Delft University of Technology
Faculty of Electrical Engineering, Mathematics & Computer Science

An Exploration of a Hierarchical Approach for MacDec-POMDPs

Kees Fani

to obtain the degree of Master of Science in Computer Science
to be defended publicly on the 14th of July 2021.

Thesis advisor:
Dr. Frans. A. Oliehoek

Daily supervisor:
Ir. Rolf A. N. Starre

Thesis committee:
Dr. Frans. A. Oliehoek
Prof dr. Catholijn M. Jonker
Dr. Matthijs T.J. Spaan
Ir. Rolf A. N. Starre

Student Number: 4437179

Acknowledgements

I would like to thank my Thesis Advisor Dr. Frans Oliehoek for this challenging but ultimately satisfying master thesis. Your expertise and intuition has been indispensable during this master project. Your insights and valuable feedback has vastly deepened my understanding and has greatly improved this thesis.
I would also like to thank my Daily Supervisor and PhD student Rolf Starre for all of his consistent and concrete feedback and for always making time for me. You have made me feel like my thesis was a priority. You have shown patience and bore with me during times where I was not able to work as hard as I wanted to. Thank you.
Furthermore I would like to thank my parents, my sister Lana and her husband Marco who have always supported me and were always there for me. I am incredibly lucky to have close family that is this warm and kind.

Contents

1 Introduction

In order to do something useful, you should know what to do and how to do it. The same goes for robots or other machines. Entities such as robots or other devices are generally referred to as *agents* within the field of Artificial Intelligence [1]. We see more and more of these agents in all kinds of different sectors. Examples of such sectors include but are not limited to; manufacturing [2], agriculture [3] and transport [4], a more specific example within the transport section are autonomous cars [5]. Advances in e.g. technology and science have made such agents more and more sophisticated and capable. This opens up a plethora of possibilities for solving problems or automating processes.
However, as agents and the problems they solve become more and more sophisticated, the software that controls these agents naturally also becomes more complex. Especially when considering systems of multiple cooperating agents. Agents receive information from their environments, and use such information to determine what to do. With rapidly increasing amounts of agents, observations and capabilities to take into account, the plans used to come to decisions become increasingly infeasible to create by hand.
Thus automated methods are necessary to create such plans. This motivates the overarching goal of this thesis, which is to advance such methods so difficult problems involving many agents can be solved more efficiently.
The approach this thesis uses is inspired by the ability of human intelligence to abstract away pedantic details when planning. We do not for example include which muscles we will utilize, when we make a plan for picking up groceries. Deciding to get groceries is on a completely different level than deciding how many centimeters we want to move a limb in a particular direction. Abstracting away such details greatly reduces the size of any potential plan. Which makes plans easier to construct, while the scope of the problem remains the same. Thus, utilizing this notion of different levels of abstraction during planning, is the approach this thesis uses to improve planning for agents.

1.1 Planning

According to the Cambridge Dictionary, a plan is defined as "a set of decisions about how to do something in the future" [6]. This is very much in line with how planning is done in the field of Artificial Intelligence. A set of decisions can be seen as a function, which would look like so:

$$\text{Information from the environment} \rightarrow \text{Action}$$

In other words, if the agent encounters information from the environment it is in, it executes a corresponding action. For example, a very rudimentary plan could be to swat a fly if a fly is seen. Seeing a fly in this case triggers the action of swatting this fly. In the field of AI, The representation of all the information of the environment is called the *state* (of the environment).
This thesis chooses an approach based on Markov Decision Processes (MDPs) [7]. This approach is chosen because it is widely used and studied. It has also already been extended in a number of useful ways [8–10].

1.2 Uncertainty

Uncertainty is everywhere. All the way from fundamental physics phenomena like quantum mechanics, to interactions between humans. The real world is messy and too infeasible to perfectly simulate. Uncertainty must be accounted for, this also goes for agents with artificial intelligence. Agents encounter uncertainty in a number of ways. First, we must consider that the actions do not always affect the environment in the same way. For example, performing the action of flipping a coin could result in an environment with the coin facing either heads or tails with a 50% chance. This is actually accounted for in the Transition function in MDPs.
Other types of uncertainty are not, such as uncertainty pertaining to understanding the environment itself. Humans do not have perfect information about its environment. To make up for this, people rely on the subjective information they perceive from the environment using their senses, such as sight, smell or touch. In other words, humans rely on their own observations. Machines or robots similarly can use their sensors to make observations, which are not always perfectly accurate. To have a more realistic artificial intelligence model, this lack of objective environment model should thus also be modeled for its agents. This is why this thesis builds upon Partially Observable MDPs (POMDPs) [11], which is a framework which extends MDPs to model individual subjective observations of agents, instead of assuming perfect information.
Another source of uncertainty is the unpredictability of others. This is the case for people as well as for other agents. It is impossible to predict with certainty what another fully independent human or agent will do. Thus, once again, to have a more realistic planning framework, multiple agents and their interactions with the environment must be modelled. This will be elaborated on in section 1.3.

1.3 Decentralization

Using a framework that models multiple agents has more benefits besides dealing with the uncertainty that comes from the addition of multiple agents. Systems that model multiple agents are called *Multiagent Systems* (*MAS*). MASs have been researched to reap the following potential benefits [12,13]. Greater opportunities for scalability, parallelization and flexibility, as adding additional agents could speed up the rate at which problems are solved. This would also increase robustness and reliability, as the failure or removal of a few agents would not necessarily result in a system failure. Also, the reusability and modularity of such agents and systems could reduce the cost of developing a MAS solution. These potential benefits are another reason why a MAS approach is used for this thesis.

There are multiple ways to control groups of agents. One of which is a centralized approach, where a central server gives orders to multiple agents. In this case, the POMDP framework could suffice [14]. As the observations from the agents are all given to the central server, which could then on its own determine the appropriate actions to take. However, this type of system would require very fast, noise free communication, which is often infeasible.
Alternatively, each agent could have its own plan for how to solve a particular problem, without a central server dictating what actions these agents should take at each point in time. These individual plans can however still be made in a centralized manner. The lack of a central decision maker would however still make this system a decentralized system. This would remove the necessity of communication during run-time and thus alleviate the requirement of centralized systems which was stated above. I.e. not requiring communication makes such a system applicable in more situations, which is an important reason why this thesis uses a decentralized approach.

In a decentralized solution, agents could each construct their own plan on their own. To make such a plan separately, the agents are required to consider what the other agents will do. However, this leads to an potentially infinite recursive process, as considering the actions of all other agents requires considering the actions of all other agents, for each agent. This thesis avoids this issue by constructing plans for all agents in the same process, at the same time. This is done by using a solution based on Decentralized POMDPs (Dec-POMDP) [15], which is an extension of POMDPs which does not require communication.

1.4 Abstraction

Solving large instances of Dec-POMDPs still remains a very difficult or infeasible task. A huge contributor to this problem is the fact that planning over many time-steps causes the number of potential solutions to increase exponentially. Thus using abstraction to allow for planning using shorter but more high-level plans would exponentially reduce the number of potential solutions. Dec-POMDPs do not use any form of abstraction when creating plans. Instead, plans are built using ground-level actions and observations. Imagine planning a holiday for a group of people by only considering muscle movements and sensory observations; it is not feasible.
If more complicated tasks must be solved, the plan must use actions that are more abstract. Executing such an abstract action, would result in a plan of low level actions to be carried out. Another way to call such an abstract action, is a *macro* action, or in other words; an action that consists of other actions. Luckily this already exists, it is called an *Option* [16]. The Macro DecPOMDP (Mac-DecPOMDP) framework [17], incorporates such options into the Dec-POMDP framework, this enables planning over macro-actions.
Although this work has shown that very large problems can be effectively modeled, it however does limit the model to a single layer of abstraction. An agent will execute a macro-action which will immediately start executing ground-level actions. Mac-DecPOMDPs do not allow macro-actions to include other macro-actions in their policies. This could potentially be a limiting factor, as humans for example also do not always limit themselves to a single level of abstraction when making a plan. It would be incredibly infeasible for example to have only a single level of abstraction that separates business operations to human muscle movements when making a business plan. Thus, this thesis introduces the Hierarchical Mac-DecPOMDP model, which enables planning using an arbitrary number of levels of abstraction.

1.5 Research Questions

The main research question of this thesis is the following:

How can the MacDecPOMDP Framework be extended to higher levels of abstraction, can existing Dec-POMDP solution methods be employed to solve such a model, and what are the resulting benefits and drawbacks?

This research question consists of smaller sub-questions. The main theme of this thesis is abstraction. Thus the first sub-research question is:

(1) How can the MacDecPOMDP Framework be extended to higher levels of abstraction?

Extending MacDecPOMDPs to arbitrary numbers of abstraction levels is not trivial, thus this question is answered by creating a model which will be called the *Hierarchical MacDecPOMDP* model (H-MacDecPOMDP). Thus, because creating the H-MacDecPOMDP model was part of this thesis, there does not exist a toolbox or other software implementations for H-MacDecPOMDPs. However, there does exist such software for Dec-POMDPs [18]. Thus it is very useful to know whether it is possible to transform H-MacDecPOMDPs into DecPOMDPs. This question is answered by providing an algorithm which transforms H-MacDecPOMDPs into Dec-POMDPs. Thus the second sun-research question is:

(2) Can H-MacDecPOMDPs be transformed into DecPOMDPs?

After constructing the H-MacDecPOMDP model, it should become more clear what some of the benefits and drawbacks of this approach are, which leads to this research question:

(3) What are the drawbacks and benefits of using H-MacDecPOMDPs as opposed to a (Mac-)Dec-POMDPs?

1.6 Outline

Chapter 2 will be a related work section which includes a brief comparison of this thesis to other works. In chapter 3, the background information will be given. This will consist of an explanation of MDPs, POMDPs, Dec-POMDPs and Mac-DecPOMDPs.
Then in chapter 4 an analysis of Mac-DecPOMDPs will be given, as it must be thoroughly understood and refined to fit with a possible extension.
In chapter 5, the H-MacDecPOMDP model will be constructed and explained. Then a new value function for H-MacDecPOMDPs is also given and explained. A proof of correctness for this value function will also be given. This chapter will thus give the answers to research question 1. Then in chapter 6, the transformation from H-MacDecPOMDPs to Dec-POMDPs is given and explained, which answers research question 2. Then in chapter 7 the drawbacks and benefits of H-MacDecPOMDPs and the transformation from H-MacDecPOMDPs to Dec-POMDPs are explained. This will also be done by constructing and analyzing a new problem; the *H-MacDecTiger* problem, which builds on the DecTiger problem from chapter 3. This problem will also be used for an experiment to further validate the correctness of the H-MacDecPOMDP framework. This chapter will thus answer research question 3.
Then the conclusion will summarize and briefly discuss what is done. There will also be a future work section, which will among other things mention some thoughts on further improvements to performance.

2 Related Work

This thesis presented H-MacDecPOMDPs, a hierarchical artificial intelligence model. There are however more hierarchical AI models. Namely in the field of *Reinforcement Learning* (RL), which is a branch of AI where the reward function is not given to the agents. Instead, the agents must learn to solve problems on the go, which is trained by reinforcing good behavior and/or penalizing bad behavior. The papers and approaches mentioned in this section are types of Reinforcement Learning. Here a distinction will be made between model-based and model-free hierarchical approaches, which will be explained in their respective subsections.

2.1 Model-Based Reinforcement Learning

In model-based reinforcement learning, the agent(s) knows a representation of the states it can be in, and also what states it might be in as a result of taking actions (i.e. the transition function is known). In this section, model-based approaches will be discussed.
First there is Kaelbling Lozano-Prez's paper [19] which integrates task planning and motion planning. This approach involves explicit modeled task abstraction as there are actions that are explicitly marked as "abstract" which may need to have previously defined conditions met before they can be executed. The similarity here is that a vertical hierarchy of abstraction is being built, i.e. actions that rely on lower level actions which rely on even lower level actions, etc. The difference is that the actions in these hierarchies are not options, instead they are defined using (a variation of) the STRIPS method [20].
Another hierarchical approach is Multi-Phase Learning which is used in the paper by Kroemer and Daniel [21]. As the name may suggest, this approach divides tasks into phases. The paper considers a set of robotic arms and hands that learns from watching humans manipulate objects. Human demonstrations are used by the robot to learn a probabilistic model of the phases and their transitions. This is done using State-Based Transitions Autoregressive Hidden Markov Model (STARHMM) [22]. A huge difference between H-MacDecPOMDPs and Multi-Phase learning is that in Multi-Phase learning, the actions themselves change per phase, i.e. an action may work completely differently from one phase to another. A phase in Multi-Phase learning can thus not be compared to an abstraction level of H-MacDecPOMDPs, instead a sequence of phases are conceptually more similar to the top level macro-actions in H-MacDecPOMDPs.
Multi-level learning is another hierarchical RL approach shown in the paper by Levy et al. [23]. Multi-level hierarchies are formed by dividing a problem into a set of problems that have a shorter horizon. These sets of problems are then solved simultaneously using Hierarchical RL approach. Hierarchies in this approach use nested levels of so-called goal-conditioned policies. These goal-conditioned policies are short sequences of actions that achieve a subgoal of the main task. The main similarity between Multi-level learning and H-MacDecPOMDPs is that hierarchies of policies are made that have shorter horizons. A big difference is the manner in which states are passed through these abstraction levels. H-MacDecPOMDPs pass information through abstraction levels using macro-actions, and Multi-level learning approaches use goal orientated states.
The paper [24] by J. Yang, et al. describes a a two-level hierarchical multi-agent reinforcement learning algorithm. This approach is different from H-MacDecPOMDP in that there are only two levels, and the hierarchy itself is structured differently. The highest of the two levels allow the agents to choose a different "skill" from which lower level actions are taken. This makes for an interesting interpretation of higher level thinking, which conceptually is more comparable to humans who choose different skill sets to work with in real life.

2.2 Model-Free Reinforcement Learning

In model-free reinforcement learning, the agent(s) does not know what states the agent(s) can be in, and also do not know what states it can transition to as a result of taking action(s). In this section, model-free approaches will be discussed.
The paper [25] by J. Cboi extends Relative Entropy Inverse RL [26] which involves learning reward functions from policies This approach is then used to provide hierarchical reasoning to reward functions. This method is different to that of H-MacDecPOMDPs in that here a hierarchy of reward functions is constructed, instead of a hierarchy of actions.
A paper by C. Daniel et al. uses Relative Entropy Policy Search [27] by Peters et al. which reduces data-loss that occurs in policy search while still approaching optimal policies. Hierarchical Relative Entropy Policy Search exploits task structures consisting of sub-tasks to find hierarchical policies consisting of sub-policies and gating networks using an Expectation Maximization [28] based method. The main similarity between this and H-MacDecPOMDPs is of course the use of "sub-policies" or policies of actions that have policies themselves. The biggest difference is the use of gating networks.
Another approach for building hierarchical reinforcement learning agents, is to build a hierarchy of sub-goals by hand for a specific problem. This is done in Dietterich et al. [29]. These sub-goals are then used to constrain the sub-policies required to reach that specific sub-goal. This is done by decomposing the single value function that used in Q-Learning and transform it into a hierarchy of value functions. The main difference with H-MacDecPOMDPs is of course that a hierarchies of subgoals (value functions) is created instead of a hierarchy of actions. This MAXQ approach can also be combined with R-MAX. R-Max, as described in *R-max-a General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning* [30], forces the agent to explore unexplored states by assigning the maximum reward possible to unexplored states. The paper *Hierarchical Model-Based Reinforcement Learning: R-max + MAXQ* by N. K. Jong and P. Stone [31] combines both the R-MAX and then MAXQ for a different hierarchical approach.

2.3 Approaches Using Communication

In this section, approaches are discussed which involve communication between the agents.
The paper by Ghavamzadeh et al. [32] is an approach based on MDPs where there is also a hierarchy of actions. This approach features an arbitrary amount of abstraction levels, however it does not make use of options. Instead, this approach uses a hierarchy of so called 'cooperative tasks'.
Then the approach of the paper [33] by R. Li, et al. describes the Dec-RAE-UPOM system. This is a partially observable and decentralized system, just like the MacDecPOMDP and by extension, H-MacDecPOMDP frameworks. This approach uses hierarchical operational models which specify how the tasks at hand can be solved cooperatively.
The paper [34] by H. Tang, et al., uses deep leaning, along with temporal abstraction to create hierarchies of actions. Temporal abstraction is conceptually quite similar to options, as it involves actions that take multiple time-steps to complete. Options are used by H-MacDecPOMDPs, which makes these hierarchical approaches more similar to each other. This approach allows agents to communicate with each other using a so called "Hierarchical Communication Network" (h-Comm).

3 Background

This chapter contains the background information necessary to understand this thesis. The H-MacDecPOMDP model that is presented in this thesis builds upon a number of other models.
First in section 3.1, an explanation of the Markov Decision Process (MDP) is given, which is a framework for single agent decision planning in fully observable environments. Then an extension of MDP called the Partially Observable MDP (POMDP) is explained in section 3.2, which as the name implies allows for partially observable environments. This is followed by section 3.3 which explains an extension of POMDP called the Decentralized POMDP (DecPOMDP), which adds support for multi agent planning. The DecPOMDP section also contains an example of a DecPOMDP problem called the Dec-Tiger problem (section 3.3.1), which is built upon in section 7.1 of the thesis. Then finally, section 3.4 explains an extension of DecPOMDP called Macro DecPOMDP (MacDecPOMDP), which introduces abstraction in the decision process by planning over macro-actions (or Options).
Recall that thesis extends the MacDecPOMDP framework to form the Hierarchical MacDecPOMDP framework (H-MacDecPOMDP), which supports arbitrary amounts of abstraction levels (chapter 5).

3.1 Markov Decision Process

A Markov Decision Process (MDP) [35], is a Markovian Stochastic Process, which provides a framework for modeling single agent decision making in a fully observable environment.
In this process, a plan is made which uses representations of the state of the environment to determine what actions to take. Such a plan is called a policy. This is where a useful property of MDPs come in to play, namely the Markov Property named after Russian mathematician Andrey Markov [36]. This property states that it is sufficient to only know the present state, as opposed to (all) previously encountered states, to predict what the future state(s) will be. Thus because states represent all of the necessary information of the environment and these states are fully observable, a policy (π) merely consists of a sequence of decision rules (e.g. $\pi = (\delta^0, \delta^1, \delta^2, \delta^3)$). Every time-step ($t$), has its own decision rule (δ^t). Such a decision rule is a function which takes in the present state and returns the probability of taking any given action. The signature of this function is shown in equation 3.1, where S represents the set of all states, and A represents the set of all actions the agent can take.

$$\delta^t : S \times A \to [0, \ldots, 1] \tag{3.1}$$

The goal of the agent is to achieve the highest possible sum of expected rewards. Rewards are given whenever an agent takes an action. The size of the reward is determined by the reward function. The reward function contains the reward for each state and action tuple. The signature of the reward function (R) is given in equation 3.2.

$$R : S \times A \to \mathbb{R} \tag{3.2}$$

Thus, the agent starts out in a state s^0, where the 0 superscript denotes the time-step. Then the agent takes an action (a^0) which is determined by the policy (π). Then the reward function returns a reward $r^0 = R(s, a)$. The agent will have interacted with the environment, which can change the state of the environment, this is called a state transition. After reaching the next state, the agent takes another action and the process continues until the horizon (h) of the problem is reached. The horizon represents the amount of time-steps the problem goes on for, i.e. the amount of actions the agent will take.

The probability of reaching a particular state s^{t+1} from state s^t after taking action a^t is determined by the Transition Function (T), which is shown in equation 3.3.

$$T : S \times A \times S \rightarrow [0, \ldots, 1] \tag{3.3}$$

Thus an MDP consists of 4 components, the set of states (S), the set of actions (A), the reward function (R) and the transition function (T), i.e. $\langle S, A, T, R \rangle$.
An overview of what happens in every time-step is given in equation 3.4 for 4 time-steps.

$$s^0, a^0, r^0, s^1, a^1, r^1, s^2, a^2, r^2, s^3, a^3, r^3 \tag{3.4}$$

The agent's goal is to maximize the sum of expected rewards, i.e. maximize the sum of the r^t values given in every time-step (t) as shown in equation 3.5.

$$\mathbb{E}\left[\sum_{t=0}^{h-1} r^t\right] = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s^t, a^t)\right] \tag{3.5}$$

The agent does this by optimizing its policy to make sure that the agent takes the actions that lead to the best rewards. The value function can be used to calculate the value of a policy when starting from a given state. The value function calculates the expected reward for the actions of the current time-step ($\sum_{a^t \in A} \pi(a^t|s^t) R(s^t, a^t)$), and will then add the expected value of the next time-step. In the next time-step the environment may be in a different state, thus the expected value of the next time-step accounts for this ($\sum_{s^{t+1} \in S} T(s^{t+1}|s^t, a^t) V^\pi_{t+1}(s^{t+1})$). The value function is shown in equation 3.6.

$$V^\pi_t(s^t) = \sum_{a^t \in A} \pi(a^t|s^t) \left[R(s^t, a^t) + \sum_{s^{t+1} \in S} T(s^{t+1}|s^t, a^t) V^\pi_{t+1}(s^{t+1}) \right] \tag{3.6}$$

When the horizon (h) of the problem is reached, the expected value of the next time-step is omitted. This base-case is shown in equation 3.7.

$$v^\pi_{h-1}(s^{h-1}) = \sum_{a^{h-1} \in A} \pi(a^{h-1}|s^{h-1}) R(s^{h-1}, a^{h-1}) \tag{3.7}$$

The initial state the environment can be in when the process starts may also need to be specified. The initial state distribution (b^0) is used to do this. The initial state distribution returns the probability of starting at any given state. E.g. if $b^0(s^b) = 0.5$, there is a 50% chance of starting at state s^b.
Thus a MDP can be 'solved' by finding a policy which maximizes the value function as shown in equation 3.8.

$$\arg\max_\pi \sum_{s^0 \in S} b^0(s^0) V^\pi_0(s^0) \tag{3.8}$$

3.2 Partially Observable MDPs

In Partially Observable MDPs (POMDPs) [37] [11], the state cannot be directly observed by the agent. Instead, the agent receives so called *observations*. Observations are designed to communicate information about the state of the environment.
These observations are observed whenever an agent takes an action. The sequence of actions, states,

rewards and observations for POMDPs looks like equation 3.9 as opposed to equation 3.4 for MDPs (when $h = 3$). In equation 3.9, o^t represents the observations.

$$s^0, a^0, r^0, s^1, o^1, a^1, r^1, s^2, o^2, a^2, r^2, s^3, o^3, a^3, r^3 \tag{3.9}$$

The set of observations is denoted by Ω. The probability of receiving an observation is determined by the Observation probability Function (O). The observation probability function returns the probability of observing the given observation after taking an action (a) which leads to a state (s). The signature of the observation probability function is shown in equation 3.10. POMDPs thus contain the following components: $\langle S, A, R, T, O, \Omega \rangle$.

$$O : A \times S \times \Omega \to [0, \ldots, 1] \tag{3.10}$$

The agent will use these observations to estimate what states the environment could be in and thus what actions the agent should take. Such an estimation of the state of the environment is known as a *belief*. The policies of agents thus take beliefs as input instead of states. A belief (b) is a vector which for every state contains the probability for the given state to be the actual state of the environment. At first the belief equals the initial state distribution (b^0). Afterwards a belief update is performed whenever the agent observes an observation. During the belief update, the belief for each state is updated. Some states are more likely to lead to particular observations than other states, the belief update exploits this information to calculate an updated belief, which is shown in equation 3.11.

$$b_{ao}^{t+1}(s^{t+1}) = \frac{O(o^{t+1}|a^t, s^{t+1}) \sum_{s^t \in S} T(s^{t+1}|s^t, a^t) b^t(s^t)}{\sum_{s^{t+1} \in S} O(o^{t+1}|a^t, s^{t+1}) \sum_{s^t \in S} T(s^{t+1}|s^t, a^t) b^t(s^t)} \tag{3.11}$$

A POMDP can be modeled as a belief-state MDP, where the beliefs form the states. This helps in constructing a value function. Such a belief MDP value function must specify the transition and reward functions for beliefs instead of states.
The belief reward function shown in equation 3.12, calculates the expected reward of the belief with respect to the states.

$$R(b^t, a^t) = \sum_{s^t \in S} b(s^t) R(s^t, a^t) \tag{3.12}$$

In the belief observation function shown in equation 3.13 gives the probability of observing the next observation by taking an action when having a particular belief. If the current belief as well as the current action and the next observations are known, the belief update will always yield the same updated belief. Thus, a separate "belief transition function" is not necessary.

$$Pr(o^{t+1}|b^t, a^t) = \sum_{s^{t+1} \in S} O(o^{t+1}|a^t, s^{t+1}) \sum_{s^t \in S} T(s^{t+1}|a^t, s^t) b(s^t) \tag{3.13}$$

Thus the value function of such belief MDPs is similar to that of MDPs (equation 3.6), however the summation over all states is replaced with a summation over all observations. Another difference is that the policy of the agent, the reward function and transition function use beliefs instead of states. The Belief MDP value function is shown in equation 3.14.

$$V_t^\pi(b^t) = \sum_{a^t \in A} \pi(a^t|b^t) \left[R(b^t, a^t) + \sum_{o^{t+1} \in \Omega} Pr(o^{t+1}|b^t, a^t) V_{t+1}^\pi(b_{ao}^{t+1}) \right] \tag{3.14}$$

3.3 Decentralized POMDPs

In Decentralized POMDPs (DecPOMDP) [15], there are multiple agents which take actions independently from each other. The identifier of each agent is put into the set I. These agents cannot share their local information, and thus must all decide separately on which actions to take based on the observations it has seen and actions it has taken. In DecPOMDPs, the agents are unable to estimate what the current state of the environment is, because the transition probability from state to state depends on the actions that **all** agents take, and the agents do not know what actions the other agents take. Thus the agents can no longer form beliefs about the state, which results in these agents having to take actions based only on the history of actions and observations the agent themselves keep track of.

