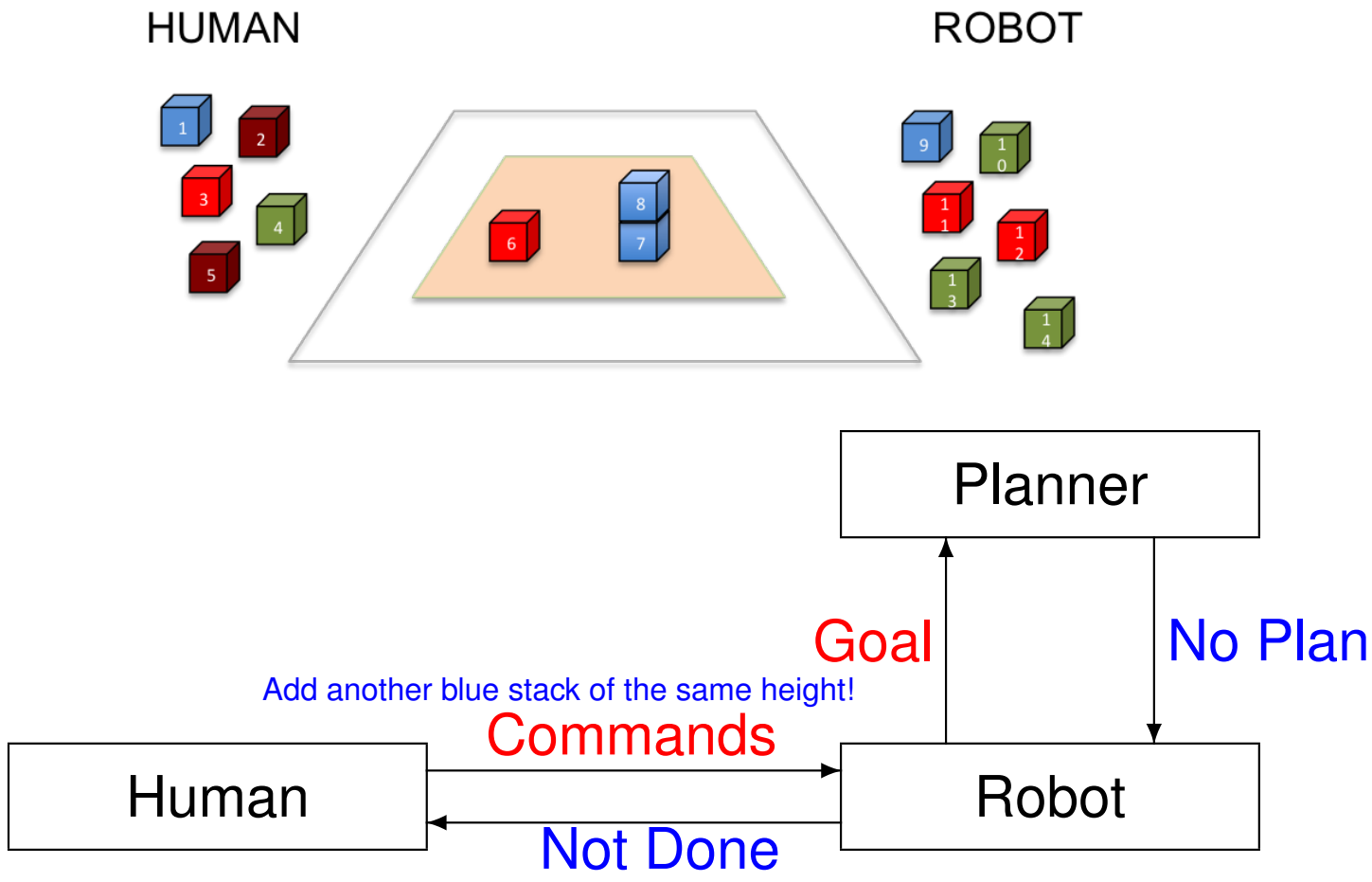




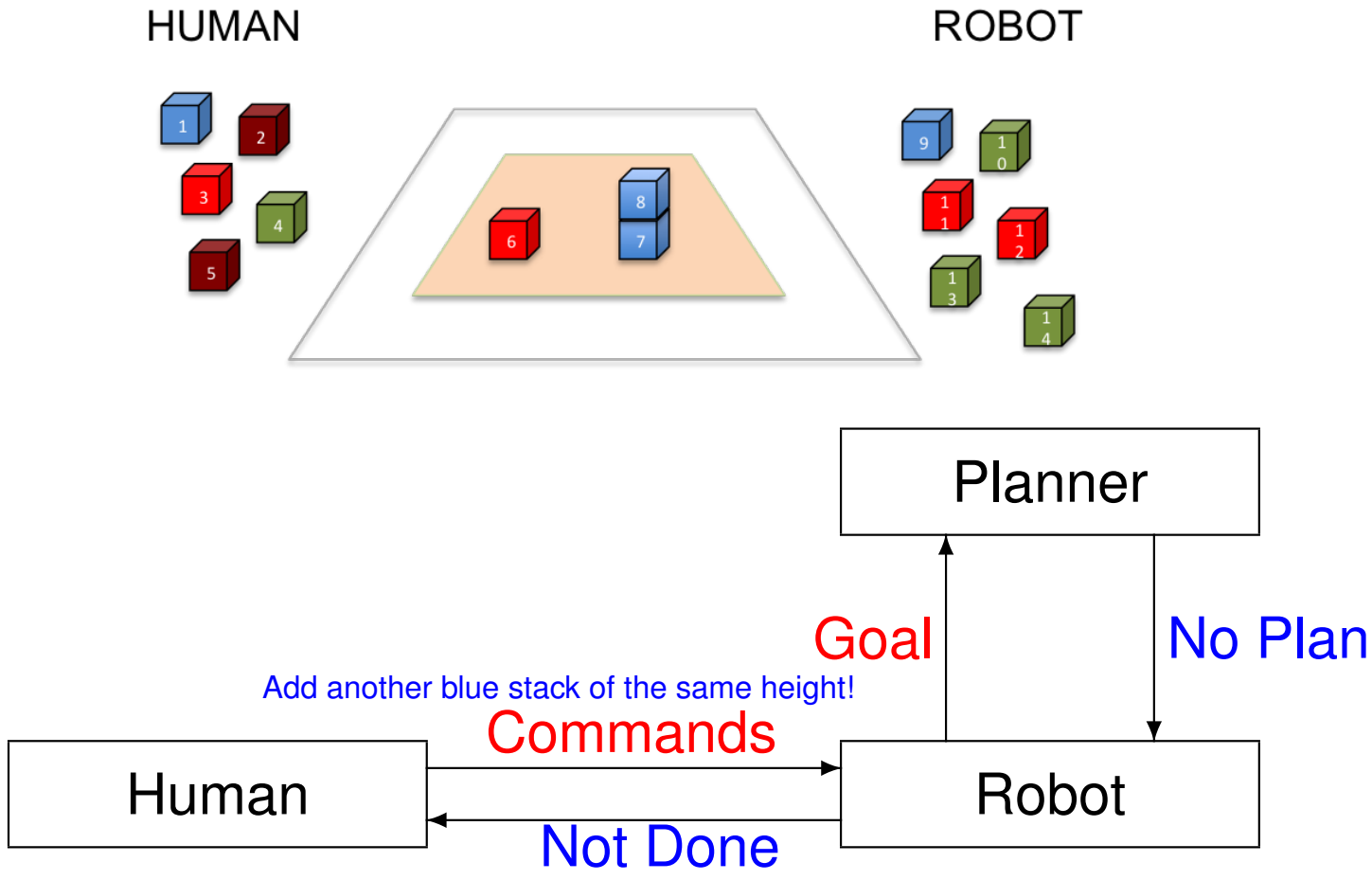
Human-Robot Interaction



Human-Robot Interaction



Challenges



- How to translate commands in NLP to goals?
- What to do if planner fails?
- How to communicate in NLP?

