Masarykova univerzita
Fakulta informatiky

Optimal Observation Scheduling for Systems under Temporal Constraints

master thesis

Eva Tesařová

Brno, Spring 2016

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Eva Tesařová

Advisor: prof. RNDr. Ivana Černá, CSc.

Acknowledgement

First of all, I would especially like to thank my supervisor Ivana Černá who was always offering me a guidance, valuable advice. I very appreciate all the motivation she gave me to work. My next big thanks goes to Mária Svoreňová for collaboration on this topic, it was a great experience to work with someone with such a a lot of enthusiasm to research and wide knowledge. I would like also to thank Jiří Barnát for providing the inspiration and comments. My thanks go also to the members of Parallel and Distributed Systems Laboratory for helpful conversations and great working environment.

Abstract

Autonomous control systems use various sensors to decrease the amount of uncertainty under which they operate. While providing partial observation of the current state of the system, sensors require resources such as energy, time and communication. We consider discrete systems with non-deterministic transitions and multiple observation modes. The observation modes provide different information about the states of the system and are associated with non-negative costs. We consider several control problems. First, we aim to construct a control and observation mode switching strategy that guarantees satisfaction of a finite-time temporal property given as a formula of syntactically co-safe fragment of LTL (scLTL) and at the same time, minimizes the worst-case cost accumulated until the point of satisfaction. Second, the bounded version of the problem is considered, where the temporal property must be satisfied within given finite time bound. Last, we provided means to allow express specification as more alternative tasks with priority rules and strategy is synthesized according to the rules. We present correct and optimal solutions to all proposed problems and demonstrate their usability on a case study motivated by robotic applications.

Keywords

control synthesis, non-deterministic system, temporal constraints, complementary safe LTL, partially observable system, least-violating strategy

Contents

1 **Introduction** 1
2 **Preliminaries** 4
2.1 Notation 4
2.2 Introduction to Control Synthesis 4
3 **Formal Model** 7
3.1 Non-deterministic Transition System 7
3.2 NTS with Observation Modes 8
3.3 Strategy 10
4 **Formal Specification** 11
4.1 Complementary Safe LTL 11
4.2 Translating scLTL Formula to Finite Automata 12
5 **Problem Formulation** 15
5.1 Optimal Task Control 15
5.2 Bounded Optimal Task Control 16
5.3 Optimal Mission Control 17
5.4 General Mission 20
6 **Related Work** 21
6.1 Optimizing Single Sensor Usage 21
6.2 Optimizing Multiple Sensor Usage 22
7 **Problem Solution** 24
7.1 Optimal Task Control 24
7.2 Bounded Optimal Task Control 30
7.3 Optimal Mission Control 33
7.4 Optimal General Mission Control 38
8 **Case Study** 39
8.1 Implementation 39
8.2 Task 39
8.3 Mission 42
9 **Conclusion** 44

CHAPTER 1

Introduction

Embedded systems used in transportation, medical and other safety critical applications typically operate under uncertainty. The source of the uncertainty can be of two types. The internal uncertainty is bounded to the system's control inputs such as noisy actuators in mobile robots. The external uncertainty arises from the system's interaction with the environment such as other robots or people operating in the same space. In order to lower the amount of uncertainty, sensors are deployed to provide information about the current state of the system. Individual sensors and their combinations may provide varying, partial observation of the current state of the system. At the same time, their deployment requires resources such as energy, time or communication.

The field of sensor scheduling studies the problem of the deployment of sensors in order to optimize estimation of a signal connected to the system's state. There is a wide range of results, *e.g.*, for linear systems [VZA+10], hybrid systems [FTR08] and for applications in robot motion planning [OGM+15]. The field of information gathering assumes a fixed set of sensors and aims to find a control strategy for the system that optimizes estimation of the signal. Recently, a problem combining optimization with temporal objectives has been considered in information gathering for discrete systems [JSB13].

Partial observability has been extensively studied for discrete systems in artificial intelligence and game theory. The main focus is typically on partially observable Markov decision processes (POMDPs) that model both partial observation and probabilistic uncertainties. The optimization objectives include expected total cost over finite horizon [CCGK15a], and expected average or discounted total cost over infinite horizon [KLC98, PGT03]. Besides optimization objectives, many systems also operate under temporal constraints. Recently, temporal logics such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) have been increasingly used to specify temporal properties of systems such as reachability, safety, stability of response. It might happen that the objective is not feasible without violating some of the temporal constraint. The aim is then to find the least violating strategy according to the priority assigned to constraints [THK+13, CCT+13].

The overview of results for partially observable stochastic games (with POMDPs as a subclass) with respect to various classes of temporal objectives can be

found in [CDH13]. It is important to note that most of the problems of quantitative nature formulated for POMDPs are undecidable to solve precisely or even to approximate [CDH13, MHC03]. All the above results consider systems with one fixed observation mode that can be seen as a deployment of a single sensor.

Comparing to the aforementioned fields of study, in this work we focus on a problem that combines the optimal and temporal control for systems with multiple observation modes. We present a discrete system for modeling the above setting referred to as a *non-deterministic transition system (NTS) with observation modes*. The non-determinism can be used to model both the internal and external uncertainty of the system whereas observation modes capture the sensing capabilities. In every step of an execution of the system, one decides which mode of partial observation to activate. Activation of each observation mode is associated with a non-negative cost. An example of a robotic system with limited energy resources and multiple sensing capabilities is a planetary rover. In [OGM+15], the authors design an optimal schedule for the use of a localization system in a rover that minimizes energy consumption while at the same time guarantees safe path following. While sensor readings are typically continuous, in this work we assume that the set of readings that affect decision-making can be represented by a finite set, *e.g.*, sets of values satisfying the same constraints.

We consider the following three problems. First, the aim is to construct a control and observation mode switching strategy for an NTS with observation modes that (i) guarantees satisfaction of a finite-time temporal property given as a formula of syntactically co-safe fragment of LTL (scLTL) and (ii) minimizes the worst-case cost accumulated until the point of satisfaction. The second problem considers the bounded version of the above problem, where the temporal property is required to be satisfied in at most $k \geqslant 1$ steps. Last, we generalized the first problem by providing means for building more complex formulas. Priorities can be assigned to building blocks of formula and strategy which takes into account priorities is then constructed.

Leveraging techniques from automata-based model checking and graph theory, we present correct and optimal solutions to all mentioned problems. We justify our restriction to objectives over finite time horizon since more intriguing problems, *e.g.*, involving infinite-time temporal properties and cost functions, and probabilistic models, turns out to be undecidable. At the same time, temporal properties over finite horizon offer lower computational and strategy complexity compared to the general class of temporal properties and cover many interesting properties typically considered, *e.g.*, in robotic applications [KYV01, JSB13, UWB14].

To the best of our knowledge, discrete systems with multiple modes were first considered only recently in [CMH08, CM11], where the authors focus on control

with respect to properties in infinite time horizon. The most related work to ours is [BG11] that considers a variation of POMDPs, where at each step the user can either choose to use the partial information or pay a fixed cost and receive the full information about the current state of the system. The authors discuss the problems of minimizing the worst-case or expected total cost before reaching a designated goal state with probability 1. While the former problem has optimal, polynomial solution, the latter proves undecidable.

The main contribution of this work is introducing a new model that extends the one in [BG11] in the sense that we allow multiple observation modes with varying costs. We design correct and optimal strategies to control such models to guarantee an scLTL formula while minimizing the corresponding cost, over bounded or unbounded time horizon.

The thesis is structured as follows. In Chapter 2 we offer a quick introduction to control synthesis. Next, in the Chapter 3 we introduce a formalism, *NTS with observation modes*, for modeling robotic systems operating under uncertainty. In Chapter 4 is presented a fragment of LTL, *co-safe LTL*, which is specification language for which we can guarantee satisfaction in a finite time. Examined problems are formulated in Chapter 5 and corresponding solutions are provided in Chapter 7. Last, in Chapter 8 we demonstrate usability of proposed algorithms on a case study motivated by robotic applications.

CHAPTER 2

Preliminaries

In this chapter we first familiarize the reader with the basic terminology used throughout this thesis. After that, we give a brief, intuitive introduction to control and planning. Real world motivation for studying proposed problems are provided.

2.1 Notation

For a set X, we use X^* to denote the set of all finite sequences of elements of X. The X^ω denotes the set of all infinite sequences.

Given a sequence $\sigma = x_0 \dots x_n \in X^*$. Prefix of σ is every word $x_0 \dots x_i$, where $0 \leqslant i \leqslant n$. Suffix of σ is every word $x_i \dots x_n$, where $0 \leqslant i \leqslant n$. Prefix (suffix) σ' of σ is a *strict prefix* (*strict suffix*) iff $\sigma \neq \sigma'$. Similarly is the above defined for infinite sequences.

A finite sequence $\sigma = x_0 \dots x_n \in X^*$ has length $|\sigma| = n + 1$, $\sigma(i) = x_i$ is the i-th element and $\sigma^i = \sigma(i) \dots \sigma(n)$ is the suffix starting with the i-th element, for $0 \leqslant i \leqslant n$.

Similarly, for an infinite sequence $\rho = x_0 x_1 \dots \in X^\omega$, $\rho(i) = x_i$ for all $i \geqslant 0$. A prefix of a finite sequence σ or an infinite sequence ρ is any sequence $\sigma(0) \dots \sigma(k)$ for $0 \leqslant k \leqslant |\sigma|$ or $\rho(0) \dots \rho(k)$ for $k \geqslant 0$, respectively.

2.2 Introduction to Control Synthesis

Robotics addresses automated physical devices with sensing, actuation, computation and moving capabilities. These devices are usually characterized by high dimensional control space. In a real world applications are devices sometimes too complex to be sufficient to specify their behavior directly. While they operate with low-level instructions, such as to set impulse to a motor or to get data from sensor, it becomes a fundamental need to automatically convert a high-level specification to this low-level description determining robot's behavior. This automated process of converting a high-level specification to the low-level description is referred as *control synthesis* and resulting description is usually called *scheduler*, *controller* or *strategy* .

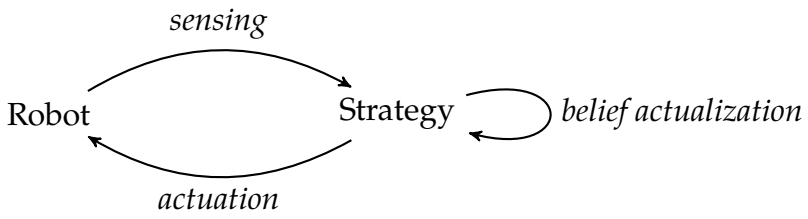


Figure 2.1: Control Flow

The problem of control synthesis is the counterpart to *formal verification*. Whereas the aim of formal verification is to prove correctness of a given system with respect to some property, *i.e.*, to verify that all runs of the system satisfy the property, the aim of control synthesis is to partially design the system such that it meets the requirements, *i.e.*, create a strategy restricting runs of the system to those satisfying the property.

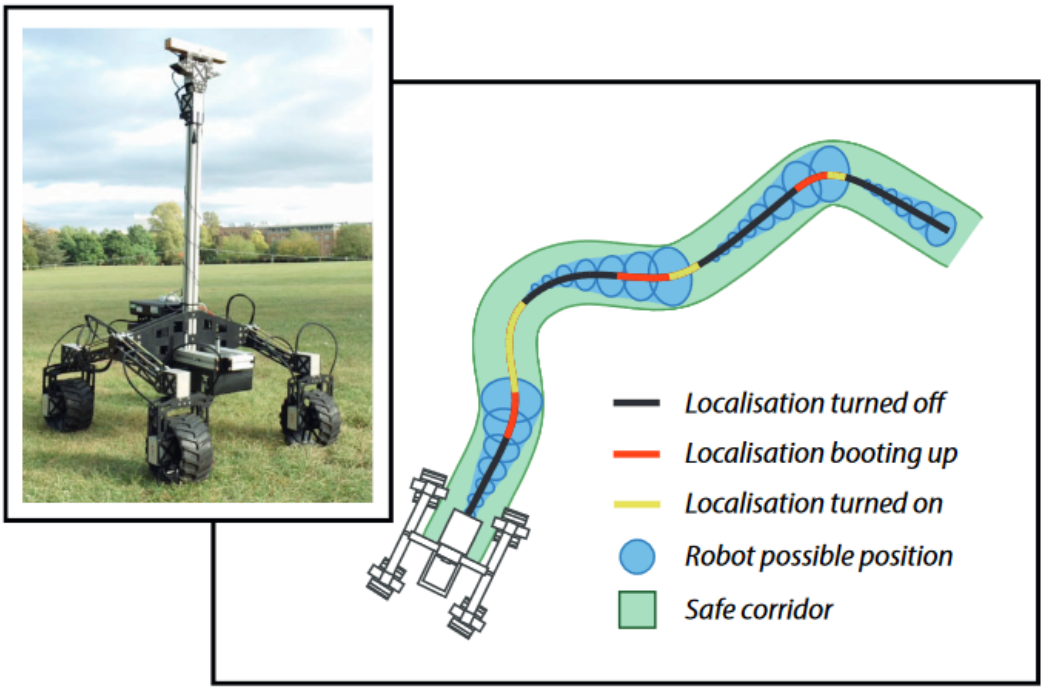
Strategy can be seen as a function, which on the basis of information from sensors and inner belief about the state of controlled system, determines how to act. The robot control loop is depicted in the Figure 2.2. If the strategy meets the requirements we say it is *feasible*.

Often it is also desirable to minimize resource consumption, such as energy or time. If the strategy is the best among all feasible strategies with respect to quality criterion, we call it *optimal*. Sometimes it is difficult to even formulate the right criterion to optimize and even if it is formulated, it may be undecidable to find an optimal strategy.

Despite the fact both environment and the robot itself are of continuous nature, they are frequently modeled by discrete state systems. States of system may capture robot's internal configuration as well as configuration of external objects. It is then natural to express a possible change of the world by specifying ensuing states. It can be done *e.g.*, by listing all possible successors of each state. Also probability distribution above successors could be provided.

