Foundations of Language Interaction

HANDOUT TWO

May 31, 2001
M. Stone
mdstone@cs.rutgers.edu

1 Introduction

Today we consider the topic of ACTIONS, PLANS and INTENTIONS, developing some key ideas.

(1) Actions are the basic choices that agents make in each step of the cycle of perception and choice.

(2) To start, we'll treat a plan is an argument that demonstrates how performing a sequence of actions in the current circumstances leads to desired effects.

(3) An intention is a plan that the agent is committed to.

An intention guides the agent's deliberation by proposing actions that the agent might choose, focusing the agent's attention to the circumstances in the world that make the action appropriate, allowing the agent to more quickly identify problems and opportunties that arise in carrying out the plan, and suggesting ways that the agent can respond to the unexpected.
Recap and extension: an agent simulator with intentions.

(4)
```
agitate State Intentions :-
    perceive State Beliefs,
    update State Beliefs Intentions NewIntentions,
    !,
    act State NewIntentions.
```

(`act` has to call `agitate` recursively, of course.)
Today's objective is to see how to flesh out this program, by looking at some simple but classic representations of ACTIONS, PLANS and CONDITIONS, and by exploring some fundamental ideas about SEARCH.

2 Actions

The central challenge of describing many kinds of real-world action is finding a good approach to INERTIA.

(5) A state in the world tends to persist "by inertia", unless an action occurs whose effects explicitly disrupt that state.

The STRIPS representation of [Fikes and Nilsson, 1971] is a classic approach, which describes actions in terms of PRECONDITIONS, ADDITIONS and DELETIONS.

(6) a The PRECONDITIONS of an action is a list of facts that must be true in order for the action to be executed (that is, executed at all, executed safely or executed successfully).

b The ADDITIONS of an action is a list of facts that specify the new states in the world that the action institutes—the positive effects of an action.

c The DELETIONS of an action is a list of facts that specify the states in the world that are in progress when the action begins but that the action disrupts.

This description describes an algorithm for simulating the effects of action that builds-in inertia in a convenient way.

(7) a Suppose facts *F* are true and you perform action *E*, with strips description *P*, *A*, *D*.

b If every fact in *P* occurs in *F*, then the state of the world after doing *E* consists of *F* with the facts in *D* removed and the facts in *A* added.

To specify STRIPS in λProlog, we have this background:

(8)
```
a  kind fact, action type.
b  type is_action
   action -> list fact -> list fact -> list fact -> o.
```

A door domain.

(9)
```
a  type closed, open, empty, keyed, locked, unlocked fact.
b  type key, turn, pull, unkey action.
c  is_action turn (keyed::locked::nil) (unlocked::nil)
   (locked::nil).
d  is_action pull (closed::unlocked::nil) (open::nil)
   (closed::nil).
```

...and so forth.

3 Plans and intentions

Plans are our first interesting data structure.

(10)
```
a  kind plan type.
b  type finish list fact -> plan.
c  type step list fact -> action -> plan -> plan.
```

Recall the intuition: a plan is an argument showing that a sequence of actions performed in the current circumstances will lead to a desired effect. Our data structure maps out this argument recursively.

(11) a The simplest plan is a plan for no actions. In this case the desired effect must be true now, and it suffices to spell out what the desired features of the current circumstances are. If this GOAL is `G`, the corresponding plan structure is `(finish G)`.

b A more complicated plan must spell out the first action to be performed, `E`. If we guarantee certain facts `C` to be true when `E` is performed, we will be able to argue that a new set of circumstances obtains after `E` has been executed. Call this `C2`. After we do `E`, then, we will have to continue with a new plan that argues, using only facts in `C2`, that a subsequent sequence of actions will lead to a desired state. If this subplan is `P`, the overall plan structure we need now is `(step C E P)`.

A medium-sized plan:

(12)
```
(step (closed::keyed::locked::nil)
 turn
 (step (closed::unlocked::nil)
  pull
  (finish (open::nil))))
```

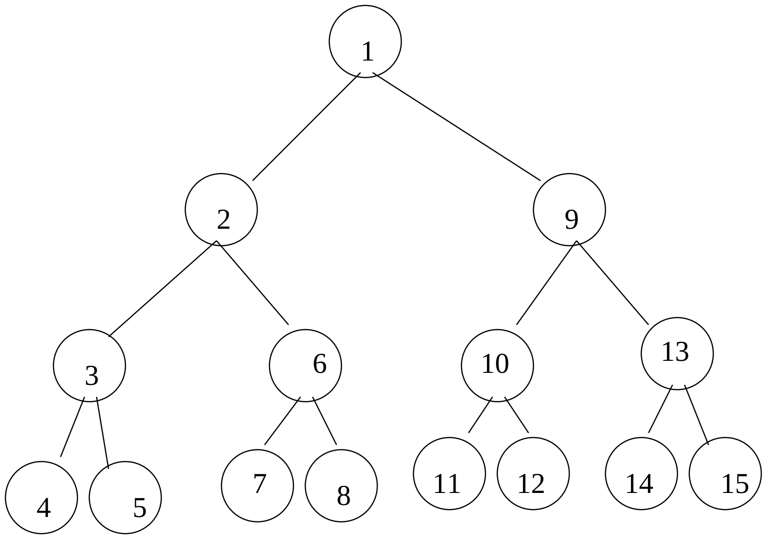
The core of a plan-building routine.