Equation 3.15 shows an example of a history for agent 1 after 3 time-steps. The timesteps are put as the superscript to avoid confusion with specifying the agent which is done using the subscript.

$$h_1 = a^0, o^1, a^1, o^2, a^2, o^3 \tag{3.15}$$

The vector of all actions that are taken at the same time are referred to as a joint action. In this thesis, this is written with a bold typeface like so: $\boldsymbol{a}$. The action of a specific agent (i) is referred to like so: a_i. Each agent has its own set of actions and observations (A_i and Ω_i), without the subscript, the sets denote the sets of all joint actions (A) and joint observations (Ω).

The observations are all given at the same time to each agent. Thus the Observation Probability function (O) provides the probability of a joint observation ($\boldsymbol{o}$) given that a joint action ($\boldsymbol{a}$) is taken, which leads to a state (s). Similarly, the Transition function (T) gives the probability of transitioning to a state when taking a joint action when in the current state. Finally the reward function (R) will also take joint actions as inputs.

Thus a DecPOMDP consists of 6 components, the set of states (S), the set of actions (A), the reward function (R) and the transition function (T), the set of observations (Ω) and the observation probability function (O) and the set of agents (I) i.e. $\langle S, A, T, R, \Omega, O, I \rangle$.

Although the execution of the model is decentralized, the agents can still be planned at the same time using a value function. This value function must keep track of the joint history ($\boldsymbol{h}$) of the agents as well as the current state. In every time-step these histories will be appended with the joint action taken and joint observation that is observed. This model is decentralized, thus the policies of the agents do not use joint actions and joint histories. Instead, regular histories and actions are used in the policies of agents (e.g. $\pi_1(a_1|h_1)$). The probabilities of taking each separate actions are then multiplied with those of the other agents to calculate the probability of taking a joint action ($\prod_{i \in I} \pi(a_i|h_i)$). Further, the value function will also loop over all joint observation probabilities. This way all possible cases for the next joint histories can be taken into account. Thus the Dec-POMDP value function is given in equation 3.16, in this equation $'$ will be used to denote that the component is of the next time-step (e.g. next joint observation $\boldsymbol{o}'$ instead of $\boldsymbol{o}_{t+1}$).

$$V^{\pi}(s, \boldsymbol{h}) = \sum_{\boldsymbol{a} \in A} \prod_{i \in I} \pi_i(a_i|h_i) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{o}' \in \Omega} O(\boldsymbol{o}'|\boldsymbol{a}, s') V^{\pi'}(s', \boldsymbol{h}') \right] \tag{3.16}$$

Although finite-horizon Dec-POMDP have at least one optimal deterministic joint policy [38], non-deterministic policies are used here as they are also used in the remainder of this thesis.

3.3.1 Dec-Tiger

In this subsection, an example of a DecPOMDP problem is given. This problem is the Dec-Tiger problem [39]. Dec-Tiger is a problem where two agents try to maximize their rewards by finding treasure. The treasure is hidden behind one of two doors. The door to the room without the treasure has a tiger in it, which will devour the agent if let free, i.e. give a big negative reward (a penalty). In each time-step, an agent will either open the left door (a_{OL}), open the right door (a_{OR}) or listen for the location of the tiger (a_{Li}). Thus the actions are shown in equation 3.17 and the observations are shown in equation 3.18.

$$A_1 = A_2 = \{a_{OL}, a_{OR}, a_{Li}\} \tag{3.17}$$

$$\Omega_1 = \Omega_2 = \{o_{HL}, o_{HR}\} \tag{3.18}$$

If both agents decide to listen, no doors will be opened leading to the tiger and treasure to always remain in the same place. Equation 3.19 shows the transitions where both agents listen. In all other cases, the transition will randomly lead to one of the two states, e.g. $T(s_r|\langle a_{OL}, a_{Li}\rangle, s_l) = 0.5$.

$$T(s_l|\langle a_{Li}, a_{Li}\rangle, s_l) = T(s_r|\langle a_{Li}, a_{Li}\rangle, s_r) = 1.0 \tag{3.19}$$

If both agents perform the listen actions (a_{Li}), both agents will separately have a probability of 0.85 of hearing (observing) the tiger location correctly, an example is shown in equation 3.20.

$$O(\langle o_{HL}, o_{HR}\rangle|\langle a_{Li}, a_{Li}\rangle, s_l) = 0.85 \cdot (1 - 0.85) \tag{3.20}$$

The content of the reward function (R) is exhaustively shown in figure 1.
The optimal policy for a horizon of 3 ($h = 3$) consists of listening in the first two steps, and opening the door on the opposite side of where the tiger was observed. If the two observations are contradictory, e.g. an agent receives o_{HL} and o_{HR}, the agent performs the listen action again. The decision tree for this policy is shown in figure 2.

$\boldsymbol{a}$	s_l	s_r
$\langle a_{Li}, a_{Li}\rangle$	-2	-2
$\langle a_{Li}, a_{OL}\rangle$	-101	+9
$\langle a_{Li}, a_{OR}\rangle$	+9	-101
$\langle a_{OL}, a_{Li}\rangle$	-101	+9
$\langle a_{OL}, a_{OL}\rangle$	-50	+20
$\langle a_{OL}, a_{OR}\rangle$	-100	-100
$\langle a_{OR}, a_{Li}\rangle$	+9	-101
$\langle a_{OR}, a_{OL}\rangle$	-100	-100
$\langle a_{OR}, a_{OR}\rangle$	+20	-50

Figure 1: The rewards for each joint action and state combination in the DecTiger problem.

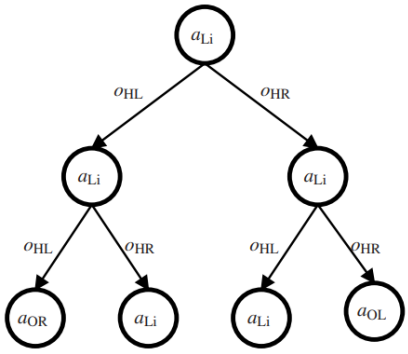


Figure 2: Dec-Tiger Optimal Policy Decision Tree [40]

3.4 Macro Dec-POMDPs

Macro DecPOMDPs (MacDecPOMDPs) [17] are DecPOMDPs where the agents have policies of *Macro-Actions* (also called *Options* [16]) instead of regular actions. These macro-actions allow the agents to plan with one level of abstraction. I.e. it allows agents to abstract away details when making a plan. These macro-actions control what "regular" actions (also referred to as primitive or ground level actions) will be executed. Imagine baking a cake, a macro-action could be to add an egg to the mixture, a ground level action could to pick up the egg, followed by another ground-level action which cracks open the egg, etc. MacDecPOMDPs consist of the following: $\langle I, S, T, R, O, M, \Omega, Z, h \rangle$, where M and Z are the two added sets of macro components.

3.4.1 Macro Components

The letter m_i is used to represent a macro-action of agent i and the capital letter M_i refers to the set of macro-actions of agent i. Macro-actions consist of three parts ($m_i = (\beta_{m_i}, \mathcal{I}_{m_i}, \pi_{m_i})$), the termination function (β_{m_i}), the initiation set ($\mathcal{I}_{m_i}$) and the policy (π_{m_i}). The termination function decides when the current macro-action of an agent should stop executing (i.e. terminate). The termination function does this by returning the probability of a given macro-action to terminate when reaching a given history of ground level actions and observations (H^A). Equation 3.21 shows the signature of the termination function. The histories of ground level actions and observations (h_i^A) are identical to the action-observation histories in DecPOMDPs.

$$\beta_{m_i} : H_i^A \to [0, \ldots, 1] \tag{3.21}$$

$$\mathcal{I}_{m_i} : H_i^M \tag{3.22}$$

$$\pi_{m_i} : H_i^A \times A_i \to [0, \ldots, 1] \tag{3.23}$$

The policy (π_{m_i}) of a macro-action m_i of agent i determines what ground level macro-actions will be taken. Such a policy is equivalent in structure to policies of DecPOMDPs, its signature is given in 3.23.
The initiation set ($\mathcal{I}_{m_i}$) of a macro-action m_i of agent i contains all macro-action and macro-observation histories ($\mathcal{I}_{m_i} \subseteq H_i^A$) that permit the agent to take that particular macro-action. I.e. before a macro-action can be taken $h_i^A \in \mathcal{I}_{m_i}$ must hold. The signature of $\mathcal{I}_{m_i}$ is given in equation 3.22.
Macro-observations (z_i) are observations where details irrelevant to the overarching goal are abstracted away. ζ is the set of macro-observations ($z_i \in \zeta_i$). The macro-observation probability function (Z_i) uses the current macro-action and next state to determine the probability of observing the given macro-observation. The function signature of the macro-observation probability function (Z_i) is shown in equation 3.24.

$$Z_i : M \times S \times \zeta_i \to [0, \ldots, 1] \tag{3.24}$$

As the name implies, macro-action and macro-observation histories (shortened to macro-histories) are histories consisting of the sequence of macro-actions and macro-observations that have been taken by the agent. Equation 3.25 shows an example of a macro-history where the agent has taken 3 macro-actions.

$$h_i^M = [m_i^0, z_i^1, m_i^1, z_i^2, m_i^2, z_i^3] \tag{3.25}$$

The amount of macro-actions taken in macro-histories does not have to be equal to the amount of time-steps that have passed. This is because macro-actions can take multiple time-steps before they terminate. Macro-actions also do not all have to take the same amount of time-steps to terminate.

Thus agents can have different amounts of macro-actions in their macro-histories at the same time-step.
The overarching policy that determines what macro-actions an agent takes is called the macro-policy (μ_i). Macro-policies use macro-histories (h_i^M) to determine what the probability of taking a macro-action is. The signature of a macro-policy is given in equation 3.26.

$$\mu_i : H_i^M \times M_i \rightarrow [0, \ldots, 1] \tag{3.26}$$

3.4.2 MacDecPOMDP Definition

This section consists of an overview of the components of MacDecPOMDPs. Every component in MacDecPOMDPs is given and briefly described to give a birdseye overview of MacDecPOMDPs.
$Model = \langle I, S, b_0, T, R, O, M, \Omega, Z, h \rangle$ where;

- I : the finite set of agents. e.g.: $\{1, 2, 3, 4, ...\}$
- S : the set of states. e.g.: $\{state1, state2, ...\}$
- b_0 : the distribution of the initial states; $S \rightarrow [0, 1]$.
- A_i : the set of actions for agent i. E.g. $a_i \in A_i$.
- T : the state transition probability function holds the probabilities of reaching any state s' from any state s by having the agents perform a ground-level joint action $\boldsymbol{a}$.
  ($T : S \times A \times S \rightarrow [0, 1]$ and $T(s'|s, \boldsymbol{a}) = Pr(s'|s, \boldsymbol{a})$).
- R : the reward function. $R(s, \boldsymbol{a})$ is the reward given for a joint ground-level action $\boldsymbol{a}$ when in a state s. ($R : S \times A \rightarrow \mathbb{R}$)
- Ω_i : the set of observations for agent i. E.g. $o_i \in \Omega_i$.
- O : the set of joint observation probability functions for the ground level holds the probabilities of getting the joint observation $\boldsymbol{o}$ from reaching state s by having the agents perform a joint ground action $\boldsymbol{a}$. ($O : A \times S \times \Omega^0 \rightarrow [0, 1]$ and $O(\boldsymbol{o}|\boldsymbol{a}, s) = Pr(\boldsymbol{o}|\boldsymbol{a}, s)$)
- M_i : the set of macro-actions for agent i. Where $m_i \in M_i$ and $m_i = (\beta_{m_i}, \mathcal{I}_{m_i}, \pi_{m_i})$.
  $\mathcal{I}_{m_i}$ is the initiation set ($\mathcal{I}_{m_i} \subseteq H_i^M$), β_{m_i} is the termination function ($\beta_{m_i} : H_i^A \rightarrow [0, 1]$) and π_{m_i} is the policy of macro-action m_i ($\pi_{m_i} : H_i^A \times M_i^A \rightarrow [0, 1]$).
- ζ_i : the set of macro-observations for agent i. E.g. $z_i \in \zeta_i$.
- Z_i : the set of macro-observation probability functions for agent i holds the probabilities of getting the observation z_i when reaching state s' by taking macro-action m_i. ($Z_i : M_i \times S \times \zeta_i \rightarrow [0, 1]$ and $Z_i(z_i|m_i, s') = Pr(z_i|m_i, s')$)
- h : the problem horizon, the amount of steps (primitive actions executions), before the problem terminates.

3.4.3 2-Agent Value Function

A MacDecPOMDP value function is given in the MacDecPOMDP paper [17], it is shown in equation 3.27. This value function is a 2-agent value function, meaning that it works for two agents.
In this value function the macro-actions that are being taken are given as inputs for the value function ($V_t^\mu(m_1, m_2, s)$). The policies of these macro-actions are then used to determine the probabilities of taking the ground-level actions. Although the execution of macro-actions often span multiple ground level time-steps (t), the value function considers all ground level time-steps separately. This is the case because the reward function takes ground-level actions as inputs, another reason is that the macro-actions do not all terminate or start execution at the same ground level time-step.
There are two summations over observation probabilities ($\sum_{o_1,o_2} O(o_1, o_2|a_1, a_2, s)$ and $\sum_{o'_1,o'_2} O(o'_1, o'_2|a_1, a_2, s')$), section 4 explains that the first is not necessary. After determining the probabilities of taking the ground level actions, the expected value of the next time-step over the states and next observations is calculated. This is similar to how this is done in the DecPOMDP value function (equation 3.16), except that the probabilities of taking new macro-actions and having macro-actions persist is calculated. With two agents, there are four cases: both macro-actions terminate, m_1 terminates, m_2 terminates and neither terminates. The probability of termination is determined by the termination function, thus the probability of a macro-action persisting is equal to one minus the probability of it terminating. If a macro-action terminates, a new macro-action (m'_i) must be chosen. This is done by the macro-policy (μ_i). Finally, if the next macro-actions are chosen or the old macro-actions have been determined to persist, the next time-step is considered with these (new) macro-actions ($V_{t+1}^\mu(m'_1, m'_2, s')$).
This value function is further explained and built upon in chapter 4.

$$
\begin{aligned}
V_t^\mu(m_1, m_2, s) = & \\
& \sum_{o_1,o_2} O(o_1, o_2|a_1, a_2, s) \sum_{a_1,a_2} \pi_{m_1}(o_1, a_1)\pi_{m_2}(o_2, a_2) \\
& \Bigg[R(s, a_1, a_2) + \sum_{s'} T(s'|a_1, a_2, s) \sum_{o'_1,o'_2} O(o'_1, o'_2|a_1, a_2, s') \\
& \Bigg(\beta_{m_1}(o'_1)\beta_{m_2}(o'_2) \sum_{m'_1,m'_2} \mu_1(o'_1, m'_1)\mu_2(o'_2, m'_2)V_{t+1}^\mu(m'_1, m'_2, s') \text{ (both terminate)} \\
& + \beta_{m_1}(o'_1)(1 - \beta_{m_2}(o'_2)) \sum_{m'_1} \mu_1(o'_1, m'_1)V_{t+1}^\mu(m'_1, m_2, s') \text{ (agent 1 terminates)} \\
& + (1 - \beta_{m_1}(o'_1))\beta_{m_2}(o'_2) \sum_{m'_2} \mu_2(o'_2, m'_2)V_{t+1}^\mu(m_1, m'_2, s') \text{ (agent 2 terminates)} \\
& + (1 - \beta_{m_1}(o'_1))(1 - \beta_{m_2}(o'_2))V_{t+1}^\mu(m_1, m_2, s') \Bigg) \Bigg] \text{ (neither terminates)}
\end{aligned}
\tag{3.27}
$$

4 A Critical Analysis of the MacDecPOMDP Framework

The MacDecPOMDP framework discussed in background section 3.4 extends the DecPOMDP framework to allow for a level of abstraction. This section's purpose is to provide an analysis, which consists of the following parts:

- A problem that arises from the current definition of macro-observations is shown.
- An example of a MacDecPOMDP that displays the above issue with macro-observations is shown.
- The definition of Macro-Observations is changed to address this problem.
- The base case of the value function is defined.
- The value function is made to incorporate ground-level histories.
- The value function is made to incorporate all macro components.
- The value function is generalized to work with an arbitrary amount of agents instead of two.
- This new value function is linked to Dec-POMDP value functions.

4.1 Macro-Observation Definition Issue

As it stands, Macro-observations in MacDecPOMDPs are defined just like regular observations. They are both defined as a function from states and actions to observations as shown in background section 3.4.
This is formalized as follows:

$$Z_i : M_i \times S \times \zeta_i \rightarrow [0, 1]$$

Where ζ_i represents the macro-observation, M_i represents the macro-action and S represents the state.

Defining Macro-observations this way, implies that the information gained from macro-observations cannot necessarily be derived from just ground level observations or actions. This is because the macro-observation probability function uses states to output a macro-observation probability. This may lead to extra information because states cannot always be derived from action/observation histories.
The extra information that is provided by macro-observations can lead to misleading results. MacDecPOMDP variants of DecPOMDP problems could be performing unreasonably well, because MacDecPOMDPs could be made to give the solver extra information through macro-observations. This is shown in section 4.2.
Thus to avoid this issue, it should be made impossible to garner information from macro-observations that is not contained in the history of actions and observations.

4.2 Trivialized MacDec-POMDP Example

In this section, we will consider the Dec-Tiger example problem which was explained in background section 3.3.1. We will show that by adding macro-observations, this problem can be trivialised. We will see that adding macro-observations, as they are defined now, can give more information to the agents than they would have without macro-observations. This example is provided to demonstrate the issue laid out by section 4.1.

4.2.1 Problem Description

Although the Dec-Tiger problem was also explained in background section 3.3.1, the full modified trivialized Dec-Tiger problem will be given for the convenience of the reader.
In the Dec-Tiger and the MacDecTiger problem, there are two rooms and two agents, one room contains treasure, while the other room has a tiger. They both either pick one of two doors, or listen to one of two doors for a tiger. Listening is an action, opening the left door is an action, and opening the right door is an action. When both agents perform the listening action, an observation is transmitted to both agents that says which door the tiger is behind. There is a 85% chance that the information in this observation is correct.

4.2.2 Trivializing The Problem

This problem can be made 'invalid' or rather, extremely trivial by defining macro-observations that contain more information than the 'primitive' observation. Here we do this by defining a macro-observation that just tells the agents where the treasure is with certainty. This macro-observation can then be used to select one of two macro-actions. One of these macro-actions picks the left door, the other picks the right door. I.e. these macro-actions have a policy that only chooses the primitive actions associated with opening the specified door. The whole point of the DecTiger problem is to reason over what doors should be opened. Here this reasoning becomes extremely trivial because the macro-observations just plainly state which door should be opened.
Thus, the only change is the addition of the macro-action (M) and macro-observation (ζ) components as well as the macro-observation probability function (Z) and macro-policies (μ).

4.2.3 Formal Definition for Invalid-MacDec-Tiger

I : The two agents called 1 and 2 ($\{1, 2\}$).

S : The two states, state for having the tiger be at the left, and one for having the tiger on the right. ($\{s_l, s_r\}$)

$A_{1,2}$: The set of actions for both agents; Opening the left door, opening the right door, and listening. ($\{a_{OL}, a_{OR}, a_{Li}\}$)

$\Omega_{1,2}$: This set denotes the possible observations, one for hearing the tiger at the left door, and one for hearing the tiger at the right door. ($\{o_{HL}, o_{HR})\}$

T : The transition function; when listening for a door, the state does not change. If an agent does not listen, (and thus opens a door), the state is randomized (one of the two states is chosen). See table 1.

Actions/Transition	$s_l \rightarrow s_l$	$s_l \rightarrow s_r$	$s_r \rightarrow s_r$	$s_r \rightarrow s_l$
$< a_{Li}, a_{Li} >$	1.0	0.0	1.0	0.0
otherwise	0.5	0.5	0.5	0.5

Table 1: Transition function for the trivialized MacDecTiger problem.

Actions/Reward	s_l	s_r
$< a_{Li}, a_{Li} >$	-2	-2
$< a_{Li}, a_{OL} >$	-101	+9
$< a_{Li}, a_{OL} >$	+9	-101
$< a_{OL}, a_{Li} >$	-101	+9
$< a_{OL}, a_{OL} >$	-50	+20
$< a_{OL}, a_{OR} >$	-100	-100
$< a_{OR}, a_{Li} >$	+9	-101
$< a_{OR}, a_{OL} >$	-100	-100
$< a_{OR}, a_{OR} >$	+20	-50

Table 2: Reward function for the trivialized MacDecTiger problem.

R : Table 2 denotes the rewards given for every joint action.

O : Given the action and state, the odds of receiving each observation is given. See table 3.

	s_l		s_r	
Actions/Observation	o_{HL}	o_{HR}	o_{HL}	o_{HR}
$< a_{Li}, a_{Li} >$	0.85	0.15	0.15	0.85
otherwise	0.5	0.5	0.5	0.5

Table 3: Observation function for the trivialized MacDecTiger problem.

$\zeta_{1,2}$: The macro-observations for both agents; one for the tiger being behind the left door, and one for the tiger being behind the right door. ($\{z_l, z_r\}$)

$M_{1,2}$: The set of macro-actions for both agents. For each agent i, these macro-actions always terminate after every step; $\beta_{m_i} : H_i^A \rightarrow 1$, they can be executed at any time; $\mathcal{I}_{m_i} : H_i^M$. The first macro-action has a policy where the left door is always opened. The second macro-action has a policy where the right door is always opened.

$$m_i = (\beta_{m_i}, \mathcal{I}_{m_i}, \pi_{m_i})$$

$$M_i = \left\{ \begin{array}{l} m_l = (H_i^A \rightarrow 1, H_i^M, \pi(a_l|\text{any history}) = 1 \text{ and } \pi(a_r|\text{any history}) = 0, \\ m_l = (H_i^A \rightarrow 1, H_i^M, \pi(a_r|\text{any history}) = 1 \text{ and } \pi(a_l|\text{any history}) = 0, \end{array} \right\}$$

$Z_{1,2}$: The macro-observation probability function for both agents. If the tiger is behind the left door, the corresponding macro-observation is sent. Similarly, if the tiger is at the right door, its respective macro-observation is sent. See table 4.

	s_l		s_r	
Actions/Observation	z_l	z_r	z_l	z_r
$m \in M_i$	1.0	0.0	0.0	1.0

Table 4: Macro-Observation function for the trivialized MacDecTiger problem.

$\mu_{1,2}$: The macro-policy of agent 1; if its macro-history ends with observing that the tiger is behind the left door, then open the right door. If the tiger is observed to be behind the right door, then open the left door. This is defined as follows, note that the default value is 0;

- $\mu_i(m_r|\{H_i^M|\ last\ item\ z_l\}) = 1$
- $\mu_i(m_l|\{H_i^M|\ last\ item\ z_r\}) = 1$

4.2.4 Discussion

In the non-macro variant, always opening a door cannot lead to a positive expected reward, the odds of opening a door with the tiger and being severely punished for it is just too big. In the macro-action case, the macro-observations will give away the location of the tiger with certainty, thus a reward of +20 will be achieved in every round. This is because the macro-action that corresponds to the correct door will always be chosen.
+20 is the maximum reward that can be given in a single round. This maximum reward is always reached in this example.
Thus caused by the problem of not being sure whether to trust the observation is now completely trivialized. To solve This, receiving "extra" information about the state from macro-observations should avoided.

4.3 Replacing States with Histories in the Macro-Observation Probability Function

As it stands, the macro-observation function is defined as follows:

$$Z_i : \zeta_i \times M_i \times S \to [0, 1] \tag{4.1}$$

To solve the issue of receiving "extra" information about the state from macro-observations, the state as a parameter of the macro-observation probability function must be removed. The next closest thing to a state are the agent's action/observation-histories. This also explains why agents use action/observation-histories as the parameter in their policies.
This new observation probability function definition is formally written in equation 4.2. This new macro-observation probability function will return the probability of any macro-observation $z \in \zeta_i$ to occur, given macro-action $m \in M_i$ and ground level history $h_i \in H_i^A$.

$$Z_i : \zeta_i \times M_i \times H_i^A \to [0, 1] \tag{4.2}$$

These macro-observations thus do not give the agents any information about the state that would not be given by the action/observation histories. This is because the new definition (equation 4.2) only differs in that states are replaced by an action/observation history.

4.4 Adding a Base Case to the Value Function

The 2-agent value function shown in section 3.4.3 does not have a base case. A base case is necessary because eventually the last time-step to consider is reached and the next time-step does not need to be calculated.
The last timestep is timestep $h-1$, as horizon h denotes the amount of timesteps the problem has, and timesteps are counted starting from 0. The base case consist of only the expected reward of the current macro-actions. This is calculated by looping over every possible ground-level action and their probabilities and multiplying their rewards. This is expressed in equation 4.3.

$$V^{\mu}_{h-1}(m_1, m_2, s^t) = \sum_{a_1,a_2} \pi_{m_1}(o_1, a_1)\pi_{m_2}(o_2, a_2)R(s, a_1, a_2) \tag{4.3}$$

4.5 Integrating Histories with the Value Function

As per the macro-action policy definition, macro-action policies are supposed to take ground level histories as inputs. The ground level history is not used in the 2-agent value function. In this subsection, the history is integrated into the 2-agent value function.
Histories are not included as parameters in the V^{μ} function. This is rectified like so:

$$V^{\mu}_t(m_1, m_2, s) \rightarrow V^{\mu}_t(m_1, m_2, s, h^A_1, h^A_2) \tag{4.4}$$

The termination function (β) and the policies (π) do not use histories. Instead they just take an observation as input. These two functions are however defined to use histories like so: $\pi_{m_i} : H^A_i \times A_i \rightarrow [0, 1]$ and $\beta_{m_i} : H^A_i \rightarrow [0, 1]$. Thus histories should be added to policies and termination functions. These changes are written as follows:

$$\sum_{a_1,a_2} \pi_{m_1}(o_1, a_1)\pi_{m_2}(o_2, a_2) \rightarrow \sum_{a_1,a_2} \pi_{m_1}(a_1|h^A_1)\pi_{m_2}(a_2|h^A_2)$$

$$\beta_{m_i}(o'_i) \rightarrow \beta_{m_i}(h^{A\prime}_i)$$

There is an additional summation over the observations their probabilities of from the current time-step. This observation presumably uses ground actions of the previous time-step, along with the current state as inputs. This is written as follows:

$$\sum_{o_1,o_2} O(o_1, o_2|a_1, a_2, s)$$

According to the paper that this 2-agent value function is from, these calculations were added to simplify the value function [17]. In a value function of form of equation 4.4, this calculation is not necessary. This is because the observation of time-step t is part of the history of time-step t, which is already given as input of the value function. The probability calculation for these observations would also be performed twice for each chosen observation, once in the current time-step and once in the one after it. Thus, I choose to remove this part of the equation.