Example 1. *In [OGM+15], the authors present the space rover as an example of the autonomous system which requires a cautious proceeding. The aim of the robot is to follow the previously learned track (referred as the safe corridor). To be able to navigate itself in space, the robot is equipped with a camera for visual localization and wheel odometry sensors aiding motor controllers. The authors focus on synthesis of perception strategy which saves perception-related energy. Figure 2.2 shows the space rover together with an example of energy-optimal perception plan.*

Figure 2.2: The space rover and an example of energy-optimal perception plan from [OGM+15]. The task of the robot is to stay in the safe corridor, decorated in green.



CHAPTER 3

Formal Model

The main motivation for the problem formulated in this work is a robotic system, *e.g.*, an autonomous car driving in an urban-like environment, that involves uncertainty originating, *e.g.*, from the motion of the robot such as noisy actuators of the car or from interaction with dynamic elements in the environment such as pedestrians on streets. Typically, the system is equipped with a set of sensors, where each sensor provides a partial information about the uncertainties. As a suitable formal model for described setting we use *non-deterministic transition system with observation modes*.

The chapter is organized as follows. First, in Section 3.1, we introduce *non-deterministic transition system* (NTS), which can be used to model the motion capabilities of the robot in a partitioned environment and its interaction with the dynamic elements. This model is used when a robot is influenced by non-controllable inputs and it is hard to estimate frequency or exact impact of these inputs. Even if we have full knowledge about factors playing a considerable role in the future development of system, still it may be so complex that its future state cannot be predicted to any great degree of certainty.

In Section 3.2 we introduce *NTS with observation modes* in order to be able to capture sensing abilities of the robot. NTS with observation modes is partially observable system where accuracy of observation depends on the active observation mode. An observation mode represents a set of deployed sensors.

Finally, in Section 3.3, we formally define *strategy* for NTS with observation modes. The definition follows the fact that robot's behavior must be determined by so far made observations.

3.1 Non-deterministic Transition System

Definition 1 (NTS). *A non-deterministic transition system (NTS) is a tuple* $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$*, where*

- S *is a non-empty finite set of states,*
- A *is a non-empty finite set of actions,*
- $T\colon S \times A \to 2^S$*, is a transition function,*

- $s_{\text{init}} \in S$ *is the initial state,*
- AP *is a set of atomic propositions,*
- $L\colon S \to 2^{AP}$ *is a labeling function.*

A run of a NTS is an infinite sequence $s_0 s_1 \ldots \in S^\omega$ such that for every $i \geqslant 0$ there exists $a \in A$ with $s_{i+1} \in T(s_i, a)$. A finite run is a finite prefix of a run of the NTS.

States of NTS can represent both, the inner configuration of the robot, *e.g.*, robot's position or rotation of robotic arm, as well as the state of the world surrounding the robot, *e.g.*, position and states of other objects. Dependencies between states are expressed by transition relation. We can capture both internal and external events in transition relation.

In general, for every state there are several possible successors and we do not have any certainty about ensuing state. Non-deterministic choice represents events which cannot be controlled by robot itself. Usually, NTS is used in a way, where individual actions represent alternatives which can be controlled by the system, *e.g.*, movements of a robot. On the other hand non-determinism which occurs after taking an action represents non-controllable effects, such as external events or inaccuracy of system's abilities, *e.g.*, pedestrian might enter the road, robot after move is not precisely where it intends to be.

Without loss of generality we suppose that states cannot be distinguished based on available actions, *i.e.*, for each two states the set of available actions coincides.

3.2 NTS with Observation Modes

In practice, to deal with system's uncertainty, sensors are used to provide information about the current state of the system. For example, in mobile robotics sensors can be used to help determine robot's position or to detected pedestrians in streets. We formalize this concept by basically giving decomposition of state space to equivalence classes with respect to some observation (such as a color or temperature). Usually some kind of resources are required to obtain information from sensors, such as time for communication or energy.

Definition 2 (NTS with observation modes)**.** *A NTS with observation modes is a tuple* $(\mathcal{N}, O, M)$*, where*

- $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$ *is NTS,*
- O *is a non-empty finite set of observations,*

- *M is a non-empty finite set of observation modes. Every observation mode $m \in M$ is associated with an observation function $\gamma_m : S \to 2^O$ and a cost $g_m \in \mathbb{R}_0^+$.*

The observation modes of the NTS then represent possible subsets of sensors and the cost of an observation mode corresponds to the amount of resources such as energy or communication needed to deploy the chosen set of sensors for a single step. During executions of the system, only the observations associated with the current state of the NTS and the chosen observation mode are available. Hence, the current state of the system might not be uniquely recognized.

Example 2. *Consider an NTS $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$, where $S = \{s_1, \ldots, s_7\}$, $A = \{a, b\}$, $s_{\text{init}} = s_1$ and the transition function is as depicted in Fig. 3.1. We let $AP = \{\star, \blacktriangle\}$ and the labeling function is indicated in Fig. 3.1,* i.e., *$L(s_6) = \{\star\}$, $L(s_5) = \{\blacktriangle\}$ and $L(s_i) = \varnothing$ for every $i \neq 5, 6$. Consider three observation modes $M = \{m_1, m_2, m_3\}$ for $\mathcal{N}$ such that their respective observation functions $\gamma_1, \gamma_2, \gamma_3$ report neither the shape nor the color, only the shape and both the shape and the color of the state as shown in Fig. 3.1. Hence, the set of observations is*

$$O = \{\texttt{white}, \texttt{blue}, \texttt{red}, \\ \texttt{circle}, \texttt{rectangle}, \texttt{diamond}\}$$

For example, for state s_2 the observation functions are defined as $\gamma_1(s_2) = \varnothing, \gamma_2(s_2) = \{\texttt{rectangle}\}$ $\gamma_3(s_2) = \{\texttt{rectangle}, \texttt{blue}\}$. The costs of the observation modes are $g_1 = 0, g_2 = 1, g_3 = 2$.

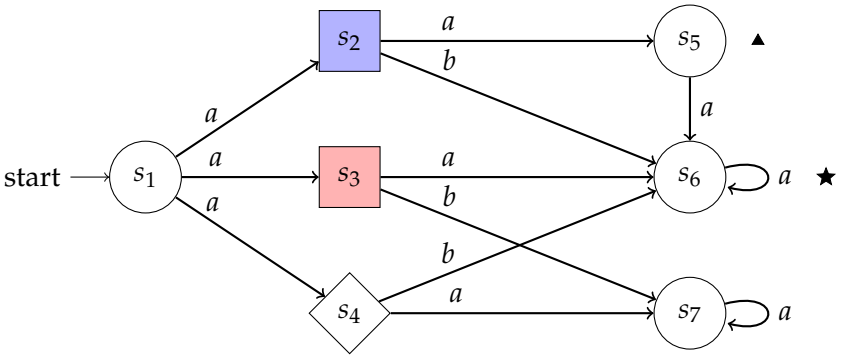


Figure 3.1: Example of an NTS with observation modes. For full description see Example 2.

A run of a NTS with observation modes is an infinite sequence $\rho = (s_0, m_0)(s_1, m_1)\ldots \in (S \times M)^\omega$ such that $s_0 s_1 \ldots$ is a run of the NTS. A finite run $\sigma = (s_0, m_0)\ldots(s_n, m_n) \in (S \times M)^*$ of the NTS with observation modes is a finite

prefix of a run. A pair $(s, m) \in S \times M$ of a state and an observation mode is called a configuration.

Given a finite run $\sigma = (s_0, m_0) \dots (s_n, m_n)$, we define the cost of σ as follows

$$g(\sigma) = \sum_{i=0}^{n} g_{m_i}. \tag{3.1}$$

The observational trace of a run $\rho = (s_0, m_0)(s_1, m_1) \dots$ is the sequence $\gamma(\rho) = \gamma_{m_0}(s_0)\gamma_{m_1}(s_1) \dots \in (2^O)^\omega$ and the propositional trace of ρ is the sequence $L(\rho) = L(s_0)L(s_1) \dots \in (2^{AP})^\omega$. The observational and propositional traces of finite runs are defined analogously.

Intuitively, the observational trace of a run ρ is a sequence of observations observed during the run. The propositional trace is a sequence of atomic propositions which held during the run.

3.3 Strategy

The robot behavior is determined by specifying the action to make and sensors to deploy while it has the choice. The decision has to be uniquely determined by observations made so far since the robot has no other means to observe its status and outside conditions.

Definition 3 (Strategy)**.** *Given a NTS with observation modes $(\mathcal{N}, O, M)$, a (observation-based control and observation scheduling) strategy is a function $C : (2^O)^* \to A \times M$ that defines the action and the observation mode to be applied in the next step based only on the sequence of past observations.*

Remark 1. *Note that this definition does not capture the case when one wants to use repeatedly different multiple observation modes in one state of NTS. Nevertheless, we can simulate this behavior with our model by adding self loop transition above every state of the NTS (under some unique action).*

We use σ_C and ρ_C to denote finite and infinite runs of the NTS $\mathcal{N}$ induced by a strategy C, respectively. Note that for every configuration (s, m), the strategy C induces a non-empty set of runs ρ_C with $\rho_C(0) = (s, m)$.

CHAPTER 4

Formal Specification

Given a model of robot's abilities and its possible interaction with outside world, it is necessary to formally describe desired properties of a system to meet design intentions. In terms of a formal model it means to identify runs which are considered as "good" and "bad" runs.

Means used to formal description of desirable properties vary depending on the used model. In connection with non-deterministic transition system temporal logic is often used, *e.g.*, Linear Temporal Logic (LTL) or Computation Tree Logic (CTL), allowing to specify temporal properties of systems such as reachability, safety, stability of response.

In this chapter we introduce Complementary Safe Linear Temporal Logic (*co-safe LTL*) as a high-level specification language. It is fragment of linear temporal logic (LTL) including those formulas which satisfaction can be guarantee in a finite time.

4.1 Complementary Safe LTL

Linear Temporal Logic (LTL) is a modal logic with modalities referring to time [Pnu77]. Formulas of LTL are interpreted over infinite words such as the propositional traces generated by runs of a NTS with observation modes.

In formal verification, *safety fragment of LTL* is widely studied, for which the problem of verification is reduced on reachability problem. The violation of the safety LTL formula can be demonstrably shown with using a finite witness of violation. Dually, for the control synthesis problem is often co-safe fragment of LTL considered. Co-safe fragment of LTL, or co-safe LTL, contains all LTL formulas such that every satisfying infinite word has a good finite prefix [KYV01]. A good finite prefix is a finite word such that every its extension to an infinite word satisfies the formula.

A class of co-safe LTL formulas that are easy to characterize are syntactically co-safe LTL formulas [Sis94]. We focused on this class of formulas since for general LTL formula the problem of deciding whether it is safety (co-safety) formula is PSPACE-complete in the size of formula [Sis94].

Definition 4 (scLTL)**.** *Syntactically co-safe LTL (scLTL) formulas over AP are the LTL formulas formed as follows:*

$$\varphi :: p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \mathbf{F}\,\varphi,$$

where $p \in AP$, $\wedge$ (conjunction) and $\vee$ (disjunction) are Boolean operators, and **X** (next)*,* **U** (until) *and* **F** (future *or* eventually) *are temporal operators.*

Remark 2. *To express properties over bounded time horizon, bounded temporal operators $\mathbf{U}_{\leqslant k}, \mathbf{F}_{\leqslant k}$ are often used in the literature. Note that these can be encoded using the operators from Definition 4.*

Definition 5 (Satisfaction relation of scLTL)**.** *The satisfaction relation $\models$ is recursively defined as follows. For a word $w \in (2^{AP})^{\omega}$, we let:*

$$\begin{aligned}
w &\models p && \Leftrightarrow && p \in w(0),\\
w &\models \neg p && \Leftrightarrow && p \notin w(0),\\
w &\models \varphi_1 \wedge \varphi_2 && \Leftrightarrow && w \models \varphi_1 \text{ and } w \models \varphi_2,\\
w &\models \varphi_1 \vee \varphi_2 && \Leftrightarrow && w \models \varphi_1 \text{ or } w \models \varphi_2,\\
w &\models \mathbf{X}\,\varphi && \Leftrightarrow && w^1 \models \varphi,\\
w &\models \varphi_1\,\mathbf{U}\,\varphi_2 && \Leftrightarrow && \text{there exists } i \geqslant 0 : w^i \models \varphi_2,\\
& && && \text{and for all } 0 \leqslant j < i : w^j \models \varphi_1\\
w &\models \mathbf{F}\,\varphi && \Leftrightarrow && \text{there exists } i \geqslant 0 : w^i \models \varphi.
\end{aligned}$$

Even though scLTL formulas have infinite-time semantics, their satisfaction is guaranteed in finite time through the concept of good finite prefixes described earlier. Given an scLTL formula φ, by $\mathsf{gfp}(\varphi)$ we denote a language of all good finite prefixes of φ. For every scLTL formula φ, the language $\mathsf{gfp}(\varphi)$ is regular [KYV01]. Thus, for every scLTL formula φ one can construct a finite automata which accepts the language $\mathsf{gfp}(\varphi)$. The construction of DFA is described in Section 4.2.

A run ρ of a NTS with observation modes satisfies an scLTL formula φ, denoted as $\rho \models \varphi$, if a propositional trace $L(\rho) \models \varphi$ or, equivalently, if there exists a finite prefix ρ^{φ} of ρ such that $L(\rho^{\varphi})$ is a good finite prefix for the formula φ. We refer to prefixes ρ^{φ} as the good finite prefixes of the run ρ for the formula φ. We say that a strategy C satisfies φ starting from a configuration (s, m) if $\rho_C \models \varphi$ for every run ρ_C such that $\rho_C(0) = (s, m)$.

4.2 Translating scLTL Formula to Finite Automata

In this section we describe translation of an scLTL formula to a finite automata, as presented in [KYV01], such that resulting automata accepts exactly those words which satisfy the formula.

