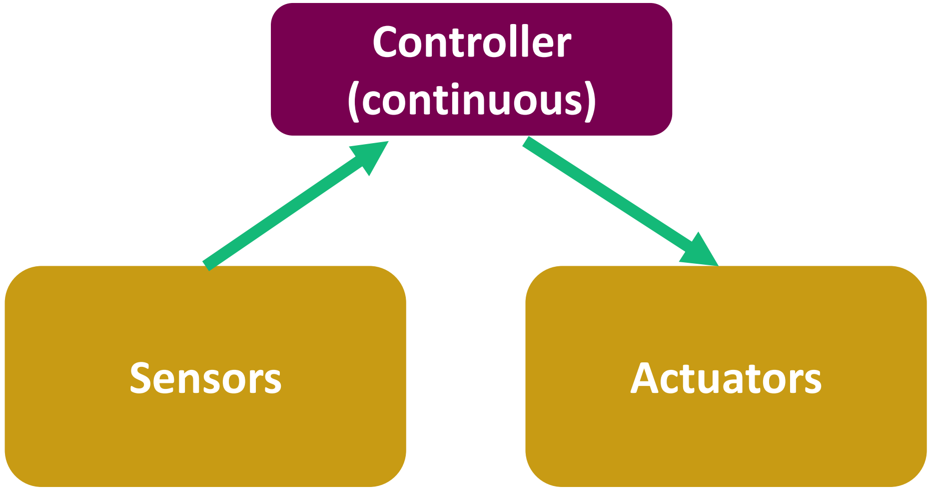
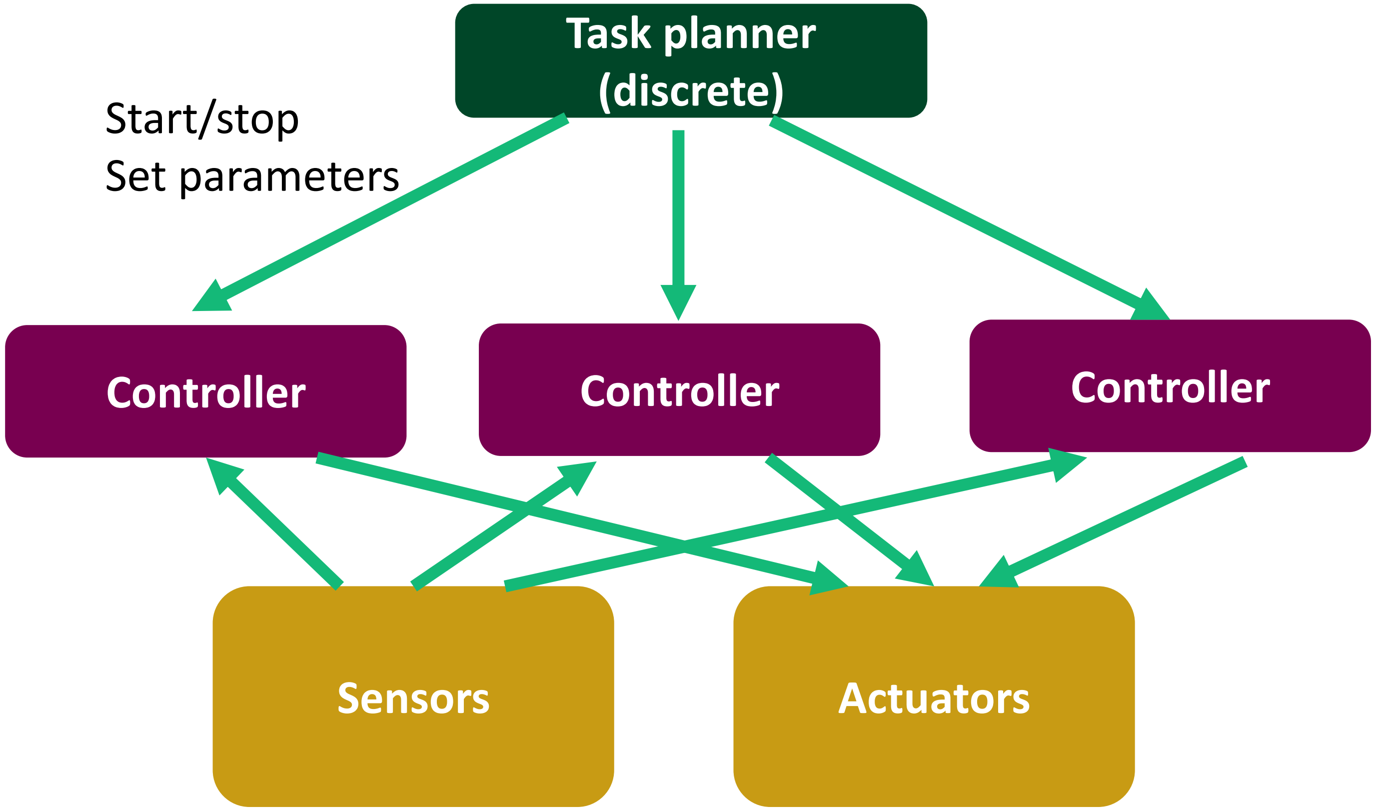
Planning Autonomous Marine Inspection Tasks Using SMT Encoding of Timelines

