Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Risk-aware HTN Planning for Agricultural Tasks

Jan Adomat

Course of Study:	Informatik
Examiner:	Dr. Ilche Georgievski
Supervisor:	Ebaa Alnazer, M.Sc.
Commenced:	May 5, 2023
Completed:	November 6, 2023

Abstract

Agriculture faces unprecedented challenges in a rapidly evolving world, defined by expanding global populations and climatic uncertainties. This necessitates the maximisation of yields, while remaining as environmentally friendly as possible. In addressing these complex obstacles, the implementation of AI within the agricultural domain has emerged as a pivotal factor. One approach is to utilise AIs capacity to handle large volumes of data and generate plans that achieve specific objectives based on multiple factors. HTN planning, a well-established AI planning technique, proves effective in generating efficient plans for real-world situations. This study commences with systematical analysing the domain of agriculture, by looking at irrigation, fertilizing and pest management. We explore how uncertainty, in the form of weather events, affects the irrigation planning. To implement this uncertainty, we use risk-aware HTN planning, which enables decision making based on a probability distribution of the cost of a action and a given risk attitude. We implement our model in JSHOP2 and evaluate it in terms of correctness, scalability and precision. The result is a model, that plans according to a given risk attitude in an efficient and sustainable way, by only using as much water as necessary to maximize the yield of a plant. Furthermore, it establishes a good foundation to expand upon it, with integrating multiple sources of uncertainty in the future.

Kurzfassung

Die Landwirtschaft steht in einer sich rasch entwickelnden Welt, die von einer wachsenden Weltbevölkerung und klimatischen Unwägbarkeiten geprägt ist, vor noch nie dagewesenen Herausforderungen. Daher müssen die Erträge maximiert und gleichzeitig die Umwelt so weit wie möglich geschont werden. Bei der Bewältigung dieser komplexen Hindernisse hat sich der Einsatz von KI in der Landwirtschaft als entscheidender Faktor erwiesen. Eine Art der Verwendung von KI besteht darin, große Datenmengen zu verarbeiten und Pläne zu erstellen, die auf der Grundlage mehrerer Faktoren bestimmte Ziele erreichen. Die HTN-Planung, eine gut etablierte KI-Planungstechnik, erweist sich als effektiv bei der Erstellung effizienter Pläne für reale Situationen. Diese Studie beginnt mit einer systematischen Analyse des Bereichs der Landwirtschaft, indem Bewässerung, Düngung und Schädlingsbekämpfung untersucht werden. Wir untersuchen, wie die Unsicherheit in Form von Wetterereignissen die Bewässerungsplanung beeinflusst. Um diese Ungewissheit zu implementieren, verwenden wir eine risikobewusste HTN-Planung, die eine Entscheidungsfindung auf der Grundlage einer Wahrscheinlichkeitsverteilung der Kosten einer Maßnahme und einer bestimmten Risikoeinstellung ermöglicht. Wir implementieren unser Modell in JSHOP2 und bewerten es im Hinblick auf Korrektheit, Skalierbarkeit und Präzision. Das Ergebnis ist ein Modell, das entsprechend einer gegebenen Risikoeinstellung auf effiziente und nachhaltige Weise plant, indem es nur so viel Wasser wie nötig einsetzt, um den Ertrag einer Pflanze zu maximieren. Darüber hinaus schafft es eine gute Grundlage, um es in Zukunft durch die Integration mehrerer Unsicherheitsquellen zu erweitern.

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

1 Introduction

Throughout history, technological advancements, especially in domains like farming systems, have consistently yielded favorable outcomes [FAO22]. These outcomes include increased productivity, higher incomes, and improved overall human well-being . In our present-day world, the necessity for technological innovations has never been more pressing. We are confronted with the challenge of feeding a rapidly growing global population, while dealing with constraints like the scarcity of agricultural land, unsustainable depletion of natural resources, and the escalating impact of various disruptions, notably climate change. Furthermore, persistent labor shortage in the agricultural sector continues to aggravate the issue [Aha23].

These innovative solutions are paramount for enhancing the effectiveness and long-term sustainability of agriculture, with the overarching goal of augmenting productivity within farming systems . In response to these challenges, there has been a significant shift away from traditional agricultural practices towards the integration of technology within the agricultural sector. These new technologies offer the potential for further enhancing productivity and environmental sustainability, through their integration with automated planning.

Hierarchical Task Network (HTN) planning, a systematic and hierarchical approach to task planning [GNT04], has proven to be particularly valuable in addressing real-world problems across different domains. Its capacity to break down complex tasks into a hierarchy of sub-tasks makes resource and operation allocation more efficient and flexible, making it well-suited for the agricultural sector. Agriculture encompasses various levels of granularity, and farmers must adapt to frequent changes. Using HTN planning to integrate this granularity with compound tasks, which can be decomposed into further compound or primitive tasks, is a logical approach. This decomposition enables the creation of expandable models, that can be further extended with additional decompositions to achieve finer granularity.

Uncertainty is another prevalent aspect of the agricultural domain, often stemming from unpredictable weather conditions or pest-related concerns. As a result, it is imperative to account for these uncertainties. One efficient approach is by utilizing risk-aware HTN planning, that can factor in these uncertainties, ultimately leading to more resilient and adaptive agricultural systems. Risk-aware HTN planning uses utility, as defined by Utility theory [AGA22], to make better informed decision depending on the expected utility of a task. The utility describes profit, but it can also be used as a way of ensuring that we use as few resources as possible. Therefore, in a system where our cost could be water, pesticide and fertilizer usage, using as little as possible leads to better environmental sustainability. Furthermore, by using different values for our variables in the utility functions, we can simulate different risk attitudes, where some might prefer taking risks for bigger gains or always opting for the safe choice.

The structure of this thesis is as follows. We begin with a background chapter, aimed at providing the essential context required to comprehend the subsequent content. This background chapter encompasses a spectrum of topics, from an introduction to automated planning to fundamental

agricultural domain knowledge. Following this, in Chapter 3, we delve into the methodological process of acquiring and categorizing knowledge specific to the agricultural domain. This serves as an important foundation for our subsequent modeling of our domain, particularly in the context of incorporating uncertainty into our model. In Chapter 4, we go over our chosen planner with which we implement our model and provide an in-depth exploration of the models design and structure. Subsequently, we methodically evaluate our implemented model, considering various categories outlined by the used evaluation framework. To conclude our thesis, we present a thorough analysis of relevant literature, followed by a concluding section that summarises our findings and insights. Additionally, we offer an outlook on the potential future expansions and applications of our model.

2 Background

In this chapter, we discuss the necessary background information to understand this study and realize its merits. Beginning with defining what *automated planning* is which *HTN planning* is a part of. After that we take a look at *risk-awareness* in HTN planning, which is the focus of this study, and present how it helps us refine an HTN model to make even better decisions. Finally, we provide a short introduction into the domain of agriculture, by going over Irrigation, Fertilizer and Pesticides.

2.1 Automated Planning

Planning is a process of deliberation that seeks to identify and organise a series of *actions*, with an estimate of their expected outcomes, in order to achieve defined objectives [GNT04]. The true benefits of planning are realised when complex tasks and goals must be achieved, rather than simple, familiar everyday tasks where planning occurs implicitly and unconsciously. These complex tasks and goals are present in many systems. For example, if we have many actors inside a system like a warehouse where multiple robots move boxes, we would not want them acting without a concrete plan, otherwise crashes or missing boxes could be a result. One way to solve the need for plans efficiently is automated planning [GNT04], meaning we let an AI decide for us a course of action to reach a predefined goal.

To be more precise, automated planning, which is a sub area of Artificial Intelligence (AI) and stands as one of its cornerstones, deals with orchestrating strategies and actions to achieve a pre defined goal in a satisfactory and autonomous way. This involves abstract, explicit reasoning that selects and organises actions by anticipating their expected outcomes which results in a concrete *plan* [GNT04].

To create such a Plan for a specific problem, we need to use a so called planner. This Planner uses a *Planning Domain* and *Planning Problem*, which contains information about the initial state and a goal state, as an Input [GA16]. The starting point of every planning problem is called the *initial state*, it describes the state in which the world exists when we start acting upon our designed plan. The *goal state* is our target, it describes how the world should be after we finished following the actions outlined by our plan. To get to this goal state from our initial state, we have to use a set of actions. Actions change the world state through effects and can be used depending on a number of prerequisites, so its up to our planner to find a sequence of actions that will change the world state from out initial state to our goal state.

For example, if we look at our warehouse robots again, the initial state could be the position of every robot when our planner is invoked and the goal state would be to move each robot to its designated box, moving the shortest path possible without crashing. The actions in this example could be, picking up a box or turning the robot to a new direction, which we could let our Planner string into a sequence of actions which would be the Plan we get as a Output.

Automated planning should only be employed in scenarios, where a straightforward list of actions to achieve a goal cannot be created with basic reasoning, as it is a resource-intensive process. Instead, it should be employed only when computational power is necessary to expose all conceivable solutions and find the optimal one.

2.2 HTN Planning

Hierarchical Task Network (HTN) planning is an artificial intelligence planning technique that operates under the same general concept discussed in the preceding section. Consequently, it utilises the same building blocks. The fundamental concepts of this technique comprise an initial state description, an initial task network serving as the primary objective, and domain knowledge consisting of networks of primitive and compound tasks [GA15]. We will now discuss in more detail, the different parts of HTN and their usage, and finish with a list of definitions, which summarize our explanations.

A Task Network comprises a hierarchy of tasks that may be primitive or compound. Primitive tasks can be executed directly, while compound tasks must first be decomposed into subtasks using methods. The initial task network includes the goal tasks that requires completion. Breaking down the initial task network, is the preliminary stage in the planning process. This continues until all compound tasks are decomposed, and an appropriate solution is obtained. The solution comprises a sequence of primitive tasks, that can be executed in the initial state of the world and achieve the initial task network.

If we take our warehouse example, then a compound task could be MoveBoxes which consists of PickUpBox and then MoveBoxToTarget as possible primitive tasks of our Plan, as can be seen in Figure 2.1.

Obviously, for a successful execution of the PickUpBox task, checking for the presence of a box is essential to create a correct plan in this instance, otherwise, the Robot will not be able to perform the task accurately.

In order to facilitate communication regarding feasible and unfeasible actions, it is necessary to record the world state in a manner that is capable of processing. To accomplish this task, HTN and other AI planning techniques, such as classical planning, utilise preconditions. Precondition describe a particular state, that is either true or false, like being in the right position to pick up an existing box. Each Precondition consists of predicates, particularly ground predicates, which represent a specific type of predicate. According to [GA15], a predicate is made up of a name and an ordered list of terms, which can be either constants or variables. The result of a predicate can only be true or false, for instance, if a specific box is missing, a predicate describing its presence would return false. However, this could change in the future if the box is moved to the desired location,

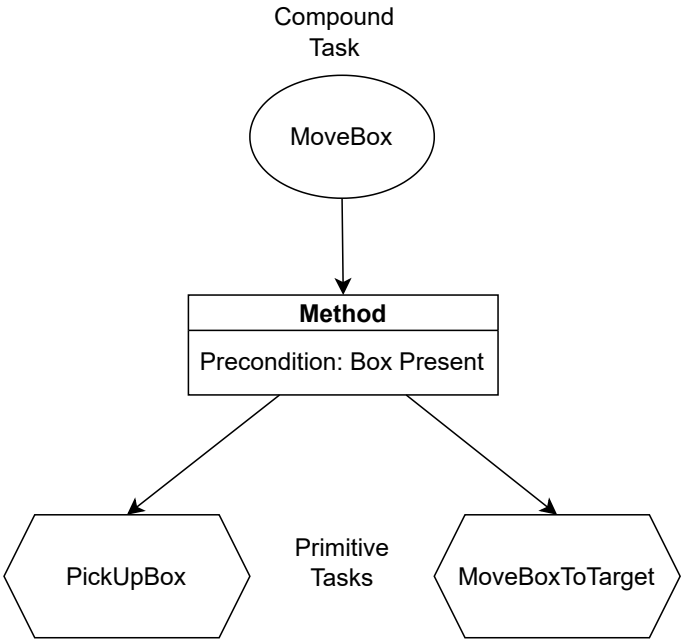

Figure 2.1: Simple Example for an HTN model

where it can be picked up. A ground predicate is a type of predicate that does not include variables, but instead only includes constants, such as a specific location for a box instead of multiple variable location options.

Now that we have learned how to convey information about the condition of our environment, the next step is to determine how to achieve the various compound tasks in our task network. HTN makes use of Methods to achieve this, wherein a Method decomposes a compound task into different types of tasks, which can be a list of primitive or compound tasks, if its preconditions are met.

After continuously selecting the appropriate method to decompose all the present compound tasks in the task network, based on the present world state conveyed through Predicates, we are left with a list of primitive tasks. This list of primitive task each have operators that executes them, achieving our initial task network.

For instance, we could have the primitive task PickUpBox, which requires a box to be present and for us to be in a position to pick it up. If the preconditions are met, we will execute the operator PickUp, which simply picks up the box without checking it or dealing with any consequences. The effects are handled by the primitive tasks, which modify the world state depending on the impact subsequent to the completion of our operator.

Now that we have established the fundamental elements, we shall proceed to discussing the kinds of HTN planners. HTN planners come in various types, dependent on their search space, but for this thesis, we will concentrate on state-based planners. This will allow for a comprehensive explanation of their characteristics. A state-based planner is searching for a state that can achieve the goal task network from the initial state. This is accomplished through decomposing compound tasks until it is no longer possible to decompose further, resulting in a failure to find a solution, or we are left with primitive tasks that form a primitive task network.

If we take a look again at Figure 2.1, an initial task network could be MoveBox(robot, box, target), which our planner would then try to decompose. Assume we have the Predicate BoxPresent(box, true), then the planer would decompose MoveBox(robot, box, target) with the Method to the task network PickUpBox(robot, box) and MoveBoxToTarget(robot, box, target). Now that we decomposed all the compound tasks of our initial task network and the resulting task networks, we are left with the solution to our primary objective.

To formalise the concepts discussed in this chapter, here are definitions from [GNT04] and [GA15] that provide a more formal perspective.

Definition 2.2.1 (Predicate)
A predicate p is defined as $p = \langle name(p), terms(p) \rangle$*, where:*

- *name(p) is the predicates name.*
- *terms(p)* $= \langle \tau_1, ..., \tau n \rangle$ *is an ordered sequence of terms, for which applies:* $\forall i \in \{1, .., n\} : \tau_i \in C \cup V$*. C represents the set of constants, while V represents the set of variables.*

A predicate can only evaluate to the values true and false.

Definition 2.2.2 (Ground Predicate)
A ground predicate is a predicate in which all of the terms are constants, meaning a ground predicate p is defined as: $p = \langle name(p), terms(p) \rangle$*, where* $terms(p) = \langle \tau_1, ..., \tau_n \rangle$ *and* $\forall i \in \{1, .., n\} : \tau_i \in C$

Definition 2.2.3 (State)
A state s is a set of ground predicates p_i*, where* $s = \langle p_1, p_2, ..., p_n \rangle$

Definition 2.2.4 (Primitive Task)
A primitive task $t_P \in T_p$ *is defined as:* $t_p = \langle name(t_p), terms(t_p) \rangle$ *, where:*

- *name(*t_p*) is the predicates name*
- $terms(t_p) = \langle \tau_1, ..., \tau_n \rangle$ *is an ordered sequence of terms.*

Definition 2.2.5 (Compound Task)
A compound task $t_c \in T_c$ *is defined as:* $t_c = \langle name(t_c), terms(t_c) \rangle$

The various components hold an equivalent definition as for primitive tasks.

Definition 2.2.6 (Task Network)
A task network is defined as $t_n = \langle T_n, C \rangle$*, where:*

- $T = T_p \cup T_c$*, meaning T is the finite set of all the primitive and compound tasks*
- *C is a set of constraints that must be satisfied by the tasks in T*

For this thesis, we will solely utilise the precedence constraint, which utilises an expression in the form of $u \prec v$*, in which 'u' and 'v' refer to tasks. The expression* $u \prec v$ *means that we have to finish with u before we can start with v.*

Definition 2.2.7 (Predicate Evaluation)
Evaluating a predicate p in a state s is done in the following way:

$$Evaluation(p, s) = \begin{cases} true, \text{ if } p \in s, \\ false, \text{ else} \end{cases}$$

Definition 2.2.8 (Operator)
An Operator o is defined as $o = \langle t_p(o), pre(o), eff(o), c(o) \rangle, where$:

- $t_p(o)$ *is a primitive task associated with the operator*
- *pre(o) are Preconditions which in turn are a collection of predicates that must be true in the current environment to execute the operator:*
  - $\forall p \in pre(o) \wedge$ *s is a state* $\wedge Bool = \langle true, false \rangle : Evaluate(p, s) \in Bool$
- $eff(o) = \langle eff^-(o), eff^+(o) \rangle$ *is the effect the operator has on the world state if its executed, this effect is shown in the predicates where some are deleted or turned false while others are added or turned true depending on the operator. Predicates deleted/turned false are denoted by* $eff^-(o)$ *and predicates added/turned true are denoted by* $eff^+(o)$*, thus we can define the new state s' after using the operator o starting from state s with* $s' = (s \setminus eff^-(o)) \cup eff^+(o)$

Definition 2.2.9 (Method)
A method $m \in M$ *is defined as* $m = \langle t_c(m), pre(m), t_n(m),$ *where:*

- *M is a set of methods*
- $t_c(m)$ *is the compound task which the method decomposes*
- *pre(m) are preconditions as defined in Operator*
- t_n *is the resulting task network we get by decomposing the compound task* t_c *with the method m*

Definition 2.2.10 (HTN planning Domain)
An HTN planning domain d is defined as $d = \langle O, M \rangle$*, where:*

- *O is a set of operators*
- *M is a set of methods*

Definition 2.2.11 (HTN planning Problem)
An HTN planning problem P is defined as $P = \langle s_0, t_{n_0}, d \rangle$*, where:*

- s_0 *is the initial state of the environment*
- t_{n_0} *is the initial task network*
- *d is the HTN planning domain*

Definition 2.2.12 (Solution)
A sequence of operators $o_1, ..., o_n$ *can be executed in a state s if there is a sequence of states* $s_0, ..., s_n$ *(also known as a trajectory), where:*

- $s_0 = s$ *and* o_i *is applicable in* s_{i-1}

- $\forall 1 \leq i \leq n : s_{i-1}(o_i) = s_i$ *meaning if we execute operator* o_i *in state* s_{i-1} *we end up in state* s_i

A solution π *of an HTN planning problem P is a sequence of operators* $o_1, o_2, ..., o_n$ *which are executable in* s_0 *by decomposing* t_{n_0}.

2.3 Considering Risk in HTN planning

What we observed in the preceding sections were decisions that were binary, as to whether a box was present or not. However, we did not consider the broader implications of our decisions, such as the potential for delays in delivering other boxes and subsequent ripple effects if we pick up an extra box. Thus, we did not evaluate the *risks* associated with our decisions. Risk must be considered in specific domains where the quality of plans are important [AGA22], particularly in agriculture where it is ever-present due to factors such as weather and pests. Therefore, it is crucial to include risk assessment and management in agricultural planning. Risk can be defined as:

Definition 2.3.1 (Risk [AGA22])
Risk is a decision-making situation in which either all outcomes and their probability of occurrence are known a priori or the probability distribution of outcomes is unknown but can be deduced using statistical inference.

For this thesis, the suggested solution cited from [AGA22] implementing the *expected utility theory* will be used. This decision theory model is chosen as it allows for the selection of actions while taking into account their potential costs probability and a risk attitude expressed via a utility function.

In decision theory, individuals assess options using expected utility, a measure that combines the desirability of potential outcomes (utility) and their likelihood of occurring (probability). The framework enables rational decision-making despite uncertainty, as individuals strive to maximise their expected satisfaction or desirability.

The origin of risk arises from operators that induce risk. For instance, when striving to efficiently water plants without overusing resources, one must consider the risk of rainfall as well. The watering operator is expected to result in costs, be it in terms of monetary expenses or environmental damage. In order to avoid wasting resources, it is necessary to minimize these costs but with the potential for rain, the cost of the watering operator becomes uncertain. Therefore, we will adopt *cost-variable operators* to determine the most cost-effective plan, even amid uncertainty.

Cost-variable operators mean that operator costs are modelled as a probability distribution, rather than having a static cost. For instance, watering can incur multiple potential costs, each with a distinct likelihood of occurrence. This means that an operator can have various potential costs, depending on the probability distribution. Nonetheless, it is crucial to choose the appropriate cost, regardless of it being the highest. Highest, as the costs are denoted by negative values, hence larger values are preferred.

Definition 2.3.2 (Cost-Variable Operator [AGA22])
A cost-variable operator o is defined as a tuple $o = \langle t_p(o), pre(o), eff(o), c(o) \rangle$, *where* $t_p(o)$, *and pre(o) are defined as before, and eff(o), and c(o) are tuples that represent the effects and the costs of the operator, respectively and are defined as follows:*

- $eff(o) = \langle (p_1(o), eff_1(o)), (p_2(o), eff_2(o)), ..., (p_n(o), eff_n(o)) \rangle$
- $c(o) = \langle (p_1(o), c_1(o)), (p_2(o), c_2(o)), ..., (p_n(o), c_n(o)) \rangle$

such that eff(o) are the variable effects of the operator and c(o) is the variable costs of the operator. Furthermore the following conditions need to be fulfilled :

- $eff_i(o)$ *and* $c_i(o)$ *are the ith effect with its corresponding cost, respectively*
- $\forall n > 0 \forall i \in [1, n], 0 < p_i(o) < 1$
- $\sum_{i=1}^{n} p_i(o) = 1$
- $c_i(o) < 0$

There are two risk-sensitive attitudes, risk-seeking and risk-averse [AGA22]. Risk seeking might involve refraining from insurance if the risk is low, since it could be cheaper to go without it. Conversely, risk-averse individuals may opt for the safest option to decrease potential damage, even if it requires more resources, such as additional insurance premiums or taking a longer but safer route. Risk attitudes may impact how individuals evaluate the utility and probability of potential outcomes to optimize their expected utility.

These two attitudes can be further divided between static and dynamic risk attitudes. For this work we use static risk attitudes, meaning the the attitude does not change during planning. The utility function transforms a operator cost into a value for the utility of this operator.

