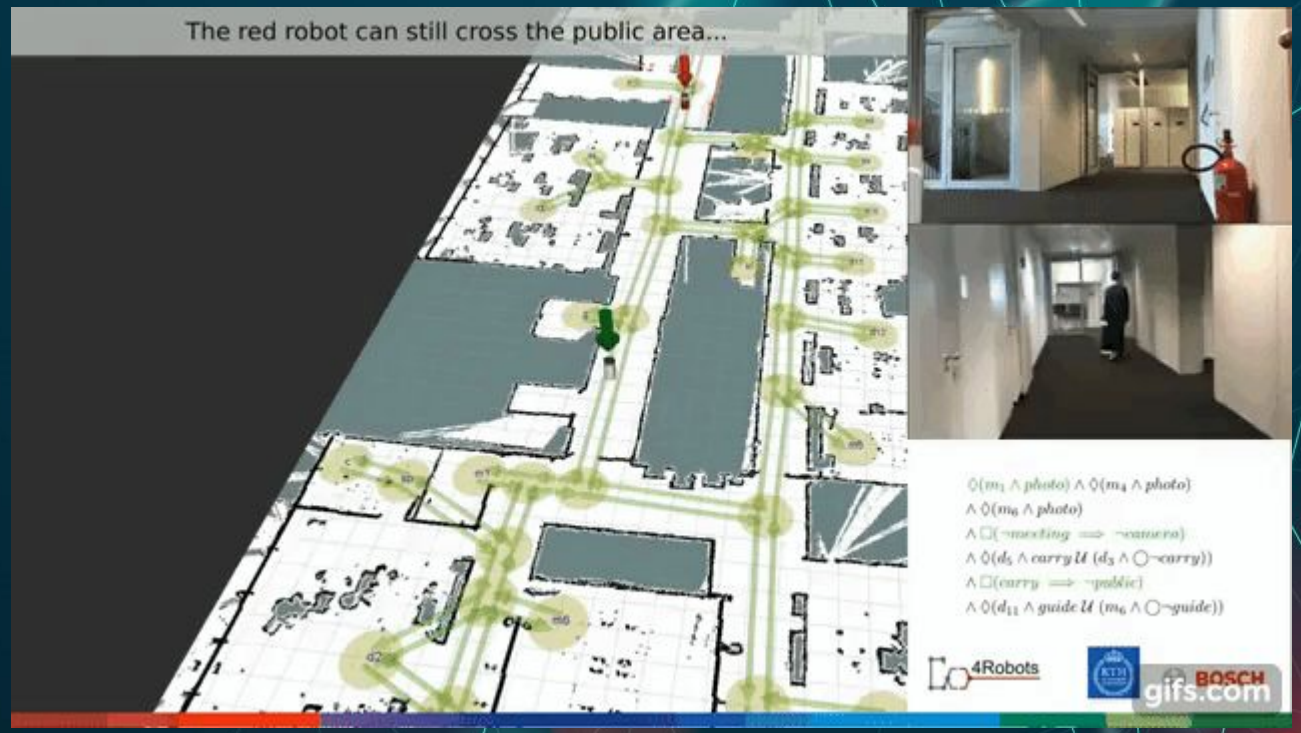
SIMULTANEOUS TASK ALLOCATION AND PLANNING UNDER UNCERTAINTY

Presented by:

1. Abdolhadi Rezaei
2. Maryam Bandali
3. Oluwatoyin Yetunde Sanni
4. Mohammed Sukhsarwala



PRESENTATION BREAKDOWN

01 INTRODUCTION & PRELIMINARIES

02 STAPU PROBLEM FORMULATION

03 STAPU WITH REALLOCATION

04 EVALUATION & SCENARIO

Application of allocating a collection of independent tasks to a team of Robots

Most existing approaches consider the separation of task allocation(TA) and planning

Advantages:

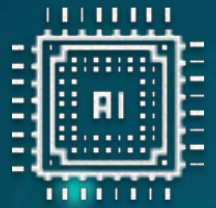
Complexity Reduction

Increase the efficiency of task allocation computation

Disadvantages:

Unable to consider TA in planning process

Simultaneous Task Allocation and Planning (STAP)



Formalization of Simultaneous Task Allocation & Planning Under Uncertainty (STAPU).

Adaptation of Techniques from formal verification of probabilistic systems to solve this problem.

An extension of approach to address reallocation when failure comes up.

INTRODUCTION

- Individual robots' capabilities and environments
  - Markov decision processes (MDPs)
- Set of task Specification
  - Linear temporal logic (Co-safe)
  - Specification of safety constraints (safe LTL)
- Generating multi robot policies
  - Basic Task allocation and Planning
    - Team MDP
    - Iteratively improvement of the policies in case of failures

MARKOV DECISION PROCESS

- We use Markov decision processes (MDPs) to model the evolution of robots and their environment. An MDP is a tuple:

$$\mathcal{M} = \langle S, \bar{s}, A, \delta_M, AP, Lab \rangle$$

- S is a finite set of states;
- $\bar{s} \in S$ is the initial state;
- A is a finite set of actions;
- $\delta_M : S \times A \times S \rightarrow [0,1]$ is a probabilistic transition function, where $\sum_{s' \in S} \delta_M(s, a, s') \in \{0, 1\}$, for all $s, s' \in S, a \in A$;
- AP is a set of atomic propositions;
- Lab: $S \rightarrow 2^{AP}$ is a labelling function, such that $p \in Lab(s)$ iff p is true in $s \in S$.

MARKOV DECISION PROCESS

- Set of enabled actions in s ∈ S as:
  - A_s ∈ {a ∈ A | $\delta_M(s, a, s') > 0$ for some s' ∈ S
- An infinite path through an MDP is a sequence:
  - $\sigma = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ where $\delta_M(s_i, a_0, s_{i+1}) > 0$ for all i ∈ $\mathbb{N}$
- A finite path
  - $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} s_n$ is a prefix of an infinite path
- The choice of action MDP is made by policy,
- A policy is a function π from finite paths of M to actions in A such that, for any finite path σ ending in state s_n, $\pi(\sigma) \in A_{s_n}$