Bjørnar Luteberget, Synne Fossøy

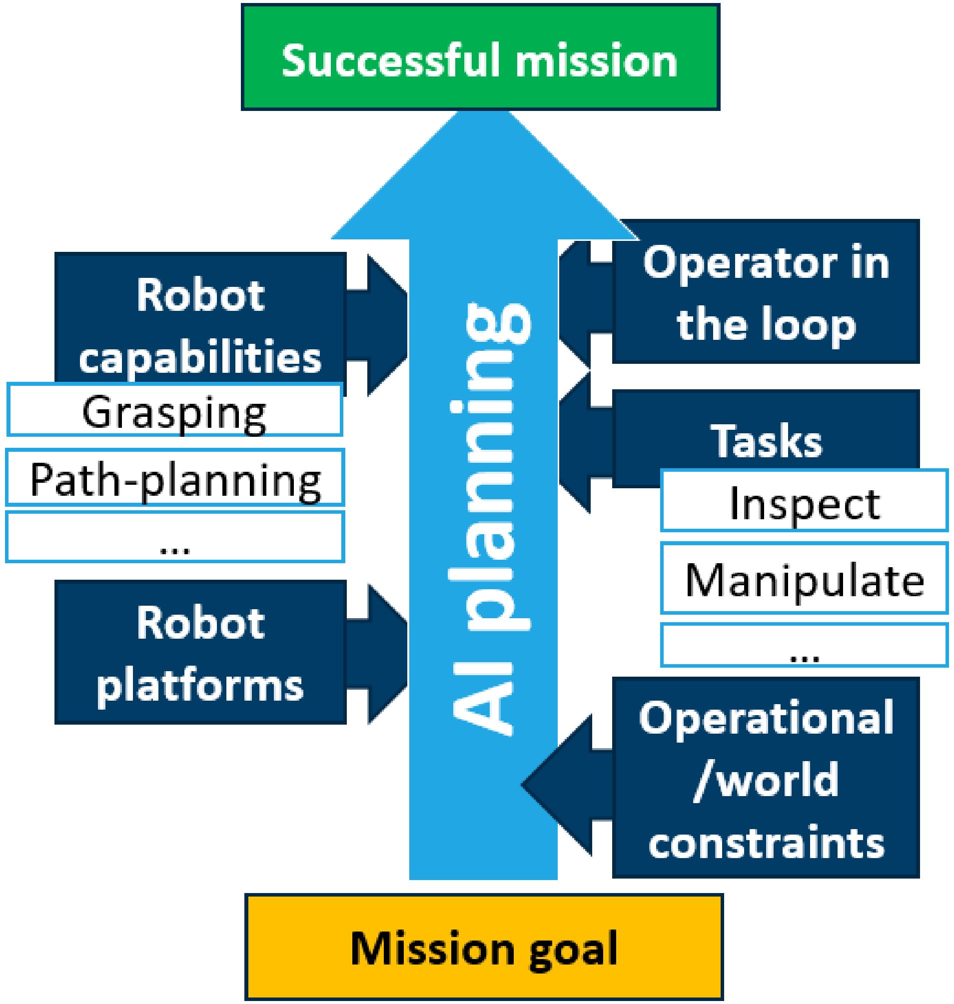
Task planning



Task planning



Automated planning and acting ≠ Path-planning



Classical planning

- **State** is Boolean variables (true/false)
  - Robot is located at point A.
- **Actions** have a precondition and an effect
  - Preconditions: conditions on the state
  - Effect: manipulations of the state
- Actions are immediate, deterministic

