Modeling and Planning for Uncertainty
within Mission Operations

Allan Hofman Johnsen - s062386

October 31, 2012

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary (English)

Part of the thesis has been a study of different approaches for planning under uncertainty. The primary focus has been the planning approaches, "*Planning Based on Markov Decision Processes (MDP's)*, *Planning for Extended Goals with Progression of Computation Tree Logic (CTL)*, and *Epistemic Planning*." Transition systems in general has also been investigated.

The basic steps with respect to using the planning approaches have been explained briefly. This includes the general model, goal type, planning problem, planning approach, solution type, and agent architecture.

The result of the investigation of the planning approaches was a comparison of the different advantages and disadvantages. The focus was mainly expressibility with respect to uncertainty. The most notable findings were:

- State transitions can not be partially observable during plan execution for any of the approaches (that observers the transitions made).
- The probabilistic transition systems could model likelihood on action outcomes.

It should be noted that the findings are purely based on the approaches as they have been specified in this report. There are many extensions/alterations to the approaches which improve on some of the limitations found in this thesis.

The *DTUsat2* satellite has been studied as part of the thesis. The study of the satellite aimed to identify some of the planning problems with respect to *DTUsat2*. The planning problems that was looked at are problems where dealing with uncertainty is paramount to correct planning and operation within the domain.

Recharging the battery of the *DTUsat2* satellite was one of these planning problems. The approaches, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL,* and *Epistemic Planning*," was tested with respect to the battery recharging planning problem.

The most notable findings regarding the models and the test executions were:

- It is necessary to model some kind of likelihood on action outcomes to avoid unrealistic plans.

- There are states that needs to be avoided for safety critical reasons. There is a risk of the battery exploding if recharging is done when the temperature of the battery is below 0 °C or above 45 °C.

Part of the thesis is an evaluation of the best planning approach for the planning problem regarding recharging the battery. This evaluation is of course limited to the three approaches that has been tested.

Planning for Extended Goals with Progression of CTL was recommended. The recommendation was made both on the general analysis of the different advantages and disadvantages but also on the test executions made.

The most prominent advantage of *Planning for Extended Goals with Progression of CTL* was the expressiveness of *Extended Goals*. Path requirements can be guaranteed to hold. It can, for example, be guaranteed before planning is done that specific states never will be visited if a plan is found. Recharging outside the temperature bounds can therefore be avoided. These kind of safety critical guarantees, that are made before planning is done, are relevant assuming planning is done on the satellite.

Summary (Danish)

Et af målene for denne afhandling har været en undersøgelse af de forskellige tilgange til planlægning under usikkerhed. Det primære fokus har været planlægningsmetoderne: "*Planning Based on Markov Decision Processes (MDP's)*, *Planning for Extended Goals with Progression of Computation Tree Logic (CTL)* og *Epistemic Planning*". Transitionssystemer generelt er også blevet undersøgt.

De basale skridt forbundet med at bruge planlægningstilgangene er blevet forklaret kort. Det omfatter den generelle model, måltype, planlægningsproblem, planlægningstilgang, løsningstype og agent arkitektur.

Undersøgelsen af planlægningstilgangene mundede ud i en sammenligning af fordele og ulemper. Fokus var primært på udtrykskraften i forhold til usikkerhed. De mest nævneværdige fund var:

- Et tilstandsskift kan ikke være delvist observerbart under eksekveringen af en plan for nogen af metoderne (der observerer de aktuelle transitioner der bliver lavet).
- De probabilistiske transitionssystemer kan modellere sandsynlighed på udfaldene af en handling.

Det skal bemærkes at undersøgelsen udelukkende bygger på metoderne som de er specificeret i denne rapport. Der findes mange udvidelser/ændringer til metoderne der forbedrer nogle af begrænsningerne der er fundet i denne afhandling.

DTUsat2 satellitten er blevet undersøgt som en del af denne afhandling. Undersøgelsen af satellitten havde til formål at identificere nogle af de planlægningsproblemer der er i forbindelse med satellitten. Planlægningsproblemerne som blev undersøgt var problemer hvor håndtering af usikkerhed var essentielt for korrekt planlægning og handling inden for domænet.

Genopladning af satellittens batteri var et af disse planlægningsproblemer. Metoderne: "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL* og *Epistemic Planning*" blev tested med hensyn til planlægningsproblemet omhandlende genopladningen af batteriet.

De mest nævneværdige fund angående modellerne og testene var:

- Det er nødvendigt at modellere sandsynligheder på udfaldene af handlinger (eller at sortere udfaldene på en kvalitativ måde) for at undgå urealistiske planer.
- Der eksisterer tilstande der skal undgås af sikkerhedskritiske årsager. Der er en risiko for at batteriet eksploderer, hvis man genlader når batteriet er under 0 °C eller over 45 °C.

En del af afhandlingen er en evaluering af den bedste planlægningstilgang for planlægningsproblemet omhandlende genopladning af satellitens batteri. Det fundne resultat er selvfølgelig kun i forhold til de tre testede planlægningsmetoder.

Den anbefalede planlægningstilgang blev fundet til at være *Planning for Extended Goals with Progression of CTL*. Anbefalingen blev baseret både på den generelle analyse af de forskellige fordele og ulemper og på testene af planlægningstilgangene.

En af hovedfordelene ved *Planning for Extended Goals with Progression of CTL* er udtrykskraften af *Extended Goals*. Der kan blandt andet garanteres krav omkring den tagne sti. Det kan for eksempel garanteres før planlægning at specifikke tilstande ikke bliver besøgt, hvis en plan bliver fundet. Genopladning af batteriet uden for temperatur rammerne kan derfor undgås. Denne slags sikkerhedskritiske garantier, der laves før planlægning, er relevante hvis man antager at planlægningen foregår på satellitten.

Preface

This thesis, 30 ECTS credits, was prepared at the department of Informatics and Mathematical Modeling at the Technical University of Denmark in fulfillment of the requirements for acquiring a M.Sc. in Informatics.

The thesis deals with different methods of planning under uncertainty. The primary focus has been the methods, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL*, and *Epistemic Planning*." These selected planning approaches have been tested on planning domains inspired by the *DTUsat2* satellite.

The thesis consists of:

- An introduction to the different areas of uncertainty.
- An introduction to different planning approaches that can deal with some uncertainty.
- Planning problems with respect to the *DTUsat2* satellite and test executions of selected planning approaches.
- Recommendations regarding best planning approach for specific planning problems.

Lyngby, November 1, 2012

Allan Johnsen

Acknowledgements

I would like to thank my supervisor Thomas Bolander and his co-supervisor Søren Bøg for the help they have provided throughout the project. It goes without saying that there would not have been a project without their help.

Next I would like to thank the current team on the *DTUsat2* project. Their combined knowledge regarding the satellite has been invaluable.

My family and friends also deserves thanks for the support they have provided and the patience they have shown as I have been submerged in this thesis.

Contents

Chapter 1

Introduction

This chapter provides background knowledge and introduces the purpose of the thesis. It also contains an overview of the report structure. In the structural overview there will be a brief description of what each chapter contains.

1.1 Background Knowledge

Planning under uncertainty is the common term for planning when aspects of the planning domain are uncertain. Actions might for instance have multiple possible outcomes. When there are multiple possible outcomes of an action there is a non deterministic choice between the outcomes. That is the actual outcome is chosen non deterministically, but it is known to be part of the set of possible outcomes. In case the outcome of an non deterministic action is not observable after the action has been executed the acting agent is said to have partial observability of the outcome. This is because the agent knows the set of possible outcomes, but the agent does not know which outcome is the actual outcome.

When modeling and planning for uncertainty within a field one course of action would be identifying the kind of uncertainty that is present within the field. Then the appropriate modeling and planning methods for the planning problems can be found.

1.2 Purpose of the Thesis

This thesis focuses on specific planning approaches, and aims to compare these approaches based on their coverage of the different areas of uncertainty in general. In addition to this the planning approaches will be compared based on manual test executions on planning problems within the field.

For this project the field pertains to mission operations in relation to the student satellite *DTUsat2*. Part of the thesis is an investigation of the planning problems within this

domain where dealing with uncertainty is paramount to correct planning and operation.

The planning approaches that will be the primary focus are, "*Planning based on MDP's*, *Planning for Extended Goals with Progression of CTL*, and *Epistemic Planning*," although less complicated approaches are also explored.

It will therefore be these selected planning approaches, that as mentioned, will be tested on different planning problems in case they are applicable. The goal is to identify key advantages and disadvantages of using the different approaches for the different planning problems. For the planning problems that have been used in the testing phase, it will be evaluated which planning approach is preferable. That is of course if an approach can be recommended.

1.3 Structure of the Report

Chapter two *Planning under Uncertainty* begins by clarifying what is meant by uncertainty when planning in *Artificial Intelligence*. Some of the different areas or topics of uncertainty are then described. The chapter proceeds to describe different planning approaches. Finally chapter two contains an analysis of the coverage of the different planning approaches with respect to the different areas of uncertainty.

In the third chapter *Domain Analysis* the DTU student satellite *DTUsat2* is investigated and some of the planning problems with respect to the satellite are looked into. The chapter proceeds to analyze what it takes to model and plan for selected planning problems, and what simplifications that are necessary for the planning approaches to be applicable. The chapter finishes by looking into the manual test executions of the planning approaches.

Chapter four *Recommendation* contains a comparison of the planning approaches based on the manual test executions and the previous analysis from chapter two. The comparison highlights the advantages and disadvantages relevant for each of the selected planning problems. Finally there is made a recommendation of the most appropriate planning approach.

Chapter five *Conclusion* is a summary of the results of the thesis. This chapter highlights the conclusions that has been made throughout the report.

Chapter 2

Planning under Uncertainty

This chapter will introduce the reader to uncertainty and how it affects planning. Different approaches for planning under uncertainty will be introduced. There are different methods to model the planning domain and different methods of extracting plans from these models. Naturally an introduction to some of these approaches will be included in the chapter. Finally the coverage of the different areas of uncertainty will be investigated for some of the introduced planning approaches.

2.1 Uncertainty

The notion of *Planning under Uncertainty* is a broad area. The term uncertainty is used frequently within the field of planning, there are however different definitions depending on the material being studied. The whole field might therefore seem a bit intangible when first introduced.

The definition from *"The STRIPS Assumption for Planning Under Uncertainty"* [1]:

"An agent plans under uncertainty whenever it can not flawlessly predict the state of the environment resulting from its actions."

This description does not cover multi-agent systems or exogenous events. Exogenous events are events that change the environment where the change has not been predicted by the agent. Therefore in this project a wider definition is used:

"An agent plans under uncertainty whenever it can not flawlessly predict the state of the environment resulting from: its actions, other agents actions, or exogenous events."

This description covers what planning under uncertainty is in general. It is however necessary to describe the different areas of uncertainty in more detail to get a deeper understanding of the field.

2.1.1 Topics of Uncertainty

The different areas of uncertainty that will be investigated in this report are, "*Partial Observability of States*, *Uncertainty with respect to Actions*, *The Time Aspect*, *Uncertainty regarding Quantities*, *Multiple Agents and Concurrency*, and *Exogenous Events*."

Partial Observability of States

Partial observability of states describes the situations where there exists uncertainty about what state the system is in. The agent might consider multiple worlds possible.

Imagine that an agent borrows a car where the gas gauge is broken. He knows that the tank is either half empty or full. He considers both situations possible. In this example the agent has partial observability of the world. Knowledge about the world might not be present during planning but could be present during execution e.g. the agent knows if the tank was full or half empty when it runs dry at his parents or half way there.

Uncertainty with respect to Actions

Uncertainty regarding an action mostly refer to uncertainty on the effects of the action. This is often modeled as actions having multiple sets of effects (outcomes) and a probability distribution on the sets. Likewise there could be modeled uncertainty on the preconditions (prerequisites) of an action.

When there is only one outcome of an action the action is deterministic. When there are multiple possible outcomes of an action there is a non-deterministic choice of the actual outcome of the action. If the actual outcome is known after execution of the action the outcome is said to be fully observable. If the outcome is not known after execution of the action the outcome is said to be partial observable because it is known that the actual outcome is part of the defined set of outcomes, but it is not known which outcome that is the actual outcome.

Take the situation of the agent with the borrowed car. Imagine the action of driving can either take the agent all the way to his parents or only halfway there. This corresponds to the action of driving having different effects. There is a non-deterministic choice of what effect will be applied. Either the agent gets to his parents, or the agent gets halfway there.

Assume that it is known that 80 % of the time the agent reaches his parents. The probability distribution on the effects of the action would then correspond to 80 % of the time the agent reaches his parents, and 20 % of the time he only goes halfway.

Because the agent knows if he is halfway to his parents or if he is at his parents after execution of the action the outcome is fully observable.

For the partial observable case consider an agent buying a Cola from a vending machine. When the agent pushes the Cola button he does not know which internal container the Cola comes from. The agent just knows that the machine gives him a Cola. If the agent considers getting a Cola from each of the internal containers a different effect he has partial observability of these effects. He does not know what the (complete) outcome of the non-deterministic action is after executing the action.

The Time Aspect

The time aspect could for instance be uncertainty about how long the effects of an action holds or uncertainty about the time an action takes.

The topic covers modeling time continuously or discretely or a hybrid hereof. See *The Basics of System Dynamics: Discrete vs. Continuous Modelling of Time* [2] for an explanation of the different approaches.

Taking the example from earlier with the agent and the borrowed car there might be uncertainty regarding how long the action of driving to the agents parents take. That is an example of uncertainty on the duration of an action.

Consider the situation of an agent eating a meal. The action of eating has the effect that the agent is not hungry. This effect only lasts a given duration. The agent might get hungry again after 4-5 hours. This is an example of uncertainty on the duration of an effect of an action.

Uncertainty regarding Quantities

This topic covers quantities and the uncertainty about consumption or production of these or uncertainty regarding the size of quantities. The topic covers continuous as well as discrete quantities.

Take the example where a rover needs to pick up soil. The amount picked up by its grab varies. It has a fixed sized cargo hold. This could be an example of uncertainty in the amount held in the cargo hold. Uncertainty on the size of a quantity.

The rover uses power when it grabs soil or moves around. The amount of power used varies depending on how much soil the grab picks up, or how far the rover needs to move. The rover has solar panels and a battery. Depending on the direction of the solar panels, and the position of the rover, the solar panels produces varying amounts of electricity. This is an example of uncertainty regarding consumption and production of power. Uncertainty on both in-going and out-going flows.

Multiple Agents and Concurrency

Multiple agents accessing resources concurrently is an example where dealing with concurrency issues is vital for correct performance.

Take the example where five people are editing copies of the same report independently of each other. When they are done they all want to save at the same time. The one that saves his copy last overwrites the changes just saved by the others. This example shows uncertainty on the result of an agents action because of multiple agents acting in the same environment concurrently. The issue arises because of writing to a resource concurrently.

It might be that only one person is allowed to work on the report at a time. This will give rise to uncertainty regarding who gets access to the resource (the report) and uncertainty on the average waiting time before one gets access to the resource. Will every person eventually get access to the resource?

Then there is the situation of cooperation, coordination, and collaboration of agents. Depending on how many of the previous described uncertainties a model can handle, the planning situation changes.

When collaborating the uncertainty present for every agent needs to be taken into account. In case of independent goals the multiple agents just need to plan so they are out of each others way. Independent plans/goals might give rise to conflicts of interest.

For an example of collaboration imagine the five people each writing a section of the report. Every section could take an uncertain amount of time. Some of the sections might have dependencies i.e. they can not be written before one or more of the other sections have been written.

The agent responsible for the conclusion or last section might be unable to start his work before the other agents have completed their sections. In this situation the agent needs to plan for the accumulated writing time of the other agents and uncertainties regarding their writing time.

This example used time as the only metric for planning, but quality of the different report sections might also be a metric. So the goal more naturally is, "*create the best report possible within the time limit*".

Exogenous Events

Exogenous Events deals with uncertainties regarding environment changes. This could for instance be uncertainties regarding propositions i.e truth values changing seemingly at random. For information on propositions see [3].

The ocean is an example of an environment which is unpredictable. For instance an area which is rich on plankton might change, from one day to the next, because of currents drifting the plankton away.

Seen from the point of view of a fish or a whale this is an exogenous event. Humans,

intelligent external observers, might however be able to predict the flow.

2.2 Transition Systems

Transition systems is one approach of modeling planning domains (with uncertainty). The section is inspired by [4] and partially [5] and [6].

2.2.1 Definitions and Examples

The most basic transition system is the deterministic transition system. Following *Ghallab et al., see [4]*, any classical planning domain can be represented as a restricted state-transition system which is just a finite deterministic transition system. The definition can be seen below:

Definition 2.2.1 *(Deterministic Transition System) - A deterministic transition system (DTS) is a triple M = (S,A,γ), where:*

- *S is a finite set of states.*
- *A is a finite set of actions.*
- *γ: $S \times A \rightarrow S$ is the state transition function.*

The example seen in figure 2.1 is a deterministic transition system illustrating that the lights in a room can either be *On* or *Off*. The actions have deterministic outcomes i.e. the action *Turn_On* will always turn the light on when in the state *Off*, and the action *Turn_Off* will always turn the light off when in the state *On*.

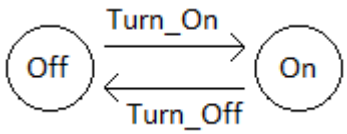

Figure 2.1: Example of a Deterministic Transition System.

The example from figure 2.1 is of course a simplification of a real world scenario. The example can be improved by for instance assuming that the action of turning on the light might fail. The acting agent might for instance not find the switch in the dark every time. Introducing two different outcomes of an action means introducing non-determinism. The definition of the non-deterministic transition system can be seen below:

Definition 2.2.2 *(Non-Deterministic Transition System) - A non-deterministic transition system (NDTS) is a triple M = (S,A,γ), where:*

- *S is a finite set of states.*

- *A is a finite set of actions.*
- γ*:* $S \times A \rightarrow 2^S$ *is the state transition function.*

The example seen in figure 2.2 is a non-deterministic transition system illustrating the improved example where turning on the lights might fail.

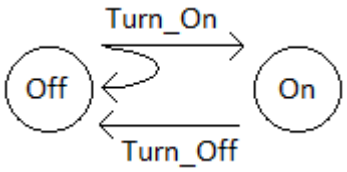

Figure 2.2: Example of a Non-Deterministic Transition System.

The example from figure 2.2 can be improved further. Take the case where it is known how often the acting agent finds the switch in the dark. Expressing this extra knowledge in the transition system would be an improvement. This can for instance be done by using a probabilistic transition system. The definition can be seen below:

Definition 2.2.3 *(Probabilistic Transition System) - A probabilistic transition system (PTS) is a triple M = (S,A,P), where:*

- *S is a finite set of states.*
- *A is a finite set of actions.*
- *P:* $S \times A \times S \rightarrow \mathbb{R}_{]0,1]}$ *is the probabilistic state transition function.*

$P(s, a, s')$ is the probability of transitioning from state s to state s' when doing action a. Another notation for the probability $P(s, a, s')$ is $P_a(s, s')$. The probability is known to be in the interval]0, 1].

The probabilities of all the different transitions from doing an action in a state should sum up to 1:

$$\sum_{s' \in S} P_a(s, s') = 1 \tag{2.1}$$

Assuming that the acting agent finds the switch 95 % of the time the probabilistic transition system can be represented as shown in figure 2.3 on the facing page. Note that 100 % probabilities are normally left implicit.

The probabilistic transition system can be extended by including costs and rewards. For simplification lets refer to this kind of transition system as a Markov Decision Process (MDP) although the term MDP covers more types of transition systems. The definition is given below:

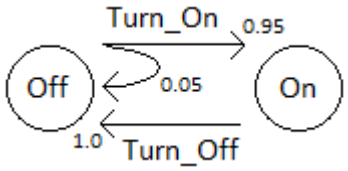

Figure 2.3: Example of a Probabilistic Transition System.

Definition 2.2.4 *(Markov Decision Process) - A MDP is a tuple M = (S,A,P,R,C) , where:*

- *S is a finite set of states.*
- *A is a finite set of actions.*
- *P is defined as in the Probabilistic Transition System, see definition 2.2.3 on the preceding page.*
- *R:* $S \to \mathbb{R}$ *is the reward function.*
  R(s) is the estimated reward of being in the state s.
- *C:* $S \times A \times S \to \mathbb{R}$ *is the cost function.*
  $C(s, a, s')$ *is the estimated cost of transitioning from s to* s' *by action a. Another notation for the cost* $C(s, a, s')$ *is* $C_{s'}(s, a)$.

Let the example MDP be given by rewards and costs as represented in figure 2.4 and state names, action names, and probabilities as shown in figure 2.3.

From figure 2.4 it is seen that there is a cost of one, for all action outcomes. The reward is 10 in the state *On* and 1 in the state *Off*.

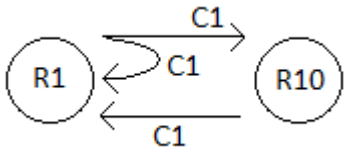

Figure 2.4: Example of a MDP - Costs and Rewards.

2.3 Goal Types

The basic planning problem is having a transition system, an initial state, and a single goal state. Instead of a single goal state there might be a set of goal states. Adding a labeling function[1] to the transition system allows expressing *Extended Goals*.

[1]The labeling function maps each state to a set of true atomic propositions. Note that other schemes are possible with respect to the labeling function. If for instance it is needed to represent partial observability of

Using a Single Goal State

The simple situation of having one goal state works fine for the lighting example from earlier. The goal state could for instance be the state *On*. Other transition systems might however have more states that should be considered as goals. If these states are not considered as goals, solutions are overlooked when planning.

Using a Set of Goal States

Defining more goal states improves planning because the problem is reduced to only reaching one of these states. More goal states does, as mentioned, also ensure that possible solutions are not overlooked.

Using Extended Goals

Extended Goals are formulae written in for instance CTL. Besides having the property of being able to describe multiple goal states, it is also possible to express path requirements and continued reachability goals. These concepts and the more expressive plans that are necessary will be described in section 2.5 on the next page and section 2.6 on page 13. Initially planning problems will be looked at in section 2.4.

2.4 Planning Problems

There are different definitions of planning problems depending on which transition system and goal type is used. The deviations are minor but for the sake of clarification let the definitions be as given in the following paragraphs.

As mentioned earlier the classical planning domain can be represented as a *Deterministic Transition System*. Given a *Deterministic Transition System* the *Classical Planning Problem* is then defined as follows:

Definition 2.4.1 *(Classical Planning Problems) - A Classical Planning Problem is a triple* $\Sigma = (M, s_0, S_g)$*, where:*

- *M is a finite deterministic state transition system.*
- s_0 *is the initial state, a member of S.*
- S_g *is the set of goal states, a subset of S.*

Note that S_g can be a singleton.

Example:
M could for example be the DTS from figure 2.1 on page 7, s_0 could be the state *Off* and S_g could be the set $\{On\}$.

labels, the labeling function might map states to both propositions and negations of propositions. In case a proposition or its negation is not part of a state the truth value of the proposition is assumed to be unknown.