Definition 2.3.3 (Utility function for static risk attitude)
We define the utility function u for an operator o with the cost c(o) as:

$$u(c(o)) = \begin{cases} c(o) & \text{if neutral} \\ \frac{a(e^{a \alpha c(o)} - 1)}{\alpha} & \text{else} \end{cases}$$

where:

- *a is an attitude-determinant coefficient*
- α *is a curving coefficient changing the shape of the utility function*
- *the (-1) is added to normalize the function*

This value changes depending on the risk attitude, with a>0 describing a risk-seeking attitude while a<0 representing a risk-averse attitude. In the case of risk-averse, negative costs are way bigger compared to risk-seeking. With α we can control the intensity of the specific attitude, meaning how extreme they act like their given attitude. For risk-averse with $\alpha = 1$, we weigh negative costs extremely low, while with $\alpha = 0.05$ we still value them way worse than risk-seeking would, however it is more moderate. For risk-seeking we do not fear negative costs as much, in the case for $\alpha = 1$ we even weigh them all the same after a certain negative cost.

With the utility function defined, we can now define the Expected utility, which additionally takes the probability of each utility into account.

Definition 2.3.4 (Expected Utility for one cost-variable operator)
We define the expected utility $EU(o)$ for an operator o with the costs $c_1(o)$ and $c_2(o)$ and their probabilities p_1 and p_2 as:

$$EU(o) = p_1 * \frac{a(e^{\alpha c_1(o)})}{\alpha} + p_2 * \frac{a(e^{\alpha c_2(o)})}{\alpha}$$

A planner would calculate the EU for each probability from the given probability distribution, considering its associated cost. For instance, suppose we have an operator for watering that has a probability distribution indicating the cost of watering for forecasted rainfall. The cost for each scenario in the probability distribution is derived by calculating it or it could be given. For instance, the negative cost will be close to zero when rainfall is not predicted and the plant is in need of water. Using the given costs and their corresponding probabilities, we compute the expected utility (EU) of irrigating. We obtain multiple EU values equivalent to the number of cases in our probability distribution. The option with the highest expected utility is selected as the optimum course of action.

This all leads us to the definition of Risk-aware HTN Planning, which we use in this work:

Definition 2.3.5 (Risk-aware HTN Planning [AGA22])
A risk-aware HTN planning problem is a 4-tuple $P_r = \langle s_0, t_{n_0}, D, U \rangle$, where s_0 is the initial state, t_{n_0} is the initial task network, $D = \langle O, M \rangle$ is a risk-involving planning domain consisting of cost-variable operators O and a set of methods M, and U is a utility function that expresses a certain attitude ATT by evaluating the operator costs. A plan π is a solution to P_r if and only if π has a maximum expected utility $EU(\pi)$ that reflects the chosen attitude ATT.

2.4 Precision Agriculture

There are a multitude of definitions what exactly precision agriculture is. However the International Society of Precision Agriculture defined in 2021 as following:

> *Precision Agriculture is a management strategy that gathers, processes and analyzes temporal, spatial and individual data and combines it with other information to support management decisions according to estimated variability for improved resource use efficiency, productivity, quality, profitability and sustainability of agricultural production* [ISP].

It is evident from the definition, that technology is utilized for information gathering and processing to optimize the utilization of resources. However, before acting upon the gathered information, we must first specify the data that needs to be collected and how to obtain it. In the ensuing sections, we will accomplish this by examining irrigation, fertilization, and pest management and detailing the types of information we need to collect.

2.4.1 Understanding Soil Moisture

Effective agriculture and plant management rely on a comprehensive understanding of *soil moisture*. Soil moisture, defined as the amount of water in the soil, has a critical impact on plant growth, soil temperature, chemical transport, and groundwater recharge [DTS17]. This subsection examines the key parameters used to measure soil water content, including *volumetric water content* and *soil matric potential*, highlighting their relevance in agricultural and irrigation practices. In order to determine the most effective irrigation strategies, it is essential to investigate essential *soil moisture thresholds*, including *saturation*, *field capacity*, and the *permanent wilting point*. These key concepts serve as the basis for our analysis of *usable field capacity*, which is essential for proper management of soil moisture to achieve successful crop cultivation.

We will begin with a description what soil is, specifically the *soil matrix*. The term soil matrix typically refers to the solid component of the soil, which consists of mineral particles, organic matter, and various microorganisms [BS00]. It is the non-fluid portion of the soil that provides structural support and forms the framework within which water and air are distributed in the soil. The soil matrix consists of various soil particles, including sand, silt, and clay, along with organic material like decayed plant and animal matter. The interplay between the soil matrix and the pore spaces containing air and water is crucial to grasp soil functioning and its ramifications for plant well-being.

Now with a basic grasp on the soil matrix, we can take a look at how water interacts with the soil matrix. For that we will use following definitions from [DTS17].

Two widely used parameters for the quantification of soil water content are volumetric water content, and soil matric potential. The volumetric water content(VWC) refers to the proportion of water volume to soil volume. It can be denoted as a ratio, percentage or depth of water per unit soil depth (assuming a uniform surface area). Soil matric potential (SMP), also known as soil suction or soil water tension, describes the forces that keep water molecules bound to solid particles and to one another in soil pores. This has implications for plant growth and survival, since it limits the movement of water throughout the soil matrix. A force greater than the SMP is necessary to extract water from it. As water is extracted from the soil, the remaining water is held more tightly, thus increasing the difficulty of water absorption by plants through their roots. Therefore SMP rises as water is extracted from the root region of the plant.

Soil moisture thresholds indicate specific levels of water availability for plant growth. These thresholds are utilised to determine the appropriate quantity and timing of irrigation.

Saturation refers to the point at which all pore spaces, located between soil particles, are saturated with water. The volumetric water content (VWC) at this threshold varies from 30 percent in sandy soil to 60 percent in clay soil. The soil moisture potential (SMP) at saturation is less dependent on soil texture and is close to zero. This indicates that there is a minimal hindrance to the movement of water, enabling plant roots to easily access water from the soil with little energy expenditure.

Field capacity (FC) is the point, at which larger pores have drained out water due to the force of gravity. It is not desirable to irrigate when the soil moisture goes beyond the FC threshold, as the extra water will seep down to lower layers and will not be accessible by plant roots. At field capacity, the moisture content of the soil is deemed to be optimal for crop growth. Therefore, it is typically regarded as the upper limit for irrigation management. At this threshold, the typical volumetric water content varies from 20 per cent in sandy soils to 40 per cent in clay soils.

The Permanent Wilting Point (PWP) represents the level where plants are unable to extract water at a sufficient rate to match their water requirement. Soil particles hold the water so tightly at PWP that it becomes problematic for roots to access it. At this threshold, plant *transpiration* and subsequent processes are impacted. The reduction in water uptake by plants, leads to a near standstill of other processes essential for plant survival, resulting in a considerable decrease in crop growth and yield. If soil moisture remains below the permanent wilting point (PWP) for an extended period, the plant will eventually perish. Therefore, irrigation should be applied well before soil moisture starts approaching the PWP. The PWP value varies depending on the plant type, soil, and climate. When presented as VWC values they range from 7% in sandy soils to 24% in clay soils.

These different variables are needed to calculate the usable field capacity nFK [Wet], which we will later use as our current and ideal soil moisture value. It can be defined as following:

Definition 2.4.1 (Usable Field Capacity)
The usable field capacity (nFK) describes the current soil moisture SM as:

- $nFK = (SM - PWP)/(FC - PWP)$

The values for nFK range from 0-100%, with values going above 100% in the case that SM is above FC.

Another important aspect of understanding soil moisture, is how the soil looses moisture. This process is called *Evapotranspiration*, which is the combination of two processes, namely evaporation and transpiration [APRS+98]. Evaporation describes the water lost through the process of vaporization, meaning liquid water gets converted into water vapour and is therefore removed from the soil. Transpiration is the process in which liquid water present in plant tissues, vaporizes and is subsequently released into the atmosphere. The primary route for water loss in most crops occurs through the stomata, which are tiny openings on the surface of plant leaves that allow for the exchange of gases and water vapor.

2.4.2 Plant Nutrition: Understanding and Assessing Nutritional Needs

A substance essential for the growth and reproduction of plants is referred to as a plant nutrient [BP15]. It is crucial for producing healthy crops and supporting the plants physiological functions. It is important to note that plant nutrients can be obtained from a variety of sources, including soil, fertilisers, and organic matter. Thus, the proper management of plant nutrients is necessary to maintain soil fertility and ensure sustainable agricultural practices.

There are Seventeen elements which can be classified as essential plant nutrients. Three of these elements are obtained from air or water, specifically Carbon, Hydrogen, and Oxygen. The remaining elements are acquired from the soil or nutrient solutions. One of these elements and the first to be classified as essential, is nitrogen which we will take as an example for the remaining section. Nitrogen plays a crucial role in multiple phases of a plants metabolism and can be mostly found nitrogen-containing proteins [BP15].

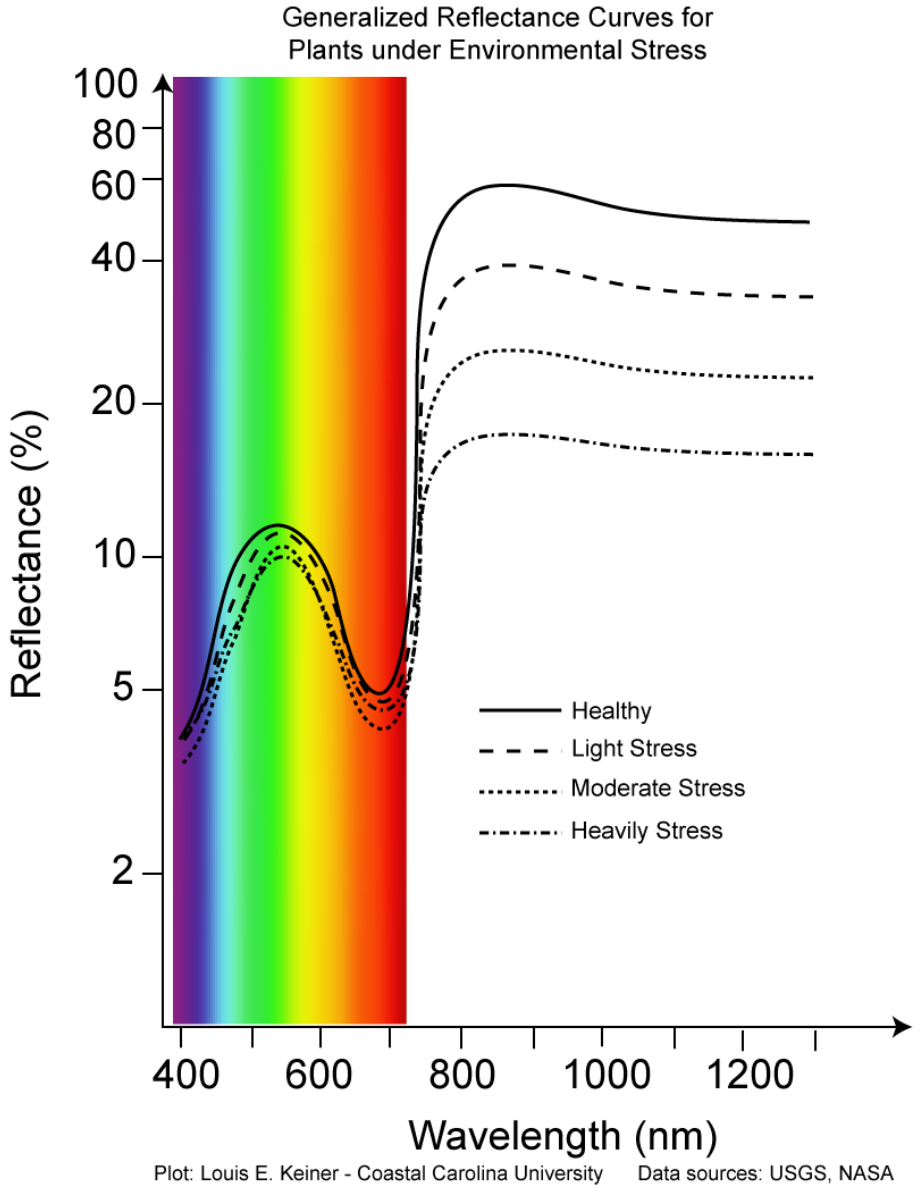

Figure 2.2: Generalized Reflectance curves from Louis E. Keiner - Coastal Carolina University [Kei] for plants under various environmental stress levels

To determine whether a plant is suffering from nitrogen deficiency or excess, one can observe its appearance. Metabolic disruptions due to nutrient deficiencies establish connections between the function of an element and the manifestation of a particular visible anomaly in plants. Disorders' symptoms offer guidance to recognize nutritional deficiencies in vegetation. A generalized example can be seen in Figure 2.2.

For example, nitrogen is essential for the production of proteins and chlorophyll. Interference with these processes leads to symptoms such as pale green or yellow leaves that start at the bottom and spread upwards or occasionally cover the entire plant, indicating a shortage of nitrogen.

Other visual cues for symptoms of a nutritional deficiency can be spotted depending on the type of element missing, like necrosis or stunted growth [Ben93]. However, the absence of essential elements can result in various symptoms [BP15]. Therefore, it is imperative to employ tools to analyse and identify the missing elements.

2.4.3 Pests and Pesticide

Pesticides play a crucial role in modern agriculture, aiming to prevent, destroy, repel, or mitigate various pests that threaten crop yields and, consequently, food production. The term pesticide encompasses a wide range of chemical compounds or combinations used for this purpose, as highlighted by Bernardes in a study from 2015 [BPPD+15].

Categorizing pesticides becomes essential for efficient pest management, and this categorization is typically based on the type of pests they target. Among the various categories, three stand out as the most common:

- Insecticide: These are designed to combat insects, inhibiting their growth or survival. Insect pests can be particularly damaging to crops and, therefore, necessitate effective control measures.
- Herbicide: Targeting weeds, plants, and grasses, herbicides serve as a vital tool for maintaining crop health. Unchecked, these undesired plant growths can choke out cultivated crops.
- Fungicide: Fungi, while not as visible as insects or weeds, can wreak havoc on crops. Fungicides are crucial for protecting plants against fungal infections.

However, it is worth noting that the label pest can be somewhat subjective and lacks ecological authenticity, as highlighted by Metcalf in a 1994 publication [ML94]. In certain ecological contexts, some insects can even provide benefits, while in others, they prove to be troublesome pests. The key determinant often lies in competition for resources. When an insect competes with humans for a resource and exists in abundance, it is usually considered a pest.

The scale of the pest problem is significant, as illustrated by the Food and Agriculture Organization (FAO), which estimates that pests are responsible for destroying up to 40% of the worlds crops and causing staggering losses, totaling $220 billion[FAOa]. This highlights the critical importance of effective pest management to ensure efficient resource usage and maximize agricultural yields, a point emphasized in a study by Pereira in 2016 [PCM+16].

However, the widespread and sometimes indiscriminate use of pesticides, has led to several concerning consequences, including the emergence of pesticide-resistant pests and environmental damage, as underscored in studies like Zelayas work in 2007 [ZOV07]. Consequently, it is imperative to seek alternative approaches to pest management that minimize reliance on traditional pesticides. One such approach is integrated pest management (IPM), a system outlined by Metcalf in his 1994 publication [ML94]. Numerous interpretations of IPM exist within the academic discourse. However, for the purpose of this discussion, the definition offered by Kogan , shall serve as our point of reference:

> *IPM is a decision support system for the selection and use of pest control tactics, singly or harmoniously coordinated into a management strategy, based on cost/benefit analyses that take into account the interests of and impacts on producers, society, and the environment* [Kog98].

The Principles of IPM can be defined as following[FAOb]:

- Ecosystem Approach: Utilise natural predators and sustainable practices to prevent and tackle pest problems. Introduce approaches such as utilising a wide range of crop varieties, crop rotation techniques, and field cleanliness measures.
- Contingency Planning: Use pesticides only when they are necessary. Invest in crop varieties that are resistant to pests and monitor the use of pesticides carefully.
- Cause Analysis: To achieve effective pest management, it is essential to objectively analyse pest outbreaks, prioritise sustainable methods, and carefully consider the use of biological controls or pest campaigns.
- Surveillance and Response: Implement real-time pest monitoring, establish tracking systems, and develop warning and diagnosis systems to ensure rapid responses.

3 Agricultural HTN planning domain with risk

Now that a foundational comprehension of HTN planning and Precision farming has been established, it is prudent to explore the potential integration of these domains in order to model the precision farming task as a risk-aware HTN planning problem. To solve a problem with automated planning, several phases must be undertaken, as illustrated in Figure 3.1. We will start by acquiring knowledge, followed by analysing the domain requirements. Next, we will model our domain before implementing and evaluating our model.

3.1 Knowledge Acquisition

The knowledge acquisition phase involves acquiring information about the domain we are modelling. To achieve this, we must adhere to a selected framework in order to familiarize ourselves with the domains specifics and the way in which it operates. Since we will work with precision farming as our foundation, we use a lot of embedded information gathering through different kinds of sensors. Furthermore we have a predefined area in which we operate through different autonomous actors like an automatic drip irrigation. All this combined results in a system that is gathering and acting upon knowledge in an autonomous way. This seamless coordinating and communicating leads to a *ubiquitous computing environment* [Kru18]. That is why we choose the conceptual framework for knowledge acquisition, as detailed by Georgievsky and Aiello [GA16], for this study, which is depicted in Figure 3.2.

This framework divides the knowledge about the domain into four categories, each containing sub-categories, as defined below:

- *Behavioural Input* is the information that represents a persons preferences for how a ubiquitous computing environment should behave. Depending on whether the satisfaction of ones wishes is necessary or desirable but not necessary, we distinguish two sub-dimensions of behavioural inputs:
  - *Requests* are examples of desired outcomes that are made in order to enforce obligatory behaviour, change the environment or organise settings. *Declarative goals*, which explicitly express the states of the environments to be created, are a type of request model. They are concerned with what needs to be achieved in a particular situation.

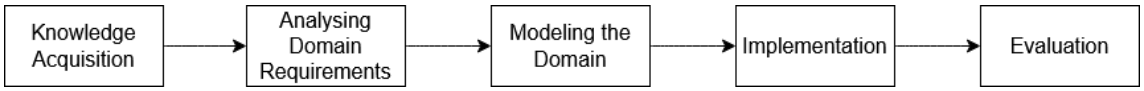

Figure 3.1: Phases of solving a problem with automated planning

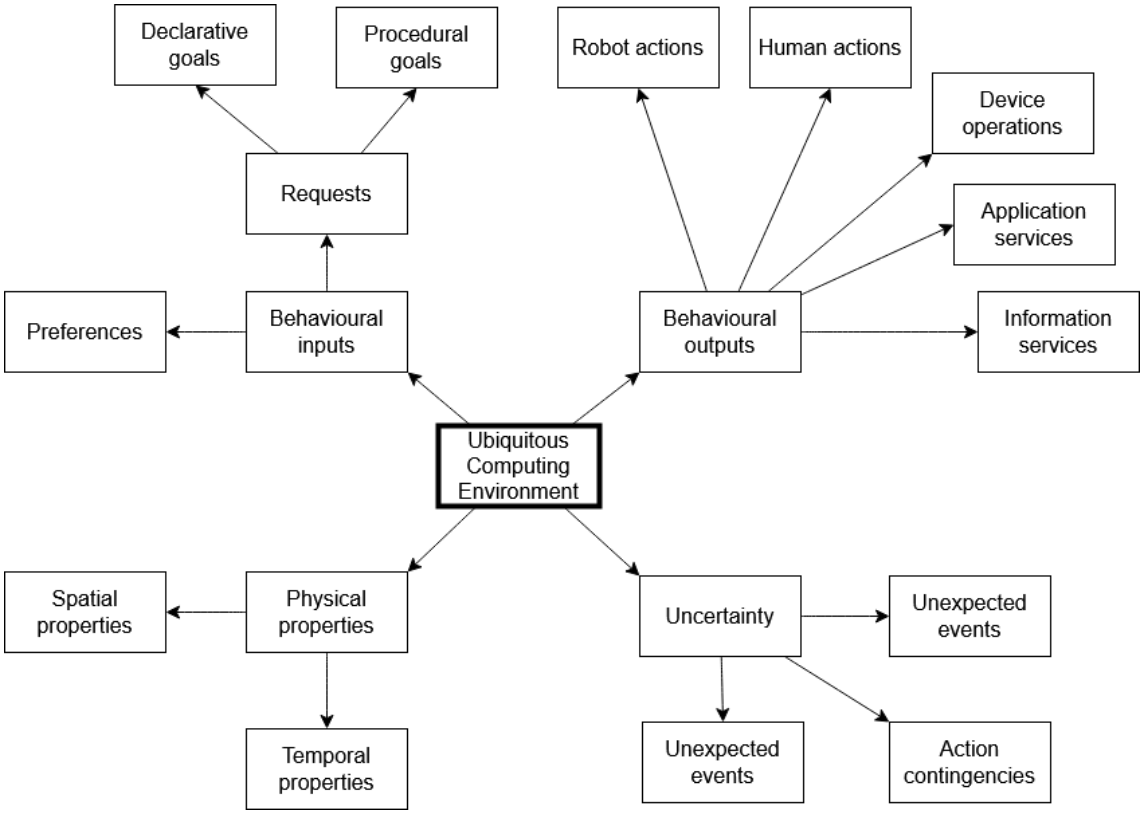

Figure 3.2: Hierarchy of dimension for ubiquitous computing environment adapted from [GA16]

Procedural goals are another type of requirement model that outlines a series of actions that must be taken to fulfil requirements. These goals have to do with how to get something done in a particular situation.

– The *preferences* sub-dimension includes a persons attitudes towards behaviour in relation to the environment. Although preferences are not mandatory, they will be met as far as possible. Preferences are thought of as soft constraints on plans, and their quality increases as more constraints are satisfied.

- *Behavioural Outputs* are defined as actions carried out in ubiquitous computing environments that alter the status of the environments. These actions can be executed by a range of physical objects, including devices, robots, software components, and humans.
- Situations in ubiquitous computing settings possess *physical properties* in terms of both space and time. These physical characteristics serve as bridges connecting individuals, objects, and the surrounding environment. The *spatial properties* determine the correlations between the entities and their environment. The *temporal features* define the interaction of entities with time.
- *Uncertainty* in ubiquitous computing environments stems from the dynamic nature of these settings, resulting in unreliable and uncertain information about the current state. This dynamism encompasses various and ongoing events, unpredictability of behavioural output, and partial observability, collectively defining three sub-dimensions of uncertainty.