- Find a sequence of actions that achieves a goal

Classical planning

- State
  - robot_at_A
  - robot_at_B
  - inspection_complete
- Actions
  - move_A_B
    - Precondition: robot_at_A
    - Effect: robot_at_B
  - inspect
    - Precondition: robot_at_B
    - Effect: inspection_complete
- Initial state
  - robot_at_A = true
- Goal state
  - inspection_complete = true
- Plan (solution)
  1. move_A_B
  2. inspect

Alternative to procedural programming

```
if((goal == "inspect" || goal == "camera") && tool == "none") {
    pick_up_camera();
} else if(goal == "inspect") {
    if(tool == "camera") {
        if(location != "B") {
            go_to("B");
        } else {
            perform_inspection();
        }
    }
}
```

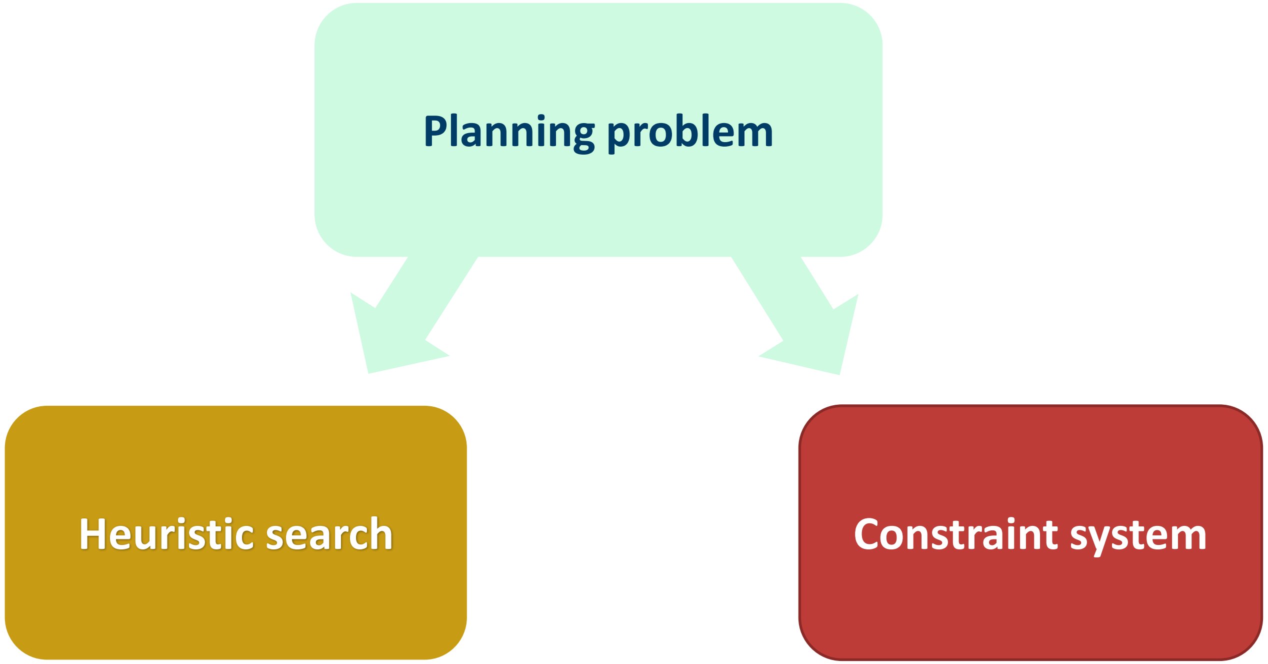
```
action move(location x, location y):
  precond: robot_at(x)
  effect:  robot_at(y)

action equip(tool t):
  effect:  have_tool(t)

action inspect(location x, tool t):
  precond: robot_at(x), have_tool(t)
  effect:  inspection_complete(x,t)

goal inspection_complete("B", "camera")
```

Algorithms: how to find the plan



- Forward/backward search
  - A* search
- Lower bound

- SAT/SMT (Boolean satisfiability) solvers
- Constraint satisfaction solvers
- Integer Programming solvers

Algorithms: how to find the plan

Planning problem

Heuristic search

- Forward/backward search
- A* search
- Lower bound

Constraint system

- SAT/SMT (Boolean satisfiability) solvers
- Constraint satisfaction solvers
- Integer Programming solvers

SAT/SMT solver

- Given a logical formula with unknowns, determine **if there exists** an **assignment** to the unknowns so that the formula becomes true.