The *Simple Planning Problem* can be defined as follows:

Definition 2.4.2 *(Simple Planning Problems) - A Simple Planning Problem is a triple* $\Sigma = (M, s_0, S_g)$*, where:*

- *M is a finite non-deterministic state transition system (take for example either a NDTS or a PTS).*
- s_0 *is the initial state, a member of S.*
- S_g *is the set of goal states, a subset of S.*

Note that S_g can once again be a singleton.

Example:
M could for example be the NDTS from figure 2.2 on page 8, s_0 could be the state *Off* and S_g could be the set $\{On\}$.

The *Extended Planning Problem* can be defined as follows:

Definition 2.4.3 *(Extended Planning Problems) - A Extended Planning Problem is a triple* $\Sigma = (M, s_0, \phi_g)$*, where:*

- *M is a finite labeled transition system (take either a DTS, a NDTS, or a PTS and add a labeling function L).*
- s_0 *is the initial state, a member of S.*
- ϕ_g *is the extended goal expressed in CTL or Probabilistic Computation Tree Logic (PCTL) in case the labeled transition system is a PTS.*

Compared to *Simple Planning Problems* the noticeable change is the requirement of transition systems having a labeling function and having an *Extended Goal* instead of having a goal state or a set of goal states.

Example:
M could for example be the labeled NDTS in figure 2.5 on page 13, s_0 could be the state *S3*, and ϕ_g could be the formula seen in equation 2.5 on page 14.

2.5 Solution Types

For the different planning problems there are different types of solutions. Some of the types of solutions can arguably be used for more than one type of planning problem. That said the solution types are in general used for the planning problems as follows:

For a *Classical Planning Problem* as given by definition 2.4.1 on the facing page a

solution is a finite sequence of actions known as a plan or a *Sequential Plan*. For the finite sequence of actions $a_1, a_2, a_3, \ldots, a_n$ to be a solution, it holds that:

$$\gamma(\gamma(\ldots\gamma(\gamma(s_0, a_1), a_2), \ldots, a_{n-1}), a_n) \in S_g \tag{2.2}$$

For a *Simple Planning Problem* as given by definition 2.4.2 on the previous page a solution is on the form of *Policies*, see definition 2.5.1.

Definition 2.5.1 *(Policy) - For a given transition system (S, A, ...) a Policy is a mapping* $\Pi : S \rightarrow A$. *That is* $\forall\ s \in S$, $\Pi(s)$ *returns an action* $a \in A$.

For an *Extended Planning Problem* as given by definition 2.4.3 on the previous page a solution is on the form of *Plans with Execution Contexts*, see definition 2.5.2.

Definition 2.5.2 *(Plan with Execution Contexts) - For a given transition system (S, A, ...) a Plan with Execution Contexts is a tuple (C,* c_0*, act, ctxt), where:*

- *C is a finite set of execution contexts.*
- c_0 *is the initial execution context.*
- *act:* $S \times C \rightarrow A$ *is the action function.*
  act(s,c) gives the next action to take ($a \in A$) *when in the given state s and execution context c.*
- *ctxt:* $S \times C \times S \rightarrow C$ *is the context transition function.*
  ctxt(s,c,s') takes the previous state s, previous execution context c, and the new current state s' and returns the new execution context ($c' \in C$) *the system should transition to.*

For all of these different solutions the term plan will be used interchangeably. This is a more liberal use than other literature where the term only covers classical plans (*Sequential Plans*).

The solutions provided by *Policies*/*Sequential Plans* can be divided into different degrees of strength. *Strong*, *Strong Cyclic*, and *Weak*. For a *Policy*/*Sequential Plan* to be *Strong* every possible path taken from the initial state(s) should lead to a goal state. Furthermore the plan must not contain any loops for it to be a *Strong Solution*.

Strong Cyclic Solutions are *Strong Solutions* where loops are allowed. *Strong Cyclic Solutions* always succeed if there is some kind of fairness on the transitions so that the loops eventually are exited. Note by definition you can not have a *Strong Cyclic Solution* when you have a *Sequential Plan* because *Sequential Plans* do not handle loops.

Weak Solutions only guarantee that there exists at least one path for which the goal will be reached. Note that you can only have *Strong Solutions* if you have a DTS as is the case of *Classical Planning Problems* (see definition 2.4.1 on page 10). *Sequential Plans* are however also used in other places like for example *Epistemic Planning*.

The different definitions on the strength of solutions can be found in [4] as well as the definition of *Policies* and *Sequential Plans*.

Plans with Execution Contexts are solutions provided all the sub goals (of the *Extended Goal*) are fulfilled.

2.6 Extended Goals and Computation Tree Logic

This section will clarify the functionality of *Extended Goals* and how they are described in CTL. The section is not meant to give a (complete) description of CTL. For a full description of CTL (and PCTL) see [6].

For experimenting with the functionality of *Extended Goals* the example transition system needs to be a bit more complex than the previous described lighting example.

Let the planning domain be that of the *Simple PacMan Game* shown in figure 2.5. PacMan is one of the often used examples within *Artificial Intelligence* (see for instance [7]) although the specific transition system is unique. In the example we do not need partial observability of atomic propositions. Therefore only true propositions are listed for states. The labels left out are assumed false.

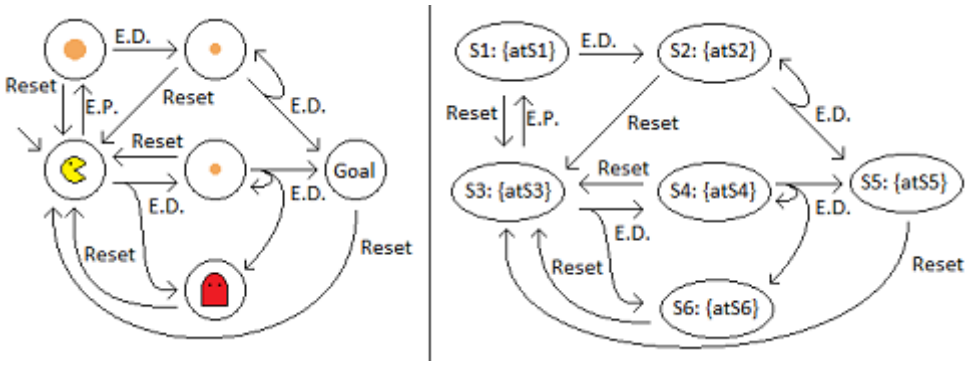

Figure 2.5: Simple PacMan Game - Overview and labeled NDTS.

From the initial state with the yellow *PacMan* there is a choice of either eating a pellet (*E.P.*) or a dot (*E.D.*). If the pellet is eaten *PacMan* becomes invulnerable to the ghosts and can therefore from then on eat the dots with out fear. Eventually after eating enough dots the goal state will be reached (if some kind of fairness is assumed).

If the pellet is not eaten initially, *PacMan* is still vulnerable to the ghosts. He might succeed in eating all the dots, but he might also be caught by the ghosts before reaching the goal.

From any state it is possible to reset the game using the action *Reset*.

When *Progressing CTL* the meaning of CTL goals deviate a bit from how CTL is normally perceived in the model checking community. The following paragraphs elaborate on the use of CTL for progression by giving an example of an *Extended Goal* and explaining it.

In natural language one example of a continued reachability goal could be, *"Reach the goal and then reset the game, continuously "*. This *Extended Goal* can be described in CTL. The corresponding CTL property:

$$\text{AG}\ (\text{AF}\ \mathit{atS5} \wedge \text{AF}\ \mathit{atS3}) \tag{2.3}$$

No plan can however fulfill the property specified in equation 2.3 for the given planning domain. The reason is the formulation of the property and the non-deterministic outcome of the action *E.D. (Eat Dot)*.

The for all operator in CTL has the strict meaning of, *"all possible paths"*. Although it is extremely unlikely (given enough time) the agent might keep looping without ever reaching the state *S5* as intended. *"All paths possible"* corresponds to a *Strong Solution*.

Therefore the goal expressed in CTL can not be satisfied for the given planning domain. The sub goal AF*atS5* will not hold because it does not hold for any plan that the state *S5* will eventually be reached on all paths possible. There can only be found a *Strong Cyclic Solution*.

Reformulating the goal in equation 2.3 gives an expression that can be satisfied. The reformulated description of the goal, in natural language, would then be, *"Try reaching the goal and then reset the game, continuously "*. The reformulated CTL goal can be seen in equation 2.4. *"There exists a path"* or EF corresponds to a *Weak Solution*.

$$\text{AG}\ (\text{EF}\ \mathit{atS5} \wedge \text{AF}\ \mathit{atS3}) \tag{2.4}$$

The new sub goal EF*atS5* is satisfiable.

For the *Simple PacMan Game* we would like to express the path requirement, "*PacMan should never be caught by ghosts*". This can be expressed in CTL as AG $\neg \mathit{atS6}$.

The combined CTL expression can be seen in equation 2.5.

$$\text{AG}\ (\text{EF}\ \mathit{atS5} \wedge \text{AF}\ \mathit{atS3}) \wedge \text{AG}\ \neg \mathit{atS6} \tag{2.5}$$

For this combined goal a plan could be found as the one described in table 2.1 on the facing page.

State	Context	Action	Next state	Next context
S3	c1	E.P.	S1	c1
S1	c1	E.D.	S2	c1
S2	c1	E.D.	S2	c2
S2	c1	E.D.	S5	c2
S2	c2	Reset	S3	c1
S5	c2	Reset	S3	c1

Table 2.1: Simplified representation of a plan Π using *Execution Contexts*.

The simplified graphical representation can be converted to a plan on the form (C, c_0, $act, ctxt$). When moving out of state two the context is changed to $c2$, no matter if we reach state two or state five. The plan does therefore not ensure that state five is ever reached.

The *Strong Cyclic* goal of reaching a state where *atS5* holds could be formulated by using the until operator. Combined with the other sub goals the expression would be as follows:

$$\text{AG (A [EF } atS5 \cup\ atS5] \wedge \text{AF } atS3) \wedge \text{AG } \neg atS6 \tag{2.6}$$

Progressing the formula in equation 2.6 could give a *Strong Solution* if the *Simple PacMan Game* was defined otherwise.

2.7 Planning

Figure 2.6 on the following page shows an overview of how planning is usually done for the different transition systems and goal types. The type of solution is also noted in the figure.

Initial states have not been included in the figure although they are required for the suggested planning approaches. Even in the case of MDP's and Policy Iteration, where an initial state would not be used for planning purposes, the initial state is used. This is because choosing an action, when following a policy, requires that the current state is known. In general, when following a policy, action outcomes are fully observable because else the next action can not be looked up.

The planning methods and how they work will be looked into in the following sub-sections.

2.7.1 Exploration of the Search Space

Exploration of the search space can be done by a search algorithm like for instance *Breadth First Search*. In case there are non-deterministic choices the strength of the solution varies in degree depending on the found path(s) to the goal state(s). See *Strong, Strong-Cyclic*, or *Weak Solutions* in section 2.5 on page 11.

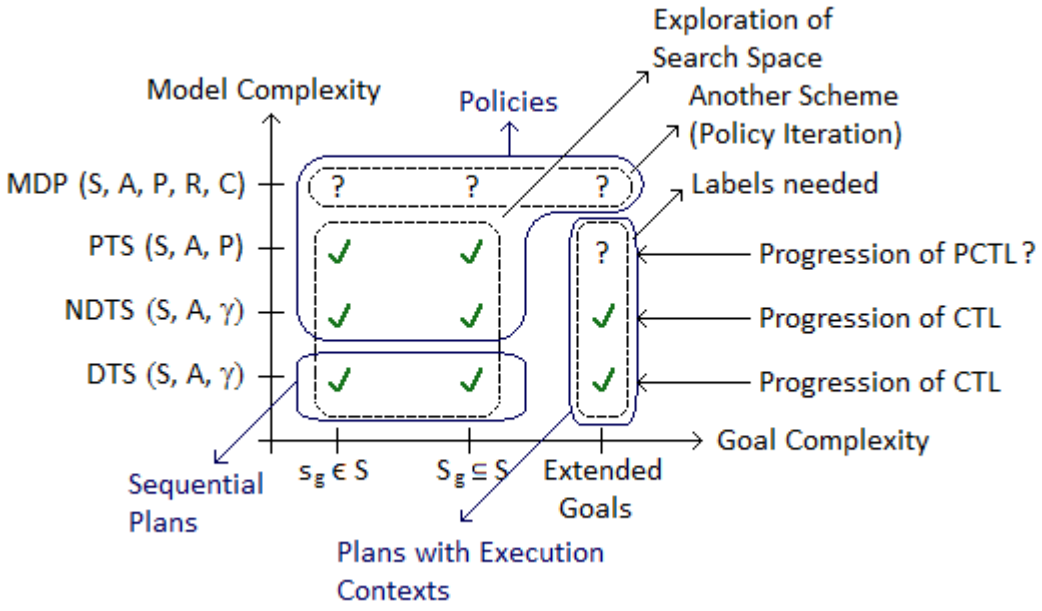

Figure 2.6: Planning Overview.

2.7.2 Progression

Progression of CTL is described in [4]. The underlying principal being that the goal formula is either satisfied in the current state, or the goal is progressed and satisfied in the next state.

In case of a goal formula containing sub goals, as with the CTL formula in equation 2.5, each sub goal is solved by *Progression of CTL*. The plan for each sub goal is then associated with an execution context. This gives a plan on the form of definition 2.5.2.

For a complete description of planning with execution contexts check [4]. Other approaches for planning with execution contexts are also covered in [4].

Progression of PCTL has not been explored as a planning approach, but it should be a reasonable assumption that *Progression of PCTL* can be done in much the same way as with *Progression of CTL*.

PCTL has been used for model checking and there are known approaches for implementing a PCTL model checker. Given a PTS and a *policy*, PCTL model checking can be used to check the PCTL properties in the same way CTL properties could have been checked for a NDTS and a *policy*.

2.7.3 Planning Based on Markov Decision Processes

MDP's and *Policy Iteration* is a bit outside the scheme with respect to the other planning approaches because there are no notions of goal states or goal formulae (as given in *Extended Goals*).

When planning for a MDP the rewards and costs could however just be ignored, and planning could be done in the same way as with PTS's.

The normal approach for planning based on MDP's is however using the rewards and costs for estimating utility. Algorithms that optimizes utility, like for instance *Policy Iteration*, are then used to determine the optimal *policy*.

When using utility no guarantees can be made before a *policy* is found regarding for instance state avoidance or state reachability goals. That said by assigning costs and rewards intelligently many of the *Extended Goals* can be simulated even though they can not be guaranteed beforehand.

Some continued reachability goals can however not be solved when using *policies*. In case the full expressiveness of *Extended Goals* is needed, using *Progression of PCTL* and *Plans with Execution Contexts* is the viable approach. Rewards and costs are, as mentioned, irrelevant in this case.

Utility

Utility can be defined as reward minus cost given the cost and reward functions previously described in definition 2.2.4 on page 9. The estimated average utility of executing an action $\Pi(s)$ for a state s can then be defined recursively by:

$$V(s, \Pi(s)) = \left(R(s) - \sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot C_{s'}(s, \Pi(s)) \right) + \left(\sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot V(s', \Pi(s')) \right) \tag{2.7}$$

The left part of the above equation is the reward of the current state s subtracted the estimated average cost of doing the action $\Pi(s)$. The summation summarizes the probability of doing a transition times the cost of doing a transition for all the possible transitions of the action $\Pi(s)$.

The right part of the equation is the estimated average utility of the subsequent states. That is the states following s after action $\Pi(s)$.

The equation does not usually converge to a finite value. Therefore it is necessary to introduce a discount factor γ. Where $0 < \gamma < 1$. Introducing γ gives us:

$$V(s, \Pi(s)) = \left(R(s) - \sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot C_{s'}(s, \Pi(s)) \right) + \gamma \cdot \left(\sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot V(s', \Pi(s')) \right) \tag{2.8}$$

Ensuring that the value converges to a finite value is not the only reason for introducing a discount factor. It can also be argued that it makes sense because distant rewards and costs should have decreased importance for a state. The further steps away the less influential the step should be considered in respect to a state.

Instead of finding the estimated average utility of a state and an action recursively solving the equation system $V(S, \Pi)$ will give the estimated average utility of all state action pairs.

Solving the planning problem can now be seen as the problem of finding actions such that the utility is optimal. There are different algorithmic approaches for finding the optimal plan (optimizing utility).

Policy Iteration

Policy Iteration is one approach of optimizing utility. The algorithm is described in pseudocode in figure 2.7 on the next page.

Initially the plan Π is set to the empty plan (in line 2). Then the plan Π' is initialized to a random plan (in line 3). The *while loop* is then entered where the plan Π is iteratively improved until no changes occur. When no improvements can be found, the optimal plan Π is returned [4].

The body of the *while loop* is the step suggesting an improvement to the plan. The body first calculates the estimated average utility for every state s and action $\Pi(s)$ in line 7. Which is the same as solving the equation system $V(S, \Pi)$.

Then all states s are traversed in line 9. If there exists an action that causes a better utility than previous, then this action is chosen as the new suggested action for $\Pi'(s)$. The *while loop* is escaped when all suggestions are the same as the previous plan ($\Pi = \Pi'$).

Example:

Reusing the *PacMan* example let the MDP M be described as shown in figure 2.8. Assume action costs are one on all transitions. The algorithm starts by selecting a random plan. This could for instance be the plan Π_1:

Policy Iteration(M, γ)
$\Pi \leftarrow \emptyset$
select any $\Pi' \neq \emptyset$
while $\Pi \neq \Pi'$ **do**
  $\Pi \leftarrow \Pi'$
  for all $s \in S$ **do** {Find the solution to the system of equations $V(S, \Pi)$}
    $V(s, \Pi(s)) \leftarrow$

$$\left(R(s) - \sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot C_{s'}(s, \Pi(s))\right)$$

$$+\gamma \cdot \left(\sum_{s' \in S} P_{\Pi(s)}(s, s') \cdot V(s', \Pi(s'))\right)$$

  end for
  for all $s \in S$ **do**
    if $\exists\, a \in A$ s.t.

$$V(s, \Pi(s)) < \left(R(s) - \sum_{s' \in S} P_a(s, s') \cdot C_{s'}(s, a)\right)$$

$$+\gamma \cdot \left(\sum_{s' \in S} P_a(s, s') \cdot V(s', \Pi(s'))\right)$$

    then
      $\Pi'(s) \leftarrow a$
    else
      $\Pi'(s) \leftarrow \Pi(s)$
    end if
  end for
end while
return Π

Figure 2.7: Policy Iteration

$$
\begin{aligned}
\Pi_1 = \{&(S1, Reset),\\
&(S2, E.D.),\\
&(S3, E.D.),\\
&(S4, E.D.),\\
&(S5, Reset),\\
&(S6, Reset)\}
\end{aligned}
$$

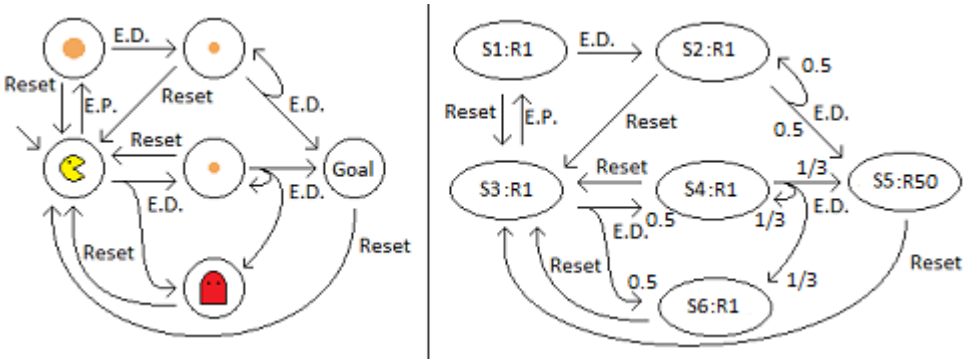

Figure 2.8: Simple PacMan Game - Overview and MDP (costs not included).

The system of equations V(S,Π_1) is then solved:

$$
\begin{aligned}
V(S1, Reset) =& R(S1) - P_{Reset}(S1, S3) \cdot C_{S3}(S1, Reset) +\\
&\gamma \cdot (P_{Reset}(S1, S3) \cdot V(S3, E.D.)\\
V(S2, E.D.) =& R(S2) - (P_{E.D.}(S2, S2) \cdot C_{S2}(S2, E.D.) + P_{E.D.}(S2, S5) \cdot C_{S5}(S2, E.D.)) +\\
&\gamma \cdot (P_{E.D.}(S2, S2) \cdot V(S2, E.D.) + P_{E.D.}(S2, S5) \cdot V(S5, Reset))\\
V(S3, E.D.) =& R(S3) - (P_{E.D.}(S3, S4) \cdot C_{S4}(S3, E.D.) + P_{E.D.}(S3, S6) \cdot C_{S6}(S3, E.D.)) +\\
&\gamma \cdot (P_{E.D.}(S3, S4) \cdot V(S4, E.D.) + P_{E.D.}(S3, S6) \cdot V(S6, Reset))
\end{aligned}
$$

$$
\begin{aligned}
V(S4, E.D.) =& R(S4) - (\ P_{E.D.}(S4, S4) \cdot C_{S4}(S4, E.D.) + P_{E.D.}(S4, S5) \cdot C_{S5}(S4, E.D.) +\\
&\quad P_{E.D.}(S4, S6) \cdot C_{S6}(S4, E.D.)) +\\
&\gamma \cdot (\ P_{E.D.}(S4, S4) \cdot V(S4, E.D.) + P_{E.D.}(S4, S5) \cdot V(S5, Reset) +\\
&\quad P_{E.D.}(S4, S6) \cdot V(S6, Reset))\\
V(S5, Reset) =& R(S5) - P_{Reset}(S5, S3) \cdot C_{S3}(S5, Reset) +\\
&\gamma \cdot (P_{Reset}(S5, S3) \cdot V(S3, E.D.))\\
V(S6, Reset) =& R(S6) - P_{Reset}(S6, S3) \cdot C_{S3}(S6, Reset) +\\
&\gamma \cdot (P_{Reset}(S6, S3) \cdot V(S3, E.D.))
\end{aligned}
$$

Assuming that γ is 0.9 then:

$$
\begin{aligned}
V(S1, Reset) =& 1 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S3, E.D.)) \\
V(S2, E.D.) =& 1 - (0.5 \cdot 1 + 0.5 \cdot 1) + 0.9 \cdot (0.5 \cdot V(S2, E.D.) + 0.5 \cdot V(S5, Reset)) \\
V(S3, E.D.) =& 1 - (0.5 \cdot 1 + 0.5 \cdot 1) + 0.9 \cdot (0.5 \cdot V(S4, E.D.) + 0.5 \cdot V(S6, Reset)) \\
V(S4, E.D.) =& 1 - \left(\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1\right) + \\
& 0.9 \cdot \left(\frac{1}{3} \cdot V(S4, E.D.) + \frac{1}{3} \cdot V(S5, Reset) + \frac{1}{3} \cdot V(S6, Reset)\right) \\
V(S5, Reset) =& 50 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S3, E.D.)) \\
V(S6, Reset) =& 1 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S3, E.D.))
\end{aligned}
$$

The corresponding matrix equation, $\mathbf{A} \cdot \mathbf{X} = \mathbf{B}$:

$$
\begin{pmatrix}
1 & 0 & -0.9 & 0 & 0 & 0 \\
0 & 0.55 & 0 & 0 & -0.45 & 0 \\
0 & 0 & 1 & -0.45 & 0 & -0.45 \\
0 & 0 & 0 & 0.7 & -0.3 & -0.3 \\
0 & 0 & -0.9 & 0 & 1 & 0 \\
0 & 0 & -0.9 & 0 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
V(S1, Reset) \\
V(S2, E.D.) \\
V(S3, E.D.) \\
V(S4, E.D.) \\
V(S5, Reset) \\
V(S6, Reset)
\end{pmatrix}
=
\begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 49 \\ 0
\end{pmatrix}
$$

The solution:

$$
\begin{aligned}
V(S1, Reset) \approx& 34.3141 \\
V(S2, E.D.) \approx& 68.1661 \\
V(S3, E.D.) \approx& 38.1268 \\
V(S4, E.D.) \approx& 50.4121 \\
V(S5, Reset) \approx& 83.3141 \\
V(S6, Reset) \approx& 34.3141
\end{aligned}
$$

It can be seen that Π_1 is not the optimal plan. When reaching line 9 it is possible to find actions that improve the utility of the plan. The algorithm does therefore not terminate at this step.