Definition 6 (Büchi automata). *A non-deterministic Büchi automata (NBA) is a tuple* $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$*, where*

- Q *is a non-empty finite set of states,*
- 2^{AP} *is the alphabet,*
- $\delta : Q \times 2^{AP} \to 2^{Q}$ *is a transition function,*
- $Q_0 \subseteq Q$ *is the set of initial states,*
- $F \subseteq Q$ *is a non-empty set of accepting states.*

A run of a NBA is an infinite sequence $q_0 q_1 \ldots \in Q^\omega$ such that $q_0 \in Q_0$ and for every $i \geqslant 0$, there exists $X \in 2^{AP}$ such that $q_{i+1} \in \delta(q_i, X)$. A run $q_0 q_1 \ldots$ is called accepting if for infinitely many $i \in \mathbb{N}$ we have $q_i \in F$. A word w is accepted by NBA if it induces an accepting run. We define the set $L(\mathcal{A})$ to be the set of exactly those words which are accepted by DFA $\mathcal{A}$. A subset of states $Q' \subseteq Q$ is called *universal* whenever holds $L(\mathcal{A}') = (2^{AP})^\omega$, where $\mathcal{A}' = (Q', 2^{AP}, \delta, Q_0, F)$. Intuitively, set of states is called universal if all words are accepted when is set as a set of initial states.

Definition 7 (DFA). *A deterministic finite automata (DFA) is a tuple* $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F)$*, where*

- Q *is a non-empty finite set of states,*
- 2^{AP} *is the alphabet,*
- $\delta : Q \times 2^{AP} \to Q$ *is a transition function,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is a non-empty set of accepting states.*

A run of a DFA is a finite sequence $q_0 q_1 \ldots q_n \in Q^*$ such that for every $i \geqslant 0$, there exists $X \in 2^{AP}$ such that $q_{i+1} = \delta(q_i, X)$. Every finite word $w \in (2^{AP})^*$ induces a run of the DFA. A run is called accepting if its last state is an accepting state. A word w is accepted by the DFA if it induces an accepting run. We define the set $L(\mathcal{A})$ to be the set of exactly those words which are accepted by DFA $\mathcal{A}$.

Translation of formula φ proceeds in the following steps:

- Let $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, F)$ be NBA for φ constructed using standard techniques, *e.g.*, the one described in [BK08].
- Construct DFA $\mathcal{A}' = (2^Q, 2^{AP}, \delta', \{Q_0\}, F')$ from $\mathcal{A}$ using subset construction [BK08]. The set of accepting states F' is consisting of all universal sets of $\mathcal{A}$.

Given NBA $\mathcal{A}$. Let Q be a set of states of $\mathcal{A}$. The universality problem for $Q' \subseteq Q$ is known to be PSPACE-complete. It is proved that maximum size of resulted DFA is doubly exponential with respect to the translated scLTL formula [KYV01].

Example 3. *Consider the set of atomic propositions $AP = \{\star\}$. An example of an scLTL formula over AP is $\varphi = \mathbf{F}\,\star$. A corresponding minimal DFA $\mathcal{A}$ for φ is shown in Fig. 3.*

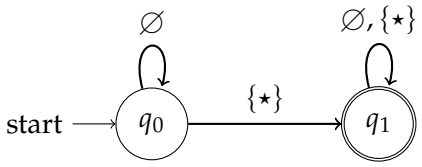


Figure 4.1: A minimal DFA for the scLTL formula $\varphi = \mathbf{F}\,\star$.

Remark 3. *In general is the size of DFA $\mathcal{A}_\varphi$ which accepts all good finite prefixes of φ is doubly exponential. However, an automata which accept at least one good prefix of every computation that does satisfy φ can be computed in single exponential time (so called fine automata) [KYV01]. For many applications it is sufficient to compute with this automata. However, for purposes of this work it is inevitable to construct DFA which accepts exactly all good finite prefixes.*

CHAPTER 5

Problem Formulation

In this chapter we formulate problems treated in this work. The main motivation for the problem formulated in this work is a robotic system, *e.g.*, a space rover driving in a dynamic environment, equipped with a high energy consumption sensors. Desirable is to economically control the robot to fulfill the given task.

First, in Section 5.1, the aim is to construct a strategy for an NTS with observation modes such that a satisfaction of assigned task, given as scLTL formula, is guaranteed and at the same time it is perception-optimal. Next, in Section 5.2, we consider the bounded version of the above problem. Last, in Section 5.3, we generalized the problem from Section 5.1 by providing operators allowing to build complex missions from tasks and at the same time supporting specification of priority rules for parts of the mission.

5.1 Optimal Task Control

We assume that the system is given a temporal objective in the form of an scLTL formula φ over the set of atomic propositions AP. From now on we refer to an scLTL formula as a *task* and we use $\text{Tasks}(AP)$ to denote the set of all tasks over AP.

Our aim is to synthesize a strategy such that any possible run meets the task, *i.e.*, we want to automatically create a plan which specifies a set of sensors to be deployed and an action to be taken in every step. At the same time it is desired to optimize the use of sensors for the worst case run. Since we cannot predict non-controllable inputs to any degree of certainty, it seems to be reasonable approach to target the optimization of the worst case.

Definition 8 (Task cost)**.** *Given a task φ, an initial configuration (s, m) and a strategy C that satisfies the task φ, we define the following cost function:*

$$V(C, (s, m), \varphi) = \max_{\rho_C, \rho_C(0)=(s,m)} \min_{\rho^\varphi, \rho^\varphi \rho' = \rho_C} g(\rho^\varphi). \tag{5.1}$$

Intuitively, the cost $V(C, (s, m), \varphi)$ of a strategy C with respect to the task φ and the configuration (s, m) is the worst-case cumulative cost of the (earliest) satisfaction of φ using C starting from configuration (s, m).

Problem 1 (Optimal task control). *Given*

- *a NTS with observation modes* $(\mathcal{N}, O, M)$*, where* $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$*,*
- *an initial observation mode* $m_{\text{init}} \in M$*,*
- *a task,* i.e.*, an scLTL formula,* φ *over* AP*,*

find an observation-based control and observation scheduling strategy C *such that*

1. C *satisfies* φ *starting from the configuration* $(s_{\text{init}}, m_{\text{init}})$*,*
2. *the cost* $V(C, (s_{\text{init}}, m_{\text{init}}), \varphi)$ *is minimized over all strategies satisfying* φ *starting from the configuration* $(s_{\text{init}}, m_{\text{init}})$*.*

5.2 Bounded Optimal Task Control

It might be desired to fulfill the task completion within some bound on number of steps taken in the transition system. Note that if the system satisfy the task then there exists a finite bound such that the completion of the task is always guaranteed within this bound.

Problem 2 (Bounded optimal task control). *Given*

- *NTS with observation modes* $(\mathcal{N}, O, M)$*, where* $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$*,*
- *an initial observation mode* $m_{\text{init}} \in M$*,*
- *a task* φ *over* AP*,*
- *a finite bound* $k \geqslant 0$*,*

find an observation-based control and observation scheduling strategy C *such that*

1. C *satisfies* φ *starting from the configuration* $(s_{\text{init}}, m_{\text{init}})$ *in at most* k *steps, and,*
2. *the cost* $V(C, (s_{\text{init}}, m_{\text{init}}), \varphi)$ *is minimized over all strategies satisfying* φ *starting from the configuration* $(s_{\text{init}}, m_{\text{init}})$ *in at most* k *steps.*

Example 4. *Consider the NTS with observation modes introduced in Example 2 with initial observation mode* m_1 *and the task from Example 3 that requires to reach the state labeled with* $\star$*,* i.e.*, state* s_6*.*

Note that only in states s_2, s_3, s_4 *there is more than one action allowed and hence it suffices to discuss strategies based on their decision in these states. No strategy* C *with* $C(\varnothing) = (a, m_1)$*,* i.e.*, a strategy that applies action* a *starting from the initial state* s_1 *and activates observation mode* m_1 *in the next state, can guarantee satisfaction of the*

formula. The reason is that the three states s_2, s_3, s_4 cannot be told apart using mode m_1 and both actions a, b always in at least one case lead to state s_7 from which s_6 cannot be reached.

Consider strategy C_1 that recognizes the shape of the three states s_2, s_3, s_4, i.e.,

$$\begin{aligned} C_1(\varnothing) &= (a, m_2), \\ C_1(\varnothing\{\texttt{rectangle}\}) &= (a, m_1), \\ C_1(\varnothing\{\texttt{diamond}\}) &= (b, m_1). \end{aligned}$$

Strategy C_1 guarantees a visit to s_6 in at most 3 steps and its cost $V(C_1, (s_{\text{init}}, m_1), \varphi) = 1$.

Alternatively, consider strategy C_2 that recognizes both the shape and the color of the three states s_2, s_3, s_4, i.e.,

$$\begin{aligned} C_1(\varnothing) &= (a, m_3), \\ C_1(\varnothing\{\texttt{rectangle}, \texttt{blue}\}) &= (b, m_1), \\ C_1(\varnothing\{\texttt{rectangle}, \texttt{red}\}) &= (a, m_1), \\ C_1(\varnothing\{\texttt{diamond}, \texttt{white}\}) &= (b, m_1). \end{aligned}$$

Strategy C_2 guarantees a visit to s_6 in 2 steps and its cost $V(C_2, (s_{\text{init}}, m_1), \varphi) = 2$. Strategy C_1 is the solution to the optimal scLTL control Problem 1 as its cost is lower than the cost of C_2. However, if we consider the bounded optimal scLTL control Problem 2 with $k = 2$, then C_2 is the solution as C_1 may need more than 2 steps to reach s_6.

5.3 Optimal Mission Control

In this section we focus on instances when either desired behavior cannot be completely guaranteed, *e.g.*, due to environment constraints, or when there are more permissible tasks. In the former case, it still might be useful to partially fulfill some of the requirements. In the latter case, we can search for a compromise between fulfillment of the most valuable tasks and the cost paid for sensor usage.

To be able to prioritize some task before other we assign to each task a reward expressing how much is valuable to fulfill it. With the alternating operator we let user to specify which possibilities are enabled. Concatenation operator is provided to facilitate the expression of requirements.

Definition 9 (Mission)**.** *A mission over AP is a regular expression [BK08] excluding a Kleene star over the set of all tasks* $\text{Tasks}(AP)$ *over AP,* i.e., *an expression formed as follows:*

$$\phi := \varphi \mid \phi + \phi \mid \phi \cdot \phi,$$

where $\varphi \in \text{Tasks}(AP)$ is a task, $+$ (alternation) and $\cdot$ (concatenation) are operators.

Every mission ϕ, as a regular expression, describes a set $\text{seq}(\phi) \subseteq \text{Tasks}(AP)^*$ of finite sequences of tasks, defined recursively as follows:

$$\begin{aligned} \text{seq}(\varphi) &= \{\varphi\} \\ \text{seq}(\phi_1 + \phi_2) &= \text{seq}(\phi_1) \cup \text{seq}(\phi_2), \\ \text{seq}(\phi_1 \cdot \phi_2) &= \text{seq}(\phi_1) \cdot \text{seq}(\phi_2). \end{aligned}$$

Recall that for a task φ is $\mathsf{gfp}(\varphi)$ the set of all good finite prefixes of φ. We define $\mathsf{mgfp}(\varphi)$ as the set of the minimal good finite prefixes of φ:

$$\mathsf{mgfp}(\varphi) = \{p \in \mathsf{gfp}(\varphi) \mid \text{no strict prefix of } p \text{ is in } \mathsf{gfp}(\varphi)\}.$$

Definition 10 (Satisfaction relation of mission)**.** *An infinite word $w \in (2^{AP})^\omega$ satisfies a mission ϕ, denoted as $w \models \phi$, if there exists a sequence $\pi \in \text{seq}(\phi)$, where $|\pi| = n$, and indices $0 = i_0 < i_1 < \ldots < i_n$ such that for every $0 \leqslant j < n$, the finite word $w(i_j) \ldots w(i_{j+1} - 1) \in \mathsf{mgfp}(\pi(j))$.*

Note that the satisfaction relies on the finite word $w(0) \ldots w(i_{n+1} - 1)$ and on a sequence π that maps onto it. Just like a task, every mission can be characterized by a set of good finite prefixes. Since the language of good finite prefixes is regular, we can represent every mission as DFA as we did in the case of tasks. The proof is given in Section 7.3.1.

A run ρ of a NTS with observation modes satisfies a mission ϕ, denoted as $\rho \models \phi$, if a propositional trace $L(\rho) \models \phi$ or, equivalently, if there exists a finite prefix ρ^ϕ of ρ such that $L(\rho^\phi)$ is a good finite prefix for the formula ϕ. We refer to prefixes ρ^ϕ as the good finite prefixes of the run ρ for the mission ϕ. We say that a strategy C satisfies ϕ starting from a configuration (s, m) if $\rho_C \models \phi$ for every run ρ_C such that $\rho_C(0) = (s, m)$.

As we mentioned above, we associate a non-negative reward with every task while composing a mission. It expresses how much valuable the satisfaction of some part of the mission is. It is meaningful in combination with the alternation operator where the strategy can choose to fulfill a particular part of the mission.

Definition 11 (Mission cost)**.** *Given a mission ϕ, an initial configuration (s, m), a strategy C that satisfies the mission ϕ and a reward function $r : \text{Tasks}(AP) \to \mathbb{R}_0^+$. Consider the following definition of a* mission cost*:*

$$V_r(C, (s, m), \phi) = \max_{\rho_C, \rho_C(0)=(s,m)} \min_{\pi \in seq_{\rho_C}(\phi), \rho_C \models \pi} g(\pi, \rho_C), \tag{5.2}$$

where

$$g(\pi, \rho_C) = -\sum_{i=0}^{|\pi|} r(\pi(i)) + \min_{\rho^\pi, \rho^\pi \rho' = \rho_C} g(\rho^\pi),$$

Intuitively, the cost $V_r(C, (s, m), \phi)$ of a strategy C is the worst-case price for a run under strategy C. The price of a run is computed as the cumulative cost for sensor usage minus reward gained during the run.

