All's well that ends well
Planning with Global Abduction

Satoh, Ken

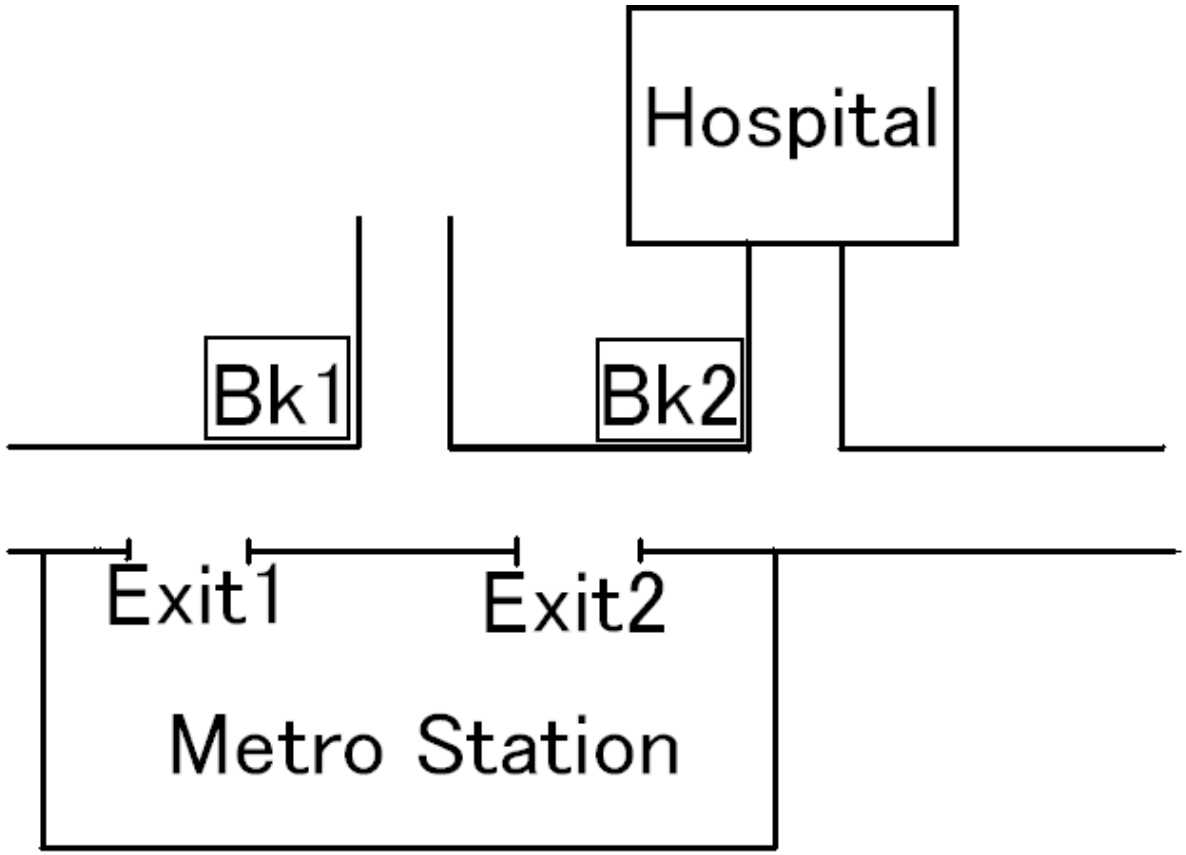
佐藤　健

NII and Sokendai

- Motivating Example
- Plan modification upon failure
- Global Abduction and "All's Well that Ends Well" Principle
- A Solution to Motivating Example
- Conclusion