Indeed the algorithm continues until the optimal plan $\Pi_{Optimal}$ is found:

$$
\begin{aligned}
\Pi_{Optimal} = \{&(S1, E.D.), \\
&(S2, E.D.), \\
&(S3, E.P.), \\
&(S4, E.D.), \\
&(S5, Reset), \\
&(S6, Reset)\}
\end{aligned}
$$

With utility:

$$
\begin{aligned}
V(S1, E.D.) =& 1 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S2, E.D.)) \\
V(S2, E.D.) =& 1 - (0.5 \cdot 1 + 0.5 \cdot 1) + 0.9 \cdot (0.5 \cdot V(S2, E.D.) + 0.5 \cdot V(S5, Reset)) \\
V(S3, E.P.) =& 1 - (1 \cdot 1) + 0.9 \cdot (1 \cdot V(S1, E.D.)) \\
V(S4, E.D.) =& 1 - \left(\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1\right) + \\
& 0.9 \cdot \left(\frac{1}{3} \cdot V(S4, E.D.) + \frac{1}{3} \cdot V(S5, Reset) + \frac{1}{3} \cdot V(S6, Reset)\right) \\
V(S5, Reset) =& 50 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S3, E.P.)) \\
V(S6, Reset) =& 1 - 1 \cdot 1 + 0.9 \cdot (1 \cdot V(S3, E.P.))
\end{aligned}
$$

$\rightarrow$

$$
\begin{aligned}
V(S1, Reset) \approx& 89.4120 \\
V(S2, E.D.) \approx& 99.3467 \\
V(S3, E.D.) \approx& 80.4708 \\
V(S4, E.D.) \approx& 83.0775 \\
V(S5, Reset) \approx& 121.4237 \\
V(S6, Reset) \approx& 72.4237
\end{aligned}
$$

Remarks:

Value Iteration is another approach for optimizing utility. The algorithm, and a short comparison of the two approaches, can be found in the appendix in section A on page 57.

2.8 Epistemic Planning

Epistemic Planning is another approach for modeling and planning for uncertainty. The definitions in this section are from the journal, *"Epistemic planning for single- and multi-agent systems"* [8]. The section has primarily drawn inspiration from [8] and partially from [9].

2.8.1 The Language

Given a finite set of atomic propositions P and a finite set of agents A, the language $L_K(P, A)$ is generated by the following *Backus Naur Form*:

$$\phi ::= \top | \bot | p | \neg\phi | \phi \wedge \phi | K_i\phi \tag{2.9}$$

$\top$ is always, $\bot$ is never, $p \in P$, and $i \in A$. ϕ can either be a compound proposition or a atomic proposition as shown above. $K_i\phi$ intuitively describes that agent i knows ϕ.

2.8.2 Epistemic Models

Epistemic Models are used when defining *Epistemic States*.

Definition 2.8.1 *(Epistemic Models) - An Epistemic Model is a triple M = (W, R, V), where:*

- *W is a finite set of worlds.*
- $R : A \rightarrow 2^{W \times W}$ *is the indistinguishability relation.*
  In case (w, v) ∈ R(a) then agent a can not distinguish between the worlds w and v. This is also denoted wR_av.
- *V:* $P \rightarrow 2^W$ *is the valuation function.*
  Each atomic proposition $p \in P$ *is mapped to a set of worlds. Intuitively every atomic proposition is mapped to every world where the proposition is true.*

Every relation $R_a \in R$ is an equivalence relation. This simply means that R_a partitions the set of worlds W such that every $w \in W$ is a member of one, and only one, partition. It also means that for any $R_a \in R$ it is given that the relation is reflexive, symmetric, and transitive.

The properties reflexivity, symmetry, and transitivity are demonstrated below.

For all $w_1, w_2, w_3 \in W$, and all $a \in A$, it holds that:

$w_1R_aw_1$ (reflexivity)

if $w_1R_aw_2$ then $w_2R_aw_1$ (symmetry)

if $w_1R_aw_2$ and $w_2R_aw_3$ then $w_1R_aw_3$ (transitivity)

2.8.3 Epistemic States

The reason we have been using the term worlds instead of states becomes apparent given the following definition of states:

Definition 2.8.2 *(Epistemic States) - An Epistemic State is a pair (M,* W_d*), where:*

- *M is an Epistemic Model.*
- W_d *is the set of designated worlds.*

Consider the state $(M, \{w\})$. The set $\{w\}$ is a singleton only containing w, and therefore the state is a global state of the domain of M. This view of the world is considered to be modeled by an omniscient external observer. Agents are however rarely omniscient external observers.

The global state can be seen from an agents local point of view as the state (M, W_d).

Here W_d is the non empty set of designated worlds $W_d \subseteq W$. Intuitively W_d is the set of worlds the specific agent considers possible. For a given agent $a \in A$ the state is derived as $(M, \{v|wR_av\})$. Where $\{v|wR_av\}$ is all the worlds that the agent can not distinguish from the actual world (including of course the actual world on account of reflexivity).

2.8.4 Event Models

Event Models are used when defining *Epistemic Actions*.

Definition 2.8.3 *(Event Models) - An Event Model is a tuple ε = (E,Q,pre,post), where:*

- *E is the set of events.*
- *$Q : A \rightarrow 2^{E \times E}$ is the indistinguishability relation.*
  In case $(e_1, e_2) \in Q(a)$ then agent a can not distinguish between the events e_1 and e_2. This is also denoted $e_1 Q_a e_2$. All the indistinguishability relations in Q are once again equivalence relations as described for Epistemic Models.
- *pre: $E \rightarrow L_K(P, A)$ is a function mapping events to preconditions. The preconditions are conjunctions of atomic propositions and negations of atomic propositions.*
- *post: $E \rightarrow L_K(P, A)$ is a function mapping events to postconditions. The postconditions are also conjunctions of atomic propositions and negations of atomic propositions.*

2.8.5 Epistemic Actions

Epistemic Actions are defined in much the same way as with *Epistemic States*.

Definition 2.8.4 *(Epistemic Actions) - An Epistemic Action is a pair (ε, E_d), where:*

- *ε is an Event Model.*
- *E_d is a set of designated events.*

Consider the *Epistemic Action* $(\varepsilon, \{e\})$ composed of the event model ε and the singleton $\{e\}$. This is a global view of the action assumed to be made by an external omniscient observer. The local view of an *Epistemic Action* can again be defined as:

$$(\varepsilon, E_d) \tag{2.10}$$

Here the local view of the *Epistemic Action* by an agent a is once again constructed as $(\varepsilon, \{e'|eQ_ae'\})$. Intuitively $E_d \subseteq E$ and E_d is the set of events the agent can not distinguish from the actual event taking place. The actual event taking place is once again included in the set of events on the account of reflexivity.

2.8.6 Product Update of a State with an Action

The product update of a state with an action is defined as follows:

Definition 2.8.5 *(Product Update of a State with an Action) - A Product Update of a State (M, W_d) with an Action (ε, E_d) where M = (W,R,V) and ε = (E,Q,pre,post) is denoted $(M, W_d) \otimes (\varepsilon, E_d)$. The resulting state $((W', R', V'), W'_d)$ is defined as follows:*

- $W' = \{(w, e) \in W \times E | M, w \models pre(e)\}$
- $R'_i = \{((w, e), (v, f)) \in W' \times W' | wR_iv \text{ and } eQ_if\}$
- $V'(p) = (\{(w, e) \in W' | M, w \models p\} - \{(w, e) \in W' | post(e) \models \neg p\}) \cup \{(w, e) \in W' | post(e) \models p\}$.
- $W'_d = \{(w, e) \in W' | w \in W_d \text{ and } e \in E_d\}$

2.8.7 Epistemic Planning Domain

The *Epistemic Planning Domain* can be described like classical planning domains with a restricted state transition system [8]. The definition can be seen below.

Definition 2.8.6 *(Epistemic Planning Domains) - An Epistemic Planning Domain can be described by a tuple $\Sigma = (S, A, \gamma)$, where:*

- *S is a finite or recursively enumerable set of epistemic states of $L_K(P, A)$.*
- *A is a finite set of actions of $L_K(P, A)$.*
- *γ is defined as:*

$$\gamma(s, a) = \begin{cases} s \otimes a & \text{if } a \text{ is applicable in } s \\ undefined & \text{otherwise} \end{cases} \tag{2.11}$$

2.8.8 Epistemic Planning Problems

Epistemic planning problems can be defined in much the same way as classical planning problems.

Definition 2.8.7 *(Epistemic Planning Problems) - An Epistemic Planning Problem is a triple (Σ, s_0, ϕ_g), where:*

- *Σ is an epistemic planning domain on (P, A).*
- *s_0 is the initial state, which is a member of S.*
- *ϕ_g is the goal formula. The set of goal states $S_g = \{s \in S | s \models \phi_g\}$.*

2.8.9 Planning

Given an *Epistemic Planning Problem*, planning can be done as with classical planning. The problem can for instance be addressed as a *Breadth First Search*. In the same manner as with a classical planning domain the actions are applied to the initial state in a breadth first order. Given enough time the goal formula will eventually be entailed by the attained state if it is possible to construct a plan for the domain. Then extracting the *Sequential Plan* is simply backtracking the actions taken.

The plan found would then be a sequence on the same form as the following example sequence:

$s_0 \otimes a_1 \otimes a_2 \otimes a_3 \ ... \otimes a_n \models \phi_g$

Example:

Let figure 2.9 represent the *Epistemic State* $s0$.

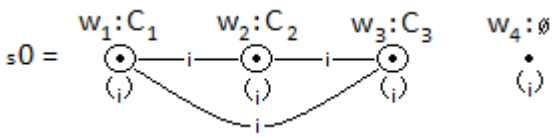

Figure 2.9: Representation of the epistemic state $s0$.

There are three atomic propositions C_1, C_2, C_3. It can be seen, as an external observer, that either one of the propositions are true ($w_1 - w_3$), or they are all false (w_4). Note that propositions not listed in a world as true are assumed false in this graphical representation of the *Epistemic State* s0.

The state is seen from agent i's point of view. In state $s0$ agent i knows that the actual world is either w_1, w_2, or w_3. Agent i does not know which world is the actual and further more finds the three worlds indistinguishable. The designated worlds ($w_1 - w_3$) are marked with $\odot$, and the world w_4, which agent i does not consider possible, is marked with $\bullet$.

The indistinguishable worlds $w_1 - w_3$ are connected by lines. Each line is marked by the agent that considers the connected worlds indistinguishable (in this case i). Note that arrowheads are omitted from the lines because the relations are bidirectional on account of symmetry.

The knowledge of agent i in state $s0$ is given as shown in equation 2.12.

$$\neg K_i C_1 \wedge \neg K_i \neg C_1 \wedge \neg K_i C_2 \wedge \neg K_i \neg C_2 \wedge \neg K_i C_3 \wedge \neg K_i \neg C_3 \tag{2.12}$$

For simplicity the impossible worlds (w_4) are normally omitted from the graph and model. The redundant indistinguishability relations are also normally omitted. These are the relations that can be derived given the property that the indistinguishability relation is transitive (line from w_1 to w_3) and reflexive (self loops).

The normal graphical representation of $s0$ can be seen in figure 2.10

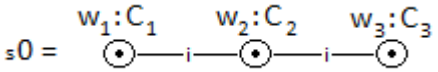

Figure 2.10: Simplified representation of the epistemic state $s0$.

Now let us assume that there are two actions available for the agent. Action $a1$ in figure 2.11 and action $a2$ in figure 2.12.

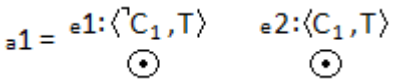

Figure 2.11: Representation of the epistemic action $a1$.

Action $a1$ has two events where the event $e1$ has $\neg C_1$ as precondition, and the event $e2$ has C_1 as precondition. Both events are considered equally possible (they are both marked with $\odot$). Both events do not change the truth values of any propositions (post-conditions = $\top$). Finally it is seen that the two events are distinguishable from each other during execution for any agent (they are not connected by any lines).

Action $a1$ is a sensing action telling the agent whether C_1 is true or not. Action $a2$ is a sensing action telling the agent if C_2 is true or not in the same way as with action $a1$.

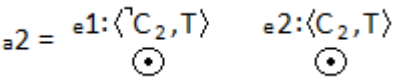

Figure 2.12: Representation of the epistemic action $a2$.

Given the two actions $a1$ and $a2$, the initial state $s0$, and the goal formula shown in equation 2.13, a plan can be found. Such a plan could for instance be $s0 \otimes a1 \otimes a2$.

The goal formula should be considered in contrast to the knowledge modeled by the initial state $s0$ seen in equation 2.12 on the preceding page. The difference being that agent i knows the truth values of the propositions in the goal state, and agent i has no knowledge about the truth values of the propositions in the initial state.

The plan suggests first updating the state $s0$ with action $a1$ as shown in figure 2.13.

$$(K_iC_1 \vee K_i\neg C_1) \wedge (K_iC_2 \vee K_i\neg C_2) \wedge (K_iC_3 \vee K_i\neg C_3) \tag{2.13}$$

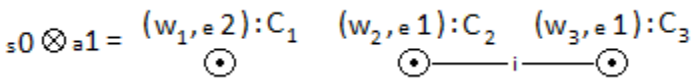

Figure 2.13: Representation of the state $s0 \otimes a1$.

The plan then suggest updating with $a2$ giving a resulting state as shown in figure 2.14.

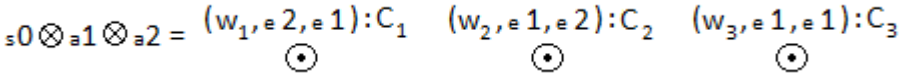

Figure 2.14: Representation of the state $s0 \otimes a1 \otimes a2$.

It is seen in figure 2.14 that agent i can distinguish between the worlds (no worlds are connected). It is also seen that the actual world is either w_1, w_2, or w_3 because agent i considers $w_1 - w_3$ designated. Because none of the designated worlds are indistinguishable with other worlds the actual world will be known by agent i after execution of the plan.

If the actual world is w_1 agent i will know: $K_iC_1 \wedge K_i\neg C_2 \wedge K_i\neg C_3$. If the actual world is w_2 agent i will know: $K_iC_2 \wedge K_i\neg C_1 \wedge K_i\neg C_3$. If the actual world is w_3 agent i will know: $K_iC_3 \wedge K_i\neg C_2 \wedge K_i\neg C_1$.

Agent i will therefore know the truth value of the propositions C_1, C_2, C_3. Therefore $s0 \otimes a1 \otimes a2 \models (K_iC_1 \vee K_i\neg C_1) \wedge (K_iC_2 \vee K_i\neg C_2) \wedge (K_iC_3 \vee K_i\neg C_3)$.

2.8.10 Plausibility

Work is currently being done in the field called *Plausibility Planning*. The work takes inspiration from *Epistemic Planning*. The simplified explanation is that *Plausibility Planning* introduce the concept of preordering. Introducing a preorder of worlds with respect to the *Epistemic States* and a preorder of events with respect to the *Epistemic Actions*.

The agent therefore knows which world it finds most plausible. The agent also knows what event (when doing an action) it considers the most plausible. Consequently the agent knows what world it will find most plausible after doing an action.

This enables the agent to plan with a qualitative approach of evaluating what actions to do. The extra knowledge available might consequently enable an agent to make better plans. The work is still in progress but the idea is to improve on the concepts from *Epistemic Planning* as described.

2.9 Agent Architecture

Depending on the problem in focus there are of course plenty of advanced agent architectures that are applicable. The following simple agent architectures will however, as a simplification, be assumed chosen throughout the report.

2.9.1 Sequential Plans

For the *sequential plan* we assume that the agent does not reason about progress and simply mindlessly execute (or try to execute) the actions as the sequence dictates.

2.9.2 Policies

Policies require the agent to keep track of the current state it is in. Otherwise the next action to take can not be looked up in the *policy*. How the agent keeps track of the current state is in principle irrelevant, but when making a transition (or afterwards) the new state needs to be observed in some way.

2.9.3 Plans with Execution Contexts

The agent is assumed to keep track of the current state and *Execution Context*. Otherwise the next action to do can not be looked up in the plan. How this is done is once again in principle irrelevant. It should however be noted that the *Context Transition Function, ctxt,* maps previous state, previous *Execution Context*, and new state to the new *Execution Context*. So the new state needs to be observed in some way as with *policies*.

2.10 Analysis of Approaches

There are many extensions and alterations to the planning approaches which takes some of their limitations into account. For the following subsections the planning approaches will however only be regarded in the way they are defined in this report. That includes the definitions of planning problems, planning approaches, solutions, and agent architectures.

2.10.1 Partial Observability of States

Table 2.2 shows an overview of the different models and their expressive power.

Transition systems can emulate *Partial Observability of States* by introducing a corresponding *Belief State* (see [3]). Seen from a modeling point of view this is just another state in the transition system. For example can a transition system without a *Labeling Function* emulate partial observability of a implicit *literal* by having an extra state in the transition system where the value of the *literal* is assumed to be unknown i.e. a *Belief State*.

For transition systems with *Labeling Functions* or in the case of *Epistemic States* the *literals* that are partially observable are just made explicit (as are the *literals* that are fully observable).

Models:	Non-Deterministic Outcome of Actions	Can Emulate Partial Observability of States
DTS (S, A, γ)	No	Yes
Labeled DTS (S, A, γ, L)	No	Yes
NDTS (S, A, γ)	Yes	Yes
Labeled NDTS (S, A, γ, L)	Yes	Yes
PTS (S, A, P)	Yes	Yes
Labeled PTS (S, A, P, L)	Yes	Yes
MDP's (S, A, P, R, C)	Yes	Yes
Labeled MDP's (S, A, P, R, C, L)	Yes	Yes
Epistemic States and Event Models	Yes	Yes

Table 2.2: Overview of Expressive Power.

2.10.2 Partial Observability of Action Outcome

Partial Observability of Action Outcome, as it is described in section 2.1.1 on page 4, can be non-deterministic actions where the actual outcome is only partial observable after action execution.

Example:

Figure 2.15 on the next page shows the two possible observations, o_1 and o_2, from turning on the lights from the state *Off*. Here we abstract away from how the observations are actually made and just assume that the o_1 corresponds to observing *On* and o_2 corresponds to observing *Off*.

In case of full observability of the action outcome when doing the action *Turn_On* we get either the observation o_1 (corresponding to transitioning to the state *On*) or the observation o_2 (corresponding to transitioning to the state *Off*).

In case of partial observability of the action outcome when doing the action *Turn_On* we get the set of observations $\{o_1, o_2\}$. This corresponds to observing the *Belief State* {On, Off}.

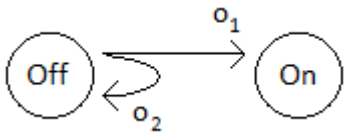

Figure 2.15: Observations - Turning the light On.

Handling Partial Observability of Action Outcome

For a planning approach to handle this kind of *Partial Observability of Action Outcomes* it is therefore required that observations are made when executing actions and making state transitions.

It is also required that these observations can be partial i.e. not knowing completely what state the system has transitioned to.

Of course the observations needs to be taken into account before further action. None of the approaches does this in their raw form (as described in this report).

The overview of the approaches can be seen in table 2.3 on the following page. Following a *policy* requires that transitions are fully observable. Otherwise the next action to take can not be looked up in the policy. Some form of observation is therefore required regarding the state change. The same is true when following a *Plan with Execution Contexts*. When following *Plans with Execution Contexts* there needs to be kept track of the current state and current execution context.

Handling this kind of *Partial Observability of Action Outcomes* could for instance be done by having a probability distribution over states and updating the distribution based on the action taken and the observations made (probabilistic approach). This of course requires that the method of choosing appropriate action is updated so it handles probability distributions over states instead of just having a current state.

Another approach is having an ordering of states and updating the ordering of states based on the action taken and the observations made (Qualitative Approach).

Planning Approach and Solution Type	Use Observations	Observations can be Partial
Exploration of Search Space and Sequential Plans	No	-
Exploration of Search Space and Policies	Yes	No
Policy Iteration and Policies	Yes	No
Progression and Plans with Execution Contexts	Yes	No
Epistemic Planning and Sequential Plans	No	-

Table 2.3: Partial Observability of Action Outcomes.

2.10.3 Likelihood of Action Outcome

Knowledge about the likelihood of an action outcome enables a more informed choice when planning. One approach is assigning probabilities on action outcomes (Probabilistic Approach). Table 2.4 shows an overview of the models expressive power in this respect. Another approach is having a pre-order of action outcomes (Qualitative Approach). None of the models use pre-orders (or other orders).

Models:	Assigns Probabilities on Outcomes
DTS (S, A, γ)	No
Labeled DTS (S, A, γ, L)	No
NDTS (S, A, γ)	No
Labeled NDTS (S, A, γ, L)	No
PTS (S, A, P)	Yes
Labeled PTS (S, A, P, L)	Yes
MDP's (S, A, P, R, C)	Yes
Labeled MDP's (S, A, P, R, C, L)	Yes
Epistemic States and Event Models	No

Table 2.4: Likelihood of Action Outcome.

2.10.4 Time and Quantities

The planning approaches all discretize time so continuous time can not be handled. The approaches either abstract away from durations on actions, or all actions are assumed to take one time unit. The approaches can not model durations on effects.