- *Unexpected events* are occurrences that take place in highly unusual and entirely unanticipated circumstances, significantly diverging from the norm and manifesting unexpectedly.
- *Action contingencies* represent circumstances during actions where execution does not proceed as planned. These contingencies may consist of instances of failures or timeouts.
- *Partial observability* relates to the imperfections and incompleteness inherent in our knowledge of environmental states. State elements, such as variables, can assume various potential actual values, some of which may remain entirely unknown. Consequently, the determinants of behavioural outputs stop relying directly on these uncertain states.

Now that we have a fundamental understanding of the framework, let us assess how these categories align with the three key pillars of our model: irrigation, fertilisation and pesticide application. For this, we will provide a more detailed overview of each of the key pillars and afterwards categorise them using the predetermined categories. The resulting table can be found in Table 3.1.

3.1.1 Declarative Goals

There are various declarative goals that could express the desired state of our environment. Depending on our primary focus, be it economy, ecology or time, different goals can be established. In this thesis, the emphasis will be placed on *plant wellness* to attain optimal crop yield whilst maximizing resource efficiency. Plant wellness is composed of the plants need for adequate moisture, nutrition, and health. Therefore we try to satisfy each of these needs with using as little resources as possible. For example, by reducing the usage of pesticides, we can enhance our environmental sustainability. Additionally, by creating an optimal environment for plant growth, we can achieve the maximum harvest and profit possible. This can further be maximized by using fewer resources, such as water, fertilizer, and pesticides which all incur costs.

3.1.2 Procedural Goals

To effectively manage the plant wellness, clear procedural goals must be established. These goals outline the specific actions and routines necessary for tasks like watering, fertilizing, and pesticide application. By defining these procedural goals, a strong foundation is laid for comprehensive plant care strategies, addressing other considerations like behavioural outputs.

When contemplating irrigation as a procedural goal in an agricultural HTN planning domain, several crucial factors arise. Efficient irrigation strategies rely on a blend of factors, including acquiring the current moisture levels, the particular soil quality or type, the growth stages of the plants in question, monitoring for water stress, and the utilization of varied watering techniques [Bre21; DTS17; NBE+73; OOST14].

Successful fertilisation strategies depend on a precise interplay of numerous factors. This comprises evaluating the current levels of soil nutrients, identifying the characteristics of the soil, considering the developmental stage of the crops being cultivated, being constantly aware of potential nutrient deficiencies, and utilising a wide range of fertilisation methods [BP15; ERB+20; WS87].

For effective and environmental sustainable pest management we use the principles of IPM [FAOb]. Therefore the use of pesticides is a last resort and in the case it is needed, should be used as efficiently as possible. For that we need ways to implement pest monitoring and ways to apply only the necessary amount of pesticides if its needed.

By delving into each of these elements in the following categories, we can establish a well-rounded approach to achieving optimal plant wellness through proper hydration, nutrition and pest management.

3.1.3 Unexpected Events

For starters we need to think about the factors that are inherently uncertain and therefore more difficult to manage. There are several factors to consider during the knowledge acquisition phase for uncertainty, such as market fluctuations and natural disasters. However, this thesis will only examine the weather-related sources of uncertainty, specifically rainfall. Rainfall affects the current soil moisture which, if ignored, may result in waterlogging. Waterlogging describes the saturation of the soil with water, which can for example lead to decreased oxygen supply for plants and seriously hinder their growth and development [PCB+08]. Therefore, we need to take this uncertainty into account in our domain model to maximise plant wellness.

3.1.4 Information Services

Next we will talk about the Information Services, we will use for our model. Information services are knowledgeable behaviours that are developed through the collection, management, and application of logic to data that could be distributed across various locations [GA16].

Because weather, as previously mentioned, is inherently uncertain, we need to collect data and then make decisions based on that data, so we estimate how much rainfall will cause what kind of moisture and how that will affect our irrigation management. When examining the irrigation aspect of our model, it is essential to forecast the required amount of water for our plants in the upcoming days. Various elements have an impact on the available moisture for our vegetation, for example how much water stays in the soil from rainfall and how much we loose through evapotranspiration. For this type of information, we utilise the Deutscher Wetterdienst [1] [2] as an application service, as it provides the anticipated precipitation, evapotranspiration and soil moisture retention rates for various soil types. However we will only use the anticipated precipitation and evapotranspiration but use our own simplified versions to calculate soil moisture retention rates. The reason is that we otherwise would need way more information about our field. For example the exact locations of different soil types with various depths, location of hills or slopes, daily temperature, wind speed, etc. [Pen48]. Therefore we use our own simplified version to predict the soil water retention which can be seen in Definition 3.2.1. We then make prognosis from the information delivered and calculated about the expected moisture available in the coming days.

[1] Precipitation: https://www.dwd.de/DE/leistungen/niederschlag24wt/niederschlag24wt.html#buehneTop
[2] Soil Moisture Information: https://www.dwd.de/DE/leistungen/bodenfeuchte/bodenfeuchte.html#buehneTop

3.1.5 Application Services

Application services control the deliberate actions of software installed on computers situated within ubiquitous computing environments [GA16]. Application systems within our domain can take many forms, such as software for managing the various valves within a drip irrigation system, GPS tracking for VRT sprayers/sprinklers or the software required to read sensor values.

We will not go into specifics pertaining the software for our variable rate technologies or managing various valves. However, we now present the different software solutions we use as potential application services, which provide the information needed for other categories like our information services. We will not actually execute them inside our planner but use sample values from these sources.

Obtaining information about the current soil moisture is crucial for achieving optimal plant wellness. One potential approach for acquiring the requisite information is to use a variety of sensors. These can range from using Neutron Probes or to measure the soil dielectric permittivity through methods like Time-Domain Reflectometry or Frequency Domain Reflectometry [CE12]. The issue with sensors is that they are only precise within their immediate vicinity, despite their accuracy [Rap14]. Since we work with large fields in agriculture, outfitting them with enough sensors to get correct data is way too expensive in terms of acquiring and maintaining the sensors, simulation models can aid in comprehending and investigating the connection between the demand for crop water and the availability of soil water [CEK23]. This is accomplished by forecasting the soil water content in the root zone based on weather predictions. This offers visibility into the forthcoming changes in water availability and can be utilized by us in decision-making when the next irrigation event for a particular field is required. There are a multitude of different models which can achieve this, however for our work we choose the model AMBAV [Löp83] as an example, since its data describes the soil moisture in Germany. The AMBAV model computes both potential and actual evapotranspiration, as well as the soil water balance, for various crop types [FL07]. This is accomplished through simulating the water balance in the crop-soil system on an hourly basis using the Penman-Monteith equation. The Penman-Monteith equation is a comprehensive model used to calculate potential evapotranspiration, taking into account factors like sunshine, temperature, humidity and wind speed [APRS+98]. The mechanistic model based on the Richards equation is used to simulate soil water dynamics.

Next, we shall discuss the DRIS fertilization software, which we presume deals with identifying and computing the amount of absent nutrients.

The Diagnosis and Recommendation Integrated System (DRIS), a commonly utilized plant analysis tool, assesses the ratios of analyzed element concentrations to determine which nutrients are likely to be lacking [Bev+91; TL+84]. It establishes values that assist in identifying such deficiencies. The aim of the DRIS is to assess plant nutrition without influence from plant type, age, or the position of its leaves [Sum77; Sum79]. Since our focus lies in the irrigation process, we will not go into more details here, since we will not use precise values for nutrition values. However the domain model could be expanded with this system to incorporate the values to identify and correct different deficiencies.

3.1.6 Human Services

We try to minimise human intervention as much as possible, so we would only use human intervention if a system cannot repair itself or needs maintenance in a way that it cannot do itself. However, we do not distinguish between an autonomous fix and a human fix, we just assume that if we get to a point where we need to fix something, it will get fixed.

3.1.7 Robot and Device Services

Since we try to minimise human interventions, we need to make use of other actors in our domain. For example, for our procedural goal of Irrigation, we use the following irrigation techniques, which we consider as device services.

Our first irrigation method of choice is the sprinkler irrigation. Sprinkler irrigation is a versatile method that mimics natural rainfall by dispersing water through a network of pipes and nozzles. Water is evenly sprayed over the crop area, forming a fine mist that gently falls onto the plants below. This method is commonly employed in a variety of crops, from field agriculture to landscaping [BPKH88]. It is particularly beneficial for cooling crops in hot climates and preventing frost damage in cold regions. Sprinkler systems are relatively easy to install, and they can cover large areas efficiently [Pai69].

Drip irrigation, also referred to as trickle irrigation, involves slowly delivering water onto the soil at extremely low rates, usually between 2 and 20 litres per hour [BPKH88]. A series of slim plastic pipes fitted with small outlets, known as emitters or drippers, accomplish this. The water is carefully dispensed near the plants, guaranteeing that only the area where the roots are situated receives moisture. This is in contrast to surface and sprinkler irrigation, which typically saturate the entire soil profile. Drip irrigation requires more frequent water applications, usually every 1 to 3 days compared to other irrigation techniques. This frequent application schedule helps maintain a consistently high level of moisture in the soil, thus creating an ideal environment for plant growth and flourishing.

We use these two irrigation techniques separately, meaning we either have a sprinkler or drip irrigation installed. This choice is defined at the beginning and does not change while running our planner.

For fertilization, we will not differentiate between different types of fertiliser application or the different types of fertilizer. The reason we differentiate the type of application for watering is that we can combine watering and fertilising, if we have a drip irrigation system installed. This process is called Fertigation and it allows us to supply essential nutrients simultaneously with the irrigation process [HL95]. In order to ensure the plant receives the necessary nutrients, precise control over the amount of water it receives is required. Therefore, a precise irrigation system such as drip irrigation is necessary for proper function. The concentration of nutrients provided can be regulated by adjusting the concentration of the irrigation water.

In the case we only need to fertilize or have a sprinkler installed, we will use a Variable Rate Technology (VRT), without specifying which concrete technology is used. Since we are using a very simplified view of nutrition and pest status, it does not matter how we apply it. We introduce the following technologies only to demonstrate that there are multiple resource-efficient applications, such as various methods for applying fertilizer through variable rate fertilizing.

Variable Rate (VR) spreaders allow farmers to tailor fertiliser application to specific field conditions and crop requirements [CZC+14]. VR spreaders use advanced technology to adjust the rate of fertiliser application as the machine travels across the field. They are equipped with GPS and software that can produce prescription maps for precise application. These maps take into account factors such as soil type, nutrient levels and historical yield data to determine the ideal fertiliser rate for each location within the field. This means that different areas of a field can receive different amounts of fertiliser, optimising the use of resources and increasing crop yields.

Another option is the usage of drones to apply fertilizer, similar to map-based VRT like VR spreaders with prescription maps [FAO22]. However as the recent report 'The State of Food and Agriculture 2022' of the FAO indicates, this technology is not yet fully established, with the commercialization only just beginning for small farms.

For Pest management, as with Fertilizing, we will assume the usage of some kind of variable rate technology. The variable rate technology available is similar to the ones used in fertilizing. For example VR sprayers use technology to adjust the pesticide application rate throughout the field [ERP+13], similar to their fertilizer-spreading counterparts. As with the use of drones for fertilising, the same problems exist for the use of drones for pesticides.

3.1.8 Temporal and Spatial Properties

A field can be viewed as a single entity, but this would be inefficient as there are many factors, like we explained in the background chapter, from the soil type and quality that are not homogeneous but do affect our plants well being. To counter that we work with cells, the exact size can be adjusted but for now we will use a cell which encloses one plant with a depth of 10 cm. Each cell will have its moisture, ideal moisture, nutrition amount,ideal nutrition and if pest are active saved in some form which we define later in Section 3.3.1. For temporal properties, we could have a schedule for the different Behavioural outputs, so we for example do not send multiple sprayers out at once. However since, as already mentioned, we do not go into much detail pertaining the implementation of the pest and fertilizing part, we do not implement a scheduler. For the irrigation part, there is no explicit need for a schedule therefore we omit the exact details how all the Behavioural outputs work together in the case we have a field with multiple cells.

3.1.9 Action Contingencies

As we work with different forms of autonomous actors, actions can be blocked by external interference. For example, in our drip irrigation systems, dirt could block the various valves and prevent us from delivering the calculated amount of moisture to our plants. Another example would be that our storage of fertilizer is empty and we therefore can not apply them if needed.

Category	Resulting Knowledge
Declarative Goals	Maximum crop yield, reduce resource usage like pesticides, fertilizer, water and other chemicals
Procedural Goals	Irrigation, Fertilization and Pest management
Information Services	Estimations about expected soil moisture for the coming days
Application Services	Deutscher Wetterdienst prognosis, DRIS, farm management systems, crop monitoring, disease detection, variable rate fertilizing and pest control
Human Actions	Maintenance, repair
Robot Actions	Drones for crop monitoring, weed detection, autonomous vehicles with VRT equipped
Device Operation	Sprinkler and drip irrigation, VR spreaders, VR sprayers
Temporal Properties	Automatic scheduling for the different machinery
Spatial Properties	Divided Field into cells which all have different needs for the amount of resources/help needed
Unexpected Events	Rainfall
Action Contingencies	Valves blocked, Fertilizer empty, machinery in need of repair
Partial Observability	Amount of rainfall, expected soil moisture

Table 3.1: Acquired Knowledge for the Agricultural HTN planning domain

3.1.10 Partial Observability

Partial observability describes the incompleteness and imperfection of data on environmental conditions [GA16]. Since many of our actions depend on making predictions based on calculations, we cannot be sure that the predicted state will actually be achieved. For example we work with probability distributions for the amount of rainfall we can expect in the coming days and calculate the needed moisture now. Furthermore, as already discussed, soil moisture sensors are not really accurate outside of their immediate vicinity, so the German Weather Service uses equations to calculate the expected soil moisture. But still, for our model and in general, these calculations are accurate enough, so we will not go into this further.

3.2 Analysing the Domain Requirements

To implement our model, we must identify credible sources that provide the necessary empirical data for our planner. This involves two crucial elements, first is acquiring a source for ideal moisture levels. Secondly, acquiring an equation that enables us to calculate the cost of operators. This equation should take into account the available moisture be it from current soil moisture or moisture supplied by weather, and the optimal moisture we would like to achieve.

For the first crucial element we will use the Deutscher Wetterdienst [Wet], the used values can be seen in Table 3.2. They describe the wellness state of the plant depending on the nFK. For example if we reach over 100% nFK, that means the soil is over saturated and the plant can suffer under lack

% nFK	Plant stage
<30	Plant is suffering under water stress, high chance for reduced yield
30 - 50	enough water for plant development
50 - 80	optimal moisture level
80 - 100	Beginning of oversupply of water, potential lack of oxygen
> 100	Oversupply and lack of oxygen

Table 3.2: Classes of soil Moisture in nFK, values from [Wet]

of oxygen. These values are general guidelines and do not target a specific plant. However with these values we can get a general grasp how the different thresholds could look like. Furthermore since we know this value can change depending on the plants type and age, we should implement it in a way that makes it part of the world state [DTS17].

For the second element, we decided to implement our own simple version of an equation that calculates the amount of missing nFK as the cost of our operators.

Definition 3.2.1 (Equation for missing nFK)
The cost of the missing % nFK for a cell c with available moisture a is defined as:

- $a = m + ((\sum_{j=0}^{5} w_j) - j * e) + i$
- $cost(c) = \begin{cases} a - om & \text{if } a < om \\ -a - 5 & \text{if } om < a < (100 - om) \\ -\infty & else \end{cases}$

where we used:

- *moisture m*
- *optimal moisture om*
- *the nFK added by irrigation i*
- *the amount of nFK supplied by weather w_j on day j*
- *the static evapotranspiration e*

A graph of cost(c) for optimal moisture 70% nFK can be seen in Figure 3.3.

This equation provides an estimate of the missing nFK in the soil, as well as the distance from ideal moisture. The weather forecast also plays a role, if there is no rainfall predicted in the upcoming days, then the sum of $(\sum_{j=0}^{5} w_j) - j * e$ will be negative. Consequently, nFK is reduced and watering is necessary if the current moisture is below optimal levels. With this cost calculation we can compare different situations and make the correct decision. For example the situation where the current soil moisture is 50% and our optimal moisture is 70%. If we now look at the cost of adding 10% nFK to the current soil moisture through irrigation compared to keeping the 50% through waiting, the cost would be higher than waiting if we do not expect any rainfall. That is because the closer we get to the optimal moisture, the higher our cost becomes, and since it is a negative

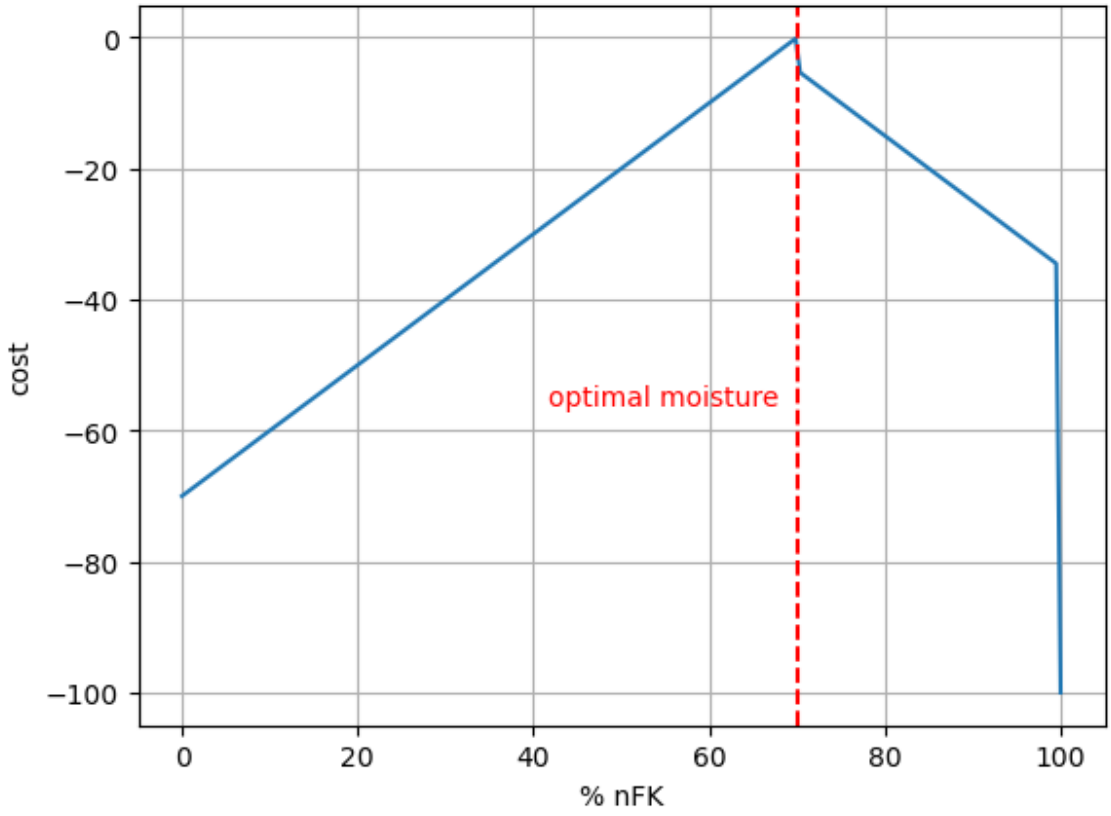

Figure 3.3: Cost distribution depending on available moisture

cost this means it is a better choice than a lower cost. So if we look at the example again where have to choose between 50% and 60% nFK we choose 60% nFK, since its cost is higher with our equation.

As the optimal level of moisture exists on a spectrum, exceeding this level slightly is preferable to having low moisture. Thus, we only assign a cost of watering of $-\infty$ if watering would lead to exceeding 100% nFK and causing damage to the plants. However, we consider going above the optimum by a certain amount to be worse than going below it by that amount. This is because we can always water the plants later when a new plan is created by our planner in case of insufficient moisture. In the event of overwatering, we must wait for the water to either runoff or be used up through evapotranspiration.

3.3 Modelling the Domain

In this section, we detail how the results from our knowledge acquisition are applied to acquire a HTN model. Firstly we start by defining the possible states that describe our world state. Afterwards we go over the different operators that affect these predicates and can change their values. Next, we

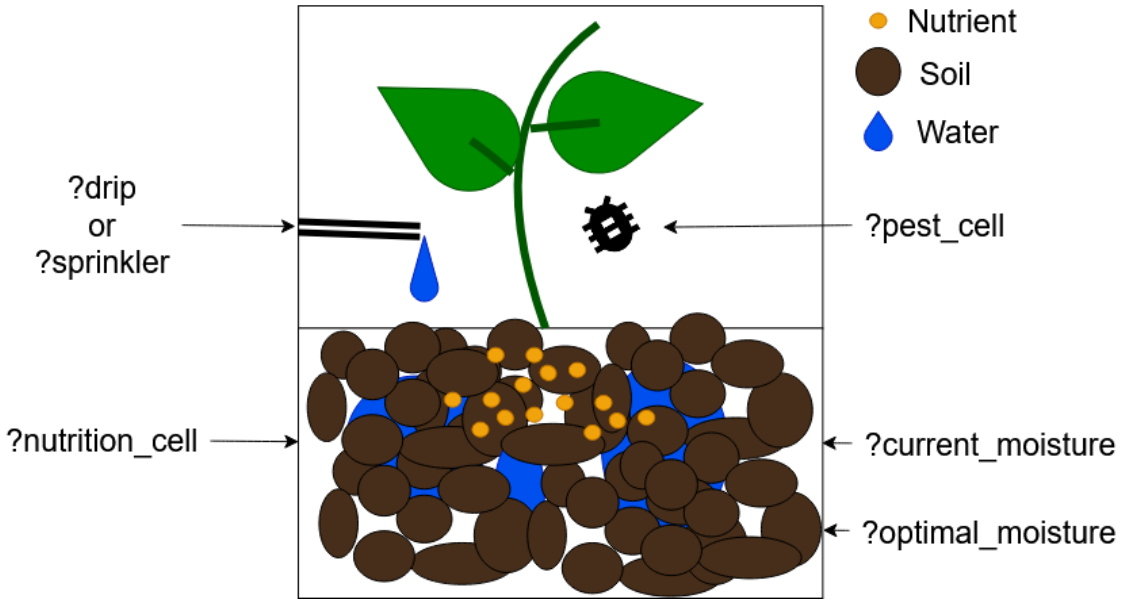

Figure 3.4: Visualization of a cell in our domain

have a look at the sequence in which the previously defined operators will be executed. Finally, we use the possible situations in which we may find ourselves to help us achieve the remaining Model.