4.6 Integrating Macro-Observations and Macro-Histories

The 2-agent value function also does not incorporate all macro-components. Macro-histories and macro-observations are not used. This is a problem because one of the main advantages of using MacDecPOMDPs, is that the plans (macro-policies (μ)) can rely on histories of macro-observations and macro-actions, and thus will not have to take every small ground level observation into account. The macro-histories are not included as parameters in the V^μ function. This is rectified like so:

$$V_t^\mu(m_1, m_2, s, h_1^A, h_2^A) \rightarrow V_t^\mu(m_1, m_2, s, h_1^A, h_2^A, h_1^M, h_2^M)$$

The regular components should be used in places where macro-observations or macro-histories are used. For example, consider the macro-action policy μ; in the 2-agent value function it is used as follows:

$$\sum_{m_i' \in M_i} \mu_i(m_i'|o_i')$$

Even though it is defined as follows in section 3.4.1: $\mu_i : H_i^M \times m_i \rightarrow [0, 1]$. Thus to rectify this, the macro-history $h_i^{M\prime}$ should be used instead of the regular observation o_i'.
In order to construct this new macro-history, the current macro-action (m_i) and a new macro-observation (z_i') must be appended to the previous macro-history (h_i^M). The current macro-action is already known, but the new macro-observation is not. Thus every new macro-observation must be considered. This is done by summing over each possible next macro-observation and their probabilities. This is written as follows:

$$\sum_{z_i' \in \zeta_i} Z_i(z_i'|m_i, h_i^{A\prime})$$

Implementing these changes results in the following notation in the case where agent 1's macro-action terminates and agent 2's macro-action persists:

$$\beta_{m_1}(h_1^{A\prime})(1 - \beta_{m_2}(h_2^{A\prime})) \sum_{m_1' \in M_1} \mu_1(m_1'|o_1')V_{t+1}^\mu(m_1', m_2, s', h_1^{A\prime}, h_2^{A\prime})$$

$$\Downarrow$$

$$\beta_{m_1}(h_1^{A\prime})(1 - \beta_{m_2}(h_2^{A\prime})) \sum_{z_1' \in \zeta_1} \sum_{m_1' \in M_1} Z_1(z_1'|m_1, h_1^{A\prime})\mu_1(m_1'|h_1^{M\prime},)V_{t+1}^\mu(m_1', m_2, s', h_1^{A\prime}, h_2^{A\prime}, h_1^{M\prime}, h_2^{M\prime})$$

Where:

$$h_i^{M\prime} = (h_i^M, m_i, z_i')$$

4.7 Generalizing the 2-Agent Value Function to n Agents

The value function shown so far only allows for 2 agents. MacDecPOMDPs however do not inherently have this limit. Thus the value function should be adjusted to accommodate for arbitrary amounts of agents.

4.7.1 n-Agent Value Function Parameters

First the value function parameters must be changed to allow for arbitrary amounts of agents. To do this, joint variables will be used. Joint macro-actions, joint histories and joint macro-histories will replace the separate ones. Joint variables are written in bold. This change is shown as follows:

$$V_t^{\mu}(m_1, m_2, s, h_1^A, h_2^A, h_1^M, h_2^M) \rightarrow V_t^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}^A, \boldsymbol{h}^M)$$

As is the case in the original 2-agent value function, ground level actions are not given as inputs in this function.

4.7.2 Component Generalization

In order to generalize the agent amount, it would be useful to be able to use joint ground-level actions as inputs to policies. However, joint variables cannot be put into the policy function (π), this is because the agents choose actions for themselves with what they themselves know. Thus a probability function $Pr(\boldsymbol{a}|\boldsymbol{h}^A)$ is used to denote the probability of the joint action that is being taken.
This change is shown as follows:

$$\sum_{a_1,a_2} \pi_{m_1}(h_1^A, a_1)\pi_{m_2}(h_2^A, a_2) \rightarrow \sum_{\boldsymbol{a}\in A} Pr(\boldsymbol{a}|\boldsymbol{h}^A)$$

This joint action probability function is defined using a product notation which is used to go through all agents and multiply the probabilities of taking specific actions per agent. This is written as follows:

$$Pr(\boldsymbol{a}|\boldsymbol{h}^A) = \left(\prod_{i\in I} \pi_{m_i}(h_i^A, a_i)\right)$$

The reward-, observation- and transition functions do accept joint variables, thus joint variables are passed into these functions. This is written as follows:

$$R(s, a_1, a_2) + \sum_{s'\in S} T(s'|a_1, a_2, s) \sum_{o_1'\in\Omega_1, o_2'\in\Omega_2} O(o_1', o_2'|a_1, a_2, s')$$

$$\Downarrow$$

$$R(s, \boldsymbol{a}) + \sum_{s'\in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{o}'\in\Omega} O(\boldsymbol{o}'|\boldsymbol{a}, s')$$

4.7.3 Superset of Terminating Agents

All cases of terminating macro-actions must be considered. To this end, the powerset of the set of agents is used $I_T \in 2^I$ (e.g. $2^{\{1,2\}} = \{\{\}, \{1\}, \{2\}, \{1,2\}\}$). I_T represents the set of terminating agents that is being considered. For each of these cases, different calculations are necessary. In the 2-agent value function, each of these cases are summed over separately like so:

$$\text{(both terminate)} + \text{(1 terminates)} + \text{(2 terminates)} + \text{(neither terminate)}$$

Here we will sum over these cases using the summation notation like so:

$$\sum_{I_T \in 2^I}$$

The calculation of each of these cases consists of three parts, the termination calculation, the macro-action and macro-observation calculations and the recursive calls.
The termination calculation is done like so:

$$\prod_{i \in I_T} \beta_{m_i}(h_i^{A\prime}) \prod_{i \in I \setminus I_T} \left(1 - \beta_{m_i}(h_i^{A\prime})\right)$$

The first product calculates the combined probability of the agents to terminate their macro-actions, and the second product calculates the probability of the remaining agents to not terminate their macro-actions.
The value function must consider every possible macro-observation and macro-action for each agent with terminating macro-actions. This is done using the following expression:

$$\sum_{\boldsymbol{z}'_{I_T} \in \zeta_{I_T}} \sum_{\boldsymbol{m}'_{I_T} \in M_{I_T}}$$

$\boldsymbol{z}'_{I_T}$ and $\boldsymbol{m}'_{I_T}$ are joint variables for only the agents with terminating macro-actions. The probabilities of choosing these macro-actions and macro-observations are expressed like so:

$$Pr(\boldsymbol{z}'_{I_T} | \boldsymbol{m}_{I_T}, \boldsymbol{h}^{A\prime}_{I_T}) Pr(\boldsymbol{m}'_{I_T} | \boldsymbol{h}^{M\prime}_{I_T})$$

Where both $Pr(\boldsymbol{z}'_{I_T} | \boldsymbol{m}_{I_T}, \boldsymbol{h}^{A\prime}_{I_T})$ and $Pr(\boldsymbol{m}'_{I_T} | \boldsymbol{h}^{M\prime}_{I_T})$ are defined as products of the macro-observations ($Z(z'_i | m_i, h_i^{A\prime})$) and macro-action probabilities from the macro policy ($\mu_i(m'_i | h_i^{M\prime})$) respectively for the terminating agents (I_T).
The recursive call to the value function contains the new joint macro-action and macro-histories, this is expressed like so:

$$V^{\mu}_{t+1}(s', \boldsymbol{m}', \boldsymbol{h}_i^{A\prime}, \boldsymbol{h}^{M\prime})$$

Do note that $\boldsymbol{m}'$ and $\boldsymbol{h}^{M\prime}$ includes the macro-actions and macro-histories of the previous time-step, for the agents with macro-actions that had not terminated. Refer to section 4.8.1 for a more precise explanation.

4.8 Mac-DecPOMDP Value Function

In this section new value function is shown in its entirety.

4.8.1 Notation Comments

The timestep after t is $t+1$ this continues until $t \geq h$, h denotes the amount of time-steps we evaluate the policy for. Thus t counts the amount of timesteps the evaluation has calculated for, and not the amount of timesteps left. The apostrophe (x') is used to denote the next $(t+1)$ (macro-)action, (macro-)observation or (macro-)history.
The upcoming histories are equal to the previous history, appended by the current (macro-)action and upcoming (macro-)observation. This is written like so:

$$h_i^{A\prime} = [h_i^A, a_i, o_i']$$

$$h_i^{M\prime} = [h_i^M, m_i, z_i']$$

To reiterate; If the macro-action of agent i does not terminate, the macro-history remains the same: $h_i^{M\prime} = h_i^M$.

4.8.2 Value Function

$$V_t^\mu(s, \boldsymbol{m}, \boldsymbol{h}_i^A, \boldsymbol{h}_i^M) = \sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{h}^A) \Bigg[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{o}' \in \Omega} O(\boldsymbol{o}'|\boldsymbol{a}, s')$$

$$\sum_{I_T \in 2^I} \prod_{i \in I_T} \beta_{m_i}(h_i^{A\prime}) \prod_{i \in I \setminus I_T} \big(1 - \beta_{m_i}(h_i^{A\prime})\big) \sum_{\boldsymbol{z}'_{I_T} \in \zeta_{I_T}} Pr(\boldsymbol{z}'_{I_T}|\boldsymbol{m}_{I_T}, \boldsymbol{h}_{I_T}^{A\prime}) \sum_{\boldsymbol{m}'_{I_T} \in M_{I_T}} Pr(\boldsymbol{m}'_{I_T}|\boldsymbol{h}_{I_T}^{M\prime}) V_{t+1}^\mu(s', \boldsymbol{m}', \boldsymbol{h}_i^{A\prime}, \boldsymbol{h}^{M\prime}) \Bigg] \tag{4.5}$$

Base case ($t = h-1$):

$$V_{h-1}^\mu(s, \boldsymbol{m}, \boldsymbol{h}_i^A, \boldsymbol{h}_i^M) = \sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{h}^A) R(s, \boldsymbol{a}) \tag{4.6}$$

Where:

$$Pr(\boldsymbol{a}|\boldsymbol{h}^A) = \prod_{i \in I} \pi_{m_i}(a_i|h_i^A)$$

$$Pr(\boldsymbol{z}'_{I_T}|\boldsymbol{m}_{I_T}, \boldsymbol{h}_{I_T}^{A\prime}) = \prod_{i \in I_T} Z(z_i'|m_i, h_i^{A\prime})$$

$$Pr(\boldsymbol{m}'_{I_T}|\boldsymbol{h}_{I_T}^{M\prime}) = \prod_{i \in I_T} \mu_i(m_i'|h_i^{M\prime})$$

4.9 Linking the New MacDecPOMDP Value Function to Other Value Functions

In this section, the MacDecPOMDP value function (equation 4.5) will be linked to Dec-POMDP value functions. This is done to better understand the MacDecPOMDP value function. In order to help with this, the MacDecPOMDP value function should be rewritten to make the link with the Dec-POMDP more clear. Thus the MacDecPOMDP will be rewriten. Afterwards the link between the value functions will be explained. The reason that there are two value function formulations is that the formulation of section 4.8.2 was derived from the 2-agent value function and was thus made to resemble the 2-agent value function. The value function of this section is made to resemble the value functions that it is compared to.

4.9.1 Value Function Rewrite Formulation

The rewrite of the MacDecPOMDP value function is given in this section. An explanation of this rewrite, which includes an explanation of how termination and persistence of macro-actions is checked (as seen in $\prod_{\{i \in I | m_i \ terminates\}}$ and $\prod_{\{i \in I | m_i \ persists\}}$), will be given in section 4.9.2.

$$V_t^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}^A, \boldsymbol{h}^M) = \sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A) \Big[R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o'} \in \Omega} Pr(s', \boldsymbol{o'}|s, \boldsymbol{a}) \sum_{\boldsymbol{z'} \in \zeta^*} \sum_{\boldsymbol{m'} \in M} Pr(\boldsymbol{z'}, \boldsymbol{m'}|\boldsymbol{m}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}, s') V_{t+1}^{\mu}(s', \boldsymbol{m'}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}) \Big] \quad (4.7)$$

Where:

$$Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A) = \prod_{i \in I} \pi_{m_i}(h_i^A, a_i) \quad (4.8)$$

$$Pr(\boldsymbol{z'}, \boldsymbol{m'}|\boldsymbol{m}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}, s') = \prod_{\{i \in I | m_i \ terminates\}} \beta_{m_i}(h_i^{A\prime}) Z_i(z_i'|m_i, h_i^{A\prime}) \mu_i(h_i'^M, m_i') \prod_{\{i \in I | m_i \ persists\}} \begin{cases} (1 - \beta_{m_i}(h_i^{A\prime})) & \text{If } m_i' = m_i \\ 0 & \text{Otherwise} \end{cases} \quad (4.9)$$

$$Pr(s', o'|s, a) = T(s'|s, a) O(o'|a, s') \quad (4.10)$$

Base case ($t = h - 1$):

$$V_{h-1}^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}^A, \boldsymbol{h}^M) = \sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A) R(s, \boldsymbol{a}) \quad (4.11)$$

4.9.2 Explanation of this Rewrite

The state and ground level observation are grouped into one:

$$Pr(s', o'|s, a) = T(s'|s, a)O(o'|a, s')$$

The calculation of the probability of the joint ground level action to be taken is also grouped into a single probability function like so:

$$Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A) = \prod_{i \in I} \pi_{m_i}(a_i|h_i^A)$$

Another change is that the whole calculation of any subsequent macro-action and macro-observation pair is abstracted away like so:

$$Pr(\boldsymbol{z'}, \boldsymbol{m'}|\boldsymbol{m}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}, s') = \prod_{\{i \in I | m_i \ terminates\}} \beta_{m_i}(h_i^{A\prime}) Z_i(z_i'|m_i, h_i^{A\prime}) \mu_i(h_i'^M, m_i')$$
$$\prod_{\{i \in I | m_i \ persists\}} \begin{cases} (1 - \beta_{m_i}(h_i^{A\prime})) & \text{If } m_i' = m_i \\ 0 & \text{Otherwise} \end{cases}$$

If the agent's macro-action does not terminate, the macro-action must not change. Thus a check must be made to eliminate cases where the macro-action does change when it should not.

And finally, the approach of using a powerset notation to determine which agents terminate their macro-actions is changed. Instead, the information within the joint macro-observation is used to determine which agents terminate. This is made possible by introducing a $\overset{*}{\zeta}$ set, which contains every element of ζ, along with the addition of *null*-observations for each agent. These new *null*-observations signify that the last macro-action of that specific agent has not terminated. In the powerset notation approach (section 4.7.3), there are no macro-observations that follow a persisting macro-action, thus all that is done here is filling this void by adding a macro-observation that signifies macro-action persistence. Thus, if the macro-action of agent i persists, the next macro-observation will equal null ($z_i' = null$).

$$\sum_{\boldsymbol{z'} \in \overset{*}{\zeta}} \sum_{\boldsymbol{m'} \in M}$$

If the next macro-action terminates, the next macro-observation will not equal null ($z_i' \neq null$). The effect on the macro-action-observation history (h_i^M) is thus merely that sometimes, the history can end with a *null*-observation (when the macro-action persists). There will not be *null*-observations at any other point in the history, as *null*-observations are replaced with regular (non-*null*) macro-observations when the last macro-action in this history does terminate. The macro-action-observation histories (h^M) thus still will not have a fixed length with respect to t (unlike the ground level action-observation histories (h^A)).
The base case is calculated in exactly the same way. Except that the $Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A)$ notation instead of the $\prod_{i \in I} Pr(a_i|m_i, h_i^A)$ notation is used for the action probability calculation. This is written like so:

$$V_{h-1}^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}_i^A, \boldsymbol{h}_i^M) = \sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A) R(s, \boldsymbol{a})$$

4.9.3 The Link Between Value Functions

The value functions 3.4.2 and 4.1.3 from the book 'A Concise Introduction to Decentralized POMDPs' [40] will be used in this section. Here the connection between these two value functions and value function (equation 4.7) above will be explained.
The two value function equations (3.4.2 & 4.1.3) do however assume deterministic policies. Thus the value function (equation 4.7) and its base case (equation 4.11) must be rewritten to account for that. This is done by first redefining the policies like so:

$$\pi_{m_i} : H_i^A \times A_i \rightarrow [0,1]$$

$$\Downarrow$$

$$\pi_{m_i} : H_i^A \rightarrow A_i$$

Now that the policies are deterministic, it is no longer necessary to loop over all possible joint ground level actions, as only the macro-action that the policy dictates will be taken. E.g. $\sum_{\boldsymbol{a} \in A} Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A)$ in equation 4.7 and $Pr(\boldsymbol{a}|\boldsymbol{m}, \boldsymbol{h}^A)$ in equation 4.11 are no longer necessary. Instead, the ground level action can be determined directly through the new deterministic policies $\left(\boldsymbol{a} = \left[\pi(h_1^A), \pi(h_{\dots}^A), \pi(h_{|I|}^A)\right]\right)$.
Thus equation 4.7 now looks like so:

$$V_t^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}^A, \boldsymbol{h}^M) = R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o'} \in \Omega} Pr(s', \boldsymbol{o'}|s, \boldsymbol{a}) \sum_{\boldsymbol{z'} \in \zeta^*} \sum_{\boldsymbol{m'} \in M} Pr(\boldsymbol{z'}, \boldsymbol{m'}|\boldsymbol{m}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}, s') V_{t+1}^{\mu}(s', \boldsymbol{m'}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}) \quad (4.12)$$

With base case:

$$V_{h-1}^{\mu}(s, \boldsymbol{m}, \boldsymbol{h}^A, \boldsymbol{h}^M) = R(s, \boldsymbol{a}) \quad (4.13)$$

Equation 3.4.2

Equation 3.4.2 from [40] is shown in equation 4.14. Do note that the equations from the book are written here using the notation used in this thesis. E.g. joint actions are written in bold for equations cited from the book.

$$V^{\pi}(s, \boldsymbol{h}) = R(s, \pi(\boldsymbol{h})) + \sum_{s' \in S} \sum_{\boldsymbol{o'} \in \Omega} Pr(s', \boldsymbol{o'}|s, \pi(\boldsymbol{h})) V^{\pi}(s', \boldsymbol{h'}) \quad (4.14)$$

with the following as base case:

$$V^{\pi}(s^{h-1}, \boldsymbol{h}^{h-1}) = R(s^{h-1}, \pi(\boldsymbol{h}^{h-1})) \quad (4.15)$$

The $\boldsymbol{h}$ represents the joint observation history of the agents instead of a joint action-observation history. This is because the actions between observations in a history can be determined with certainty because deterministic policies are used here. Histories are fed into policies π to retrieve a joint action $\boldsymbol{a}$. I.e.:

$$\pi(\boldsymbol{h}) = \boldsymbol{a}$$

Thus, substituting $\boldsymbol{a}$ into this value function yields the following:

$$V^{\pi}(s, \boldsymbol{h}) = R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o}' \in \Omega} Pr(s', \boldsymbol{o}' | s, \boldsymbol{a}) V^{\pi}(s', \boldsymbol{h}') \tag{4.16}$$

with the following as base case:

$$V^{\pi}(s^{h-1}, \boldsymbol{h}^{h-1}) = R(s^{h-1}, \boldsymbol{a}^{h-1}) \tag{4.17}$$

Equation 4.1.3

Equation 4.1.3 from [40] is as follows:

$$V(s, q^{\tau}) = \begin{cases} R(s, \boldsymbol{a}) & \text{if } t = h - 1, \\ R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o}' \in \Omega} Pr(s', \boldsymbol{o}' | s, \boldsymbol{a}) V(s, q^{\tau} \Downarrow_o) & \text{otherwise} \end{cases} \tag{4.18}$$

A different notation is used here for the separation of the recursive and base cases. For completeness sake, this function can be rewritten like so:

$$V(s, q^{\tau}) = R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o}' \in \Omega} Pr(s', \boldsymbol{o}' | s, \boldsymbol{a}) V(s, q^{\tau} \Downarrow_o) \tag{4.19}$$

with the following base case:

$$V(s^{h-1}, q^{\tau}) = R(s^{h-1}, \boldsymbol{a}^{h-1}) \tag{4.20}$$

In equation 4.1.3, q^{τ} represents a sub-tree policy. The progression through this policy, by assuming that joint observation $\boldsymbol{o}$ is encountered, is represented like so: $q^{\tau} \Downarrow_o$. q^{τ} implicitly contains the information for both the histories, as well as the policies.

Connection

We can see that all three equations (MacDecPOMDP Value Function, equation 3.4.2 and 4.1.3) consist of first calculating:

$$R(s, \boldsymbol{a}) + \sum_{s' \in S} \sum_{\boldsymbol{o}' \in \Omega} Pr(s', \boldsymbol{o} | s, \boldsymbol{a})$$

This is followed by performing a recursive call. The only difference here is that equation 4.7 also contains the calculations for macro-actions/observations;

$$\sum_{\boldsymbol{z}' \in \zeta^*} \sum_{\boldsymbol{m}' \in M} Pr(\boldsymbol{z}', \boldsymbol{m}' | \boldsymbol{m}, \boldsymbol{h}^{A\prime}, \boldsymbol{h}^{M\prime}, s')$$

We can also see that the base cases for all three equations consist of just returning the reward for the given joint action at the last time-step; $R(s^{h-1}, \boldsymbol{a}^{h-1})$.
Finally, all three equations have a state and some form of histories and policies information in their value function calls. In equation 4.7 the histories/policies are represented by $\boldsymbol{m}$, $\boldsymbol{h}^A$ and $\boldsymbol{h}^M$. In equation 3.4.2, the policies/histories are represented by π and $\boldsymbol{h}$, and finally in equation 4.1.3, this is represented by q^{τ}.

4.10 Limited Abstraction

Abstraction is a core principle of this thesis. This is particularly clear in the research question:

(2) How can we extend the Mac-Dec-POMDP model to higher levels of abstraction?

Unfortunately, MacDecPOMDPs only supports a single level of abstraction. To achieve higher levels of abstraction, e.g. macro-actions that have policies for other macro-actions, etc, another model is needed. This is what the Hierarchical MacDecPOMDP model described in chapter 5 provides.

5 Hierarchical MacDec-POMDP

This chapter answers the following research question taken from section 1.5;

How can the MacDecPOMDP Framework be extended to higher levels of abstraction?

In order to add layers of abstraction of planning to MacDecPOMDPs, the model must be extended. Extending the model is not trivial, thus a new recursively hierarchical variant of the MacDecPOMDP model (H-MacDecPOMDP) will be given in this section. This is a model with an arbitrary amount of abstraction levels in its hierarchy. This model uses macro-actions, which are actions that execute a lower level policy when the macro-action is taken. This makes macro-actions in Hierarchical MacDecPOMDP different to those in MacDecPOMDP, as MacDecPOMDP have policies that plan over ground-level actions, instead of other macro-actions. The arbitrary amount of abstraction levels is achieved by allowing macro-actions to contain policies of other lower level macro-actions which themselves consist of policies of even lower level macro-actions. The lowest level of macro-actions will then lead to the execution of ground level actions. Section 7.1 provides an example of an H-MacDecPOMDP which can be used to help understand the model presented in this chapter (5). This section contains the following:

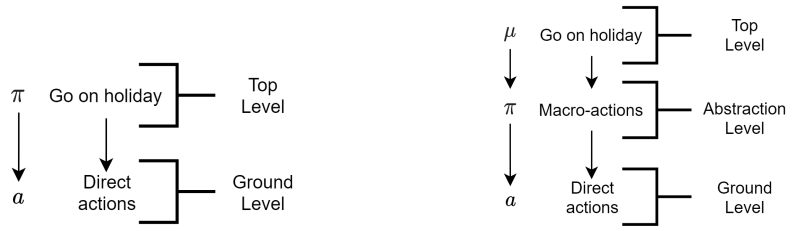
- A conceptual explanation of abstraction in Hierarchical MacDecPOMDPs.
- A definition of Hierarchical MacDecPOMDPs.
- An explanation of Hierarchical MacDecPOMDPs and its value function.
- A proof of correctness for the H-MacDecPOMDP value function.

5.1 Conceptual Idea

Human planning is often done explicitly in different levels of abstraction. For example, consider a plan for a holiday trip. First on the highest abstraction level, the decision is made to plan the trip, this is done using a history of macro-observations and macro-actions in their own personal life. An example of such macro-observations is noticing that they are often tired while having sufficient sleep and thus construing the macro-observation that they are overworked.
In this human example, ground-level observations would be observations gathered from human senses; i.e. smell, touch, sight, kinesthetic, hearing or taste. While macro-observations would be observed by looking at the history of lower level observations (or actions).
After the decision to go on a trip is made, i.e. after inserting the high level history containing the 'overworked' macro-observations into a top-level policy and choosing the macro-action to go on a holiday; The main holiday activity is chosen such as skiing or lying on the beach. These decisions are also made using a history of macro-observation/actions(s) such as noticing that the last time they chose a particular type of holiday, it was effective in replenishing their motivation or happiness. Then to go down further in plan abstractions, the duration of the holiday is chosen, the location and all the activities are chosen. Then comes the logistics, and the plan of leaving for this holiday which involves moving to the correct locations and interacting with other people is executed. This would lead to the ground level plan of moving muscles and perceiving human senses.
Every task here has its own subtasks, it seems to be a naturally recursive process. This recursive planning process is what we aim to emulate using the H-MacDecPOMDP model.

5.2 Abstraction Levels

To better understand the notion of multiple abstraction levels, we should take a closer look at abstraction levels in Dec-POMDPs and MacDecPOMDPs.