None of the approaches can handle continuous quantities. They can all discretize a quantity so long as the quantity and production/consumption of the quantity can be represented by a finite number of states (different amounts of the quantity).

2.10.5 Exogenous Events

The approaches does not handle exogenous events as they are described in this report.

Chapter 3

Domain Analysis

This chapter will introduce the reader to the *DTUsat2* satellite and some of the planning problems regarding *DTUsat2*. The chapter also contains an analysis of one or more of the planning problems. Results from the manual test executions of the planning approaches will be outlined, and observations will be made. The planning approaches that will be tested are, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL* and *Epistemic Planning*."

3.1 DTUsat2

For a full description of the *DTUsat2 Student Project* see *Solving the mystery of bird migration - Tracking small birds from space* in [10] and *DTUsat Mission Overview* in [11].

Mission Description

The mission of the DTU student satellite is to track from space the migration of small birds.

The cuckoo bird normally lays its eggs in the nests of other birds thus leaving the fostering of its chicks to other species of birds. When the (grown) chicks decide to migrate from northern Europe to Africa it is therefore without having met their parents.

Figure 3.1 on the following page shows an example of the migratory routes.

How these birds manage to navigate flawlessly several thousand kilometers alone and in the night is still unclear. The *DTUsat2 Student Project* is supposed to help scientist understand this phenomenon better by tracking the migratory routes of the birds.

Figure 3.1: Example of migratory routes.

3.2 DTUsat2 Planning Problems

There are several planning problems regarding *DTUsat2*. For the following selected planning problems dealing with uncertainty is paramount to correct planning and operation within the domain.

3.2.1 Charging the Battery

The battery must only be charged when it is between 0 °C and 45 °C. This constraint was defined because there exists a risk of the battery exploding if charging at too low or too high temperatures.

There are several temperature sensors on *DTUsat2*. There are however only three that measure the temperature on the battery. They each measure independently of each other. They are of different type so they have different confidence intervals.

The conditions the sensors work under might cause the sensors to return wrong temperature measurements now and then. The sensors might also break down completely.

For enabling recharging of the battery more often a heater is placed on the battery. Power consumption of the heater should be kept at a minimum so the power can be used by other parts of the satellite. Heating the battery to an acceptable temperature for recharging should not be done if the power generated from the solar panels is too small.

Taking all these different aspects into consideration, recharging can be seen as an interesting planning problem (*the Battery Recharging Planning Problem - BRPP*) where there are both critical consequences and less critical consequences of recharging the battery.

3.2.2 Attitude Control Subsystem

The *Attitude Control Subsystem* is responsible for orienting the satellite correctly towards earth. The side of the satellite, on which the antenna is located, should be pointed

towards earth.

The satellite transmits a beacon of selected housekeeping data every 30 seconds. The ground station should be able to receive this data no matter what orientation the satellite has. Likewise the satellite needs to be able to receive commands from the ground station independently of the satellite orientation.

The *Communication Subsystem* is however also able to enter a high rate data mode upon instruction from the ground station. This high rate data mode will be able to send more detailed housekeeping data, and/or data from the PPL (Primary Payload) and PICOCAM (see [11]).

Data sent in the high rate data mode can only be received by the ground station when the satellite is oriented towards earth correctly. The ground station also needs to use a high gain antenna (directional antenna) for it to receive data in high rate data mode.

These limitations are present because more data can be transmitted for the same amount of energy. The *Attitude Control Subsystem* is therefore critical for the mission success. Part of the *Attitude Control Subsystem* is a 3-axis magnetometer, three magnetic torquers, a sun sensor board, and a deployable gravity gradient boom. These are some of the available sensing capabilities and means of influence.

Besides this there is a sensor that measures if the gravity gradient boom has been released. The time taken for the gravity gradient boom to release is known to vary. The *Attitude Control Subsystem* should shift state from detumbling to dampening when the gravity gradient boom has been deployed. The latter of the two states is the less power consuming state.

There are also heat sensors attached to the sides of the satellite so it is possible to sense which side of the satellite is approximately pointing towards earth and which sides are pointing out into space.

The Planning Problem

Deploying the gravity gradient boom will be outlined as a planning problem (*the Gravity Gradient Boom Deployment Planning Problem - GGBDPP*) in the following paragraphs:

Based on the measurements from the 3-axis magnetometer the satellite's rotation should be slowed using the magnetic torquers (detumbling). Then the heat sensors and the sun sensor board should be used to determine if the satellite is approximately pointing towards earth.

If the orientation of the satellite is acceptable the gravity gradient boom should be deployed so stabilization can be achieved passively. With out powering the magnetic torquers too much at least.

The 3-axis magnetometer, the heat sensors, and the sun sensor board return different confidence intervals on different measurements. Once again the values might not always be returned correctly because of the working conditions. There also exists a risk that the equipment breaks down or is already broken.

The gravity gradient boom should be deployed when the orientation of the satellite is correct, else the satellite might be stabilized pointing the wrong way.

When the gravity gradient boom is deployed the attitude control system should change state to save power. The deployment of the gravity gradient boom is known to vary in time. If necessary the satellite should be flipped if the satellite stabilized pointing away from earth.

3.3 Applicability of Planning Approaches

The planning approaches, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL*, and *Epistemic Planning*," should be able to model and plan for the BRPP and the GGBDPP given the following simplifying assumptions:

- Observations as they are described in section 2.10.2 on page 30 can not be partial.
- Time can be discretized and action durations assumed to take one time unit. There is no need to have durations on effects.
- There is no need to represent continuous quantities.
- *Exogenous Events* can be abstracted away from during planning and plan execution.

3.4 Extended Battery Recharging Planning Domain

The *Extended Battery Recharging Planning Domain* (EBRPD) is inspired from the description of the BRPP from section 3.2.1 on page 36.

3.4.1 Specification of the EBRPD

The general specification of the EBRPD abstracts away from power production and consumption. Heating the battery is simply assumed to always be a viable choice.

Let there be given two atomic propositions *OK* and *Charging*. The atomic proposition *OK* describes if the temperature is acceptable for recharging or not. The atomic proposition *Charging* describes if the system is recharging or not.

Let the actions available be *SenseTemp*, *StabilizeTemp*, *BeginCharging*, and *StopCharging*.

The action *SenseTemp* is assumed to change the truth value of the atomic proposition *OK* if the environment has changed temperature.

Assume that the action *StabilizeTemp* tries to adjust the heater on the battery so that the temperature gets acceptable for recharging (tries to make the atomic proposition *OK* true).

The action *StabilizeTemp* is able to fail in stabilizing the temperature if the action is applied when the temperature is unacceptable for recharging (if *OK* is false initially it might remain that way).

The result of executing the action *StabilizeTemp* is assumed to be unknown when applying the action in a state where the temperature is unacceptable for recharging.

The action *BeginCharging* starts recharging of the battery (the proposition *Charging* gets true). It is fully observable and the action is assumed to always succeed when the action is applicable. The action is only applicable when the battery is not charging (*Charging* is false).

The action *StopCharging* stops recharging of the battery (the proposition *Charging* gets false). It is fully observable and the action is assumed to always succeed when the action is applicable. The action is only applicable when the battery is charging (*Charging* is true).

3.4.2 Models of EBRPD and Example Runs

The complete documentation of the models and the example runs can be found in the appendix in section B.3 on page 71.

Policy Iteration - Model

Figure 3.2 on the following page and figure 3.3 on the next page are the graphical representation of a MDP representing the EBRPD.

Note that the actions are abbreviated in the following way, "*SenseTemp* = ST, *StabilizeTemp* = STB, *BeginCharging* = BC and *StopCharging* = SC". Strictly speaking MDP's do not contain information regarding atomic propositions. The combinations of the atomic propositions are used for naming the states instead. This will make later comparison of models easier. In the case of (OK $\vee$!OK) the truth value of OK is unknown i.e. it is not meant as a tautology but just used to visualize that there is no knowledge of the truth value of the atomic proposition OK.

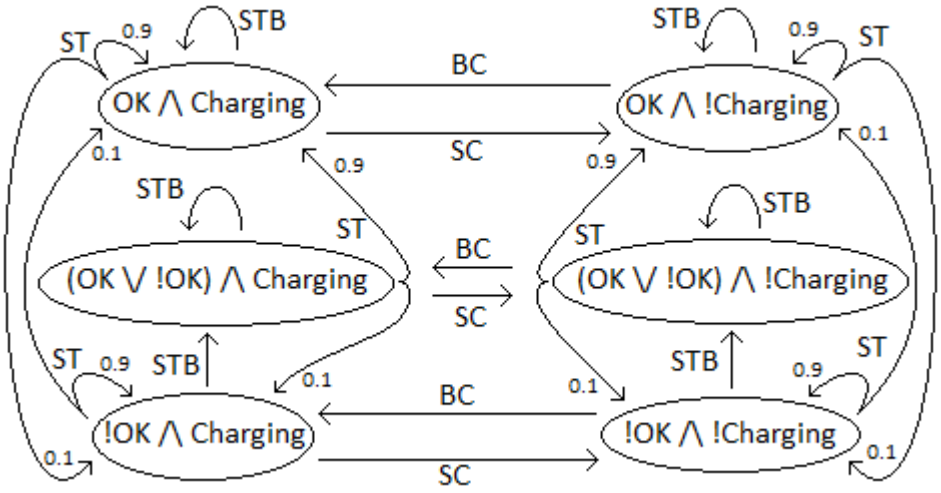

Figure 3.2: Charging Battery - Probabilities and Actions.

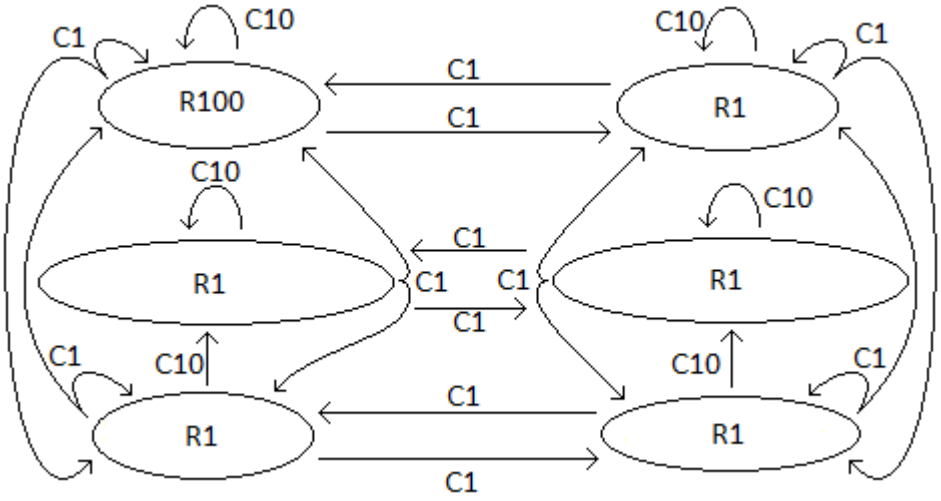

Figure 3.3: Charging Battery - Costs and Rewards.

Policy Iteration - Solution

The *policy* Π_2 seen below has been found by the *Policy Iteration Algorithm* where γ was assumed to be 0.9.

$$\begin{aligned}\Pi_2 = \Big\{&(''OK \wedge Charging'', StabilizeTemp),\\ &(''OK \wedge \neg Charging'', BeginCharging),\\ &(''(OK \vee \neg OK) \wedge Charging'', SenseTemp),\\ &(''(OK \vee \neg OK) \wedge \neg Charging'', SenseTemp),\\ &(''\neg OK \wedge Charging'', StabilizeTemp),\\ &(''\neg OK \wedge \neg Charging'', StabilizeTemp)\Big\}\end{aligned} \tag{3.1}$$

Progession of CTL - Model

Figure 3.4 on the following page shows a representation of the EBRPD modeled by a NDTS.

Labels are omitted from the figure although state names are made so they correspond to labels. The same notations and abbreviations used in the MDP is used here.

Let the labeling function L be given as:

$$\begin{aligned}L(''OK \wedge Charging'') &= \{OK, Charging\}\\ L(''OK \wedge \neg Charging'') &= \{OK, \neg Charging\}\\ L(''(OK \vee \neg OK) \wedge Charging'') &= \{Charging\}\\ L(''(OK \vee \neg OK) \wedge \neg Charging'') &= \{\neg Charging\}\\ L(''\neg OK \wedge Charging'') &= \{\neg OK, Charging\}\\ L(''\neg OK \wedge \neg Charging'') &= \{\neg OK, \neg Charging\}\end{aligned} \tag{3.2}$$

Because of the need to have partial observable propositions there is no closed world assumption. That means both positive and negative propositions needs to be part of the labeling function. If both the positive and negative version of a proposition is omitted there is no knowledge present about the truth value of the proposition. This is the case for the atomic proposition OK in the state $''(OK \vee \neg OK) \wedge \neg Charging''$ and the state $''(OK \vee \neg OK) \wedge Charging''$.

Progession of CTL - Goal and Solution

The goal is the CTL formula shown in equation 3.3.

$$\begin{aligned}\text{AG}\ (\text{A}\,(\text{EF}\,(OK \wedge Charging) \cup OK \wedge Charging) \wedge \text{AF}\ (OK \wedge Charging))\\ \wedge\ \text{AG}\ (OK \vee \neg Charging)\end{aligned} \tag{3.3}$$

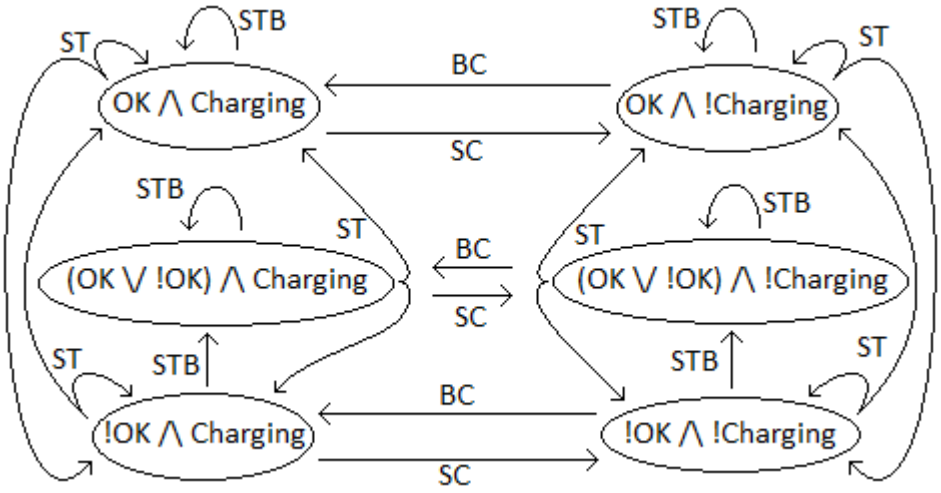

Figure 3.4: Transition System with Labels Omitted.

Progressing the CTL subformulas gives the plan shown in table 3.1 if "$(OK \vee \neg OK) \wedge \neg Charging$" is assumed to be the initial state.

State	Context	Action	Next state	Next context
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge \neg Charging$"	c1	BeginCharging	"$OK \wedge Charging$"	c2
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge Charging$"	c2	StabilizeTemp	"$OK \wedge Charging$"	c1
"$OK \wedge Charging$"	c1	StabilizeTemp	"$OK \wedge Charging$"	c2

Table 3.1: Plan Found by Progressing the CTL Formula in Equation 3.3.

Epistemic Planning - Model

The *Epistemic Planning Domain* Σ is a model of the EBRPD. It can be represented graphically as shown in figure 3.5.

Note that some lines are dashed. This is simply a visual aid telling that the action not necessarily leads to the exact same *Epistemic State*, but that the action leads to an *Epistemic State* which is bisimilar to the one the dashed arrow points to. Therefore, depending on the exploration path, dashed arrows might need to be exchanged with normal arrows and vice versa if *Epistemic States* are drawn as they are explored.

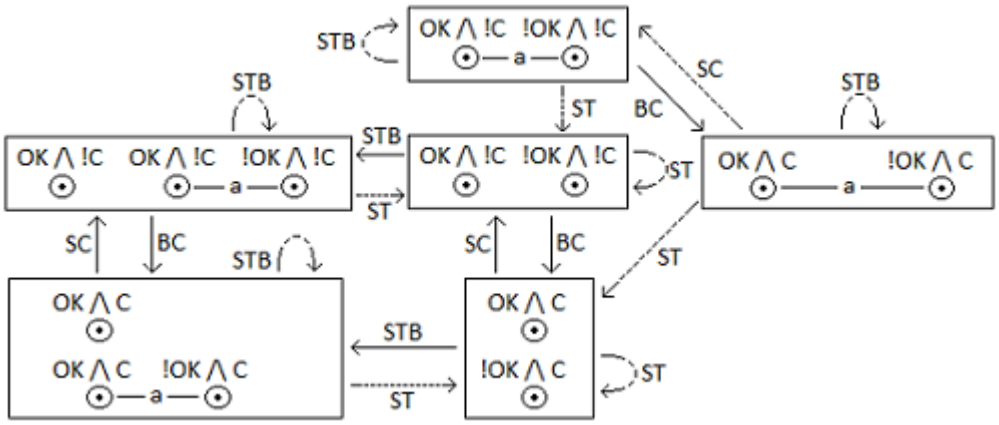

Figure 3.5: Epistemic Planning Domain - EBRPD.

Epistemic Planning - Problems and Solutions

The example *Epistemic Planning Problem* using the previous described *Epistemic Planning Domain*:

Problem One

- Σ = figure 3.5.
- s0 = figure 3.6.
- $\Phi_g = OK \wedge Charging$.

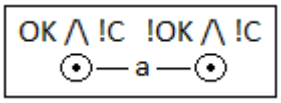

Figure 3.6: Problem One - Initial Epistemic State.

For *ProblemOne* there are no strong solutions. Assuming that actions will be explored in a breadth first manner and chosen in alphabetical order the first weak solution that is found is $s0 \otimes BC \otimes ST$.

The term weak solution means that not all of the designated worlds of the reached state (and the partitions they are part of) entails the goal formula, but at least one designated world does. Where a strong solution means that all the designated worlds (and the partitions they are part of) entails the goal formula. At least when defined for the single agent case.

3.4.3 Observations

The following observations are made with respect to the previous described manual test executions.

Observations - Policy Iteration

The goal was reaching the state "$OK \wedge Charging$" and staying there. This was captured quite efficiently by the way the costs and rewards were modeled.

The *policy* found was acceptable. It would have been better if the state "$!OK \wedge Charging$" and the state "$(OK \vee !OK) \wedge Charging$" was mapped to *StopCharging* as a safety precaution. That said at least the states "$!OK \wedge Charging$" and "$!OK \wedge !Charging$" was not mapped to the action *SenseTemp*. The probabilities on the transitions ensured that this was not the case.

Hoping that the temperature will get back in the accepted area by itself might be a possibility, but it is a fair assumption that this is not considered efficient enough.

Observations - Progression of CTL

The plan found was acceptable but not optimal. The problem is that from the state "$!OK \wedge !Charging$" the action *SenseTemp* is picked in hopes of the temperature normalizing by itself. This should be considered in contrast to picking *StabilizeTemp* where the temperature gets stabilized by the satellite.

Without probabilities (or some other measure of describing the most likely state change) it is not possible to know how often the temperature will change on its own when calling the action *SenseTemp*.

The plan found does however guarantee that it is impossible to reach a state where there is a risk of the battery exploding. This was achieved by the formulation of the goal. More generally the planning approach guarantees that the complete extended goal, as given by the CTL formula, will hold if a plan is found.

Observations - Epistemic Planning

The continued goal of reaching a state where $OK \wedge Charging$ holds makes Epistemic Planning a bit unsuited in it's simplest form. That said the actual reached world will be known after execution of the plan. Measures like for instance replanning could therefore be implemented to make up for the failed attempts at reaching a state where $OK \wedge Charging$ holds.

The plan might lead to a world where the battery is recharging when the temperature is outside the acceptable bounds.

The planning approach insures that one of the designated worlds models the formula

$OK \wedge Charging$ if a plan is found. It can therefore be proven that there is a chance that the correct world will be reached before planning is done.

3.5 Other Domain Examples

Besides the EBRPD experiments with other planning domains have been made. The problem regarding deployment of the gravity gradient boom has for instance also been investigated. The different models and test executions of the planning approaches can be found in the appendix. See section B.2 on page 62 and section B.4 on page 82.

Chapter 4

Recommendation

This chapter will make a recommendation of planning approach for the EBRPD based on both the domain analysis and the previous comparison of the approaches. The recommendation is based on the three tested approaches, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL* and *Epistemic Planning*."

4.1 Analysis of Best Practice for the EBRPD

There are several pros and cons of using the different planning approaches with respect to the EBRPD.

4.1.1 Consideration of the Likelihood of Action Outcomes

The probabilities used in the MDP ensured that the action *SenseTemp* was not chosen in situations where waiting for the temperature to change on its own was not viable. This is of course an advantage compared to the other approaches that does not consider the likelihood of action outcomes.

Regarding assignment of the probabilities in the MDP it should be said that the assignments has not been made on empirical collected data or some other source of statistical information. The probabilities are therefore most likely wrong.

It could be argued that in cases where the probabilities are unknown it does not make sense to pick the MDP approach to model the domain. This of course depends on the worst case deviation when estimating probabilities. Some of the probabilities might be known even though they have not been used in the testing process. In any case, assuming that the actual deviations are small, there is a clear advantage of not picking actions with a small possibility of success.

Although sensing the temperature was not chosen at inappropriate points when *Planning Based on MDP's* no guarantees could however be made beforehand. When the

actual planning needs to take place on the satellite the advantage is therefore limited.

4.1.2 Guarantees and Path Requirements

As mentioned one of the disadvantages of using MDP's and *Policy Iteration* is that no guarantees can be made beforehand about reaching a desired state or avoiding a undesired state. When the planning is done on the satellite this is not good enough. For the domain the desired state would be the state where the battery is recharging and the temperature is OK. The undesired states would be the states where the battery is recharging while the temperature might be or is outside the acceptable bounds.

When *Planning for Extended Goals with Progression of CTL* it is possible to make guarantees if a plan is found. For the specific domain no plan can be found where all paths leads to the desired state. The state where the battery is recharging while the temperature is OK. It can however be guaranteed that attempts to reach the state will be made continuously. If there is any kind of fairness it is ensured that the state will eventually be reached. It can also be guaranteed that the undesirable states are never visited.

Epistemic Planning also enables one to make guarantees beforehand. As mentioned no strong plan to reach the desired state can be found for the specific domain. That said, in case a plan is found, it can once again be guaranteed that there is a chance that the reached world will be the desired world. The world where the battery is recharging while the temperature is OK. With intelligent replanning and assumptions of fairness the desired world will eventually be reached. *Epistemic Planning* does not enable path requirements and therefore no guarantees can be made with respect to never visiting undesirable states.

4.1.3 Partial Observability of Action Outcome

The model of the EBRPD has been simplified in order for the planning approaches to be applicable. Most of the simplifying assumptions, like abstracting away from time durations on effects or actions, can arguably be said to have negligible effect.