- SAT solver: logical = propositional logic
  - Variables $\mathbf{x}, \mathbf{y}, \mathbf{z} \in$ { true, false } = $\mathbb{B}$
  - $\left(\boldsymbol{x} \Rightarrow (\neg \boldsymbol{y} \wedge \boldsymbol{z})\right) \wedge (\boldsymbol{y} \Rightarrow (\neg \boldsymbol{z} \vee \boldsymbol{x}))$

- SMT solver: logical = other types of formulas
  - Variables $x, y, z \in \mathbb{B}$, t,r $\in \mathbb{R}$
  - $\left(\boldsymbol{x} \Rightarrow (\boldsymbol{t} \geq \mathbf{5.0})\right) \wedge \left(\boldsymbol{y} \Rightarrow \left(\boldsymbol{x} \vee (\boldsymbol{r} \geq \boldsymbol{t})\right)\right)$

SAT/SMT encoding of classical planning

- Guess (!) a number N of actions that you need.
- For each state variable x, write $x_n \in \mathbb{B}$ for the variable's value in state n.
- For each action a, write $a_n \in \mathbb{B}$ indicating whether action a was the n'th action.
- Initial state: set $x_0 = T$ or $x_0 = F$
- Goal state: set $x_N = T$ or $x_N = F$
- **Only one** action at a time:
- Action **preconditions**:
- Action **effects**:
- Only effects change values:

$$x_n \in \mathbb{B}\,,\, a_n \in \mathbb{B}$$

$$x_0 = (x \in I),\;\; x_n = (x \in G)$$

$$\forall n: \forall a^1 \neq a^2: (a_n^1 \Rightarrow \neg a_n^2)$$

$$\forall n: \forall a: \forall c \in \text{prec}(a): a_n \Rightarrow c_{n-1}$$

$$\forall n: \forall a: \forall e \in \text{eff}(a): \quad a_n \Rightarrow e_n$$

$$\forall n: \forall x: (\neg x_{n-1} \wedge x_n) \Rightarrow \bigvee_{a \in \{a \mid x \in \text{eff}(a)\}} a$$

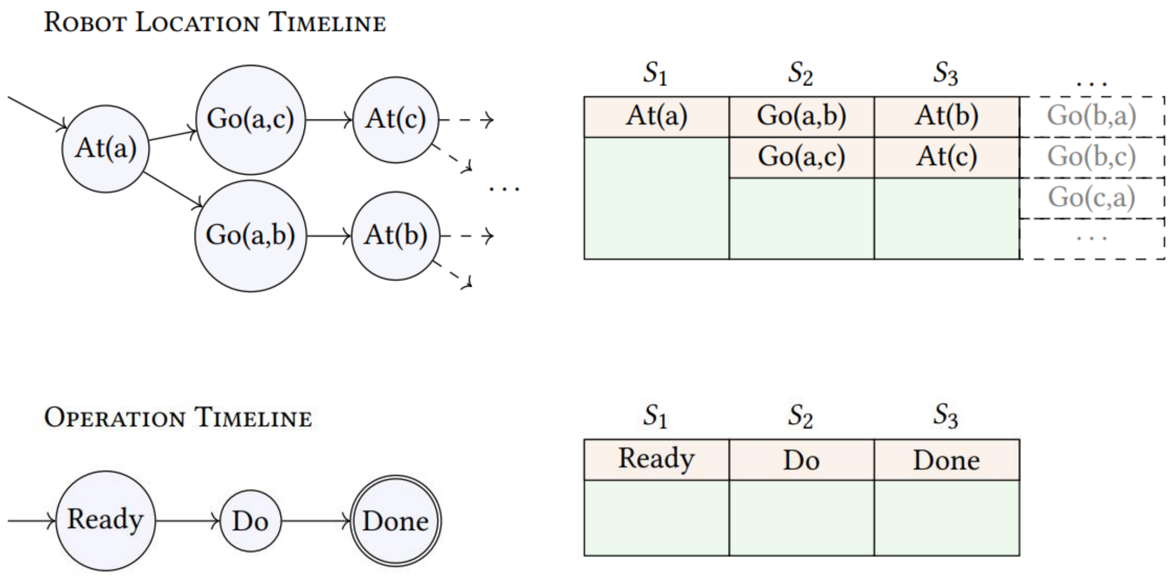
$$\forall n: \forall x: (x_{n-1} \wedge \neg x_n) \Rightarrow \bigvee_{a \in \{a \mid \neg x \in \text{eff}(a)\}} a$$