(a) DecPOMDPs (b) Mac-DecPOMDPs

Figure 3: Overview of the (lack of) abstraction levels.

This thesis uses l to denote the amount of abstraction levels a model. Not every level is an abstraction level. A MacDecPOMDP has 3 levels, of which 1 abstraction level ($l = 1$). As shown in figure 3a, there are no levels of abstraction between the top and ground level in DecPOMDPs ($l = 0$). The actions are chosen using a top level policy of ground level actions. As shown in figure 3b, Mac-DecPOMDPs have one level of abstraction ($l = 1$) as macro-actions have policies which choose ground-level actions. H-MacDecPOMDPs can have unlimited levels of abstraction, as macro-actions recursively have policies leading to other macro-actions until ground-level actions are reached. This is shown in figure 4.

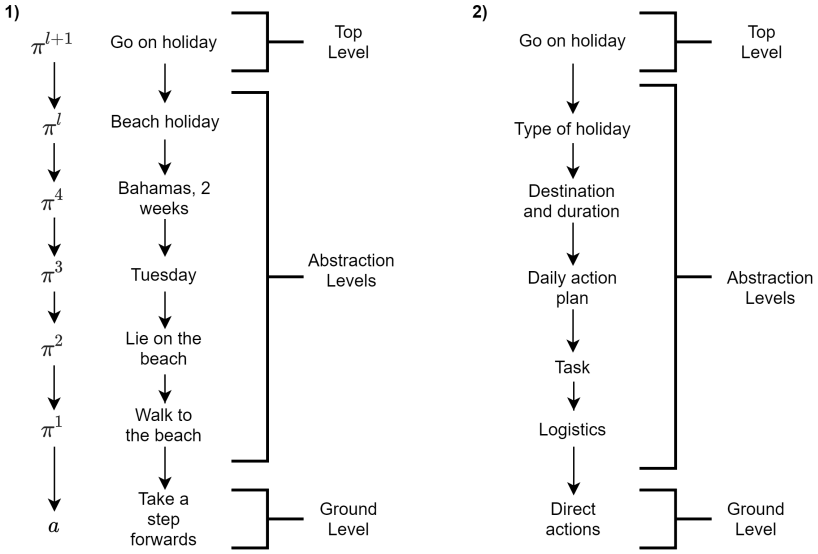


Figure 4: Overview of the of abstraction levels in H-MacDecPOMDPs ($l = 5$). (1) shows an example of the state of the abstraction levels during execution. And (2) shows the more general descriptions of the abstraction levels.

5.3 Macro Action Termination

Macro-actions in abstraction levels do not execute indefinitely. If they did, all of the abstraction levels above it would become meaningless. For example, when on a holiday, you might walk on the beach, but you do not do this forever. Eventually you stop and grab dinner, or go on to do another activity. The macro-action of walking on the beach eventually terminates, and the abstraction level above it should be used to choose a new macro-action.
Termination of macro-actions cannot happen arbitrarily. Thus it is impossible for a macro-action to terminate while macro-actions in levels below it do not terminate. Intuitively, the reason for that design choice is that subtasks (lower level macro-actions) that are meant to complete a macro-action, are no longer relevant when a higher level macro-action does not need to be completed anymore or has already been completed.
Thus termination happens in a bottom-to-top fashion, as shown in figure 5. In every timestep, the first level macro-action checks whether it should terminate based on the actions and observations on the level below. If it does not terminate, no other macro-action can terminate. If it does terminate, we must recursively consider the abstraction levels above until we reach an abstraction level on which the macro-action does not terminate. This always happens because the top level macro-action never terminates. This whole system of termination is walked through in every timestep of planning and executing policies.
The ground level action always terminates because it always takes one time-step to complete. This action is akin to taking a step or saying a word in a real world example.

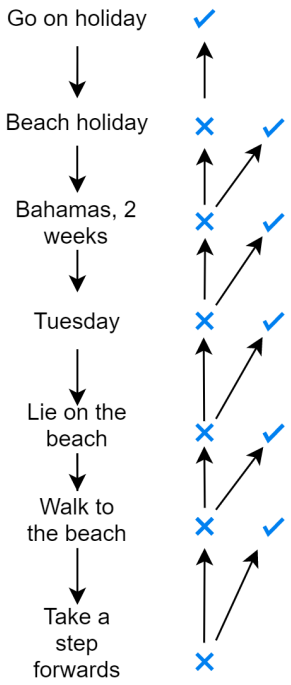


Figure 5: Overview of the macro-action termination. The cross represents macro-action termination, and the checkmark represents non-termination.

5.4 Higher Level Histories

The histories of the ground level are not different from those in DecPOMDPs. However, the histories on levels above the ground level are different, as they do not exist in DecPOMDPs. Macro-histories contain macro-actions and macro-observations instead of ground-level actions and observations. The higher level histories are shorter (or equal in length if each macro-action leads to a single action on the level below). It is impossible within an agent, for a higher level history to be longer than a history of a level under it. Although there are differences in length, i.e. amount of macro-actions/observations per abstraction level, the amount of ground level time-steps that have passed between these histories remains equal. This can also be seen in 6.
The macro-observations after each macro-action depend on the histories up till that point of time of the level under it. Thus it is only the observations on the ground level that directly depend on the underlying state. Because the macro-observations only depend on the history under this macro-observation, macro-observations only indirectly depend on the state of the environment.

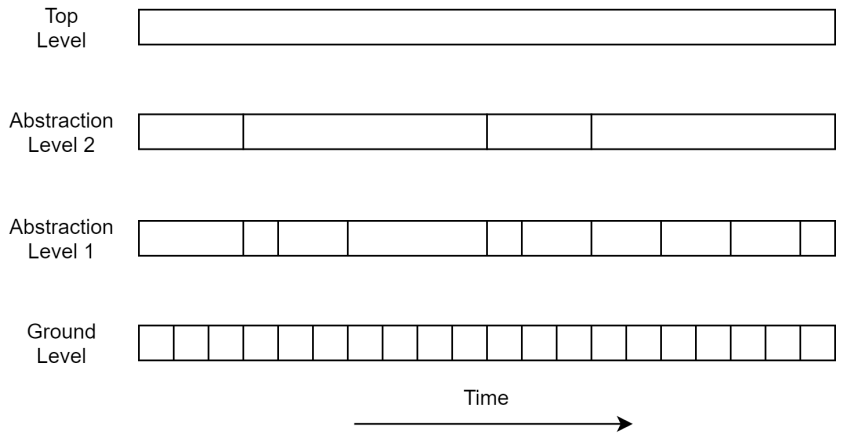


Figure 6: Overview of histories for an agent. Each layer represents a (macro-)action-(macro-)observation history. Each rectangle/square represents a (macro-)action, while each border between two rectangles/squares represents a (macro-)observation.

You may have noticed in figure 6 that the top level macro-action never terminates. This is because the top level macro-action is more or less just a top-level policy. An example of a history of abstraction level j and agent i is given in equation 5.1, here θ_i^0 and θ_i^1 are respectively equivalent to h_i^A and h_i^M of MacDecPOMDPs. θ is used instead of h to avoid confusion with the horizon of the problem (h).

$$\theta_i^{j\prime} = [\theta_i^j, m_i^j, z_i^{j\prime}] \tag{5.1}$$

5.5 Utilizing *null*-Observations for Termination

Similar to the approach in section 4.9, *null* observations are used to denote that the last macro-action has not terminated. However, in H-MacDecPOMDPs *null* can also be found at the end of a history. Adding the *null* observations to histories enables histories to signify macro-action persistence. This simplifies the transformation from H-MacDecPOMDPs to DecPOMDPs described in chapter 6. It also allows the calculations of the value function relating to actions and macro-actions to be grouped, which in turn leads to a simpler formulation.

As became apparent in section 5.4, the top level macro-action never terminates and is always present. Thus the history the value function is initially executed with; the *initial history*, consists of an empty joint history for every level, except for the top level (level $l+1$). This level will always contain the macro-action which contains the top level policy, which was called μ in MacDecPOMDPs. This top level macro-action is also always non-terminating, thus it is always unchanged and followed by a *null*-observation.

$$\begin{aligned} \theta_i^4 &= (m_i^4, null) \\ \theta_i^3 &= () \\ \theta_i^2 &= () \\ \theta_i^1 &= () \\ \theta_i^0 &= () \end{aligned}$$

$$i \in I$$

Figure 7: An overview of the initial history ($\boldsymbol{\theta}^{start}$), with $l = 3$.

5.6 Full Formal Definition

This section consists of the formal definition of H-MacDecPOMDPs. Every component in H-MacDecPOMDPs are given and briefly described. $Model = \langle I, S, b_0, T, R, O, M, \Omega, Z, h, l \rangle$ where;

I : the finite set of n agents. e.g.: $\{1, 2, 3, ...n\}$

S : the set of states. e.g.: $\{state1, state2, ...\}$

b_0 : the distribution of the initial states; $S \rightarrow [0, 1]$.

T : the state transition probability function holds the probabilities of reaching any state s' from any state s by having the agents perform a ground-level joint action $\boldsymbol{a}$.
($T : S \times A \times S \rightarrow [0, 1]$ and $T(s'|s, \boldsymbol{a}) = Pr(s'|s, \boldsymbol{a})$).

R : the reward function. $R(s, \boldsymbol{a})$ is the reward given for a joint ground-level action $\boldsymbol{a}$ when in a state s. ($R : S \times A \rightarrow \mathbb{R}$)

O : the set of joint observation probability functions for level 0 holds the probabilities of getting the joint observation $\boldsymbol{o}$ from reaching state s by having the agents perform a joint ground action $\boldsymbol{a}$. ($O : A \times S \times \Omega^0 \rightarrow [0, 1]$ and $O(\boldsymbol{o}|\boldsymbol{a}, s) = Pr(\boldsymbol{o}|\boldsymbol{a}, s)$)

M_i^j : the set of (macro)-actions for agent i on level j. Where $m_i^j \in M_i^j$.

- Macro-actions consist of three parts: $m_i^j = (\beta_{m_i^j}, \mathcal{I}_{m_i^j}, \pi_{m_i^j}) \in M_i^j$

  $\mathcal{I}_{m_i^j}$ The initiation conditions, the set of macro-action/observation histories in which the corresponding agent (i) must be to start performing macro action m_i^j. ($\mathcal{I}_{m_i^j} \subseteq H_i^j$)

  $\beta_{m_i^j}$ The termination conditions, the probability of terminating a running macro-action, given θ_i^{j-1} on agent i and level j ($\beta_{m_i^j} : H_i^{j-1} \rightarrow [0, 1]$).

  $\pi_{m_i^j}$ The action policy of a macro-action on agent i and level j. I.e. the probability of taking an action given a history of actions and observations. ($\pi_{m_i^j} : H_i^{j-1} \times M_i^{j-1} \rightarrow [0, 1]$)

Ω_i^j : the set of (macro)-observations for agent i on level j. E.g. $z_i^j \in \Omega_i^j$.
If an asterisk is put above such a set (e.g. $\overset{*}{\Omega}$), then $null$ is included in the sets of observations in its abstraction/top levels. Thus for $\overset{*}{\Omega}$: $\underset{i \in I, j \in \{1, ...l+1\}}{\forall} null \in \overset{*}{\Omega}_i^j$ holds.

Z_i^j : the set of observation probability functions for level j (> 0) on agent i holds the probabilities of getting the observation z_i^j from history θ_i^{j-1}.
($Z_i^j : H_i^{j-1} \times M_i^j \times \Omega_i^j \rightarrow [0, 1]$ and $Z_i^j(z_i^j|m_i^j, \theta_i^{j-1}) = Pr(z_i^j|m_i^j, \theta_i^{j-1})$)

h : the problem horizon, the amount of steps (primitive actions executions), before the problem terminates.

l : the amount of abstraction levels in the macro-action/observation hierarchy. As written in section 5.2, l excludes both the ground level (level 0), and the top level ($l + 1$). There are thus $l + 2$ levels in total. L represents the set of all levels from 0 up to and including l ($L = \{0, ..., l\}$), i.e. all levels excluding the top level.

5.6.1 Histories and Outer Level Macro-actions

Histories in H-MacDecPOMDPs are similar to those in MacDecPOMDPs, except numbers are used instead of letters to denote the levels, E.g. $H^0 = H^A$. And as discussed in section 5.5, at most a single *null* observation can be present in macro-histories (histories of all levels except the ground level). If a *null*-observation is present, it can only be at the end of a macro-history.
The top level macro-action and the ground level action have trivial initiation and termination sets. The top and ground level actions can be initiated from all histories ($\mathcal{I}_{m_i^{l+1}} = H_i^{l+1}$ for all agents i). The ground level actions always terminate ($\beta_{m_i^0} \to 1$ for all agents i). The ground level action has a trivial policy which only executes its ground level action in the environment.

5.6.2 Notation Comments

There are two useful ways to group (macro-)actions, (macro-)observations and (macro-)histories, namely by agent or by levels. To reiterate; prefix *joint* is used to refer to grouping all these elements for all agents, e.g. m^1 is the level 1 joint macro-action, or in other words; the selected macro-actions for every agent on level 1.
The *stack* suffix is used to refer to grouping elements for every level, e.g. m_1 is the action stack of agent 1, or in other words; the selected macro-actions for every level of agent 1.
Finally, these two terms can also be combined. When the two are combined, the bold notation is used to accentuate that the element is a joint stack. The joint and stack prefixes can also be used for (macro-)observations and (macro-)histories. The same notation can also be used for their sets, e.g. M^2 is the set of level 2 joint macro-actions.

$$m^0 = \boldsymbol{a} \qquad (5.2) \qquad\qquad z^0 = \boldsymbol{o} \qquad (5.3)$$

The bold notation is also used for joint ground level actions ($\boldsymbol{a}$) and observations ($\boldsymbol{o}$), where $\boldsymbol{a} = m^0$ and $\boldsymbol{o} = z^0$. As the ground level actions and observations can be represented using the letters a and o respectively as shown in equations 5.2 and 5.3. Refer to equations 5.4, 5.5 and 5.6 for examples of a joint stack macro-actions, a joint macro-action and a macro-action stack respectively.

$$\boldsymbol{m} = \begin{vmatrix} m_1^3 & m_2^3 & m_3^3 \\ m_1^2 & m_2^2 & m_3^2 \\ m_1^1 & m_2^1 & m_3^1 \\ m_1^0 & m_2^0 & m_3^0 \end{vmatrix} \qquad (5.4) \qquad m^1 = \begin{vmatrix} m_1^1 & m_2^1 & m_3^1 \end{vmatrix} \qquad (5.5) \qquad m_2 = \begin{vmatrix} m_2^3 \\ m_2^2 \\ m_2^1 \\ m_2^0 \end{vmatrix} \qquad (5.6)$$

5.7 Value Function Overview

In order to check the effectiveness of the chosen policies of the agents, a value function is needed to evaluate policies. This value function should take the state of the world, the histories so far and the policies of the agents as inputs. With this information the value function will return the expected value of the policies of the model. The state is needed so that the effects and rewards of the actions of the agents can be evaluated. The histories are needed to determine what actions will be chosen according to the policies of the agents.

5.7.1 Overview Explanation

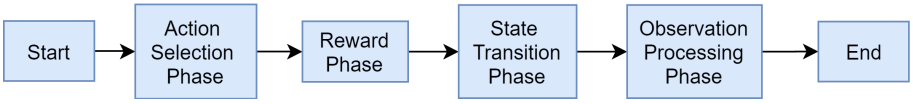


Figure 8: High-level overview of the execution of a single time-step in a H-MacDecPOMDP Value Function.

The value function explained in this chapter (equation 5.8) will calculate the expected value which consists of the probability-adjusted sum of all (future) rewards. In each time-step the same calculations will recursively perform until the final timestep $(h-1)$ that we want to consider is reached. Thus the calculation of a single time-step in the value function boils down to the following (equation 5.7):

$$\begin{aligned} Value(\text{state}, \text{histories}) = {} & (\text{For every joint-action-stack} \\ & \text{calculate its probability} \\ & \times (\text{the reward given by taking this action}) \\ & + \ \text{For every resulting state} \\ & \text{and each resulting joint-observation-stack,} \\ & \text{calculate their probabilities} \\ & \times \text{the expected value of the next time-step}) \end{aligned} \tag{5.7}$$

The base case of this value function is identical to that of the MacDecPOMDP value function.
A diagram for the overview of the value function can be found on figure 8 and 9. The inner workings of this value function will be thoroughly explained in sections 5.8 to 5.12, the figure will then also be thoroughly explained. The value function for Hierarchical H-MacDecPOMDPs is shown here (5.8):

$$V_t^\pi(s, \boldsymbol{\theta}) = \sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \overset{*}{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V_{t+1}^\pi(s', \boldsymbol{\theta}') \right] \tag{5.8}$$

Each phase of figure 8 corresponds to a number of the lines of equation 20. Each phase is matched with a part of equation 5.7 below. Aside from this, the link between these phases and the parts of the actual value function is also be shown here. Once again, note that the details of these equations will be explained in sections 5.8 to 5.12.

- Start: $Value_t(\text{state}, \text{histories})$: $V_t^\pi(s, \boldsymbol{\theta})$
- Action Selection Phase: *For every joint-action-stack calculate its probability*: $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta})$
- Reward Phase: $\times$ *the reward given by taking this action*: $R(s, \boldsymbol{a})$
- State Transition Phase: *For every resulting state*: $\sum_{s' \in S}$
- Observation Processing Phase: *and each resulting joint-observation-stack,*: $\sum_{\boldsymbol{z}' \in \overset{*}{\Omega}}$
- For both the state and observation Phases: *calculate their probabilities*: $T(s'|s, \boldsymbol{a}) Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s')$
- End: $\times$ *the expected value of the next time-step*: $V_{t+1}^\pi(s', \boldsymbol{\theta}')$

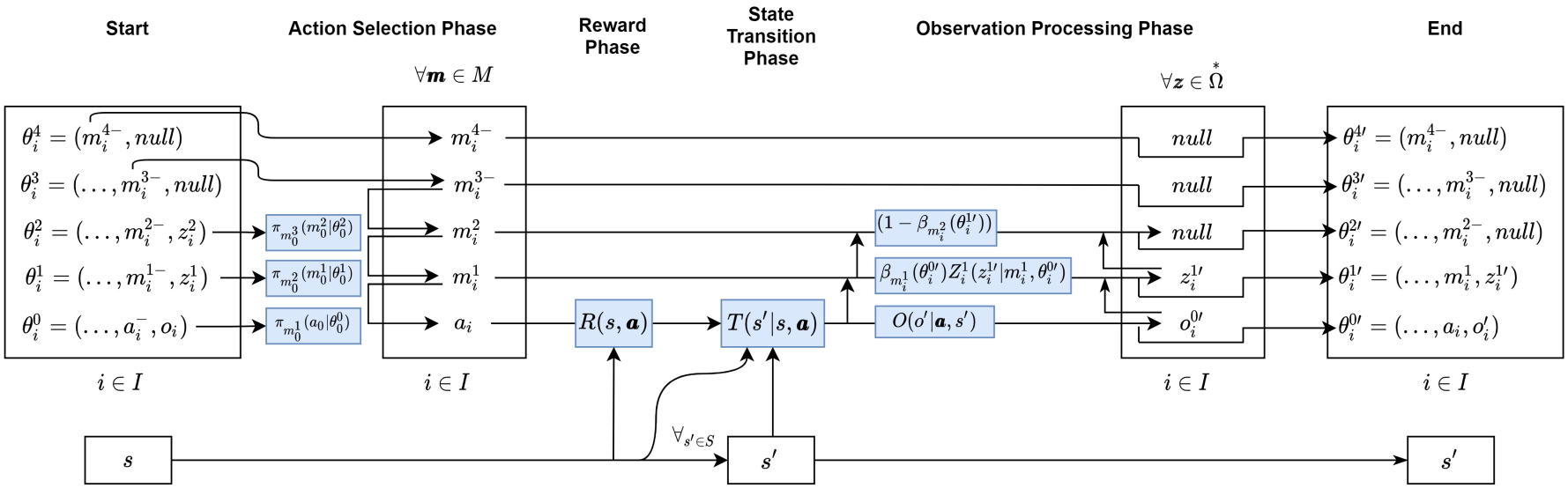


Figure 9: This diagram shows an overview of the execution of a single time-step in the H-MacDecPOMDP value function. In this case there are 3 abstraction levels ($l = 3$) where for an agent i, first abstraction layer one and two terminate, and later just the first terminates. The minus superscript (e.g. m_i^{1-}) denotes that the (macro-) action is from the previous (time-)step.

5.8 Action Selection Phase

In the Action Selection Phase, all joint action stacks are looped over. This is shown in equation 5.9.

$$\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \tag{5.9}$$

To reiterate; a joint action-stack, written as $\boldsymbol{m}$, represents a configuration of (macro-)actions that are chosen for each level and each agent. In figure 10, the right rectangle is meant to represent a joint action-stack, the action-stacks of the other agents in this figure can be imagined as being 'behind' the action-stack of agent i. The $i \in I$ expression should help make this more clear, as I represents the set of agents, which are numbered starting from 1.
The probability of executing a joint action stack is calculated using equation 5.10. Here we can see that the probabilities of actions for every agent and every level are multiplied together.

$$Pr(\boldsymbol{m}|\boldsymbol{\theta}) = \prod_{i \in I} \prod_{j \in L} Pr(m_i^j | \theta_i^j, m_i^{j+1}) \tag{5.10}$$

In order to calculate the probability of taking a macro-actions, equation 5.11 is used.

$$Pr(m_i^j | \theta_i^j, m_i^{j+1}) = \begin{cases} 1 & \text{If } z_i^j = null \wedge m_i^j = m_i^{j,t-1} \\ \pi_{m_i^{j+1}}(m_i^j | \theta_i^j) & \text{Else if } z_i^j \neq null \wedge \theta_i^j \in \mathcal{I}_{m_i^j} \\ 0 & \text{Otherwise} \end{cases} \tag{5.11}$$

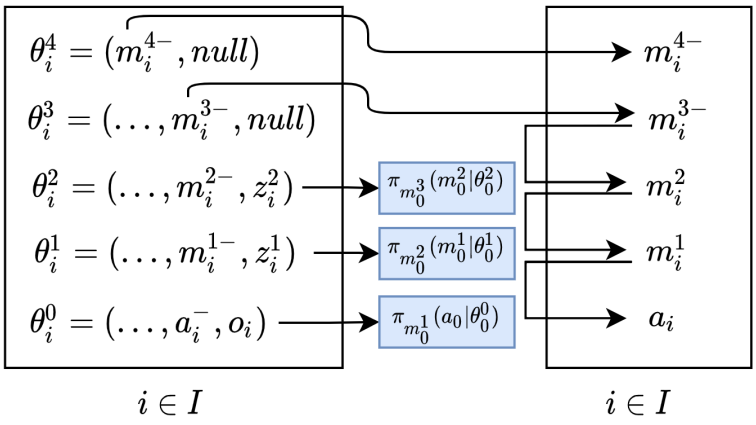


Figure 10: An Overview of the Action Selection Phase with agent 0 and $l_0^T = 2$, excluding the probability calculations.

The minus symbol ($-$) in the superscript of (macro-)actions in figure 10 denotes that it is from the previous time-step. Macro-actions m_i^{4-} and m_i^{3-} persist (i.e. do not terminate). Thus these macro-actions must also be used in the current time-step. This is reflected in the first case of equation 5.11, where the probability of taking a macro-action equals 1 when the macro-action persists. This can also be seen in figure 10, here m_i^{4-} and m_i^{3-} are reused.
Recall that persisting macro-actions are followed by *null*-observations. The rest of the (macro-)actions must then be considered. All terminating macro-actions must be followed by a macro-action that adheres to the initiation set constraint ($\mathcal{I}_{m_i^j}$) which is also present in MacDecPOMDPs. Recall that this constraint states that the current history must be an element of the initiation set of the chosen macro-action ($\theta_i^j \in \mathcal{I}_{m_i^j}$). If any macro-action does not adhere to either of these cases, the chosen joint action stack is invalid. Invalid macro-actions have a probability of 0 to occur, because in the third case of equation 5.11 multiplies the probability with 0.

5.9 Observation Processing Phase

Similarly to the action selection phase, the observation processing phase loops over all possible joint observation-stacks and their probabilities of occurring. This is expressed in equation 5.12.

$$\sum_{\boldsymbol{z}' \in \overset{*}{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') \tag{5.12}$$

This probability is calculated using equation 5.13.

$$Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') = O(\boldsymbol{o}'|\boldsymbol{a}, s') \prod_{i \in I} \prod_{j \in \{1,\ldots,l\}} Pr(z_i^{j\prime}|m_i^j, \theta_i^{j-1\prime}) \tag{5.13}$$

Here the probability of the ground level joint observation to occur is first calculated using $O(\boldsymbol{o}'|\boldsymbol{a}, s')$. This is also shown in figure 11 for observation $o_i^{0\prime}$, but here it must be remembered that the ground level observations of the other agents is also put into $O(\boldsymbol{o}'|\boldsymbol{a}, s')$.
Afterwards, this probability is multiplied with the probabilities of all other macro-observations for each agent and each abstraction level. The probability of a macro-observation to occur is calculated using equation 5.14.

$$Pr(z_i^{j\prime}|m_i^j, \theta_i^{j-1\prime}) = \begin{cases} 1 & \text{If } z_i^{j-1\prime} = null \wedge z_i^{j\prime} = null \\ 0 & \text{Else if } z_i^{j-1\prime} = null \wedge z_i^{j\prime} \neq null \\ (1 - \beta_{m_i^j}(\theta_i^{j-1\prime})) & \text{Else if } z_i^{j-1\prime} \neq null \wedge z_i^{j\prime} = null \\ \beta_{m_i^j}(\theta_i^{j-1\prime}) Z_i^j(z_i^{j\prime}|m_i^j, \theta_i^{j-1\prime}) & \text{Else if } z_i^{j-1\prime} \neq null \wedge z_i^{j\prime} \neq null \end{cases} \tag{5.14}$$

The probability calculation depends on whether the macro-observation of the current level and the level below equal null. In figure 11 it can be seen that any null observation above another null observation does not have a probability calculation associated with it, i.e. it has probability 1. This is because any macro-action above a persisting macro-action must also persist. This is reflected in the first two cases of equation 5.14, where a probability of 1 is given if a null-observation occurs above another null-observation, and a probability of 0 is given when the macro-observation above a null-observation is not equal to null. This also avoids the issue shown in figure 12, where the invalid situation is shown where a non-null macro-observation is sandwiched between two null-observations.