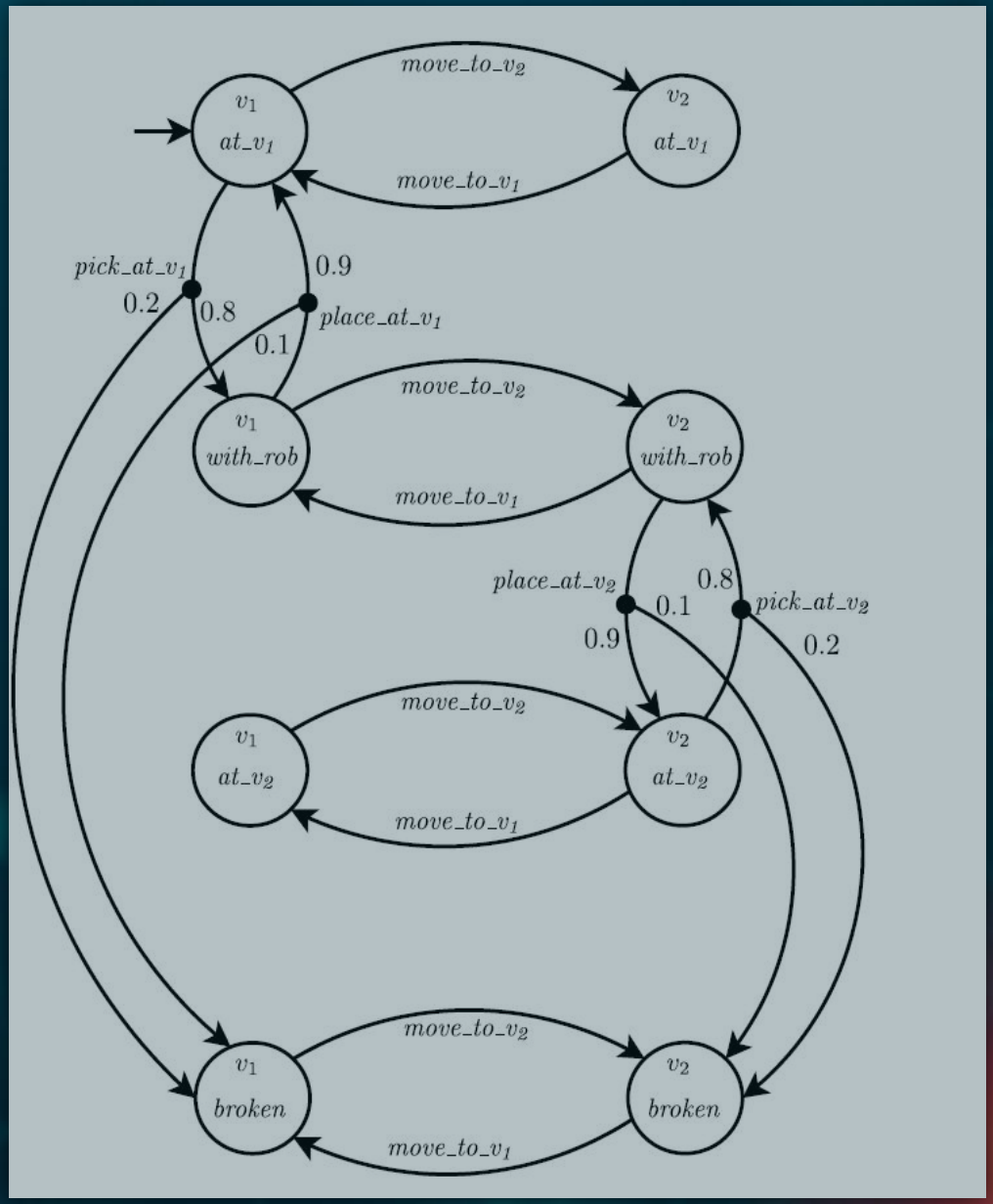
EXAMPLE

Consider a robot that can move between two locations v1 and v2.

A water bottle is placed in location v1 and the robot can pick it up and place it back in both locations.

We assume a probability 0.8 of successfully picking up the object without breaking it, a probability 0.9 of successfully placing the object without breaking it, and that the robot can navigate between locations v1 and v2 without failing.

We can model this example as a factored MDP, with X = {robot loc; obj state}.



Linear Temporal Logic

- Linear temporal logic (LTL) is an extension of propositional logic which allows reasoning about infinite sequences of states.

- LTL formulas φ over atomic propositions AP are defined using the following grammar:

φ ::= true | p | $\neg\varphi$ | $\varphi \wedge \varphi$ | $\mathrm{X}\varphi$ | $\varphi\, U\varphi$. Where p ∈ AP.

- The “next” operator X meaning that the formula it precedes will be true in the next state.

- The U operator, “until”, meaning that its second argument will eventually become true in some state, and the first argument will be continuously true until this point.

- “eventually” operator Fφ, which requires that φ is satisfied in some future state
- The “always” operator G φ, which requires φ to be satisfied in all future states

SAFE AND CO-SAFE NOTIONS IN LTL

- We are interested in specifying behaviors that occur within finite time. So we use two subsets of LTL which are safe and co-safe LTL
  - Based on notion of bad and good prefix

- Bad prefix:

  In order to formally define what safety properties are, we refer to computations of a nonterminating system as infinite words over an alphabet Σ . Typically, Σ = 2^AP, where AP is the set of the system's atomic propositions. Consider a language L of infinite words over Σ. A finite word x over Σ is a bad prefix for L, if for all infinite words y over Σ , the concatenation x . y of x and y is not in L.

- Good prefix

  Let φ be an LTL formula and σ = σ_0 σ_1... $\in$ $[\![[\![(2)\!]$ ^AP)$]\!]$ ^ωsuch that $\sigma \vDash \varphi$. We say that σ has a good prefix for φ if there exists n $\in \mathbb{N}$ for which the truncated finite sequence $\sigma|n|$ = σ_0 σ_1... σ_n is such that for every σ^'$\in$ $[\![[\![(2)\!]$ ^AP)$]\!]$ ^ω the concatenation $\sigma|n|$. σ^' $\vDash \varphi$.

SAFE AND CO-SAFE NOTIONS IN LTL

- Syntactic Restriction. we assume a syntactic restriction for safe and co-safe LTL. We assume that all formulas are in positive normal form (negation can only appear next to atomic propositions).
- Syntactically, safe LTL is the set of formulas for which only the G and X temporal operators occur
- Syntactically , co-safe LTL is the set of formulas for which only the X, F and U temporal operators occur

SAFE PROPERTY EXAMPLE

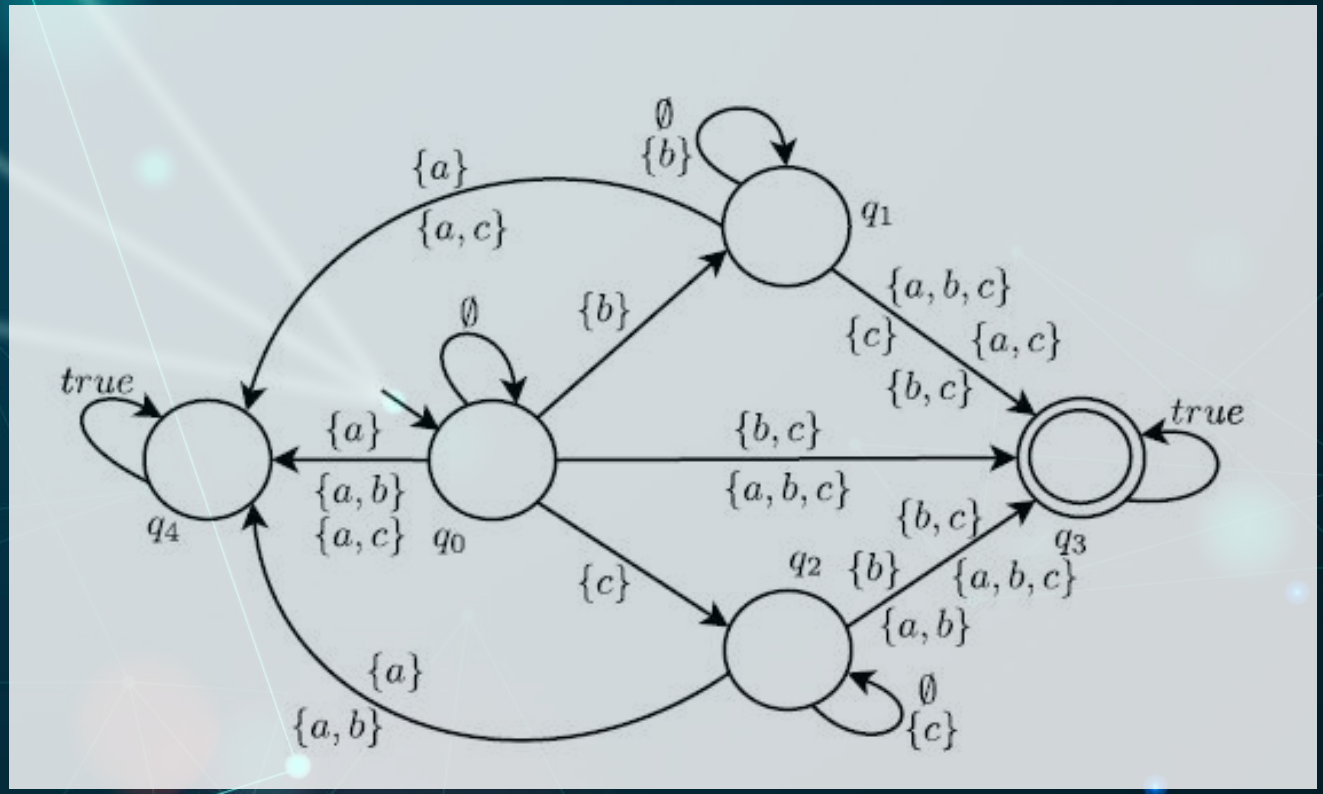
1.

A language L is a safety language if every word not in L has a finite bad prefix.

For example, if $\Sigma = \{0, 1\}$, then L = $\{0^\omega, 1^\omega\}$ is a safety language. To see this, note that every word not in L contains either the sequence 01 or the sequence 10, and a prefix that ends in one of these sequences cannot be extended to a word in L.

SAFE PROPERTY EXAMPLE

Consider the syntactically co-safe LTL specification φ = ((¬a) U b) ∧ ((¬a) U c).