- Advantages of using SAT/SMT-based planning:
  - Very little code compared to a heuristic planner
  - SAT/SMT solvers are efficient, and have made good progress ca. 1990-now.

- Works well in practice, but there are a few pitfalls:
  - We don't know N, so need to guess $\mathrm{N} = 1, 2, 3, 4, \ldots$
  - Increasing N, all the states and actions are copied, creates large formula.
  - Number of input actions and number of plan steps can be correlated, giving quadratic number of variables.

Timelines: independent components

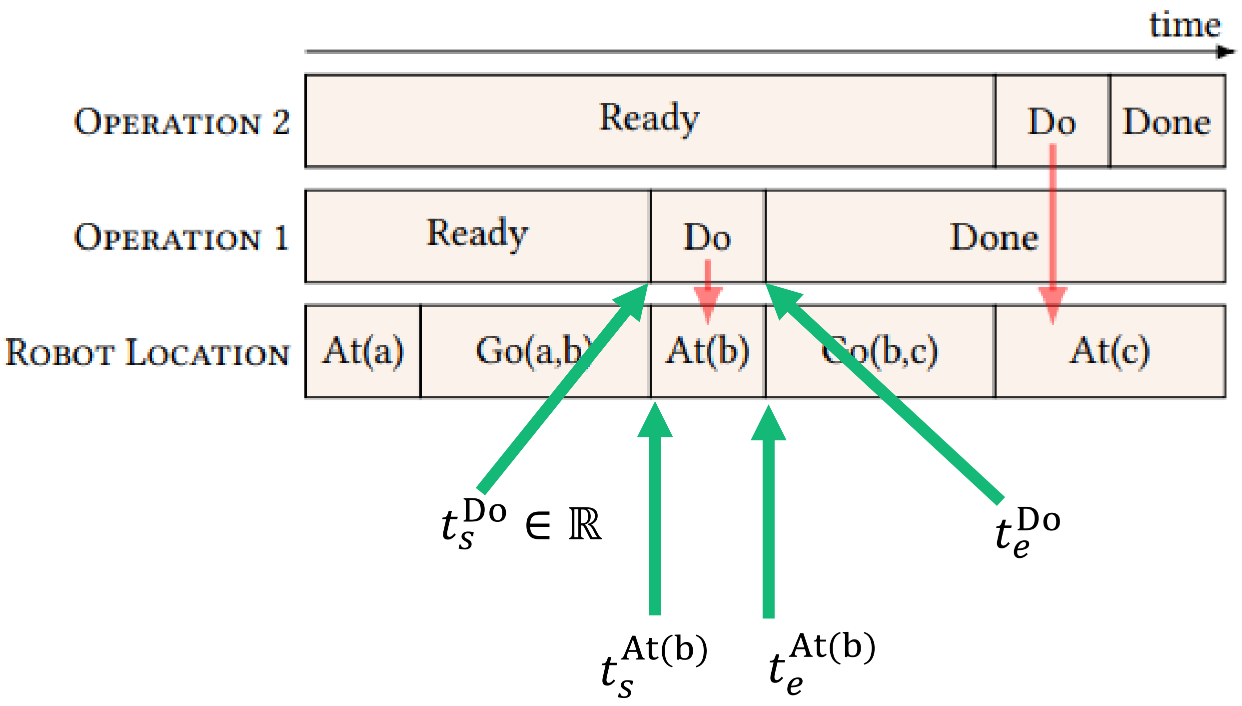
- To help with the number of copies of state variables, we can split the world into **components**, which each have their own **timeline.**
- Some timelines have a **known** number of states.
- Each transition **always changes** the component.