Challenges and Our Proposal

- How to translate commands in NLP to goals?
- What to do if planner fails?
- How to communicate in NLP?

Challenges and Our Proposal

- How to translate commands in NLP to goals?
  - Use NLP tool to translate command to goal description (This is a challenge to NLP! Initial attempt using kparser)
  - Planning with goal description is needed
- What to do if planner fails?
- How to communicate in NLP?

Challenges and Our Proposal

- How to translate commands in NLP to goals?
  - Use NLP tool to translate command to goal description (This is a challenge to NLP! Initial attempt using kparser)
  - Planning with goal description is needed
- What to do if planner fails?
  - Develop a framework for planning failure analysis
- How to communicate in NLP?

Challenges and Our Proposal

- How to translate commands in NLP to goals?
  - Use NLP tool to translate command to goal description (This is a challenge to NLP! Initial attempt using kparser)
  - Planning with goal description is needed
- What to do if planner fails?
  - Develop a framework for planning failure analysis
- How to communicate in NLP?
  - Future work

From NLP Commands to Goals

Command: “Add another blue stack of the same height!”
Use NLP tools to

- generate a set of goal conditions that describes the goal state, e.g.,
  - goal_condition(S,is,stack) :- block(S).
  - goal_condition(S,type,another) :- block(S).
  - goal_condition(S,color,blue) :- block(S).
  - goal_condition(S,height,same) :- block(S).

  This requires the understanding about the domain (e.g., a stack can be identified by its top block, a stack is said to be blue if all of its blocks are blue, etc.).
- provide rules for goal conditions.

Answer Set Planning with Goal Description

Answer set planning can deal with goal description by providing

- Rules need to be provided for each goal condition.
- Rules for checking all goal conditions.

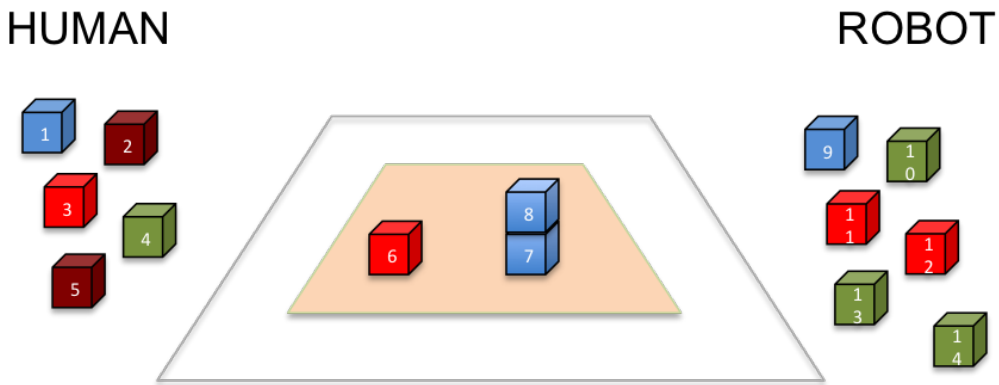
Answer Set Planning with Goal Description

Answer set planning can deal with goal description by providing

- Rules need to be provided for each goal condition. For example,
  - S represents a stack:
    satisfied(S,is,stack,T) :- block(S), time(T), clear(S,T).
  - Stack identified by S is blue:
    satisfied(S,color,blue,T) :- block(S), time(T),
    color(S,blue),clear(S,T),
    #count{U:above(U,S,T), not color(U,blue)}==0.
- Rules for checking all goal conditions. For example,