DETERMINISTIC FINITE AUTOMATON (DFA)

- A Deterministic Finite Automaton (DFA) is a tuple:

$$< Q, \bar{q}, Q_F, \Sigma, \delta_{\mathrm{A}} >$$

- Q is a finite set of states;
- $\bar{q} \in$ Q is the initial state;
- $Q_F \in$ Q is the set of accepting states;
- 2^{AP} is the alphabet;
- $\delta_{A_\varphi}: Q \times 2^{AP} \rightarrow Q$ is a transition function

- For any (co-)safe LTL formula φ written over AP, we can build a DFA, A_φ
- As a consequence, in this paper, LTL formula φ could be safe or co-safe.

OPTIMAL POLICIES FOR (CO)-SAFE SPECIFICATIONS

- Suppose we have an MDP $\mathcal{M}$ and a DFA A_φ obtained from co-safe LTL formula φ:

$$\mathcal{M} = < \mathrm{S}, \bar{s}, \mathrm{A}, \delta_M, \mathrm{AP}, \mathrm{Lab} >$$
$$A_\varphi = < Q, \bar{q}, \{q_F\}, 2^{AP}, \delta_{A_\varphi} >$$

- We can build a product MDP

$$M_\varphi = \mathrm{M} \otimes A_\varphi = < S_\varphi, \bar{s}_\varphi, \mathrm{A}, \delta_{\mathrm{M}_\varphi}, \mathrm{AP}, Lab_\varphi >$$

- $S_\varphi = \mathrm{S} \times Q$,
- $\bar{s}_\varphi = (\bar{s}, \delta_{A_\varphi}(\bar{q}, \mathrm{Lab}(\bar{s})))$
- $\delta_{M_\varphi}((\mathrm{s},\mathrm{q}), \mathrm{a}, (\mathrm{s}',\mathrm{q}')) = \begin{cases} \delta_M(\mathrm{s}, \mathrm{a}, \mathrm{s}') & \text{if } \mathrm{q}' = \delta_{A_\varphi}(\mathrm{q}, \mathrm{Lab}(\mathrm{s}')) \\ 0 & \text{otherwise,} \end{cases}$
- $Lab_\varphi((\mathrm{s},\mathrm{q})) = \mathrm{Lab}(\mathrm{s})$
- M_φ behaves like the original MDP M, but is augmented with information about the satisfaction of φ. What is Probabilistic Satisfaction?

PROBABILISTIC SATISFACTION

- Let

$$\mathcal{M} = < \mathrm{S}, \bar{s}, \mathrm{A}, \delta_M, \mathrm{AP}, \mathrm{Lab} >$$

Be an MDP, and φ an LTL formula over AP. The maximum probability of satisfying φ, along with the corresponding optimal policy π^* is as follows:

$$Pr_{M,s}^{max}(\varphi) = Pr_{M,s}^{max}(\{\sigma \in IPath_{M,s} \mid \sigma \vDash \varphi\})$$

$$\pi^* = \arg\max_{\pi} Pr_{M,s}^{\pi}(\varphi)$$

- this problem can be reduced to a probabilistic reachability problem on an MDP obtained by the product of M with a DFA, What is probabilistic reachability?

PROBABILISTIC REACHABILITY

- Let

$$\mathcal{M} = < \mathrm{S}, \bar{s}, \mathrm{A}, \delta_M, \mathrm{AP}, \mathrm{Lab} >$$

and G ⊆ S, we define $reach_G \subseteq IPath_{M,s}$ as:

$$reach_G = \{(s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots) \mid s_i \in \mathrm{G} \text{ for some } i \in \mathbb{N}\}$$

$$Pr_{M,s}^{max}(reach_G)$$

$$\pi^* = \arg\max_{\pi} Pr_{M,s}^{\pi}(reach_G)$$

STAPU - PROBLEM FORMULATION

ROBOTS

$R = \{r_1, \ldots, r_n\}$ is the set of robot(agents)

The operation of each agent r_i is modelled by an MDP($\mathcal{M}_i$).

Probabilities in each MDP may represent either uncertainty in its environment or the possibility of failure.

$M = (\Phi, \varphi_{safe})$ where

- $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ set of co-safe LTL task specifications
- φ_{safe} is a single safety specification.

MISSION

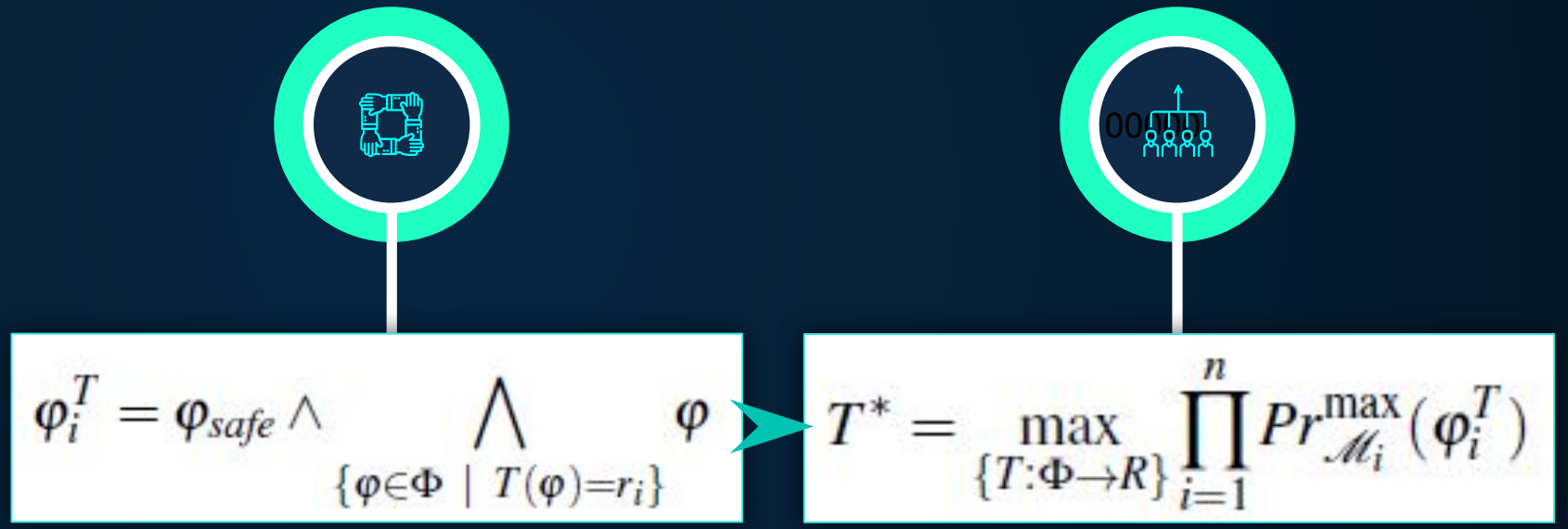
LTL

In particular, the LTL specifications here are decomposed such that they are independent.

Given R and M from previous slides, we then define STAPU problem as a Task allocation mapping in a joint manner such that: $T^*: \phi \square R$

We describe the LTL specification φ_i^T as the conjunction of tasks for each robot, given task allocation T.

STAPU - PROBLEM FORMULATION



STAPU - PROBLEM FORMULATION

OPTIMAL POLICIES

As we have observed, solving a STAPU problem is effectively finding a joint policy such that we have a task allocations mapping, and actions to each robot as seen with $\mathbf{T}^*$

INDEPENDENT TASKS

We consider each robot taking turns independently when solving a single STAPU problem because the tasks are executed in a parallel manner.

LOCAL MDP

In order to find an optimal policy for each robot, it is assumed that tasks have no interdependence and can each be completed by a single robot. Hence, this brings us to a local MDP.

LOCAL MDP

Each MDP encodes the dynamics of each robot, its tasks and the extent of task completion. A product MDP with a separate DFA for each LTL formula is defined as

$$\square_i^M = \square i \otimes A\varphi_1 \otimes \ldots\ldots \otimes A\varphi\square \otimes A\varphi_{safe}$$

$$\Downarrow$$

$$\square_i^M = \langle S_i^M, \overline{s_i^M}, A_i, \delta_{\mathcal{M}_i^M}, AP, Lab \rangle$$

Given $Q\varphi i$ as the DFA for each task, the state space S_i^M of the local MDP $\square_i^M$ is denoted as

$$S_i^M = S_i \times Q\varphi_1 \times \ldots \times Q\varphi_n \times Q\varphi_{safe}$$

which allows us to keep track of the state of satisfaction for each part of the mission specification separately.

