
ECOLE
POLYTECHNIQUE
DE BRUXELLES

# Enabling research on complex tasks in swarm robotics

## Novel conceptual and practical tools

Arne Brutschy
Ph.D. Thesis

Promoteur de Thèse:
Prof. Marco Dorigo

Co-promoteur de Thèse:
Prof. Mauro Birattari

Thèse présentée en vue de l'obtention du titre de
Docteur en Sciences de l'Ingénieur

Année académique 2014–2015

This dissertation has been submitted in partial fulfillment of the requirements for an advanced degree at Université Libre de Bruxelles. The dissertation describes an original research carried out by the author. It has not been previously submitted to the Université Libre de Bruxelles or to any other university for the award of any degree. Nevertheless, some chapters of this dissertation are partially based on articles that, during his doctoral studies, the author, together with a number of co-workers, submitted for publication in the scientific literature. A detailed listing of these publications can be found in Chapter 1.

*für Christine*

# Abstract

Research in swarm robotics focuses mostly on how robots interact and cooperate to perform tasks, rather than on the details of task execution. As a consequence, researchers often consider abstract tasks in their experimental work. For example, foraging is often studied without physically handling objects: the retrieval of an object from a source to a destination is abstracted into a trip between the two locations—no object is physically transported. Despite being commonly used, so far task abstraction has only been implemented in an *ad hoc* fashion.

In this dissertation, I propose a collection of tools for flexible and reproducible task abstraction. At the core of this collection is a physical device that serves as an abstraction of a single-robot task to be performed by an e-puck robot. I call this device the TAM, an acronym for *task abstraction module*. A complex multi-robot task can be abstracted using a group of TAMs by first modeling the task as the set of its constituent single-robot subtasks and then representing each subtask with a TAM. I propose a novel approach to modeling complex tasks and a framework for controlling a group of TAMs such that the behavior of the group implements the model of the complex task.

The combination of the TAM, the modeling approach, and the control framework forms a collection of tools for conducting research in swarm robotics. These tools enable research on cooperative behaviors and complex tasks with simple, cost-effective robots such as the e-puck—research that would be difficult and costly to conduct using specialized robots or *ad hoc* solutions to task abstraction. I present proof-of-concept experiments and several studies that use the TAM for task abstraction in order to illustrate the variety of tasks that can be studied with the proposed tools.

# Acknowledgements

First and foremost, I would like to thank my supervisors Marco Dorigo and Mauro Birattari. I am very grateful for the opportunity to work at IRIDIA, become a scientist, and, in the process, grow as a person. Learning from you was invaluable.

I would also like to thank everyone else at IRIDIA. In particular, I would like to thank all my fellow students, past and present, far too numerous (and the stories too wild) to be listed here. Working with all of you was great, and it was your presence that made IRIDIA such a special place for me. Special thanks go to Giovanni Pini, Rehan O'Grady, and Jérémie Dubois-Lacoste.

An meine Familie: vielen Dank, dass Ihr mich immer unterstützt, beraten und ermutigt habt – ohne Euch wär's wohl nix geworden.

# Contents

Chapter 1

# Introduction

Robotic systems have been traditionally designed following engineering principles established during the industrial revolution: a robot is treated as a single monolithic machine that possesses all the capabilities necessary to address the tasks it faces. Traditional methods from the field of artificial intelligence, devised for controlling such robots, also follow a monolithic approach: a robot possesses a single "brain", for which the yardstick of intelligence is, of course, human intelligence.

Collective robotics is an extension of this traditional approach to groups of robots. Essentially, collective robotics faces similar problems as cooperating humans; it has therefore traditionally been approached with solutions similar to the ones used to coordinate groups of humans. Today, centralized control, planning, and external support-infrastructure are prevalent for controlling groups of robots. Unfortunately, systems designed using monolithic approaches to artificial intelligence and centralized approaches to coordination have several issues, mostly related to their lack of flexibility and robustness.

An alternative approach to the design of robot systems is *swarm robotics* (Dorigo et al., 2014). Swarm robotics is a relatively recent approach that employs large groups of robots, called swarms, to address the mission at hand. Contrary to traditional approaches, swarm robotics does not rely on centralized control or planning (Beni, 2005). Instead, the collective behavior of the robot swarm results from the local interactions between the individuals of the swarm and between these individuals and the surrounding environment (Dorigo et al., 2014). Consequently, the behavior of the group is not monolithic but emerges from the behavior of the many individuals of the swarm. The design

of robot swarms follows the principles of *swarm intelligence* (Beni and Wang, 1989), which promote the creation of systems that are *fault tolerant*, *scalable*, and *flexible* (Dorigo and Birattari, 2007).

Swarm robotics appears to be a viable approach to applications that benefit from the attributes listed above. Moreover, applications that benefit are those that require the execution of a large number of concurrent activities or develop in environments in which the establishment of the infrastructure required to centrally control a large number of robots is extremely difficult. Accordingly, it is considered advantageous to apply swarm robotics systems to tasks that are potentially hazardous, cover a large area, or develop in highly time-variant environments. Examples of this kind of tasks are search and rescue missions, surveillance, de-contamination and de-mining, as well as exploration of hazardous environments such as outer space or the deep sea.