One of the simplifying assumptions is however that it is not necessary to allow partial observations. There are situations where this is not a reasonable assumption. Take for instance the action of sensing the temperature. There are multiple ways of defining the outcomes of the action. In figure 4.1 on the next page a simple set of possible outcomes and a more complex set is shown.

The two epistemic states shows two different cases of the resulting state after sensing the temperature. The left example is the one used in section 3.4.2 on page 39 (see *Epistemic Planning - Model*). For the left state it is assumed that the action of sensing the temperature tells us that it is either OK to recharge the battery or not OK to recharge the battery. Both cases are assumed to be fully observable.

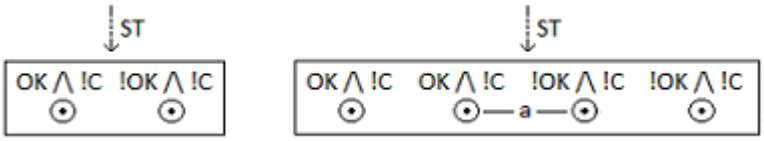

Figure 4.1: Sensing the Temperature - A simple and a more complex set of possible outcomes.

A slightly more reasonable model is the one to the right. There are three different outcomes of the action of sensing the temperature. Either we get the result that the temperature is OK, or the result that it is not OK, or the result that the temperature is still unknown. Thus allowing for the temperature sensing to fail in some way.

In reality the problem of sensing the temperature is however more complex. The right example comes close at modeling the real world sufficiently. This is because when the temperature is measured the value returned can of course be either in a OK region or a not OK region. The sensors are also known to deviate so in case of temperature values just on the border of OK or not OK, the actual temperature region can not be decided (temperature remains unknown).

There is however also the case of the sensor failing because of the environment the sensor is working in, and simply giving a completely wrong value. This value might be so far out of the reasonable range that it is detected. Depending on the modeling approach detected failed attempts at sensing the temperature could simply be ignored leading to the world where the temperature is unknown. The situation might also need to be explicitly pointed out leading to a world where for instance a specific fail proposition is flagged true.

When the temperature sensor fails, the value might however also simply fall within the OK region, or the not OK region, or on the border of the two regions. This situation of the sensor failing can therefore not be distinguished from any of these three situations.

When a value is returned from measuring the temperature, the correct approach is therefore assuming that we have partial observability over the actual outcome. Simply always transitioning to a state where the temperature is unknown does however not capture the knowledge gained by sensing the temperature.

One approach is getting a probability distribution of the temperature regions or possible states that the action sensing the temperature leads to (*Probabilistic Approach*). There are other approaches than the probabilistic for allowing this kind of *Partial Observability of Action Outcomes*. *Partial Observable Markov Decision Processes (POMDP's)* however use the probabilistic approach. Transition systems in general could take ad-

vantage of the approach.

For *Epistemic Planning* a preorder of equivalence classes or worlds might be used to indicate the most likely partition/world in a qualitative way. Using preorders for *Epistemic Planning* is looked into in the article *Plausibility Planning* [9]. The probabilistic approach could of course also be used for *Epistemic Planning* and likewise could the qualitative approach be used for transition systems.

4.1.4 Evaluation and Recommendation

Planning Based on MDP's has an advantage over the other approaches because it is the only approach that models the likelihood of the outcomes as described in section 4.1.1 on page 47.

Epistemic Planning and *Planning for Extended Goals with Progression of CTL* however has the advantage that there can be made guarantees as described in section 4.1.2 on page 48. *Planning for Extended Goals with Progression of CTL* has a slight advantage over *Epistemic Planning* because *Extended Goals* can be expressed and guaranteed. Path requirements, avoidance goals in particular, where relevant for the domain.

The continued goal of recharging was also better captured by *Extended Goals* compared to the *Epistemic Planning Approach*. *Epistemic Planning* only plans for reachability goals.

Assuming that the computational power of the satellite is sufficient for the planning approaches, and assuming that the planner will be implemented on the satellite and the satellite just receives the planning problem, *Planning for Extended Goals with Progression of CTL* will be my recommendation for the EBRPD.

The recommendation is also based on the assumption that a simple representation of the action *SenseTemp* is sufficient. Measuring the temperature can be assumed not to fail if for instance the temperature is sampled several times or other similar fixes.

With the limited computational power available on *DTUsat2* it does not really make sense to implement a general purpose planner on the satellite. Solving the planning problems on the ground station and simply transmitting the solutions to the satellite is however feasible.

This however negates some of the arguments for using *Planning for Extended Goals with Progression of CTL* and might make *Planning Based on MDP's* a viable choice. Desired reachability or avoidance goals can for instance be checked, after planning, using model checking techniques (ensuring sound *Policies*). That said, speculating in alterations or extensions would also allow the approach *Planning for Extended Goals with Progression of CTL* to be improved. The likelihood of action outcomes could for instance be taken into account by using *Progression of PCTL*. The initial recommendation therefore stands.

The *DTU Student Satellite Project* does not only cover *DTUsat2* but evolves satellite construction in general. When the time comes for launching a satellite with more computational power the plan is to implement a general purpose planner. The work done in this thesis can therefore be saved for future use.

Chapter 5

Conclusion

This chapter summarizes the primary points made in the different chapters focusing on the conclusions and the findings made.

5.1 Planning Under Uncertainty

The different areas of uncertainty were clarified and the different planning approaches defined. There was an initial analysis of the different planning approaches. This analysis described the general coverage of the approaches with respect to the different areas of uncertainty.

The planning approaches only had small deviations in advantages and disadvantages. Capturing the likelihood of action outcomes should be mentioned as one of the notable differences between the approaches. The probabilistic modeling approaches can capture the likelihood of action outcomes.

Common for the approaches is that none of them handle partial observations as they are defined in this report.

Given a deeper analysis more deviations might be found. Planning with multiple agents were for instance not looked into. This was mainly because the satellite planning problems were primarily cases of single agent planning.

5.2 Domain Analysis

The satellite and its purpose was outlined. The primary purpose being tracking the migratory routes of cuckoo birds. Then two of the planning problems of the *DTUsat2* satellite were examined.

Taking the planning approaches into consideration the EBRPD was defined. Simplifying assumptions were made compared to the more general description of the planning

problem. These assumptions where necessary for the planning approaches to be applicable.

The three planning approaches, "*Planning Based on MDP's*, *Planning for Extended Goals with Progression of CTL* and *Epistemic Planning,*" was tested on the EBRPD. The most notable observations were:

- The *MDP* captured the likelihood of the outcomes of the action *SenseTemp* and therefore gave better plans.
- No guarantees could be made beforehand when using *MDP's* and *Policy Iteration*. *Epistemic Planning* allowed to make guarantees beforehand on reachability goals. *Planning for Extended Goals with Progression of CTL* allowed for making guarantees beforehand on both reachability and avoidance goals.

5.3 Recommendation

It was found that *Planning for Extended Goals with Progression of CTL* was the recommended approach for planning with respect to the EBRPD. That is of course out of the three tested approaches. Expressing and guaranteeing avoidance of specific states where the primary reason for the approach to be recommended.

The recommendation was of course based on a lot of assumptions. A less simplified planning problem can probably be handled more sufficiently by *Planning for Extended Goals with Progression of PCTL* using an agent architecture that allows partial observations. This was argued in the chapter but should of course be documented by further investigation. *POMDP's* might for instance be an better alternative depending on where the actual planning is done and how the different attributes are weighed.

Bibliography

[1] ***The STRIPS Assumption for Planning Under Uncertainty.***
Authors: Michael P. Wellman.
From: AAAI-90 Proceedings. © 1990, AAAI (www.aaai.org). All rights reserved.

[2] ***The Basics of System Dynamics: Discrete vs. Continuous Modelling of Time.***
Authors: Günther Ossimitz and Maximilian Mrotzek.
Presented at the International System Dynamics Conference 2008, Athens/-Greece.

[3] ***Artificial Intelligence: A Modern Approach.***
Authors: Stuart Russell and Peter Norvig.
Book published by Prentice Hall.

[4] ***Automated Planning: Theory and Practice.***
Authors: Malik Ghallab, Dana Nau and Paolo Traverso.
Book published by Morgan Kaufmann Publishers.

[5] ***Markov Decision Processes.***
Author: M. L. Puterman.
Book published by John Wiley & Sons, Inc., 1994

[6] ***Principles of Model Checking.***
Authors: Christel Baier and Joost-Pieter Katoen.
Book published by the MIT Press.

[7] ***Graph-Based Semantics of the .NET Intermediate Language.***
Author: N.B.H. Sombekke.
Journal published May, 2007.

[8] ***Epistemic planning for single- and multi-agent systems.***
Authors: Thomas Bolander and Mikkel Birkegaard Andersen.
Journal of Applied Non-Classical Logics. Volume 21 - No. 1/2011, pages 9 to 33.

[9] ***Plausibility Planning.***
Authors: Mikkel Birkegaard Andersen, Thomas Bolander, Martin Holm Jensen.

[10] ***Solving the mystery of bird migration: Tracking small birds from space.***
Authors: Kasper Thorup.

[11] ***DTUsat-SRD: System Requirements Document.***
Authors: Jonas Bækby Bjarnø and SYSENG.
Ref: DTUsat_PA_SRD, Issue: NOM-2.00, Date: Feb. 20th 2007, Page:1/141.

Appendix A

Value Iteration

Value Iteration is another approach than policy iteration (seen in figure 2.7 on page 19) to optimize utility. The algorithm is described in pseudocode in figure A.1 on the following page.

The idea behind value iteration is much the same as Policy Iteration. The difference being that value iteration uses old estimates of utility when recalculating utility of an altered plan.

It can be shown that there exists a maximum number of iterations needed to guarantee that Value-Iteration returns an optimal policy [4].

Normally a stopping criterion is used. The policy is returned when the difference of the previous estimated utility and the current estimated utility is below some value ε. For such a plan the found utility does not differ from the optimum utility with more than ε [4].

In the pseudocode in figure A.1 on the next page the old utility values are initialized in the beginning of the algorithm. Then the while loop is entered. The while loop iterates k times and is escaped afterward.

The body of the while loop estimates every utility value of every action in every state. Then the best of these actions are chosen for each state and their utility values are saved as the new ”old estimate” of utility for that state.

For insuring an ε-optimal policy the algorithm should be altered such that instead of iterating k times, the while loop is exited when the margin of difference between two plans are below the threshold ε.

Value Iteration(M,γ)
for all $s \in S$ **do**
    select any initial value $V_0(s)$
end for
$k \leftarrow 1$
while $k <$ maximum number of iterations **do**
    for all $s \in S$ **do**
        for all $a \in A$ **do**

$$V_{temp}(s,a) \leftarrow \left(R(s) - \sum_{s' \in S} P_a(s,s') \cdot C_{s'}(s,a)\right)$$

$$+\gamma \cdot \left(\sum_{s' \in S} P_a(s,s') \cdot V_{k-1}(s')\right)$$

        end for
        $V_k(s) \leftarrow max_{a \in A}\ V_{temp}(s,a)$
        $\Pi(s) \leftarrow \arg\ max_{a \in A}\ V_{temp}(s,a)$
    end for
end while
return Π

Figure A.1: Value Iteration

A.1 Comparison of Policy Iteration and Value Iteration

The two approaches are identical in some aspects.

"They both compute a sequence of plans: Π_1, Π_2, Π_3,...,Π_i,... *and a sequence of sets of values* $V_i(s)$ *for each* $s \in S$*."* [4]

The difference between the two approaches is that value iteration estimates V_i from the value V_{i-1} where as policy iteration calculates the whole equation system V_i every step. Intuitively Value Iteration is computational cheaper each step than Policy Iteration.

Policy Iteration however takes fewer iterations to converge [4]. Deciding on which approach to use should therefore be done either based upon further research or empirical testing.

Appendix B

Example Models and Manual Test Executions

This chapter contains the complete documentation of the example models and manual test executions used in the report. The chapter also contains some extra example models and some extra manual test executions.

B.1 Description of SBRPD

This section contains a specification of the *Simple Battery Recharging Planning Domain* (SBRPD) inspired from the battery recharging planning problem from section 3.2.1 on page 36.

For simplicity let the domain only consist of three different states, *TempUnknown*, *Charging* and *BadTemp*. Assume that the battery is not charging in the states *TempUnknown* and *BadTemp*. Assume that the battery is charging in the state *Charging*.

Given that there is a heater present on the battery assume that the action *StabilizeTemp* tries to adjust the setting on the heater so that the temperature of the battery reaches the accepted temperature for recharging. The action *StablizeTemp* should always result in a transition to the state *TempUnknown*. It is assumed it is always profitable to stabilize the temperature of the battery for recharging.

The action of sensing the temperature (*SenseTemp*) is simplified so it does not return three temperatures or the confidence interval of three temperatures. The action is assumed to give the result that either the temperature is acceptable, the temperature is unacceptable or the action fails and the temperature remains unknown.

If the temperature is acceptable with respect to recharging, the system should transition to the state *Charging*. If the temperature is unacceptable the system should transition to the state *BadTemp*. In case the action *SenseTemp* fails, when in a state where the

temperature has been decided, the previous temperature is assumed to be the correct temperature.

The actions *StablizeTemp* and *SenseTemp* are therefore assumed to be applicable in all states but transitions should be defined as described above.

B.2 Simple Battery Recharging Planning Problem

This section will contain different models of the SBRPD from the section Description of SBRPD on page 61. The section will also contain example test runs of the planning approaches using the models defined.

B.2.1 SBRPD Represented as a MDP

This sub section defines a MDP modeling the SBRPD.

Beginning with the formal representation, let the states be:

$$S = \{TempUnknown, BadTemp, Charging\} \tag{B.1}$$

Let the actions be:

$$A = \{StabilizeTemp, SenseTemp\} \tag{B.2}$$

Let the transition probabilities be:

$$\begin{aligned} P_{action}(s_{from}, s_{to}) = \Big\{ & \Big(P_{StabilizeTemp}(BadTemp, TempUnknown) = 1\Big), \\ & \Big(P_{StabilizeTemp}(Charging, TempUnknown) = 1\Big), \\ & \Big(P_{StabilizeTemp}(TempUnknown, TempUnknown) = 1\Big), \\ & \Big(P_{SenseTemp}(TempUnknown, TempUnknown) = 0.1\Big), \\ & \Big(P_{SenseTemp}(TempUnknown, Charging) = 0.8\Big), \\ & \Big(P_{SenseTemp}(TempUnknown, BadTemp) = 0.1\Big), \\ & \Big(P_{SenseTemp}(Charging, BadTemp) = 0.1\Big), \\ & \Big(P_{SenseTemp}(Charging, Charging) = 0.9\Big), \\ & \Big(P_{SenseTemp}(BadTemp, BadTemp) = 0.95\Big), \\ & \Big(P_{SenseTemp}(BadTemp, Charging) = 0.05\Big)\Big\} \end{aligned} \tag{B.3}$$

The transition probabilities listed are the probabilities above zero. The zero probabilities, e.g. $P_{StabilizeTemp}(TempUnknown, Charging) = 0$, are left out.

Let the rewards be:

$R(TempUnknown) = 1, R(BadTemp) = 1, R(Charging) = 100$

Let the costs be:

$$C_{s_{to}}(s_{from}, action) = \Big\{ \Big(C_{TempUnknown}(TempUnknown, StabilizeTemp) = 50\Big), \\ \Big(C_{TempUnknown}(BadTemp, StabilizeTemp) = 1\Big), \\ \Big(C_{TempUnknown}(Charging, StabilizeTemp) = 50\Big), \\ \Big(C_{TempUnknown}(TempUnknown, SenseTemp) = 1\Big), \\ \Big(C_{BadTemp}(TempUnknown, SenseTemp) = 1\Big), \\ \Big(C_{Charging}(TempUnknown, SenseTemp) = 1\Big), \\ \Big(C_{BadTemp}(BadTemp, SenseTemp) = 50\Big), \\ \Big(C_{Charging}(BadTemp, SenseTemp) = 50\Big), \\ \Big(C_{Charging}(Charging, SenseTemp) = 1\Big), \\ \Big(C_{BadTemp}(Charging, SenseTemp) = 1\Big)\Big\} \tag{B.4}$$

Figure B.1 and figure B.2 are the graphical representation of this MDP.

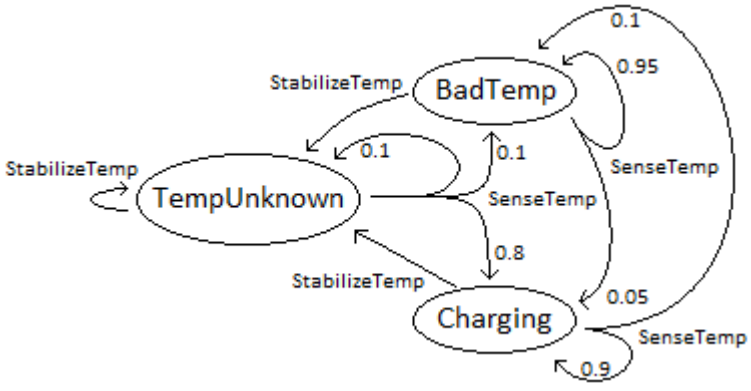

Figure B.1: Charging Battery - Probabilities and Actions.

B.2.2 Test Run of Policy Iteration on SBRPD

This section will give a test execution of *Policy Iteration* on the MDP representing the SBRPD.

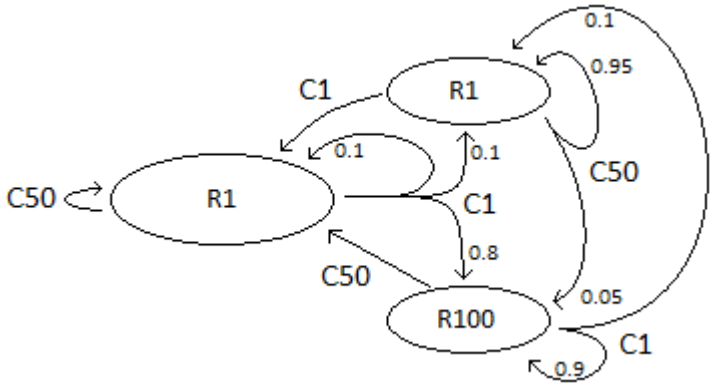

Figure B.2: Charging Battery - Probabilities, Costs and Rewards.

Let the initial policy be given by:

$$\Pi_1 = \Big\{(TempUnknown, StabilizeTemp), \\ (BadTemp, StabilizeTemp), \\ (Charging, StabilizeTemp)\Big\} \tag{B.5}$$

With a value for γ the equation system V_{Π_1} can be solved. Assuming that γ is equal to 0.9 the result will be:

$$V_{\Pi_1}(TempUnknown) = 1 - 1 \cdot 50 + 0.9 \cdot 1 \cdot V_{\Pi_1}(TempUnknown)$$
$$V_{\Pi_1}(BadTemp) = 1 - 1 \cdot 1 + 0.9 \cdot 1 \cdot V_{\Pi_1}(TempUnknown)$$
$$V_{\Pi_1}(Charging) = 100 - 1 \cdot 50 + 0.9 \cdot 1 \cdot V_{\Pi_1}(TempUnknown)$$

$\rightarrow$

$$V_{\Pi_1}(TempUnknown) = -490$$
$$V_{\Pi_1}(BadTemp) = -441$$
$$V_{\Pi_1}(Charging) = -391$$

The algorithm will then suggest the following plan:

$$\Pi_2 = \Big\{(TempUnknown, SenseTemp), \\ (BadTemp, StabilizeTemp), \\ (Charging, SenseTemp)\Big\} \tag{B.6}$$

It can be seen that the action taken from the state *BadTemp* will still be *StabilizeTemp* because it gives the best expected average utility. The chosen actions have changed to *SenseTemp* for the other states. The algorithm then solves the equation system with the new plan:

$$\begin{aligned} V_{\Pi_2}(TempUnknown) &= 1 - 1 + 0.9 \cdot \Big(0.8 \cdot V_{\Pi_2}(Charging) + 0.1 \cdot V_{\Pi_2}(BadTemp) \\ &\qquad + 0.1 \cdot V_{\Pi_2}(TempUnknown)\Big) \\ V_{\Pi_2}(BadTemp) &= 1 - 1 + 0.9 \cdot 1 \cdot V_{\Pi_2}(TempUnknown) \\ V_{\Pi_2}(Charging) &= 100 - 1 + 0.9 \cdot \Big(0.9 \cdot V_{\Pi_2}(Charging) + 0.1 \cdot V_{\Pi_2}(BadTemp)\Big) \end{aligned}$$

$\rightarrow$

$$\begin{aligned} V_{\Pi_2}(TempUnknown) &\approx 719 \\ V_{\Pi_2}(BadTemp) &\approx 647 \\ V_{\Pi_2}(Charging) &\approx 827 \end{aligned}$$

The policy Π_2 is the policy found by the algorithm because no alterations to the policy can be made that improves the expected average utility.

B.2.3 SBRPD Represented as a Transition System with Labels

The transition system representing the SBRPD can be constructed as follows:

$$\Sigma = (S, A, \gamma, L) \tag{B.7}$$

Where the states are given as:

$$S = \{TempUnknown, BadTemp, Charging\} \tag{B.8}$$

The actions are given as:

$$A = \{StabilizeTemp, SenseTemp\} \tag{B.9}$$

The state transition function γ is given as:

$$\begin{aligned}
\gamma(BadTemp, StabilizeTemp) &= \{TempUnknown\} \\
\gamma(Charging, StabilizeTemp) &= \{TempUnknown\} \\
\gamma(TempUnknown, StabilizeTemp) &= \{TempUnknown\} \\
\gamma(TempUnknown, SenseTemp) &= \{TempUnknown, BadTemp, Charging\} \\
\gamma(BadTemp, SenseTemp) &= \{BadTemp, Charging\} \\
\gamma(Charging, SenseTemp) &= \{BadTemp, Charging\}
\end{aligned} \tag{B.10}$$

The labeling function L is given as:

$$\begin{aligned}
L(TempUnknown) &= \{atTempUnknown\} \\
L(BadTemp) &= \{atBadTemp\} \\
L(Charging) &= \{atCharging\}
\end{aligned} \tag{B.11}$$

Figure B.3 shows the transition system with the labels omitted.

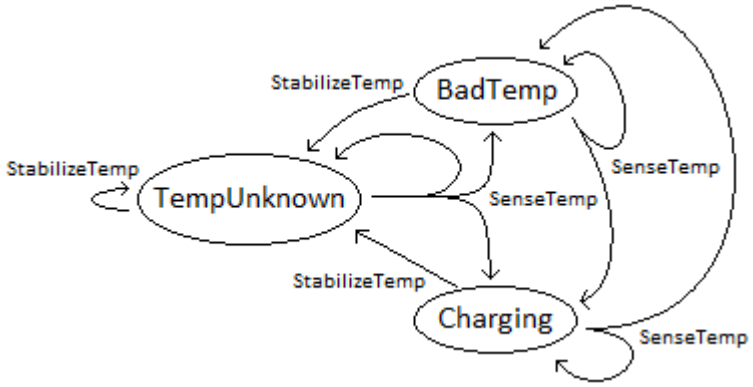

Figure B.3: Transition System with Labels Omitted.