TEAM MDP

The team MDP $\mathcal{G}$ is the union of the *n* local product MDPs

$\square_{i}^{M} \ldots \square_{n}^{M}$

We construct a team MDP that represent each robot sequentially.

MDP $\mathcal{G}$ is a tuple

$$\langle S_{\mathcal{G}}, \overline{s_{\mathcal{G}}}, A_{\mathcal{G}}, \delta_{\mathcal{G}}, AP_{\mathcal{G}}, Lab_{\mathcal{G}} \rangle$$

- S_G keeps track of the current robot and its current state within its local MDP.
- $\overline{s_{\mathcal{G}}} = (1, \overline{s_1^M})$, r_1 in its initial state

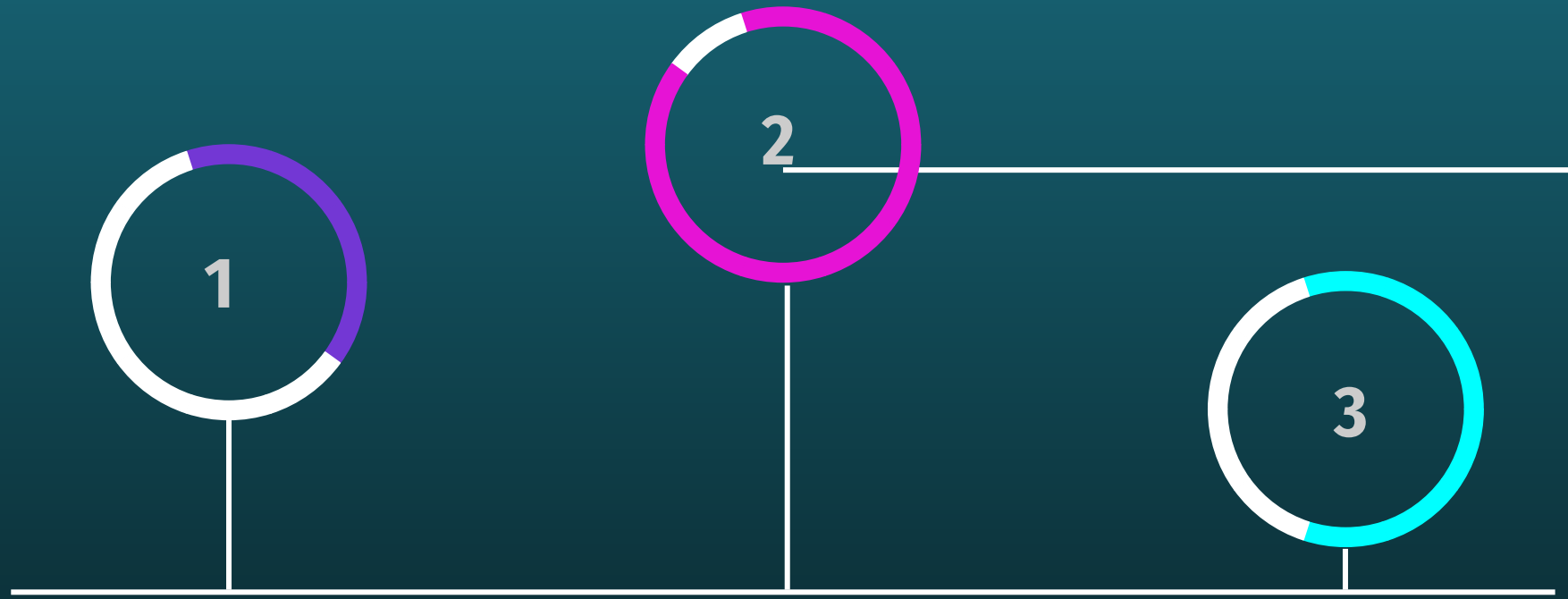
- $A_G = \{\zeta\} \cup \cup_{i=1}^{n} A_i$ is the special switch transition ζ used for tasks allocations from r_i to r_{i+1} UNION the action space that contains individual robot actions

TEAM MDP

For an action space $\cup^{n}_{i=1} A_i$, the corresponding local product MDP $\square_i^M$ transition function is mapped this way:

$\delta_G((i,s_i,), a, (i,s'_i,)) = \delta_{\mathcal{M}^M}(s_i, a, s'_i,)$.

For an action $a = \zeta$, the transition function updates the system to plan for the next robot $\delta_{\mathcal{G}}((i,s), \zeta, (j,s')) = 1$ if some conditions hold.



j = 1 + (i % n)

the robots are connected in a ring topology

The state of all the tasks is preserved and we do not switch during the execution of a task.

s' corresponds to an initial state of robot j. i.e s'= $(\overline{s_j}, q)$ where represents the LTL specifications.

For other pairs of states where $\delta G((i,s), \zeta, (j,s')) = 0$, we omit the propositions $AP_{\mathcal{G}}$ and labelling function $Lab_{\mathcal{G}}$

PROBLEMS WITH STAPU

Even though the tasks in team MDP are addressed in a sequential order, the generated policies are to be executed in parallel by the team. This leads us to AMBIGUITIES at execution time. How?

Because they run in parallel, it's hard for r_{i+1} to know how the 1,...i's policies will evolve. This is a partial observability scenario.

OLUWATOYIN SANNI

THE SOLUTION

How do we solve this?

With the use of a particular class of single robot MDPs where this uniqueness of a switch transition occurs. MDPs with a designated failure state.

$$\delta(s,a,s') = 1$$

$$\delta(s,a,s') + \delta(s,a,s_{\perp}) = 1$$

where actions either move to a next state s' with probability x or fail with probability 1 - x.

OLUWATOYIN SANNI

What happens when a task fails to be completed by a Robot?

Reallocation to another member of the TEAM MDP is the approach.

How is the Reallocation DONE??

A new switch transition is added which are used to perform this reallocation. We build a joint model representing the synchronized evolution of the system just under the computed STAPU policies, i.e., we build the synchronized multi-robot policy for the STAPU solution, where, at each state, each robot executes the action corresponding to its policy in a joint fashion.

STAPU WITH REALLOCATION

BREAKDOWN

This new switch transition outcome is used to describe the states each robot might be at the point in time when failure occurred. However, the need to synchronize in the TEAM MDP is crucial.

At each state s, each robot r(i) executes the action corresponding to its policy in a joint fashion. This ri model has a set of reallocation states corresponding to situations where some robots have previously failed. We then choose one of these reallocation states as the initial state of a new STAPU instance with a switch transition.

STAPU WITH REALLOCATION

STAPU WITH REALLOCATION

Theoretical Approach to the Breakdown

Let Sr= (s_1,.....,s□, qφ1,...,qφm,qφsafe) be an example of a reallocation state in the synchronized multi-robot policy, where si is the failure state to be addressed.

A modification was made to the previous team MDP as shown below

Previous Team MDP

MDP $\mathcal{G} = \langle S_{\mathcal{G}}, \overline{s_{\mathcal{G}}}, A_{\mathcal{G}}, \delta_{\mathcal{G}}, AP_{\mathcal{G}}, Lab_{\mathcal{G}} \rangle$

where $\overline{s_{\mathcal{G}}} = (1, \overline{s_1^M})$. which represents the r_1 in its initial state. Here, the switch transition point to the initial state of the next robot.