Regardless to its many benefits, to date, there are no known real-world applications of swarm robotics, that is, applications outside of the highly abstracted environments used in research laboratories. This lack is mostly due to the following two issues.

The first issue is related to the design of such systems: current methods to design a swarm are based on bottom-up techniques driven by a trial-and-error approach (Brambilla et al., 2013). The drawback of these methods is that they rely heavily on the experience of the designer and therefore lack in repeatability. Proponents of *swarm engineering* contend that swarm robotics lacks an engineering methodology for robot swarms (Kazadi, 2000; Dorigo et al., 2014). Hence, such methodologies are the object of recent and ongoing research (e.g., Brambilla et al., 2014; Francesca et al., 2014b).

The second issue is related to the materials available for studying swarm robotics systems. Today's robotic platforms are unreliable, expensive, and rather limited in their capabilities (Cao et al., 1997). Conducting experiments considering large groups of robots is therefore very costly—both monetary and otherwise. Furthermore, the complexity of the task considered in a study strongly influences this cost as well: studying complex tasks consisting of many, interrelated subtasks increases costs as resources have to be spent on implementing details specific to the execution of each subtask.

In this dissertation, I conjecture that costs strongly influence the complexity of the problems studied in robotics: researchers consider problems that can be studied using experiments of reasonable cost, and

omit other problems that are too costly to study. This effect is especially pronounced in swarm robotics, where large swarms increase the cost of an experiment to the point that considering anything other than simple problems becomes quickly prohibitive (Carlson et al., 2004). As a result, most studies in swarm robotics consider problems of low complexity.

One solution to this issue is to provide robots that are more reliable and capable, while at the same time being cheaper. Unfortunately, trade-offs make it impossible to provide robots that satisfy all three requirements—for example, the Kilobot is cheap and reliable (Rubenstein et al., 2012), but does so at the expense of its capabilities.

An alternative solution is to provide researchers with the means to reduce the cost of studying complex problems using today's robots. In this dissertation, I pursue this solution: I propose conceptual and practical tools that reduce the cost of engineering-oriented studies that consider complex tasks. These novel and unique tools enable research on problems that concern tasks with various types of complex interrelationships; problems that were, to date, confined to simulation by the fact that they were too costly to be studied using real robots.

The motivation for creating novel tools for swarm robotics research stems from my first experiments in task allocation: At the beginning of my doctoral studies, I realized that the vast majority of works in the swarm robotics literature consider tasks of low complexity. In fact, most studies consider only simple tasks that do not exhibit interrelationships. Simple tasks without interrelationships can be executed independently of other tasks—examples include picking up an object or disabling an alarm.

I set out to fill this gap in the literature by conducting an experiment that considered several interrelated tasks (Brutschy et al., 2014c). In the experiment, robots had to harvest objects from a source and store them in a nest—see Figure 1.1 for a snapshot of the experiment that explains the tasks and their interrelationships. The focus of the study was on how robots interact and cooperate to address these tasks, rather than on the details of task execution. In particular, the focus was on a novel method for self-organized task allocation and not on the behaviors required to recognize, manipulate and transport objects. However, I quickly had to realize that developing and conducting experiments that included these details of task execution, which were inessential to the study, was extremely time-consuming. In fact, performing ex-

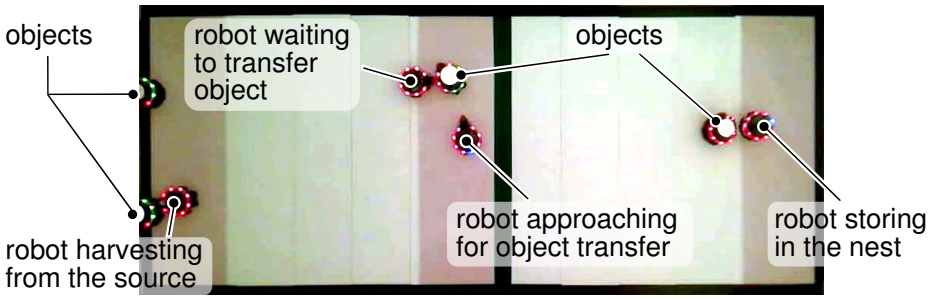



Figure 1.1: A snapshot of an experiment in which the swarm has to perform several interrelated tasks (Brutschy et al., 2014c). The overall task of the robots is to harvest objects from the source (left) and store them in the nest (right). The task is partitioned into two subtasks: *harvest* and *store*. Robots working on the *harvest* subtask can transfer objects to the robots working on the *store* subtask using the area located at the center. Each subtask consists itself of two subtasks: harvesting/transferring and receiving/storing an object, respectively. All interrelationships between these subtasks are of a sequential nature: subtasks have to be executed in a given sequence in order to complete the overall task once.

periments using real robots was so time-consuming that we could not conduct more than a single proof-of-concept experiment. As a result, the majority of the experiments had to be conducted in simulation.

