

Lecture 16

Dealing with inexact information

The real world usually include information that is not exact, e.g., the agent might be missing information, and some information known by the agent might be incorrect.

We will discuss how to deal with such situations when we build and execute plans.

Reference:

- Textbook Section 13.1–13.2

AI(0270)

New example problem

Consider a situation in which we are in a car, and **one of the tire is flattened**. But we have a spare tire.

- Start:** $\neg Flat(Spare), Intact(Spare), Off(Spare), On(Tire1), Flat(Tire1)$.
A tire is intact if it is still okay to use it, e.g., no hole in it.
- Finish:** $On(x), \neg Flat(x)$.
We have extended our language a bit to allow negated sentences. The change is actually trivial.
- Action 1:** $Remove(x)$. Precondition $On(x)$; Effect $\neg On(x), ClearHub$.
The $ClearHub$ predicate tells us whether we can put another tire on it. Note that the predicates is just for convenience, because $ClearHub \Leftrightarrow \neg \exists x On(x)$.
- Action 2:** $PutOn(x)$. Precondition $Off(x), ClearHub$; Effect $On(x), \neg ClearHub$.
- Action 3:** $Inflate(x)$. Precondition $Intact(x), Flat(x)$; Effect $\neg Flat(x)$.

AI(0270)-16.1

Sources of inexact information

- So far, our planner works in an **accessible, static and deterministic** environment. Knowledge are guaranteed to be complete and correct.
- But this time it is not really the case...
 - Some information of the world may **not be known to the agent** at the beginning, and the agent must execute some step to **find it out**.

E.g., we don't know whether $Intact(Tire1)$ is true. By close-world assumption, we assume it is false. But the plan can be much better if it turns out to be true.

- Some information of the world may be **incorrect**, or another agent work at the same time which interfere the plan of the agent.

E.g., it might turn out that $\neg Intact(Spare)$.

They differ only in the way we see it. E.g., we might say that we wrongly think that $\neg Intact(Tire1)$, or don't know whether $Intact(Spare)$ is true.

AI(0270)-16.2

Strategies for dealing with inexact information

Depending on **when** to deal with inexact information, 2 strategies are commonly used.

- We may **always check the condition**, and plan for **both the cases in which the condition is true or false**. We call this **conditional planning** or **contingency planning**.
- We may **assume that the favourable case occurs**, but **constantly check the percepts** and replan if bad things happen. We call this **execution monitoring**.

Their relation: execution monitoring **defer the check** about the condition until the time when we **execute** the plan.

AI(0270)-16.3

Allowing conditional Planning: the sensing action

- In **conditional planning**, we **plan for the acquisition of extra knowledge**, and plan ahead what to do **for each possible result**.
- We will have some **actions** which **gives out extra information**. We call these **sensing actions**. E.g., we might have the following action:

- Action $CheckTire(x)$:

Precondition: none;

Effect: $KnowsWhether("Intact(\underline{x})")$.

The double quotes means that what's within is not a function (i.e., object), but instead a predicate.

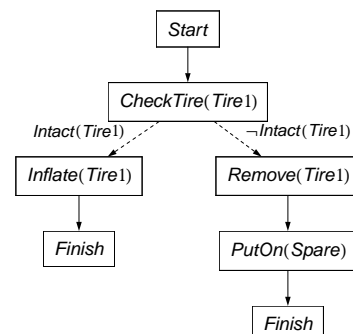
The underlined x means that the x is referring to the argument x, even though it is within double quotes.

- The **result** of a sensing action is to add the new knowledge into the KB. In this case, either $Intact(x)$ or $\neg Intact(x)$.

AI(0270)-16.4

A plan that has a sensing action

A plan with a sensing action looks like this:



AI(0270)-16.5

How to execute such plan?