Modified Team MDP

MDP $\mathcal{G} = \langle S_{\mathcal{G}}, \overline{s_{\mathcal{G}}}, A_{\mathcal{G}}, \delta_{\mathcal{G}}, AP_{\mathcal{G}}, Lab_{\mathcal{G}} \rangle$

where $\overline{s_{\mathcal{G}}}$ = maps to Sr above. Here, the switch transition point to the current state of the next robot in the reallocation state s1 + (i mod n)

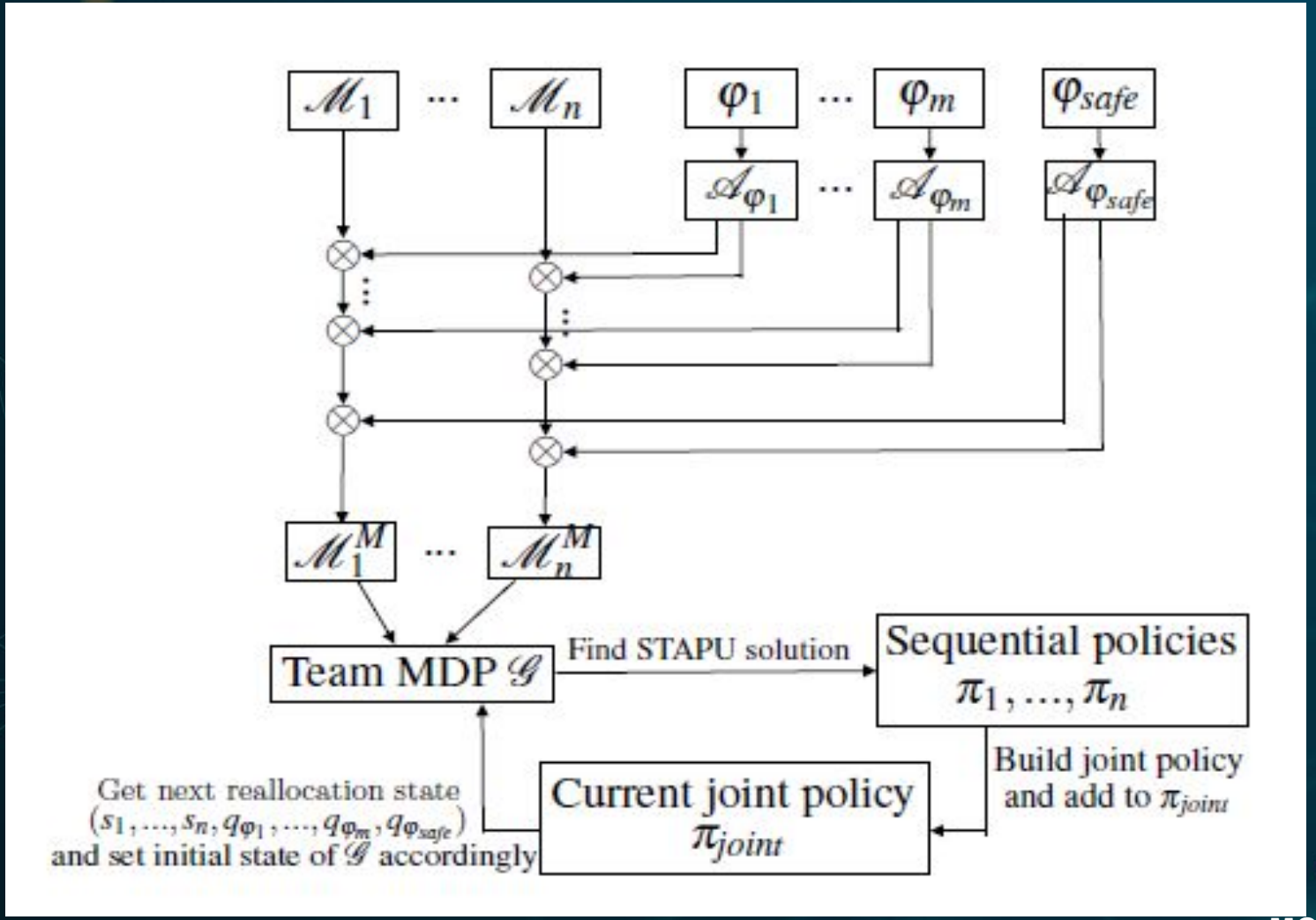
STAPU WITH REALLOCATION

By solving this new STAPU instance, we effectively find a reallocation policy from the chosen reallocation state. We can then use the reallocation policy to continue building the synchronized multi-robot policy from Sr= (s_1,.....,s□, qφ1,..., qφm,qφsafe) and then choose a new reallocation state to address. We choose the next reallocation state to address in decreasing order of reachability probability, i.e., we start by addressing the most probable reallocation states.

When all reallocation states have been addressed, we terminate the procedure. Note that because we choose the next reallocation state to address in decreasing order of reachability probability the algorithm is anytime, in that it can be terminated at any point and provide a (incomplete) solution, and the longer it runs the more cases it will cover.

We finish by noting that this approach can be extended to also include expected team sum of costs minimization by defining a cost structure (representing navigation duration for example) for each robot and using nested value iteration to generate policies.

STAPU WITH REALLOCATION DIAGRAM ILLUSTRATION



STAPU EVALUATION

The paper implemented STAPU with reallocation in the PRISM model checker, which supports solving MDPs for LTL properties. In this example, they had 30 states, 2 robots with an initial 5 failure points per robot model. The mission $M = (\phi, \varphi_{safe})$ where

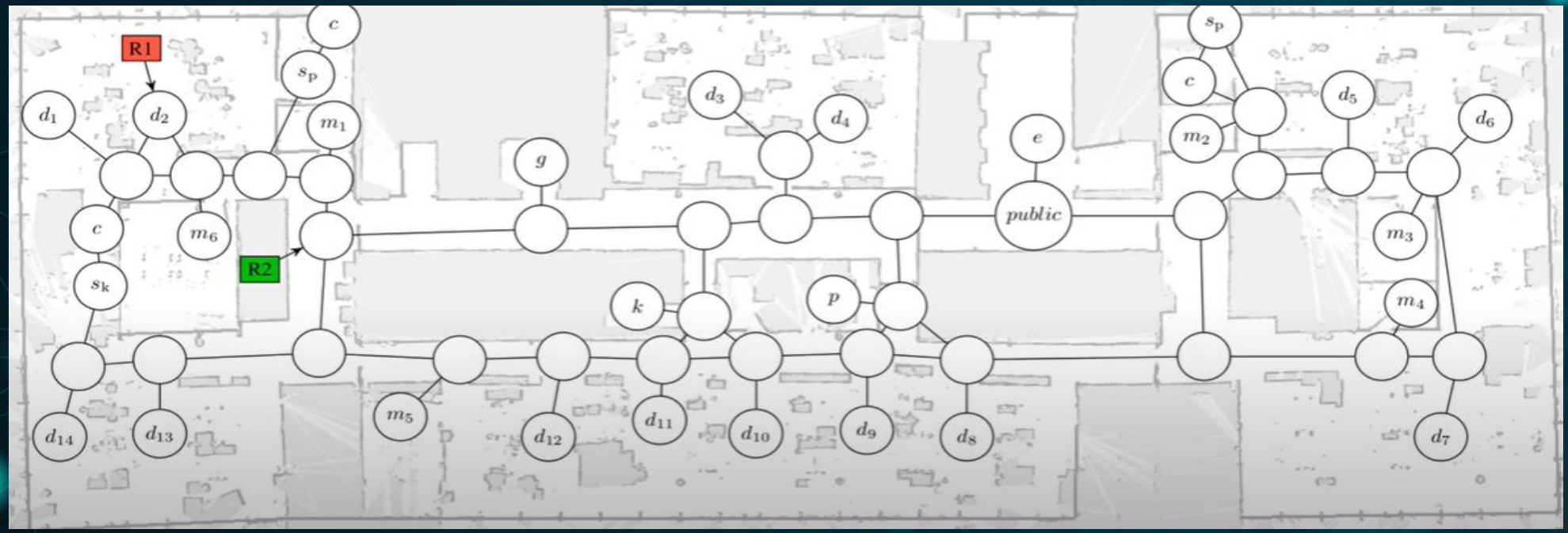
- $\phi = \{\varphi_1,..,\varphi_{\square}\}$
- $\varphi_l = \Diamond p_i$
- $\varphi_{safe} \boxplus \neg p$ and $p_i \in AP$

They evaluated the scalability of the approach with **respect to the number of robots and tasks** in the model, and the time of completion. As observed, STAPU had lesser model sizes and computation time, compare to Value Iteration algorithm over full multi-agent MDP(MAMDP.)