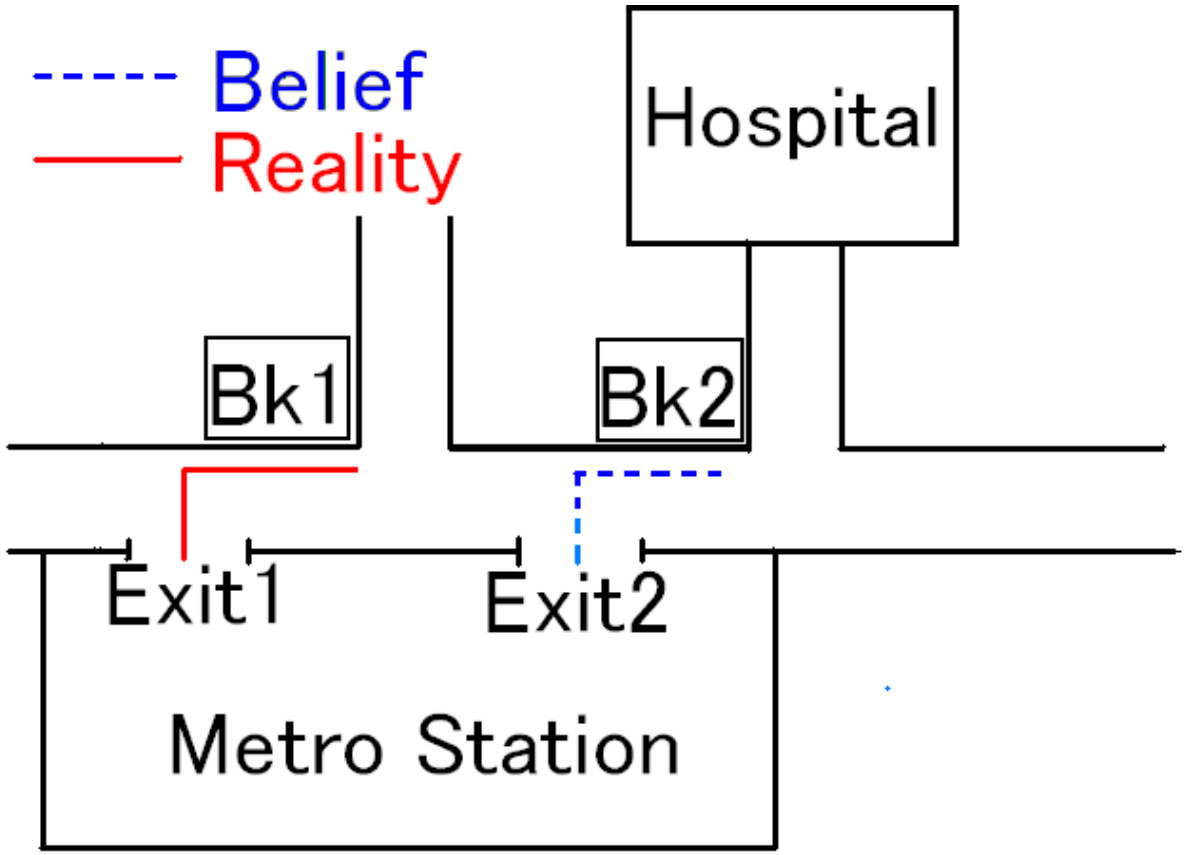
Motivating Example

Suppose that a robot was told to go to the hospital from the metro station with the following map. Suppose that it was out of the exit without knowing which.



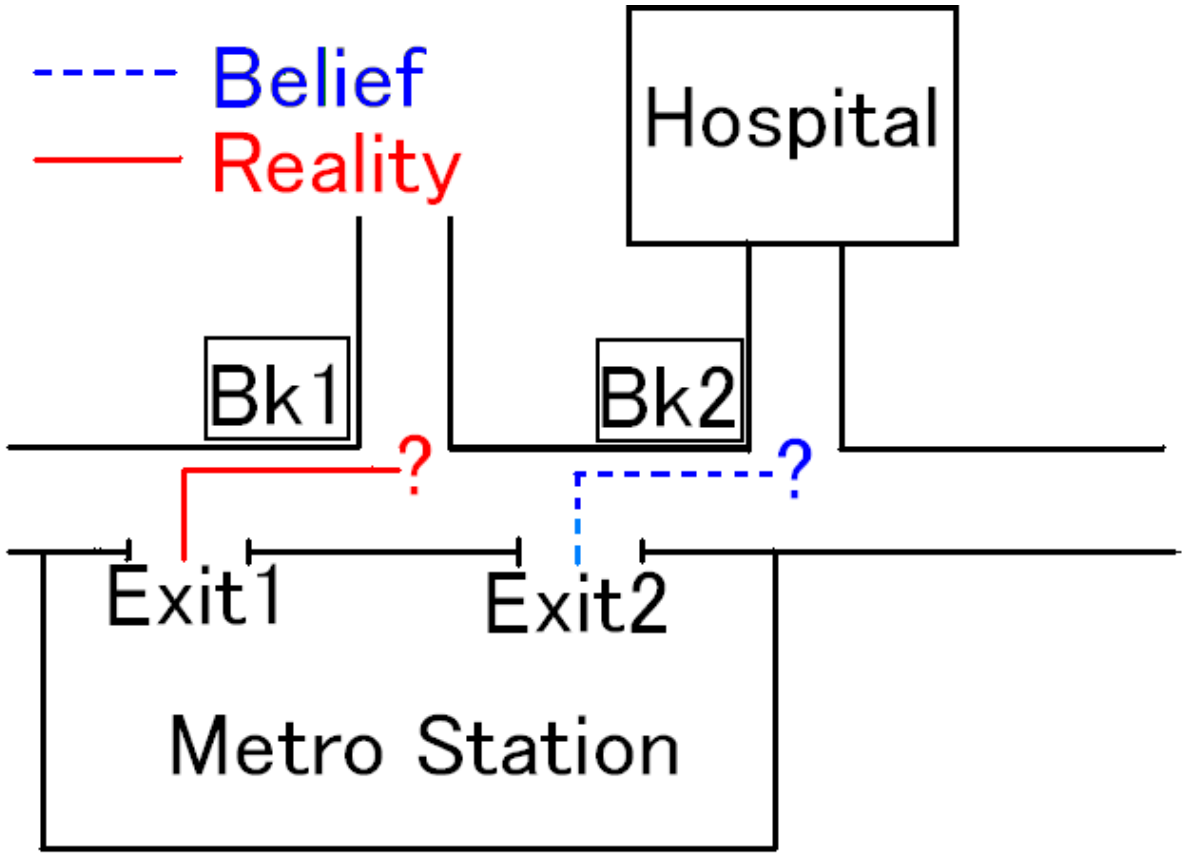
Motivating Example(continued)

Suppose that the robot believed that it had got out of the exit2, while actually the robot got out of the exit1. Then, the actual move and the believed move are like: (We assume that a robot cannot distinguish bank1 from bank2)