3.3.1 Predicates

Beginning with our field, we need to decide how to model its characteristics in a way that makes planning efficient and precise. From our knowledge acquisition phase, we can deduce various information we need to acquire, for example the current moisture. Therefore we define a cell with the following Information in the form of predicates that describe our world state:

- (cell-type ?cell ?irrigation-type)
- (moisture-cell ?cell ?moisture)
- (optimal-moisture-cell ?cell ?optimal-moisture)
- (nutrition-cell ?cell ?nutrition-status)
- (pest-cell ?cell ?pest-status)

As we described in the category Device Services, we use two types of irrigation techniques. However since we do not have them both installed at once in a cell, we need to create a predicate that saves if we have a drip or sprinkler installed. For that we use the predicate cell-type, which describes what kind of irrigation system ?irrigation-type is installed in a cell ?cell. For example, if we have a drip irrigation d1 installed in a cell c1 we would have (cell-type c1 d1), the same can be done for sprinkler.

The current moisture is another crucial element, like we explained in our procedural goals. Therefore we use (moisture-cell ?cell ?moisture), where ?moisture describes the moisture in %nFK for the cell ?cell. Another important factor that we described in our procedural goals is the optimal moisture ?optimal-moisture. The values for the optimal moisture range are acquired as explained in Section 3.2 and are saved for each cell ?cell.

The last two goals, from our procedural goals, describe a need for a way to track the current status of our plants health and nutrition. For that, we have nutrition-cell and pest-cell to track the current status, which we simplify, meaning we only have two states for each. If the status is 0, the plant needs tending for the specific need, if it is 1 the plant is content. For example, (pest-cell c1 0) would describe that we have a pest problem in cell c1.

An illustration, how a cell could look like can can be seen in Figure 3.4. It shows the just discussed predicates, like ?drip or ?current-moisture, in combination with the rest of the discussed predicates.

We continue with our procedural goal of irrigation. Their predicates can be defined as:

- (cell ?cell-id)
- (drip ?drip-id)
- (sprinkler ?sprinkler-id)
- (cell-type ?cell ?irrigation-id)
- (irrigation-step ?irrigation-type ?step-size)
- (drip-functional ?drip ?functional)
- (storage ?status)

The goal of irrigation employs various components from the previously defined categories. The key factors are sprinkler and drip irrigation, which impact the moisture-cell predicate. Since we can have multiple cells, we need a way to differentiate them and their equipment, that is why we use (sprinkler ?sprinkler-id), (drip ?drip-id) and (cell ?cell-id). Furthermore, we must distinguish between the accuracy of the method of irrigation employed. Drip irrigation permits one to finely regulate the quantity of moisture, that is desired to be added effectively. Therefore we add a new Predicate, named irrigation-step, to specify the varying levels of moisture that can be delivered to the cell. For example, drip irrigation has the predicate (irrigation-step ?drip 5) meaning we can add moisture in steps of 5% to the nFK.

As described in the Action Contingencies category, we also need to watch our for any malfunction of our equipment. For that, we also add the Predicate drip-functional, that describes if our various valves have a blockade. For example, (drip-functional d1 1) would describe a functional drip. In the Action Contingencies category, we identified another scenario where our fertilizer storage may be empty. To account for this possibility, we include (storage ?status) to verify if we have enough or if it is empty.

3.3.2 Operators

Now that we have established our Predicates, we can proceed to define the operators which can alter their respective values. To achieve this, we shall assess our procedural goals once again and directly translate each of them into an operator.

- (!activate-irrigation ?cell)
- (!apply-pesticide ?cell)
- (!apply-fertilizer ?cell)
- (!apply-fertigation ?cell)
- (!wait ?cell)

For irrigation, regardless of irrigation technique, we use (!activate-irrigation c1) which increases the current-moisture of cell c1 by a specific amount. The amount, with which we can increase the current-moisture depends on the step size in (irrigation-step ?irrigation-type ?step-size) as explained earlier. In the case we have (cell-type c1 drip) and (nutrition-cell c1 0), we can use (!apply-fertigation c1) which increases the moisture and restores the nutrition. Since watering depends on the uncertain factor of weather in combination with the current moisture and optimal moisture, we also need the possibility to skip watering by waiting. That is why we also need the operator (!wait ?cell), which does not change the current moisture. To change the values of the predicates (nutrition-cell ?cell ?nutrition-status) and (pest-cell ?cell ?pest-status) from 0 to 1, we use (!apply-pesticide ?cell) and (!apply-fertilizer ?cell). The only thing missing for now, are the Operators that deal with our Action Contingencies.

- (!repair ?cell ?drip)
- (!refill ?cell)

If our drip irrigation is not functional, the (!repair c1 d1) operator changes the value of (drip-functional d1 0) to 1. The same can be said about our storage, if is empty, (!refill ?cell) changes (storage 0) to 1.

In the case, where our cell is in need of assistance pertaining each procedural goal, we need to decide a order of tasks. For that, we look at each pair and decide which one should come first, beginning with irrigation and fertilization. In this case we start by watering the plant before we use our fertilizer. The reason is, that watering affects how much fertilizer stays in the soil [BP09]. This is especially true for nitrogen, which is especially susceptible to leaching loss if combined with watering. Contrary, fertilizing does not affect how much moisture stays in the soil, therefore it is safe to apply afterwards [ZNY+19][BP09].

Next, we will examine the relationship between watering and pesticide application. Considering that pesticide leaching into the groundwater is already a concern in systems without combining it with irrigation, we have decided to prioritise irrigation before pesticide application [Flu96].

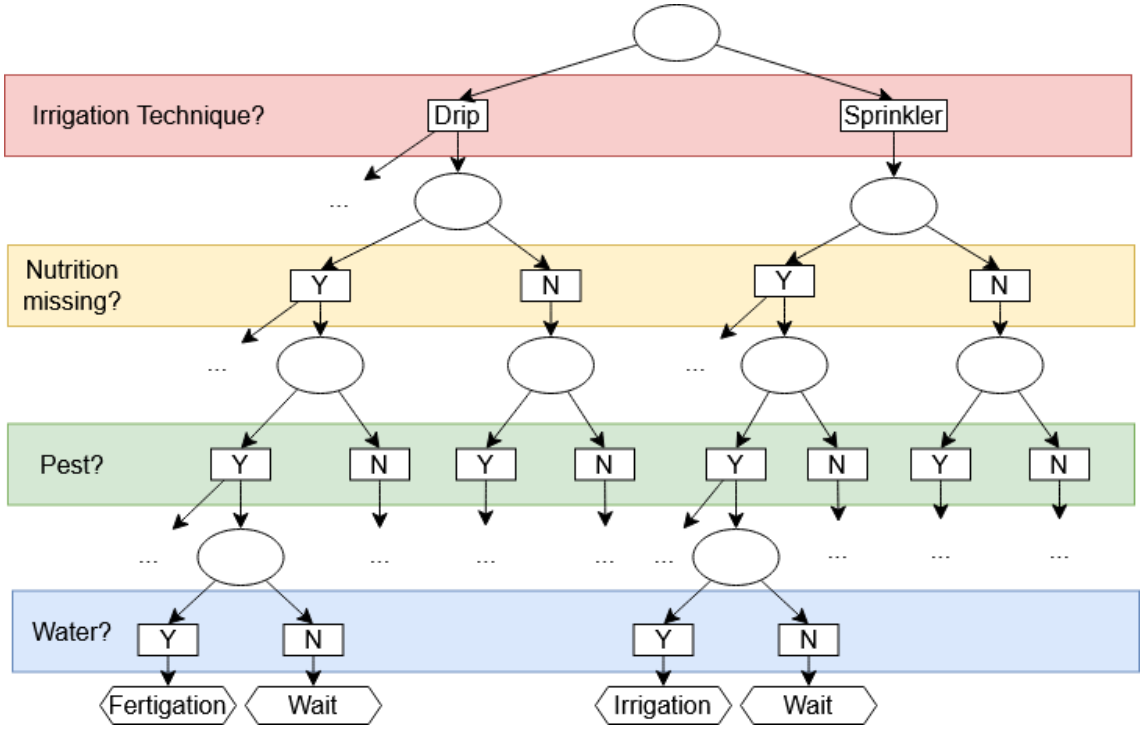

Figure 3.5: Simplified representation of our domain model

3.3.3 Methods and Compound Tasks

Now that we have established our operators and predicates we need to define the situations in which they can occur. First of all, since we want to be able to use irrigation, fertilization and pest management all together, we need a domain structure that allows this. Therefore, we decided to use a kind of binary tree structure for our domain. A simplified version to represent the structure can be seen in Figure 3.5 and a more elaborate can be found in Appendix B.

At the start, we decompose the main compound task that stands at the top, by checking if the cell is watered by a drip or sprinkler. We verify this by looking at the predicate (cell-type ?cell ?irrigation), where ?irrigation is either a drip or sprinkler. In the case that we have a drip irrigation installed, we decompose the main compound task into the compound tasks maintenance and check-nutrition-drip. Otherwise we only decompose it into check-nutrition-sprinkler, since we assume the sprinkler, in the case we want to use it, is functional.

Maintenance checks if the drip is functional, this is getting tested by two methods that check the predicate (drip-functional ?d ?status). In the case status is 1, the resulting task network is empty, if it is 0 we call the operator (!repair ?cell ?d).

Both compound tasks, check-nutrition-drip and check-nutrition-sprinkler, are getting decomposed depending on their nutrition status. In the case that the nutrition-status of predicate (nutrition-cell ?cell ?nutrition-status) is 1, both get decomposed into check-pest-default. If nutrition-status is 0, they get decomposed into the compound task check-storage and check-pest-drip or check-pest-sprinkler

depending on the irrigation type. The reason both get decomposed into check-pest-default, in the case of status = 1, is because the reason why we even differentiate them in the first place is missing, meaning the possibility to use fertigation.

Check-storage has the same structure as maintenance, meaning we check the predicate (storage ?status) and depending on the value, call (!refill ?cell) or do nothing.

Next up is checking the pest status of the cell in question. This is achieved by decomposing one of the specific compound tasks check-pest-drip/sprinkler/default, depending on the value of (pest-cell ?cell ?pest-status). As before ?pest-status is either 1 or 0 and depending on the value, we decompose into the compound task irrigation/fertigation and number a of primitive tasks depending on which check-pest we decompose. For example, check-pest-drip decomposes into the compound task Fertigation and depending on ?pest-status, the primitive task (!apply-pesticide ?cell). Check-pest-sprinkler on the contrary, decomposes into the compound task Irrigation, primitive task (!apply-fertilizer ?cell) and again depending on the ?pest-status, primitive task (!apply-pesticide ?cell). Check-pest-default, behaves like check-pest-drip with the contrast being that the compound task is not Fertigation but Irrigation.

Finally, we need to verify the extent to which the plants moisture requirements have been met. However since this is depending on two cost-variable operators,(!activate-irrigation/fertigation ?cell) and (!wait ?cell), the methods that decide if we should water or wait need a different form. That is why, we go over this part in the next section in more detail.

3.4 Incorporating Risk

As described before, our cost variable operators are in the lowest possible leaf. This is illustrated in Figure 3.6, where we have progressed through each layer, as outlined in Section 3.3.3 At this juncture, we must determine whether to irrigate or to wait.

To incorporate risk awareness, we need to decide between the Now and Wait method depending on the EU of their respective task networks. This is achieved by calculating the EU of watering and waiting beforehand. For that we use the operators (!calc-water-cost ?cell) and (!calc-wait-cost ?cell). They both need to calculate the EU depending on the cost associated with each operator, if we would choose this particular operator. The results are then saved in the Predicates (irrigation-cost ?cell ?irrigation-technique ?EU) and (wait-cost ?cell ?EU). Afterwards we decompose the compound task irrigation and select the appropriate method based on which has a higher EU. The reason why we designed our model in this way and why we need to pre calculate the EU, is explained in the next chapter.

What we do not incorporate into our model, but still use for the calculations are the weather forecasts. We chose not to incorporate them into the model, since they have to be acquired through some form of external calls.

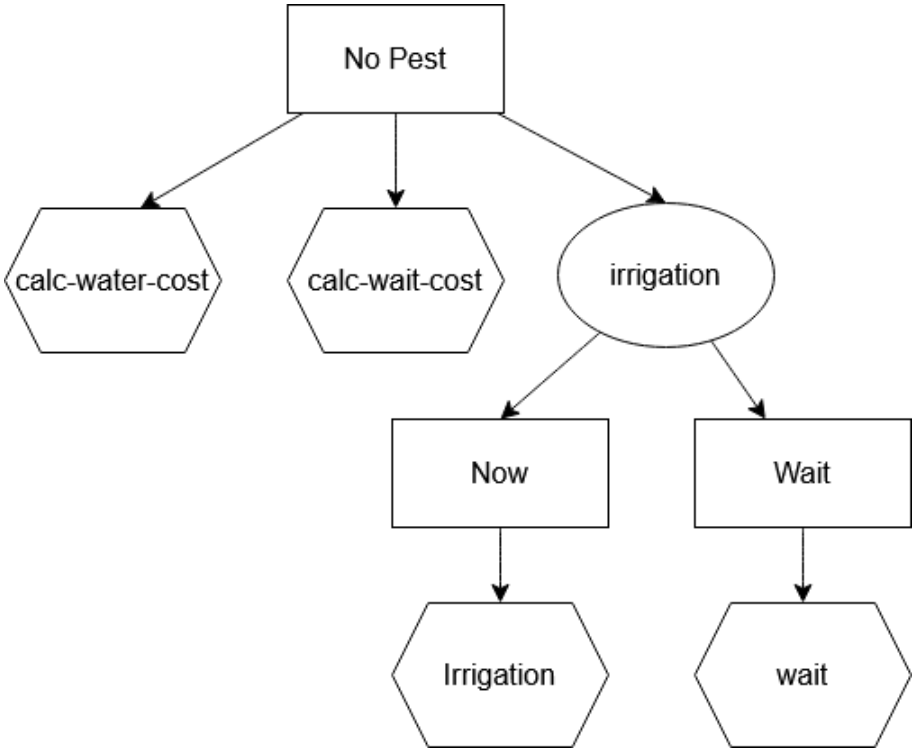

Figure 3.6: Incorporating Uncertainty in our domain model

4 Implementation

4.1 Choosing the Right Planner

To implement the domain model that we have devised, we need to select a planner that enables us to put it into effect. Since not all planners are equal and some have different additional features, this decision is crucial because it can impact the feasibility of our model or necessitate modifications for compatibility. For our model we have opted to utilise Java Simple Hierachical Ordered Planner 2 (JSHOP2) which is a domain-independent planning system [Ilg06]. JSHOP2 has the ability to recognize the present state of the world in every step, hence it follows a state-based planning methodology. Furthermore it possesses substantial expressive power, enabling the execution of external program calls within the preconditions of operators and methods. We use this possibility for calling external functionalities, to calculate the costs of our cost-variable operators. Moreover, JSHOP2 offers the capability to craft highly efficient, domain-specific planning algorithms. The efficiency can be seen later in the Evaluation Chapter, where we test the performance of the planner handling our domain for large problems. Additionally, JSHOP2 seamlessly incorporates multiple features from the Planning Domain Definition Language (PDDL), such as support for quantifiers and conditional effects in methods and operators. These features enhance its adaptability and the modeling and resolution of complex planning problems.

4.2 Implementing our Domain Model

Before we implement our Domain, we firstly define the Syntax of JSHOP2.

- **Symbols** can be one of the following:
  - Variable symbols are all symbols that start with a ? for example ?cell.
  - Primitive task symbols are all symbols that start with a ! for example !activate-irrigation
  - Compound task, predicate and constant symbols are all symbols that start with a letter or an underline
  - Function symbols are all valid Java identifier
- **Terms** can be described as one of the following
  - Variable symbol
  - Constant symbol
  - Call term, which is of the form (call f t_1 ... t_n) where f is a function symbol for calling external functionalities or a built-in function.

- **Predicates** are defined as (p t_1 ... t_n) where p is the predicate symbol and t_i is a term
- **Operators** are defined as (:operator h P D A [c]) where:
  - h is the operators head and is a primitive task atom
  - P is the logical precondition of the operator
  - D is the delete list, which describes the predicates from h or P to be deleted
  - A is the add list, which is similar to D but adds predicates
  - c is the cost of the operator and is optional, the default value is 1.
- **Methods** are defines as (method h L T) where:
  - h is the methods head, which is the compound task the method decomposes
  - L is a logical precondition
  - T is the task list that the method decomposes h into.

Now that we have defined the Syntax, we continue with the Requirements to run the Planner. JSHOP2 requires two files to be able to run and create a plan.

- The **planning domain** is defined as (defdomain domain-name ($d_1 d_2 ... d_n$))
- The **planning problem** is defined as (defproblem problem-name domain-name ([$a_1 a_2 ... a_n$]) T), where:
  - problem-name and domain-name are symbols, most of the time they are just the name of the files
  - a_i are the predicates that describe the initial state
  - T is a task list that acts as the goal

Now that the necessary information for implementing our domain model has been established, we can proceed with the concrete implementation. For that, we only look at one example now, since we already explained the structure of our model and the syntax of JSHOP2. The entirety of the implemented domain model can be found in Appendix B.

For now, we present how an operator, method and compound task from our domain model is transformed into JSHOP2 code. In Listing 4.1 we see the operator (!repair ?cell ?drip), that changes the predicate (drip-functional ?drip ?status). As explained earlier, JSHOP2 operators are defined as (:operator h P D A [c]). Therefore (!repair ?cell ?drip) describes the head h of the operator by which it can be called from some kind of method. The initial (functioning drip ?drip 0) is a precondition for the operator, indicating that the drip must be non-functional to carry out repairs. The second (drip-functional ?drip 0) is included in the list of operators to be deleted, which means that if the operator is called and its precondition is satisfied, it will delete this predicate. After deleting the predicates in the delete list, it adds the elements from the third entry, in this case (drip-functional ?drip 1), as a predicate to the world state.

Listing 4.1 Small part of our Domain Model as an example

```
(:operator (!repair ?cell ?drip)
       ((drip-functional ?drip 0))
       ((drip-functional ?drip 0))
       ((drip-functional ?drip 1))
       -1
       )
(:method
(maintenance ?cell ?d)
((cell ?cell)(cell-type ?cell ?d)(drip ?d)(drip-functional ?d 0))
((!repair ?cell ?d))
)
```

Methods are defined as (method h L T), meaning in this case the head of the method is (maintenance ?cell ?d), which describes the compound task this method decomposes. L is the precondition of this method and in this case, we check if the cell has a drip installed and if it is non-functional. If the preconditions are met, we decompose (maintenance ?cell ?d) into the operator (!repair ?cell ?d).

The reason why we check (drip-functional ?drip 0) in the method and in the operator, is because to decide if we need to repair, we need to check it in the method first. However for the operator to be able to change the predicate, it must be part of its precondition, as was described in the definition of Operators before. Therefore we need it in both cases as a precondition, first to find out if we need to repair through the method and then to be able to change the predicate.

4.3 Uncertainty in JSHOP2

In previous chapters, we have outlined the necessity of employing a cost-variable operator for watering operations to enable risk awareness in instances of uncertainty. For this, additional functionalities are required, as we are unable to integrate these calculations into our domain model. JSHOP2 has the capability to execute external functions via its Calculate Interface, which we must implement in order to compute the cost of our cost-variable operators.

One issue is that when there are two applicable methods, JSHOP2 always selects the first one, which presents issues for implementing risk awareness. This is because when both methods contain cost variable operators somewhere in their respective task network, we would like to calculate the expected utility (EU) for each methods task network and select the one with the higher EU.

One option would be to modify JSHOP2 to enable comparison with EU task networks as a whole. Nevertheless, we chose to develop our model in a way that integrates risk awareness without requiring a major overhaul. Since the rest of our methods are deterministic, the only method affected by the cost variable operators would be the methods Now and Wait. Therefore, the decision to turn the irrigation system on or keep it off, is located at the lowest level of our Domain model. This is done to avoid factoring in the costs from other operators in the plan, when we calculate the expected utility of these two cost variable operators.

Listing 4.2 Precalculation of the EU for both waiting and watering

```
(:operator (!calc-water-cost ?cell)
       ((irrigation-cost ?cell ?irrigation ?old-cost)(cell-type ?cell ?irrigation)(moisture-
cell ?cell ?moist)(optimal-moisture-cell ?cell ?optmoist)(irrigation-step ?irrigation ?water))
       ((irrigation-cost ?cell ?irrigation ?old-cost))
       ((irrigation-cost ?cell ?irrigation (call CalculatePlantWellness ?optmoist ?moist ?
water )))
       0
       )

(:operator (!calc-wait-cost ?cell)
       ((wait-cost ?cell ?old-cost)(cell-type ?cell ?irrigation)(moisture-cell ?cell ?moist)(
optimal-moisture-cell ?cell ?optmoist))
       ((wait-cost ?cell ?old-cost))
       ((wait-cost ?cell (call CalculatePlantWellness ?optmoist ?moist 0 )))
       0
       )
```

Now, let us look at how exactly our program implements risk awareness for the decision between !activate-irrigation/fertigation and !wait. For that we use the possibility to call external functionalities with (call CalculatePlantWellness ?optmoist ?moist ?cost), which calculates the EU depending on the ?cost. As was described earlier, we pre calculate the EU by calling the two operators (!calc-water-cost ?cell) and (!calc-wait-cost ?cell), which can be seen in Listing 4.2. What we can see in the listing is, that we delete the old value ?old-cost with (irrigation-cost ?cell ?irrigation ?old-cost) and add the calculated value from (call CalculatePlantWellness ?optmoist ?moist (?water or 0)) as the new cost. In this case we called the external function with:

- the optimal moisture ?optmoist from (optimal-moisture-cell ?cell ?optmoist)
- the current moisture ?moist from (moisture-cell ?cell ?moist)
- the varying levels of moisture ?water that can be delivered from (irrigation-step ?irrigation ?water). In the case of waiting, since we do not add any moisture, the value is 0 instead of ?water.

What is also observable is that we gave these two operators the cost of 0. The reason is, since we only pre calculate these values because there was no easy way to implement risk awareness in JSHOP2, we decided that they should not cost anything. This way, we should have the same plan cost as we would have, if we did not have to resort to simplification to implement risk awareness.

After calculating and saving the EU for both operators, we select the appropriate method based on which has a higher EU, which can be seen in Listing 4.3. For the selection, we use the built-in function calls < and >=, which make it possible for us to compare two values as a precondition. In the case of (call >= ?water ?wait), we would check the condition if ?water >= ?wait, which would return true or false. This way, we can compare the pre calculated EU and choose the correct operator, depending which one is larger.

To obtain the EU, we use the equation from Definition 2.3.4 and compute the EU for both irrigation and waiting. The equation needs a probability and the cost of a operator.