Tasks	STAPU with reallocation			VI over MAMDP model		
	Model Size		Time	Model Size		Time
	$\|S_G\|$	$\|\delta_G\|$	(s)	$\|S_G\|$	$\|\delta_G\|$	(s)
3	480	1442	0.09	7200	64800	1.17
5	1920	5788	0.12	28800	259200	11.96
7	7680	23226	0.51	115200	1036800	195.55
9	30720	93176	4.40	460800	4147200	3218.47

SCENARIO

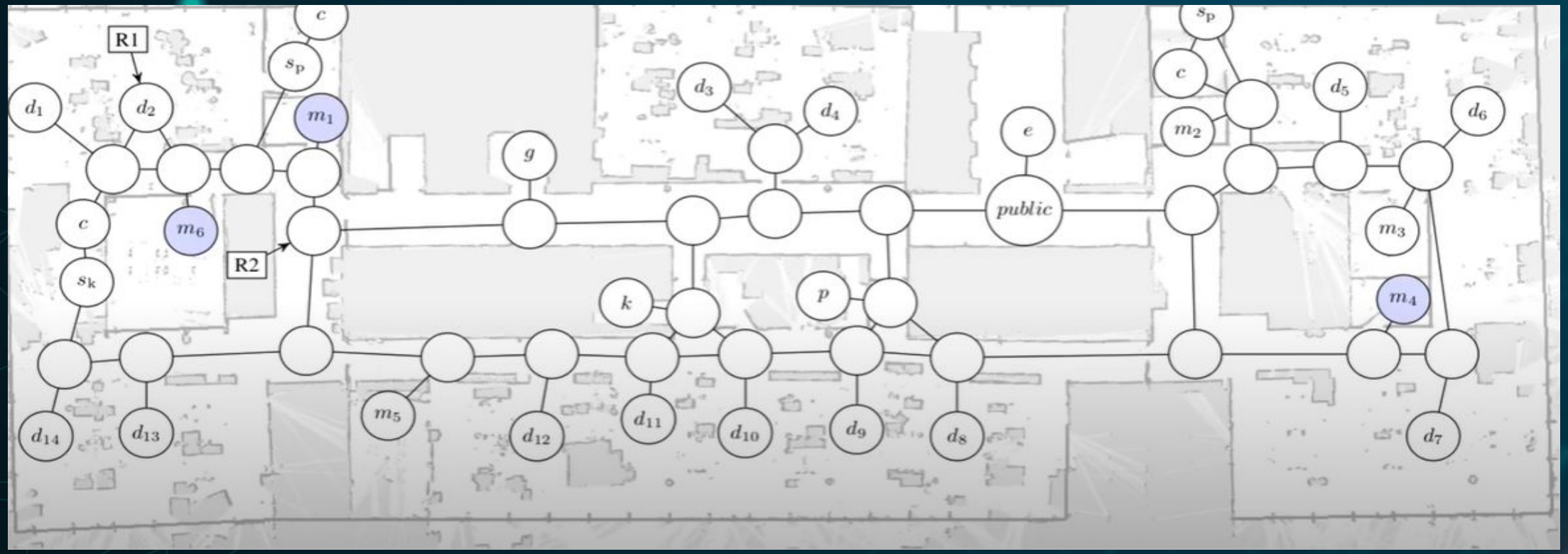
- This scenario illustrates applicability of the presented simultaneous task allocation and planning (STAP) framework to real-world problems.
- We consider an office scenario with 2 robots available at the indicated locations.



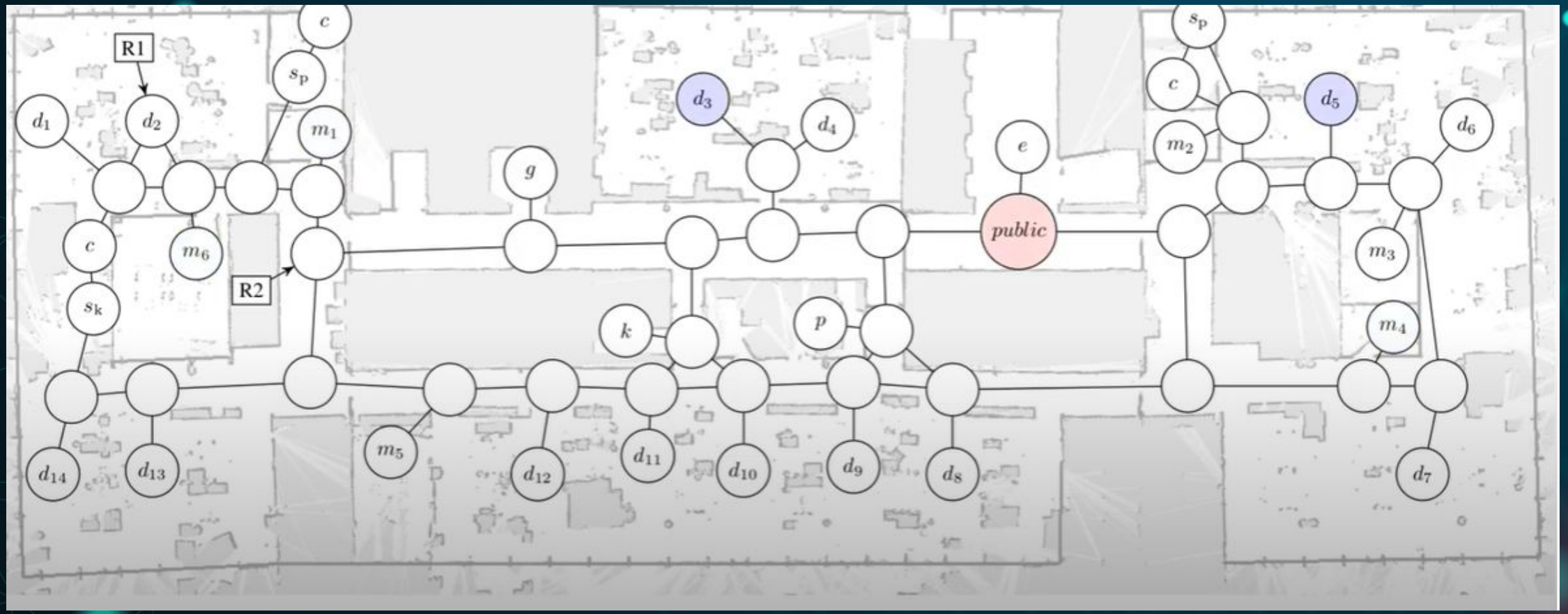
Tasks to be completed:

- Take a photo in the meeting rooms m1 , m4 , and m6
- Deliver a document from desk d5 to d3
- guide a person waiting at desk d11 to meeting room m6
- The camera has to be turned off for privacy reasons while not in meeting rooms
- The document is internal such that it should not be delivered through any public areas.”

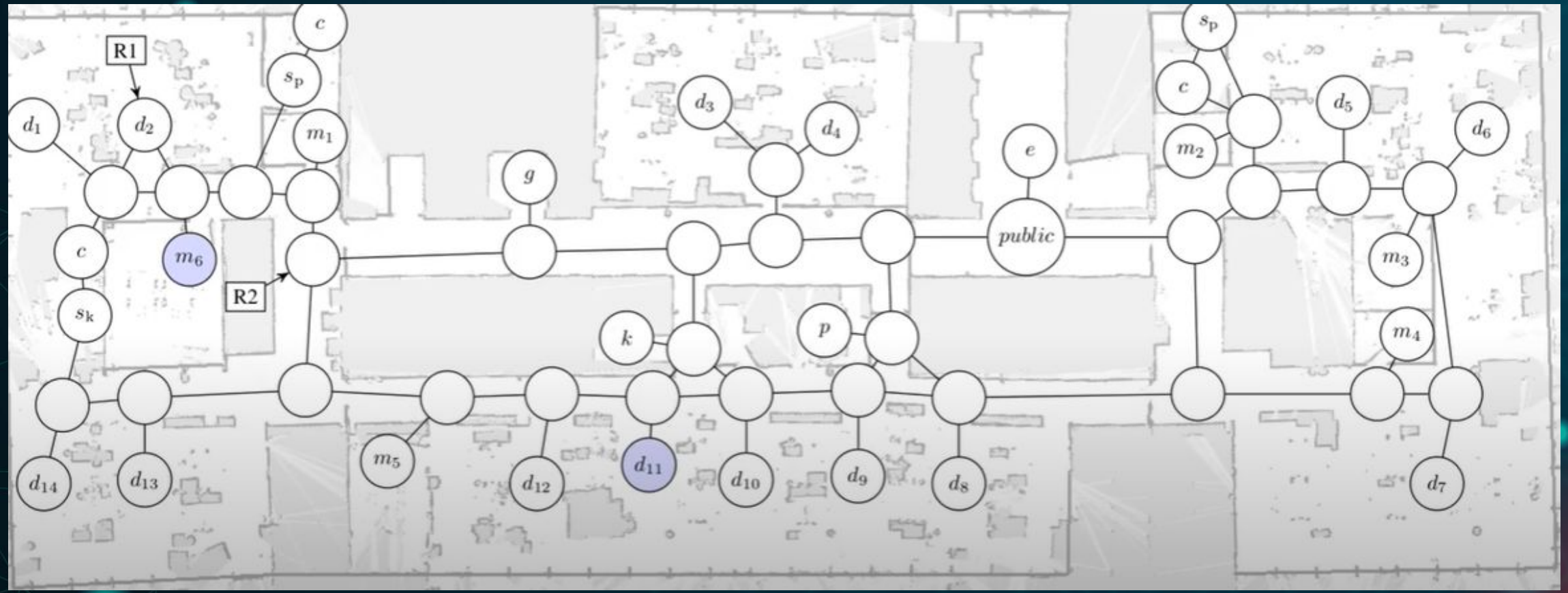
- The robots should take a photo of notes in each of the highlighted meeting rooms.