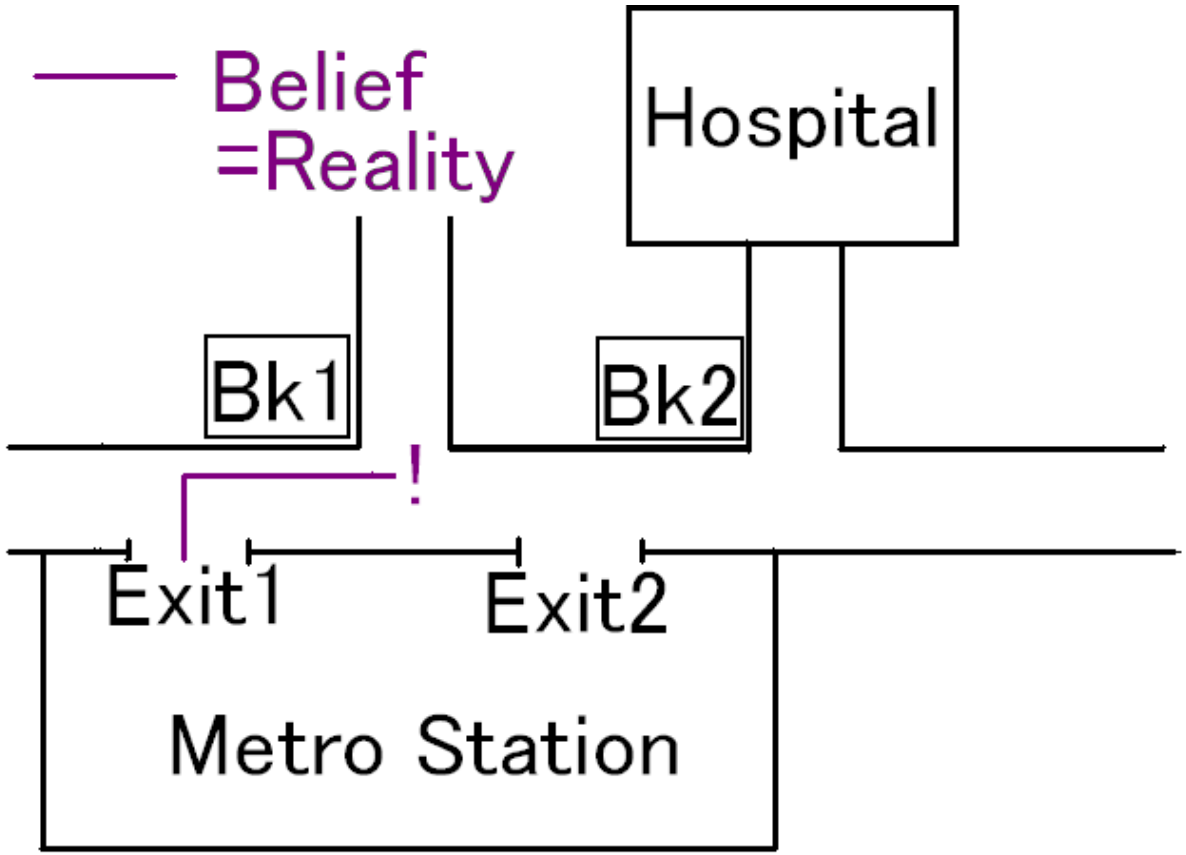
Motivating Example(continued)

Then, at the corner of the bank, the robot tried to turn left to the hospital, but the robot found that the hospital does not exists.



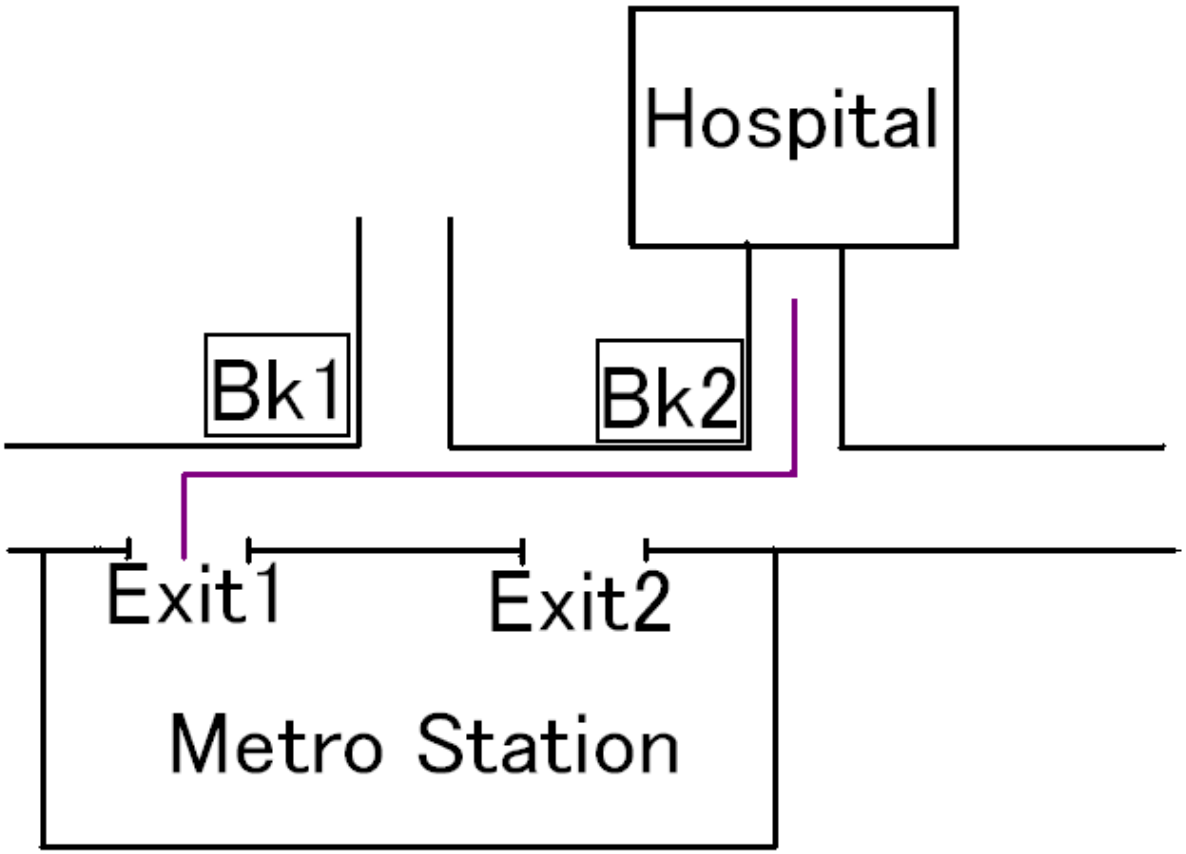
Motivating Example(continued)

Then, the robot revised his initial belief and the robot came to believe that the robot went out from the exit1.