Listing 4.3 Irrigation compound task to decide if to water or not

```
(:method (irrigation ?cell)
((irrigation-cost ?cell ?irrigation ?water)(wait-cost ?cell ?wait)(call >= ?water ?wait))
((!activate-irrigation ?cell))
)
(:method (irrigation ?cell)
((irrigation-cost ?cell ?irrigation ?water)(wait-cost ?cell ?wait)(call < ?water ?wait))
((!wait ?cell))
)
```

Listing 4.4 Weather representation in our implementation

```
double p1 = 1.3 ;                                  //equals 70%
double[] weather_1= {10.0, 0.0, 5.0, 10.0, 8.0}; //Values are in % nFK
double p2 = 1.7 ;                                  //equals 30%
double[] weather_2= {0.0, 3.0, 0.0, 0.0, 0.0} ;  //Values are in %nFK
```

The probability is presented through two different weather forecasts $weather_1$ and $weather_2$ that our program acquires. Both weather forecasts have the expected rainfall for the next 5 days, including the day of planning. They each also have a probability p_1 and p_2 for this specific prediction to occur, the implementation can be seen in Listing 4.4.

Using these forecasts, we can calculate the cost of both irrigation and waiting based on the provided predictions. To do so, we employ the equation outlined in Definition 3.2.1 to determine the missing nFK which is our domain models cost representation. These calculations provide us with the cost of the operators and combined with the probability of this cost, we can now calculate the EU for both irrigation and waiting.

What is important to note is that since we use negative numbers and care about the differences between costs, we do not use, for example 0.7, as a probability but 1.3 . With this we make the value smaller by the margin it would have been made smaller if the number was positive. Therefore the differences stay the same for our comparison between costs. This transformed probability representation can be defined as:

Definition 4.3.1 (Probability transformation for our Costs)
A probability for positive numbers p_p can be transformed to a probability for negative numbers p_n:

$p_n = 2 - p_p$

where, $0 < p_p \leq 1$ *and* $1 \leq p_n < 2$. *This leads to the same difference d, which can be shown with the following formula:*

$d = cost - cost * p$

If take an example with cost of 10 and -10 and the probability of 0.8 and 1.2, we get:

- $d = 10 - 10 * 0.8 = 2$
- $d = 10 - (-10) * 1.2 = 2$

Therefore we have the same difference, and consequently can still compare our operator costs even if they have negative costs.

5 Evaluation

In this chapter, we thoroughly evaluate our domain model. For this purpose, we employ the identical framework as we did for our knowledge acquisition process [GA16]. While for our knowledge acquisition, we used the Environment dimension, for our Evaluation we use the Interpretation dimension. The Interpretation dimension, is used to provide insights into our domain model, meaning how it works in different situations. These different situations can be divided as shown in Figure 5.1. Firstly, we will explore the *Demonstrations* sub dimension, that can be divided into *Examples* or *Scenarios*. Subsequently, we will conduct a *Quantitative Evaluation* to appraise how well our domain model functions in case we plan for a whole field, with hundreds of cells. Lastly, a *Qualitative Evaluation* will be carried out to assess the quality of our plan in terms of a specific parameter.

5.1 Demonstration

As mentioned, Demonstrations can be divided into Examples and Scenarios. Examples are an effective means of enhancing the clarity and comprehension of concepts being introduced. They can be presented as descriptive text or via a chosen syntax, where they may include excerpts from state representations, goal illustrations, domain knowledge segments, and plan samples. In the field of ubiquitous computing, examples are a preferred method for illustrating planning challenges [GA16].

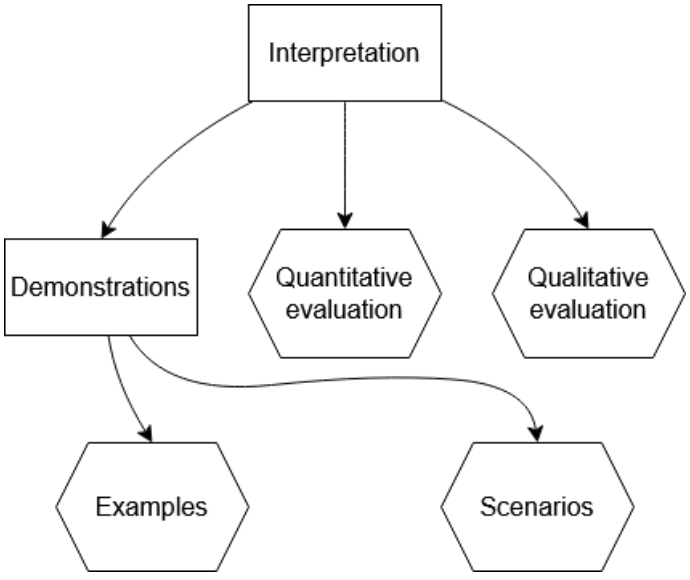

Figure 5.1: Hierarchy of Interpretation with its sub dimensions

Scenarios describe the potential situations or contexts, that a system or plan is expected to operate in. They provide a broad context for understanding how an automated planning system should behave, under different conditions. Scenarios assist in defining the problem space and the range of possible situations, that require consideration in the planning process.

5.1.1 Scenario with one cell

We begin with creating a scenario, from which two specific situations will be used as an example. Let us consider a small farm, with one field that grows one type of plant. The soil type is exactly the same for the whole field and there are no slopes or hills present in the field. The field is equipped with a functional drip irrigation system and uses sensors to acquire the current soil moisture. The optimal soil moisture, for this specific plant, at its current growth stage is 70% nFK. Pest are not present and the nutrition is on an optimal level, therefore we only look at the moisture. For this scenario, one cell represents the whole field. Lastly there is no rainfall predicted, we only work with the current moisture and the moisture we loose over the coming days. The problem definition used for the following examples, can be found in Listing C.1, which describes the explained scenario.

We begin with a straightforward example, to demonstrate how our domain model addresses the most frequent situations for one cell, where the solution to the problem is evident. For this purpose, we will test it once with arid soil and soil moisture considerably exceeding our optimal moisture level. The distinction among the two examples we will examine, lies in the line "(moisture-cell c1 X)", wherein X is the variable we will modify depending on the example presented.

For our straightforward Examples, we will use values of X=0 and X=71 for the current moisture, and the results align with expectations. When the current moisture is 0 % nFK, the planner determines that we should activate our irrigation system, while for 71 % nFK, it advises us to wait. These results happen for both risk attitudes, risk seeking and risk averse.

5.1.2 Scenario with multiple cells

The following scenario is equal to the previous one, yet this time the field possesses slight variation in soil type at every position. As a result, the optimal nFK will depend on the position for each cell, which now encompasses only a single plant. Therefore we now have as many cells, as there are plants in the field. For the sake of clarity, the following example will only look at the result of three plants, but it should be noted that the number of cells can be expanded, as demonstrated in Section 5.2. As before the problem description can be found in Listing C.2.

In this example, as opposed to before, we give each plant its custom value for current and optimal moisture. Each plant is more or less around its optimal moisture level however, since the values differ, the EU of irrigation or waiting does differ too. For cell 0, we are a little bit below its optimal level, cell 1 is slightly above and cell 2 is at its optimal level. The resulting plan chose to activate irrigation for cell 0 and cell 2 and wait for cell 1. The reason for activating the irrigation for cell 1, even if we are at the optimal moisture, is because we predict a loss of moisture each day. Therefore, to counter this loss, we water cell 1 with the smallest amount we can supply.

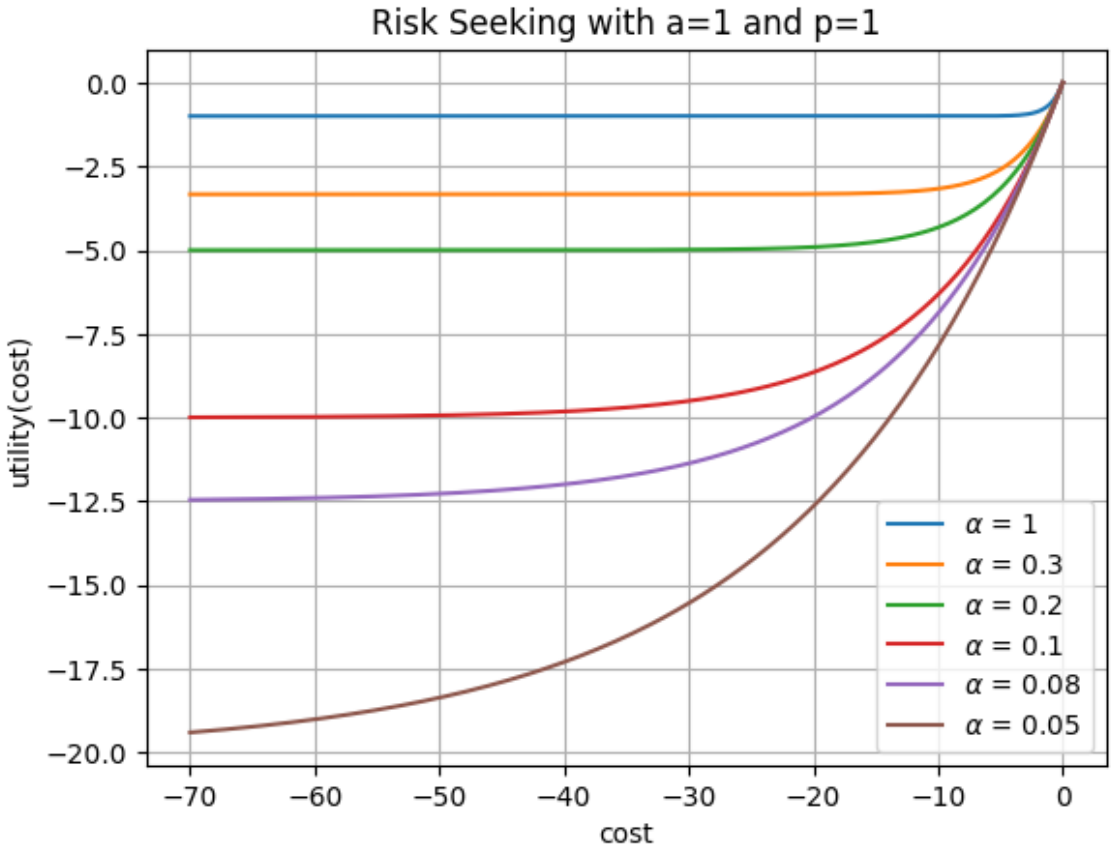

Figure 5.2: Risk seeking utility functions

5.1.3 Scenario with rainfall

Before we look at a concrete example, let us first take a look at how our cost graph, seen in Figure 3.3, is transformed via the utility function from Definition 2.3.3. The resulting graph for risk seeking can be seen in Figure 5.2 and the one for risk averse can be found in Figure 5.3 and Figure 5.4. What we can see in Figure 5.2, is that a risk seeking attitude values low nFK, meaning high cost, the same after a certain point. This point depends on the value for α and gets pushed pack further the lower α becomes. For example we can see that for $\alpha = 1$, the utility stays more or less the same after a cost of -15% nFK. What is missing in these figures is the utility for overwatering, which follows the same curve and therefore we put it in an extra figure, which we will talk about later.

Next we examine the graphs for a risk averse attitude, for that we only look at the cases where α=1 in Figure 5.3 or α=0.05 in Figure 5.4. The whole spectrum can be found in Appendix C. Compared to the risk seeking, we can observe that a risk averse attitude values low % nFK as way worse than being close to the optimum. If we look at the cost axis, the costs explode in their negative value. This is especially evident in the case for α=1, which would be a very extreme risk averse attitude.

Next we examine how the probability affects these utility graphs. However, for that we only examine the graphs for α=0.1, since it represents a moderate version of the specific attitude. Beginning with risk seeking, which we can see in Figure 5.5.

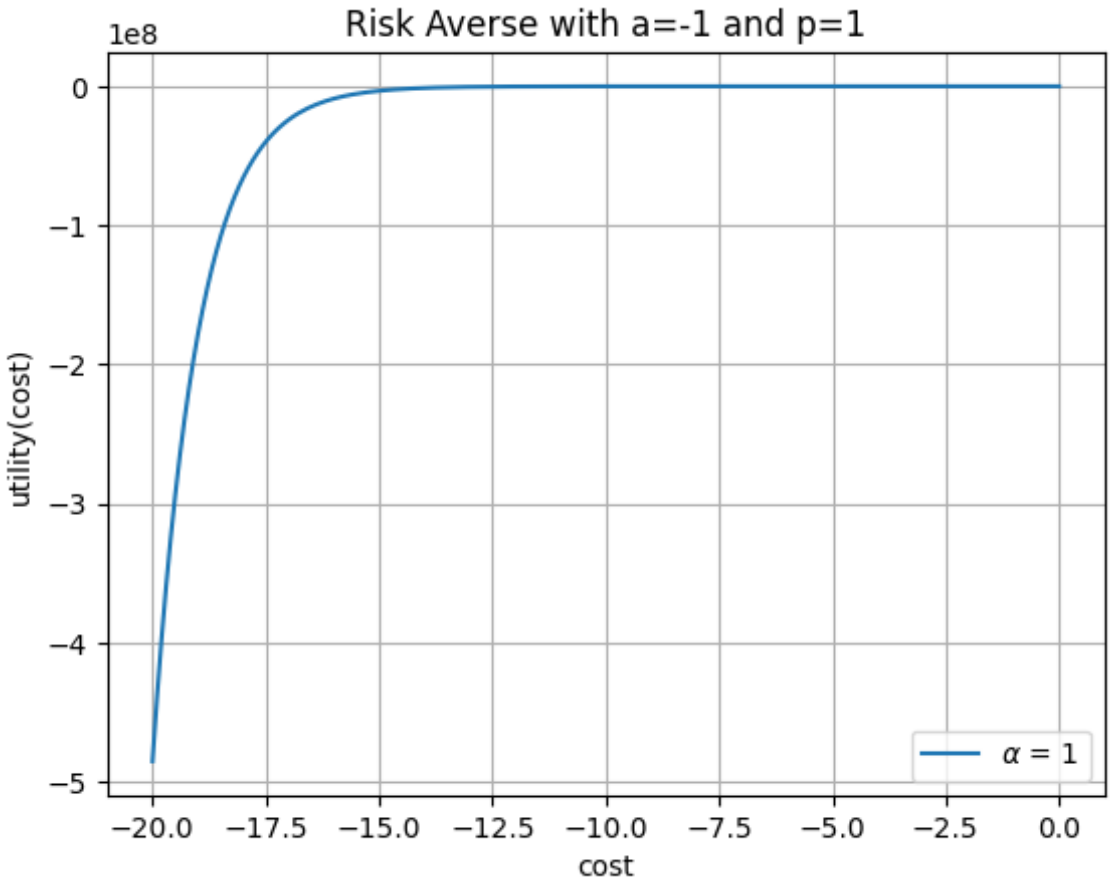

Figure 5.3: Risk averse utility function with α=1

It is evident that the lower the probability, the lower the cost, which should not be surprising since the EU just multiplies the utility with the probability. The graph for risk averse can be seen in Figure 5.6, where we can observe the same effect. As a reminder our probability used for the calculations is defined as described in Definition 4.3.1, however we use the normal probability in our graphs to make it easier to understand.

Looking at the difference between the various curves, we can see a strong contrast between risk-seeking and risk-averse. Whereas for risk seeking, the values for each specific cost do not differ much, for risk averse the costs differ by a larger margin. This reflects the expected attitude, where with a risk-averse attitude we rate low-chance operators much lower than risk-seekers.

Now if we look at how overwatering is modeled, we can see that the cost for overwatering follows the same curve as can be seen in Figure 5.7 and Figure 5.8. However it has a lower starting point than the end point of the curve for being below optimal levels. While the curve pre optimal converges against zero, the curve for overwatering cuts the last part before zero out. This means, in the case where we are 5 %nFK below optimal, we have a higher cost depending on how worse we want to value overwatering, compared to overwatering by 5%. With this we can see the difference between risk averse and risk seeking clearly.

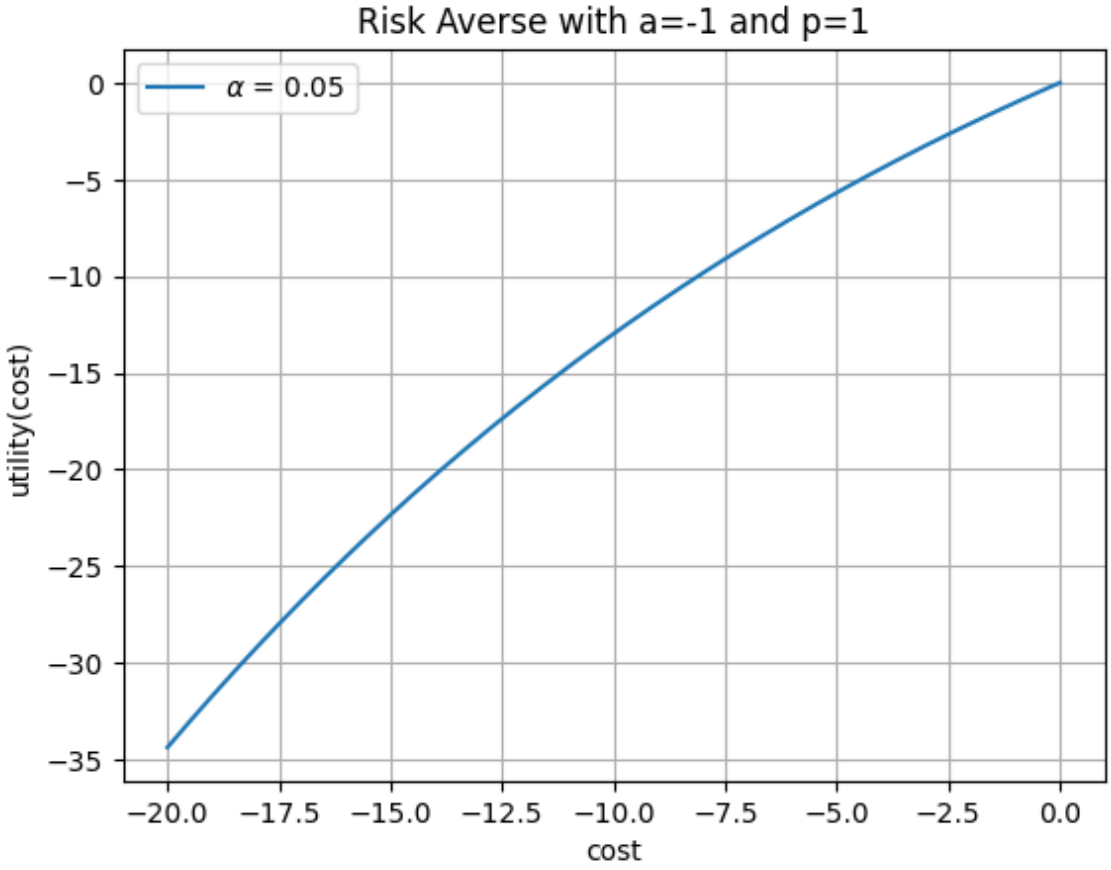

Figure 5.4: Risk averse utility function with α=0.1

For now, we go over different examples to show how the attitudes differ in the decision they make. An important note is that the differences only occur at around the optimal moisture value. The reason is that this is the only point where the answer is not as clear as before. Before we can present our examples, we define a new scenario. In this scenario, we investigate how our model deals with rainfall. Our analysis focuses on objectively evaluating the models performance in handling this new variable. We utilise the same scenario as was described in Section 5.1.1, but with the addition of rainfall and $\alpha = 0.1$. That means the cell does not need tending in any other way and the optimal moisture is 70% nFK.

For the first example in this scenario, we consider a situation where the soil moisture level is just below the optimal level. The problem can be seen in Listing C.1, where the value for X would be 65 in this example, meaning 65% nFK current moisture. Our Rainfall prediction looks like this:

- 20% : $\langle 20, 0, 0, 0, 0 \rangle$ rainfall in % nFK
- 80% : $\langle 0, 0, 0, 0, 0 \rangle$ rainfall in % nFK

Consequently, we observe two different outcomes, the risk-averse strategy opts for patience with a plan cost of -74.71, whereas the risk-seeking approach favours activating the irrigation systems with a plan cost of -18.75. The alternative plan cost for risk averse was -156.18 and -21.4 for risk seeking . In Table 5.1 we can observe the calculated EU for each case, highlighting a significant

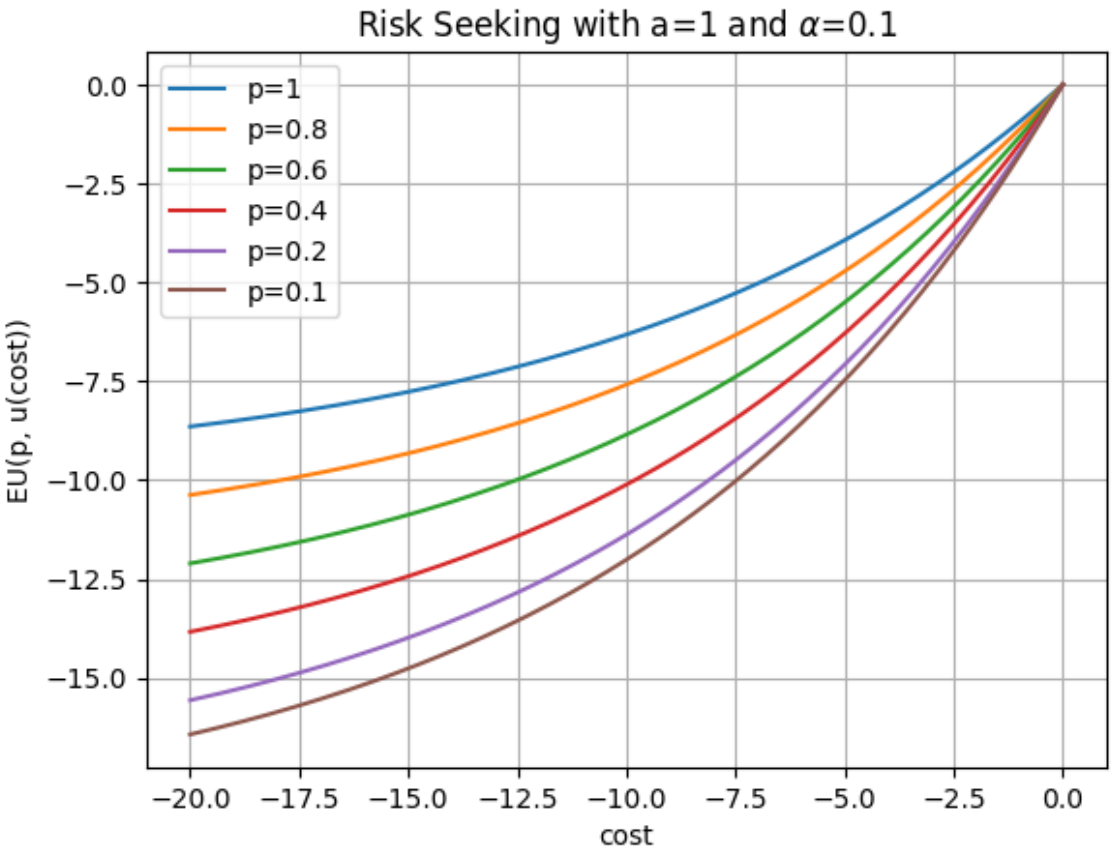

Figure 5.5: Risk seeking EU for α=0.1 with varying probabilities and cost in missing %nFK

	Risk Seeking		Risk Averse	
Action	**Rainfall**	**No Rainfall**	**Rainfall**	**No Rainfall**
Watering	−16.1	−2.65	−152.77	−3.4
Waiting	−12.84	−8.56	−44.82	−29.88

Table 5.1: EU cost for risk seeking and risk averse planning in the case of rainfall and just below optimal moisture levels

contrast in the calculated values, particularly in the context of risk-averse planning. This result is in line with our expectations, since a risk-averse mindset gives significantly more weight to the potential negative consequences of decisions, especially when compared to risk-seeking strategies. Therefore, the sub-optimal choice of irrigating when sufficient rainfall is expected, carries a greater penalty in the context of risk-averse decision making.