- Deliver a document from d5 to d3 and avoid public areas while carrying it.



... And guide a person at d11 to the meeting room m6.

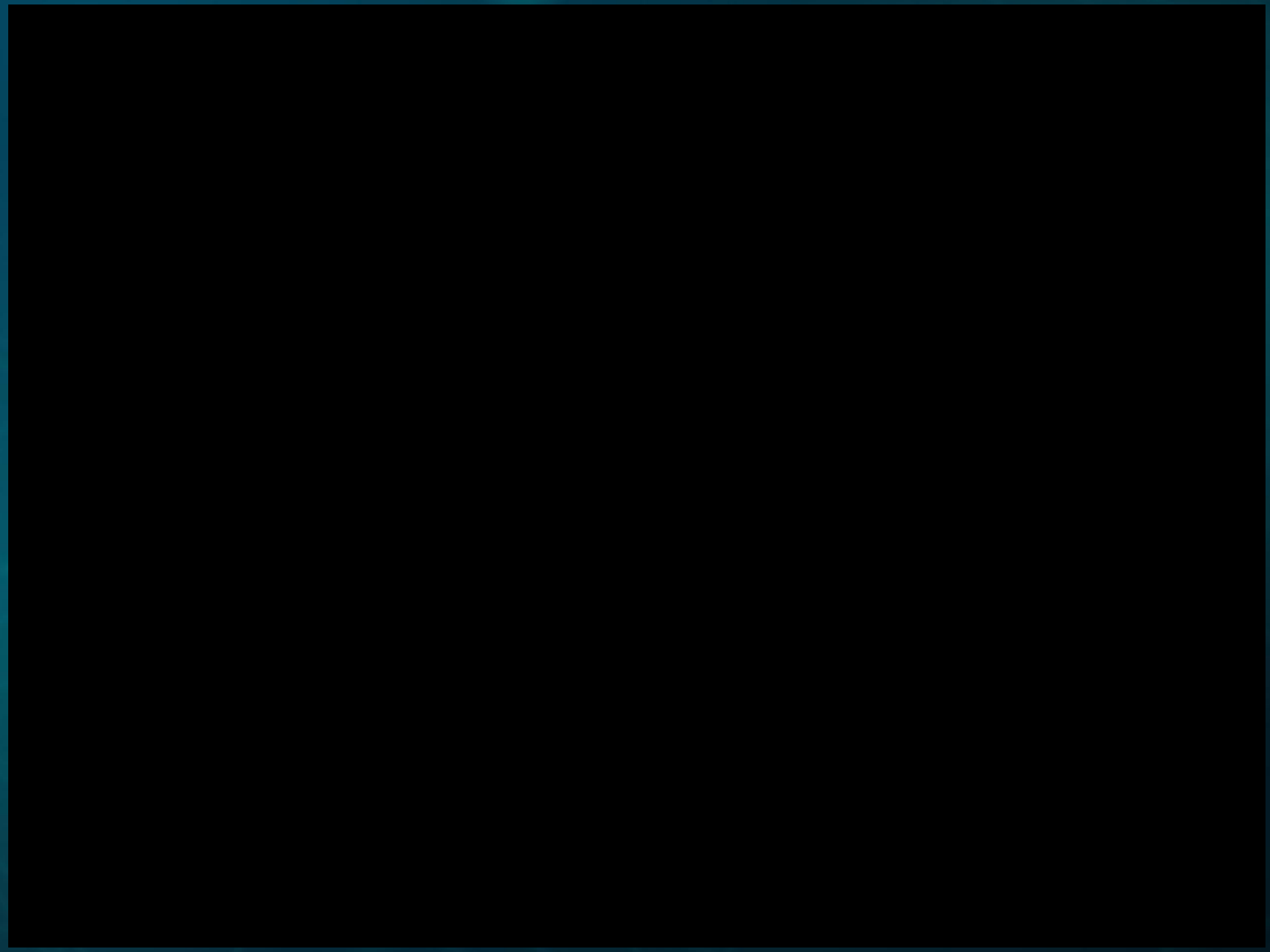


$$\phi_{\text{video}} = \Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo) \wedge \Box(\neg meeting \implies \neg camera) \wedge \Diamond(d_5 \wedge carry\, \mathcal{U}(d_3 \wedge \bigcirc \neg carry)) \wedge \Box(carry \implies \neg public) \wedge \Diamond(d_{11} \wedge guide\, \mathcal{U}(m_6 \wedge \bigcirc \neg guide))$$

- Robot behaviors are generated to fulfill the given specification:
  - Independent parts (“tasks”) of the specification are automatically identified
  - Task allocation and planning is done simultaneously
  - The planner finds actions to minimize the maximal execution time each robot will execute its assigned plan independently

$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry } U\ (\ d_3 \wedge \bigcirc \neg carry\ )\ )$$
$$\wedge \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide } U\ (\ m_6 \wedge \bigcirc \neg guide\ )\ )$$

- Fulfilled parts are highlighted in green.
- While carrying something avoid public areas.



Done: “ Take a photo in meeting room m1”

$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry } U\ (\ d_3 \wedge \bigcirc \neg carry\ )\ )$$
$$\wedge \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide } U\ (\ m_6 \wedge \bigcirc \neg guide\ )\ )$$

On the way, it takes the photo in meeting room m1.

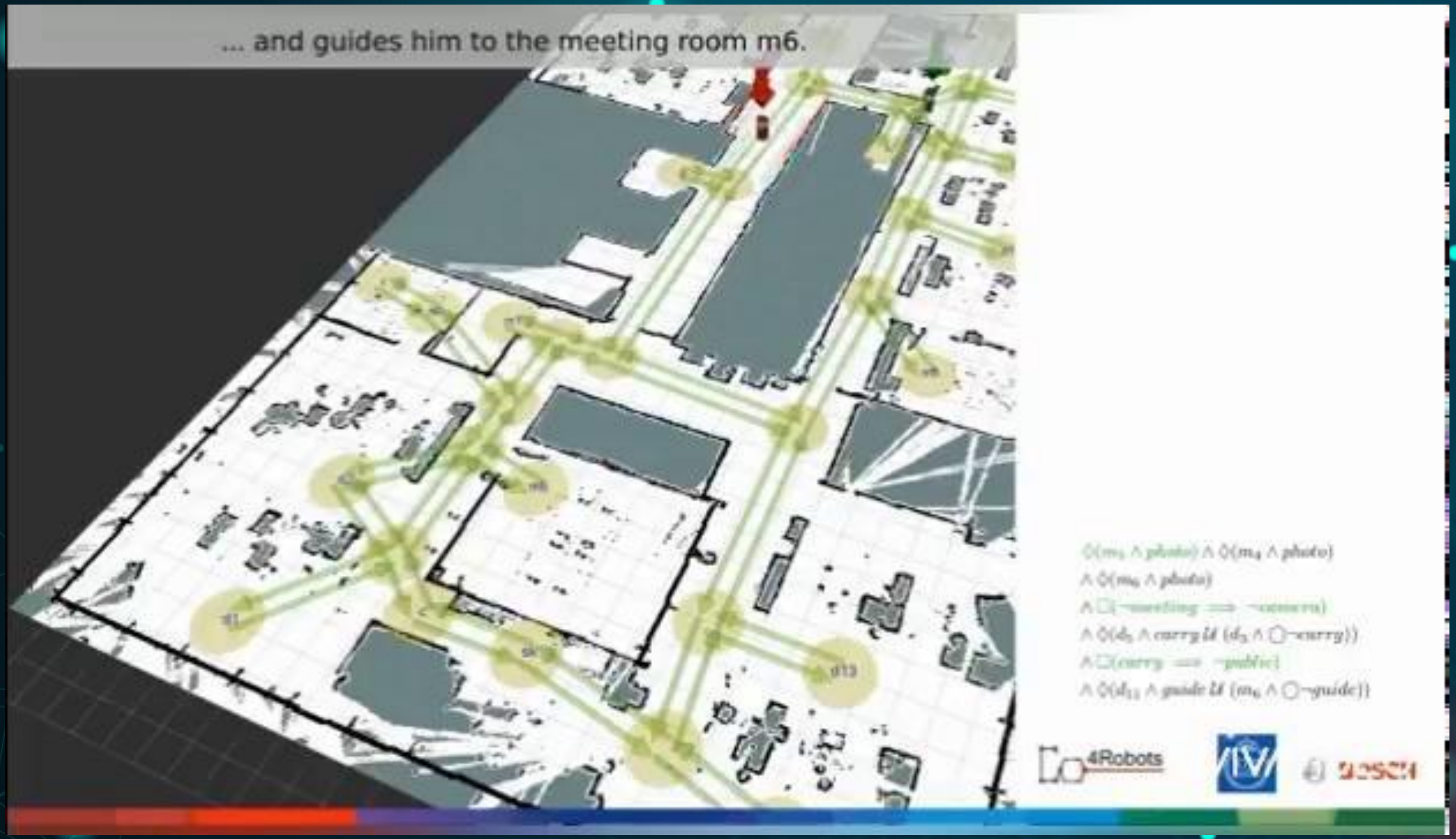
$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo)$
$\wedge \Diamond(m_6 \wedge photo)$
$\wedge \Box(\neg meeting \implies \neg camera)$
$\wedge \Diamond(d_5 \wedge carry\, \mathcal{U}\, (d_3 \wedge \bigcirc \neg carry))$
$\wedge \Box(carry \implies \neg public)$
$\wedge \Diamond(d_{11} \wedge guide\, \mathcal{U}\, (m_6 \wedge \bigcirc \neg guide))$