Motivating Example(continued)

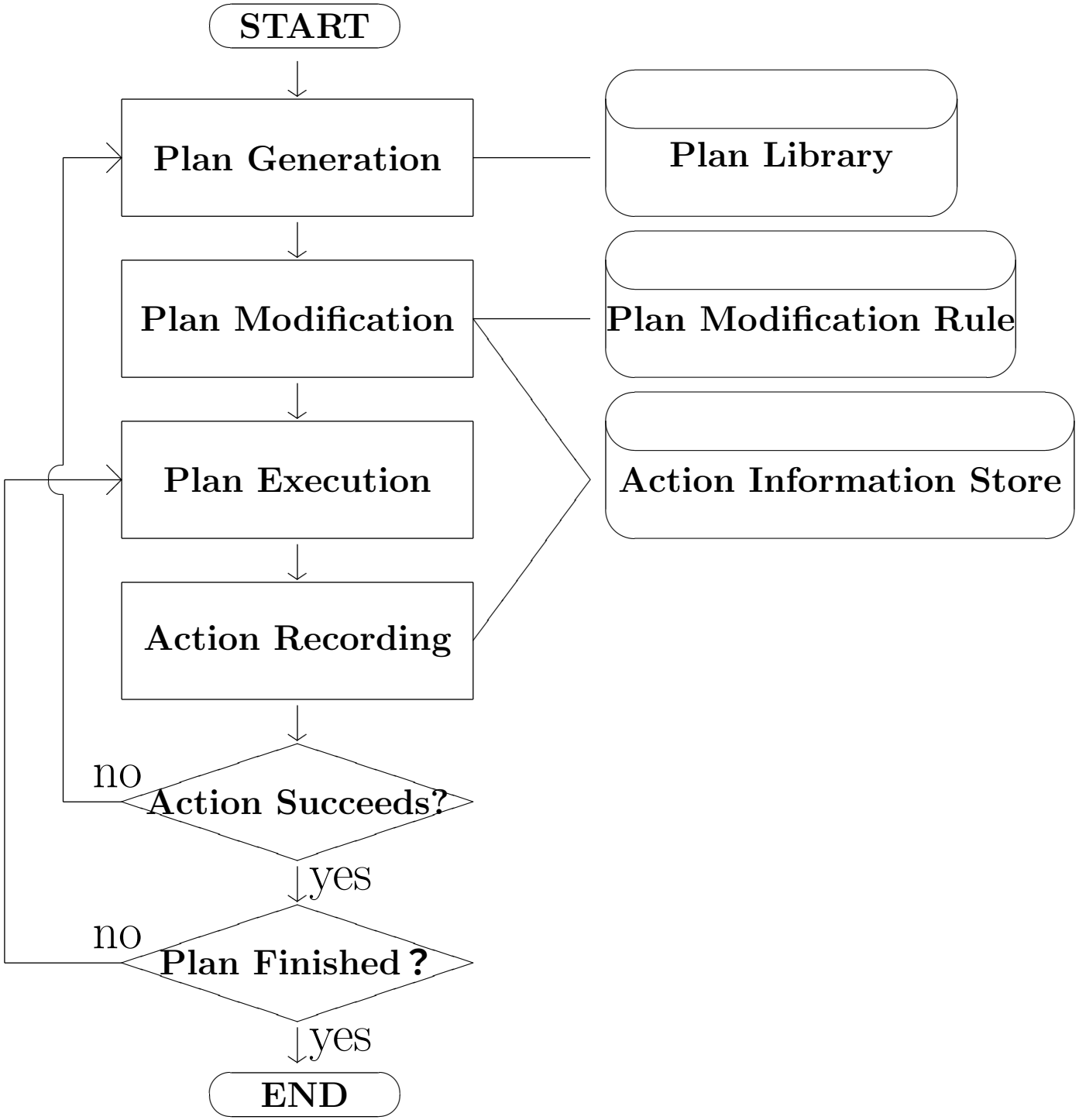
The robot understood that he should go further forward to find the correct bank which leads to the hospital.



Characteristic of the Above Example

- There is a failure of a plan which cannot be detected at the time of planning.
- Agent makes actions with side-effects.
- Agent changes a plan on the fly when the failure occurs.
- Agent must consider the side-effects by the already-executed actions in changing a plan.
- Agent can make use of these side-effects in a changed plan.

Plan Modification Upon Failure



Technical Problem

We would like to solve the above problem by logic programming with backtracking.
Issues to be solved:

- How to propagate the information of already-executed actions in one path (plan) to another path (plan)? even with backtracking
- How to modify a plan when failure occurs?
- What is the semantics for this robot behavior to talk about the correctness of internal mechanism?

A Solution

- How to propagate the information of already-executed actions in one path (plan) to another path (plan)? even with backtracking
  $\Rightarrow$ **Propagation of previous action information by global abduction**
- How to modify a plan when failure occurs?
- What is the semantics for this robot behavior to talk about the correctness of internal mechanism?

Global Abduction

- Abduction in current abductive logic programming can locally be used in one search path. Therefore, hypothesis is retracted upon backtracking.
- Global abduction is a new abduction mechanism where abuduced literals are accessible in other search paths.
- We use two kinds of anotations for global abduction:
  - `announce`(L) to globally abduce a literal L.
  - `hear`(L) to check the truth-value of the abduced literal L.
- We propose "all's well that ends well" principle for correctness of global abduction.