If the macro-observation is not *null*, and the (macro-)observation under it is also not null, the current macro-action terminates while it was not forced to terminate. If a macro-action terminates, a non-null macro-observation must be chosen. Thus the probability of obtaining that macro-observation is equal to the probability of taking that macro-action termination times the probability of newly receiving that macro-observation. This is written as: $\beta_{m_i^j}(\theta_i^{j-1\prime}) Z_i^{j\prime}(z_i^{j\prime}|m_i^j, \theta_i^{j-1\prime})$. This is reflected in the third case of equation 5.14 and macro-observation $z_i^{1\prime}$ in figure 11.
If the macro-observation is equal to *null* and the (macro-)observation under it is also not equal to null, the macro-action does not terminate while it was not forced to. If a macro-action does not terminate, the null-observation is always given. Thus the probability of receiving this observation equals the probability of termination of the macro-action. Thus this probability equals: $(1 - \beta_{m_i^j}(\theta_i^{j-1\prime}))$. This is reflected in the fourth case of equation 5.14 and the null-observation above $z_i^{1\prime}$ in figure 11.

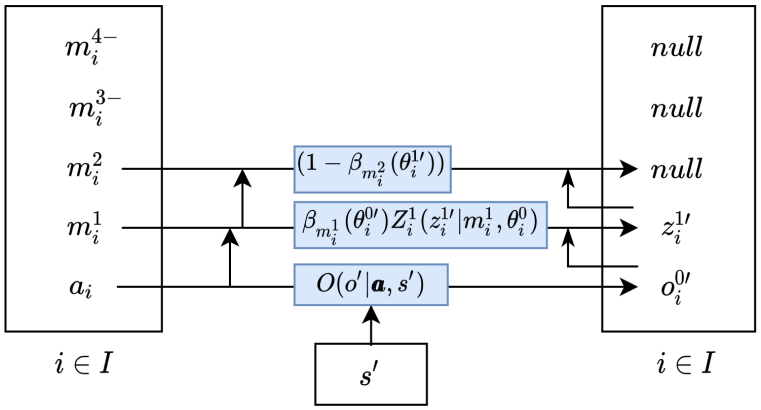


Figure 11: An overview of an example of the calculations of probabilities of (macro-)observations during the observation processing phase for an agent i.

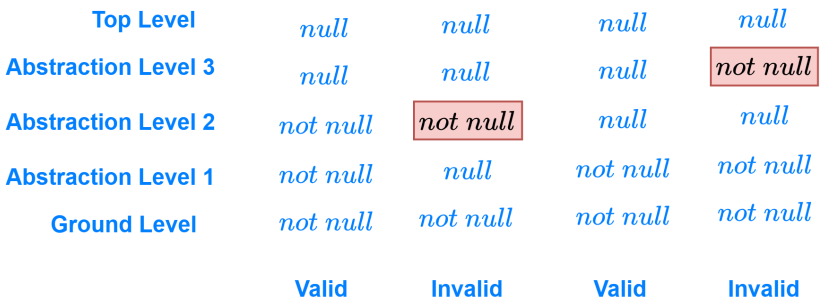


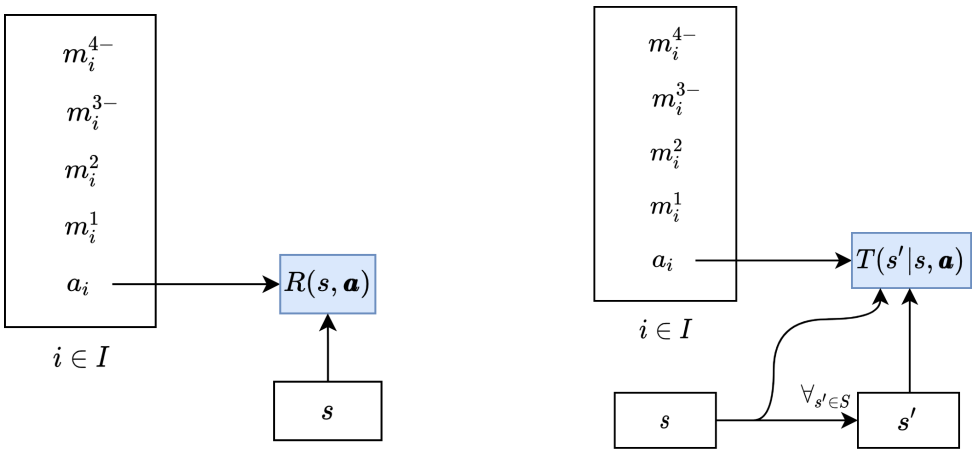
Figure 12: This diagram shows examples of valid and invalid termination configurations for observation stacks.

5.10 Reward Determination

As is done in DecPOMDPs, the reward is determined by taking the joint ground-level action, and inserting this into the reward function along with the current state. There also is no difference between this and how it is done in MacDecPOMDPs.

$$R(s, \boldsymbol{a})$$

In diagrams 9 and 13a, this is represented by drawing arrows from the ground-level actions to $R(s, \boldsymbol{a})$. Along with an arrow from the current state to $R(s, \boldsymbol{a})$.



(a) Reward phase.

(b) Transition phase.

Figure 13: Overview of the Transition and Reward Phases of H-MacDecPOMDP.

5.11 State Transition

The transition from one state to another in Hierarchical MacDecPOMDPs is also done in the same manner as how this is done in Dec-POMDPs and MacDecPOMDPs. First all possible next states are considered: $\sum_{s' \in S}$, and then the probability of making this transition is determined: $T(s'|s, \boldsymbol{a})$.

$$\sum_{s' \in S} T(s'|s, \boldsymbol{a})$$

In figure 13b we can see that first all possible next states are considered with this expression: $\forall_{s' \in S}$. With the old state, the new state and the ground-level action, the probability of making this state transition is determined. This is done in $T(s'|s, \boldsymbol{a})$, using the information that is gathered from the arrows leading into it.

5.12 Value Function Recursion

The input to the next recursive call in the value function consists of the next state and next history, as can be seen in figure 14. This is shown in equation 5.15.

$$V_{t+1}^{\pi}(s', \boldsymbol{\theta}') \tag{5.15}$$

The next state is determined in the transition phase. The next history is constructed by combining the previous history and the current joint action stack and next joint observation-stack using formula 5.16.

$$\theta_i^{j\prime} = \begin{cases} \text{replace } \mathit{null} \text{ in } \theta_i^j \text{ with } z_i^{j\prime} & \text{if } \theta_i^j = (..., \mathit{null}) \\ (\theta_i^j, m_i^j, z_i^{j\prime}) & \text{otherwise} \end{cases} \tag{5.16}$$

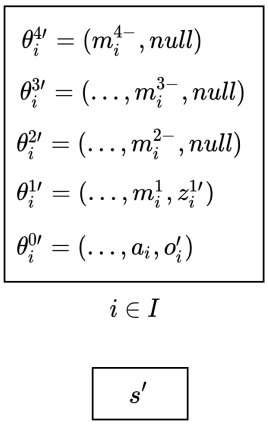


Figure 14: An overview of the input of $V_{t+1}^{\pi}(s', \boldsymbol{\theta}')$.

Equation 5.16 must be performed for each level j and agent i to construct the next joint history-stack.
In the first of the two cases the macro-action of the previous time-step for agent i and level j persisted. This resulted in a history which ends with *null*-observation (i.e. which has a *null*-observation at the current time-step). To construct the next history in this case, the *null*-observation must be replaced with the (macro-)observation of the next time-step. Do note that this new (macro-)observation can in turn also be a *null*-observation if it still did not terminate.
In the other case, the current (macro-)action and the next (macro-)observation can just be concatenated to the history.
Do note that a component with a prime (e.g. $z_i^{j\prime}$) denotes the next time-step, and without a prime denotes the current time-step.

5.13 H-MacDec-POMDP Value Function Formulation

Here we find a complete overview of the value function and its helper functions.
As you may recall from earlier in this chapter, null macro-actions/observations are used to denote non-terminating macro-actions. To reiterate; the $*$ in $\overset{*}{\Omega}$ indicates that a *null* entry has been added to every abstraction- and top-level macro-observation set (Ω) for every agent.

$$V_t^{\pi}(s,\boldsymbol{\theta}) = \sum_{\boldsymbol{m}\in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s,\boldsymbol{a}) + \sum_{s'\in S} T(s'|s,\boldsymbol{a}) \sum_{\boldsymbol{z}'\in \overset{*}{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta},\boldsymbol{m},s') V_{t+1}^{\pi}(s',\boldsymbol{\theta}') \right]$$

Where:

$$Pr(\boldsymbol{m}|\boldsymbol{\theta}) = \prod_{i\in I} \prod_{j\in L} Pr(m_i^j|\theta_i^j, m_i^{j+1})$$

$$Pr(m_i^j|\theta_i^j, m_i^{j+1}) = \begin{cases} 1 & \text{If } z_i^j = null \wedge m_i^j = m_i^{j,t-1} \\ \pi_{m_i^{j+1}}(m_i^j|\theta_i^j) & \text{Else if } z_i^j \neq null \wedge \theta_i^j \in \mathcal{I}_{m_i^j} \\ 0 & \text{Otherwise} \end{cases}$$

$$Pr(\boldsymbol{z}'|\boldsymbol{\theta},\boldsymbol{m},s') = O(\boldsymbol{o}'|\boldsymbol{a},s') \prod_{i\in I} \prod_{j\in\{1,\ldots,l\}} Pr(z_i^{j\prime}|\theta_i^{j-1\prime}, m_i^j)$$

$$Pr(z_i^{j\prime}|\theta_i^{j-1\prime}, m_i^j) = \begin{cases} 1 & \text{If } z_i^{j-1\prime} = null \wedge z_i^{j\prime} = null \\ 0 & \text{Else if } z_i^{j-1\prime} = null \wedge z_i^{j\prime} \neq null \\ (1 - \beta_{m_i^j}(\theta_i^{j-1\prime})) & \text{Else if } z_i^{j-1\prime} \neq null \wedge z_i^{j\prime} = null \\ \beta_{m_i^j}(\theta_i^{j-1\prime}) Z_i^j(z_i^{j\prime}|m_i^j, \theta_i^{j-1\prime}) & \text{Else if } z_i^{j-1\prime} \neq null \wedge z_i^{j\prime} \neq null \end{cases}$$

$$\theta_i^{j\prime} = \begin{cases} \text{replace } null \text{ in } \theta_i^j \text{ with } z_i^{j\prime} & \text{If } \theta_i^j = (\ldots, null) \\ (\theta_i^j, m_i^j, z_i^{j\prime}) & \text{Otherwise} \end{cases}$$

Base case:

$$V_{h-1}^{\pi}(s,\boldsymbol{\theta}) = \sum_{\boldsymbol{m}\in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s,\boldsymbol{a})$$

5.14 H-MacDecPOMDP Value Function Proof

This section contains a proof by induction of the value function given in section 5.13, does indeed return the expected value of the given joint top level policy.
This proof consists of two parts, the base case and the inductive step. In the base case, the value for the last time step ($t = h - 1$) is proven to be equal to:

$$\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s, \boldsymbol{a})$$

Then in the inductive step, the value for the remaining time-steps (t) is proven to be equal to:

$$\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \vec{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V^{\pi}_{t+1}(s', \boldsymbol{\theta}') \right]$$

5.14.1 Base Case

Lemma 5.1. *For the last time step $t = h-1$, the value ($V^{\pi}_{h-1}(s, \boldsymbol{\theta})$) is given by $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s, \boldsymbol{a})$.*

Proof. For the last time step the value is determined only by the reward function, thus it depends on the state the agent is in and the joint ground level action that is taken. Given the joint stack of histories ($\boldsymbol{\theta}$), the execution of the joint action stack ($\boldsymbol{m}$) depends on $Pr(\boldsymbol{m}|\boldsymbol{\theta})$ as defined in equation 5.10. The reward must be defined as $R(s, \boldsymbol{a})$ given the joint ground level action $\boldsymbol{a}$, defined as $a = m^0$ in equation 5.2. $\sum_{\boldsymbol{m} \in M}$ must be performed if every possible joint action stack must be considered. Putting this together, we get: $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s, \boldsymbol{a})$. □

5.14.2 Inductive Step

Lemma 5.2. *For the remaining time-steps $0 < t < h - 1$, the value ($V^{\pi}_{t}(s, \boldsymbol{\theta})$) is given by:*
$\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \vec{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V^{\pi}_{t+1}(s', \boldsymbol{\theta}') \right]$.

Proof. Per induction hypothesis, assume that the value of the next timestep (t + 1) is given by:

$$V^{\pi}_{t+1}(s, \boldsymbol{\theta}) = \sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \vec{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V^{\pi}_{t+2}(s', \boldsymbol{\theta}') \right]$$

We can divide this into two terms, the immediate reward $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s, \boldsymbol{a})$ and the future value, $\sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \vec{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V^{\pi}_{t+1}(s', \boldsymbol{\theta}')$. Here $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) R(s, \boldsymbol{a})$ is constructed in the same way as in the proof of lemma 5.2. The future value is based on the state that we reach after taking joint ground level action ($\boldsymbol{a}$) and the next stack of histories ($\boldsymbol{\theta}$).
Given the state s, the next state s' depends on $T(s'|s, \boldsymbol{a})$. This must be done for every next state, which must be done using: $\sum_{s' \in S}$. Given that the joint ground level observations (z^0) depend on $\boldsymbol{a}$ and s', and given that the macro-observations ($\boldsymbol{z}^{1,\ldots,l}$) depend on $\boldsymbol{m}^{1,\ldots,l}$ and $\boldsymbol{\theta}^{1,\ldots,l}$, the probability of $\boldsymbol{z}$' is $Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s')$ as defined in equation 5.13. This must be done for every next joint observation stack: $\sum_{\boldsymbol{z}' \in \Omega}$.
Given that every time-step must be considered, the value of the next time-step is determined: $V^{\pi}_{t+1}(s', \boldsymbol{\theta}')$.
Putting this together, we get: $\sum_{\boldsymbol{m} \in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \left[R(s, \boldsymbol{a}) + \sum_{s' \in S} T(s'|s, \boldsymbol{a}) \sum_{\boldsymbol{z}' \in \vec{\Omega}} Pr(\boldsymbol{z}'|\boldsymbol{\theta}, \boldsymbol{m}, s') V^{\pi}_{t+1}(s', \boldsymbol{\theta}') \right]$.
□

6 Solving H-MacDecPOMDPs

This chapter is about solving H-MacDecPOMDPs. First a brute force solution is briefly described, and afterwards a research question related to solving H-MacDecPOMDPs is answered. This research questions is:

Can H-MacDecPOMDPs be transformed into DecPOMDPs?

As stated in section 1.5, this research question is relevant to solving H-MacDecPOMDPs because it allows existing Dec-POMDP solving methods to be used to solve H-MacDecPOMDPs. Thus the main goal of this chapter is to answer the research question above by providing and explaining the transformation algorithm from H-MacDecPOMDPs to DecPOMDPs, along with giving the pseudocode for this transformation procedure.

6.1 Brute Force Planning

Solving H-MacDecPOMDPs comes down to finding a top level joint policy π that results in the highest expected value. Equation 6.1 gives a mathematical representation of this.

$$\arg\max_{\pi \in \Pi} \sum_{s \in S} b_0(s) V_t^{\pi}(s, \boldsymbol{\theta}^{start}) \tag{6.1}$$

The top joint policy is a part of the top level joint macro-action (m^{l+1}), i.e. $\pi = \{\times \pi_{m_i^{l+1}}\}_{i \in I}$. Here Π is defined as the set of all joint top level policies. The set Π is finite if the top level policies are deterministic, similarly Π is infinite if the top level policies are non-deterministic. The goal is to find the best top level policy ($\pi \in \Pi$), which maximizes the expected value of the given H-MacDecPOMDP, this is what equation 6.1 expresses.
Furthermore, all non-top level macro-actions are fixed. All possible states must be considered; $\sum_{s \in S}$. For each of these states, the probability of having that state as the initial state must be determined: $b_0(s)$. Finally, this state along with the initial history are put into the value function: $V_t^{\pi}(s, \boldsymbol{\theta}^{start})$.
Recall from section 5.5 that the initial history ($\boldsymbol{\theta}^{start}$) consists of a joint history-stack, which is empty except for the top level (level $l+1$). The top level will always contain the macro-action which contains the top level policy, which was called μ in MacDecPOMDPs. This is the policy that is being optimized. The macro-action on the top level will never terminate, thus the top-level history will always remain unchanged.

6.2 Transforming H-MacDec-POMDPs Into DecPOMDPs

There are useful tools that can be used to solve DecPOMDPs. Thus it would be desirable to transform H-MacDec-POMDPs into Dec-POMDPs, as there are currently no tools or frameworks to solve H-MacDecPOMDPs. This transformation should be done in such a way that the optimal policy that results from planning the resulting Dec-POMDP, should be equivalent to the optimal top level macro policy acquired by solving the H-MacDec-POMDP.
In this section, an overview of the transformation algorithm will be given. This will be followed by a detailed explanation of the Dec-POMDPs that result from applying this transformation.

6.2.1 Transformation I/O

To better understand what this transformation entails, the input and output of the transformation must be made clear. A simple overview of this is found in figure 15.

Figure 15: High-level overview of the transformation procedure.

The input of such a transformation is any H-MacDecPOMDP. The full formal definition for H-MacDec-POMDP can be found in section 5.6.
The output of this transformation is a Dec-POMDP that is functionally equivalent to the given H-MacDec-POMDP. Functionally equivalent in this case means that the same optimal policy is achieved when planning the given problem as both H-MacDec-POMDPs and Dec-POMDPs.
As described in the background section 3.3, Dec-POMDP consist of the following components: $model = \langle I, S, b_0, \{A_i\}_{i \in I}, T, R, \{\Omega_i\}_{i \in I}, O, h \rangle$. Thus these components also need to be present in Dec-POMDPs that were generated from H-MacDecPOMDPs.

6.2.2 Transformation Overview

The transformation algorithm transforms a H-MacDecPOMDP to a DecPOMDP by "grounding" abstraction levels. I define grounding abstraction levels as merging abstraction levels with the ground level. In figure 16, this is shown by the G (Grounding) function, which produces a new grounded level when given the ground level and an abstraction level.
When an abstraction level is grounded, the states set, the states initialization-, transition-, observation- and reward functions (S, b_0, T, O, R) get modified. These modifications allow these functions to use macro-actions, macro-states and macro-histories as of a higher abstraction level as their inputs. An overview of this can be seen in figure 16. In figure 16, the arrow upwards $(\uparrow)$ represents that the set or function has been modified as a result of a grounding, the number $(\uparrow_n)$ represents how often this has occurred.

After j groundings, the newly modified set of macro-states $(S^{\uparrow_j})$ consists of every possible macro-state that can be reached by performing level j macro-actions. Macro-states are explained in section 6.4.1. The new transition functions $(T^{\uparrow_j})$ consist of every transition between macro-states, resulting from executing level j macro-actions. The new ground level observations function $(O^{\uparrow_j})$ contains the probabilities of receiving every joint level j macro-observation given the current macro-action and next macro-state. Finally, the new reward function $(R^{\uparrow_j})$ returns the expected value of every joint level j macro-action and level j macro-state pair. These new grounded Dec-POMDP components are further explained in sections 6.4 to 6.7. Components M^j, Ω^j, I, h do not use the $\uparrow_j$ notation in figure 16, because these components are not modified when grounded.

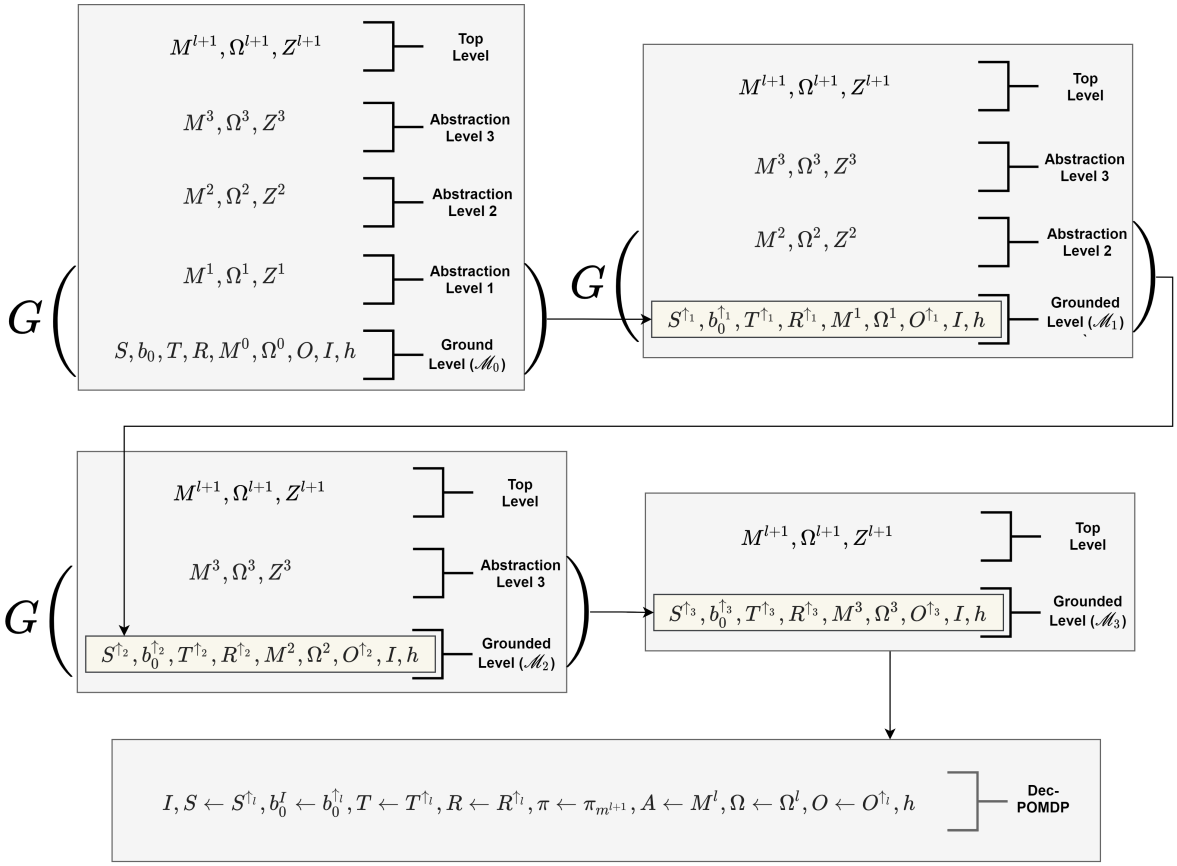


Figure 16: This diagram shows an overview of the H-MacDecPOMDP to Dec-POMDP transformation. In every step, a level of abstraction is grounded.

The transformation happens in a bottom-to-top fashion. First abstraction level one is grounded, then two, all the way to abstraction level l. In figure 16, grounding is represented using a function called G, which encompasses the two levels that are being merged; the abstraction level and the ground level. After each grounding, the resulting model is a completely valid H-MacDecPOMDP, containing the same information as it did before the grounding. The difference is that this information is represented with one abstraction layer fewer. In figure 16, each of the five "blocks" represents one of these H-MacDecPOMDPs.
Each ground(ed) level ($\mathcal{M}_j$) is functionally equivalent to a Dec-POMDP, As these components can form a Dec-POMDP. This is done as follows:

$$S \leftarrow S^{\uparrow_j}, b_0 \leftarrow b_0^{\uparrow_j}, T \leftarrow T^{\uparrow_j}, R \leftarrow R^{\uparrow_j}, A \leftarrow M^{l\uparrow_j}, \Omega \leftarrow \Omega^{l\uparrow_j}, O \leftarrow O^{\uparrow_j}$$

The more often groundings have taken place, the more of the H-MacDecPOMDP abstraction levels will be incorporated into the grounded level. Eventually, after grounding all abstraction levels, only the ground level and top level will be left. When this is the case, the original H-MacDecPOMDP becomes a Dec-POMDP.

6.3 Transformation Algorithm

The transformation algorithm goes over each abstraction level and grounds it. To ground an abstraction level, five steps must be completed. An overview of these steps are shown in figure 17. a more detailed explanation of these steps will be given in subsections 6.4 to 6.7. The $\times l$ in figure 17 specifies that the arrow under it is followed l times, once for each abstraction level.

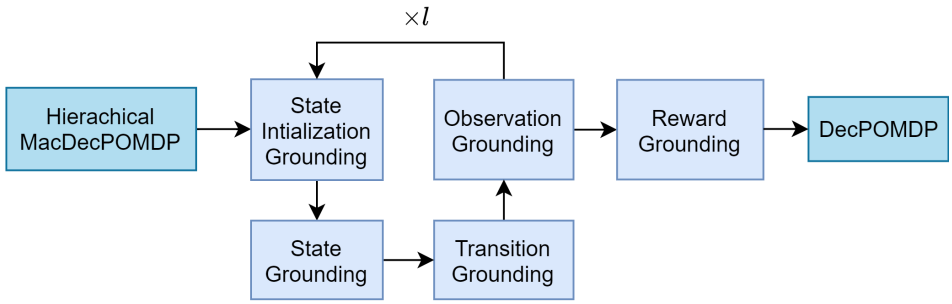


Figure 17: This diagram shows an overview of the steps in the H-MacDecPOMDP to DecPOMDP transformation.