- The plan **execution** is similar to our previous plans.
- Everytime, we find a step which has **no remaining ordering constraints**, i.e., no other arrows pointing to it.
- Then we **execute** the step.
- If the step is a sensing action, then there will be a **then-part** and an **else-part**.
- **Depending on the knowledge added** by the sensing action, we choose one part of the plan, and ignore the other part.
- When **making** the plan, we will have to make sure that **the preconditions of the finish state is achieved** in both parts of the plan.

AI(0270)-16.6

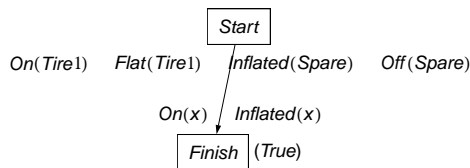
Extension to our state: execution context

- How to **make such plans**? It is not too much different from POP.
- Whenever we want to **choose** an action to satisfy an open precondition c of step S , we are allowed to **use a sensing action** that **may** add the condition.
- If we choose a sensing action, **the step S acquires a context** of c , i.e., it **assumes** that the sensing action adds a particular sentence.
- We call the causal link added by such sensing action to be **conditional links**: they represent the condition in the final plan.
- So every step has a **context** associated with it. At the beginning, the context is *True*, meaning that nothing is assumed.
- Context **propagate through causal and conditional links**. That is, if a step acquires a context x , then all steps with a causal link from it also get that context.

AI(0270)-16.7

First step

We have our normal "trivial initial plan":



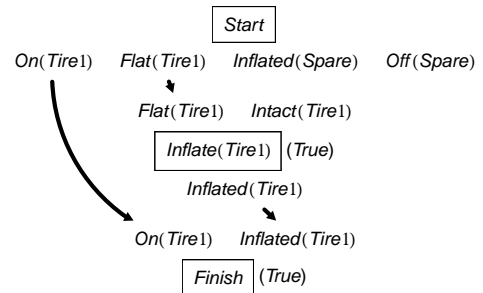
Note that we have added a context into the finish node. We might need to **change the context** if we find that we are planning only for a **specific case**.

The Start node always has the *True* context, so we avoid writing them out. It cannot have some other context because no condition node can be executed before it to introduce a context for it.

AI(0270)-16.8

The easier route...

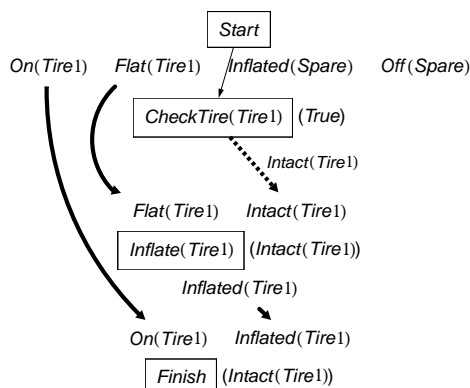
It is probably much easier (i.e., cost less) to inflate a tire than to put it off and put on another again. So our first try...



But there is a problem now... there is no way to establish *Intact(Tire1)*.

AI(0270)-16.9

Introducing a condition



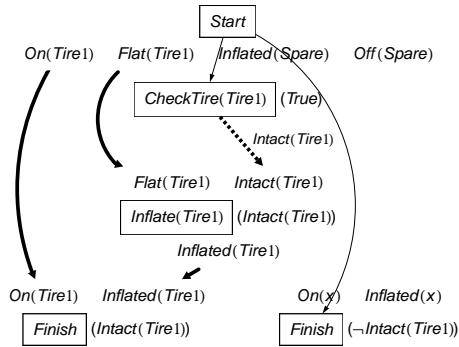
AI(0270)-16.10

A few things to note...