Intuitive meaning of $\texttt{announce}(L)$

Announcing literal $\texttt{announce}(L)$:

- We assume the **global belief state** which expresses a set of literals abduced so far.
- Addition of L to the global belief state.
- We can make this assertion accessible from another search path, thus "global abduction"

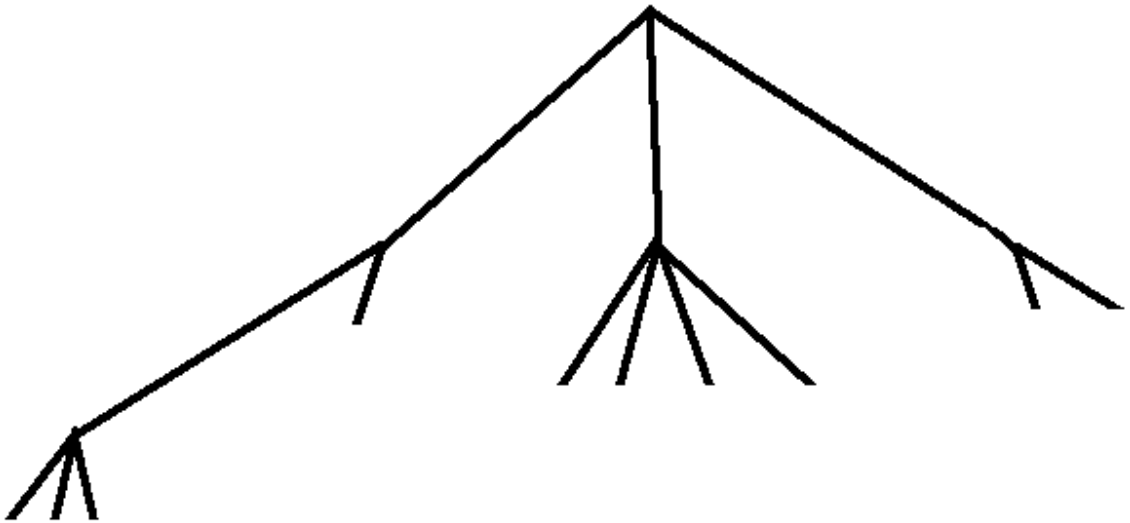
Intuitive meaning of `hear`(L)

Hearing literal **`hear`**(L):

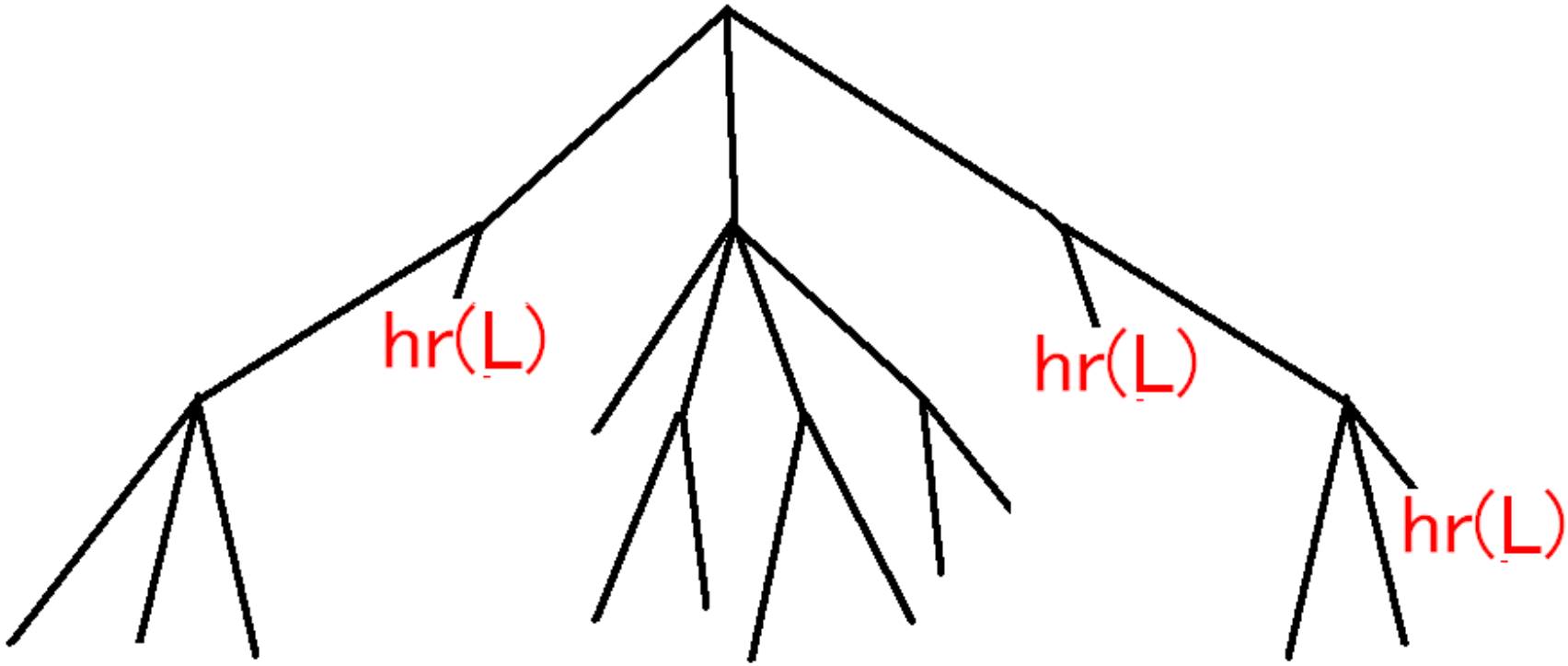
- An access method of the current belief state.
- The truth value of **`hear`**(L):
  - true if L in the current belief state
  - false if $\overline{L}$ in the current belief state
  - undefined if neither L nor $\overline{L}$ in the current belief state.
- In a procedure semantics, undefined **`hear`**(L) makes a search of a path suspended.