Problem 3 (Optimal mission control)**.** *Given*

- *NTS with observation modes* $(\mathcal{N}, O, M)$*, where* $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$*,*
- *an initial observation mode* $m_{\text{init}} \in M$*,*
- *a mission* ϕ *over* AP*,*
- *a reward function* $r : \text{Tasks}(AP) \to \mathbb{R}$*,*

find an observation-based control and observation scheduling strategy C *such that*

1. *C satisfies ϕ starting from the configuration $(s_{\text{init}}, m_{\text{init}})$,*
2. *the cost $V_r(C, (s_{\text{init}}, m_{\text{init}}), \phi)$ is minimized over all strategies satisfying ϕ starting from the configuration $(s_{\text{init}}, m_{\text{init}})$.*

Remark 4. *The reason for not allowing Kleene star in missions is to ensure that there exists an optimal solution for the problem that we intend to solve. Moreover, later we also discuss relaxed version of this problem allowing Kleene star which turns out to be undecidable.*

Example 5. *Consider the NTS with observation modes introduced in Example 2 with initial observation mode m_1, the tasks $\varphi_1 = \mathbf{F} \star$ with reward 1, $\varphi_2 = \mathbf{F} \blacktriangle$ with reward 3 and finally $\varphi_3 = \neg \blacktriangle \mathbf{U} \star$ also with reward 3. We compose a mission $\phi = \varphi_1 + \varphi_2 + \varphi_3$. One can see that the task φ_2 is not feasible since due to non-determinism no strategy can guarantee avoidance of the state s_4 and the state s_5 is not reachable from there.*

Consider strategy C_1 from Example 4 that recognizes the shape of the three states s_2, s_3, s_4. Strategy C_1 guarantees a visit to s_6, i.e.*, to fulfill task φ_1 and hence to complete the mission ϕ. We cannot guarantee avoidance of the state s_5,* i.e.*, we cannot guarantee to fulfill the task φ_3, since we cannot distinguish the state s_2 from the state s_3. Hence cost $V_r(C_1, (s_{\text{init}}, m_1), \phi)$ is equal to 0.*

Further, consider strategy C_2 from Example 4 that recognizes both the shape and the color of the three states s_2, s_3, s_4. Strategy C_2 guarantees an avoidance of the state s_5 and at the same time in guarantee a visit to s_6. Hence its cost is $V_r(C_2, (s_{\text{init}}, m_1), \varphi) = -1$.

Strategy C_2 is the solution to the Problem 3 as its cost is lower than the cost of C_1.

5.4 General Mission

As the next step following formulated problems could be to consider more general form of mission. In Section 5.3 we defined mission as regular expression excluding Kleene star over tasks. We might also be interested in full-featured regular expression over tasks, *i.e.*, including also Kleene star.

Definition 12 (General Mission Control)**.** *A general mission over AP is a regular expression over the set of all tasks* $\mathrm{Tasks}(AP)$, i.e., *an expression formed as follows:*

$$\phi_G := \varphi \mid \phi_G + \phi_G \mid \phi_G \cdot \phi_G \mid \phi_G^*,$$

*where $\varphi \in \mathrm{Tasks}(AP)$ is a task, $+$ (alternation),$\cdot$ (concatenation) and * (Kleene star) are operators.*

In the same way as every mission ϕ describes a set $\mathrm{seq}(\phi)$, the general mission describes a set $\mathrm{seq}(\phi_G)$. We need only declare how is defined $\mathrm{seq}(\phi_G^*)$:

$$\mathrm{seq}(\phi_G^*) = \mathrm{seq}(\phi_G)^*$$

The satisfaction relation is also defined in a similar way as for mission using sets $\mathrm{seq}(\cdot)$. Also mission cost $V_r(\cdot)$, see Equation 5.2, can be easily adopted for general missions.

In Remark 4 we pointed the fact that the reason to not allowing Kleene star is to ensure the existence of the optimal solution for Problem 3. In order to define well the problem, we relax former requirements and in a new setting we are interested in finding a surely winning strategy which ensures the cost is at most some given bound.

Problem 4 (General mission control)**.** *Given*

- *NTS with observation modes $(\mathcal{N}, O, M)$, where $\mathcal{N} = (S, A, T, s_{\mathrm{init}}, AP, L)$,*
- *an initial observation mode $m_{\mathrm{init}} \in M$,*
- *a general mission ϕ_G over AP,*
- *a reward function $r : \mathrm{Tasks}(AP) \to \mathbb{R}$,*
- *a real value bound b*

find an observation-based control and observation scheduling strategy C such that

1. *C satisfies ϕ_G starting from the configuration $(s_{\mathrm{init}}, m_{\mathrm{init}})$,*
2. *the cost $V_r(C, (s_{\mathrm{init}}, m_{\mathrm{init}}), \phi_G)$ is at most b*

CHAPTER 6

Related Work

In this chapter, we provide a quick overview of the research in the field of the control synthesis closely related to the problems treated in this work. The aim of this chapter is not to provide exhausting survey through the literature, rather it more presents in detail referred works and works which are mostly related to ours to the best of our knowledge. We address several papers varying so by used formalism for modeling of the system, so by examined objectives to achieve. For each paper, we state the most significant differences from our work.

6.1 Optimizing Single Sensor Usage

In this section we present papers where authors deal with problem of minimizing usage of single sensor in order to achieve the established goal.

6.1.1 Space Rover

First, we present the work [OGM+15] which served mostly as an inspiration for our work. We have already provided quick insight into the problem in Example 1. A problem of reducing a robot's energy consumption while following a trajectory by turning off the main localization subsystem and switching to a lower-powered, less accurate odometry source. Robot is allowed to make small deviation from original trajectory, but it ought to stay in so called *safe corridor*.

More formally, the possible robot placement is represented as a belief Markov Decision Process [TBF05] (belief MDP) and the amount of released energy is represented by a cost function over edges of the system. States in the belief MDP represent possible deflection from robot trajectory at the given point, *i.e.*, the robot has only partial information about current position. At every time step the robot has a choice to either localize or not. If the robot choose to localize itself, it is given a precise information about the current trajectory deviation, *i.e.*, it obtains a precise information about its location. The aim is to synthesize a plan minimizes the energy consumption attributed to the localization system and at the same time provide a probabilistic guarantee to stay in the safe corridor.

To approach the proposed problem, the authors present two algorithms. First simple algorithm let sensors turned off as long as the probability of going out of

the safe corridor is sufficiently small. This probability can be directly estimated from the current state of belief MDP. If the robot is exposed to the appreciable chance of leaving the corridor, it turns on the main localization system and returns to the original track. In the second algorithm, they leverage dynamic programming techniques and the exact solution is not as relevant for our work.

Despite the fact this work considers similar problem as ours, presented results are not applicable in this work. First, they consider single type of sensor and at any instant time they can reveal exact state of the system. Second, they suppose a belief MDP to be in a highly specialize form (for details see the original paper) so it is not applicable to wide range problems studied in this work.

6.1.2 Minimal Disclosure in POMDP

Partially Observable Markov Decision Processes (POMDP) is well studied partially observable generalization of MDP [Put94]. It can be seen also as probabilistic counterpart to NTS with observation modes presented in this work. In [BG11], the authors tackle the problem of the minimal information a user needs to achieve a simple goal. In brief, given a POMDP with single observation mode able to reveal the exact current state and a *Goal* state, the aim is to find an optimal strategy which guarantee visit to a *Goal* with probability 1. They consider two optimality criteria, (i) minimize number of steps made in the worst case run, (ii) Minimize expected number of steps during the run.

They show that problem with the first criterion is EXPTIME-complete whereas the problem considering the second criterion turns out to be undecidable, nor there exists approximative algorithm which solves it.

Despite the fact POMDP is basically a generalization of NTS with observation modes, the paper [BG11] cannot be seen as an extension of our work. Reason is apparent, they consider only single observation mode which allowing to reveal full information.

6.2 Optimizing Multiple Sensor Usage

In this section we mention two papers [CMH08,CM11], where the authors focus on control of discrete systems with multiple observation modes with respect to properties in infinite time horizon. We focus only on the work [CMH08].

Both presented papers are from the field of game theory, where is widely studied problem of *games with imperfect information*. In the context of presented papers, a *game structure* can be seen as a *NTS with observation modes* defined in Section 3.2. A game is played as follows, in each turn, Player 1 chooses an action to take together with an observation mode to deploy and Player 2 resolves non-determinism by choosing the successor state. So far, most works make the

assumption of fixed partial information, *i.e.*, in terms of NTS with observation modes, there is only one observation mode available.

6.2.1 Synthesis With Budget Constraints

The motivation for the paper [CMH08] is design and implementation of controllers for resource-constrained embedded systems, where a controller may not have enough power, time, or bandwidth to obtain data from all sensors in each round.

As mentioned above, a game of imperfect information with multiple observation modes is used to model the problem. A given fixed budget cost is associated with making an observation, and a controller can make only a limited number of observations in each round so that the total cost of the used observation modes does not exceed a given fixed budget.

The authors study several problems. The general problem is formulated as follows, given a game with imperfect information, a budget constraint B, and ω-regular objective [BL90], the aim is to find out the minimum budget with which a controller can achieve its goals. There are several optimization criteria discussed in the work, *e.g.*, to minimize a long-run average sensing cost or to minimize the worst case cost paid at any single round. Also different types of ω-regular objectives are considered.

Since problems examined in the paper operate with infinite runs and optimization criterion is defined also over infinite horizon, we don't see a straightforward way how to apply presented results in this work.

CHAPTER 7

Problem Solution

In this chapter we present solutions to problems proposed in Chapter 5. All presented solutions follows the same principle, presented in detail in Section 7.1.

7.1 Optimal Task Control

In this section, we describe the algorithm to solve Problem 1 in detail. To approach the problem, we leverage automata-based model checking techniques that analyze the state space using graph algorithms. First, we construct a synchronous product of the NTS $\mathcal{N}$ and a a DFA $\mathcal{A}_\varphi$ for the task φ, where the runs of the NTS satisfying the formula can be easily identified through accepting states of the DFA. Next, to account for the non-determinism and partial observation, we use a belief construction over the product that determines the set of possible current states of the product given any finite sequence of past observations. Using graph algorithms, we construct a strategy for the belief product that guarantees a visit of an accepting state and minimizes a function derived from the costs of the associated observation modes. Finally, we map the strategy from the belief product to the original NTS and prove that the resulting strategy is a solution to presented problem.

7.1.1 Product

Definition 13 (Product). *Let $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$ be a NTS and $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F)$ be a DFA. The synchronous product is a tuple*

$$\mathcal{P} = \mathcal{N} \times \mathcal{A} = (S \times Q, A, T_{\mathcal{P}}, (s_{\text{init}}, q_0), AP, F_{\mathcal{P}}),$$

where

- *$S \times Q$ is the set of states,*
- *A is the alphabet,*
- *$T_{\mathcal{P}} \colon S \times Q \times A \to 2^{S \times Q}$ is a transition function such that a state $(s', q') \in T_{\mathcal{P}}((s, q), a)$ if and only if $s' \in T(s, a)$ and $\delta(q, L(s)) = q'$,*

- (s_{init}, q_0) *is the initial state,*
- $F_{\mathcal{P}} = \{(s, q) \mid q \in F\}$ *is the set of product accepting states.*

Note that the product can be seen as an NTS with a set of accepting states. This allows us to adopt the definitions of an infinite and finite runs for the product as well as a notion of an accepting finite run.

We abuse the notation by using γ_α to denote the observation function of a sensor $\alpha \in \Theta$ as well as its extension to $S \times Q$, *i.e.*, $\gamma_\alpha((s, q)) = \gamma_\alpha(s)$ for all $(s, q) \in S \times Q$.

Example 6. *In Fig. 7.1, we depict the product constructed for the NTS with observation modes presented in Example 2 and the DFA from Example 3.*

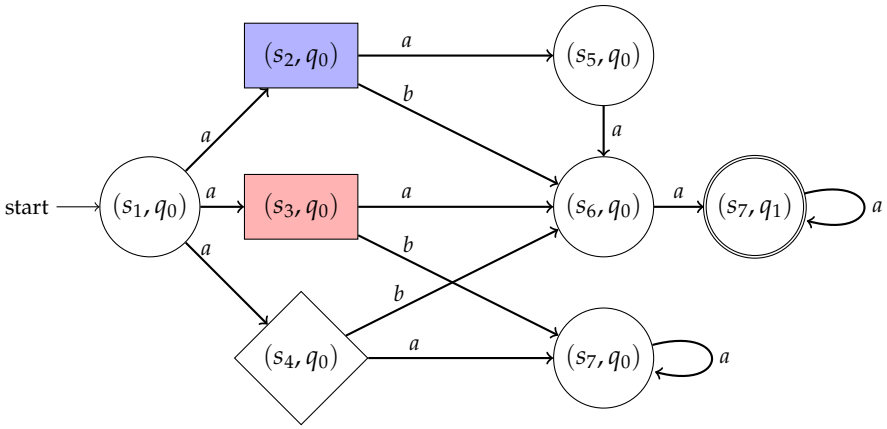


Figure 7.1: Product of the NTS from Example 2 and the DFA from Example 3.

7.1.2 Weighted Belief Product

The belief construction over the product follows the standard principles used for partially observable systems. Besides keeping track of the states that the product can currently be in, we also keep track of the observation mode deployed in the current state.

Definition 14 (Weighted belief product)**.** *Given a product* $\mathcal{P} = (S \times Q, A, T_{\mathcal{P}}, (s_{\text{init}}, q_0), AP, L_{\mathcal{P}}, F_{\mathcal{P}})$ *built over the NTS with observation modes* $(\mathcal{N}, O, M)$ *with the initial observation mode* m_{init}*, we define the weighted belief product*

$$\mathcal{B} = (\mathbf{B}, \mathbf{A}, T_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, F_{\mathcal{B}}, w)$$

over $\mathcal{P}$*, where*

- $\mathbf{B} \subseteq 2^{S \times Q}$ *is the set of all belief states, where a belief state* $\mathbf{b} \in \mathbf{B}$ *is a set of product states such that there exists an observation mode* $m \in M$ *such that all states in* $\mathbf{b}$ *have the same observations in mode* m*,*

- $\mathbf{A} = A \times M$ *is the set of belief actions of the form* $\mathbf{a} = (a, m)$*, where* $a \in A$ *is an action of* $\mathcal{P}$ *and* $m \in M$ *is an observation mode,*

- $T_{\mathcal{B}} : \mathbf{B} \times \mathbf{A} \to 2^{\mathbf{B}}$ *is the transition function such that a belief state* $\mathbf{b}' \in T_{\mathcal{B}}(\mathbf{b}, (a, m'))$ *if and only if* $\mathbf{b}'$ *is the set of all product states that can be reached in one step from a state in* $\mathbf{b}$ *using action* a *and have the same observations in mode* m'*,*

- $\mathbf{b}_{\text{init}} = \{(s_{\text{init}}, q_0)\}$ *is the initial state,*

- $F_{\mathcal{B}} = \{\mathbf{b} \mid \mathbf{b} \subseteq F_{\mathcal{P}}\}$ *is the set of accepting belief states,*

- $w : \mathbf{B} \times \mathbf{A} \to \mathbb{R}_0^+$ *is the weight function such that* $w(\mathbf{b}, (a, m)) = g_m$*.*

Weighted belief product can be seen as a NTS with a set of accepting states and a weight function on transitions. We adopt definitions of finite and infinite runs of the belief product and an accepting finite run.