For the second example in this scenario, where the value for the current moisture is 60 %nFK with the following rainfall prediction: Our Rainfall prediction looks like this:

- 80% : $\langle 30, 0, 0, 0, 0 \rangle$ rainfall in % nFK

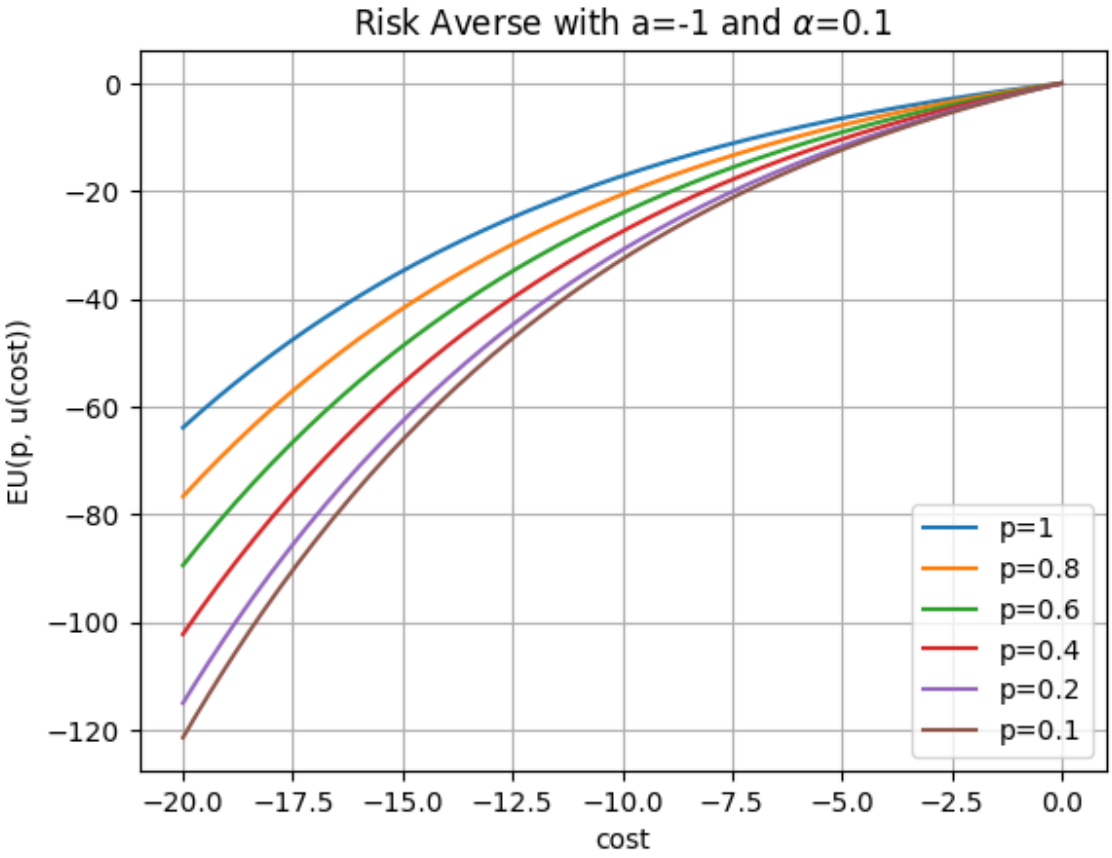

Figure 5.6: Risk averse EU for α=0.1 with varying probabilities and cost in missing %nFK

- 20% : $\langle 0, 0, 0, 0, 0 \rangle$ rainfall in % nFK

The resulting plan for risk seeking decides to water with the plan cost of -20.73 , while risk averse waits with a plan cost of -142.64. It appears that the values have increased compared to the initial example, with the risk-averse seeing a greater discrepancy. This is to be expected, as we are dealing with a more critical situation compared to before, where the difference is not just being slightly above or below the optimal moisture. In this example, we are either within an acceptable range but below optimal, or at the initial stage of water oversupply in case of the predicted rainfall. If we look at Table 5.2, we see that watering in the case of no rainfall would be the best decision. Risk averse values this case with -20.1 and risk seeking with -9.49, however we see a strong contrast for the cost of watering in the high chance that it rains. This adequately represents the two attitudes, since risk averse does not choose the best decision but chooses the safest bet, while risk seeking goes with the risky choice in hopes for the best case.

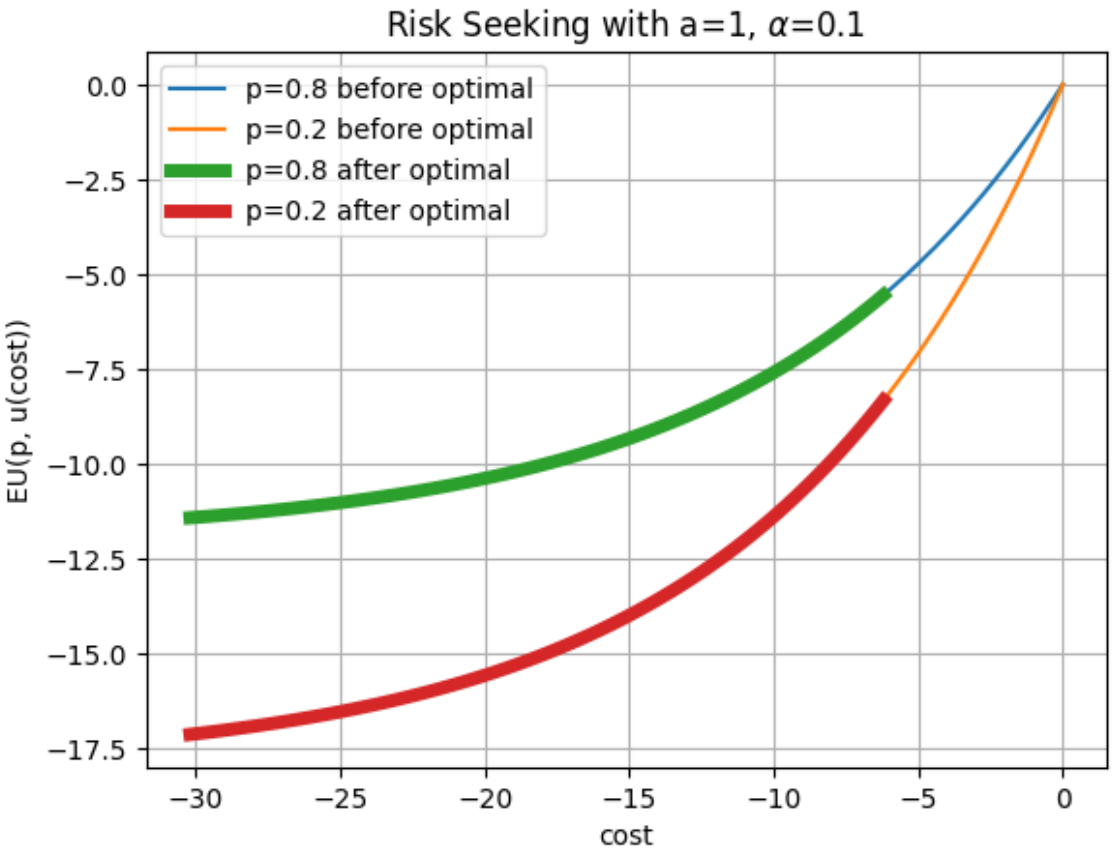

Figure 5.7: Risk seeking EU for α=0.1 with varying probabilities, cost in missing %nFK and the cost for overwatering visualized

	Risk Seeking		Risk Averse	
Action	**Rainfall**	**No Rainfall**	**Rainfall**	**No Rainfall**
Watering	−11.23	−9.49	−175.71	−20.1
Waiting	−9.91	−14.87	−57.05	−85.58

Table 5.2: EU cost for risk seeking and risk averse planning in the case of rainfall and just below optimal moisture levels

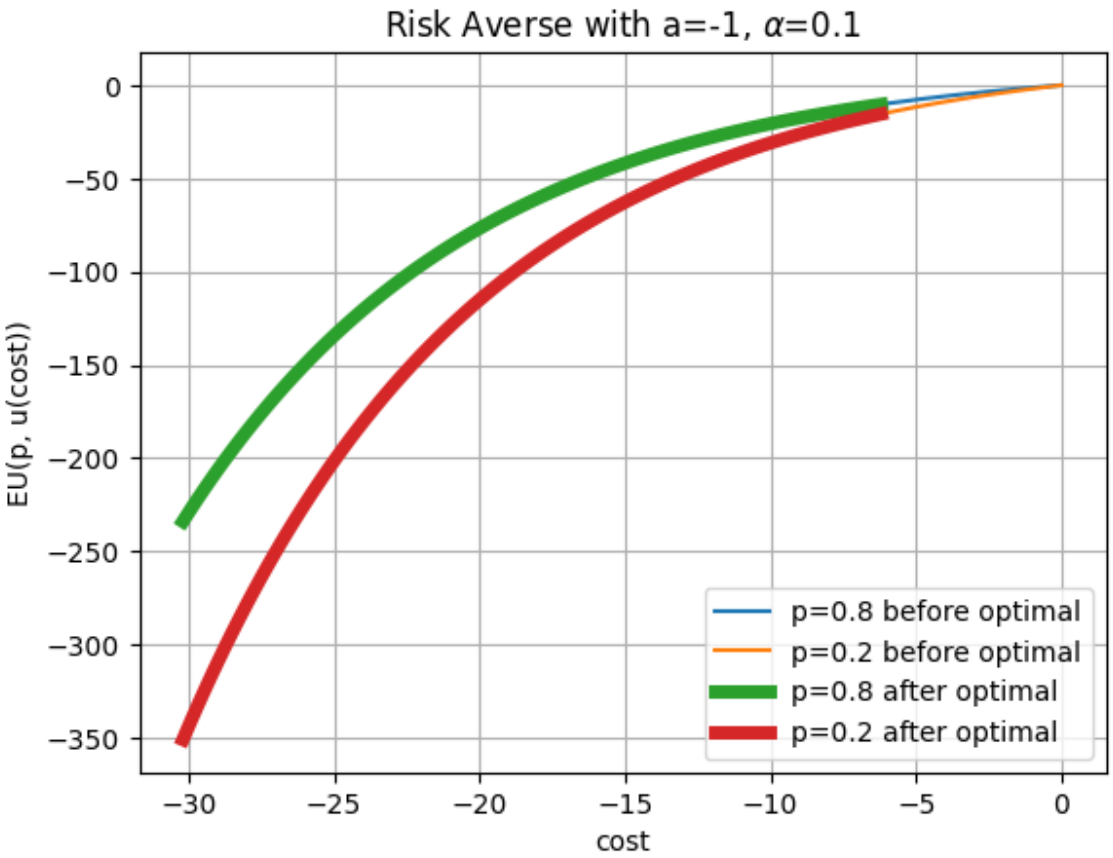

Figure 5.8: Risk averse EU for α=0.1 with varying probabilities, cost in missing %nFK and the cost for overwatering visualized

5.2 Quantitative Evaluation

In our quantitative analysis, we evaluate the time our planner needs for planning in relation to the number of cells used. To create the problem file, a simple Python script is used to generate the problem description in a text file. The code can be viewed in Listing C.3. It is noteworthy that the problem scenario is designed to illustrate the most demanding circumstances for every cell. This indicates that the every cells plants experience malnutrition, pest infestation, and other adverse conditions. Furthermore, it should be mentioned that JSHOP2 might face constraints when handling over 130 cells. The results can be seen in Figure 5.9.

As evident from the data, the planning time exhibits a linear increase in proportion to the number of cells. This observation is logical, as individual cells do not exert an influence on one another. This finding aligns seamlessly with one of our objectives, the scalability, meaning in the case we have a big field and decide that each plant has its own cell, the planner is still efficient in its planning time. This approach harmonizes with the principles of precision farming, where we cater to the precise requirements of individual plants.

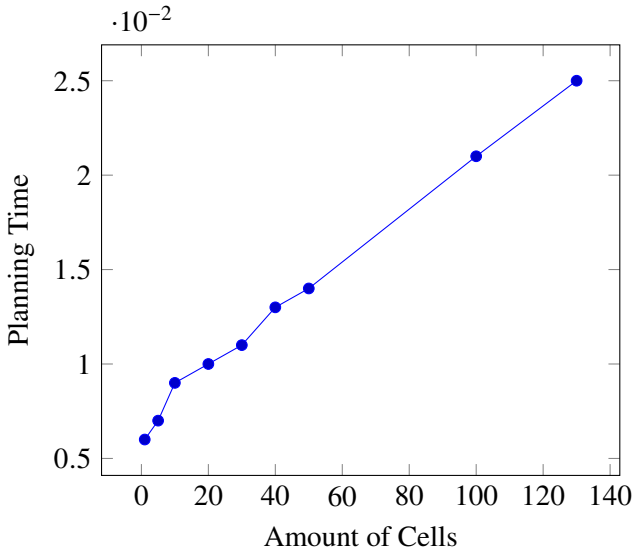

Figure 5.9: Planning time depending on the amount of cells

Days	Downfall in mm	% nFK for 10 cm soil depth
21.10.	0	79
22.10.	0	77
24.10.	10.8	67
25.10.	6.7	81

Table 5.3: Downfall and usable field capacity Source: Deutscher Wetterdienst

5.3 Qualitative Evaluation

Our qualitative evaluation aims to asses the precision of our model, for predicting the current soil moisture. To do this, we will analyze two sets of weather forecasts and evaluate how closely our predictions align with the values calculated by the Deutscher Wetterdienst.

To make this comparison, we first need to establish a way to equate precipitation in millimeters to usable field capacity (nFK), which is the metric we employ in our model. We can achieve this by converting nFK to millimeters using the formula:

$nFKmm = (\frac{nFK\%}{100}) *$ Gesamtdicke der Bodenschicht (in mm) [Wet].

For instance, if nFK is 10% and the soil depth is 10 cm, this conversion would result in 10mm of available water. With this conversion in mind, we are now in a position to compare the predictions from our model with the data in Table 5.3, obtained from the Deutscher Wetterdienst for two date ranges: the 21st to 22nd October 2023 and the 24th to 25th October 2023.

We can see that there was no downfall on our first pair of days and that the nFK went down by 2%. In comparison to our model where we predict a static reduction in nFK of 1.5% per day through evapotranspiration. Since on some days the evapotranspiration is below 1% we choose the static value of 1.5% meaning we are not that far off with our prediction.

On days with downfall, as described in our second pair of days, our model predicts a nFK of 76% instead of the measured 81% which also is not that far off considering we use our own simplified version of calculating the water retention.

6 Related Work

In recent years, the application of AI planning techniques in agriculture has received considerable attention due to its potential to increase productivity, optimise resource use and address various challenges in the agricultural sector [FAO22].

A study focusing on fertilisation rather than irrigation is [FPG+21]. They describe the requirements for designing an application that provides recommendations for decisions related to fertiliser use, such as type of fertiliser, frequency, amount of fertiliser and when to apply it. However, as they do not select a specific type of AI planning, their framework could be implemented with HTN planning. It could also be incorporated into our own domain model as they only focused on gaining knowledge and presenting how a system could be built.

A different system, which is also similar to our work here are agricultural decision support systems. An agricultural decision support system (ADSS) is a human-computer system that utilises data from various sources to provide farmers with advisory recommendations to aid their decision-making in different situations [ZMBM20]. This is similar to our model as we also utilize various information sources to offer a plan that can be viewed solely as a recommendation.

There are many ADSSs, for example the Watson Decision Platform for Agriculture developed by IBM Watson and The Weather Company. The Watson Decision Platform for Agriculture utilises artificial intelligence, machine learning, and advanced analytics to analyse Essential Field Relevant (EFR) data, extracting valuable insights and providing automated guidance for making more informed decisions. It provides a unified dashboard that permits growers to visualise vital information, such as weather forecasts, soil conditions, evapotranspiration rates, and crop stress alerts. The platform employs AI-based visual recognition of drone-acquired footage to automatically identify distinct categories and levels of pest and disease harm.

Another decision support system with a focus on irrigation, is DSIRR [Baz05]. DSIRR, a Decision Support System (DSS) tailored for assessing the economic and environmental aspects of agricultural activities, with a primary focus on irrigation, is designed to cater to both public and private sector requirements. This software simulates the economically motivated decision-making processes of farmers, enabling a precise representation of production and irrigation practices in terms of technology and agronomics. It supports the execution of both short-term and long-term analyses, with the latter incorporating inherent investment options. Solutions are derived through the application of multicriterial mathematical programming techniques. The constructed farm models allow users to quantify the consumption of water, labor, and machinery by different farm types, while considering various factors such as soil types, irrigation systems, water-yield functions, and seasonality.

In summary, numerous similar initiatives aim to generate recommendations based on available data to enhance optimal agricultural practices. These efforts collectively play a vital role in accelerating the progress of Agriculture 4.0.

7 Conclusion and Outlook

In this thesis, we set out to explore the integration of risk-aware HTN planning in the agricultural field. Our study led us through a diverse cast of topics encompassing automated planning, domain modelling, and the pressing need for risk awareness in the ever-evolving agricultural sector. We began with forming a foundational understanding of HTN planning, setting the stage for the subsequent incorporation of risk-awareness into HTN planning. Afterwards we examined the unique intricacies of agriculture and the way in which its characteristics are quantified. One of the central achievements of this research is the development of a risk-aware HTN planning model tailored to the agricultural sector with a focus on irrigation. This model offers a structured approach for considering and addressing uncertainties, like weather-related fluctuations, in the decision-making process. The implementation of this model allowed us to create theoretical situations, to test how capable our model is in addressing real-world agricultural challenges. By methodically evaluating our model across diverse categories, we gained valuable insights into its effectiveness and robustness. What we found was, that our model can correctly predict soil water retention in an acceptable range and is scalable to work with hundreds of cells. It also creates correct plans for both risk attitudes, with choosing the safest route with a risk averse attitude while taking risk with a risk seeking attitude. Therefore, we think that our model has the potential to empower farmers and decision-makers with the tools they need, to navigate complex agricultural scenarios, while mitigating risk.

Outlook

Our research has laid the foundation for future work in several areas, as we believe that our domain model can be extended and refined to more fully address the complexities of risk-aware HTN planning for farming tasks. Furthermore, given the lack of a state-based risk-aware planner, we recognise the need for further development in this area.

Our current domain model serves as a basic starting point. It provides a simplified representation of the risk-aware HTN planning problem, which is valuable for initial exploration. However, future work should focus on making it more robust and applicable to a wider range of scenarios.

For example, we could extend our domain model to include multiple cost variable operators. This would allow us to see an even greater difference in the choices made by the planner. To do this, we would also have to change the way we calculate our costs, i.e. they could consist of a score that we give according to how close a crop is to its ideal state. This ideal state could be constructed from three sources, one could be our current missing moisture cost transformed into a value from 0-100 where 100 could be the optimal state. The second could be, how close a plant is to its optimal nutrition and the last could be the plants health, in the form of pest damage or risk of damage.

Let us assume we add a cost variable operator for applying pesticides, which depends on a probability distribution of the occurrence of pests. However, we cannot apply pesticides and activate our irrigation system at the same time, so we would have to choose between watering and applying pesticides. In this situation we would have the pairs (wait-water, apply-pesticide) or (apply-water, wait-pesticide). Now the planner would have to calculate the EU of these two operators and compare them to choose the right pair. If not applying pesticide will cause serious damage, therefore the cost of waiting to apply pesticide is high, the risk-averse planner would prefer to apply pesticide rather than water. If keeping the crop at the optimum moisture level will significantly increase yield and therefore has a chance of achieving a really good 'cost', the risk-seeking planner would take the risk of not applying pesticide.

Another way of extending the system could be to schedule the different actors. For example, if we decide to use drones for fertilisation and pesticide application, we would need to schedule their flight when we have multiple cells. Otherwise there could be crashes or multiple drones trying to tend to one field.

Furthermore it could be possible to look into performance enhancing algorithms, that would try and minimize the application of big machinery, like a VR spreader, by grouping cells in need of assistance. Then the planner would have to decide between the cost of fertilising versus the cost of keeping plants in a state of malnutrition. This also could be affected by risk, since the lower the nutrition, the higher the risk for permanent damage [Ben93; BP15], which would also make for an interesting avenue of expansion.

Bibliography

[AGA22] E. Alnazer, I. Georgievski, M. Aiello. “Risk Awareness in HTN Planning”. In: *arXiv preprint arXiv:2204.10669* (2022) (cit. on pp. 17, 24–26).

[Aha23] T. Ahamed. *IoT and AI in Agriculture: Self-sufficiency in Food Production to Achieve Society 5.0 and SDG’s Globally*. Springer Nature, 2023 (cit. on p. 17).

[APRS+98] R. G. Allen, L. S. Pereira, D. Raes, M. Smith, et al. “Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56”. In: *Fao, Rome* 300.9 (1998), p. D05109 (cit. on pp. 28, 37).