Timelines: synchronization conditions

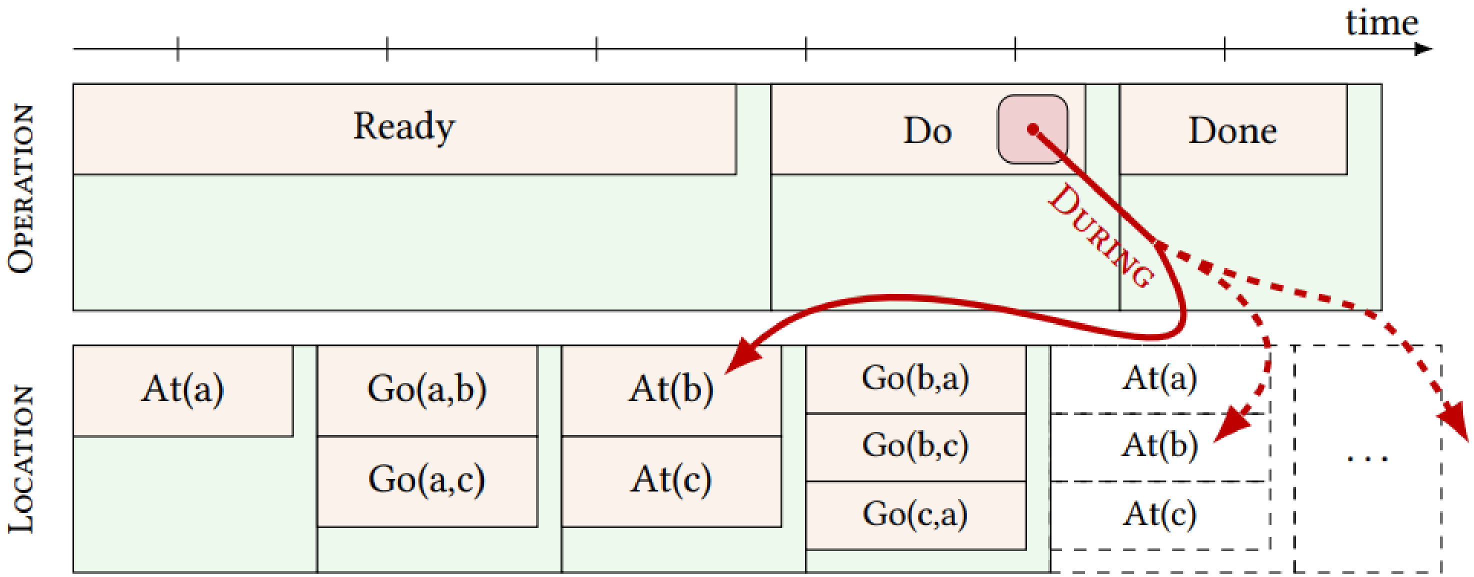
- Components can affect each other
- Add time variables $t_s, t_e \in \mathbb{R}$
- Add synchronization conditions on the time variables, for example *during*:

$$\forall n \in [0, N^{\mathrm{Op}}]:\ v_{\mathrm{n}}^{\mathrm{Do}} \Rightarrow \bigvee_{v_i^{\mathrm{At(b)}}} (v_i^{\mathrm{At(b)}} \wedge \left(t_s^{\mathrm{At(b)}} \le t_s^{\mathrm{Do}}\right) \wedge \left(t_e^{\mathrm{Do}} \le t_e^{\mathrm{At(b)}}\right)$$



Refining the number of steps

- Now we need to guess $\mathrm{N} = 1, 2, 3, 4, \ldots$ for every timeline!
- Our contribution: use the SMT solver's unsatisfiable core to tell *which* timelines need to be expanded.
- The trick:
  - When we have a disjunction: $a_1 \vee \cdots \vee a_{i-1} \vee a_i \vee \cdots \vee a_n$
  - Add a new variable x and split: $(a_1 \vee \cdots \vee a_{i-1} \vee x) \wedge (\neg x \vee a_i \vee \cdots \vee a_n)$
  - Leave out the second half: $a_1 \vee \cdots \vee a_{i-1} \vee x$
  - Tell the solver to assume $\neg x$.
- Unsatisfiable core : when the formula is unsatisfiable, the solver returns a **subset of assumptions** that are infeasible.