Search Tree

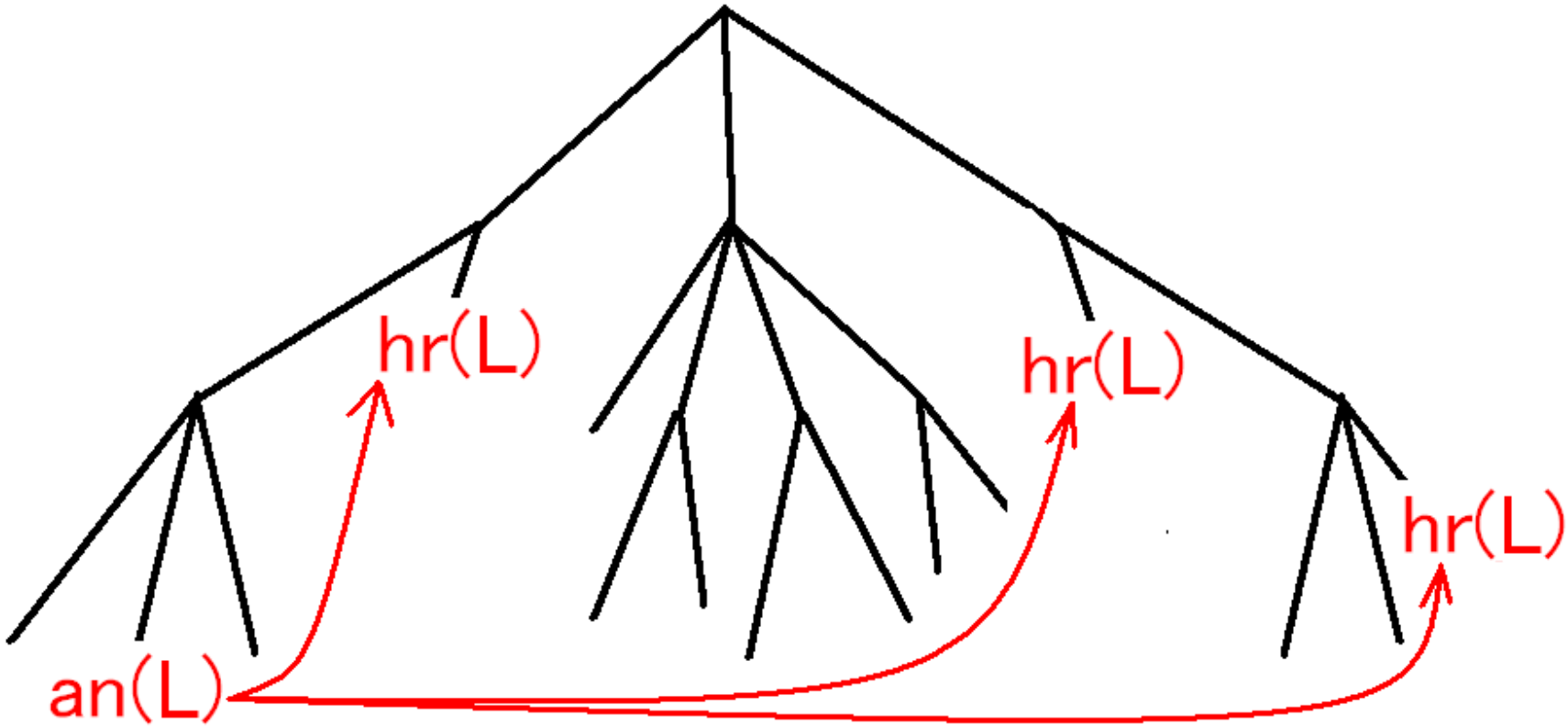
Search Tree



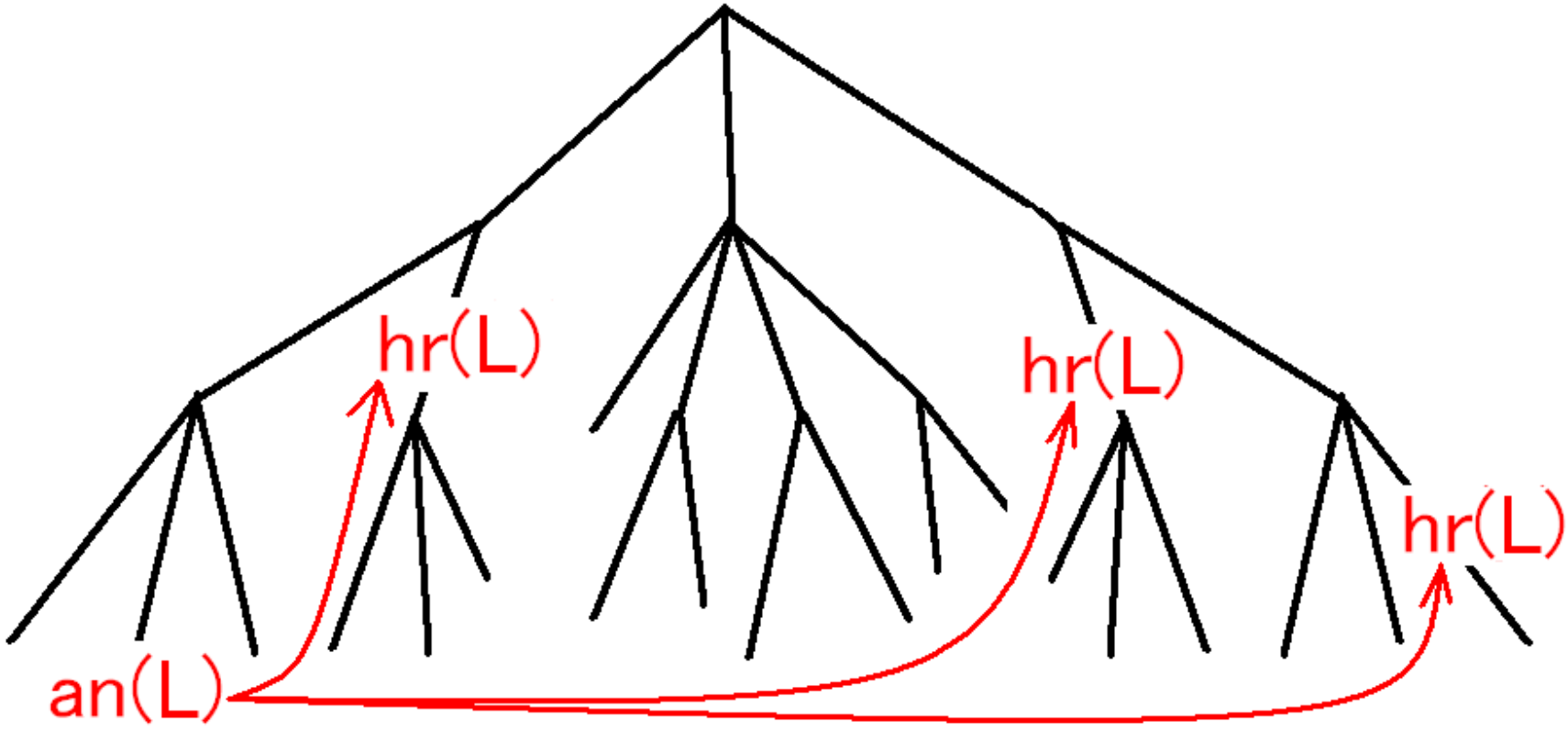
Some Paths are Suspended by a Hearing Literal



When a Path Globally Abduces the Literal,...