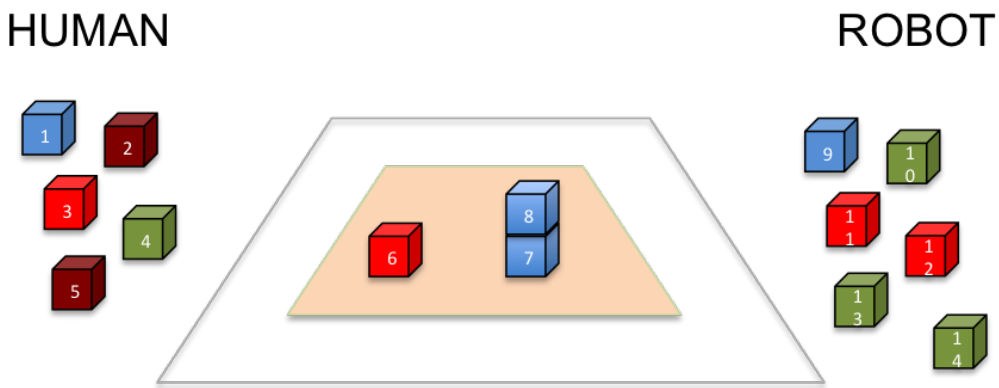
```
not_sat_goal(S,T) :- block(S),goal_condition(X,Y,Z),
      not satisfied(X,Y,Z,T).
sat_goal(S, T) :- not not_sat_goal(T).
:- X = #count {S : sat_goal(S, length)}, X ==0.
```

Planning Failure Analysis



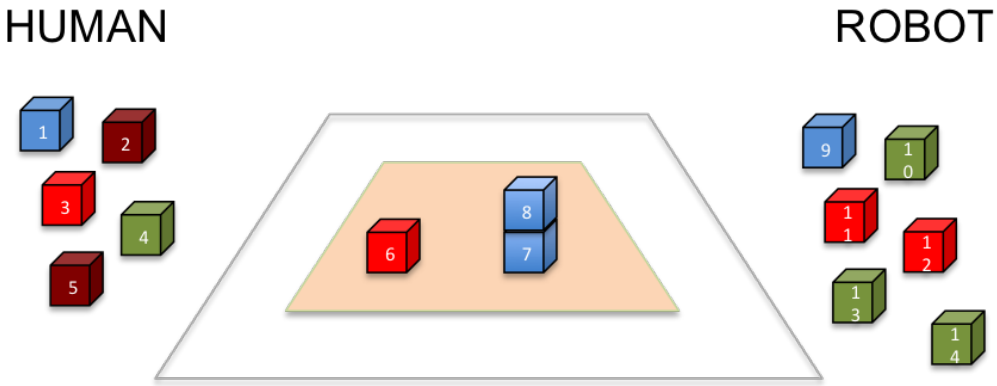
The robot cannot complete the command because its planner fails to generate a plan. What should it do?

Planning Failure Analysis



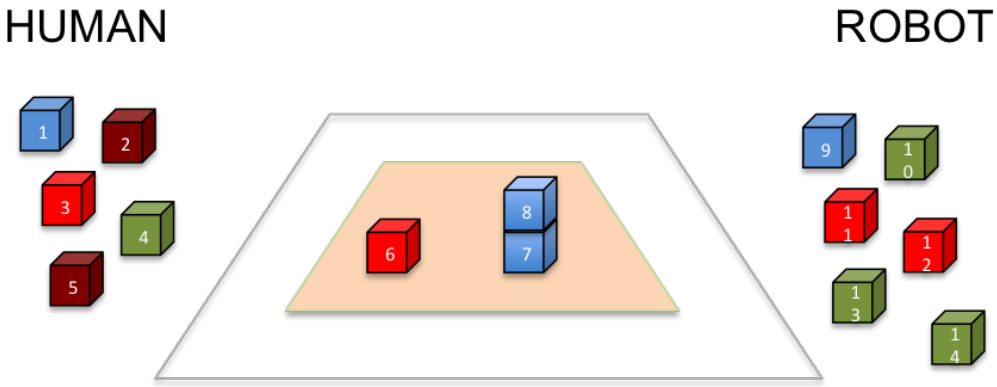
The robot cannot complete the command because its planner fails to generate a plan. What should it do?
It needs a planning failure analysis!