6.3.1 Transform Function

The pseudocode below executes the grounding procedures for the State Initialization function, States set, Transition Function and Observation function l times, once for each abstraction level. Then the reward grounding function is executed once. Finally the components that result from these grounding procedures are returned. These resulting components form a Dec-POMDP as described in section 6.2.2.

function TRANSFORM$(I, S, b_0, T, R, M, \Omega, \{Z_i^j\}_{i\in I}^{j\in\{1,..,l\}}, O, h, l)$
  $b_0^{\uparrow_0} \leftarrow b_0, S^{\uparrow_0} \leftarrow S, T^{\uparrow_0} \leftarrow T, O^{\uparrow_0} \leftarrow O, R^{\uparrow_0} \leftarrow R$
  while $j \in \{1, ..., l\}$ **do**
    $b_0^{\uparrow_j} \leftarrow StateInitializationGrounding(b_0^{\uparrow_{j-1}}, S^{\uparrow_{j-1}})$
    $S^{\uparrow_j} \leftarrow StateGrounding(I, S^{\uparrow_{j-1}}, O^{\uparrow_{j-1}}, b_0^{\uparrow_j}, T^{\uparrow_{j-1}}, M^{j-1,j}, \Omega^{j-1}, h)$
    $T^{\uparrow_j} \leftarrow TransitionGrounding(M^{j-1,j}, \Omega^{j-1}, S^{\uparrow_{j-1,j}}, O^{\uparrow_{j-1}}, T^{\uparrow_{j-1}})$
    $O^{\uparrow_j} \leftarrow ObservationGrounding(I, S^{\uparrow_{j-1,j}}, M^j, \Omega^{j-1,j}, Z^j, O^{\uparrow_{j-1}})$
  end while
  $R^{\uparrow_l} \leftarrow RewardGrounding(I, S^{\uparrow_l}, M, R^{\uparrow_0})$
  return $\langle I, S^{\uparrow_l}, b_0^{\uparrow_l}, T^{\uparrow_l}, R^{\uparrow_l}, M^l, \Omega^l, O^{\uparrow_l}, h\rangle$
end function

6.4 State Grounding

This section describes what grounded states are and why they are needed. Then the algorithm for grounding states is given and explained. These states are referred to as *macro-states* in the remainder of this thesis.

6.4.1 Grounded States Definition

In the given H-MacDec-POMDP, the reward function, the ground level observation probability function and the state transition function do not accept macro-actions and macro-observations as inputs. Instead, they use ground level actions and ground level observations.
Thus in order to define a state transition and reward function that does allow for macro-actions to be used instead of ground actions, more information needs to be given to these functions. Namely, information that can be used to determine what the next ground-actions will be. In Dec-POMDPs, there are no other parameters that can be used in these functions to convey this ground-action information. T and R only take an action and a state, while the observation function only takes an observation and a state. Since the actions and observations are being replaced with macro-actions and macro-observations, the only solution here is to embed the necessary additional information into the state. The ground-level actions that are taken are determined by the policies of macro-actions. Policies use histories to decide what actions should be taken. Thus states should contain histories. This results in the following new set of states:

$$\mathbb{S}^1 = \{(s, \theta^0) | s \in S, \theta^0 \in H^0\}$$

These new states ($s^1 \in \mathbb{S}^1$) could then be used in functions that use macro-actions or macro-observations of abstraction level 1. This description is similar to the NOMDP [41] formulation of a DecPOMDP, however this description is generalized to hierarchical settings.
Generalizing over the amount of abstracted levels (j), these abstracted states are defined as follows:

$$s^j = (s^{j-1}, \theta^{j-1}) \in \mathbb{S}^j$$

These sets of states ($\mathbb{S}^j$) can grow to be very large, section 8.1 elaborates on a possible solution.
The notion of abstracting away levels will be further explained in subsection 6.2.2. Such a grounded state s^j, will be referred to as a *level j macro-state*. So for example, if 4 levels are abstracted away, the level 4 macro-state would look like so:

$$s^4 = ((((s, \theta^0), \theta^1), \theta^2), \theta^3) \in \mathbb{S}^4$$

Macro-states are defined recursively because this eases the process of constructing macro-states with more grounded levels.
The $\mathbb{S}^j$ sets contain **all** level j macro-states, even those that are not possible. For example, it is impossible for the level 2 macro-history (θ^2) of a macro-state to contain more macro-actions and macro-observations than its level 1 macro history. The reasoning for this is the same as in section 5.4. However, such unreachable states are still present in $\mathbb{S}^j$ sets. Such a set ($\mathbb{S}^j$) is referred to as a *total level j macro-state set*.
The definition of $\mathbb{S}^j$ sets is necessary in order to define $S^{\uparrow_j}$; the *grounded level j macro-state* set. The grounded level j macro-state set ($S^{\uparrow_j}$), is a subset of the total level j macro-state set ($\mathbb{S}^j$).

$$S^{\uparrow_j} \subset \mathbb{S}^j$$

However, each element of $S^{\uparrow_j}$ must be *reachable*, this is because non-reachable macro-states see no use in H-MacDecPOMDPs. A macro-state s^j is considered reachable if there is a joint history of level j macro-actions and a history of level j macro-states that has a greater than 0 chance of reaching s^j. An example of this is shown in figure 18. The history of level j macro-actions thus does not rely on any top level policy.

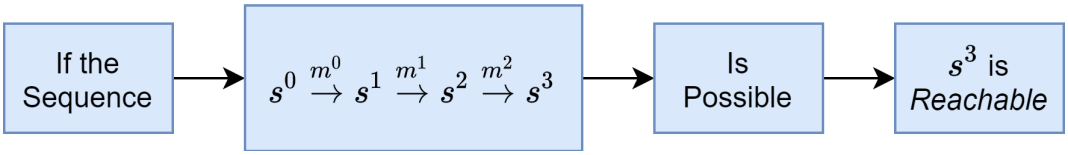


Figure 18: This diagram shows an example of a reachable state consisting of three time-steps.

These two histories are defined as follows:

- $\bar{\mathbb{S}}^j$ represents the set of all histories of macro-states in $\mathbb{S}^j$. This history has a state for each time-step.
- $\bar{M}^j$ represents the set of all joint histories of level j macro-actions (M^j). This history has a level j joint macro-action for each time-step.

A macro-state s^j is considered reachable if it can be reached by performing a series of joint level j macro-actions ($\bar{m}^j$).
Termination and initiation conditions of the level j macro-actions are not considered here, the reasoning for this is explained in section 6.8. The level j grounded transition function ($T^{\uparrow_j}$) does however implicitly deal with the termination and initiation conditions for levels under j, this will become apparent in sections 6.4 and 6.5.
Thus, equation 6.2 shows the formal definition for the set of level j grounded states ($S^{\uparrow_j}$) (i.e. all reachable level j macro-states).
The probability of reaching a particular level j macro-state (s^j) is calculated by multiplying the probabilities of transitioning to all level j macro-states ($\bar{s}^{j,t}$) that are reached along the way to the desired level j macro-state (s^j). The probability of reaching all level j macro-states ($s^{j,t}$) in the given macro-state history ($\bar{s}^j$) is calculated by first determining the probability of starting at the first macro-state in the history ($b_0^{\uparrow_j}(s^{j,0})$) and multiplying this by the probabilities of transitioning to each subsequent level j macro-state ($T^{\uparrow_j}(\bar{s}^{j,t}|\bar{m}^{j,t-1}, \bar{s}^{j,t-1})$). If the resulting probability is greater than 0, the level j macro-state (s^j) in question is deemed reachable.

$$S^{\uparrow_j} = \left\{ s^j \in \mathbb{S}^{\uparrow_j} \middle| \exists_{\bar{m}^j \in \bar{M}^j} \exists_{\bar{s}^j \in \bar{\mathbb{S}}^{\uparrow_j} | \bar{s}^{j,|s^j:\theta^0|-1} = s^j} b_0^{\uparrow_j}(\bar{s}^{j,0}) \prod_{t=1}^{|s^j:\theta^0|-1} T^{\uparrow_j}(\bar{s}^{j,t}|\bar{m}^{j,t-1}, \bar{s}^{j,t-1}) > 0 \right\} \quad (6.2)$$

6.4.2 Notation Comments

The second superscript of a macro-state history ($\bar{m}^j$) or joint macro-action history ($\bar{s}^j$) is used to point to a particular macro-state or joint macro-action of that time-step (t) (e.g. $\bar{s}^{j,t}$ or $\bar{m}^{j,t}$).
The $|s^j : \theta^0|$ notation, returns the amount of level 0 actions that were taken on the ground level history (θ^0). The amount of actions that are taken is equal for all agents in the ground level, thus the agent subscript (i in θ_i^0) can be omitted here. This notation can be generalized for the amount of level k ($0 < k < j$) (macro-)actions of any agent i for any level j macro-state, this is done in equation 6.3.

$$|s^j : \theta_i^k| \tag{6.3}$$

$\exists_{\bar{s}^j \in \bar{\mathbb{S}}^{\uparrow j} \,|\, \bar{s}^{j,|s^j:\theta^0|-1} = s^j}$ in equation 6.2 states that there must exist a history of level j macro-states ($\bar{s}^j$), such that the last macro-state of this history ($\bar{s}^{j,|s^j:\theta^0|-1}$) equals the level j macro-state that is being examined (s^j). This must be true because s^j is the macro-state that must be reachable.

6.4.3 State Initialization Grounding

The State Initialization Function is grounded by creating new initial states from old initial states, and setting the probability of receiving the new initial state equal to the probability of receiving the old initial state. This new initial (grounded) state is equal to the initial state of the level under it, packaged with an empty history, such that the $s^j = (s^{j-1}, \theta^{j-1})$ format is achieved. Note that an initial state refers to a state where 0 steps have been taken. The pseudocode for this procedure is given below.

function StateInitializationGrounding($b_0^{\uparrow j-1}, S^{\uparrow j-1}$)
  $b_0^{\uparrow j} \leftarrow \emptyset$
  for all $s^{j-1} \in S^{\uparrow j-1}$ s.t. $|s^{j-1} : \theta^0| = 0$ **do**
    $s^j \leftarrow (s^{j-1}, \emptyset)$
    $b_0^{\uparrow j}(s^j) \leftarrow b_0^{\uparrow j-1}(s^{j-1})$ ▷ Which equals $b_0(s^0)$
  end for
  return $b_0^{\uparrow j}$
end function

6.4.4 State Grounding Algorithm

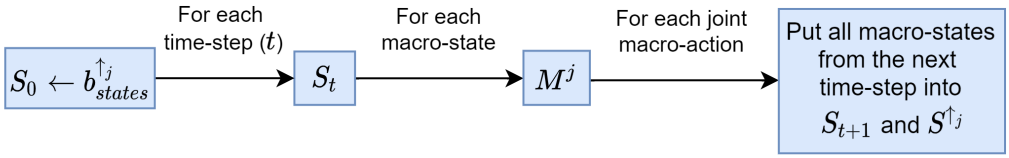


Figure 19: This diagram shows an overview of the state grounding step in the H-MacDecPOMDP transformation. If a time-step has been exhausted, S_{t+1} becomes S_t because t increments.

During the state grounding procedure, every reachable state is constructed. This is done by considering every known macro-state (s_t^j), and determining what other macro-states (s_{t+1}^j) can be reached from each considered macro-state (s_t^j). Thus if $S_t^{\uparrow j}$ refers to all level j macro-states where the ground level history length equals t, the goal is to find $S^{\uparrow j} \leftarrow \cup_{t=0}^{h-1} S_t^{\uparrow j}$.

First every initial macro-state gathered from the state initiation function is considered ($\{s \in S^{\uparrow_j} | b_0^{\uparrow_j}(s) > 0\}$), then combined with every possible macro-action (m^j) that can be taken (without any regard for a top level policy), every possible resulting macro-state (s^j_{t+1}) that can be reached is constructed. Every newly constructed macro-state is then stored after which the process starts again but with every newly constructed macro-state as the starting points, this process will thus iteratively construct even more new macro-states. This is repeated until horizon h is reached. Figure 19 shows an overview of this process.

The *NextStepStates* procedure is used to determine every macro-state that can be reached from another macro-state when given a macro-action (blue square 4 in figure 19). To understand how the NextStepStates procedure does this, recall that a macro-state s^j consists of a lower level macro-state and a lower level macro-history ($s^j = (s^{j-1}, \theta^{j-1})$). Thus every possible ($s^j = (s^{j-1}, \theta^{j-1})$) pair must be found. To do this, the next histories will be constructed ($\theta_i^{j-1\prime} = (\theta_i^{j-1}, m_i^{j-1}, z_i^{j-1\prime})$). This is done by first considering every $j-1$ level macro-action (m^{j-1}) that has a greater than 0 probability to be taken. This level $j-1$ macro-action (m^{j-1}) is then used to find every level $j-1$ macro-state ($s^{j-1\prime}$) with a transition probability that is greater than 0. Then every possible level $j-1$ macro-observation that has a greater than 0 probability of occurring is considered. Then these new level $j-1$ macro-states ($s^{j-1\prime}$) are combined with their respective level $j-1$ joint macro-actions (m^{j-1}) and macro-observations ($z^{j-1\prime}$) histories ($\theta^{j-1\prime}$) to return as new macro-states ($s^{j\prime} = (s^{j-1\prime}, \theta^{j-1\prime} = (\theta^{j-1}, m^{j-1}, z^{j-1\prime}))$).

The pseudocode for these procedures is given below:

function StateGrounding($I, S^{\uparrow_{j-1}}, O^{\uparrow_{j-1}}, b_0^{\uparrow_j}, T^{\uparrow_{j-1}}, M^{j-1,j}, \Omega^{j-1}, h$)
  $S_0 \leftarrow \{s \in S^{\uparrow_j} | b_0^{\uparrow_j}(s) > 0\}$ ▷ S_0 is set equal to all possible initial states.
  $S^{\uparrow_j} \leftarrow S_0$
  for all $t \in \{0, ..., h-1\}$ **do**
    $S_{t+1} \leftarrow \emptyset$
    for all $s^j \in S_t$ **do**
      for all $m^j \in M^j$ **do**
        $S_{t+1} \leftarrow S_{t+1} \cup nextStepStates(s^j, m^j, M^{j-1}, \Omega^{j-1}, S^{\uparrow_{j-1}}, T^{\uparrow_{j-1}}, O^{\uparrow_{j-1}})$
      end for
    end for
    $S^{\uparrow_j} \leftarrow S^{\uparrow_j} \cup S_{t+1}$
  end for
  return $S^{\uparrow_j}$
end function

function nextStepStates($s^j, m^j, M^{j-1}, \Omega^{j-1}, S^{\uparrow_{j-1}}, T^{\uparrow_{j-1}}, O^{\uparrow_{j-1}}$)
  $S_{new} \leftarrow \emptyset$
  for all $m^{j-1} \in M^{j-1}$ s.t. $\prod_{i \in I} Pr(m_i^{j-1} | \theta_i^{j-1}, m_i^j) > 0$ **do** ▷ θ_i^{j-1} is taken from s^j
    for all $s^{j-1\prime} \in S$ s.t. $T^{\uparrow_{j-1}}(s^{j-1\prime} | m^{j-1}, s^{j-1}) > 0$ **do**
      for all $z^{j-1\prime} \in \overset{*}{\Omega}^{j-1}$ s.t. $O^{\uparrow_{j-1}}(z^{j-1\prime} | m^{j-1}, s^{j-1\prime}) > 0$ **do**
        $\theta^{j-1\prime} \leftarrow appendHistory(\theta^{j-1}, m^{j-1}, z^{j-1\prime})$
        $s^{j\prime} \leftarrow (s^{j-1\prime}, \theta^{j-1\prime})$
        $S_{new} \leftarrow S_{new} \cup \{s^{j\prime}\}$
      end for
    end for
  end for
  return S_{new}
end function

The $Pr(m_i^{j-1} | \theta_i^{j-1}, m_i^j)$ calculation is the same as in equation 5.11 of section 5.8.
The *appendHistory* given below works in the exact same way as in equation 5.16 of section 5.12.

function appendHistory($\theta^{j-1}, m^{j-1}, z^{j-1\prime}$)
  $\theta^{j-1\prime} \leftarrow \theta^{j-1}$
  for all $i \in I$ **do**
    if $\theta_i^{j-1} = (..., null)$ **then** ▷ The last level $j-1$ action had persisted.
      Replace *null* in $\theta_i^{j-1\prime}$ with $z_i^{j-1\prime}$
    else ▷ The last level $j-1$ action had terminated.
      $\theta_i^{j-1\prime} \leftarrow (\theta_i^{j-1}, m_i^{j-1}, z_i^{j-1\prime})$
    end if
  end for
  return $\theta^{j-1\prime}$
end function

6.5 Transition Grounding

The new transition probabilities for the transition function of these newly generated Dec-POMDPs now also incorporate the probability of reaching a new history $\theta^{j-1\prime}$. This is because the probability of appending history θ^{j-1} to form $\theta^{j-1\prime}$ is now part of state transitions as these histories are inside macro-states;

$$T^{\uparrow_j}(s^{j\prime}|s^j, m^j) = T^{\uparrow_j}((s^{j-1\prime}, \theta^{j-1\prime})|(s^{j-1}, \theta^{j-1}), m^j)$$

Thus the probability of transitioning from s^j to $s^{j\prime}$ is equal to the transition probability of $s^{j-1} \to s^{j-1\prime}$ multiplied with the probability of reaching history $\theta^{j-1\prime}$ from θ^{j-1}.
In the *transitionGrounding* function, the states s^j in $S^{\uparrow_j}$ are considered as starting points in the transition. However, the states in the last time-step ($t = h - 1$) must be excluded, as they cannot be the starting point of a transition as there are no macro-states in $S^{\uparrow_j}$ that they can transition to.
Thus we can see that the transitionGrounding procedure goes through each macro-state, each macro-action and each possible state that can be transitioned to using the previously discussed nextStepStates procedure. Afterwards the probability of making this transition is calculated and assigned to the relevant transition entry of the new grounded transition function.

function TRANSITIONGROUNDING($M^{j-1,j}, \Omega^{j-1}, S^{\uparrow_{j-1,j}}, O^{\uparrow_{j-1}}, T^{\uparrow_{j-1}}$)
  for all $s^j \in S^{\uparrow_j}$ s.t. $|s^j : \theta^0| < h - 1$ **do**
    for all $m^j \in M^j$ **do**
      for all $s^{j\prime} \in nextStepStates(s^j, m^j, M^{j-1}, \Omega^{j-1}, S^{\uparrow_{j-1}}, T^{\uparrow_{j-1}})$ **do**
        $T^{\uparrow_j}(s^{j\prime}|s^j, m^j) \leftarrow calculateTransition(s^j, m^j, s^{j\prime}, O^{\uparrow_{j-1}}, T^{\uparrow_{j-1}})$
      end for
    end for
  end for
  return $T^{\uparrow_j}$
end function

The *calculateTransition* function calculates the probability of transitioning from macro-state s^j to $s^{j\prime}$, this is shown in equation 6.4. The probability of appending the history within the given macro-states (equation 6.5) is calculated by first calculating the probability of adding the relevant $j-1$ joint macro-action to this history, and multiplying this by the probability of retrieving the relevant level $j-1$ macro-observation.

$$T^{\uparrow_j}((s^{j-1\prime}, \theta^{j-1\prime})|(s^{j-1}, \theta^{j-1}), m^j) = T^{\uparrow_{j-1}}(s^{j-1\prime}|s^{j-1}, m^{j-1}) Pr(\theta^{j-1\prime}|\theta^{j-1}, m^j, s^{j-1\prime}) \tag{6.4}$$

$$Pr(\theta^{j-1\prime}|\theta^{j-1}, m^j, s^{j-1\prime}) = \left[\prod_{i \in I} Pr(m_i^{j-1}|\theta_i^{j-1}, m_i^j)\right] O^{\uparrow_{j-1}}(z^{j-1}|m^{j-1}, s^{j-1\prime}) \tag{6.5}$$

The pseudocode for the transition probability is given below;

function CALCULATETRANSITION($s^j, m^j, s^{j\prime}, O^{\uparrow_{j-1}}, T^{\uparrow_{j-1}}$)
  $P_{history} \leftarrow \left[\prod_{i \in I} Pr(m_i^{j-1}|\theta_i^{j-1}, m_i^j)\right] O^{\uparrow_{j-1}}(z^{j-1\prime}|m^{j-1}, s^{j-1\prime})$ ▷ $Pr(\theta^{j-1\prime}|\theta^{j-1}, m^j, s^{j-1\prime})$
  return $T^{\uparrow_{j-1}}(s^{j-1\prime}|s^{j-1}, m^{j-1}) \cdot P_{history}$
end function

6.6 Observation Grounding

The grounded observation function determines the probability of receiving macro-observation z using histories under it. It is possible for z to be a *null*-observation, as the probability of termination is also accounted for in the probability calculation of z. This means that unlike in the normal O function, $O^{\uparrow_j}$ where $j \geq 1$, *null*-observations can have a probability that is greater than 0.
The Observation Grounding procedure works by going over every possible (Joint macro-action, state, valid joint macro-observation) tuple and calculating the probability of receiving that macro-observation. An overview of this can be found in figure 20.

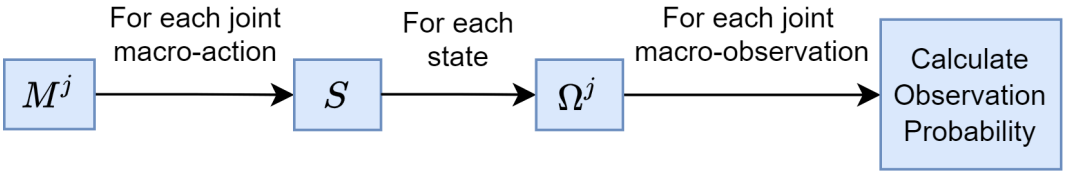


Figure 20: This diagram shows an overview of the Observation Grounding function.

The observation probability calculation works by multiplying the macro-observation probability for each agent, this is shown in equation 6.6. The $Pr(z_i^{j\prime}|\theta_i^{j-1\prime}, m_i^j)$ here is the same as equation 5.14 in section 5.9.

$$O^{\uparrow_j}(z^{j\prime}|m^j, s^{j\prime}) = \prod_{i \in I} Pr(z_i^{j\prime}|\theta_i^{j-1\prime}, m_i^j) \tag{6.6}$$

The pseudocode for the Observation Grounding phase is found here:

function ObservationGrounding($I, S^{\uparrow_{j-1,j}}, M^j, \Omega^{j-1,j}, Z^j, O^{\uparrow_{j-1}}$)
  $O^{\uparrow_j} \leftarrow \emptyset$
  for all $m^j \in M^j$ **do**
    for all $s^{j\prime} \in S^{\uparrow_j}$ **do** ▷ $s^{j\prime} = (s^{j-1\prime}, \theta^{j-1\prime})$
      for all $z^{j\prime} \in \overset{*}{\Omega}^j$ **do**
        $O^{\uparrow_j}(z^{j\prime}|m^j, s^{j\prime}) \leftarrow \prod_{i \in I} Pr(z_i^{j\prime}|\theta_i^{j-1\prime}, m_i^j)$
      end for
    end for
  end for
  return $O^{\uparrow_j}$
end function

6.7 Reward Grounding

The grounded reward function returns the expected value of the ground-level rewards achieved when in a level l macro-state and when taking a level l macro-action. This is done by multiplying the probabilities of taking each possible joint action-stack $\boldsymbol{m}$ constrained by taking macro-action m^j, with the reward achieved when taking joint ground level action $\boldsymbol{a}$ in $\boldsymbol{m}$.
The Reward Grounding phase works by going through every l level macro-state and every level l joint macro-action and calculating its expected reward. This is also shown in figure 21. This is sufficient because $R^{\uparrow j}$ still returns the expected reward for a single time-step, and time-steps are never skipped, even if no macro-actions terminate.

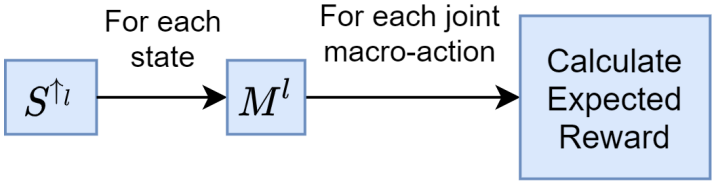


Figure 21: This diagram shows an overview of the Reward Grounding function.