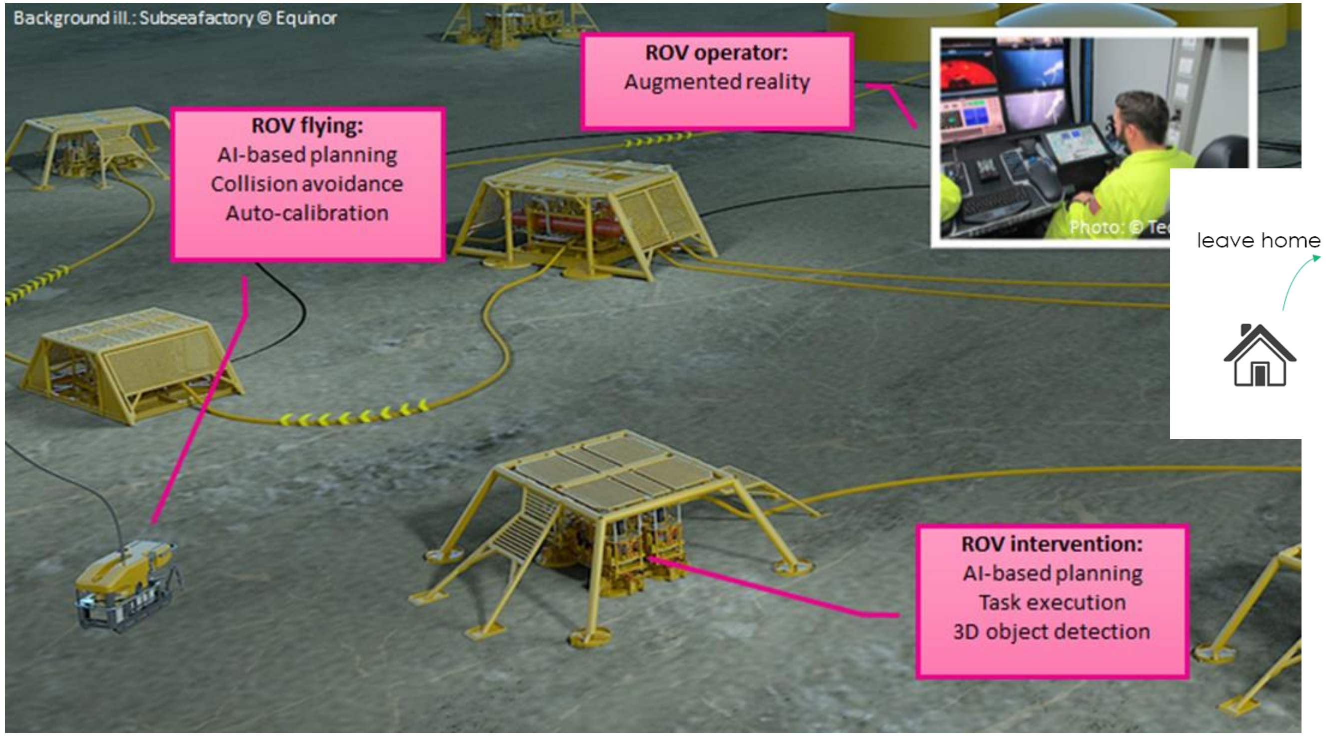


$$v_2^{\mathrm{Do}} \Rightarrow v_3^{\mathrm{At(b)}} \vee v_5^{\mathrm{At(b)}} \vee v_7^{\mathrm{At(b)}} \vee \cdots$$