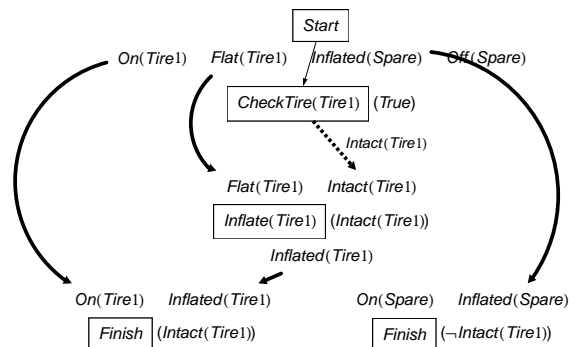
- Now, if the check succeed, we get a cheap way to achieve the goal.
- However, the *Intact(Tire1)* condition really **depend on a conditional link**. So the context of *Inflat(Tire1)* becomes *Intact(Tire1)*.
- The context is **propagated** to the Finish node, because the Finish node **depend on a causal link supported by *Inflat(Tire1)***.
- So the plan is **completed**, because all the pre-conditions are supported by causal links...
- ... if the conditional link is supported by the sensing action.
- But **we can't stop here**. The context of Finish is **not general enough** to cover all the possible combinations of conditions.
- What to do now? **Generate a new Finish state...**

AI(0270)-16.11

What if the condition does not hold?



Continue our plan



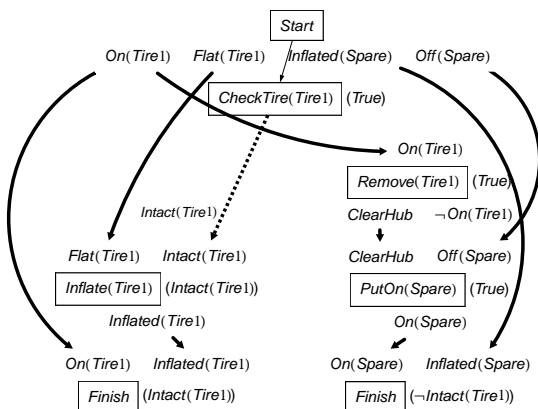
Note that we have a new context $\neg Intact(Tire1)$. This helps us to rule out the same plan, because it is inconsistent with the context.

There is only one other way to get $Inflated(x)$, so try it.

AI(0270)-16.12

AI(0270)-16.13

Adding more steps



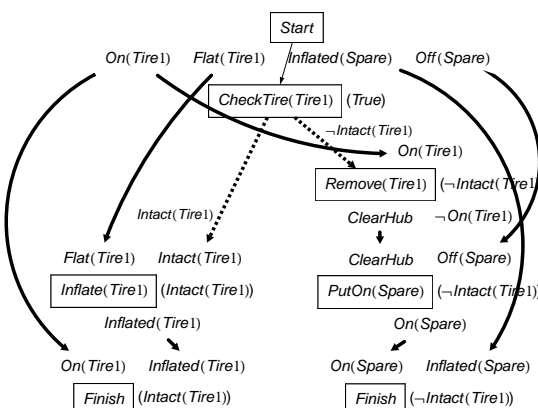
More observations

- The newly introduced steps have the context *True*, because we don't know that they **can't be executed in other contexts**.
- All the preconditions are satisfied now. So we are left with one thing: **checking the plan for conflicts to actions**.
- There is one problem such problem: the $\neg On(Tire1)$ effect of *Remove(Tire1)* conflict with the causal link towards the **first** Final state.
- This needs consideration because **the intersection between the two context is non-empty**.
- We can perform promotion and demotion. But in conditional planning, we have one more choice: **conditioning**.
- Basically, if we **change the context** of the interfering node so that **the intersection becomes empty**, then the step is no longer threatening the causal link.

AI(0270)-16.14

AI(0270)-16.15

Resolving conflicts



How this is achieved

- To remove conflict, we add the $\neg Intact(Tire1)$ condition to the context of *RemoveTire(Tire1)*.
- This requires a step to provide it the context $\neg Intact(Tire1)$. This is done by **adding the conditional link** from *CheckTire(Tire1)*.
- The context propagates to the step *PutOn(Spare)*, so it gets the same context.
- The plan is now completed, as the **Finish** states has contexts encompassing all situations.
- So the three ingredients of conditional planning are **adding sensing step, generating final state with new context, and resolving conflicts by conditioning**.