Corollary 1. *From the definition of the weighted belief product* $\mathcal{B}$ *it follows that every finite run of* $\mathcal{B}$ *corresponds to exactly one finite sequence of observations in* $(2^O)^*$ *and at the same time, every finite sequence of observations in* $(2^O)^*$ *corresponds to at most one finite run of* $\mathcal{B}$*.*

Example 7. *In Fig.. 7.2, we depict part of the weighted belief product constructed for the NTS with observation modes presented in Example 2 and the DFA from Example 3.*

Definition 15 (Strategy)**.** *Given a weighted belief product* $\mathcal{B} = (\mathbf{B}, \mathbf{A}, T_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, F_{\mathcal{B}}, w)$*, a strategy for* $\mathcal{B}$ *is a function* $C \colon \mathbf{B}^* \to \mathbf{A}$*.*

We call a strategy C memoryless if it can be defined as a function $C \colon \mathbf{B} \to \mathbf{A}$. If the context is clear, we use σ_C and ρ_C to denote finite and infinite runs of $\mathcal{B}$ induced by a strategy C, respectively.

7.1.3 Strategy for the Weighted Belief Product

In this section, we propose an algorithm that constructs a memoryless strategy for the weighted belief product that guarantees a visit to an accepting state (if such a strategy exists) and minimizes the worst-case cumulative weight. We prove that such a strategy then maps to a strategy for the original NTS with

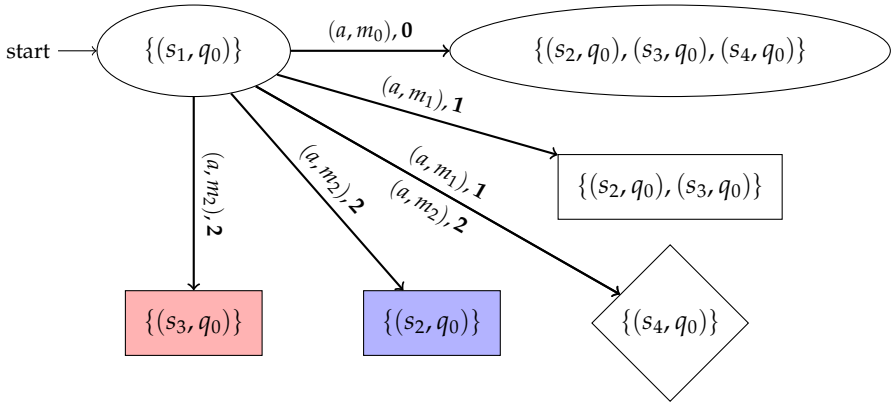


Figure 7.2: Part of the weighted belief product for the NTS with observation modes presented in Example 2 and the DFA from Example 3. The costs of individual belief actions are written in bold.

observation modes that solves Problem 1. The algorithm can be seen as a combination of the standard algorithm for computing winning states in non-deterministic systems [BK08] and Dijkstra's algorithm for computing shortest paths in a weighted graph [CSRL01].

In the algorithm, we incrementally compute a value $\mathrm{wtg}(\mathbf{b})$ (weight-to-go) for every belief state $\mathbf{b}$ that is the minimum worst case weight of reaching an accepting state starting from $\mathbf{b}$. Initially, the value is 0 for accepting belief states and ∞ otherwise. We use W_i to denote the set of belief states for which the value $\mathrm{wtg}(\mathbf{b}) \neq \infty$ after i-th iteration. In i-th iteration, we consider the belief state $\mathbf{b}_{\min} \in \mathbf{B} \setminus W_{i-1}$ and its action $\mathbf{a}_{\mathbf{b}_{\min}}$ that leads to the set W_{i-1} and minimizes the worst-case sum of the weight of the action and the value wtg of a successor state. The algorithm terminates when the initial belief state $\mathbf{b}_{\mathrm{init}}$ is added to the set W_i or when there exists no state $\mathbf{b} \in \mathbf{B} \setminus W_i$ with an action leading to W_i. If the resulting set W_i contains the initial belief state, the strategy consisting of the above actions for each belief state in W_i is returned. The algorithm is summarized in Algorithm 1.

Proposition 1 (Correctness). *Algorithm 1 results in a strategy C for the weighted belief product such that every run under C that starts in $\mathbf{b}_{\mathrm{init}}$ eventually visits an accepting belief state, if such a strategy exists.*

Proof. The property can be proved by induction on the iteration counter i and proving that starting from the state $\mathbf{b}_i \in W_i, \mathbf{b}_i \notin W_{i-1}$, strategy C guarantees a visit to an accepting belief state in at most i steps. □

Algorithm 1 Constructing a strategy for the weighted belief product that maps to a solution of Problem 1.

Require: $\mathcal{B} = (\mathbf{B}, \mathbf{A}, T_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, F_{\mathcal{B}}, w)$
Ensure: memoryless strategy C for the belief product $\mathcal{B}$

$W_0 := F_{\mathcal{B}}$
$\forall\, \mathbf{b} \in W_0 :\ \text{wtg}(\mathbf{b}) := 0$
$\forall\, \mathbf{b} \in (\mathbf{B} \backslash W_0) :\ \text{wtg}(\mathbf{b}) := \infty$
$i := 1$
while $\mathbf{b}_{\text{init}} \notin W_i$ and exist $\mathbf{b} \in \mathbf{B} \backslash W_{i-1}$ and $\mathbf{a} \in \mathbf{A}$ such that $\varnothing \neq T_{\mathcal{B}}(\mathbf{b}, \mathbf{a}) \subseteq W_{i-1}$ **do**
  $\mathbf{b}_{\min} := \bot \qquad \mathbf{a}_{\min} := \bot \qquad \Delta_{\min} := \infty$
  for every $\mathbf{b} \in \mathbf{B} \backslash W_{i-1}$ and $\mathbf{a} \in \mathbf{A}$ such that $\varnothing \neq T_{\mathcal{B}}(\mathbf{b}, \mathbf{a}) \subseteq W_{i-1}$ **do**
    $\Delta := \max_{\mathbf{b}' \in T_{\mathcal{B}}(\mathbf{b}, \mathbf{a})} \{w(\mathbf{b}, \mathbf{a}) + \text{wtg}(\mathbf{b}')\}$
    if $\Delta < \Delta_{\min}$ **then**
      $\mathbf{b}_{\min} := \mathbf{b} \qquad \mathbf{a}_{\min} := \mathbf{a} \qquad \Delta_{\min} := \Delta$
    end if
  end for
  $W_i := W_{i-1} \cup \{\mathbf{b}_{\min}\}$
  $C(\mathbf{b}_{\min}) := \mathbf{a}_{\min}$
  $\text{wtg}(\mathbf{b}_{\min}) := \Delta_{\min}$
  $i := i + 1$
end while
if $\mathbf{b}_{\text{init}} \in W_i$ **then**
  return C
else
  return no suitable strategy exists
end if

Proposition 2 (Optimality)**.** *Let C be the strategy resulting from Algorithm 1. Then among all strategies that guarantee a visit to an accepting belief state, C minimizes the value*

$$V_{\mathcal{B}}(C, \mathbf{b}_{\text{init}}) = \max_{\rho_C, \rho_C(0) = \mathbf{b}_{\text{init}}} \ \min_{\rho^{\text{acc}} \rho' = \rho_C} w(\rho^{\text{acc}}, C) \tag{7.1}$$

where ρ^{acc} is a finite run ending in an accepting state and

$$w(\rho^{\text{acc}}, C) = \sum_{i=0}^{|\rho^{\text{acc}}|-2} w\big(\rho^{\text{acc}}(i), C(\rho^{\text{acc}}(0) \dots \rho^{\text{acc}}(i))\big).$$

Intuitively, the value $V_{\mathcal{B}}(C, \mathbf{b})$ of a strategy C with respect to a belief state $\mathbf{b}$ is the worst-case cumulative weight of the (earliest) visit to an accepting state using C starting from $\mathbf{b}$.

Proof. We show by induction that after every iteration $i \geqslant 1$, it holds that $V_{\mathcal{B}}(C, \mathbf{b}_i) = \text{wtg}(\mathbf{b}_i)$ for all states $\mathbf{b}_i \in W_i$, *i.e.*, the strategy C realizes the values $\text{wtg}(\mathbf{b}_i)$,

and that the strategy C minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_i)$ among all strategies that guarantee visit to an accepting state.

Assume that the strategy C is computed in n iterations of the "while" cycle in line 4, *i.e.*, W_n is the resulting fixed point set. Consider a belief state $\mathbf{b}_i$ that was added to the set W_n in i-th iteration, *i.e.*, $\mathbf{b}_i \notin W_{i-1}, \mathbf{b}_i \in W_i$. Assume that for all $j < i$ it holds that C minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_j)$ for every $\mathbf{b}_j \in W_j$ and that $V_{\mathcal{B}}(C, \mathbf{b}_j) = \mathrm{wtg}(\mathbf{b}_j)$. Trivially, C minimizes the value for all accepting belief states $\mathbf{b} \in F_{\mathcal{B}}$ as $V_{\mathcal{B}}(C, \mathbf{b}) = 0 = \mathrm{wtg}(\mathbf{b})$. We show that C then also minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_i)$ over all strategies and $V_{\mathcal{B}}(C, \mathbf{b}_i) = \mathrm{wtg}(\mathbf{b}_i)$.

Assume by contradiction that there exists a (possibly not memoryless) strategy C' for $\mathcal{B}$ such that $V_{\mathcal{B}}(C', \mathbf{b}_i) < V_{\mathcal{B}}(C, \mathbf{b}_i)$. As C is optimal for all $\mathbf{b}_j, j < i$, it must hold that

$$V_{\mathcal{B}}(C', \mathbf{b}_j) = \mathrm{wtg}(\mathbf{b}_j) = V_{\mathcal{B}}(C, \mathbf{b}_j). \tag{7.2}$$

Let $\mathbf{b} \notin W_{i-1}$ be a belief state such that there exists a run $\sigma_{C'}$ under C' that leads from $\mathbf{b}_i$ through $\mathbf{b}$ to an accepting belief state and $T_{\mathcal{B}}(\mathbf{b}, C'(\sigma^{\mathbf{b}}_{C'})) \subseteq W_{i-1}$, where $\sigma^{\mathbf{b}}_{C'}$ is a prefix of $\sigma_{C'}$ ending in the state $\mathbf{b}$. Note that such $\mathbf{b}$ must exist since C' guarantees a visit to an accepting state and $F_{\mathcal{B}} \subseteq W_{i-1}$ (be aware that $\mathbf{b}$ can be $\mathbf{b}_i$ itself). Since the cumulative weight $w(\sigma^{\mathbf{b}}_{C'}, C')$ is non-negative and the action $C(\mathbf{b}_i)$ minimizes the value in line 7, it holds that the cumulative weight $w(\sigma_{C'}, C')$ is higher or equal to the cumulative weight of any run σ_C under C starting in $\mathbf{b}_i$ leading to an accepting belief state. Hence $V_{\mathcal{B}}(C', \mathbf{b}_i) \geqslant V_{\mathcal{B}}(C, \mathbf{b}_i)$ and strategy C is optimal. □

7.1.4 Strategy for NTS

Let $C_{\mathcal{B}}$ be the strategy for the weighted belief product $\mathcal{B}$ resulting from Algorithm 1. Consider the following (observation-based control and observation scheduling) strategy C for the NTS $\mathcal{N}$ with observation modes M. For a finite sequence of observations $\sigma_O \in (2^O)^*$, we define

$$C(\sigma_O) = C_{\mathcal{B}}(\mathbf{b}), \tag{7.3}$$

where $\mathbf{b}$ is the last state of the finite run $\sigma_{\mathcal{B}}$ of the belief product that corresponds to σ_O as described in Corollary 1, if such run exists.

Theorem 1. *Let $C_{\mathcal{B}}$ be the strategy for the weighted belief product $\mathcal{B}$ resulting from Algorithm 1. Then the strategy C for the NTS with observation modes constructed according to Equation 7.3 is a solution to Problem 1.*

Proof. The correctness with respect to the task φ follows directly from Proposition 1. The optimality of C follows from Proposition 2 and the fact that

$$V(C, s_{\text{init}}, m_{\text{init}}), \varphi) = V_{\mathcal{B}}(C_{\mathcal{B}}, \mathbf{b}_{\text{init}}).$$

□

7.1.5 Complexity

Given an scLTL formula φ, the number of states of a corresponding minimal DFA $\mathcal{A}$ is in general doubly exponential in the size of the formula. However, compared to the size of the NTS, the size of the automata typically does not play a crucial role in the overall complexity. The product $\mathcal{P}$ of the NTS $\mathcal{N}$ and $\mathcal{A}$ is then of size $\mathcal{O}(|S| \times |Q|)$. The belief product $\mathcal{B}$ involves a subset construction over the product, hence its size is in $\mathcal{O}(2^{|S|\times|Q|})$. In order to minimize the complexity in practice, only the reachable states of both the product and the belief product are constructed. With a proper choice of a data structure storing the belief product $\mathcal{B}$, Algorithm 1 runs in time $\mathcal{O}(|\mathbf{B}| \cdot \log |\mathbf{B}| + |\mathbf{A}| \cdot \mathrm{dn})$, where dn is the degree of non-determinism of the NTS $\mathcal{N}$, *i.e.*, the maximum number of possible successors given a state and an action. Note that while the algorithms are polynomial with respect to their input, *i.e.*, the belief product, they are exponential in the size of the original NTS.