(13)
```
a  type bp1 list fact -> plan -> plan -> o.
b  bp1 F P P :- circ P C, sublist C F.
c  bp1 F P R :-
       is_action A Pre Post Del,
       circ P C,
       remove Post C PC,
       disjoint PC Del,
       union Pre PC AC,
       bp1 F (step AC A P) R.
```

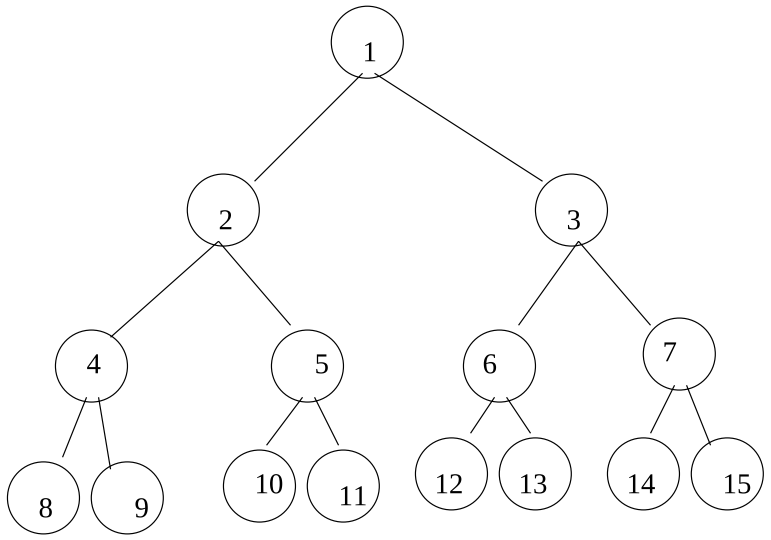
It has the right logic, but it doesn't work because of search.

4 Search

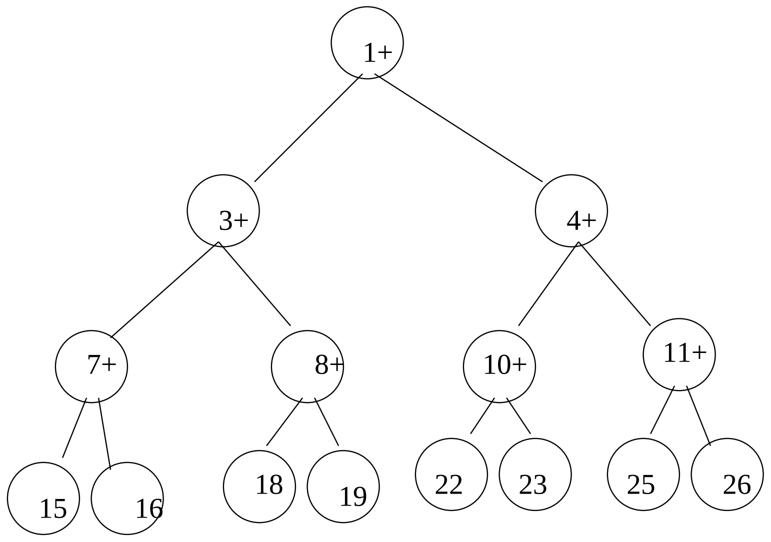
Logic programming languages do DEPTH-FIRST SEARCH.



Bad news if there's infinite paths.
An alternative is BREADTH-FIRST SEARCH.



But this means scrapping the logic programming search all together. Plus it takes an obscene amount of space.
Standard quick compromise ITERATIVE DEEPENING SEARCH.



Do depth-first search, but have a maximum depth that you're willing to consider at any stage. If you find no solution, increase the depth bound.
Nice λProlog higher-order implmentation of iterative deepening.

(14) a `type id (int -> o) -> int -> o.`
b `id P D :- P D.`
c `id P D :- N is D+1, id P N.`

Better plan search:

(15) `id (n \ bpn F P R n) 0`

5 Two final things

Handling unexpected events with intentions.

```
(16) a  type update state -> list fact -> plan -> plan -> o.
     b  update State Facts Plan Plan :-
            circ Plan C, sublist C Facts.
     c  update State Facts Plan Next :-
            id (n \ (build_plan Facts Plan Next (a  true) n;
            repair_plan Facts Plan Next n)) 0.
```

```
(17) a  type repair_plan list fact -> plan -> plan -> int -> o.
     b  repair_plan Facts (step Pre B Post) Next Depth :-
             build_plan Facts Post Next (a \ not (a = B)) Depth.
     c  repair_plan Facts (step Pre B Post) Next Depth :-
            repair_plan Facts Post Next Depth.
```

Using intentions to decide what to do next.

```
(18) a  type act state -> plan -> o.
     b  act State (finish G).
     c  act State (step C A P) :-
            execute State A,
            agitate (do A State) P.
```

An intention guides the agent's deliberation by proposing actions that the agent might choose, focusing the agent's attention to the circumstances in the world that make the action appropriate, allowing the agent to more quickly identify problems and opportunties that arise in carrying out the plan, and suggesting ways that the agent can respond to the unexpected!

References

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.