[Baz05] G. M. Bazzani. “An integrated decision support system for irrigation and water policy design: DSIRR”. In: *Environmental Modelling & Software* 20.2 (2005), pp. 153–163 (cit. on p. 67).

[Ben93] W. Bennett. “Plant nutrient utilization and diagnostic plant symptoms”. In: *Nutrient deficiencies and toxicities in crop plants* 1 (1993) (cit. on pp. 30, 70).

[Bev+91] R. B. Beverly et al. *A practical guide to the Diagnosis and Recommendation Integrated System (DRIS).* Micro-Macro Publishing, Inc., 1991 (cit. on p. 37).

[BP09] S. Behera, R. Panda. “Effect of fertilization and irrigation schedule on water and fertilizer solute transport for wheat crop in a sub-humid sub-tropical region”. In: *Agriculture, ecosystems & environment* 130.3-4 (2009), pp. 141–155 (cit. on p. 45).

[BP15] A. V. Barker, D. J. Pilbeam. *Handbook of plant nutrition*. CRC press, 2015 (cit. on pp. 28, 30, 35, 70).

[BPKH88] C. Brouwer, K. Prins, M. Kay, M. Heibloem. “Irrigation water management: irrigation methods”. In: *Training manual* 9.5 (1988), pp. 5–7 (cit. on p. 38).

[BPPD+15] M. F. F. Bernardes, M. Pazin, L. C. Pereira, D. J. Dorta, et al. “Impact of pesticides on environmental and human health”. In: *Toxicology studies-cells, drugs and environment* (2015), pp. 195–233 (cit. on p. 30).

[Bre21] O. Brendel. “The relationship between plant growth and water consumption: a history from the classical four elements to modern stable isotopes”. In: *Annals of Forest Science* 78.2 (2021), pp. 1–16 (cit. on p. 35).

[BS00] J. Baldock, J. Skjemstad. “Role of the soil matrix and minerals in protecting natural organic materials against biological attack”. In: *Organic geochemistry* 31.7-8 (2000), pp. 697–710 (cit. on p. 27).

[CE12] J. L. Chávez, S. R. Evett. “Using soil water sensors to improve irrigation management”. In: *Proceedings of the 2012 Central Plains irrigation conference, Colby, Kansas, February 21-22*. Colorado State University. Libraries. 2012 (cit. on p. 37).

[CEK23] D. Cammarano, F. K. van Evert, C. Kempenaar. *Precision Agriculture: Modelling*. Springer Nature, 2023 (cit. on p. 37).

[CZC+14] H. S. Chattha, Q. U. Zaman, Y. K. Chang, S. Read, A. W. Schumann, G. R. Brewster, A. A. Farooque. "Variable rate spreader for real-time spot-application of granular fertilizer in wild blueberry". In: *Computers and Electronics in Agriculture* 100 (2014), pp. 70–78 (cit. on p. 39).

[DTS17] S. Datta, S. Taghvaeian, J. Stivers. *Understanding soil water content and thresholds for irrigation management*. Tech. rep. Oklahoma Cooperative Extension Service, 2017 (cit. on pp. 27, 35, 41).

[ERB+20] A. Erler, D. Riebe, T. Beitz, H.-G. Löhmannsröben, R. Gebbers. "Soil nutrient detection for precision agriculture using handheld laser-induced breakdown spectroscopy (LIBS) and multivariate regression methods (PLSR, Lasso and GPR)". In: *Sensors* 20.2 (2020), p. 418 (cit. on p. 35).

[ERP+13] A. Escolà, J. Rosell-Polo, S. Planas, E. Gil, J. Pomar, F. Camp, J. Llorens, F. Solanelles. "Variable rate sprayer. Part 1–Orchard prototype: Design, implementation and validation". In: *Computers and electronics in agriculture* 95 (2013), pp. 122–135 (cit. on p. 39).

[FAOa] FAO. *Climate change fans spread of pests and threatens plants and crops, new FAO study*. URL: https://www.fao.org/news/story/en/item/1402920/icode/ (visited on 10/20/2023) (cit. on p. 30).

[FAOb] FAO. *Pest and Pesticide Management - Inegrated Pest Management - Principles and practices*. URL: https://www.fao.org/pest-and-pesticide-management/ipm/principles-and-practices/en/ (visited on 10/21/2023) (cit. on pp. 31, 36).

[FAO22] FAO. *The State of Food and Agriculture 2022. Leveraging automation in agriculture for transforming agrifood systems.* Tech. rep. Rome, Italy: FAO, 2022 (cit. on pp. 17, 39, 67).

[FL07] H. Friesland, F.-J. Löpmeier. "The performance of the model AMBAV for evapotranspiration and soil moisture on Müncheberg data". In: *Modelling water and nutrient dynamics in soil–crop systems: Proceedings of the workshop on "Modelling water and nutrient dynamics in soil–crop systems" held on 14–16 June 2004 in Müncheberg, Germany*. Springer. 2007, pp. 19–26 (cit. on p. 37).

[Flu96] M. Flury. "Experimental evidence of transport of pesticides through field soils—a review". In: *Journal of environmental quality* 25.1 (1996), pp. 25–45 (cit. on p. 45).

[FPG+21] E. Firmansyah, B. Pardamean, C. Ginting, H. G. Mawandha, D. P. Putra, T. Suparyanto. "Development of artificial intelligence for variable rate application based oil palm fertilization recommendation system". In: *2021 International Conference on Information Management and Technology (ICIMTech)*. Vol. 1. IEEE. 2021, pp. 6–11 (cit. on p. 67).

[GA15] I. Georgievski, M. Aiello. "HTN planning: Overview, comparison, and beyond". In: *Artificial Intelligence* 222 (2015), pp. 124–156 (cit. on pp. 20, 22).

[GA16] I. Georgievski, M. Aiello. "Automated planning for ubiquitous computing". In: *ACM Computing Surveys (CSUR)* 49.4 (2016), pp. 1–46 (cit. on pp. 19, 33, 34, 36, 37, 40, 55).

[GNT04] M. Ghallab, D. Nau, P. Traverso. *Automated Planning: theory and practice*. Elsevier, 2004 (cit. on pp. 17, 19, 22).

[HL95] J. Hagin, A. Lowengart. “Fertigation for minimizing environmental pollution by fertilizers”. In: *Fertilizer research* 43 (1995), pp. 5–7 (cit. on p. 38).

[Ilg06] O. Ilghami. “Documentation for JSHOP2”. In: *Department of Computer Science, University of Maryland, Tech. Rep* (2006), pp. 41–42 (cit. on p. 49).

[ISP] ISPA. *Precision Ag Definition*. URL: https://www.ispag.org/about/definition (visited on 10/31/2023) (cit. on p. 26).

[Kei] L. Keiner. *Physical Oceanography Animations-Other Resources*. URL: https://ci.coastal.edu/~lkeiner/Animations/ (visited on 10/22/2023) (cit. on p. 29).

[Kog98] M. Kogan. “Integrated pest management: historical perspectives and contemporary developments”. In: *Annual review of entomology* 43.1 (1998), pp. 243–270 (cit. on p. 31).

[Kru18] J. Krumm. *Ubiquitous computing fundamentals*. CRC Press, 2018 (cit. on p. 33).

[Löp83] F.-J. Löpmeier. *Agrarmeteorologisches Modell zur Berechnung der aktuellen Verdunstung (AMBAV)*. Dt. Wetterdienst, Zentrale Agrarmeteorologische Forschungsstelle . . ., 1983 (cit. on p. 37).

[ML94] R. L. Metcalf, W. H. Luckmann. *Introduction to insect pest management*. Vol. 101. John Wiley & Sons, 1994 (cit. on p. 30).

[NBE+73] D. R. Nielsen, J. W. Biggar, K. T. Erh, et al. “Spatial variability of field-measured soil-water properties”. In: (1973) (cit. on p. 35).

[OOST14] Y. Osakabe, K. Osakabe, K. Shinozaki, L.-S. P. Tran. “Response of plants to water stress”. In: *Frontiers in plant science* 5 (2014), p. 86 (cit. on p. 35).

[Pai69] C. H. Pair. *Sprinkler irrigation*. Sprinkler Irrigation Association, 1969 (cit. on p. 38).

[PCB+08] C. Parent, N. Capelli, A. Berger, M. Crèvecoeur, J. F. Dat. “An overview of plant responses to soil waterlogging”. In: *Plant stress* 2.1 (2008), pp. 20–27 (cit. on p. 36).

[PCM+16] V. J. Pereira, J. P. A. R. da Cunha, T. P. de Morais, J. P. Ribeiro-Oliveira, J. B. de Morais, et al. “Physical-chemical properties of pesticides: concepts, applications, and interactions with the environment.” In: *Bioscience Journal* 32.3 (2016), pp. 627–641 (cit. on p. 30).

[Pen48] H. L. Penman. “Natural evaporation from open water, bare soil and grass”. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 193.1032 (1948), pp. 120–145 (cit. on p. 36).

[Rap14] T. B. Raper. *In-season drought monitoring: Testing instrumentation and developing methods of measurement analysis*. University of Arkansas, 2014 (cit. on p. 37).

[Sum77] M. E. Sumner. “Preliminary N, P, and K Foliar Diagnostic Norms for Soybeans 1”. In: *Agronomy Journal* 69.2 (1977), pp. 226–230 (cit. on p. 37).

[Sum79] M. Sumner. “Interpretation of Foliar Analyses for Diagnostic Purposes 1”. In: *Agronomy Journal* 71.2 (1979), pp. 343–348 (cit. on p. 37).

[TL+84] P. B. Tinker, A. Läuchli, et al. “Advances in plant nutrition.” In: *Advances in Plant Nutrition* 1 (1984) (cit. on p. 37).

[Wet] D. Wetterdienst. *Erlaeuterungen zur nutzbaren Feldkapazitaet*. URL: https://www.dwd.de/DE/fachnutzer/landwirtschaft/dokumentationen/allgemein/bf_erlaeuterungen.pdf?__blob=publicationFile&v=7 (visited on 10/19/2023) (cit. on pp. 28, 40, 41, 64).

[WS87] J. Walworth, M. Sumner. "The Diagnosis and Recommendation Integrated System (DRIS)". In: vol. 6. Jan. 1987, pp. 149–188. ISBN: 978-1-4612-9112-1. DOI: 10.1007/978-1-4612-4682-4_4 (cit. on p. 35).

[ZMBM20] Z. Zhai, J. F. Martinez, V. Beltran, N. L. Martinez. "Decision support systems for agriculture 4.0: Survey and challenges". In: *Computers and Electronics in Agriculture* 170 (2020), p. 105256 (cit. on p. 67).

[ZNY+19] H. Zhou, X. Niu, H. Yan, N. Zhao, F. Zhang, L. Wu, D. Yin, R. Kjelgren. "Interactive effects of water and fertilizer on yield, soil water and nitrate dynamics of young apple tree in semiarid region of northwest China". In: *Agronomy* 9.7 (2019), p. 360 (cit. on p. 45).

[ZOV07] I. A. Zelaya, M. D. Owen, M. J. VanGessel. "Transfer of glyphosate resistance: evidence of hybridization in Conyza (Asteraceae)". In: *American Journal of Botany* 94.4 (2007), pp. 660–673 (cit. on p. 30).

All links were last followed on August 17, 2023.

A Predicates

General Predicates

- (storage ?status)

Cell equipment and identification

- (cell ?cell-id)
- (drip ?drip-id) or (sprinkler ?sprinkler-id)
- (cell-type ?cell ?drip) or (cell-type ?cell ?sprinkler)

For decision making, if we should water or not

- (irrigation-cost ?cell ?irrigation-technique ?cost)
- (wait-cost ?cell ?cost)

Precision of our irrigation technique, describes step size for added moisture in nFK

- (irrigation-step d1 10)

Initial state

- (moisture-cell ?cell ?current-moisture)
- (optimal-moisture-cell ?cell ?optimal-moisture)
- (nutrition-cell ?cell ?status)
- (pest-cell ?cell ?status)
- (drip-functional ?cell ?status)

B Domain Model

Figure B.1: Part 1 of the domain model

Listing B.1 First part of the Planning Domain

```
;;------------------------Watering

(:operator (!calc-water-cost ?cell)
       ((irrigation-cost ?cell ?irrigation ?old-cost)(cell-type ?cell ?irrigation)(moisture-
cell ?cell ?moist)(optimal-moisture-cell ?cell ?optmoist)(irrigation-step ?irrigation ?water))
       ((irrigation-cost ?cell ?irrigation ?old-cost))
       ((irrigation-cost ?cell ?irrigation (call CalculatePlantWellness ?optmoist ?moist ?
water )))
       0
       )

(:operator (!calc-wait-cost ?cell)
       ((wait-cost ?cell ?old-cost)(cell-type ?cell ?irrigation)(moisture-cell ?cell ?moist)(
optimal-moisture-cell ?cell ?optmoist))
       ((wait-cost ?cell ?old-cost))
       ((wait-cost ?cell (call CalculatePlantWellness ?optmoist ?moist 0 )))
       0
       )

 (:operator (!activate-fertigation ?cell)
       ((moisture-cell ?cell ?moist)(nutrition-cell ?cell 0)(cell-type ?cell ?irrigation)(drip
 ?irrigation)(optimal-moisture-cell ?cell ?optmoist)(irrigation-step ?irrigation ?water))
       ((moisture-cell ?cell ?moist)(nutrition-cell ?cell 0))
       ((moisture-cell ?cell ?optmoist)(nutrition-cell ?cell 1))
       (call CalculatePlantWellness ?optmoist ?moist ?water)
       )

;;could update new moisture with watering cost since it represents how much water we add in a
way?
;;
(:operator (!activate-irrigation ?cell)
       ((moisture-cell ?cell ?moist)(cell-type ?cell ?irrigation)(optimal-moisture-cell ?cell
?optmoist)(irrigation-step ?irrigation ?water))
       ((moisture-cell ?cell ?moist))
       ((moisture-cell ?cell ?optmoist))
       (call CalculatePlantWellness ?optmoist ?moist ?water)
       )


(:operator (!wait ?cell)
       ((moisture-cell ?cell ?moist)(optimal-moisture-cell ?cell ?optmoist))
       ()
       ()
       (call CalculatePlantWellness ?optmoist ?moist 0 )
       )
```

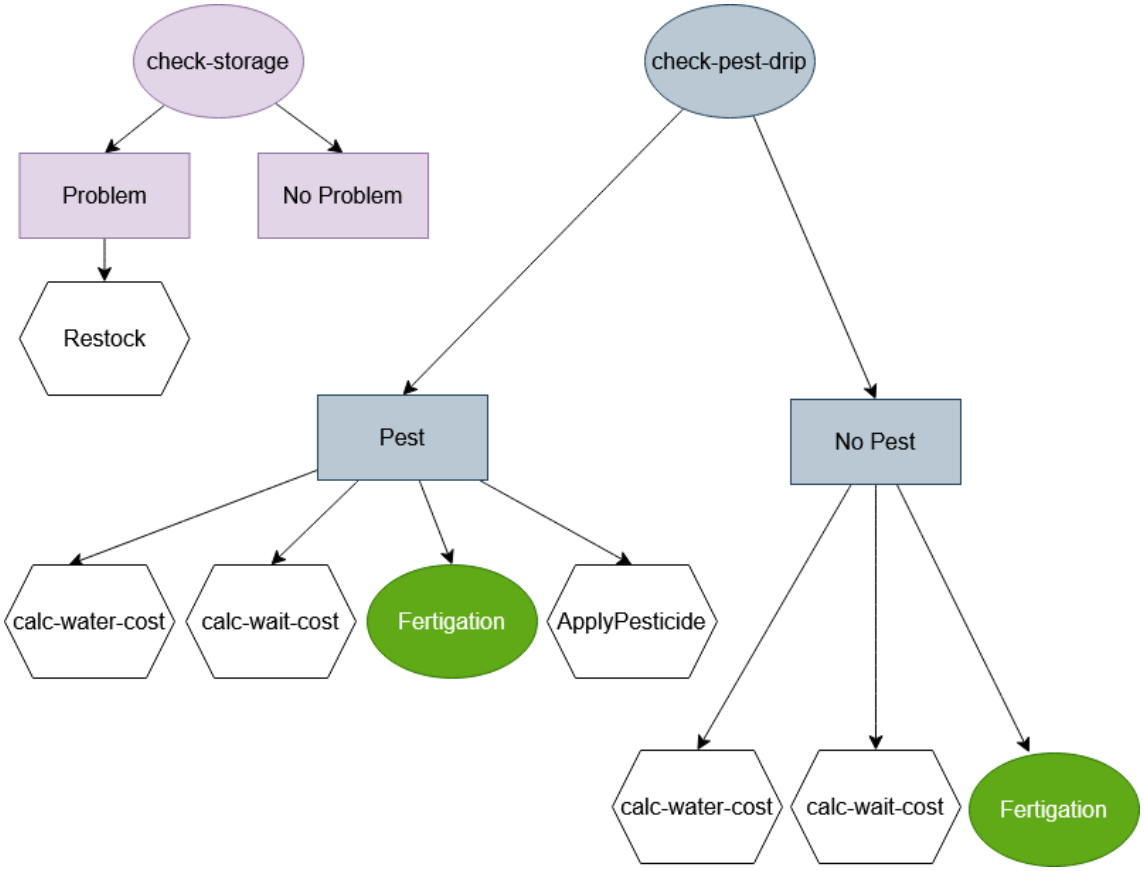

Figure B.2: Part 2 of the domain model

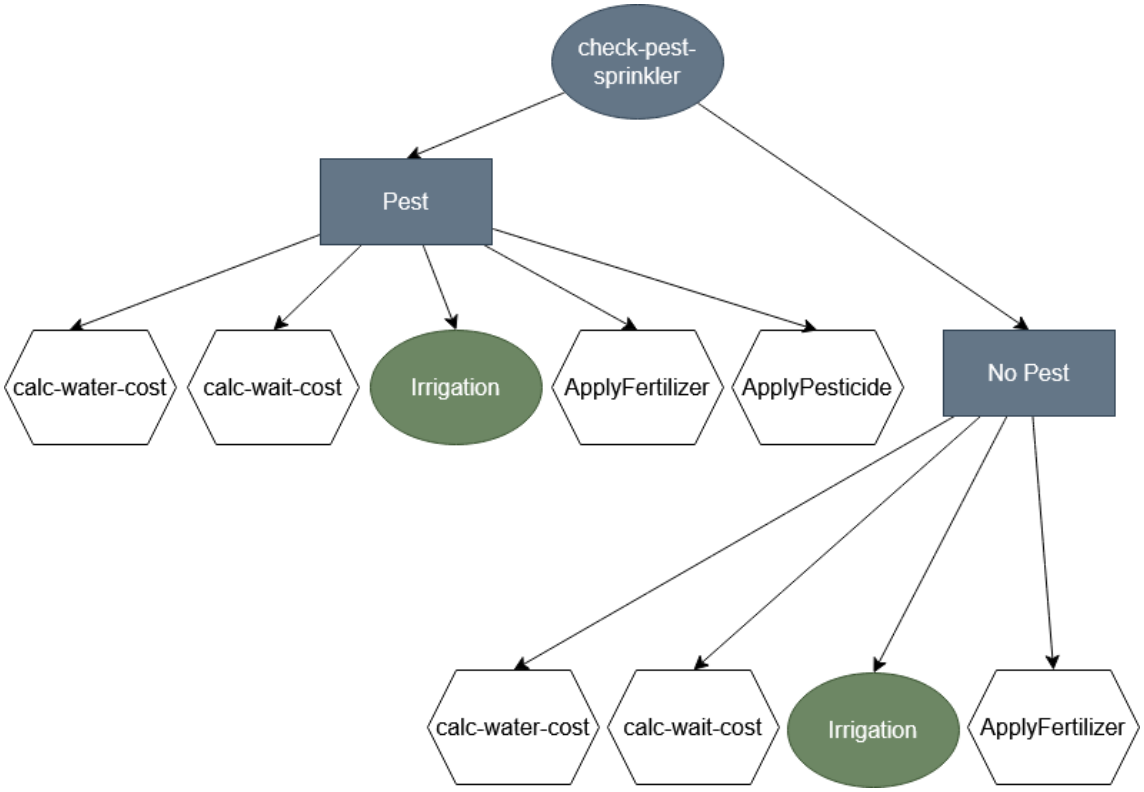

Figure B.3: Part 3 of the domain model

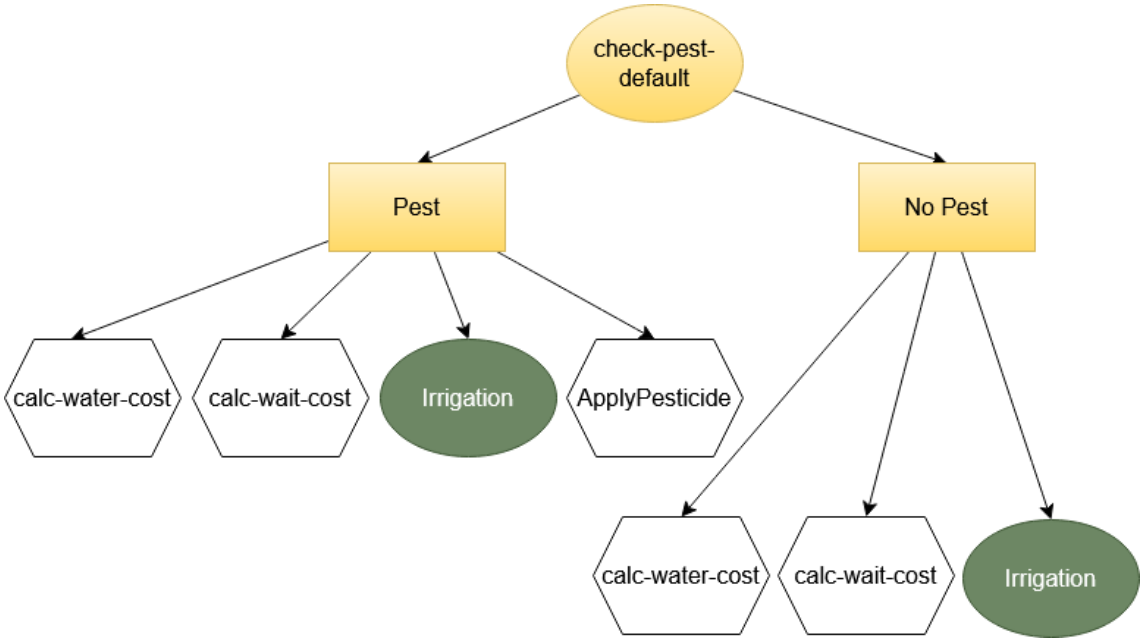

Figure B.4: Part 4 of the domain model

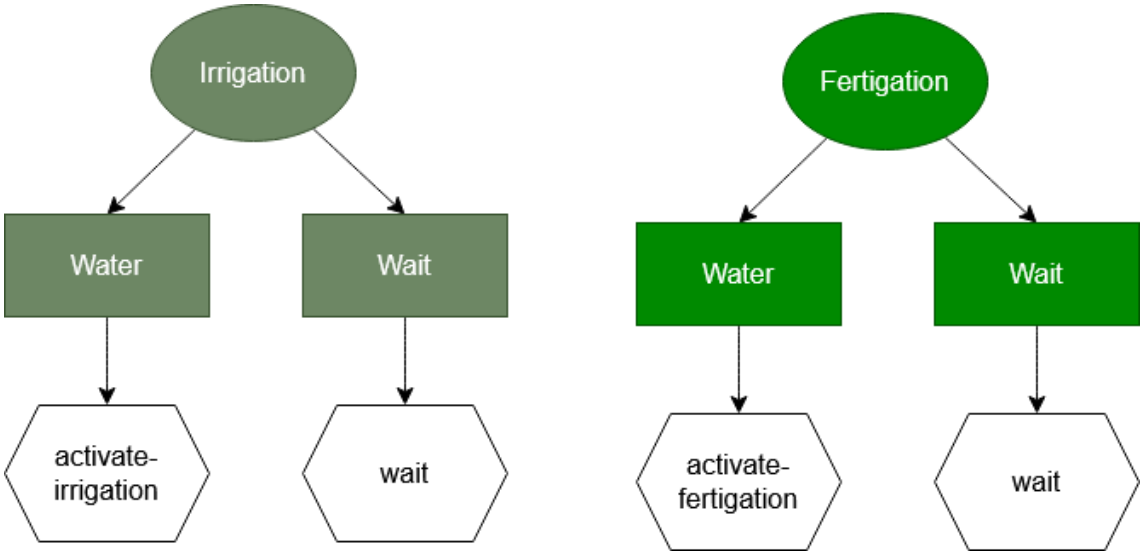

Figure B.5: Part 5 of the domain model

Listing B.2 Second part of the Planning Domain

```
;;--------Maintenance

(:operator (!repair ?cell ?drip)
       ((drip-functional ?drip 0))
       ((drip-functional ?drip 0))
       ((drip-functional ?drip 1))
       -1
       )

;;--------Refilling Storage

(:operator (!refill ?cell)
       ((storage 0))
       ((storage 0))
       ((storage 1))
       -1
       )

;;------------------------Pest/Fertilization

(:operator (!apply-pesticide ?cell)
       ((pest-cell ?cell 0))
       ((pest-cell ?cell 0))
       ((pest-cell ?cell 1))
       -1
       )

(:operator (!apply-fertilizer ?cell)
       ((nutrition-cell ?cell 0))
       ((nutrition-cell ?cell 0))
       ((nutrition-cell ?cell 1))
       -1
       )
```