7.2 Bounded Optimal Task Control

In this section we describe the algorithm to solve Problem 2. In order to solve the problem, we proceed as follows. As in the case for the general problem, we first construct the product $\mathcal{P}$ of the NTS $\mathcal{N}$ with a DFA $\mathcal{A}$ for the scLTL formula φ and the corresponding belief product $\mathcal{B}$ as proposed in Section 7.1.1 and 7.1.2, respectively. To compute a strategy for the belief product from Section 7.1.3, we use an alternation of Algorithm 1 presented below and summarized as Algorithm 2. Intuitively, as Algorithm 1 builds on the principles of Dijkstra's algorithm, Algorithm 2 follows the idea behind Bellman-Ford algorithm for solving the bounded shortest path problem in weighted graphs [CSRL01]. We prove properties of the resulting strategy $C_{\mathcal{B}}$ for the belief product and argue that when mapped to the original system as described in Section 7.1.4, we obtain a correct and optimal solution to Problem 2.

7.2.1 Strategy for the Weighted Belief Product

In this section, we give description of an algorithm that constructs a strategy for the weighted belief product that guarantees a visit to an accepting belief state within k steps (if such a strategy exists) and minimizes the worst-case cumulative weight.

Instead of computing the weight-to-go value $\mathrm{wtg}(\mathbf{b})$ for a single well-chosen belief state $\mathbf{b}$ at a time as in Algorithm 1, in Algorithm 2 we update the value in parallel for all states in every iteration. We show that the set W_i which is the set of all belief states for which $\mathrm{wtg}(\mathbf{b}) \neq \infty$ after i-th iteration, consists of all belief

states $\mathbf{b}$ that can reach an accepting belief state in at most i steps and with the worst-case cumulative weight $\mathrm{wtg}(\mathbf{b})$. The algorithm terminates after k, but at most $|\mathbf{B}| - 1$, iterations. If the resulting set W_i contains the initial belief state, the strategy consisting of the chosen actions for each belief state in W_i is returned.

Algorithm 2 Constructing a strategy for the weighted belief product and the given bound that maps to a solution of Problem 2.

Require: $\mathcal{B} = (\mathbf{B}, \mathbf{A}, T_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, F_{\mathcal{B}}, w)$, bound $k \geqslant 1$
Ensure: strategy C for the belief product $\mathcal{B}$

$W_0 := F_{\mathcal{B}}$
$\forall\, \mathbf{b} \in W_0 : \mathrm{wtg}(\mathbf{b}) := 0$
$\forall\, \mathbf{b} \in (\mathbf{B} \setminus W_0) : \mathrm{wtg}(\mathbf{b}) := \infty$
$i := 1$
while $i \leqslant k$ **do**
  for every $\mathbf{b} \in \mathbf{B}$ **do**
    $\mathbf{a}^{\mathbf{b}}_{\min} := \bot \qquad \Delta^{\mathbf{b}}_{\min} := \mathrm{wtg}(\mathbf{b})$
    for every $\mathbf{a} \in \mathbf{A}$ such that $\varnothing \neq T_{\mathcal{B}}(\mathbf{b}, \mathbf{a}) \subseteq W_{i-1}$ **do**
      $\Delta := \max_{\mathbf{b}' \in T_{\mathcal{B}}(\mathbf{b}, \mathbf{a})} \{w(\mathbf{b}, \mathbf{a}) + \mathrm{wtg}(\mathbf{b}')\}$
      if $\Delta < \Delta^{\mathbf{b}}_{\min}$ **then**
        $\mathbf{a}^{\mathbf{b}}_{\min} = \mathbf{a} \qquad \Delta^{\mathbf{b}}_{\min} := \Delta$
      end if
    end for
  end for
  $W_i := W_{i-1}$
  for every state $\mathbf{b} \in \mathbf{B}$ **do**
    $C(\mathbf{b}) := \mathbf{a}^{\mathbf{b}}_{\min}$
    $\mathrm{wtg}(\mathbf{b}) := \Delta^{\mathbf{b}}_{\min}$
    if $\mathrm{wtg}(\mathbf{b}) < \infty$ **then**
      $W_i := W_i \cup \{\mathbf{b}\}$
    end if
  end for
  $i := i + 1$
end while
if $\mathbf{b}_{\text{init}} \in W_i$ **then**
  return C
else
  return no suitable strategy exists for given bound
end if

Proposition 3 (Correctness). *Algorithm 2 results in a strategy C for the weighted belief product such that every run under C that starts in $\mathbf{b}_{\text{init}}$ visits an accepting belief state in at most k steps, if such a strategy exists.*

Proof. The property can be proved by induction on the iteration counter i and proving that starting from any state $\mathbf{b}_i \in W_i$, strategy C guarantees a visit to an accepting belief state in at most i steps. □

Proposition 4 (Optimality)**.** *Let C be the strategy resulting from Algorithm 2. Then among all strategies that guarantee a visit to an accepting belief state in at most k steps, C minimizes the value in Equation 7.1.*

Proof. Let C_i and wtg_i denote the strategy and the weight-to-go computed by Algorithm 2 before start of the $(i+1)$-th iteration. We show by induction that after every iteration $i \geqslant 1$, it holds that $V_{\mathcal{B}}(C_i, \mathbf{b}_i) = \text{wtg}_i(\mathbf{b}_i)$ for all states $\mathbf{b}_i \in W_i$, and that strategy C_i minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_i)$ among all strategies that guarantee visit to an accepting state in at most i steps.

Assume that for all $j < i$ it holds $V_{\mathcal{B}}(C_j, \mathbf{b}_j) = \text{wtg}_j(\mathbf{b}_j)$ for all states $\mathbf{b}_j \in W_j$, and that strategy C_j minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_j)$ among all strategies that guarantee visit to an accepting state in at most j steps. Trivially, C_0 minimizes the value for all accepting belief states $\mathbf{b} \in F_{\mathcal{B}}$ as $V_{\mathcal{B}}(C_0, \mathbf{b}) = 0 = \text{wtg}_0(\mathbf{b})$. We show that C_i then minimizes the value $V_{\mathcal{B}}(\cdot, \mathbf{b}_i)$ over all strategies that guarantee visit to an accepting state in at most i steps and $V_{\mathcal{B}}(C_i, \mathbf{b}_i) = \text{wtg}_i(\mathbf{b}_i)$.

Assume by contradiction that there exists a strategy C' for $\mathcal{B}$ such that $V_{\mathcal{B}}(C', \mathbf{b}_i) < V_{\mathcal{B}}(C_i, \mathbf{b}_i)$ and guarantee visit to an accepting state for all $\mathbf{b}_i \in W_i$ in at most i steps. Recall that W_{i-1} contains exactly those states that guarantee visit to an accepting state in at most $i-1$ steps (see proof of Proposition 3). Let $\mathbf{b}_i$ be an arbitrary belief state from W_i. Since C' is winning in at most i steps, it holds $T_{\mathcal{B}}(\mathbf{b}_i, C'(\mathbf{b}_i)) \subseteq W_{i-1}$. From the induction hypothesis and the fact the action $C_i(\mathbf{b}_i)$ minimizes the value in line 7 it follows that $V_{\mathcal{B}}(C', \mathbf{b}_i) \geqslant \text{wtg}_i(\mathbf{b}_i) = V_{\mathcal{B}}(C_i, \mathbf{b}_i)$ implying strategy C_i is optimal strategy among all strategies that guarantee visit to an accepting state in at most i steps. □

7.2.2 Strategy for NTS

We construct a strategy for the NTS from a strategy for the belief product in the same way as we did in the solution of the Problem 1.

Theorem 2. *Let $C_{\mathcal{B}}$ be the strategy for the weighted belief product $\mathcal{B}$ resulting from Algorithm 2. Then the strategy C for the NTS with observation modes constructed according to Equation 7.3 is a solution to Problem 2.*

Proof. The correctness with respect to the task φ follows directly from Proposition 3 and the optimality of C follows from Proposition 4 and the fact that

$$V(C, s_{\text{init}}, m_{\text{init}}), \varphi) = V_{\mathcal{B}}(C_{\mathcal{B}}, \mathbf{b}_{\text{init}}).$$

□

7.2.3 Complexity

The size of a minimal DFA for φ, the product and the belief product are discussed in Section 7.1.4. Similarly as for Algorithm 1, with a proper choice of a data structure storing the belief product $\mathcal{B}$, Algorithm 2 runs in time $\mathcal{O}(k \cdot |\mathbf{B}| \cdot |\mathbf{A}| \cdot \mathrm{dn})$, where dn is the degree of non-determinism of $\mathcal{N}$, *i.e.*, the maximum number of possible successors given a state and an action. Here, the value $|\mathbf{B}| \cdot |\mathbf{A}| \cdot \mathrm{dn}$ serves as the upper bound on the number of all edges in the belief product. Note that Algorithm 2 can be terminated prematurely if the current iteration i of the while loop in line 4 did not imply any change in the function wtg or if $i \geqslant |\mathbf{B}| - 1$ as every strategy for the belief product that guarantees a visit to an accepting belief state must do so in at most $|\mathbf{B}| - 1$ steps due to non-determinism.

Remark 5. *Note that Algorithm 2 can be used not only to solve the bounded Problem 2, but also the general Problem 1 by considering $k = |\mathbf{B}| - 1$. However, this solution to Problem 1 has higher computational complexity in practice than the one presented in Section 7.1 using Algorithm 1.*

7.3 Optimal Mission Control

In this section we describe the algorithm to solve Problem 3 in detail. Similarly as in previous cases, we first construct the product of the NTS and a DFA corresponding to the mission. We show how to construct a DFA for the mission so that it accepts exactly those words which satisfy the mission. Moreover we add to DFA a labeling function in order to keep track of the satisfied tasks.

Also the weighted belief product is constructed as in previous cases, but we slightly modify the construction for the technical reasons. The algorithm we presented in Section 7.1 is then run on the modified belief product.

7.3.1 Constructing a Mission DFA

We show that given a mission ϕ, the DFA, which accepts exactly those words which satisfy the mission, can be constructed. We perform some steps during the construction, which are not necessary for proving regularity of mission language, however, they are needed for solution of Problem 3. While constructing DFA for mission we attach every state a label – a real value representing collected reward.

Before the construction itself, we *unfold* a mission (see Definition 17), so it is in a form of sum of concatenation of tasks. DFA is then constructed from the unfolded version.

The proof proceeds as follows, first we show how to unfold a mission (Definition 18) and we prove that the result is equal to the original one with respect

to the satisfiability (stated as Lemma 1). Last, we translate the unfolded mission to a labeled DFA (Definition 19). We take an advantage of the fact that for every task φ the language $\mathsf{mgfp}(\varphi)$ is regular (Lemma 2).

Definition 16 (Labeled DFA)**.** *A labeled DFA is a tuple* $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F, L_{\mathcal{A}})$*, where* $(Q, 2^{AP}, \delta, q_0, F, L_{\mathcal{A}})$ *is a DFA according to Definition 7 and* $L_{\mathcal{A}} : Q \to \mathbb{R}$ *is a labeling function.*

We adopt for labeled DFA the all notation used for DFA.

Definition 17 (Unfolded mission)**.** *Let* ϕ *be a mission over* AP*, we say* ϕ *is unfolded if it is in the form of sum of concatenation of tasks,* i.e., $\phi = \sum_{i=1}^{k} \phi_i$*, where* $\phi_i = \prod_{j=1}^{n_i} \varphi_j$*, where* $\varphi_j \in \mathrm{Tasks}(AP)$*.*

Definition 18 (Transformation to unfolded mission)**.** *Let* ϕ *be a mission over* AP*, the unfolded mission* ϕ' *for* ϕ *is defined according to the structure of* ϕ *as follows:*

- $\phi \in \mathrm{Tasks}(AP)$*, then* $\phi' = \phi$*,*
- $\phi = \phi_1 + \phi_2$*, then* $\phi' = \phi'_1 + \phi'_2$*, where* ϕ'_1, ϕ'_2 *are unfolded missions of* ϕ_1 *and* ϕ_2 *respectively,*
- $\phi = \phi_1 \cdot \phi_2$*, let* $\phi'_1 = \sum_{i=1}^{k} \phi'_{1i}$ *be an unfolded mission of* ϕ_1*,* $\phi'_2 = \sum_{j=1}^{l} \phi'_{2j}$ *be an unfolded mission of* ϕ_2*, then*

$$\phi' = \sum_{i=1}^{k} \sum_{j=1}^{l} \phi'_{1i} \cdot \phi'_{2j}.$$

From now on we refer to ϕ' as to the unfolded mission for a mission ϕ.

Lemma 1. *Let* ϕ *be a mission over* AP*, then for every run* $\rho \in (2^{AP})^{\omega}$ *holds*

$$\rho \models \phi \iff \rho \models \phi', \tag{7.4}$$

where ϕ' *is the unfolded mission for* ϕ *according to Definition 18.*

Proof. We prove by induction with respect to the structure of ϕ stronger statement claiming that for every ϕ and ϕ' holds $\mathrm{seq}(\phi) = \mathrm{seq}(\phi')$ which implies validity of the Equation 7.4.

Assume that for every ϕ with syntactical depth up to some k hold $\mathrm{seq}(\phi) = \mathrm{seq}(\phi')$. We show that then also for a mission ϕ with syntactical depth up to $k+1$ holds $\mathrm{seq}(\phi) = \mathrm{seq}(\phi')$. A mission ϕ can only have one of the following forms:

- ϕ is a task, then equality Equation 7.4 trivially holds,

- $\phi = \phi_1 + \phi_2$, then according to Definition 18 is $\phi' = \phi'_1 + \phi'_2$. Recall $\text{seq}(\phi) = \text{seq}(\phi_1) \cup \text{seq}(\phi_2)$. When taking into the account also the induction hypothesis then it holds $\text{seq}(\phi) = \text{seq}(\phi_1) \cup \text{seq}(\phi_2) = \text{seq}(\phi'_1) \cup \text{seq}(\phi'_2) = \text{seq}(\phi')$.

- $\phi = \phi_1 \cdot \phi_2$, let $\phi'_1 = \sum_{i=1}^{k} \phi'_{1i}$ and $\phi'_2 = \sum_{j=1}^{l} \phi'_{2j}$, then $\phi' = \sum_{i=1}^{k} \sum_{j=1}^{l} \phi'_{1i} \cdot \phi'_{2j}$ according to Definition 18. Recall that $\text{seq}(\phi_1 \cdot \phi_2)$ is defined to be $\text{seq}(\phi_1) \cdot \text{seq}(\phi_2)$. Taking into the account the induction hypothesis we get $\text{seq}(\phi_1) = \text{seq}(\phi'_1) = \bigcup_{i=1}^{k} \text{seq}(\phi'_{1i})$, similarly $\text{seq}(\phi_2) = \text{seq}(\phi'_2) = \bigcup_{j=1}^{l} \text{seq}(\phi'_{2j})$. We have $\text{seq}(\phi') = \bigcup_{i=1}^{k} \bigcup_{j=1}^{l} \text{seq}(\phi'_{1i}) \cdot \text{seq}(\phi'_{2j})$ which is thanks to the concatenation's right distributivity over union equal to the $\bigcup_{i=1}^{k} \text{seq}(\phi'_{1i}) \cdot \bigcup_{j=1}^{l} \text{seq}(\phi'_{2j})$ which is from the induction hypothesis and Definition 18 equal to $\text{seq}(\phi_1) \cdot \text{seq}(\phi_2) = \text{seq}(\phi)$.