When a Path Globally Abduces the Literal, the Suspended Paths are Resumed



A Solution

- How to propagate the information of already-executed actions in one path (plan) to another path (plan)? even with backtracking
- How to modify a plan when failure occurs?
  $\Rightarrow$ **Simulation of previous actions globally abduced in other search pathes**
- What is the semantics for this robot behavior to talk about the correctness of internal mechanism?

Metro Example

Rules for plan generation:

```
plan(X,X,[]):-!.
plan(X,Y,[turnright(X,Z)|L]):-
    right(X,Z),plan(Z,Y,L).
plan(X,Y,[turnleft(X,Z)|L]):-
    left(X,Z),plan(Z,Y,L).
plan(X,Y,[goforward(X,Z)|L]):-
    forward(X,Z),plan(Z,Y,L).


exit(exit2).
exit(exit1).
right(exit2,bank2).
left(bank2,hospital1).
right(exit1,bank1).
forward(bank1,bank2).
```

Metro Example(continued)

Rules for plan modification:
If there are already executed actions, these actions is not executed again in other search pathes. (But we simulate them as if it were executed.)

```
modify_plan([],[]):-!.
modify_plan([turnright(X,Y)|Plan],
            RevisedPlan):-
  X is_a Cx, Y is_a Cy,
  hear(turnedright(Cx,Cy)),
  !,modify_plan(Plan,RevisedPlan).
modify_plan([turnleft(X,Y)|Plan],
            RevisedPlan):-
  X is_a Cx, Y is_a Cy,
  hear(turnedleft(Cx,Cy)),
  !,modify_plan(Plan,RevisedPlan).
```

Metro Example(continued)

Rules for plan modification:

```
modify_plan([goforward(X,Y)|Plan],
            RevisedPlan):-
  X is_a Cx, Y is_a Cy,
  hear(wentforward(Cx,Cy)),
  !,modify_plan(Plan,RevisedPlan).
modify_plan([Action|Plan],
            [Action|RevisedPlan]):-
  !,modify_plan(Plan,RevisedPlan).
```

Metro Example(continued)

Rules for execution:

```
execute([]):-!.
execute([turnright(X,Y)|Plan]):-
     X is_a Cx, Y is_a Cy,
     hear(seeright(Cy)@sensor),
     !,
     announce(turnright(Cx,Cy)@actuator),
     announce(turnedright(Cx,Cy)),
     execute(Plan).

execute([turnleft(X,Y)|Plan]):-
     X is_a Cx, Y is_a Cy,
     hear(seeleft(Cy)@sensor),
     !,
     announce(turnleft(Cx,Cy)@actuator),
     announce(turnedleft(Cx,Cy)),
     execute(Plan).
```

Metro Example(continued)

Rules for execution:

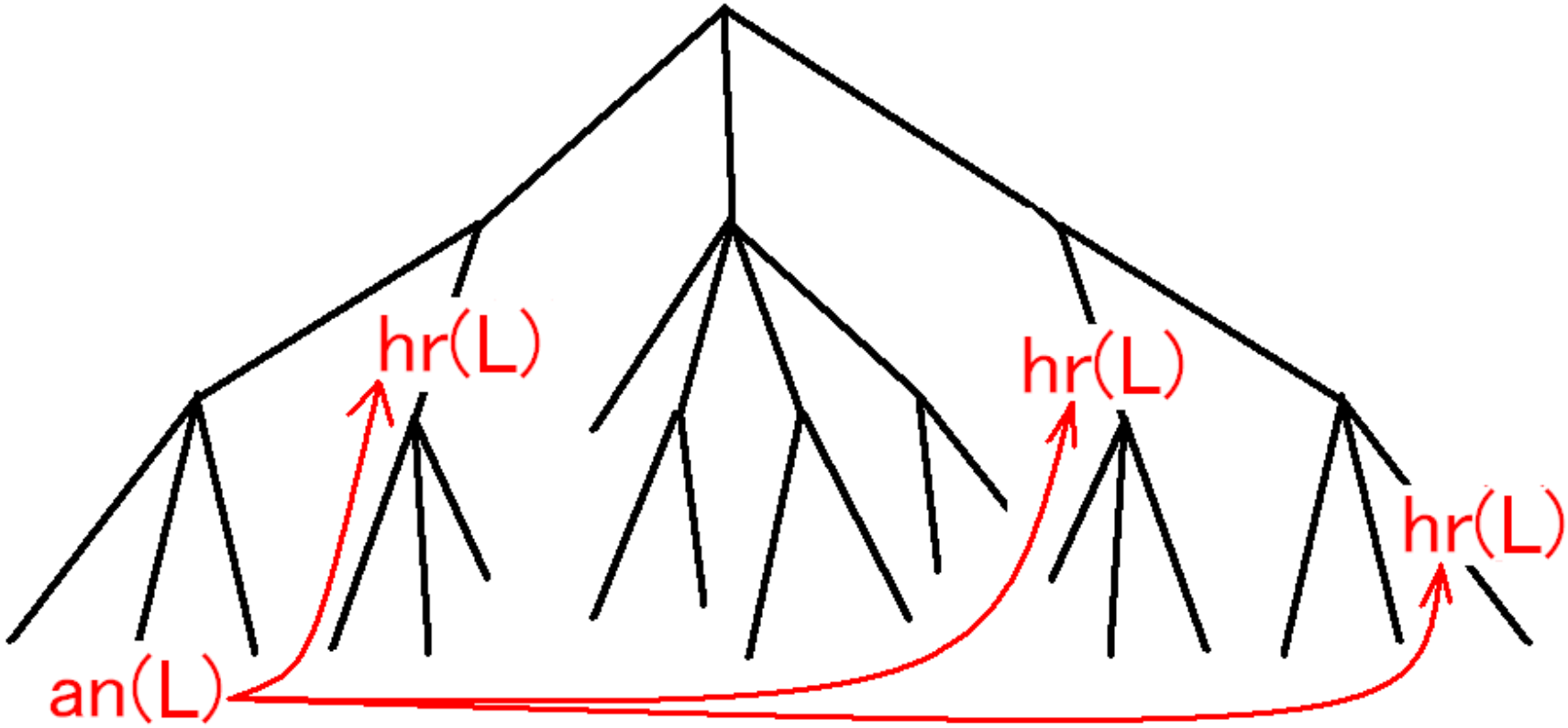
```
execute([goforward(X,Y)|Plan]):-
      X is_a Cx, Y is_a Cy,
      hear(seeforward(Cy)@sensor),
      !,
      announce(goforward(Cx,Cy)@actuator),
      announce(wentforward(Cx,Cy)).
```