AI(0270)-16.16

AI(0270)-16.17

Conditional Planning Algorithm CPOP

Start with a trivial plan. Then repeat the followings

- If all preconditions are closed, and **Finish contexts** are C_1, \dots, C_n :
 - If the contexts exhaust all scenarios, done.
 - **Generate a Finish state** with context $\neg C_1 \wedge \dots \wedge \neg C_n$.
- Pick an open precondition, say c of step S .
- Choose an action A , possibly sensing action, which **may** result in c .
- If A is a **sensing action**, modify the context of S , and propagate the context through causal and conditional links.
- while some causal link is threatened by an effect:
 - Choose to perform promotion, demotion or **conditioning**.

AI(0270)-16.18

Weakness of conditional planners, monitoring

- For conditional planning to work, **the world must be accurately represented** by the initial states and the actions.
- In many case, that means we must **make many of the knowledge conditional**, based on observations.
- The more serious problem is that **we have to plan everything** regardless of how often the event occurs.
- This usually leads to a **very slow planning**.
- Another approach: **ignore those possibilities altogether**, assuming the favourable outcome.
- The agent will have to **keep its eyes open** during the execution phase to detect **any wrong assumption**.
- Also useful if **another agent** interfere the plan of the agent.
Conditional planning assumes the sensed condition to never become false.

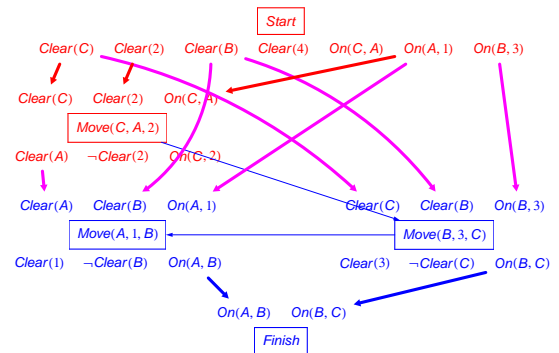
AI(0270)-16.19

When and what to check?

- Monitoring includes two parts: to detect that something is wrong, and a to replan in order to correct the error.
- **How the agent know what is wrong?** Two approaches:
 1. **Execution monitoring:** check that all the needed effects of executed steps are not violated.
How to know an effect is a needed effect? We can **check the causal link**. Basically, **if there is a causal link** from an executed action to an unexecuted action, then the effect is needed.
 2. **Action monitoring:** check that all the **pre-conditions** of each step holds before execution.
- Action monitoring looks ahead of time when the action is executed, and can thus **deduce errors earlier**. On the other hand, action monitoring is simpler.

AI(0270)-16.20

Example: execution monitoring



If we have executed $Move(C, A, 2)$, need to check $Clear(C)$, $Clear(B)$, $On(A, 1)$, $On(B, 3)$ and $Clear(A)$.

AI(0270)-16.21

What if we detect failures?

- Now that we detected an error, **what to do?**
- One simple possibility: **Just plan from the current state!**
- But planning is usually time-consuming, and **we can usually accept sub-optimal plan** when the planner is in execution.
- E.g., we might choose a point in the whole plan at which **all needed preconditions are satisfied**, and continue onwards.
- In general, we can **choose a point that is easy to reach**, and plan from the current position to the chosen point.
How to choose a point? E.g., use a heuristic of number of unmet pre-conditions, etc.

AI(0270)-16.22

Weakness of monitoring and replanning

- Can we always do monitoring and replanning instead of conditional planning?
It is much less time consuming!
- Unluckily, no. Sometimes **if we don't plan early, it is impossible to achieve the goal**.
- E.g., in the tire flattening example, we need the spare tire to be carried by the car (e.g., $Carried(Spare)$) to execute $PutOn(Spare)$.
- At the garage, a conditional planner would **anticipate that if the tire is flat and found to be non-intact**, we need a spare tire. It thus get a spare tire, **even though there is no immediate need**.
- A monitoring planner will **delay fixing the problem** until it finds a non-intact $Tire1$, when it is too late and the problem can't be fixed.

AI(0270)-16.23