□

Lemma 2. *For every task φ is a language $\mathsf{mgfp}(\varphi)$ regular.*

Proof. Recall that for every task φ there is a DFA $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, F)$ which accepts all its good finite prefices [KYV01]. Without lost of generality suppose it is minimal. Furthermore, if p_φ is a good finite prefix of φ, then every its infinite extension satisfies φ and thus every finite word with prefix p_φ is also a good finite prefix of φ. It concludes that every accepting state q_f is such that $\delta(q_f, a) = q_f$ for every $a \in \Sigma$ and therefore there is exactly one accepting state (since $\mathcal{A}_\varphi$ is minimal).

Let DFA $\mathcal{A}_{\mathsf{mgfp}(\varphi)} = (Q, \Sigma, \delta', q_0, F)$ be a modification of $\mathcal{A}_\varphi$, where

$$\delta'(q, a) = \begin{cases} \bot & \text{if } q \in F \\ \delta(q, a) & \text{otherwise} \end{cases}$$

Then $L(\mathcal{A}_{\mathsf{mgfp}(\varphi)}) = \mathsf{mgfp}(\varphi)$ and hence $\mathsf{mgfp}(\varphi)$ is a regular language. The language equality follows from the fact, that we "cut" self-loop above the accepting state, *i.e.*, we "throw" from $\mathsf{gfp}(\varphi)$ every word which has prefix in $\mathsf{gfp}(\varphi)$ differs from itself. □

Definition 19 (Mission DFA). *Let AP be a set of atomic propositions, ϕ be an unfolded mission over AP,* i.e., *$\phi = \sum_{i=1}^{k} \phi_i$, where $\phi_i = \prod_{j=1}^{n_i} \varphi_j$, where $\varphi_j \in \text{Tasks}(AP)$. Let $r : \text{Tasks}(AP) \to \mathbb{R}$ be a reward function.*

- *For every mission $\phi_i = \prod_{j=1}^{n_i} \varphi_j$ which is concatenation of tasks we construct labeled DFA $\mathcal{A}_{\phi_i}$ as follows. Let $\mathcal{A}_{\mathsf{mgfp}(\varphi_j)} = (Q_j, 2^{AP}, \delta_j, q_{0_j}, F_j)$ be the minimal DFA for the language $\mathsf{mgfp}(\varphi_j)$ for all $j < n_i$ and let $\mathcal{A}_{\mathsf{gfp}(\varphi_{n_i})} = (Q_{n_i}, 2^{AP}, \delta_{n_i}, q_{0_{n_i}}, F_{n_i})$ be the minimal DFA for $\mathsf{gfp}(\varphi_{n_i})$. Without loss of generality suppose that set of states are disjoint for every two automata. We define $\mathcal{A}_{\phi_i} = (\bigcup_{j \leqslant n_i} Q_j \setminus \bigcup_{1<j\leqslant n_i} \{q_{0_j}\}, 2^{AP}, \delta, q_{0_1}, F_{n_i}, L)$, where the transition function δ is defined for all $a \in 2^{AP}$ as*

  - $\delta(q_{0_1}, a) = \delta_1(q_{0_1}, a)$,
  - $\delta(q, a) = \delta_j(q, a)$, *where* $q \in Q_j \setminus F_j \setminus \{q_{0_j}\}$ *for every* $j \leqslant n_i$,
  - $\delta(q, a) = \delta_{j+1}(q_{0_{j+1}}, a)$, *where* $q \in F_j$ *for every* $j < n_i$,
  - $\delta(q_{n_i}, a) = \delta_{n_i}(q_{n_i}, a)$ *for all* $q_{n_i} \in F_{n_i}$.

  Labelling function L is defined as

$$L(q) = \begin{cases} \sum\limits_{j \leqslant n_i} r(\varphi_j) & \text{if } q \in F \\ 0 & \text{otherwise} \end{cases}$$

- *Let $\mathcal{A}_{\phi_i} = (Q_i, 2^{AP}, \delta_i, q_{0_i}, F_i, L_i)$ be an automata for concatenation a mission ϕ_i, automata for ϕ is given as $\mathcal{A} = (\bigtimes_{i \leqslant k} Q_i, 2^{AP}, \delta, (q_{0_1}, \ldots, q_{0_k}), \bigtimes_{i \leqslant k} F_i, L)$, where δ is defined as for standard parallel composition and labelling function is*

$$L(q_1, \ldots, q_n) = \max\{L(q_1), \ldots, L(q_n)\}.$$

Proposition 5. *Let ϕ be an unfolded mission and $\mathcal{A}_\phi$ a labeled DFA for ϕ. Then for every infinite word w holds*

$$w \models \phi \iff w \in L(\mathcal{A}_\phi) \tag{7.5}$$

Proof. Validity follows immediately from the soundness of standard technique for parallel composition and from the fact we concatenate automata for the minimal good finite prefixes which have the unique final state with no outgoing transitions(see proof of Lemma 2). □

7.3.2 Mission Product

Product of an NTS and a labeled DFA is defined similarly as product of an NTS and a DFA presented in Section 7.1.1. We add labeling function to conserve DFA labels.

Definition 20 (Labeled Product). *Let $\mathcal{N} = (S, A, T, s_{\text{init}}, AP, L)$ be a NTS and $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F, L_{\mathcal{A}})$ be a labeled DFA. The labeled synchronous product is a tuple $(S \times Q, A, T_{\mathcal{P}}, (s_{\text{init}}, q_0), AP, F_{\mathcal{P}}, L_{\mathcal{P}})$ such that*

1. *$(S \times Q, A, T_{\mathcal{P}}, (s_{\text{init}}, q_0), AP, F_{\mathcal{P}})$ is a synchronous product of $\mathcal{N}$ and $(Q, 2^{AP}, \delta, q_0, F)$,*
2. *$L_{\mathcal{P}} : S \times Q \to \mathbb{R}$ is a labeling function defined as $L_{\mathcal{P}}(s, q) = L_{\mathcal{A}}(q)$.*

7.3.3 Mission Weighted Belief Product

The weighted belief product is modificated in order to capture rewards collected during the run. The new final state is added with transitions leading into it from the former final states. New added transitions are weighted with the negation of the worst case cumulated reward, *i.e.*, the lowest value of the label of product states.

Definition 21 (Mission Weighted Belief Product). *Given a labeled product $\mathcal{P} = (S \times Q, A, T_{\mathcal{P}}, (s_{\text{init}}, q_0), AP, F_{\mathcal{P}}, L_{\mathcal{P}})$ built over the NTS with observation modes $(\mathcal{N}, O, M)$ with the initial observation mode m_{init}. Let $\mathcal{B} = (\mathbf{B}, \mathbf{A}, T_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, F_{\mathcal{B}}, w)$ be the weighted belief product over $\mathcal{P}$ according to Definition14.*

The mission belief product $\mathcal{B}'$ over $\mathcal{P}$ is given as follows:

$$\mathcal{B}' = (\mathbf{B} \cup \{b_f\}, \mathbf{A} \cup \{a_f\}, T'_{\mathcal{B}}, \mathbf{b}_{\text{init}}, O, \{b_f\}, w'),$$

where

- *b_f is the newly added accepting state s.t. $b_f \notin \mathbf{B}$,*
- *$a_f \notin \mathbf{A}$ is the newly added action leading from the former accepting states to b_f,*
- *$T'_{\mathcal{B}} : \mathbf{B} \times (\mathbf{A} \cup \{a_f\}) \to 2^{\mathbf{B}}$ is the transition function defined as*

$$T'_{\mathcal{B}}(\mathbf{b}, \mathbf{a}) = \begin{cases} T_{\mathcal{B}}(\mathbf{b}, \mathbf{a}) & \mathbf{b} \in \mathbf{B}, \mathbf{a} \in \mathbf{A} \\ \{b_f\} & \mathbf{b} \in F_{\mathcal{B}}, \mathbf{a} = a_f \end{cases}$$

- *$w' : \mathbf{B} \times (\mathbf{A} \cup \{a_f\}) \to \mathbb{R}$ is the weight function defined as*

$$w'(\mathbf{b}, \mathbf{a}) = \begin{cases} w(\mathbf{b}, \mathbf{a}) & \mathbf{b} \in \mathbf{B}, \mathbf{a} \in \mathbf{A} \\ -\min\limits_{s \in \mathbf{b}} L_{\mathcal{P}}(s) & \mathbf{b} \in F_{\mathcal{B}}, \mathbf{a} = a_f \end{cases}$$

7.3.4 Strategy

In order to solve Problem 3 we first apply Algorithm 1 to construct a strategy for a mission weighted belief product and then we map a resulting belief strategy back to an original NTS as Equation 7.3 describes.

Proposition 6 (Correctness)**.** *Algorithm 1 results in a strategy C for the mission weighted belief product such that every run under C that starts in $\mathbf{b}_{\text{init}}$ visits an accepting belief state, if such a strategy exists.*

Proof. The soundness of Proposition 6 follows immediately from soundness of Proposition 1. □

Proposition 7 (Optimality)**.** *Let C be the strategy resulting from Algorithm 1 for a mission weighted belief product. Then among all strategies that guarantee a visit to an accepting belief state, C minimizes the value in Equation 7.1.*

Proof. The proof of Proposition 7 follows the proof of Proposition 2. Since only negative edges are those leading to a new added state, and it has no outgoing transition, all arguments used in the proof Proposition 2 retains valid. □

Theorem 3. *Let $C_{\mathcal{B}}$ be the strategy for the mission weighted belief product $\mathcal{B}$ resulting from Algorithm 1. Then the strategy C for the NTS with observation modes constructed according to Equation 7.3 is a solution to Problem 3.*

Proof. The correctness with respect to the mission ϕ follows directly from Proposition 6. The optimality of C follows from Proposition 7 and the fact that

$$V_r(C, s_{\text{init}}, m_{\text{init}}), \phi) = V_{\mathcal{B}}(C_{\mathcal{B}}, \mathbf{b}_{\text{init}}).$$

□

7.3.5 Complexity

The complexity for solution of Problem 3 can be derived from complexity of Problem 1 since we use in both cases Algorithm 1. However, note that by unfolding a mission we create a formula which can have an exponential size with respect to the original one.

7.4 Optimal General Mission Control

It can be shown that even decide whether Problem 4 has solution is undecidable. The proof is given by reduction from the halting problem for two-counter (Minsky) machines. The proof uses ideas similar to those presented in **??**. The following result has been done by Tomáš Zábojník and is included in this work just to provide a complete picture.

CHAPTER 8

Case Study

8.1 Implementation

We implemented the algorithms from Section 7.1, Section 7.2 and Section 7.3 in C++11. In this section, we demonstrate their use on a case study motivated by examples in [CCGK15b,SCL+15,THK+13]. All executions were performed on Windows 7 with 2.4 GHz Intel Core i5 450M processor and 4 GB DDR3 memory.

Implementation is accessible on GitHub repository `https://github.com/tesarova/optimal-sensor-util` where the description of the usage and some examples presented in this work are also provided.

8.2 Task

Consider a mobile robot moving in an environment partitioned into a grid of 5×5 equally sized regions. The grid contains a starting, a target and possibly multiple dangerous regions, where the robot is detected and captured. The robot knows the locations of the starting and the target regions but it does not know the exact locations of dangerous regions. Nevertheless, the robot knows that the grid takes one of the three forms depicted in Fig. 8.1. The robot moves deterministically in (up to) four directions corresponding to the movement in the four compass directions. To learn the presence of dangerous regions in it's immediate surroundings, the robot can deploy one of the two sensors in Fig. 8.2. The first sensor partitions the neighboring area into quadrants and reports the set of all quadrants that contain at least one dangerous region. The second sensor reports the exact regions in robot's immediate surroundings that are dangerous. In every step, the robot can decide which sensor to activate, if any. The costs of deployment of the two sensors is 1 and 2, respectively. The cost can be interpreted as the amount of resources needed for the use of each sensor. Alternatively, it may model the amount of information received by the enemy in dangerous regions. The goal of the robot is to reach the target region from the starting region without being detected, while minimizing the cost.

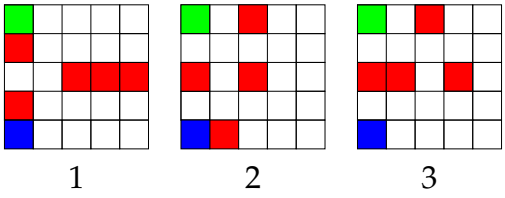


Figure 8.1: The environment of a mobile robot partitioned into a grid of 5×5 equally sized regions. The three grids correspond to three possible placements of dangerous regions, shown in red. The locations of the starting region, in green, and the target region, in blue, are known and hence their placement is the same in all three grids.

The NTS with observation modes that models the above system has 76 states $S = \{s_{\text{init}}, s_{ijk} \mid 1 \leqslant i \leqslant 3, 1 \leqslant j, k \leqslant 5\}$ and 5 actions $A = \{a, \mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}\}$. States s_{ijk} correspond to the regions in the three grids, where $1 \leqslant i \leqslant 3$ is the grid identifier and $1 \leqslant j, k \leqslant 5$ determine the row and column coordinate, respectively. For example, s_{111} is the top left corner of the first grid. The initial state s_{init} has only one transition $T(s_{\text{init}}, a) = \{s_{111}, s_{211}, s_{311}\}$ that corresponds to the enemy choosing one of the three grids in Fig. 8.1. The transitions of all s_{ijk} are deterministic and correspond to moving in compass directions $\mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}$. The set $AP = \{\texttt{dang}, \texttt{target}\}$ and the labeling function is such that $L(s_{\text{init}}) = \varnothing$ and $L(s_{ijk})$ indicates the target and dangerous regions as in Fig. 8.1. The set of observations is $O = \{\texttt{N}, \texttt{S}, \texttt{W}, \texttt{E}, \texttt{NW}, \texttt{NE}, \texttt{SW}, \texttt{SE}, \text{det}\}$. The NTS has 3 observation modes corresponding to not activating any sensor, activating the first sensor and activating the second sensor, respectively. The respective observation functions $\gamma_1, \gamma_2, \gamma_3$ are defined in Fig. 8.2 and $g_1 = 0, g_2 = 1, g_3 = 2$. Note that in every step of an execution of the system, we know the robot's position in the grid precisely, only the identifier of the grid is unknown.