The expected reward calculation works by summing the probability of performing every single valid joint (macro-)action stack and multiplying each of these probabilities by its respective reward. Do note that these joint action stacks excludes layer l, because the probability of it being chosen is 1 when it is given as a parameter to the grounded $R^{\uparrow l}$ function.
The calculation of the probability of taking a joint macro-action stack is the same as in section 5.8 $\left(\sum_{\boldsymbol{m}\in M} Pr(\boldsymbol{m}|\boldsymbol{\theta}) \text{ and } Pr(\boldsymbol{m}|\boldsymbol{\theta}) = \prod_{i\in I}\prod_{j\in L} Pr(m_i^j|\theta_i^j, m_i^{j+1})\right)$, except that the abstraction level l is omitted.
To calculate the reward of a joint-action stack, its ground level joint action along with the ground level state is put into the old reward function. The equation for the calculation of the expected reward of a level l macro-state and level l joint macro-action stack is given in equation 6.7. U stands for every *underground* level, i.e. L ($L = \{0, ..., l\}$) but excluding l ($U = L \setminus \{l\} = \{0, ..., l-1\}$).

$$R^{\uparrow l}(s^l, m^l) = \sum_{m^U \in M^U} \prod_{i\in I}\prod_{j\in U} Pr(m_i^j|\theta_i^j, m_i^{j+1}) R^{\uparrow 0}(s^0, \boldsymbol{a}) \tag{6.7}$$

The pseudocode for the Reward Grounding procedure is found below;

function RewardGrounding($I, S^{\uparrow l}, M, R^{\uparrow 0}$)
  $R^{\uparrow l} \leftarrow \emptyset$
  $U \leftarrow \{0, ..., l-1\}$
  for all $s^l \in S^{\uparrow l}$ **do**
    for all $m^l \in M^l$ **do**
      $R^{\uparrow l}(s^l, m^l) \leftarrow \sum_{m^U \in M^U} \prod_{i\in I}\prod_{j\in U} Pr(m_i^j|\theta_i^j, m_i^{j+1}) R^{\uparrow 0}(s^0, \boldsymbol{a})$
    end for
  end for
  return $R^{\uparrow l}$
end function

6.8 Initialization and Termination Heuristics

DecPOMDPs do not have any built in mechanisms to enforce termination and initiation constraints. Thus, these constraints will be enforced by using heuristics. In this section, other alternative ways of enforcing these constraints are explained, along with the reasoning behind not choosing these methods. Recall that when a macro-action does not terminate, the same macro-action must be executed for another time-step. Additionally, if the previous macro-action did terminate, a macro-action can only be executed if the current history is part of this macro-action's initiation set. These two constraints are inherent to macro-actions and are called the Termination and Initiation Constraints respectively.
In underground ($< l$) abstraction levels, these constraints are upheld by the transition function. The transition function assigns a probability of zero to any macro-state where its histories do not satisfy these constraints. Empty histories are also histories which can be present in initiation sets, so this also works for initial macro-states. Recall that any macro-state of level j (s^j) consists a tuple of another level $j-1$ macro-state and history. When calculating the transition from $s^j = (s^{j-1}, \theta^{j-1})$ to $s^{j\prime} = (s^{j-1\prime}, \theta^{j-1\prime})$, the probability of appending a level $j-1$ macro-action is calculated. This is done using equation 5.11 from section 5.8 $\left(Pr(m_i^{j-1}|\theta_i^{j-1}, m_i^j)\right)$. Equation 5.11 checks for the initiation and termination constraint, which by induction results in all underground initiation and termination constraint to be enforced. However, the resulting level l grounded transition function $\left(T(s^{l\prime}|s^l, m_i^l)\right)$ does not have $\left(Pr(m_i^l|\theta_i^l, m_i^{l+1})\right)$ calculations embedded in them, leading to the termination and initiation constraint of level l macro-actions to not be automatically enforced in DecPOMDPs transformed from H-MacDecPOMDPs. H-MacDecPOMDPs however do enforce these constraints in $Pr(m_i^j|\theta_i^j, m_i^{j+1})$ (equation 5.11), see section 5.8.
The level l macro-states in the level l grounded transition function only have access to the histories *under* l, thus the transition function simply does not have the means necessary (θ_i^l) to check whether the initiation constraint holds for level l. The same goes for the termination constraint, the level l macro-state is also not supplied with the level l macro-action that persisted in the previous time-step. In H-MacDecPOMDPs, the macro-action probability function ($Pr(m_i^j|\theta_i^j, m_i^{j+1})$) checks whether these constrains are satisfied. Dec-POMDPs however do not have this notion of Termination and Initiation constraints and thus do not do such checks. One way to solve this is to embed the level l action-observation history into the level l macro-states, and make entries that do not satisfy the Termination and Initiation constraints transition to an *error state* with probability 1. Reaching such an error state would then lead to an enormous penalty dealt by the reward function.
However, such a solution is incredibly unwieldy, as the amount of macro-states would in the worst case be multiplied by the amount of possible level l histories. As section 7.2 will show, the amount of histories increases exponentially with the square of the horizon. Each possible joint history would be considered many different times for each state, this macro-state complication would thus be much more expensive than solving the resulting DecPOMDP. Thus, this thesis chooses not to further complicate macro-states. Any efficient planner for these DecPOMDPs would have to implement the heuristics that are shown below, thus it is much more efficient and painless to assume that these few simple heuristics are in place when planning policies. These heuristics are the following:

1. Termination Heuristic: If the previous level l grounded observation equals *null*, always choose the previous level l grounded action.

2. Initiation Heuristic: If the previous level l grounded observation does not equal *null*, the level l history must be present in the initiation set of the newly chosen level l grounded action.

7 H-MacDecPOMDP Example and Analysis

This chapter is about analyzing H-MacDecPOMDPs and providing an example of an H-MacDecPOMDP. An explanation of the effectiveness of the H-MacDecPOMDP approach, along with its transformation to Dec-POMDPs will be provided, which answers research question 3;

What are the drawbacks and benefits of H-MacDecPOMDPs as opposed to (Mac-)Dec-POMDPs?

Thus the goal of this chapter is to answer this research question. This is done by first defining an example problem (the H-MacDecTiger problem), to be analyzed. Then an approach of determining the policy search space is provided. Afterwards the policy search space of the example problem is analyzed. Afterwards, other benefits and drawbacks of H-MacDecPOMDPs and their transformation to Dec-POMDPs are described. Afterwards, an experiment using the H-MacDecTiger example problem is given to further validate the correctness of the H-MacDecPOMDP model.

7.1 Hierarchical MacDecTiger Problem

In this section, a variant of the DecTiger problem is defined. This problem is defined as a H-MacDecPOMDP, and will be further referred to as the *H-MacDecTiger Problem*. In this problem, the two agents still have the same top level actions as the original Dec-Tiger problem. However, the agents also have a location and orientation and ground-level actions that allow agents to move and rotate. Orientation and location are of course ripe for abstraction, as these two properties of the agent are not relevant when making the high level decision of whether to open a door or listen for the tiger. The H-MacDecTiger problem will be formally defined in this section.

7.1.1 Summary

The H-MacDecTiger problem is a H-MacDecPOMDP based on the Dec-Tiger problem. The DecTiger problem has no levels of abstraction, while the H-MacDecTiger problem has two. The second level of abstraction can be seen as the original Dec-Tiger problem, where the actions and observations are limited to listening and opening doors. The level of abstraction under that (level 1) abstracts away the locations of the agents. A second abstraction level macro-action such as "Open the left door" will have a policy of first abstraction level macro-actions which move the agent to the correct location. The first abstraction level macro-actions consist of moving left, right or interacting with the environment. Reasons to use multiple abstraction levels can be found in section 7.3.
In order to take a step to the left or right, the agent must rotate to the correct orientation and take a step forward. Rotate and Go are the two ground level actions. The rotate action will rotate the agent. The Go action will move the player forward or interact with the environment if it is facing a contraption it can use. Such a contraption can be lever to open a door, or a headphone to listen for the tiger, which will be further explained in section 7.1.3. Thus the second abstraction level macro-observations are the tiger location sounds. The first abstraction level macro-observations consists of both the tiger sounds, but also the location of the agent. The ground level observations consists of the tiger sounds and the orientation of the agent. Thus, in order to construct a first level macro-observation about the location, the ground level history of orientation observations, rotate actions and go actions must be examined to determine the current location macro-observations, this is done by the macro-observation probability function of the first abstraction level (Z^1). The observations that relay the location of the tiger can be trivially passed upwards when the agents open doors or listen.

7.1.2 States

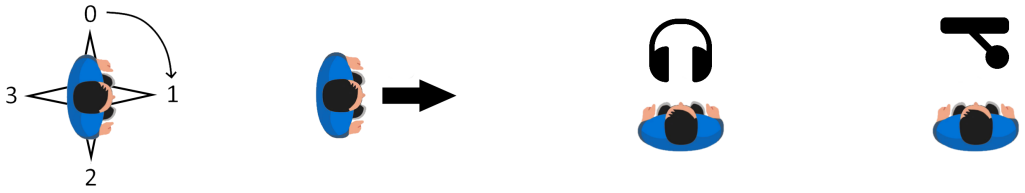
This section describes the states of the H-MacDecTiger Problem. The states must be able to hold the information of behind which door the tiger is and what the location and orientation of both agent are. The amount of states in the set of states (S) equals 800 instead of 2. This is the case because there are 2 locations of the tiger (loc_T), 5 possible locations for each of the 2 agents, and 4 orientations for each of the 2 agents. This leads to $2 \cdot 5^2 \cdot 4^2 = 800$. The positions of the agents (loc_1 and loc_2) go from 1 to 5 (as can be seen in figure 22). The directions go from 0 to 3. The location of the tiger is either L (left) or R (right). The mathematical representation of the states can be found in equation 7.1.

$$\langle loc_T, loc_1, loc_2, dir_1, dir_2 \rangle = \langle \{L, R\}, \{1, ..., 5\}, \{1, ..., 5\}, \{0, ..., 3\}, \{0, ..., 3\} \rangle \tag{7.1}$$

Figure 22 shows the graphical representation of the two initial states. State $\langle L, 2, 4, 0, 0 \rangle$ of figure 22a has a 50% chance of being the initial state ($b_0(\langle L, 2, 4, 0, 0 \rangle) = 0.5$). State $\langle R, 2, 4, 0, 0 \rangle$ of figure 22b has a 50% chance of being the initial state ($b_0(\langle R, 2, 4, 0, 0 \rangle) = 0.5$). Thus there is one initial state per possible tiger location.

(a) $\langle L, 2, 4, 0, 0 \rangle$ (b) $\langle R, 2, 4, 0, 0 \rangle$

Figure 22: A graphical representation of the two initial states in the H-MacDecTiger problem. The blue person represents agent 1, and the orange person represents agent 2.



(a) a_{ro} (b) a_f (c) a_{li} (d) a_{pull}

Figure 23: A graphical representation of the four actions in the H-MacDecTiger problem.

7.1.3 Ground Level Actions

The set of actions for both agents; rotate, forward, listen or pull ($M_i^0 = \{a_{ro}, a_f, a_{li}, a_{pull}\}$). These actions are shown in figure 23. The rotate action (a_{ro}), will rotate the agent 90° in the clockwise direction. The forward action (a_f) will move the agent a step forward into the direction it is facing. The listen action (a_{li}) will make the agent use the headphone it is facing to listen for the tiger. The pull action (a_{pull}) will make the agent pull a lever it is facing to open a door. Both agents need to either use listen using headphones or pull a lever **at the same time** for it to have any effect.

7.1.4 Transition Function

This section describes the transition function of the H-MacDecTiger problem. The rotate action a_{ro} rotates the agent clockwise. This is done by incrementing the *direction* parameter of the agent. If the *direction* parameter goes over 3, it should go back to 0. Mathematically this is represented like so: $direction + 1 \mod 4$. Equation 7.2 shows an example where both agents rotate from east to south on locations 2 and 4 respectively, with the tiger being on the left.

$$T\left(\langle L, 2, 4, 2, 2\rangle \middle| \langle L, 2, 4, 1, 1\rangle, \langle a_{ro}, a_{ro}\rangle\right) = 1 \tag{7.2}$$

If an agent is facing east or west and performs the forward action (a_f), the agent's location moves one position east or west respectively. However, left of location 1 and right of location 5, there are walls. These walls make it impossible to move further than 1 or 5 westwards or eastwards respectively. Mathematically this is written as $\max(location - 1, 1)$ and $\min(location + 1, 5)$ respectively. An example is shown in equation 7.3, where both agents go to the west from location 1 and 5 with the tiger on the left.

$$T\left(\langle L, 1, 4, 3, 3\rangle \middle| \langle L, 1, 5, 3, 3\rangle, \langle a_f, a_f\rangle\right) = 1 \tag{7.3}$$

Moving to the north or south is impossible, thus performing such actions does not change the location parameter. This is mathematically represented in equation 7.4 where agent one tries to move to the north, and agent 2 tries to move south.

$$T\left(\langle L, 2, 4, 0, 2\rangle \middle| \langle L, 2, 4, 0, 2\rangle, \langle a_f, a_f\rangle\right) = 1 \tag{7.4}$$

Finally there are the actions that involve listening or opening doors. These are a_{li} for listening and a_{pull} for pulling a lever to open a door. The a_{li} action will only work if the agent is directly facing headphones it can use. An agent is directly facing headphones when it is facing north in location 3. Similarly, performing the a_{pull} action is only effective if the agent is facing a lever it can pull. These levers can be found in locations 1 and 5 when facing north.
Both agents need to either use headphones to listen or pull a lever to open a door at the same time for it to have any effect. If for example agent 1 pulls the lever in location 1 and agent 2 is rotating, pulling a lever will do nothing. This is expressed in equation 7.5

$$T\left(\langle L, 1, 4, 0, 2\rangle \middle| \langle L, 1, 4, 0, 3\rangle, \langle a_{pull}, a_{ro}\rangle\right) = 1 \tag{7.5}$$

If both agents at the same time face north and listen at location 3 or pull while at location 1 or 5, the environment will reset. Resetting the environment entails putting the state back into one of its two initial positions. An examples is given in equation 7.6, in this example the tiger is left and agent 1 pulls the left lever while agent 2 listens, a hashtag (#) is used to denote that the transition probability here does not depend on the tiger's location.

$$T\left(\langle \#, 2, 4, 0, 0\rangle \middle| \langle \#, 1, 3, 0, 0\rangle, \langle a_{pull}, a_{li}\rangle\right) = 0.5 \tag{7.6}$$

However, there is an exception. If both agents listen using the headphones, the location of the tiger does not change. This is expressed in equation 7.7.

$$T\left(\langle L, 2, 4, 0, 0\rangle \middle| \langle L, 3, 3, 0, 0\rangle, \langle a_{li}, a_{li}\rangle\right) = T\left(\langle R, 2, 4, 0, 0\rangle \middle| \langle R, 3, 3, 0, 0\rangle, \langle a_{li}, a_{li}\rangle\right) = 1 \tag{7.7}$$

7.1.5 Ground Level Observations

The ground level observations of the H-MacDecTiger problem consists of only the orientation information and the location of the tiger. This leads to 6 observations; one for facing north, one for east, south, west, one for hearing the tiger on the left and one for hearing the tiger on the right. Both agents have the same set of observations ($\Omega_1^0 = \Omega_2^0 = \{o_0, o_1, o_2, o_3, o_{hl}, o_{hr}\}$).
If both agents listen using headphones or open door(s) by pulling lever(s), they will hear the location of the tiger by observing either o_{hl} or o_{hr}. However, these observations are not accurate. If both agents listen, there is a probability of 0.85 per agent that the correct tiger location is observed. Thus, in that case the probability of both agents receiving the correct tiger location equals 0.85^2. In equation 7.8, you will find an example where both agents listen, and agent 1 observes the correct location, while agent 2 does not.

$$O\left(\langle o_{hl}, o_{hr}\rangle \middle| \langle a_{li}, a_{li}\rangle, \langle L, 3, 3, 0, 0\rangle\right) = 0.85 \cdot 0.15 = 0.1275 \tag{7.8}$$

In all other cases where both agents either pull a lever or listen to headphones, there is a probability of 0.5 to receive the correct observation per agent. I.e. in these cases it is random whether the correct tiger location observation is observed. An example is shown in equation 7.9, where agent 1 listens and agent 2 pulls a lever.

$$O\left(\langle o_{hl}, o_{hr}\rangle \middle| \langle a_{li}, a_{pull}\rangle, \langle L, 3, 5, 0, 0\rangle\right) = 0.5^2 = 0.25 \tag{7.9}$$

In all other cases, there is a probability of 1 that the agents will receive an observation containing the orientation it ends up at. An example of this is shown in equation 7.10, here agent 1 tries to listen while facing a lever, and agent 2 rotates.

$$O\left(\langle o_0, o_2\rangle \middle| \langle a_{li}, a_{ro}\rangle, \langle L, 1, 4, 0, 2\rangle\right) = 1 \tag{7.10}$$

7.1.6 Reward

Figure 24 denotes the rewards given for every joint action. It is the same as for the DecTiger problem, except that the states also account of the correct location and orientation of the agents.

Joint Actions	**States**	**Rewards**
$\langle a_{pull}, a_{li}\rangle$	$\langle L, 1, 3, 0, 0\rangle, \langle R, 5, 3, 0, 0\rangle$	-101
$\langle a_{li}, a_{pull}\rangle$	$\langle L, 3, 1, 0, 0\rangle, \langle R, 3, 5, 0, 0\rangle$	-101
$\langle a_{pull}, a_{pull}\rangle$	$\langle L, 1, 5, 0, 0\rangle, \langle L, 5, 1, 0, 0\rangle, \langle R, 1, 5, 0, 0\rangle, \langle R, 5, 1, 0, 0\rangle$	-100
$\langle a_{pull}, a_{pull}\rangle$	$\langle L, 1, 1, 0, 0\rangle, \langle R, 5, 5, 0, 0\rangle$	-50
$\langle a_{li}, a_{li}\rangle$	$\langle L, 3, 3, 0, 0\rangle, \langle R, 3, 3, 0, 0\rangle$	-2
$\langle a_{pull}, a_{li}\rangle$	$\langle R, 1, 3, 0, 0\rangle, \langle L, 5, 3, 0, 0\rangle$	9
$\langle a_{li}, a_{pull}\rangle$	$\langle R, 3, 1, 0, 0\rangle, \langle L, 3, 5, 0, 0\rangle$	9
$\langle a_{pull}, a_{pull}\rangle$	$\langle R, 1, 1, 0, 0\rangle, \langle L, 5, 5, 0, 0\rangle$	20
Otherwise	*Otherwise*	0

Figure 24: The rewards for the H-MacDecTiger Problem.

7.1.7 Macro-Observations

In this section, all macro-observations and their probabilities of occurring are explained. In figure 25 an overview of all observations and macro-observations are given. At the right of this diagram, the main responsibility of the (abstraction) level is given. E.g. Orientation is the responsibility of the ground level, which is abstracted away by the level above.

Abstraction Level 2 (Ω_i^2)	z_l^2, z_r^2	**Dec-Tiger**
Abstraction Level 1 (Ω_i^1)	$z_1^1, z_2^1, z_3^1, z_4^1, z_5^1, z_{hl}^1, z_{hr}^1$	**Location**
Ground Level (Ω_i^0)	$o_0, o_1, o_2, o_3, o_{hl}, o_{hr}$	**Orientation**

Figure 25: An overview of the (macro-)observations in the H-MacDecTiger problem.

In the set of level 1 macro-observations there are 5 macro-observations for each of the possible locations of the agent, one for hearing the tiger at the left door, and one for hearing the tiger at the right door.

$$\Omega_1^1 = \Omega_2^1 = \{z_1^1, z_2^1, z_3^1, z_4^1, z_5^1, z_{hl}^1, z_{hr}^1\}$$

The macro-observation probability functions of the H-MacDecTiger problem mainly uses the history of the level below to determine the probabilities of their macro-observations to occur, which is deterministic in this problem. First the hearing observations are passed upwards if they are the last perceived observations. If the ground level history passed into Z_i^1 ends with o_{hl}, there is a probability of 1 that z_{hl}^1 is perceived. The same goes for a history ending with o_{hr} which leads to macro-observation z_{hr}^1.
For the location observations i.e. $z_1^1, z_2^1, z_3^1, z_4^1, z_5^1$, the agent's steps are retraced to determine the location. This is done by first only looking at the observations and actions after the last reset (i.e. all observations and actions up to and including those of the time-step of the last o_{hl} and o_{hr} observations). The agent knows that a reset has occurred when it receives either the o_{hr} or o_{hl} ground level observations. Only the actions and observations after the last reset are considered because all movements before resets are no longer relevant when it comes to determining the current location. Thus, to determine the current location of an agent, this shortened history is looped over and every time a movement is done, the location parameter is updated to reflect this change. Recall that a movement is successfully done if an agent faces west or east while performing the forward action (a_f). Note that the macro-action that is given as a parameter to the first level macro-observation probability function remains unused. This is because the ground level actions and observations are sufficient when it comes to determining the current location of the agent or the observation of the location of the tiger.
The second level macro-observations are the same as those in the original Dec-Tiger problem. These macro-observations correspond to hearing the agents on either the left or the right.

$$\Omega_1^2 = \Omega_2^2 = \{z_l^2, z_r^2\}$$

Thus, the z_l^2 observation should be given when the tiger is heard on the left, and z_r^2 should be given when the tiger is heard on the right. The second level macro-observation probability function just looks at the first level history and returns z_l^2 with a probability of 1 if this macro-history ends with o_{hl} and z_r^2 if the macro-history ends with z_{hr}^1. Thus the macro-action is also not used in Z_i^2. This is shown in equations 7.11 and 7.12. All other values of Z_i^2 result in a probability of 0.

$$Z_i^2(z_l^2|\theta_i^1 \textit{ ends with } z_{hl}^1, m^2) = 1 \qquad (7.11) \qquad Z_i^2(z_r^2|\theta_i^1 \textit{ ends with } z_{hr}^1, m^2) = 1 \qquad (7.12)$$

The pseudocode for the first level macro-observation probability function (Z_i^1) is given below;

function $Z_i^1(z^1|\theta_i^1, m^1)$
  if θ_i^1 ends with o_{hl} **then**
    return $z^1 = z_{hl}^1$ ▷ This returns 1 if $z^1 = z_{hl}^1$ holds and 0 if it does not.
  else if θ_i^1 ends with o_{hr} **then**
    return $z^1 = z_{hr}^1$
  else
    $short\_\theta_i^1 \leftarrow$ all observations and actions from time-steps after the last o_{hl} and o_{hr} from θ_i^1
    $location \leftarrow 2i$ ▷ Set location to starting position (2 for agent 1, 4 for agent 2).
    for all $(o, a) \in short\_\theta_i^1$ **do** ▷ Every observation and action per time-step.
      if $(o, a) = (o_1, a_f)$ **then** ▷ If the agent is facing and then moving east.
        $location \leftarrow \min(location + 1, 5)$
      else if $(o, a) = (o_3, a_f)$ **then** ▷ If the agent is facing and then moving west.
        $location \leftarrow \max(location - 1, 1)$
      end if
    end for
    return $z^1 = z_{location}$ ▷ This returns 1 if $z^1 = z_{location}^1$ holds and 0 if it does not.
  end if
end function

7.1.8 Macro-Actions

In this section, macro-actions and their components are explained. In figure 26 all actions and macro-actions are shown.

Abstraction Level 2 (M_i^2)	m_l^2, m_r^2, m_{li}^2
Abstraction Level 1 (M_i^1)	$m_{ml}^1, m_{mr}^1, m_{list}^1, m_{pull}^1$
Ground Level (M_i^0)	$a_{ro}, a_f, a_{li}, a_{pull}$

Figure 26: An overview of the (macro-)actions in the H-MacDecTiger problem.

At the bottom there are the ground level actions that were explained in section 7.1.3. Then there are the level 1 macro-actions, of which two are related to movement, namely the move left and move right macro-actions (m_{ml}^1, m_{mr}^1). These macro-actions abstract away the rotation (a_{ro}) and forward (a_f) ground level actions. The concept of orientation is abstracted away, and the concept of the location of the agent remains. Then there is the level 2 macro-action of pulling a lever (m_{pull}^1), which no longer requires the agent to be facing a particular direction to function. Instead the level 1 pull macro-action (m_{pull}^1) will automatically rotate the agent to the correct orientation before performing the ground level pull action (a_{pull}). The same holds for the level 1 listening macro-action (m_{li}^1).

Equation 7.13 defines the set of level 1 macro-actions and equation 7.14 defines the set of level 2 macro-actions. The set notations used in the macro-action components ($\beta, \mathcal{I}, \pi$) contains the macro-observations the relevant history must end with.

The level 1 macro-actions for moving left or right (m^1_{ml}, m^1_{mr}) work by performing the forward action if the correct direction is being faced and rotating if that is not the case. These macro-actions terminate if the forward action has been taken while facing the correct direction. These two macro-actions can be taken from any location, because walking into a wall is well-defined.
The level 1 listening action can only be performed at the location of the headphones. The agent will rotate until it is facing north and then take the listen ground level action (a_{li}). If the agent hears the tiger (i.e. receives either the o_{hl} or o_{hr} observations), the listening macro-action terminates. The level 1 macro-action for pulling a lever works in a similar manner. The agent must be in a location with a lever (i.e. 1 or 5) and it will rotate to the correct orientation before taking the a_{pull} action. The termination condition is the exact same as that of the level 1 listening macro-action (m^1_{li}).

$$M^1_1 = M^1_2 = \left\{ \begin{array}{ll} m^1_{ml} = & \left(\begin{array}{l} \beta_{m^1_{ml}} : \textit{Any level 0 history ending with } (a_f, o_3) \\ \mathcal{I}_{m^1_{ml}} : \textit{Any level 1 history} \\ \pi_{m^1_{ml}} : \{o_3\} \rightarrow a_f \textit{ else } a_{ro} \end{array} \right) \\ m^1_{mr} = & \left(\begin{array}{l} \beta_{m^1_{mr}} : \textit{Any level 0 history ending with } (a_f, o_1) \\ \mathcal{I}_{m^1_{mr}} : \textit{Any level 1 history} \\ \pi_{m^1_{mr}} : \{o_1\} \rightarrow a_f \textit{ else } a_{ro} \end{array} \right) \\ m^1_{li} = & \left(\begin{array}{l} \beta_{m^1_{li}} : \{o_{hl}, o_{hr}\} \\ \mathcal{I}_{m^1_{li}} : \{z^1_3\} \\ \pi_{m^1_{li}} : \{o_0\} \rightarrow a_{li} \textit{ else } a_{ro} \end{array} \right) \\ m^1_{pull} = & \left(\begin{array}{l} \beta_{m^1_{pull}} : \{o_{hl}, o_{hr}\} \\ \mathcal{I}_{m^1_{pull}} : \{z^1_1, z^1_5\} \\ \pi_{m^1_{pull}} : \{o_0\} \rightarrow a_{pull} \textit{ else } a_{ro} \end{array} \right) \end{array} \right\} \tag{7.13}$$

The level 2 macro-actions are the same as those of the original Dec-Tiger problem. The components of these macro-actions are similar to each other. They all have the exact same initiation and termination conditions. The level 2 macro-action can be taken at any time. The termination condition is that the last observation in the history is a tiger location observation (i.e. z^1_{hl} or z^1_{hr}). All level 2 macro-action policies first involve moving to the correct locations. Then the m^1_{pull} macro-action is taken if a door must be opened and m^1_{li} will be taken if the level 2 listen macro-action was taken.

$$M^2_1 = M^2_2 = \left\{ \begin{array}{ll} m^2_l = & \left(\begin{array}{l} \beta_{m^2_l} : \{z^1_{hl}, z^1_{hr}\} \\ \mathcal{I}_{m^2_l} : \textit{Any level 2 history} \\ \pi_{m^2_l} : \{z^1_1\} \rightarrow m^1_{pull} \textit{ else } m^1_{ml} \end{array} \right) \\ m^2_r = & \left(\begin{array}{l} \beta_{m^2_r} : \{z^1_{hl}, z^1_{hr}\} \\ \mathcal{I}_{m^2_r} : \textit{Any level 2 history} \\ \pi_{m^2_l} : \{z^5_1\} \rightarrow m^1_{pull} \textit{ else } m^1_{mr} \end{array} \right) \\ m^2_{li} = & \left(\begin{array}{l} \beta_{m^2_{li}} : \{z^1_{hl}, z^1_{hr}\} \\ \mathcal{I}_{m^2_{li}} : \textit{Any level 2 history} \\ \pi_{m^2_{li}} : \left\{ \begin{array}{l} \{z^1_1, z^1_2\} \rightarrow m^1_{mr} \\ \{z^1_3\} \rightarrow m^1_{li} \\ \{z^1_3, z^1_4\} \rightarrow m^1_{ml} \end{array} \right. \end{array} \right) \end{array} \right\} \tag{7.14}$$

7.2 Policy Search Space

When given any kind of MDP, generally the goal is to find the best possible policy. There are many policies, of which many are objectively better or worse than others. In order to ease the process of finding a good policy, it would help to (greatly) reduce the amount of policies that can be considered. The amount of policies that must be searched through is referred to as the *policy search space*. In this section, a method is derived which can be used to determine the size of the policy search space. Afterwards, the size of the policy search space is calculated for the H-MacDecTiger problem of section 7.1.