B.2.4 Test Run of Progression of CTL on SBRPD

This section will construct a plan using execution contexts through progression of an example test goal written in CTL.

Let the goal be the CTL formula shown in equation B.12.

$$\text{AG}\ (\text{A}\,(\text{EX}\ atCharging \cup atBadTemp) \wedge (\text{AF}\ atTempUnknown)) \tag{B.12}$$

The first sub goal from equation B.12 is:

$$\text{A}\,(\text{EX}\ atCharging \cup atBadTemp) \tag{B.13}$$

The second sub goal from equation B.12 is:

$$\text{AF}\ atTempUnknown \tag{B.14}$$

Assume that *TempUnknown* is the initial state, the sub goal in equation B.13 is associated with execution context c1 and the sub goal in equation B.14 is associated with execution context c2.

The plan will then be found in the following manner:

Starting from *TempUnknown* it is seen that $atBadTemp \notin \text{L}(TempUnknown)$. Therefore EX *atCharging* needs to be satisfied and $\text{A}\,(\text{EX}\ atCharging \cup atBadTemp)$ has to be satisfied for all the possible successor states.

The applicable actions in *TempUnknown* is *SenseTemp* and *StabilizeTemp*. It is seen that *SenseTemp* is the only action which has the possibility of leading to a state where *atCharging* holds. *SenseTemp* is therefore chosen as the given action to take in the state *TempUnknown* when the execution context is *c1*.

The three possible states *SenseTemp* can lead to from the state *TempUnknown* is: *TempUnknown, Charging* and *BadTemp*.

The execution context should not be altered when transitioning to *TempUnknown* because the progressed goal is the same as the initial goal and the states of course are the same.

The execution context should not be altered when reaching the state *Charging* because it does not satisfy *atBadTemp*. Some action therefore needs to be found such that EX *atCharging* is satisfied and $\text{A}\,(\text{EX}\ atCharging \cup atBadTemp)$ is satisfied in the successor states.

The execution context should however be altered to *c2* when reaching the state *BadTemp* because *atBadTemp* holds in the state.

The partial plan found this far can now be listed as shown in table B.1 on the next page.

The actions applicable in the state *Charging* is *SenseTemp* and *StabilizeTemp*. It is only the action *SenseTemp* that satisfies EX *atCharging*, it is therefore naturally chosen.

State	Context	Action	Next state	Next context
TempUnknown	c1	SenseTemp	Charging	c1
TempUnknown	c1	SenseTemp	TempUnknown	c1
TempUnknown	c1	SenseTemp	Badtemp	c2

Table B.1: Partial Plan.

The two successor states of *Charging* when applying the action *SenseTemp* are the states *Charging* and *BadTemp*. The execution context does not need to be altered for the successor state *Charging* because the same state with the same goal has already been progressed (using execution context *c1*). The execution context can therefore remain *c1*.

When reaching *BadTemp* we once again have to change the execution context because *atBadTemp* is satisfied. Last time the execution context was changed to *c2* when *BadTemp* was reached. This can be done again because both times when reaching *BadTemp* the new goal to be satisfied was AF $atTempUnknown$.

The new partial plan can be seen in table B.2.

State	Context	Action	Next state	Next context
TempUnknown	c1	SenseTemp	Charging	c1
TempUnknown	c1	SenseTemp	TempUnknown	c1
TempUnknown	c1	SenseTemp	Badtemp	c2
Charging	c1	SenseTemp	Charging	c1
Charging	c1	SenseTemp	Badtemp	c2

Table B.2: Extended Partial Plan.

The only goal left to satisfy now is AF $atTempUnknown$ for the state *BadTemp*. There are two applicable actions in *BadTemp*, the action *SenseTemp* and the action *StabilizeTemp*.

Out of the two actions only *StabilizeTemp* ensures that all paths eventually leads to $atTempUnknown$. As it happens the only successor state of *BadTemp* when applying the action *StabilizeTemp* is the state *TempUnknown* where $atTempUnknown$ holds.

The execution context can therefore once again be altered to *c1*. Seeing the initial goal was progressed in the execution context *c1* and the initial goal is the same as the new goal. This completes the plan because all CTL goals have been progressed.

The complete plan can be seen in table B.3 on the next page.

State	Context	Action	Next state	Next context
TempUnknown	c1	SenseTemp	Charging	c1
TempUnknown	c1	SenseTemp	TempUnknown	c1
TempUnknown	c1	SenseTemp	Badtemp	c2
Charging	c1	SenseTemp	Charging	c1
Charging	c1	SenseTemp	Badtemp	c2
BadTemp	c2	StabilizeTemp	TempUnknown	c1

Table B.3: Complete Plan.

B.2.5 SBRPD Represented as an Epistemic Planning Domain

This section will model the SBRPD as an epistemic planning domain.

Given the set of atomic propositions $\{atTempUnknown, atBadTemp, atCharging\}$ and a single agent a let the epistemic model $M = (W, R, V)$ be given by:

$$W = \{TempUnknown, BadTemp, Charging\} \tag{B.15}$$

$$R(a) = \{(TempUnknown, TempUnknown), (BadTemp, BadTemp), (Charging, Charging)\} \tag{B.16}$$

$$\begin{aligned} V(atTempUnknown) &= \{TempUnknown\} \\ V(atBadTemp) &= \{BadTemp\} \\ V(atCharging) &= \{Charging\} \end{aligned} \tag{B.17}$$

Let the action $(StabilizeTemp, \{e_1, e_2, e_3\})$ be abbreviated STB and the action $(SenseTemp, \{e'_1, e'_2, e'_3, e'_4, e'_5, e'_6, e'_7\})$ be abbreviated ST. Let the epistemic actions available be defined as represented graphically in figure B.4 on the following page and figure B.5.

Assume that the state $(M, \{TempUnknown\})$ is the initial state. The epistemic planning domain can be explored through a breadth fist search. The planning domain is represented graphically in figure B.6.

B.2.6 Test Run of Epistemic Planning on SBRPD

The section will give an example of an epistemic planning problem. For the specific planning problem a plan will be found through an example test run of the epistemic

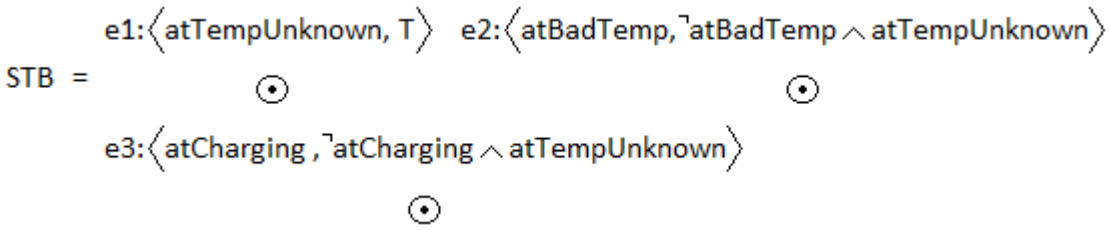

Figure B.4: Epistemic Action STB.

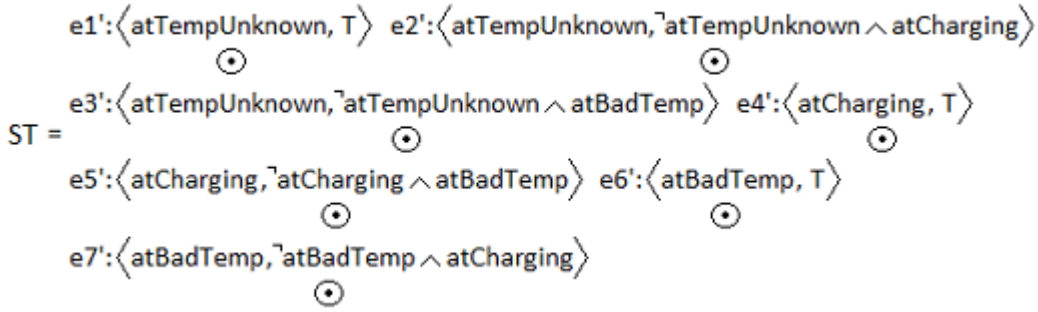

Figure B.5: Epistemic Action ST.

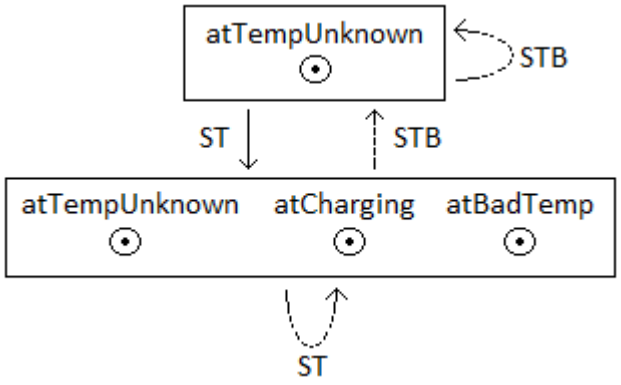

Figure B.6: Epistemic Planning Domain - SBRPD.

planning approach.

Take an epistemic planning domain, an initial state and a goal formula and you have a epistemic planning problem on the form (Σ, s_0, ϕ_g).

Let the problem be given using the previous described epistemic planning domain Σ:

$$Problem\ One = (\Sigma, (M, \{TempUnknown\}), atCharging) \tag{B.18}$$

Solutions can be found through exploration of the planning domain.

For *Problem One* no strong solutions exist. In case the first weak solution that is found is returned the algorithm will return $s0 \otimes ST$. This is because the action of sensing the temperature transitions to an epistemic state where at least one of the designated worlds (and its partition) entails the goal formula (*atCharging*).

B.3 Extended Battery Recharging Planning Problem

This section will define different models of the EBRPD from section 3.4 on page 38. The section will also contain example test runs of the planning approaches using the models defined.

B.3.1 EBRPD Represented as a MDP

The MDP representing the EBRPD has as states combinations of the atomic propositions *OK* and *Charging*.

Strictly speaking MDP's do not contain information regarding atomic propositions. The combinations of the atomic propositions are used for naming the states instead. This will make later comparison of models easier.

Combinations of *OK* and *Charging* is of course not enough seeing we have partial observability of the atomic proposition *OK*. Therefore the MDP also contains belief states that are combinations of $(OK \vee !OK)$ and the atomic proposition *Charging*. That is if $(OK \vee !OK)$ is part of a state name it is not meant as a tautology but just used to visualize that there is no knowledge of the truth value of the atomic proposition OK.

Given an assignment of probabilities, costs and rewards the MDP representing the EBRPD can be constructed as the example described by figure B.7 on the following page and figure B.8 on the next page.

Note that the actions are abbreviated in the following way: *SenseTemp* = ST, *StabilizeTemp* = STB, *BeginCharging* = BC and *StopCharging* = SC.

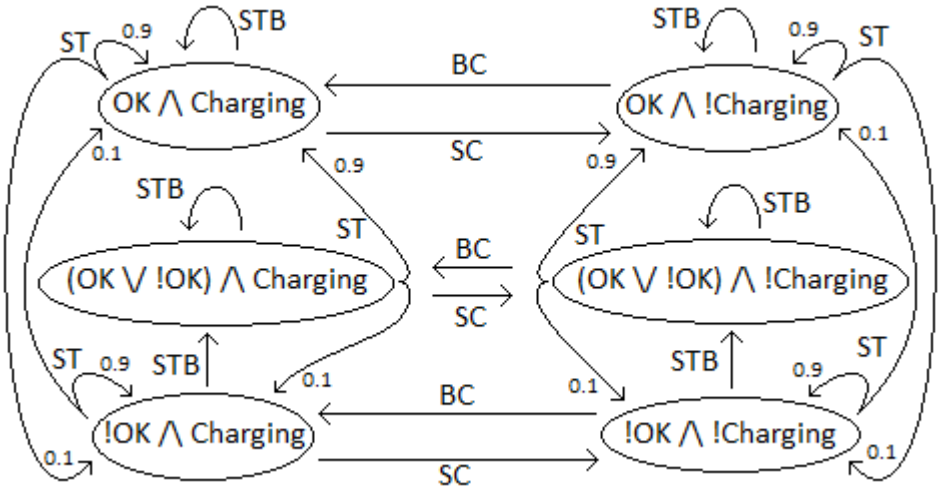

Figure B.7: MDP Representing the EBRPD - Probabilities and Actions.

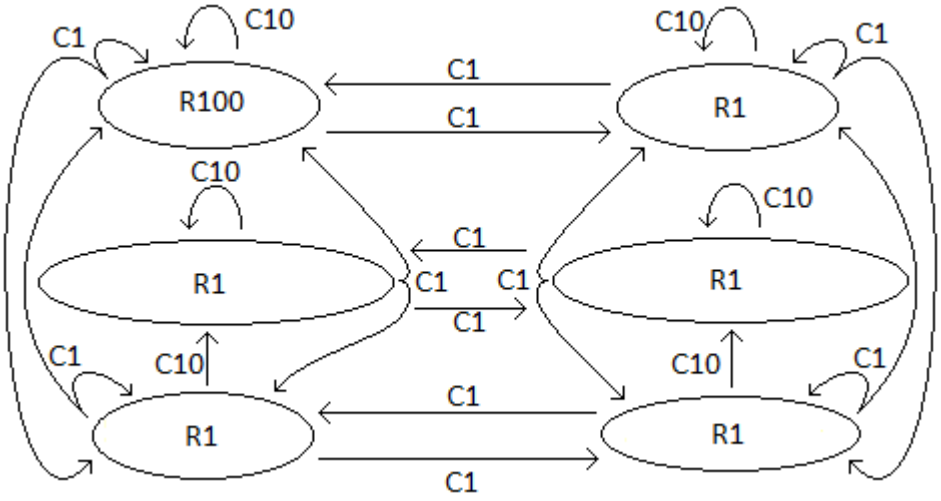

Figure B.8: MDP Representing the EBRPD - Costs and Rewards.

B.3.2 Test Run of Policy Iteration on EBRPD

This section will give a test execution of *Policy Iteration* on the MDP representing the EBRPD.

Let the initial policy be given by:

$$\begin{aligned}\Pi_1 = \Big\{&(''OK \wedge Charging'', StabilizeTemp),\\ &(''OK \wedge \neg Charging'', BeginCharging),\\ &(''(OK \vee \neg OK) \wedge Charging'', StopCharging),\\ &(''(OK \vee \neg OK) \wedge \neg Charging'', SenseTemp),\\ &(''\neg OK \wedge Charging'', StopCharging),\\ &(''\neg OK \wedge \neg Charging'', StabilizeTemp)\Big\}\end{aligned} \tag{B.19}$$

With a value for γ the equation system V_{Π_1} can be solved. Assuming that γ is equal to 0.9 the result will be:

$$\begin{aligned}V_{\Pi_1}(''OK \wedge Charging'') &= 100 - 1 \cdot 10 + 0.9 \cdot 1 \cdot V_{\Pi_1}(''OK \wedge Charging'')\\ V_{\Pi_1}(''OK \wedge \neg Charging'') &= 1 - 1 \cdot 1 + 0.9 \cdot 1 \cdot V_{\Pi_1}(''OK \wedge Charging'')\\ V_{\Pi_1}(''(OK \vee \neg OK) \wedge Charging'') &= 1 - 1 \cdot 1 + 0.9 \cdot 1 \cdot V_{\Pi_1}(''(OK \vee \neg OK) \wedge \neg Charging'')\\ V_{\Pi_1}(''(OK \vee \neg OK) \wedge \neg Charging'') &= 1 - 1 + 0.9 \cdot (0.9 \cdot V_{\Pi_1}(''OK \wedge \neg Charging'')\\ &\qquad + 0.1 \cdot V_{\Pi_1}(''\neg OK \wedge \neg Charging''))\\ V_{\Pi_1}(''\neg OK \wedge Charging'') &= 1 - 1 \cdot 1 + 0.9 \cdot 1 \cdot V_{\Pi_1}(''\neg OK \wedge \neg Charging'')\\ V_{\Pi_1}(''\neg OK \wedge \neg Charging'') &= 1 - 1 \cdot 10 + 0.9 \cdot 1 \cdot V_{\Pi_1}(''(OK \vee \neg OK) \wedge \neg Charging'')\end{aligned}$$

$\rightarrow$

$$\begin{aligned}V_{\Pi_1}(''OK \wedge Charging'') &\approx 900\\ V_{\Pi_1}(''OK \wedge \neg Charging'') &\approx 810\\ V_{\Pi_1}(''(OK \vee \neg OK) \wedge Charging'') &\approx 641.742\\ V_{\Pi_1}(''(OK \vee \neg OK) \wedge \neg Charging'') &\approx 713.047\\ V_{\Pi_1}(''\neg OK \wedge Charging'') &\approx 569.468\\ V_{\Pi_1}(''\neg OK \wedge \neg Charging'') &\approx 632.742\end{aligned}$$

The algorithm will then suggest the following plan:

$$\begin{aligned}\Pi_2 = \Big\{&(''OK \wedge Charging'', StabilizeTemp),\\ &(''OK \wedge \neg Charging'', BeginCharging),\\ &(''(OK \vee \neg OK) \wedge Charging'', SenseTemp),\\ &(''(OK \vee \neg OK) \wedge \neg Charging'', SenseTemp),\\ &(''\neg OK \wedge Charging'', StabilizeTemp),\\ &(''\neg OK \wedge \neg Charging'', StabilizeTemp)\Big\}\end{aligned} \tag{B.20}$$

The algorithm then solves the equation system with the new plan:

$$\begin{aligned}V_{\Pi_2}(''OK \wedge Charging'') &= 100 - 1 \cdot 10 + 0.9 \cdot 1 \cdot V_{\Pi_2}(''OK \wedge Charging'')\\ V_{\Pi_2}(''OK \wedge \neg Charging'') &= 1 - 1 \cdot 1 + 0.9 \cdot 1 \cdot V_{\Pi_2}(''OK \wedge Charging'')\\ V_{\Pi_2}(''(OK \vee \neg OK) \wedge Charging'') &= 1 - 1 + 0.9 \cdot (0.9 \cdot V_{\Pi_2}(''OK \wedge Charging'')\\ &\quad + 0.1 \cdot V_{\Pi_2}(''\neg OK \wedge Charging''))\\ V_{\Pi_2}(''(OK \vee \neg OK) \wedge \neg Charging'') &= 1 - 1 + 0.9 \cdot (0.9 \cdot V_{\Pi_2}(''OK \wedge \neg Charging'')\\ &\quad + 0.1 \cdot V_{\Pi_2}(''\neg OK \wedge \neg Charging''))\\ V_{\Pi_2}(''\neg OK \wedge Charging'') &= 1 - 1 \cdot 10 + 0.9 \cdot 1 \cdot V_{\Pi_2}(''(OK \vee \neg OK) \wedge Charging'')\\ V_{\Pi_2}(''\neg OK \wedge \neg Charging'') &= 1 - 1 \cdot 10 + 0.9 \cdot 1 \cdot V_{\Pi_2}(''(OK \vee \neg OK) \wedge \neg Charging'')\end{aligned}$$

$\rightarrow$

$$\begin{aligned}V_{\Pi_2}(''OK \wedge Charging'') &\approx 900\\ V_{\Pi_2}(''OK \wedge \neg Charging'') &\approx 810\\ V_{\Pi_2}(''(OK \vee \neg OK) \wedge Charging'') &\approx 792.372\\ V_{\Pi_2}(''(OK \vee \neg OK) \wedge \neg Charging'') &\approx 713.047\\ V_{\Pi_2}(''\neg OK \wedge Charging'') &\approx 704.135\\ V_{\Pi_2}(''\neg OK \wedge \neg Charging'') &\approx 632.742\end{aligned}$$

The policy Π_2 is the policy found by the algorithm because no alterations to the policy can be made that improves the expected average utility.

B.3.3 EBRPD Represented as a Transition System with Labels

The transition system representing the EBRPD can be constructed as follows:

$$\Sigma = (S, A, \gamma, L) \tag{B.21}$$

Where the states are given as:

$$\begin{aligned} S = \{&''OK \wedge Charging'', \\ &''OK \wedge \neg Charging'', \\ &''(OK \vee \neg OK) \wedge Charging'', \\ &''(OK \vee \neg OK) \wedge \neg Charging'', \\ &''\neg OK \wedge Charging'', \\ &''\neg OK \wedge \neg Charging''\} \end{aligned} \tag{B.22}$$

The actions are given as:

$$A = \{SenseTemp, StabilizeTemp, BeginCharging, StopCharging\} \tag{B.23}$$

The state transition function γ is given as:

$$\begin{aligned} \gamma(''OK \wedge Charging'', StabilizeTemp) &= \{''OK \wedge Charging''\} \\ \gamma(''OK \wedge \neg Charging'', StabilizeTemp) &= \{''OK \wedge \neg Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge Charging'', StabilizeTemp) &= \{''(OK \vee \neg OK) \wedge Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge \neg Charging'', StabilizeTemp) &= \{''(OK \vee \neg OK) \wedge \neg Charging''\} \\ \gamma(''\neg OK \wedge Charging'', StabilizeTemp) &= \{''(OK \vee \neg OK) \wedge Charging''\} \\ \gamma(''\neg OK \wedge \neg Charging'', StabilizeTemp) &= \{''(OK \vee \neg OK) \wedge \neg Charging''\} \\ \gamma(''OK \wedge Charging'', SenseTemp) &= \{''OK \wedge Charging'','' \neg OK \wedge Charging''\} \\ \gamma(''OK \wedge \neg Charging'', SenseTemp) &= \{''OK \wedge \neg Charging'','' \neg OK \wedge \neg Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge Charging'', SenseTemp) &= \{''OK \wedge Charging'','' \neg OK \wedge Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge \neg Charging'', SenseTemp) &= \{''OK \wedge \neg Charging'','' \neg OK \wedge \neg Charging''\} \\ \gamma(''\neg OK \wedge Charging'', SenseTemp) &= \{''OK \wedge Charging'','' \neg OK \wedge Charging''\} \\ \gamma(''\neg OK \wedge \neg Charging'', SenseTemp) &= \{''OK \wedge \neg Charging'','' \neg OK \wedge \neg Charging''\} \\ \gamma(''OK \wedge Charging'', StopCharging) &= \{''OK \wedge \neg Charging''\} \\ \gamma(''OK \wedge \neg Charging'', BeginCharging) &= \{''OK \wedge Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge Charging'', StopCharging) &= \{''(OK \vee \neg OK) \wedge \neg Charging''\} \\ \gamma(''(OK \vee \neg OK) \wedge \neg Charging'', BeginCharging) &= \{''(OK \vee \neg OK) \wedge Charging''\} \\ \gamma(''\neg OK \wedge Charging'', StopCharging) &= \{''\neg OK \wedge \neg Charging''\} \\ \gamma(''\neg OK \wedge \neg Charging'', BeginCharging) &= \{''\neg OK \wedge Charging''\} \end{aligned} \tag{B.24}$$

The labeling function L is given as:

$$\begin{aligned} L(''OK \wedge Charging'') &= \{OK, Charging\} \\ L(''OK \wedge \neg Charging'') &= \{OK, \neg Charging\} \\ L(''(OK \vee \neg OK) \wedge Charging'') &= \{Charging\} \\ L(''(OK \vee \neg OK) \wedge \neg Charging'') &= \{\neg Charging\} \\ L(''\neg OK \wedge Charging'') &= \{\neg OK, Charging\} \\ L(''\neg OK \wedge \neg Charging'') &= \{\neg OK, \neg Charging\} \end{aligned} \tag{B.25}$$

Because of the need to have partial observable propositions there is no closed world assumption. That means both positive and negative propositions needs to be part of the labeling function. If both the positive and negative version of a proposition is omitted there is no knowledge present about the truth value of the proposition. This is the case for the atomic proposition OK in the state $''(OK \vee \neg OK) \wedge \neg Charging''$ and the state $''(OK \vee \neg OK) \wedge Charging''$.