The scLTL formula specifying the robot's task is $(\neg\texttt{dang})\,\mathbf{U}\,\texttt{target}$ and the corresponding minimal DFA $\mathcal{A}$ has 3 states. The product $\mathcal{P}$ of $\mathcal{N}$ and $\mathcal{A}$ has 208 states and 667 (possibly non-deterministic) transitions, and was constructed in less than 0.1 seconds. The weighted belief product $\mathcal{B}$ has 375 states and 2634 transitions, and was constructed in 1.5 seconds.

The strategy for the robot is as follows. In the starting region, use the first sensor. If the reported observations are SE and SW, then the robot is in grid 1 from Fig. 8.1. If the set of reported observations is empty, the robot is moving either in grid 2 or grid 3. In the former case, do not deploy any sensors anymore and move in directions $\mathsf{E}, \mathsf{S}, \mathsf{S}, \mathsf{S}, \mathsf{S}, \mathsf{W}$ to reach the target region. In the latter case, do not use any sensor anymore and move in directions $\mathsf{S}, \mathsf{E}, \mathsf{E}, \mathsf{E}, \mathsf{E}, \mathsf{S}, \mathsf{S}, \mathsf{S}, \mathsf{W}, \mathsf{W}, \mathsf{N}, \mathsf{W}, \mathsf{W}, \mathsf{S}$. The worst-case cost of the strategy is 1 and the maximum number of transitions

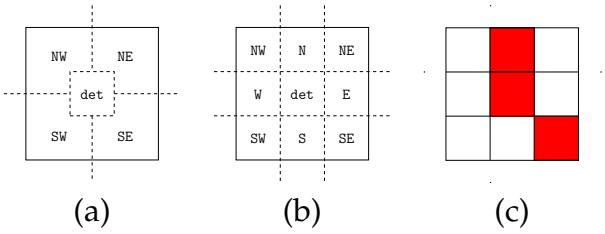


Figure 8.2: Two sensors that provide information about the presence of dangerous regions in robot's immediate surroundings. We also show the names of the corresponding observations learned by the robot. For the first sensor in (a), the surrounding area is divided into quadrants and the sensor reports all quadrants containing a dangerous region. For the second sensor in (b), the exact set of dangerous regions is reported. For example, let (c) show the immediate surroundings of the robot with it's current position in the middle and dangerous regions in red. The first sensor reports the set of observations $\{\texttt{NW}, \texttt{NE}, \texttt{SE}, \text{det}\}$ and the second sensor reports $\{\texttt{N}, \texttt{SE}, \text{det}\}$.

performed by the robot to reach the target region is 14, *i.e.*, in the NTS $\mathcal{N}$ it is 15 including the first step for choosing the grid.

Next, we used Algorithm 2 to solve the bounded version of the problem for bounds $k < 15$, where the task must be satisfied faster than using the strategy above. For all choices of k below, the algorithm terminated in less than 3 seconds. For $k \leqslant 8$, there does not exist a suitable strategy. For $k = 13$ and $k = 14$, the optimal strategy has the same structure as the one resulting from Algorithm 1 with the following exception. If the robot learns that it moves either in grid 2 or 3, the sequence of directions is $\mathsf{S}, \mathsf{E}, \mathsf{E}, \mathsf{E}, \mathsf{E}, \mathsf{S}, \mathsf{S}, \mathsf{W}, \mathsf{W}, \mathsf{W}, \mathsf{W}, \mathsf{S}$. The maximum number of steps needed to reach the target region is 13 and the worst-case cost of the strategy is 1.

Finally, for $9 \leqslant k \leqslant 12$ there exists a solution and the corresponding optimal strategy for the robot is as follows. From the starting region, move in directions E and then S without deploying any sensor. Then move in direction E and activate the second sensor. If the reported observations are $\texttt{S}$ and $\texttt{SE}$ then the robot is moving in grid 1. In such a case, do not use any sensors anymore and move in directions $\mathsf{W}, \mathsf{S}, \mathsf{S}, \mathsf{S}, \mathsf{W}$ to reach the target region. Similarly, if the observations are $\texttt{S}$ an $\texttt{N}$ then the robot is in grid 2, do not use any sensors anymore and move in directions $\mathsf{W}, \mathsf{S}, \mathsf{S}, \mathsf{W}, \mathsf{S}$. Finally, if the observations are $\texttt{SW}, \texttt{SE}$ and $\texttt{N}$ then the robot is in grid 3, do not use any sensors and move in directions $\mathsf{S}, \mathsf{S}, \mathsf{S}, \mathsf{W}, \mathsf{W}$. While the maximum number of steps needed to reach the target region is 9, the worst-case cost of the strategy is 2.

8.3 Mission

We consider a military rescue mission inspired by the case study presented in [THK+13]. Friendly units F_1, F_2 were captured on the enemy's territory. Unit F_1 is guarded by the target T. The aim of the robot is to release friendly units and to transport them safely to the friendly base. In order to liberate friendly unit F_1, robot must first destroy the target T whereas the friendly unit F_2 can be liberated directly.

The robot does not have precise information about dangerous regions in enemy territory. Nevertheless, it knows that placement of objects is one of those described in Fig. 8.3. In order to orientate in the enemy's territory, the robot is equipped with sensors as described in Figure 8.2.

The territory is modeled as a grid 6x6 analogously as in Section 8.2. However, in this example we model a danger implicitly, *i.e.*, there are no outgoing transitions from the region with a danger. We also encode in the transition system the requirement that in order to release friendly unit F_1, first a region with the target must be visit. The system has 217 states, $S = \{s_{\text{init}}, (s_{ijk}, T) \mid 1 \leqslant i \leqslant 3, 1 \leqslant j, k \leqslant 5, T \in \{0, 1\}\}$ and 5 actions $A = \{a, \mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}\}$. States (s_{ijk}, T) correspond to the regions in the three grids, where $1 \leqslant i \leqslant 3$ is the grid identifier,$1 \leqslant j, k \leqslant 6$ determine the row and column coordinate, respectively and T is a bool variable indicating whether the target has been already destroyed. If the target is destroyed then the transitions to the region with friendly unit F_1 are enabled. Observation modes for the system are as described in Figure 8.2, *i.e.*, only surrounding danger can be detected.

There is an atomic proposition attached to every position in the grid representing the presented objects. Whenever there is the friendly unit F_1 (F_2) in a region, we append the proposition $\mathtt{F}_1$ ($\mathtt{F}_2$) to that region. We label the region where the friendly base is presented with the atomic proposition `base`.

It is desired to save both the friendly units from enemy's territory, however, the deployment of the sensors increase the chance of being detected. So we create a separate task for each unit representing how much it is valuable to save each unit, or alternatively, how much is it worth to risk the detection. Consider the task $\varphi_{12} = \mathbf{F}\,\mathtt{F}_1 \wedge \mathbf{F}\,\mathtt{F}_2$ with reward 3 and $\varphi_2 = \mathbf{F}\,\mathtt{F}_2$ with reward 1.

We compose a mission ϕ for the robot as follows:

$$\phi = (\varphi_{12} + \varphi_2) \cdot \mathtt{base}$$

One can see that if the robot wants to save F_1, it needs to deploy sensors at least two times in the worst case. First at the very beginning, since only "safe" action is E. After taking this action robot cannot move nowhere to not be exposed by danger being destroyed. So it has to deploy sensor at least to be able to distinguish the grid 3 from the grid 1 and the grid 2. No matter which sensor it use, it

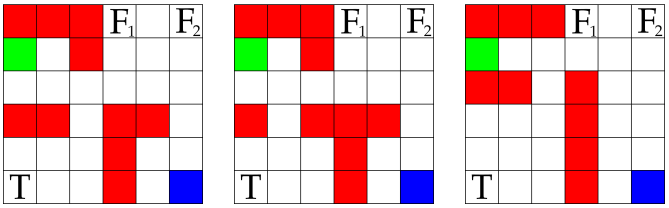


Figure 8.3: The environment of a mobile robot partitioned into a grid of 6×6 equally sized regions. The three grids correspond to three possible placements of dangerous regions, shown in red. The locations of the starting region, in green, friendly units F_1, F_2 (marked by corresponding letters), the target guarding F_1, marked with letter T, and the friendly base region, in blue, are known and hence their placement is the same in all three grids.

cannot distinguish grid 1 from grid 2 and so it has to use the sensor second time to guarantee to safely reach and destroy the target T. Hence, we can conclude that an optimal strategy can turn on the cheaper sensor in the first step. One can see that it is sufficient to use again, for the second time, the cheaper sensor to distinguish the grid 1 from the grid 2 and so the optimal strategy for saving both friendly units has the mission cost $V_r(\cdot) = -1$ (we pay 2 for sensors and gain a reward 3).

Consider now the second case, *i.e.*, try to find the strategy which does not save the friendly unit F_1. Note, that no such a strategy can have lower mission cost then the one mentioned above, since for saving only friendly unit F_2 we are given reward 1 which covers the price for deploying sensors in the first step and so we cannot go below zero, *i.e.*, for every strategy which does not release F_1 is the mission cost at least $V_r(\cdot) = 0$.

CHAPTER 9

Conclusion

We consider non-deterministic transition systems with multiple observation modes with fixed non-negative costs. We present correct and optimal algorithms to solve two optimal temporal control problems. The first aims to construct a control and observation mode switching strategy that guarantees satisfaction of a finite-time temporal property given as a formula of scLTL (a task) and minimizes the worst-case cost accumulated until the point of satisfaction. We also consider the bounded version of the problem with a bound on the time of satisfaction. Both algorithms are demonstrated on a case study motivated by robotic application.

Next we show a more general version of previous problems, when temporal property is given as a regular expression excluding Kleene star over tasks (a mission). Each task, as a building block, can be reward and the aim is to minimize the worst-case cost similarly as in case of task minus collected reward while guarantee satisfaction of a mission.

We also discussed extension of formulated problems, when we allow to use also Kleene star as operator for building a mission and we aim to find strategy which guarantee to fulfill the mission with the cost not worse than some given bound. This problem turns out to be undecidable.

Bibliography

[BG11] Nathalie Bertrand and Blaise Genest. Minimal Disclosure in Partially Observable Markov Decision Processes . In *Proc. of FSTTCS*, volume 13, pages 411–422, 2011.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[BL90] J. Richard Buchi and Lawrence H. Landweber. *The Collected Works of J. Richard Büchi*, chapter Solving Sequential Conditions by Finite-State Strategies, pages 525–541. Springer New York, New York, NY, 1990.

[CCGK15a] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal Cost Almost-Sure Reachability in POMDPs. In *Proc. of AAAI*, pages 3496–3502, 2015.

[CCGK15b] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Qualitative analysis of POMDPs with temporal logic specifications for robotics applications. In *Proc. of ICRA*, pages 325–330, 2015.

[CCT+13] Luis I. Reyes Castro, Pratik Chaudhari, Jana Tumova, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Incremental sampling-based algorithm for minimum-violation motion planning. *CoRR*, abs/1305.1102, 2013.

[CDH13] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A survey of partial-observation stochastic parity games. *Formal Methods in System Design*, 43(2):268–284, 2013.

[CM11] Krishnendu Chatterjee and Rupak Majumdar. Minimum Attention Controller Synthesis for Omega-Regular Objectives. In *Proc. of FORMATS*, volume 6919 of *Lecture Notes in Computer Science*, pages 145–159, 2011.

[CMH08] Krishnendu Chatterjee, Rupak Majumdar, and ThomasA. Henzinger. Controller Synthesis with Budget Constraints. In *Proc. of HSCC*, volume 4981 of *Lecture Notes in Computer Science*, pages 72–86, 2008.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[FTR08] Z.G. Feng, K.L. Teo, and V. Rehbock. Hybrid method for a general optimal sensor scheduling problem in discrete time. *Automatica*, 44(5):1295 – 1303, 2008.

[JSB13] A. Jones, M. Schwager, and C. Belta. A receding horizon algorithm for informative path planning with temporal logic constraints. In *Proc. of ICRA*, pages 5019–5024, 2013.

[KLC98] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99 – 134, 1998.

[KYV01] Orna Kupferman and Moshe Y. Vardi. Model Checking of Safety Properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[MHC03] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1–2):5 – 34, 2003.

[OGM+15] Peter Ondruska, Corina Gurau, Letizia Marchegiani, Chi Hay Tong, and Ingmar Posner. Scheduled Perception for Energy-Efficient Path Following. In *Proc. of ICRA*, 2015.

[PGT03] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *Proc. of IJCAI*, pages 1025–1030, 2003.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Oct 1977.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[SCL+15] Maria Svorenova, Martin Chmelik, Kevin Leahy, Hasan Ferit Eniser, Krishnendu Chatterjee, Ivana Cerna, and Calin Belta. Temporal logic motion planning using POMDPs with parity objectives: case study paper. In *Proc. of HSCC*, pages 233–238, 2015.

[Sis94] A. Prasad Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–511, 1994.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[THK+13] Jana Tumova, Gavin C. Hall, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Least-violating control strategy synthesis with safety rules. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC '13, pages 1–10, New York, NY, USA, 2013. ACM.

[UWB14] A. Ulusoy, T. Wongpiromsarn, and C. Belta. Incremental Controller Synthesis in Probabilistic Environments with Temporal Logic Constraints. *Int. Journal of Robotics Research*, 33(8):1130–1144, 2014.

[VZA+10] M.P. Vitus, Wei Zhang, A. Abate, Jianghai Hu, and C.J. Tomlin. On efficient sensor scheduling for linear dynamical systems. In *Proc. of ACC*, pages 4833–4838, 2010.