7.2.1 Policy Search Space Calculation

This section describes how the amount of deterministic joint policies can be calculated. There is always an infinite amount of non-deterministic policies, so these are not considered in this thesis. Equation 3.4.5 from [40] can be used to calculate the amount of deterministic joint policies. This equation is also shown in equation 7.15. Here n refers to the amount of agents ($|I|$), $|\mathbb{A}_\dagger|$ denotes the largest individual action set and $|\mathbb{O}_\dagger|$ denotes the largest individual observation sets.

$$O\left(|\mathbb{A}_\dagger|^{\frac{n(|\mathbb{O}_\dagger|^h-1)}{|\mathbb{O}_\dagger|-1}}\right) \tag{7.15}$$

7.2.2 H-MacDecTiger Policy Search Space

The size of the top level policy search space can be calculated using equation 7.15. As an example, it will be assumed that 3 level 2 macro-actions will be taken. This will simulate a regular (see section 3.3.1) DecTiger problem with horizon 3. Agent 1 has three level 2 macro-actions, agent 2 idem. Thus $|\mathbb{A}_\dagger|$ will be equal to 3. Similarly, there are two level 2 macro-observations, leading to $|\mathbb{O}_\dagger| = 2$ level 2 macro-observations. Putting this in equation 7.15 results in:

$$|\mathbb{A}_\dagger|^{\frac{n(|\mathbb{O}_\dagger|^h-1)}{|\mathbb{O}_\dagger|-1}} = 3^{\frac{2(2^3-1)}{2-1}} = 4782969$$

$h = 3$ can be used here because we assume only three macro-actions will be taken. The actual underlying horizon dealing with ground level actions and observations is much higher, as these ground level actions and observations are abstracted away by using H-MacDecPOMDPs. In order to find the required size of the ground level horizon, the maximum amount of ground level actions used in the optimal policy must be found.
As is written in section 3.3.1, the optimal joint policy for the ($h = 3$) DecTiger problem has both agents perform the listen action twice, followed by having each agent either open the door on the side where the tiger is heard twice, or listening again for a third time if the tiger is heard on both sides.
In the H-MacDecTiger problem, the second level macro-actions all involve rotating 4 times and taking either 1 or 3 steps. The full rotation is necessary because any agent must first rotate from the starting position in order to move, and must then complete the rotation in order to use either a lever or headphones. An agent needs to take just one step forward when it either listens, or goes to pull a lever it spawned next to. The other lever that is further away requires three steps forward to reach.
Thus, in the optimal joint policy for solving the H-MacDecTiger problem where three level 2 steps are taken, the third macro-action takes longest to terminate if at least one agent must walk to the lever that is furthest away from it. Such policies take 20 ground level actions to complete. This can also

be seen in the policy of agent 1 and 2 shown in figures 28 and 29 respectively where both agents are forced to open the levers furthest from themselves. Agent 1 hears the location of the tiger correctly and pulls the correct lever while agent 2 misheard the tiger twice, leading him to pull the wrong lever. Figure 27 shows the sequence of states that correspond to these histories. As becomes apparent in these figures, 20 actions must be taken (on timesteps 0 to 19), leading to a horizon of 20 for the longest optimal policy.

If there were no abstraction levels, and policies were planned directly using ground level actions and observations, the policy search space would be much greater. There are four ground level actions per agent, leading to $|A_\dagger| = 4$ actions. Similarly there are 6 ground level observations per agent, leading to $|O_\dagger| = 6$ observations. Filling in these parameters in equation 7.15, leads to the following policy search space size:

$$|\mathbb{A}_\dagger|^{\frac{n(|\mathbb{O}_\dagger|^h-1)}{|\mathbb{O}_\dagger|-1}} = 4^{\frac{2(6^{20}-1)}{6-1}} = 4^{1462463376025190}$$

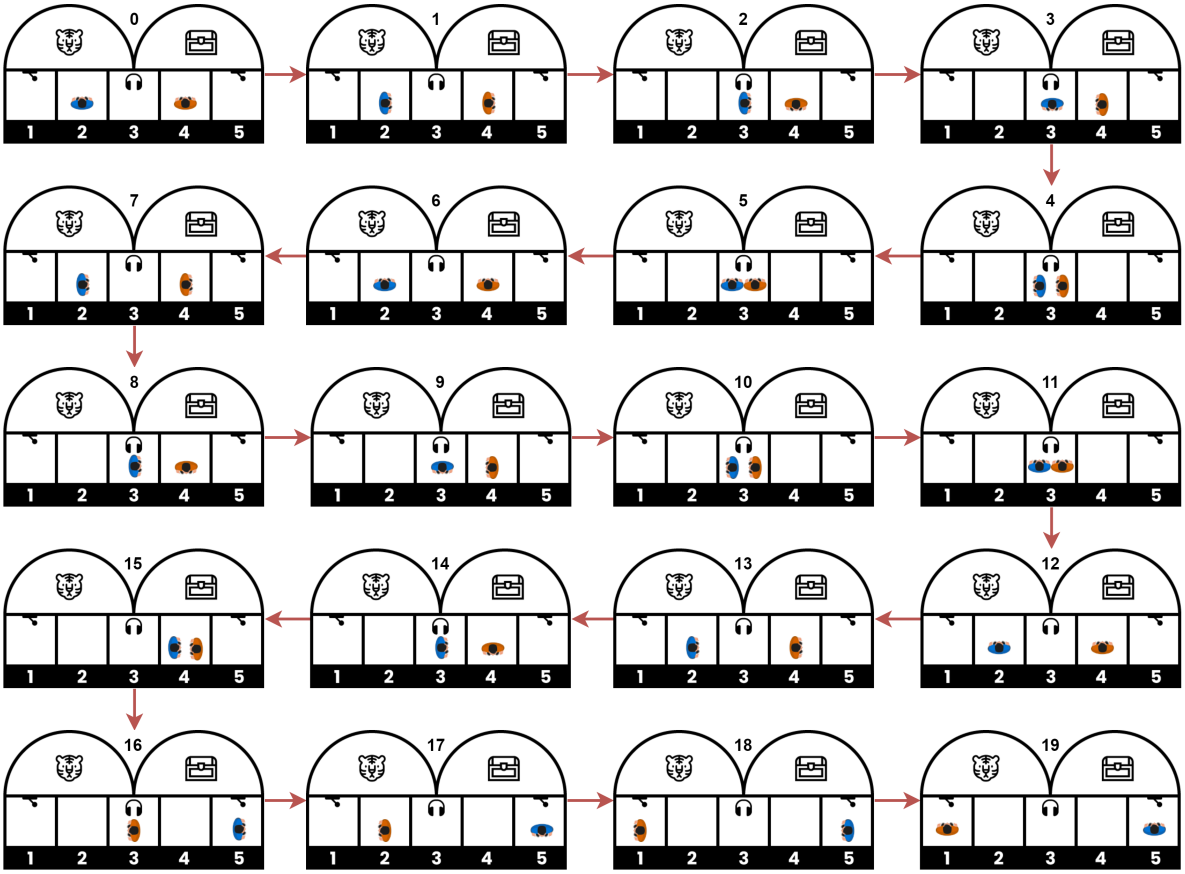


Figure 27: This diagram shows the longest history of states for an optimal policy where three second level macro-actions are taken.

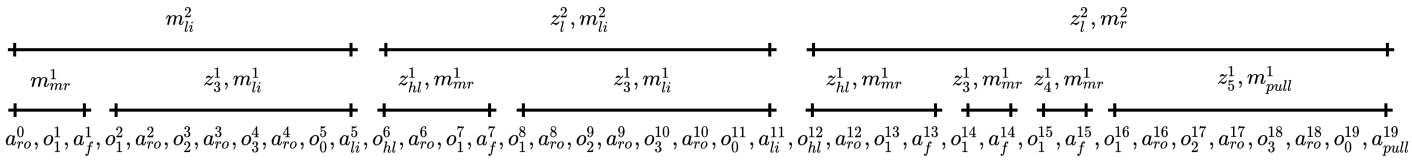


Figure 28: This diagram shows the longest ground action-observation history of agent 1 for an optimal policy where three second level macro-actions are taken.

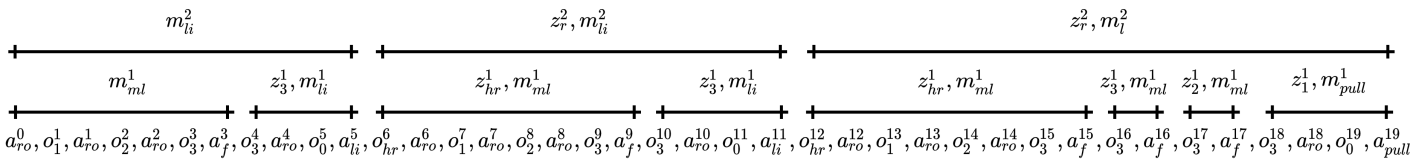


Figure 29: This diagram shows the longest ground action-observation history of agent 2 for an optimal policy where three second level macro-actions are taken.

7.3 Benefits and Drawbacks of Using H-MacDecPOMDPs

With the concept of policy search space in mind (section 7.2), this section answers research question three, namely:

(3) What are the drawbacks and benefits of using a H-MacDecPOMDP as opposed to a (Mac-)Dec-POMDP?

Sections 7.3.1 and 7.3.2 describe two benefits, and sections 7.3.3 and 7.3.4 describe two drawbacks.

7.3.1 Policy Search Space Reduction

As shown in section 7.2.2, the policy search space of a problem can be greatly reduced by using a H-MacDecPOMDP. Reducing the amount of policies to look through, greatly speeds up the process of finding the optimal policy. The H-MacDecTiger problem example had a top level policy search space size of 4782969 due to the defined macro-actions on levels below. Without abstraction levels and these macro-actions and macro-observations, the policy search space size of the remaining Dec-POMDP would have been at least $4^{1462463376025190}$. The addition of more than 1 abstraction levels can enable the horizons of macro-action policies to be smaller. In the H-MacDecTiger example, removing the first abstraction level would have forced the level 2 abstraction level macro-actions to make ground level policies that take orientation into account. These policies must use a much larger portion of the ground level history. This is because much of the information processing work that is done in the macro-observation probability function, must then be done by these policies themselves. Thus the addition of multiple abstraction levels can be beneficial in reducing the size of the total policy search space (i.e. the sum of the policy search space sizes of each level), however the top level policy search space remains unaffected. This could be useful in future research where policies are found for macro-actions.

7.3.2 Improved Human Interpretability

Policies that work with long histories are much harder for humans to understand. It may be hard to understand why certain actions were planned as DecPOMDPs do not offer any additional information to help interpret such policies. Macro-actions and macro-observations can be given clear purposes and very understandable descriptions. Which may help future engineers or scientists to more easily diagnose problems or make better design decisions for their AIs. The first level macro-observations in the H-MacDecTiger problem for example translate orientation and movement data into location data, which is easier to work with. Perceiving a macro-observation that states the location of an agent conveys much more useful information of the state than an observation that contains the agent's orientation. Abstracting away details such as the orientation or even the location of the agent makes it much easier to focus on solving overarching problems.

7.3.3 Requirement of Additional Structure

H-MacDecPOMDPs require constructing macro-actions and macro-observations. Designing such components may require more knowledge of the problem at hand than designing equivalent Dec-POMDPs. This thesis does not provide any automated method to design H-MacDecPOMDP macro-actions and macro-observations. This leads to greater upfront costs in the design phase of the model, as this is done manually, although there is research being done to learn macro-observation probability functions [42] and macro-actions [43]. E.g. the amount of abstraction levels and the amount of macro-action, macro-observations and their purposes are determined manually. Although, if done correctly, these upfront costs should lead to a good return on investment during the planning phase of the model, this greater upfront effort must be kept in mind.

7.3.4 Lack of Existing Efficient or Approximate Planning Methods

The H-MacDecPOMDP model is new and thus does not have any efficient planning methods, the same goes for approximation methods. It is however reasonable to suspect that (variants) of methods used to solve MacDecPOMDPs could also be used to solve H-MacDecPOMDPs. Thus more research should be done to find approaches to more efficiently solve H-MacDecPOMDPs directly. In order to help mitigate this issue, an algorithm is given that can transform H-MacDecPOMDPs into Dec-POMDPs. This algorithm and its explanation can be found in section 6.2. Section 7.4 further discusses this transformation.

7.4 Benefits and Drawbacks of Transforming H-MacDecPOMDPs into Dec-POMDPs

This section answers research question four, namely:

(4) What are the drawbacks and benefits of transforming H-MacDecPOMDPs into Dec-POMDPs?

Subsection 7.4.1 describes a benefit, and section 7.4.2 describes a drawback.

7.4.1 Enable the Use of Existing Planning Methods

The main benefit to the transformation from H-MacDecPOMDPs to Dec-POMDPs is of course enabling the use of existing methods of solving Dec-POMDPs that were created by transforming H-MacDecPOMDP problems. Such methods may include: the use of the MADP Toolbox [18], MAA* [44] Dynamic Programming [45], evolutionary approaches [46] and approximate Q value functions [38].

7.4.2 Increased State Complexity

Dec-POMDPs that are transformed from H-MacDecPOMDPs contain many more states than a run-of-the-mill Dec-POMDP. This is because the transformation procedure generates a new state for every reachable tuple of states and level 0 to l joint histories. Thus the planner that is used to solve the resulting Dec-POMDP has many more states to go through. In other words, the policies that must be planned are much shorter, however the amount of states is much larger than it was in the original H-MacDecPOMDP. The benefits of an exponential increase in performance due to having to find much shorter policies is still there, however it is likely that this benefit is lower than it could be due to the great amount of states in such Dec-POMDPs. It may be possible to reduce this problem, which will be elaborated on in the future work section (section 8.1).

7.5 Experiment

In this section an experiment is desribed which was performed to further validate the correctness of the H-MacDecPOMDP model. In this experiment, a prototype of the H-MacDecTiger problem (section 7.1) is built, which is tasked with finding the best possible deterministic top level joint policy. The brute force approach explained in section 6.1 is used in this experiment, which also explains why the policies had to be deterministic.
Measuring performance of the planner or model is not a goal of this experiment, thus these types of metrics were not gathered. The only relevant result here is whether the optimal top joint policy has indeed been found. The value of a the top joint policy is measured using a deterministic version of the value function given in section 5.13.
The optimal joint policy is the same as that of section 3.3.1 (figure 2), the top level horizon that was used in this experiment was also equal to 3.
The prototype was built in Java, and a link can be found in the footnote of this page [1].
The result of this experiment is that the correct top level joint policy can indeed be found using the H-MacDecPOMDP model.

[1] https://github.com/keesfani/hmacdectiger

8 Conclusion

To conclude, this thesis' findings are given in the form of answers to the research questions.

(1) How can the MacDecPOMDP Framework be extended to higher levels of abstraction?

The MacDecPOMDP Framework can be extended to higher levels of abstraction using the H-MacDecPOMDP Framework explained in this thesis. In this framework, the MacDecPOMDP framework is extended in such a way that macro-actions have policies that lead to other macro-actions. This leads to a hierarchy of macro-actions, which allows planning with an arbitrary number of abstraction levels.

(2) Can H-MacDecPOMDPs be transformed into DecPOMDPs?

H-MacDecPOMDPs can be transformed into DecPOMDPs, to allow the use of existing DecPOMDP solving methods on H-MacDecPOMDPs. This transformation is done by abstracting away levels, this process is referred to as "grounding" levels. In this grounding procedure, macro-states are formed which contains both the state and the histories of all agents of the level below. These macro-states are necessary because they allow the transition, reward and observation functions to take macro-actions as their inputs. If macro-actions are used as inputs to these functions, the histories of the levels below must be known, because these histories determine what the behavior of these macro-actions will be. When all abstraction levels are grounded, the resulting model only needs a few simple heuristics and is then equivalent to a Dec-POMDP.

(3) What are the drawbacks and benefits of H-MacDecPOMDPs as opposed to (Mac-)Dec-POMDPs?

The benefits of H-MacDecPOMDPs are that the policy search space can be greatly reduced. This was shown by comparing the policy search space size of a H-MacDecPOMDP (the H-MacDecTiger problem), which was equal to $4^{1462463376025190}$, to the policy search space of the same problem but without the levels of abstraction (4782969). This may significantly increase performance, because H-MacDecPOMDPs exploit and explicitly construct inherent hierarchical structures present in problems the model tackles. The structure that H-MacDecPOMDPs provide can also make H-MacDecPOMDP policies much easier for humans to comprehend than their Dec-POMDP counter parts. This is in part because macro-observations give higher level information where irrelevant details are abstracted away. Another reason is that shorter policies are themselves easier to comprehend. The requirement of this additional structure can also be a drawback for H-MacDecPOMDPs. More time and effort must be put in to model the problem. Although the solutions to these problems will be found quicker if the problem is well designed, the extra effort required for designing these problems should be thoroughly considered. Another drawback of H-MacDecPOMDPs is that there are of course no designs or implementations of efficient or approximate solving methods.
A drawback of transforming H-MacDecPOMDPs into Dec-POMDPs is that states become more numerous and complex. Ground level states are generally easy to comprehend in DecPOMDPs, but integrating multi-level histories into them does make these resulting Dec-POMDPs more complex than other run-of-the-mill Dec-POMDPs. The benefit is of this transformation is that it allows the use of existing DecPOMDP solution methods to be employed for H-MacDecPOMDPs.

8.1 Future Work

There is still plenty of room for additional research regarding H-MacDecPOMDPs. This thesis defines a value function for H-MacDecPOMDPs, however it does not define an efficient approach for solving H-MacDecPOMDPs directly, i.e. without first transforming it into a Dec-POMDP. Useful places to start would be adapting the planning methods for MacDecPOMDPs discussed in Amato, et al. [17]. These include their Dynamic Programming, Memory-Bounded Dynamic Programming and Direct Cross Entropy Policy Search planning methods.
It would also be useful to research whether efficient direct planning methods for H-MacDecPOMDPs outperform approaches where the H-MacDecPOMDP is first transformed into a Dec-POMDP. The same goes for approximate planning approaches.
Another useful direction to research would be planning policies for macro-actions, and even automatically finding structures ripe for exploitation using macro-actions and macro-observations in Dec-POMDPs, it is not unthinkable that useful macro-actions and macro-observations could be found automatically.
If a H-MacDecPOMDP has already been transformed into a Dec-POMDP, it is impossible to change the policy of a macro-action in an abstraction level without having to do the transformation all over again. It would be useful to create some kind of update procedure that can modify a Dec-POMDP created from a H-MacDecPOMDP to reflect changes that were done to a single macro-action.
The transformation from H-MacDecPOMDPs to Dec-POMDPs can also be done partially, i.e. by not grounding every abstraction level. It could be useful to find out whether performing the grounding procedure on only a few abstraction levels could be beneficial in any sort of way.
Transforming H-MacDecPOMDPs into Dec-POMDPs leads to an explosive increase in the amount of states. It would be beneficial to reduce this. One way this might be done is by researching approaches where entire histories do not need to be included in macro-states. Instead, only the parts of histories that can change the behavior of the agent would really need to be included in macro-states. Recall that the only reason histories are in macro-states at all, is that histories change the behavior of macro-actions and macro-observations, thus it makes sense to only keep the parts of histories that actually do change this behavior. For instance, in the H-MacDecTiger example, the first level history could be almost completely omitted from macro-states. In this case only the last macro-action and macro-observation are necessary because these are the only parts of the history that are looked at by termination conditions and policies of second level macro-actions, as well as the macro-observation probability function of the second abstraction level. The first level macro-states could probably also work if the parts of histories that store observations and actions before "resets" were removed. Recall that states in the H-MacDecTiger problem "reset" when both agents either pull a lever or listen using headphones. Thus the history of actions and observations before histories could be removed from macro-states because the first level macro-observation function does not look any further into the history than that, and termination functions and policies only look at the last action and observation of the histories on the ground level.
It might be possible to remove unnecessary histories from macro-states automatically. This could be done by modifying the *appendHistory* function per abstraction level inside the State Grounding procedure. Every abstraction level would have its own *appendHistory* function which could implement these rules. These new *appendHistory* functions however do need to be made manually per problem per abstraction level, however it is not unthinkable that it might be detectable that parts of histories in macro-states remain unused. Reducing the complexity of states in Dec-POMDPs made from H-MacDecPOMDPs is thus another promising aspect of H-MacDecPOMDPs that should be researched in the future.

References

[1] Stuart Russell and Peter Norvig. A modern, agent-oriented approach to introductory artificial intelligence. *ACM SIGART Bulletin*, 6(2):24–26, 1995.

[2] John G Everett and Alexander H Slocum. Automation and robotics opportunities: construction versus manufacturing. *Journal of construction engineering and management*, 120(2):443–452, 1994.

[3] Fred Sistler. Robotics and intelligent machines in agriculture. *IEEE Journal on Robotics and Automation*, 3(1):3–6, 1987.

[4] Mahyar Amirgholy, Mehrdad Shahabi, and H Oliver Gao. Traffic automation and lane management for communicant, autonomous, and human-driven vehicles. *Transportation research part C: emerging technologies*, 111:477–495, 2020.

[5] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 392–399. IEEE, 2014.

[6] Plan — meaning in the cambridge english dictionary. `https://dictionary.cambridge.org/dictionary/english/plan`. (Accessed on 01/13/2021).

[7] Chelsea C White III and Douglas J White. Markov decision processes. *European Journal of Operational Research*, 39(1):1–16, 1989.

[8] Aurélie Beynier and Abdel-Illah Mouaddib. Applications of dec-mdps in multi-robot systems. In *Robotics: Concepts, Methodologies, Tools, and Applications*, pages 143–165. IGI Global, 2014.

[9] Eitan Altman. Applications of markov decision processes in communication networks: a survey. 2000.

[10] Shun-Pin Hsu and Aristotle Arapostathis. Safety control of partially observed mdps with applications to machine maintenance problems. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 1, pages 261–265. IEEE, 2004.

[11] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[12] Nikos Vlassis. A concise introduction to multiagent systems and distributed artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 1(1):1–71, 2007.

[13] Katia P Sycara. Multiagent systems. *AI magazine*, 19(2):79–79, 1998.

[14] David V Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of artificial intelligence research*, 16:389–423, 2002.

[15] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[16] Doina Precup, Richard S Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, pages 382–393. Springer, 1998.

[17] Christopher Amato, George Konidaris, Leslie P Kaelbling, and Jonathan P How. Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, 64:817–859, 2019.

[18] Matthijs TJ Spaan, Frans A Oliehoek, et al. The multiagent decision process toolbox: software for decision-theoretic planning in multiagent systems. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 107–121, 2008.

[19] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. in 2011 ieee icra, 2011.

[20] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[21] Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510. IEEE, 2015.

[22] Oliver Kroemer, Herke Van Hoof, Gerhard Neumann, and Jan Peters. Learning to predict phases of manipulation tasks as hidden states. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4009–4014. IEEE, 2014.

[23] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight.

[24] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *arXiv preprint arXiv:1912.03558*, 2019.

[25] Jaedeug Choi and Kee-Eung Kim. Hierarchical bayesian inverse reinforcement learning. *IEEE transactions on cybernetics*, 45(4):793–805, 2015.

[26] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

[27] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[28] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[29] Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[30] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[31] Nicholas K Jong and Peter Stone. Hierarchical model-based reinforcement learning: R-max+ maxq. In *Proceedings of the 25th international conference on Machine learning*, pages 432–439. ACM, 2008.

[32] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13(2):197–229, 2006.

[33] Ruoxi Li, Sunandita Patra, and Dana Nau. Decentralized acting and planning using hierarchical operational models. 2020.

[34] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Zhaopeng Meng, Changjie Fan, et al. Hierarchical deep multiagent reinforcement learning with temporal abstraction. *arXiv preprint arXiv:1809.09332*, 2018.

[35] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[36] Andrei Andreevich Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375, 1954.

[37] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[38] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[39] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711, 2003.

[40] Frans A Oliehoek, Christopher Amato, et al. A concise introduction to decentralized pomdps. In *A concise introduction to decentralized POMDPs*, volume 1, chapter 2.3.1, pages 17–19. Springer, 2016.

[41] Frans A Oliehoek and Christopher Amato. Dec-pomdps as non-observable mdps. 2014.

[42] Shayegan Omidshafiei, Shih-Yuan Liu, Michael Everett, Brett T Lopez, Christopher Amato, Miao Liu, Jonathan P How, and John Vian. Semantic-level decentralized multi-robot decision-making using probabilistic macro-observations. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 871–878. IEEE, 2017.

[43] Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357, 2016.

[44] Daniel Szer, François Charpillet, and Shlomo Zilberstein. Maa*: A heuristic search algorithm for solving decentralized pomdps. *arXiv preprint arXiv:1207.1359*, 2012.

[45] Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for dec-pomdps. In *IJCAI*, pages 2009–2015, 2007.

[46] Barış Eker and H Levent Akın. Using evolution strategies to solve dec-pomdp problems. *Soft Computing*, 14(1):35–47, 2010.