Listing B.3 Third part of the Planning Domain

```
;;--------Main

(:method
(plan-main ?cell )
((cell ?cell)(cell-type ?cell ?irrigation)(drip ?irrigation))
((maintenance ?cell ?irrigation)(check-nutrition-drip ?cell))
)

(:method
(plan-main ?cell )
((cell ?cell)(cell-type ?cell ?irrigation)(sprinkler ?irrigation))
((check-nutrition-sprinkler ?cell))
)

;;--------Check Nutrition

;;missing nutrition
(:method
(check-nutrition-drip ?cell)
((cell ?cell)(nutrition-cell ?cell 0))
((check-storage ?cell)(check-pest-drip ?cell))
)

;;nutrition good
(:method
(check-nutrition-drip ?cell)
((cell ?cell)(nutrition-cell ?cell 1))
((check-pest-default ?cell))
)

;;missing nutrition
(:method
(check-nutrition-sprinkler ?cell)
((cell ?cell)(nutrition-cell ?cell 0))
((check-storage ?cell)(check-pest-sprinkler ?cell))
)

;;nutrition good
(:method
(check-nutrition-sprinkler ?cell)
((cell ?cell)(nutrition-cell ?cell 1))
((check-pest-default ?cell))
)
```

Listing B.4 Fourth part of the Planning Domain

```
    ;;--------Check Pest

;;pest alarm
(:method
(check-pest-drip ?cell)
((cell ?cell)(pest-cell ?cell 0))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(fertigation ?cell)(!apply-pesticide ?cell))
)

;;no pest
(:method
(check-pest-drip ?cell)
((cell ?cell)(pest-cell ?cell 1))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(fertigation ?cell))
)

;;pest alarm
(:method
(check-pest-sprinkler ?cell)
((cell ?cell)(pest-cell ?cell 0))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(irrigation ?cell)(!apply-fertilizer ?cell)(!
apply-pesticide ?cell))
)

;;no pest
(:method
(check-pest-sprinkler ?cell)
((cell ?cell)(pest-cell ?cell 1))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(irrigation ?cell)(!apply-fertilizer ?cell))
)
;;pest alarm
(:method
(check-pest-default ?cell)
((cell ?cell)(pest-cell ?cell 0))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(irrigation ?cell)(!apply-pesticide ?cell))
)
;;no pest
(:method
(check-pest-default ?cell)
((cell ?cell)(pest-cell ?cell 1))
((!calc-water-cost ?cell)(!calc-wait-cost ?cell)(irrigation ?cell))
)
;;--------Checking Storage
(:method
(check-storage ?cell)
((cell ?cell)(storage 0))
((!refill ?cell))
)
(:method
(check-storage ?cell)
(storage 1)
()
)
```

Listing B.5 Fifth part of the Planning Domain

```
;;--------Maintenance
(:method
(maintenance ?cell ?d)
((cell ?cell)(cell-type ?cell ?d)(drip ?d)(drip-functional ?d 0))
((!repair ?cell ?d))
)
(:method
(maintenance ?cell ?d)
(drip-functional ?d 1)
()
)
;;------------------------Watering
;;--------deciding if to water or not
(:method (fertigation ?cell)
((cell-type ?cell ?irrigation)(drip ?irrigation)(irrigation-cost ?cell ?irrigation ?water)(
wait-cost ?cell ?wait)(call >= ?water ?wait))
((!activate-fertigation ?cell))
)
(:method (fertigation ?cell)
((cell-type ?cell ?irrigation)(drip ?irrigation)(irrigation-cost ?cell ?irrigation ?water)(
wait-cost ?cell ?wait)(call < ?water ?wait))
((!wait ?cell))
)
(:method (irrigation ?cell)
((irrigation-cost ?cell ?irrigation ?water)(wait-cost ?cell ?wait)(call >= ?water ?wait))
((!activate-irrigation ?cell))
)
(:method (irrigation ?cell)
((irrigation-cost ?cell ?irrigation ?water)(wait-cost ?cell ?wait)(call < ?water ?wait))
((!wait ?cell))
)
```

C Problem Descriptions for Evaluation

Listing C.1 Problem definition for testing one cell as an Example

```
(defproblem problem agriculture
 (
 ;;general
  (storage 1)

;;cell 1
  (cell c1)
  (drip d1)

  ;;for the decision if we should water or not
  (irrigation-cost c1 d1 0)
  (wait-cost c1 0)

  ;;how precise our irrigation is
  ;;10 means we can provide moisture in steps of 10 %nFK
  (irrigation-step d1 10)

  ;;initial state
  (cell-type c1 d1)
  (moisture-cell c1 X)              <--- X value that differs
  (optimal-moisture-cell c1 70)
  (nutrition-cell c1 1)
  (pest-cell c1 1)
  (drip-functional d1 1)
  )
 (
;;goals
  (plan-main c1)
 )
)
```

Listing C.2 Problem definition for testing multiple Cells as an Example

```
(defproblem problem agriculture
 (
 (storage 1)
;;cell 0
  (cell c0 )
  (drip d0 )
  (irrigation-cost c0 d0 0)
  (wait-cost c0 0)
  (irrigation-step d0 10)
  (cell-type c0 d0 )
  (moisture-cell c0 55)             <---Current
  (optimal-moisture-cell c0 61)     <---Optimal
  (nutrition-cell c0 1)
  (pest-cell c0 1)
  (drip-functional d0 1)
;;cell 1
  (cell c1 )
  (drip d1 )
  (irrigation-cost c1 d1 0)
  (wait-cost c1 0)
  (irrigation-step d1 10)
  (cell-type c1 d1 )
  (moisture-cell c1 65)             <---Current
  (optimal-moisture-cell c1 59)     <---Optimal
  (nutrition-cell c1 1)
  (pest-cell c1 1)
  (drip-functional d1 1)
;;cell 2
  (cell c2 )
  (drip d2 )
  (irrigation-cost c2 d2 0)
  (wait-cost c2 0)
  (irrigation-step d2 10)
  (cell-type c2 d2 )
  (moisture-cell c2 60)             <---Current
  (optimal-moisture-cell c2 60)     <---Optimal
  (nutrition-cell c2 1)
  (pest-cell c2 1)
  (drip-functional d2 1)

  ...
 )
 (
(plan-main c0)
(plan-main c1)
(plan-main c2)
...
 )
 )
```

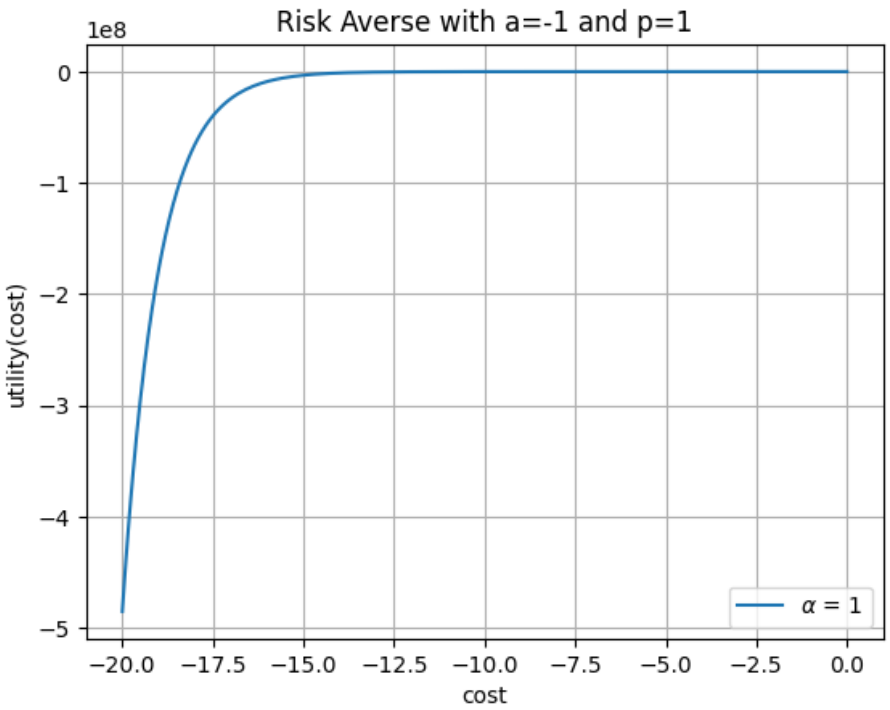

Figure C.1: Risk Averse utility function, with $\alpha = 1$

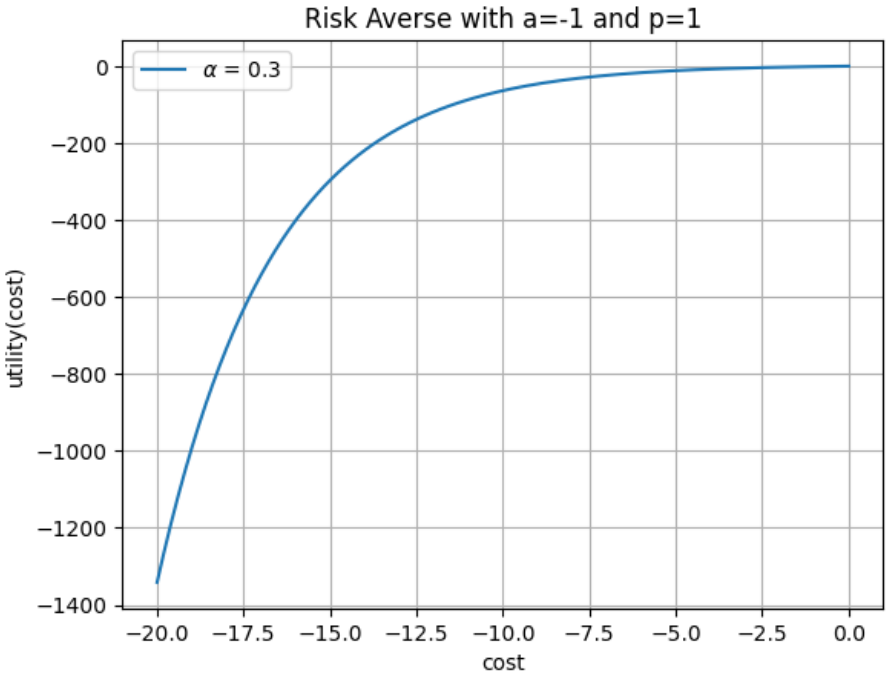

Figure C.2: Risk Averse utility function, with $\alpha = 0.3$

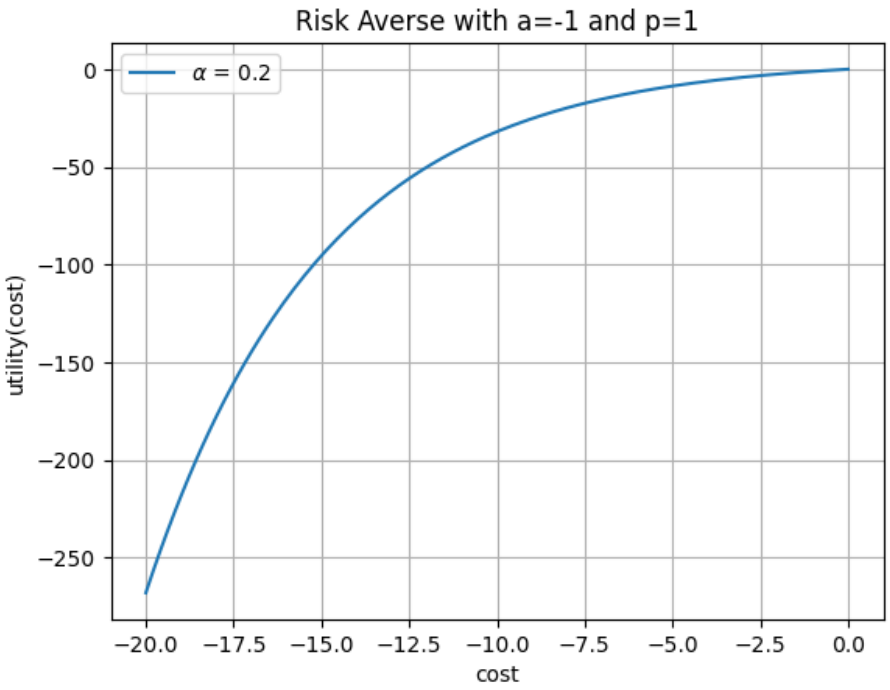

Figure C.3: Risk Averse utility function, with $\alpha = 0.2$

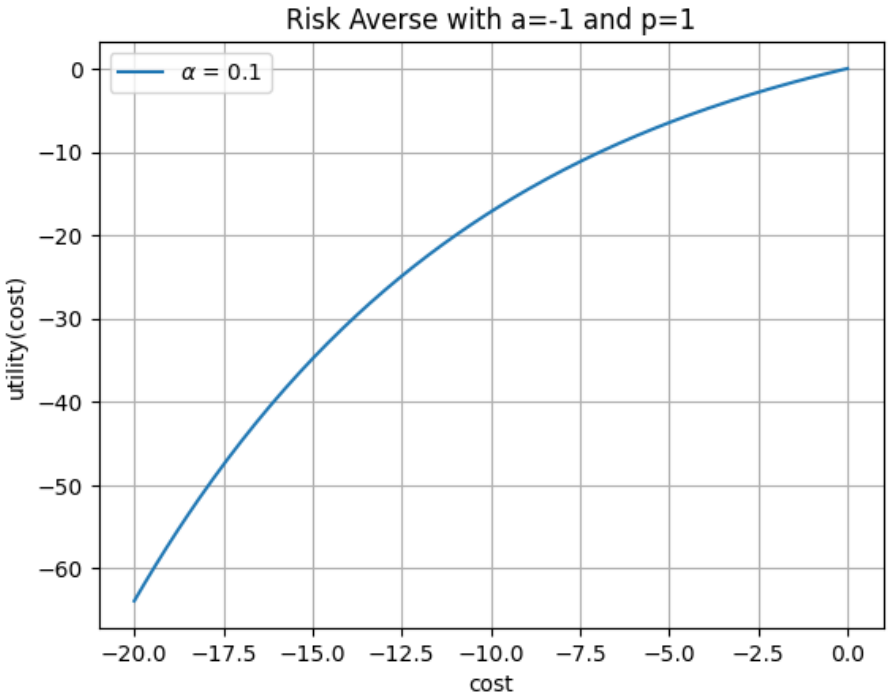

Figure C.4: Risk Averse utility function, with $\alpha = 0.1$

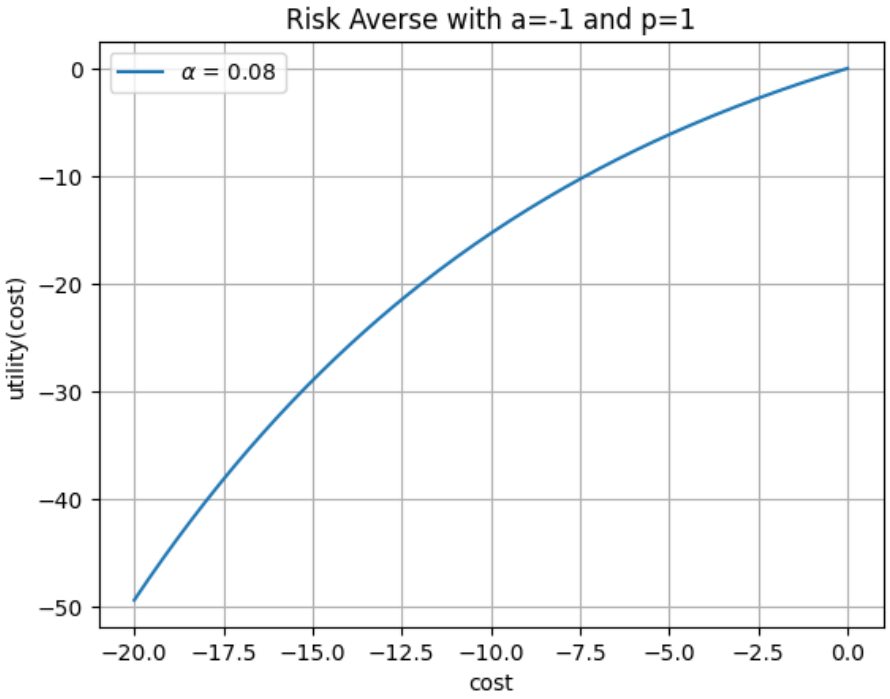

Figure C.5: Risk Averse utility function, with $\alpha = 0.08$

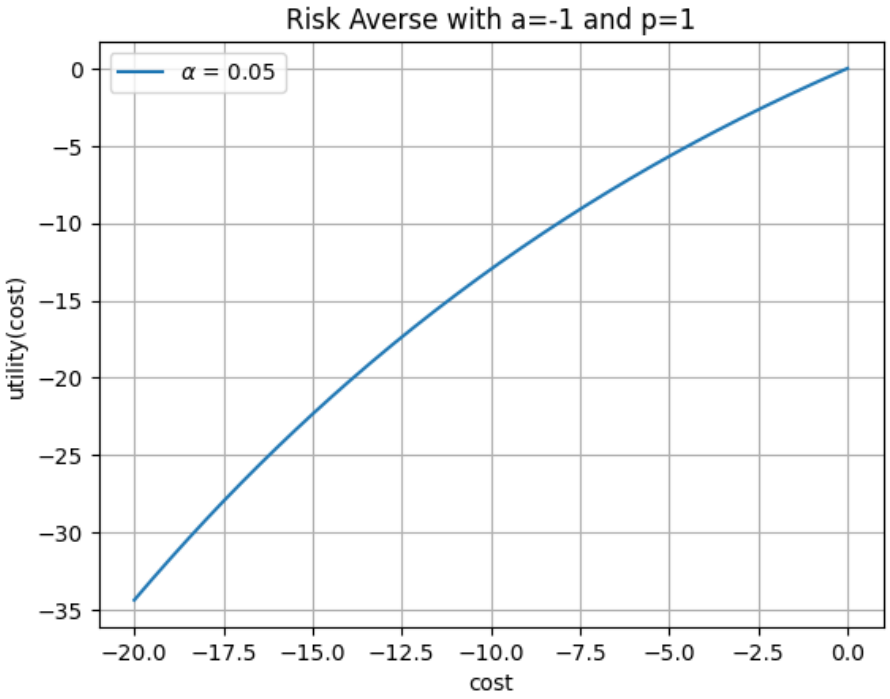

Figure C.6: Risk Averse utility function, with $\alpha = 0.05$

Listing C.3 Python Script to create a variable amount of cells to test for our planner

```
cell =  """;;cell x
  (cell c0 )
  (drip d0 )
  (irrigation-cost c0 d0 0)
  (wait-cost c0 0)
  (irrigation-step d0 10)
  (cell-type c0 d0 )
  (moisture-cell c0 20)
  (optimal-moisture-cell c0 70)
  (nutrition-cell c0 0)
  (pest-cell c0 0)
  (drip-functional d0 0)\n"""

temp_cell= cell
#amount of cells we want to test
X = 100
for i in range(X):
    new_index = i+1
    temp_cell= temp_cell.replace('c'+str(i), 'c'+str(new_index))
    temp_cell= temp_cell.replace('d'+str(i), 'd'+str(new_index))
    cell = cell +temp_cell

goals = '(plan-main c0)\n'
temp_goals = goals
for i in range(X):
    new_index = i+1
    temp_goals= temp_goals.replace('c'+str(i), 'c'+str(new_index))
    goals = goals +temp_goals

problem = "(defproblem problem agriculture \n (\n (storage 1)\n"+cell+"\n ) \n (\n"+goals+"\n
) \n )"

with open('problem.txt', 'w') as file:
  file.write(problem)
```

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Vaihingen, 06.11.2023, J Adomat

place, date, signature