Planning Failure Analysis: Previous Approaches



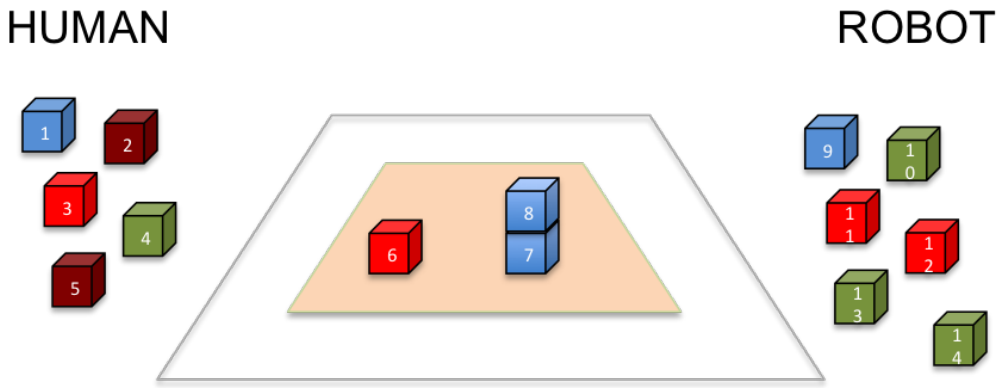
- the initial state is incomplete: assumption-based planning (McIlraith et al.)
  - identify a set of fluents whose values can be assumed (e.g., color of block 14 might be blue);
  - generate assumption-based plans whose execution conform with the values of the assumed fluents along its trajectory ([put_on_table(9), stack(14,9)]).
- partially satisfying the goal might be sufficient: partial satisfaction planning (Benton et al.)
  - assume that each subgoal has some utility; and
  - identify a set of satisfiable subgoals with maximal aggregated utility ([put_on_table(9)]).

Planning Failure Analysis: Previous Approaches



- the initial state is incomplete: assumption-based planning (McIlraith et al.)
  - identify a set of fluents whose values can be assumed (e.g., color of block 14 might be blue);
  - generate assumption-based plans whose execution conform with the values of the assumed fluents along its trajectory ([put_on_table(9), stack(14,9)]).
- partially satisfying the goal might be sufficient: partial satisfaction planning (Benton et al.)
  - assume that each subgoal has some utility; and
  - identify a set of satisfiable subgoals with maximal aggregated utility ([put_on_table(9)]).

Planning Failure Analysis: Our Approach



Action domain used by the planning system could be incomplete: there are actions that the robot does not want to use if they are not needed, e.g.,

- asking to use the human's blocks
- paint some blocks blue

Formalization (Recently Developed)

Given: a planning problem $P = (F, A, I, G)$

- P needs a planning failure analysis if it has no solution.
- (AF, AC), where AF is a set of fluents and AC is a set of action, is an extension of the problem P.
- A planning failure analysis of P w.r.t. an extension (AF, AC) is a pair (F', A') such that $F' \subseteq AF$, $A' \subseteq AC$, and there is an interpretation I' of F' so $(F \cup F', A \cup A', I \cup I', G)$ has a solution.
- (F', A') and (F'', A'') are analyses of P w.r.t. (AF, AC), (F', A') is more preferred to (F'', A'') if $A' \subsetneq A''$.
- More preferred analyses can be computed using answer set programming.

Computing Preferred Analyses

- $P = (F, A, I, G)$ encodes as usual with declarations of actions, fluents, etc. and action generation rule:
  1 {occ(A, T): action(A)} 1 :- time(T).
- Add description of (AF, AC) with the choice rule
  {is_ac(a)}.
  for each $a \in AC$
- Add the rule
  action(A) :- is_ac(A).
- Minimizing the set of additional actions
  number_actions(N) :- #count {A : is_ac(A)}.
  #minimize {N : number_actions(N)}.

This implementation is sound but incomplete. Develop new implementation based on CR-Prolog for completeness.

Conclusions and Future Work

Conclusions

- Propose a formalization of planning failure analysis.
- Use answer set programming for computing preferred analyses

Future Work

- Continue with the translation from commands (communications) to goals
- Generating communications in NLP