4Robots KTH BOSCH

gifs.com

$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry } U\ (\ d_3 \wedge \bigcirc \neg carry\ )\ )$$
$$\wedge \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide } U\ (\ m_6 \wedge \bigcirc \neg guide\ )\ )$$

Done: “ Guide a person at d11 to the meeting room m6”


... and guides him to the meeting room m6.
4Robots

$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry } U\ (d_3 \wedge \bigcirc \neg carry))$$
$$\wedge \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide } U\ (m_6 \wedge \bigcirc \neg guide))$$

Done: “ Take a photo in meeting room m6”


Done: "Guide a person at d11 to the meeting room m6"
4Robots

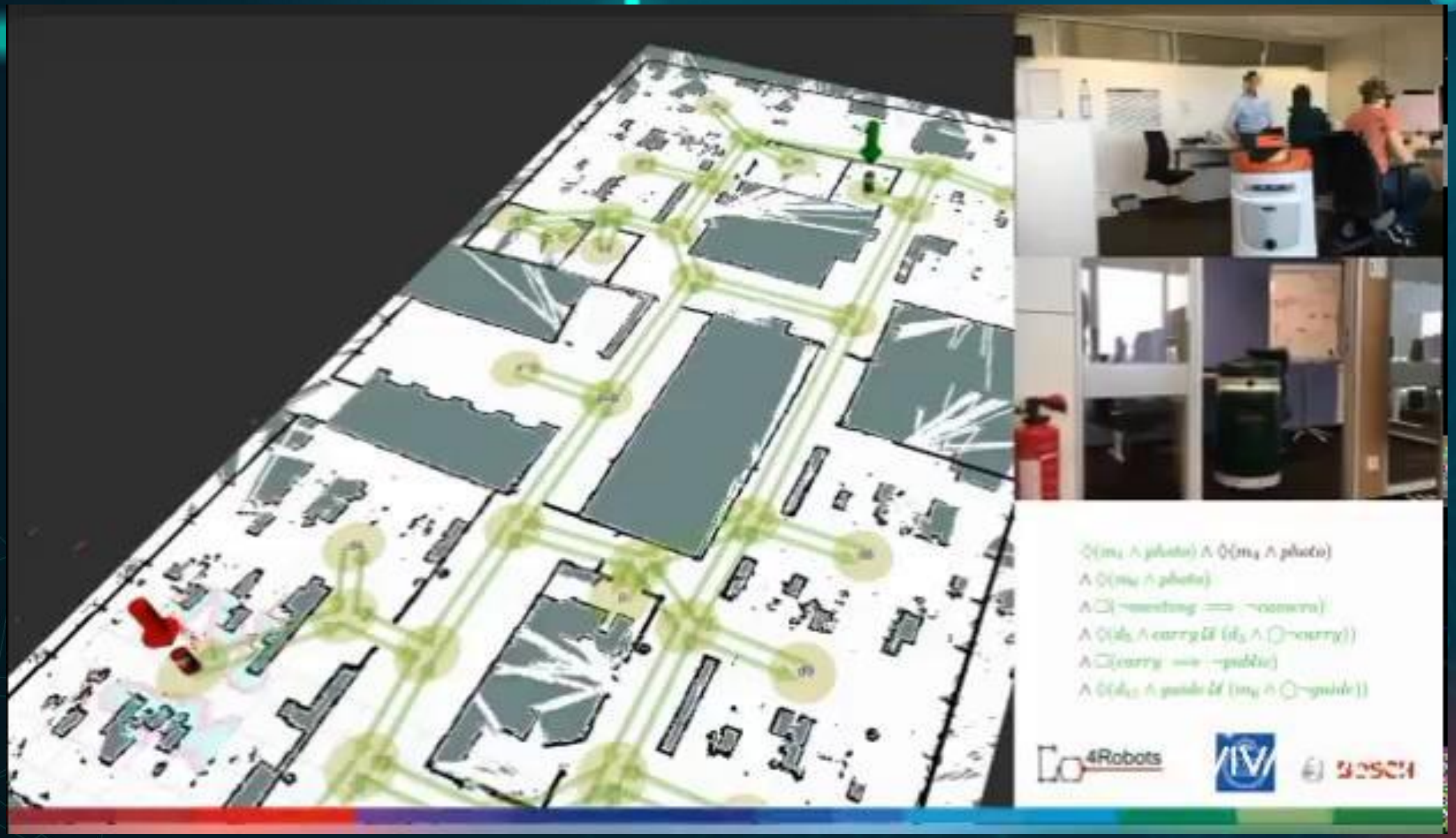
$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry}\ U\ (d_3 \wedge \bigcirc \neg carry))$$
$$\wedge \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide}\ U\ (m_6 \wedge \bigcirc \neg guide))$$

Done: “ Deliver a document from d5 to d3 ”


4Robots

Done: “Take a photo in meeting room m4”

$$\Diamond(m_1 \wedge photo) \wedge \Diamond(m_4 \wedge photo) \wedge \Diamond(m_6 \wedge photo)$$
$$\wedge \ \Box(\neg meeting \implies \neg camera)$$
$$\wedge \Diamond(d_5 \wedge \text{carry}\ U\ (\ d_3 \wedge \bigcirc \neg carry\ )\ )$$
$$\wedge \ \Box(carry \implies \neg public)$$
$$\wedge \Diamond(d_{11} \wedge \text{guide}\ U\ (\ m_6 \wedge \bigcirc \neg guide\ )\ )$$


4Robots

References

- F. Faruq, B. Lacerda, N. Hawes and D. Parker, “Simultaneous Task Allocation and Planning Under Uncertainty”, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Madrid, Spain, October 1-5, 2018
- P. Schillinger, M. Bürger, and D. V. Dimarogonas, “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems,” IJRR, vol. 37, no. 7, 2018.
- P. Schillinger, M. Burger, and D. V. Dimarogonas, “Decomposition of finite LTL specifications for efficient multi-agent planning,” in DARS, 2016.
- B. Lacerda, D. Parker, and N. Hawes, “Nested value iteration for partially satisfiable co-safe LTL specifications,” in AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents, 2015.
- Principles of model checking / Christel Baier and Joost-Pieter Katoen ; foreword by Kim Guldstrand Larsen.