Figure B.9 shows the transition system with the labels omitted (state names correspond to labels though). The same notations and abbreviations are used as with the MDP.

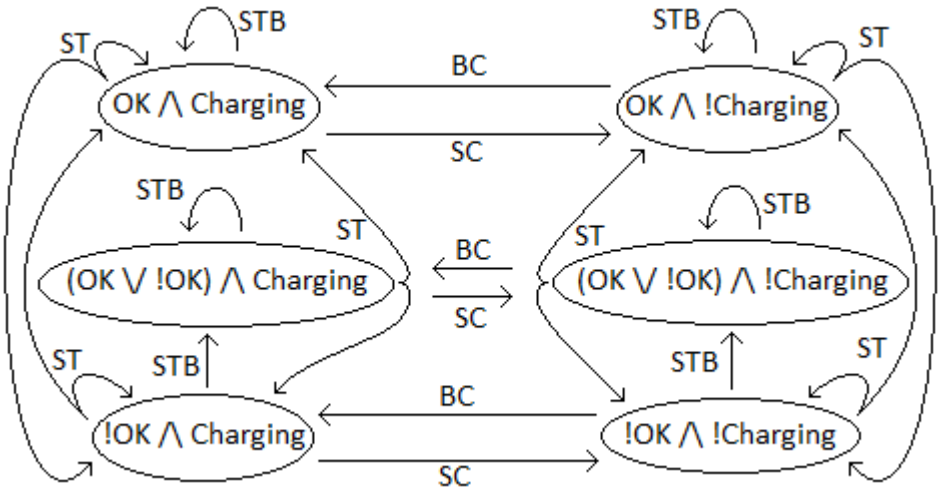

Figure B.9: Transition System with Labels Omitted.

B.3.4 Test Run of Progression of CTL on EBRPD

This section will construct a plan using execution contexts through progression of an example test goal written in CTL.

Let the goal be the CTL formula shown in equation B.26.

$$\begin{aligned} \text{AG } (\text{A}\,(\text{EF}\,(OK \wedge Charging) \cup OK \wedge Charging) \wedge \text{AF } (OK \wedge Charging)) \\ \wedge \text{AG } (OK \vee \neg Charging) \end{aligned} \tag{B.26}$$

The first sub goal from equation B.26 is:

$$\text{A}\,(\text{EF}\,(OK \wedge Charging) \cup OK \wedge Charging) \tag{B.27}$$

The second sub goal from equation B.26 is:

$$\text{AF } (OK \wedge Charging) \tag{B.28}$$

The last part AG $(OK \vee \neg Charging)$ ensures that it always holds that either the temperature is OK for recharging or the battery is not recharging.

Assume that "$(OK \vee \neg OK) \wedge \neg Charging$" is the initial state, the sub goal in equation B.27 is associated with execution context *c1* and the sub goal in equation B.28 is associated with execution context *c2*.

The plan will then be found in the following manner:

Starting from "$(OK \vee \neg OK) \wedge \neg Charging$" it is seen that L("$(OK \vee \neg OK) \wedge \neg Charging$") $\not\models OK \wedge Charging$. Therefore EF($OK \wedge Charging$) needs to be satisfied and A(EF($OK \wedge Charging$) $\cup$ $OK \wedge Charging$) has to be satisfied for all the possible successor states.

The applicable actions in "$(OK \vee \neg OK) \wedge \neg Charging$" is *SenseTemp, StabilizeTemp* and *BeginCharging*. *BeginCharging* cannot be applied because it would lead to a state violating AG $(OK \vee \neg Charging)$. Through exploration it is seen that both *SenseTemp* and *StabilizeTemp* can lead to $OK \wedge Charging$. Assuming the exploration approach returns the action resulting in the shortest path, *SenseTemp* is returned and explored first. *SenseTemp* is therefore chosen as the given action to take in the state "$(OK \vee \neg OK) \wedge \neg Charging$" when the execution context is *c1*.

The two possible states *SenseTemp* can lead to from the state "$(OK \vee \neg OK) \wedge \neg Charging$" is: "$\neg OK \wedge \neg Charging$" and "$OK \wedge \neg Charging$".

The execution context should not be altered when transitioning to "$\neg OK \wedge \neg Charging$" because the progressed goal is not fulfilled yet. The same goes for "$OK \wedge \neg Charging$". Some action therefore needs to be found such that EF($OK \wedge Charging$) is satisfied and A(EF($OK \wedge Charging$) $\cup$ $OK \wedge Charging$) is satisfied in the successor states for both states.

From "$OK \wedge \neg Charging$" the shortest path that leads to $OK \wedge Charging$ is taking

the action *BeginCharging*. The only successor state from "$OK \wedge \neg Charging$" after applying the action *BeginCharging* is the state "$OK \wedge Charging$" that of course satisfies the goal $OK \wedge Charging$. The execution context is therefore altered to *c2* when transitioning from "$OK \wedge \neg Charging$".

From "$\neg OK \wedge \neg Charging$" the shortest path that leads to $OK \wedge Charging$ is taking the action *SenseTemp*.

The execution context should not be altered when looping because the progressed goal is the same as previous.

The other successor state "$OK \wedge \neg Charging$" has already been explored with the same progressed goal so here the execution context does not need to be changed and no new states needs to be explored as well.

The partial plan found this far can now be listed as shown in table B.4.

State	Context	Action	Next state	Next context
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge \neg Charging$"	c1	BeginCharging	"$OK \wedge Charging$"	c2
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1

Table B.4: Partial Plan.

The actions applicable in the state "$OK \wedge Charging$" is *SenseTemp, StabilizeTemp* and *StopCharging*. Taking the action *SenseTemp* violates AG $(OK \vee \neg Charging)$. The two paths $\{StabilizeTemp\}$ and $\{StopCharging, BeginCharging\}$ both satisfies AF $(OK \wedge Charging)$ but as before lets assume the shortest path is returned i.e. *StabilizeTemp*. The only successor state from "$OK \wedge Charging$" when applying the action *StabilizeTemp* is "$OK \wedge Charging$". The state satisfies the goal $(OK \wedge Charging)$ so the execution context is changed to *c1* again.

The new partial plan can be seen in table B.5 on the next page.

From "$OK \wedge Charging$" multiple paths can be found that satisfies A(EF($OK \wedge Charging$)$\cup$ $OK \wedge Charging$) and does not violate AG $(OK \vee \neg Charging)$. Assuming the shortest is returned *StabilizeTemp* is the chosen action. Once again it is seen that the only successor state from "$OK \wedge Charging$" when applying the action *StabilizeTemp* is "$OK \wedge Charging$". The state satisfies the goal $OK \wedge Charging$ so the execution context is changed to *c2* again. This completes the plan.

The complete plan can be seen in table B.6 on the facing page.

State	Context	Action	Next state	Next context
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge \neg Charging$"	c1	BeginCharging	"$OK \wedge Charging$"	c2
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge Charging$"	c2	StabilizeTemp	"$OK \wedge Charging$"	c1

Table B.5: Extended Partial Plan.

State	Context	Action	Next state	Next context
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$(OK \vee \neg OK) \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge \neg Charging$"	c1	BeginCharging	"$OK \wedge Charging$"	c2
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$\neg OK \wedge \neg Charging$"	c1
"$\neg OK \wedge \neg Charging$"	c1	SenseTemp	"$OK \wedge \neg Charging$"	c1
"$OK \wedge Charging$"	c2	StabilizeTemp	"$OK \wedge Charging$"	c1
"$OK \wedge Charging$"	c1	StabilizeTemp	"$OK \wedge Charging$"	c2

Table B.6: Complete Plan.

B.3.5 EBRPD Represented as an Epistemic Planning Domain

This section will model the EBRPD as an epistemic planning domain Σ.

Given the set of atomic propositions $\{OK, Charging\}$, where Charging is abbreviated to C, and a single agent a let the epistemic model $M = (W, R, V)$ be given by:

$$W = \{w_1, w_2, w_3, w_4\} \tag{B.29}$$

$$R(a) = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_4, w_4), (w_2, w_4), (w_4, w_2)\} \tag{B.30}$$

$$\begin{aligned} V(OK) &= \{w_1, w_2\} \\ V(Charging) &= \{w_1, w_3\} \end{aligned} \tag{B.31}$$

Let the action $(StabilizeTemp, \{e'_1, e'_2, e'_3\})$ be abbreviated STB, the action $(SenseTemp, \{e_1, e_2\})$ be abbreviated ST, the action $(BeginCharging, \{e''_1\})$ be abbreviated BC and the action $(StopCharging, \{e'''_1\})$ be abbreviated SC. Assume (M, $\{w_2, w_4\}$) is the initial state.

Assume the epistemic actions available are defined as represented graphically in figure B.10.

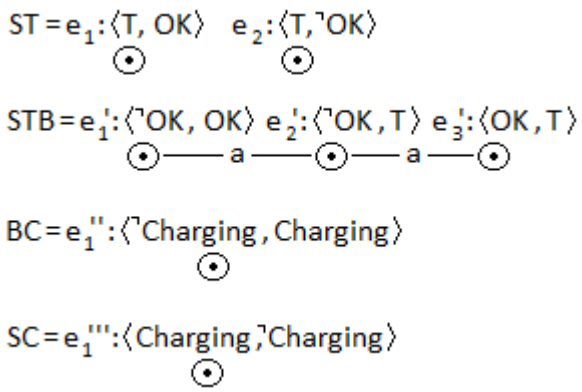

Figure B.10: Epistemic Actions.

The epistemic planning domain has been explored through a breadth first search starting from the initial state. The epistemic planning domain is represented graphically as shown in figure B.11. Note that some of the transitions rely on states being bisimilar.

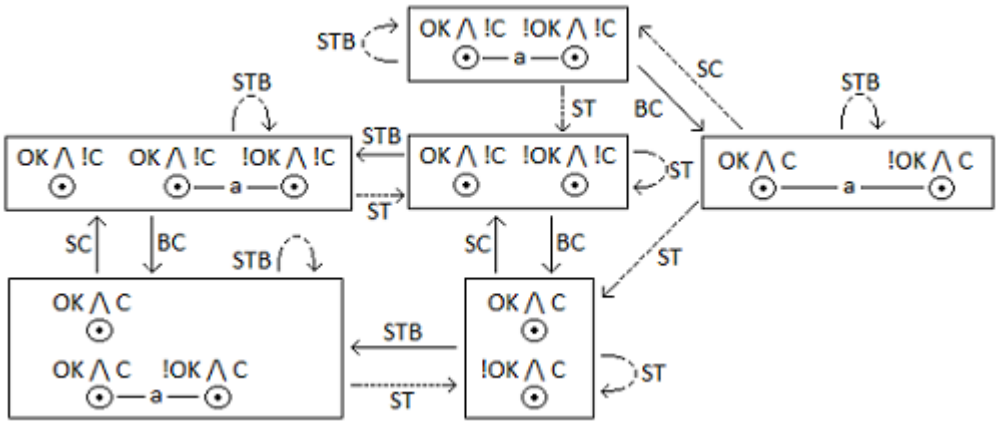

Figure B.11: Epistemic Planning Domain - EBRPD.

B.3.6 Test Run of Epistemic Planning on EBRPD

Some example epistemic planning problems (with different goals and initial states) will be defined in this section. Plans for these epistemic planning problems will then

be found through example test runs.

Let the planning problems be defined as follows:

Problem One

- Σ = figure B.11 on the preceding page.
- s0 = figure B.12.
- $\Phi_g = OK \wedge Charging$.

Problem Two

- Σ = figure B.11 on the preceding page.
- s0 = figure B.13.
- $\Phi_g = OK \wedge Charging$.

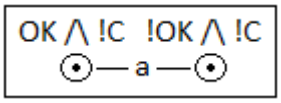

Figure B.12: Problem One - Initial Epistemic State.

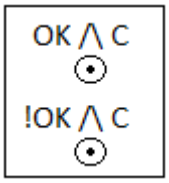

Figure B.13: Problem Two - Initial Epistemic State.

Solutions can be found through breadth first searches. In the following it is assumed that actions will be chosen in alphabetical order. It will be noted if the solution found is strong or weak and if there exists a strong solution.

For *ProblemOne* no strong solutions exist. In case the first weak solution that is found is returned the algorithm will return $s0 \otimes BC \otimes ST$. The resulting state is the same as represented in figure B.13.

Note that the state does not entail $OK \wedge Charging$ because not all of the designated

worlds (and their partitions) entail the formula. There is however one designated world, or more precisely a designated world and its partition, that entails the formula. The solution is therefore a weak solution.

For *ProblemTwo* no strong solution exists. In case no actions taken is an approved solution, the first weak solution to be found is simply $s0$. Otherwise the first weak solution to be returned is $s0 \otimes ST$. The resulting state is the same as the initial state in both cases.

B.4 Gravity Gradient Boom Deployment Planning Problem

The deployment of the *Gravity Gradient Boom* is a stepwise procedure. The order of the actions taken are important and the actions are stepwise dependent of each other. The problem does not really entail much planning because of this. That said the planning domain will still be explored for the different planning procedures in hope of revealing more pros and cons of the different approaches.

Assume the specification of the *Gravity Gradient Boom Deployment Planning Domain* (GGBDPD) is given as described in the following paragraphs.

Let there be given the following states:

InitialState
Released
Detumbling
Detumbled
OrrientedTowardsEarth (OTE)
BoomReleaseBegun (BRB)
BoomDeployed (BD)
PowerSaving
OrrientedTowardsSpace (OTS)
OrrientedTowardsEarthandPowerSaving (OTEandPS)

From the initial state (*InitialState*) it is assumed there is an action that releases the satellite (*SatelliteRelease - SR*). This leads to the state *Released*.

For simplicity one could assume that the initial state is *Released* seeing that the actual control of satellite release is probably handled by and external system. The satellite is probably not even ”online” before the release is complete. Including the state does however give a more thorough description of the actual steps completed in the stabilization of the satellite. For the examples it is therefore assumed to be present.

In *Released* one should be able to begin detumbling (*BeginDetumbling - BD*) tran-

sitioning to the state *Detumbling*. The satellite uses a B-dot algorithm to detumble. Detumbling is done using the 3-axis magnetometer to measure the magnetic field of the earth and the three magnetic torquers to apply counter spin (see [11]).

From *Detumbling* one should be able to check if the satellite has slowed down its rotation enough (*CheckDetumbling - CD*). This action should either lead to the state *Detumbled* or make no state transition in case the satellite has not slowed down yet.

When the satellite is detumbled the orrientation of the satellite should be able to be checked (*CheckOrrientation - CO*). This is assumed to be done using the sun sensor, the heat sensors and the 3-axis magnetometer. If the satellite is oriented towards earth correctly the action should lead to the state *OrrientedTowardsEarth (OTE)*.

From *OTE* it should be possible to do the action *BoomRelease BR*. This action should lead to the state *BoomReleaseBegun (BRB)*.

From *BRB* it should be possible to check the progress of the boom release - is it released or not - using the sensor present for the task. Let this action be called *CheckBoomReleaseProgress (CBRP)*. Let the action lead to the state *BoomDeployed (BD)* if the boom is released. If the boom is not released yet no state transition should be done.

From the state *BD* the attitude control subsystem should shift service level from detumbling to the more power saving GG Boom dampening. Let this action be called *ShiftMode (SM)* and let it lead to the sate *PowerSaving*.

The satellite might still have stabilized orienting towards space. Therefore from the state *PS* the action *Check Orrientation (CO)* should be applied again. If the satellite is oriented correctly the action should lead to the state *OrrientedTowardsEarthandPowerSaving (OTEandPS)*. If the satellite is oriented incorrectly the action should lead to the state *OrrientedTowardsSpace (OTS)*.

From the state *OTS* the action *Flip* should be applicable. This actions should lead to the state *OTEandPS*.

The state *OTEandPS* is assumed to be the final state is this planning problem. It should be noted though that after reaching this state the satellite should be able to transmit data at a high rate to the ground station. In a power sufficient manner.

B.4.1 GGBDPD Represented as a MDP

Given an assignment of probabilities, costs and rewards the MDP representing the GGBDPD can be constructed as the example described by figure B.14 and figure B.15.

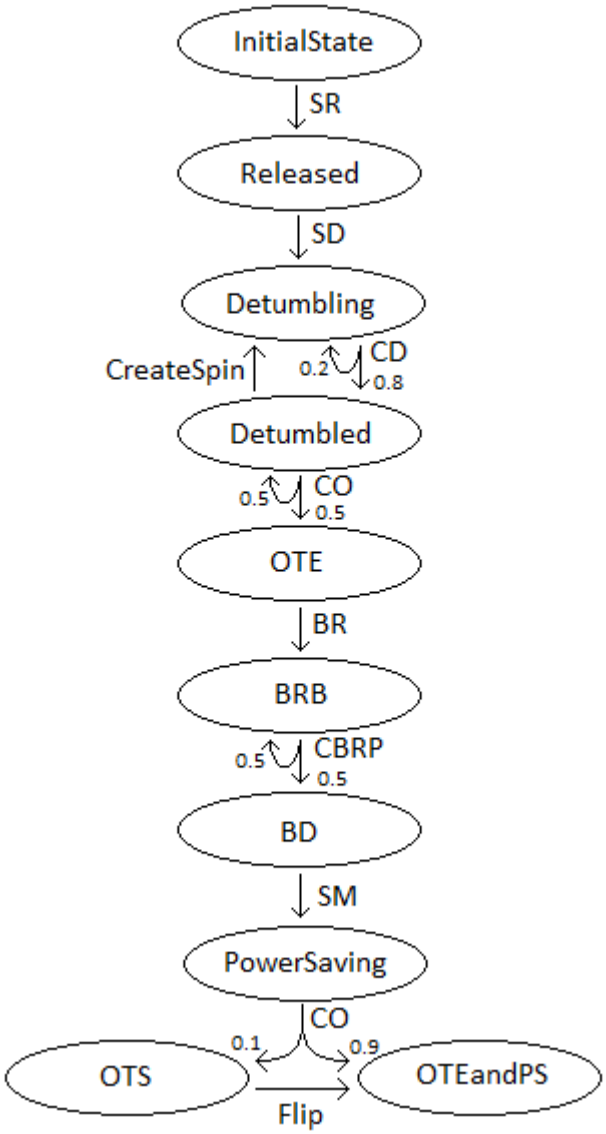

Figure B.14: MDP Representing the GGBDPD - Probabilities and Actions.

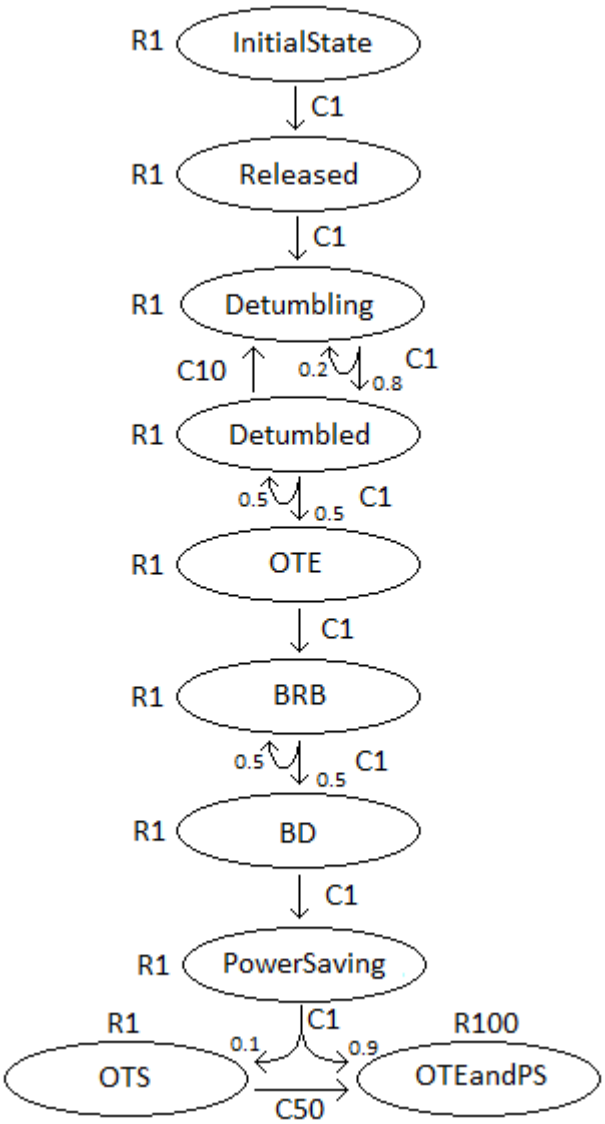

Figure B.15: MDP Representing the GGBDPD - Costs and Rewards.

B.4.2 Test Run of Policy Iteration on GGBDPD

This section will give a test execution of *Policy Iteration* on the MDP representing the GGBDPD.

Let the initial policy be given by:

$$\Pi_1 = \Big\{ \begin{array}{l} (InitialState, SR) \\ (Released, SD) \\ (Detumbling, CD) \\ (Detumbled, CreateSpin) \\ (OTE, BR) \\ (BRB, CBRP) \\ (BD, SM) \\ (PowerSaving, CO) \\ (OTS, Flip) \\ (OTEandPS, NoOp) \Big\} \end{array} \tag{B.32}$$

Note that *NoOp* is an action assumed to be applicable in *OTEandPS* resulting in no state change (the state transition is a self loop) and having the cost of 1.