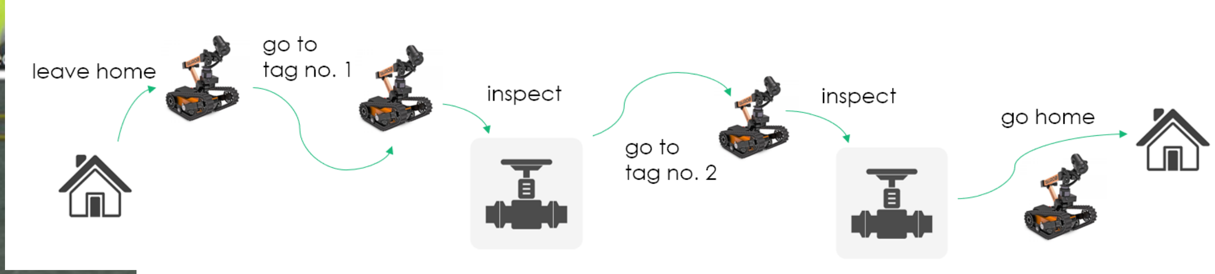
$$v_2^{\mathrm{Do}} \Rightarrow v_3^{\mathrm{At(b)}} \vee x \qquad [\text{assume } \neg x]$$

Application context

- **SEAVENTION project**: Autonomous subsea intervention - empowered by people and AI.
- **ROBPLAN project**: Autonomous robot missions with AI-based planning and acting



- **Challenges:**
  - Plan failure
  - Change in the goals to be achieved by the plan
  - Unexpected events



SINTEF

Future work

- Performance on modified inputs:
  - Failed actions.
  - New goals (priorities).
  - Exogenous events.
- User-friendly modelling.

Algorithms and theory → ← **Applications**

SINTEF

Teknologi for et bedre samfunn

Automated task planning

- Describe the world (**state**) and how we can manipulate it (**actions**).
- Describe the current state of the world (**initial**) and how you want it to look (**goal**).
- Find **plan** that leads from **initial** to **goal**.

Performance evaluation

- Manually translated benchmark instances show promising performance
- Note that our planner does not yet support a standard input format, so comparisons are affected by the translation

Instance	oRatio	our planner
cc, 1 plate, 5 dishes	0.109	0.026
cc, 1 plate, 50 dishes	10.682	0.294
cc, 1 plate, 100 dishes	T/O	1.880
cc, 2 plates, 50 dishes	11.185	0.298
cc, 2 plates, 100 dishes	T/O	1.860
tms, 1 oven, 2 items	0.073	0.014
tms, 2 ovens, 4 items	0.849	0.021
tms, 2 ovens, 6 items	5.921	0.037
tms, 4 ovens, 6 items	15.234	0.029
tms, 5 ovens, 10 items	T/O	0.158
goac, 5 locs., 3 time windows	0.252	0.056
goac, 5 locs., 5 time windows	0.300	0.027
goac, 7 locs., 3 time windows	0.469	0.049
goac, 7 locs., 5 time windows	0.973	0.045
goac, 9 locs., 3 time windows	1.589	0.142
goac, 9 locs., 5 time windows	1.420	0.154

Instance	FAPE	our planner
Airports-2, 1 vehicle, 17 locations	10.4	0.4
Airports-4, 1 vehicle, 40 locations	73.8	0.2
Airports-6, 2 vehicles, 40 locations	T/O	1.5
Airports-8, 3 vehicles, 40 locations	T/O	18.7
Airports-10, 1 vehicle, 44 locations	142.9	0.2
Satellite-2, 2 tools, 8 directions	4.3	24.5
Satellite-4, 3 tools, 10 directions	6.2	0.9
Satellite-6, 5 tools, 11 directions	6.7	1.0
Satellite-8, 10 tools, 15 directions	41.0	19.4
Satellite-10, 11 tools, 17 directions	T/O	17.1
Pipesworld-2, 2 pipes, 6 deliveries	32.2	0.7
Pipesworld-4, 2 pipes, 8 deliveries	9.4	0.4
Pipesworld-6, 2 pipes, 10 deliveies	T/O	0.8
Pipesworld-8, 2 pipes, 12 deliveries	T/O	6.1
Pipesworld-10, 2 pipes, 14 deliveries	T/O	3.8

Application context

- **ROBPLAN project**: Autonomous robot missions with AI-based planning and acting

  - **Automatically generate sequence of waypoints/tags** to visit by a robot – based on list from user.
  - **Plan failure / resources (battery management)**: **Battery depletion** go faster than expected.
  - **Goal emergence**: New waypoints can be added by human user or "sensor data analysis module"
  - **Waypoint replanning.** Tag/waypoints can be blocked.