A Solution

- How to propagate the information of already-executed actions in one path (plan) to another path (plan)? even with backtracking
- How to modify a plan when failure occurs?
- What is the semantics for this robot behavior to talk about the correctness of internal mechanism?
  $\Rightarrow$ **"All's Well that Ends Well" Principle**

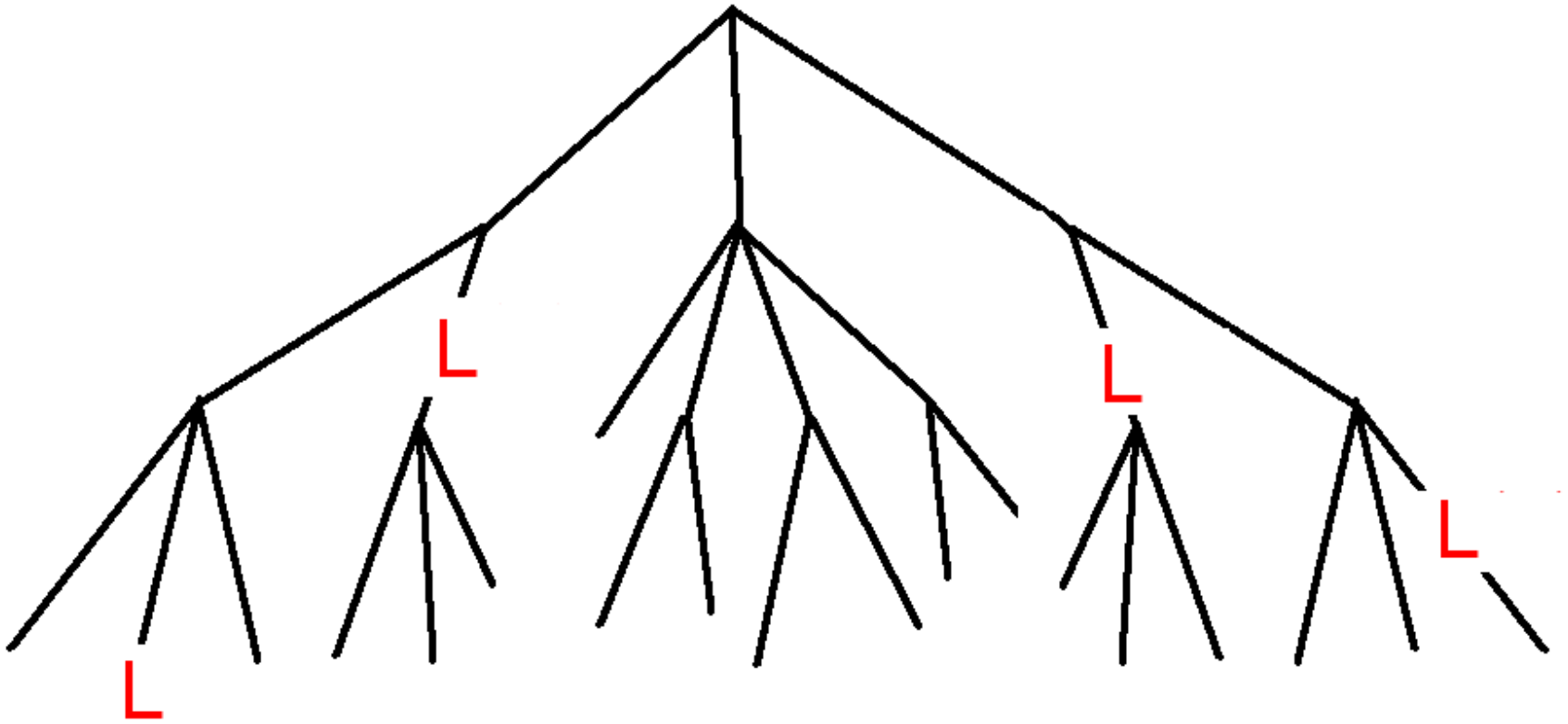
"All's Well that Ends Well" Principle

Suppose that we obtain a solution from the search tree below.



"All's Well that Ends Well" Principle(continued)

If we ignore annotations in the program, and add to the initial program the last set of abduced literals when the solution is obtained, then the same result should be derived by the usual SLDNF procedure.



Conclusion

- We propose **global abduction** in which we abduce a belief literal and can use abduced belief in a different search path.
- We formalize a semantics of global abduction as **"all's well that ends well"** principle.

Future Works

- Develop a user-friendly language for plan modification
- Give a semantics for global abduction during execution
  A semantics for best-effort?