With a value for γ the equation system V_{Π_1} can be solved. Assuming that γ is equal to 0.9 the result will be:

$$\begin{aligned}
V_{\Pi_1}(InitialState) &= 0.9 \cdot 1 \cdot V_{\Pi_1}(Released) \\
V_{\Pi_1}(Released) &= 0.9 \cdot 1 \cdot V_{\Pi_1}(Detumbling) \\
V_{\Pi_1}(Detumbling) &= 0.9 \cdot (0.8 \cdot V_{\Pi_1}(Detumbled) + 0.2 \cdot V_{\Pi_1}(Detumbling)) \\
V_{\Pi_1}(Detumbled) &= -9 + 0.9 \cdot 1 \cdot V_{\Pi_1}(Detumbling) \\
V_{\Pi_1}(OTE) &= 0.9 \cdot 1 \cdot V_{\Pi_1}(BRB) \\
V_{\Pi_1}(BRB) &= 0.9 \cdot (0.5 \cdot V_{\Pi_1}(BD) + 0.5 \cdot V_{\Pi_1}(BRB)) \\
V_{\Pi_1}(BD) &= 0.9 \cdot 1 \cdot V_{\Pi_1}(PowerSaving) \\
V_{\Pi_1}(PowerSaving) &= 0.9 \cdot (0.1 \cdot V_{\Pi_1}(OTS) + 0.9 \cdot V_{\Pi_1}(OTEandPS)) \\
V_{\Pi_1}(OTS) &= -49 + 0.9 \cdot 1 \cdot V_{\Pi_1}(OTEandPS) \\
V_{\Pi_1}(OTEandPS) &= 99 + 0.9 \cdot 1 \cdot V_{\Pi_1}(OTEandPS)
\end{aligned}$$

→

$$
\begin{aligned}
V_{\Pi_1}(InitialState) &\approx -30.5163 \\
V_{\Pi_1}(Released) &\approx -33.907 \\
V_{\Pi_1}(Detumbling) &\approx -37.6744 \\
V_{\Pi_1}(Detumbled) &\approx -42.907 \\
V_{\Pi_1}(OTE) &\approx 581.662 \\
V_{\Pi_1}(BRB) &\approx 646.292 \\
V_{\Pi_1}(BD) &\approx 789.912 \\
V_{\Pi_1}(PowerSaving) &\approx 877.68 \\
V_{\Pi_1}(OTS) &\approx 842 \\
V_{\Pi_1}(OTEandPS) &\approx 990
\end{aligned}
$$

The algorithm will then suggest the following plan:

$$
\begin{aligned}
\Pi_1 = \Big\{ & (InitialState, SR) \\
& (Released, SD) \\
& (Detumbling, CD) \\
& (Detumbled, CO) \\
& (OTE, BR) \\
& (BRB, CBRP) \\
& (BD, SM) \\
& (PowerSaving, CO) \\
& (OTS, Flip) \\
& (OTEandPS, NoOp) \Big\}
\end{aligned}
\tag{B.33}
$$

The algorithm then solves the equation system with the new plan:

$$
\begin{aligned}
V_{\Pi_2}(InitialState) &= 0.9 \cdot 1 \cdot V_{\Pi_2}(Released) \\
V_{\Pi_2}(Released) &= 0.9 \cdot 1 \cdot V_{\Pi_2}(Detumbling) \\
V_{\Pi_2}(Detumbling) &= 0.9 \cdot (0.8 \cdot V_{\Pi_2}(Detumbled) + 0.2 \cdot V_{\Pi_2}(Detumbling)) \\
V_{\Pi_2}(Detumbled) &= 0.9 \cdot (0.5 \cdot V_{\Pi_2}(Detumbled) + 0.5 \cdot V_{\Pi_2}(OTE)) \\
V_{\Pi_2}(OTE) &= 0.9 \cdot 1 \cdot V_{\Pi_2}(BRB) \\
V_{\Pi_2}(BRB) &= 0.9 \cdot (0.5 \cdot V_{\Pi_2}(BD) + 0.5 \cdot V_{\Pi_2}(BRB)) \\
V_{\Pi_2}(BD) &= 0.9 \cdot 1 \cdot V_{\Pi_2}(PowerSaving) \\
V_{\Pi_2}(PowerSaving) &= 0.9 \cdot (0.1 \cdot V_{\Pi_2}(OTS) + 0.9 \cdot V_{\Pi_2}(OTEandPS)) \\
V_{\Pi_2}(OTS) &= -49 + 0.9 \cdot 1 \cdot V_{\Pi_2}(OTEandPS) \\
V_{\Pi_2}(OTEandPS) &= 99 + 0.9 \cdot 1 \cdot V_{\Pi_2}(OTEandPS)
\end{aligned}
$$

→

$$
\begin{aligned}
V_{\Pi_2}(InitialState) &\approx 338.473 \\
V_{\Pi_2}(Released) &\approx 376.082 \\
V_{\Pi_2}(Detumbling) &\approx 417.868 \\
V_{\Pi_2}(Detumbled) &\approx 475.906 \\
V_{\Pi_2}(OTE) &\approx 581.662 \\
V_{\Pi_2}(BRB) &\approx 646.292 \\
V_{\Pi_2}(BD) &\approx 789.912 \\
V_{\Pi_2}(PowerSaving) &\approx 877.68 \\
V_{\Pi_2}(OTS) &\approx 842 \\
V_{\Pi_2}(OTEandPS) &\approx 990
\end{aligned}
$$

The policy Π_2 is the policy found by the algorithm because no alterations to the policy can be made that improves the expected average utility.

B.4.3 GGBDPD Represented as a Transition System with Labels

The transition system representing the GGBDPD can be constructed as follows:

$$\Sigma = (S, A, \gamma, L) \tag{B.34}$$

Where the states are given as in the description of the domain in section B.4 on page 82. For each state assume that there is a label on the form ”atStateName” which is true in the given state (for instance atInitialState is true in InitialState).
The actions given are:

$$A = \{SR, SD, CD, CreateSpin, CO, BR, CBRP, SM, Flip\} \tag{B.35}$$

Where the state transition function γ is given as described in the introduction of section B.4 on page 82.

Figure B.16 on the facing page shows the transition system with the labels omitted (state names correspond to labels though).

B.4.4 Test Run of Progression of CTL on GGBDPD

This section will construct a plan using execution contexts through progression of an example test goal written in CTL.

Let the goal be the CTL formula shown in equation B.36.

$$\text{AG}\ (\text{EF}\,(atOTEandPs)) \tag{B.36}$$

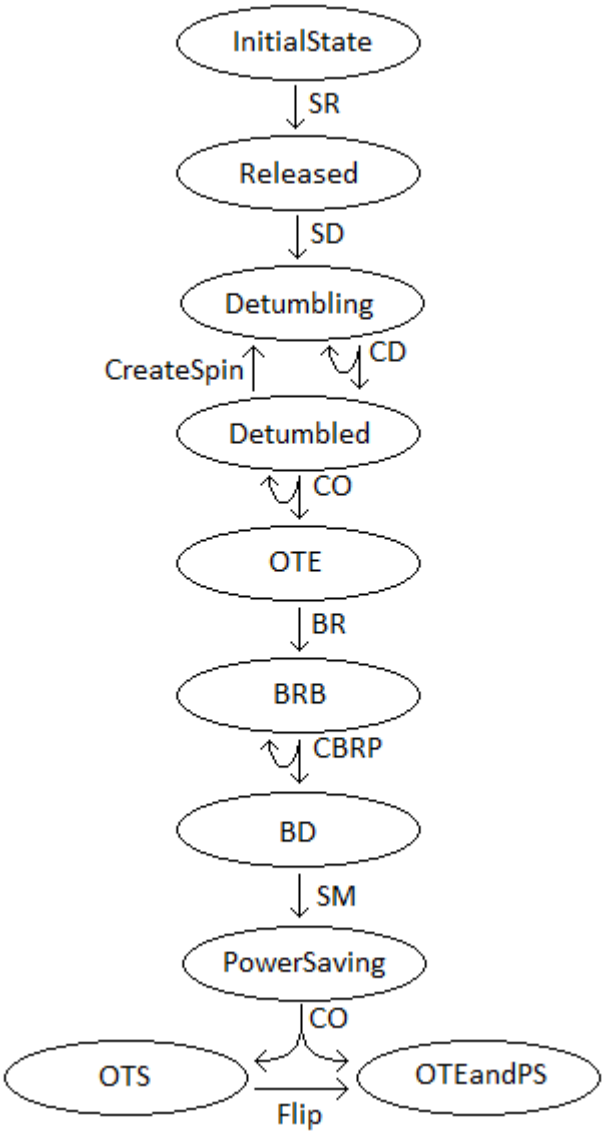

Figure B.16: Transition System with Labels Omitted.

The only sub goal from equation B.36 on page 88 is:

$$\mathrm{EF}\,(atOTEandPs) \tag{B.37}$$

Assume that *InitialState* is the initial state and that the sub goal in equation B.37 is associated with execution context *c1*.

The plan will then be found in the following manner:

Starting from *InitialState* it is seen that L(*InitialState*) $\not\models$ *atOTEandPS*. Therefore EF (*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *SR* is possible, luckily choosing this action satisfies EF(*atOTEandPS*).

It is seen that the labels of the successor state L(*Released*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *SD* is possible, luckily choosing this action satisfies EF(*atOTEandPS*).

It is seen that the labels of the successor state L(*Detumbling*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *CD* is possible, luckily choosing this action satisfies EF(*atOTEandPS*) for both successor states. The successor state *Detumbling* does not need to be expanded further because it has already been expanded with the initial execution context (*c1*).

It is seen that the labels of the successor state L(*Detumbled*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). The only action satisfying EF(*atOTEandPS*) in *Detumbled* is *CO*. The successor state *Detumbled* does not need to be expanded further because it has already been expanded with the initial execution context *c1*.

It is seen that the labels of the successor state L(*OTE*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *BR* is possible, luckily choosing this action satisfies EF(*atOTEandPS*).

It is seen that the labels of the successor state L(*BRB*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *CBRP* is possible, luckily choosing this action satisfies EF(*atOTEandPS*) for both successor states. The successor state *BRB* does not need to be expanded further because it has already been expanded with the initial execution context *c1*.

It is seen that the labels of the successor state L(*BD*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the cho-

sen action). Only the action *SM* is possible, luckily choosing this action satisfies EF(*atOTEandPS*).

It is seen that the labels of the successor state L(*PowerSaving*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *CO* is possible, luckily choosing this action satisfies EF(*atOTEandPS*) for both successor states.

It is seen that the labels of the successor state L(*OTS*) $\not\models$ *atOTEandPS*. Therefore EF(*atOTEandPS*) needs to be satisfied in the state and for all the possible successor states (for the chosen action). Only the action *Flip* is possible, luckily choosing this action satisfies EF(*atOTEandPS*).

When reaching *OTEandPS* the execution context should normally be changed to the next sub goal but we only have one sub goal so the execution context remains *c1*. In both cases of reaching *OTEandPS* we find that L(*OTEandPS*) $\models$ *atOTEandPS*. Assuming there is a *NoOp* action that leads to *OTEandPS* from *OTEandPS* this action can be applied in *OTEandPS*. The complete plan can then be listed as shown in table B.7.

State	Context	Action	Next state	Next context
InitialState	c1	SR	Released	c1
Released	c1	SD	Detumbling	c1
Detumbling	c1	CD	Detumbled	c1
Detumbling	c1	CD	Detumbling	c1
Detumbled	c1	CO	Detumbled	c1
Detumbled	c1	CO	OTE	c1
OTE	c1	BR	BRB	c1
BRB	c1	CBRP	BRB	c1
BRB	c1	CBRP	BD	c1
BD	c1	SM	PowerSaving	c1
PowerSaving	c1	CO	OTS	c1
PowerSaving	c1	CO	OTEandPS	c1
OTS	c1	Flip	OTEandPS	c1
OTEandPS	c1	NoOp	OTEandPS	c1

Table B.7: Complete Plan.

B.4.5 Epistemic Planning Domain Representing the GGBDPD

This section will model the GGBDPD as an epistemic planning domain Σ.

Assume there is a single agent a and let the epistemic model $M = (W, R, V)$ be given

by:

$$W = \{InitialState, Released, Detumbling, Detumbled, \\ OTE, BRB, BD, PowerSaving, OTS, OTEandPS\} \tag{B.38}$$

$$R(a) = \{(InitialState, InitialState), (Released, Released), (Detumbling, Detumbling), \\ (OTE, OTE), (BRB, BRB), (BD, BD), (PowerSaving, PowerSaving), \\ (OTS, OTS), (OTEandPS, OTEandPS)\} \tag{B.39}$$

$$\begin{aligned} V(atInitialState) &= \{InitialState\} \\ V(atReleased) &= \{Released\} \\ V(atDetumbling) &= \{Detumbling\} \\ V(atDetumbled) &= \{Detumbled\} \\ V(atOTE) &= \{OTE\} \\ V(atBRB) &= \{BRB\} \\ V(atBD) &= \{BD\} \\ V(atPowerSaving) &= \{PowerSaving\} \\ V(atOTS) &= \{OTS\} \\ V(atOTEandPS) &= \{OTEandPS\} \end{aligned} \tag{B.40}$$

Let the epistemic actions available be defined as represented graphically in figure B.17.

From the epistemic state (M, {*InitialState*}) the planning domain has been explored in a breadth first manner. The state transition system shown in figure B.18 on page 94 and figure B.19 on page 95 is the partially explored epistemic planning domain where the depth explored corresponds to eight actions taken.

Note that some of the transitions rely on states being bisimilar.

B.4.6 Test Run of Epistemic Planning on GGBDPD

This section will give an example epistemic planning problem. For this planning problem a plan will be found through a test run of the epistemic planning approach.

Let the planning problem be defined as follows:

Problem One

- Σ corresponds to the partially explored domain shown in figure B.18 and figure B.19.

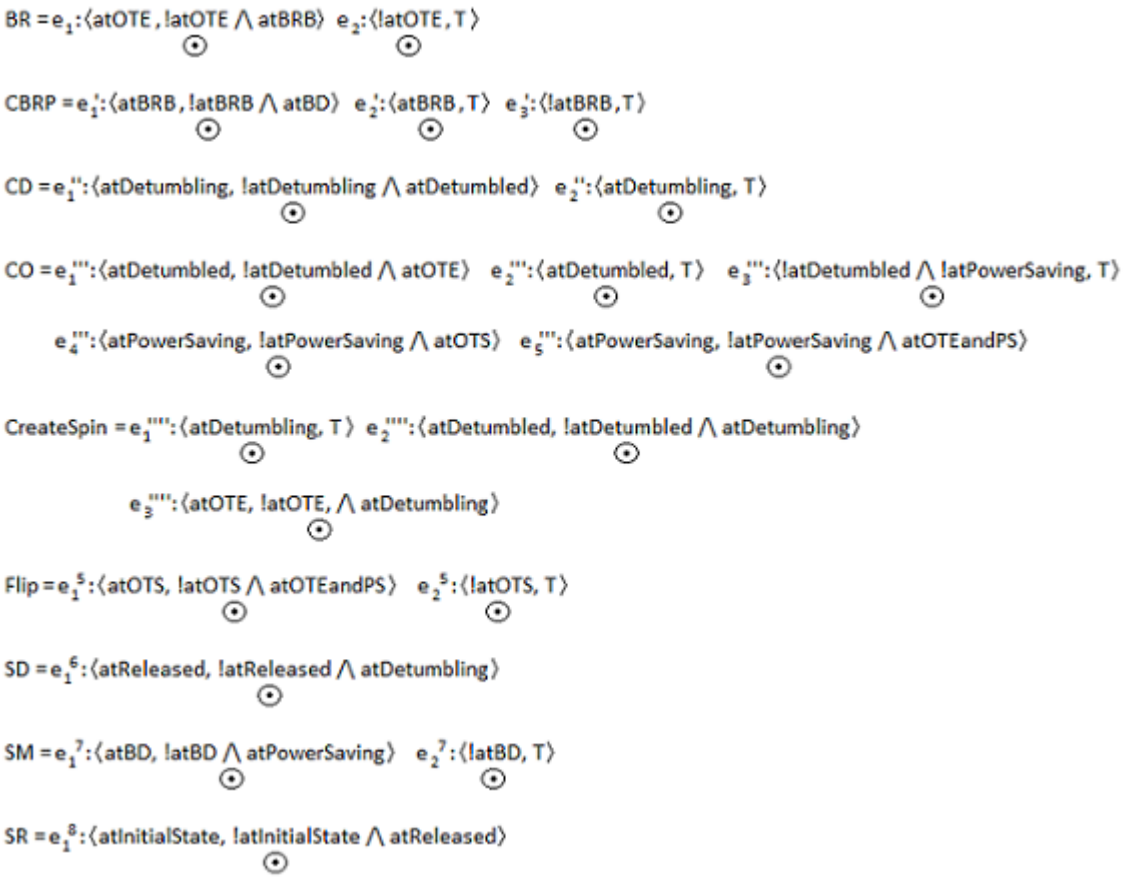

Figure B.17: Epistemic Actions.

- s0 = figure B.20 on page 95.
- $\Phi_g = atOTEandPS$.

The plan is found through an exploration of the epistemic planning domain. In the following it is assumed that actions will be chosen in alphabetical order. It will be noted if the solution found is strong or weak and if there exists a strong solution.

No strong solutions exists. The first weak solution to be found is $s0 \otimes SR \otimes SD \otimes CD \otimes BR \otimes CBRP \otimes SM \otimes CO$. Note that the resulting state does not entail $atOTEandPS$ because not all of the designated worlds (and their partitions) entails the formula. There is however one designated world that entails the formula and the solution is therefore a weak solution.

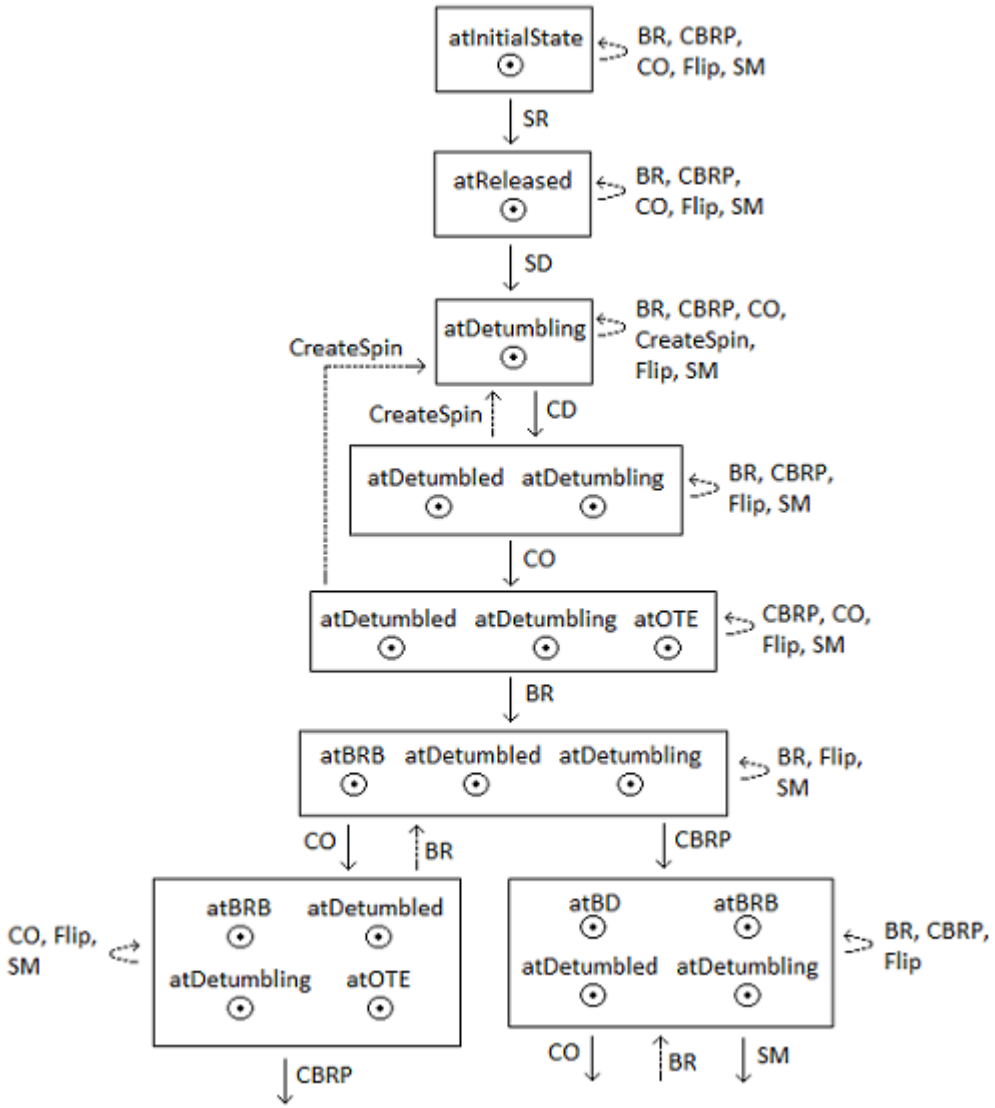

Figure B.18: Partial Epistemic Planning Domain, Part one - GGBDPD.

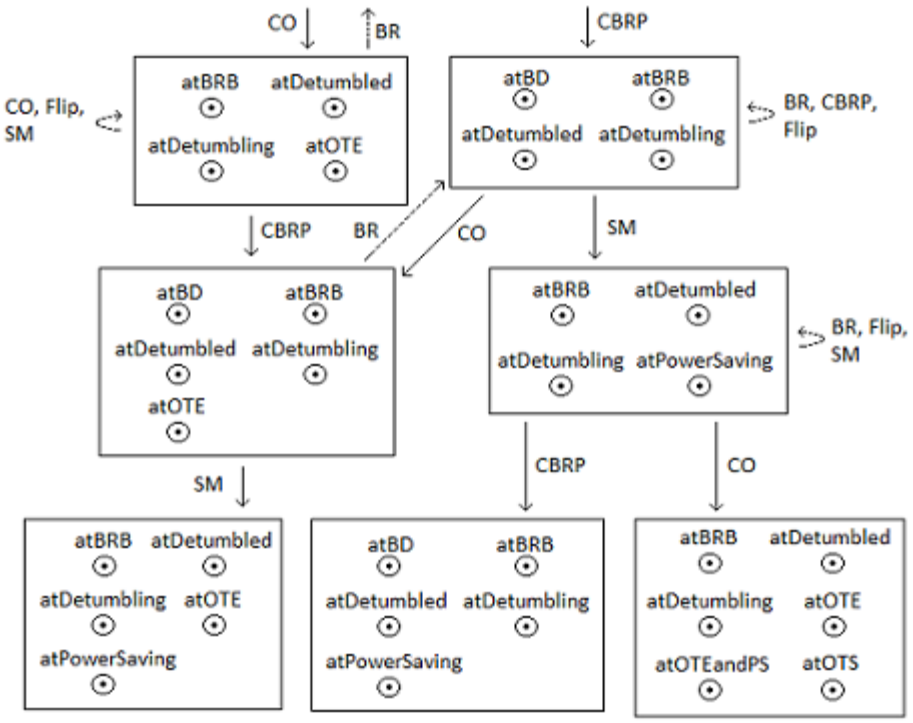

Figure B.19: Partial Epistemic Planning Domain, Part two - GGBDPD.

Figure B.20: Problem One - Initial Epistemic State.