From this experience, I realized that most research in swarm robotics faces similar issues: if the focus of a study is to develop coordination mechanisms that allow robots to tackle tasks with a certain kind of logical relationship, it might be desirable to isolate the logical relationship from the details of task execution and focus on it, rather than spending resources on inessential aspects of the implementation. Examples of interrelated tasks are tasks that have to be executed in a specific order, for instance, first harvesting, then transporting, and finally storing food items. I call task abstraction the process by which one focuses on the logical relationship between tasks and omits the details on their execution. Research in swarm robotics is likely to benefit from task abstraction, as it commonly focuses on how robots interact and cooperate to perform tasks, rather than on the details of their execution (Brambilla et al., 2013).

Task abstraction is not a novel concept in swarm robotics research; in fact, it has been used implicitly in numerous studies—for a compre-

hensive review of the swarm robotics literature, see Brambilla et al. (2013). However, up to now, task abstraction was either i) confined to simulation or ii) conducted using some sort of *ad hoc* solution. Simulation has the advantage of being inexpensive, but approaches developed solely in simulation may suffer from the so called "reality gap" (Jakobi et al., 1995; Francesca et al., 2014b). This is particularly relevant in complex systems, where small but unavoidable differences between simulation and reality could lead to widely diverging behaviors.

*Ad hoc* solutions are specific abstractions that are tightly connected to the nature of the experiment at hand. They cannot be easily and directly exploited in other experiments. For example, *ad hoc* solutions were used to abstract the act of physically transporting objects with a trip between two locations (e.g. Kernbach et al., 2012; Acerbi et al., 2007; Francesca et al., 2014b,a); to abstract manipulation tasks with tailor-made inanimate physical objects (e.g., Ijspeert et al., 2001; Tuci et al., 2006); and to abstract tasks with some dynamic property using electronic devices (e.g. Matarić et al., 2003). *Ad hoc* solutions are suitable only for simple tasks that can be tackled by a single robot without any relations to other robots or tasks. Indeed, tasks that require multiple robots are much harder to abstract due to the interrelationships between the constituent single-robot subtasks and the actions of the robots. Additionally, experiments that use *ad hoc* solutions are costly and difficult to replicate by other researchers—a fact that effectively limits the complexity of the tasks studied in the literature.

In this dissertation, I propose a collection of tools for task abstraction. At the core of this collection is a physical device that serves as an abstraction of single-robot tasks to be performed by an e-puck robot. I call this device the TAM, an acronym for *task abstraction module*. In abstract terms, I say that a robot performs a single-robot task if it is busy for a given amount of time at a specific location and at a specific moment in time. A TAM represents such an abstracted single-robot task in real-robot experiments.

Complex multi-robot tasks can be abstracted and represented as follows. First, a complex task is modeled as the set of its constituent single-robot subtasks and their interrelationships. Second, each single-robot subtask is represented by a single TAM and the behavior of the TAMs is coordinated such that it reflects the interrelationships identified by the model. I propose a novel approach to modeling complex tasks and a framework for controlling a group of TAMs such that the behavior of the group implements the model of the complex task.

The combination of the TAM, the modeling approach, and the control framework enable research on cooperative behaviors and complex tasks with simple, cost-effective robots such as the e-puck (Mondada et al., 2009)—research that would be prohibitively difficult and costly to conduct using specialized robots or *ad hoc* solutions to task abstraction. I present proof-of-concept experiments and several studies that use the TAM for task abstraction in order to illustrate the variety of tasks that can be studied with the proposed tools.

## Goals of this dissertation

The goal of this dissertation is to provide the conceptual and practical tools for modeling and representing complex task in swarm robotics studies. Furthermore, the dissertation aims to provide a framework for conducting experiments involving complex tasks and large swarms of robots.

## Scientific contributions

In this section, I present the contributions to the scientific literature that I made over the course of my doctoral studies. See Page 197 for a detailed list of all publications referenced below.

### Scientific contributions of this dissertation

In the following, I enumerate the scientific contributions of this dissertation and list the publications that resulted.

1. A definition of complex tasks, subtasks and task interrelationships. Based on these definitions, a novel approach to model complex multi-robot tasks (Brutschy et al., 2014a).

2. A review of the swarm robotics literature from the perspective of the tasks studied and abstractions used (Brutschy et al., 2014a).

3. A systematic approach to task abstraction, unique in the literature (Brutschy et al., 2014a). This approach is based on the TAM, a novel device that serves as an abstraction of single-robot tasks to be performed by an e-puck robot.

4. A verification of the concept of the TAM and the proposed approach to modeling complex tasks in a proof-of-concept experiment using a swarm of e-puck robots (Brutschy et al., 2014a).

5. A study of the costs and benefits of behavioral specialization in a swarm of robots (Brutschy et al., 2012c, 2011). The study uses the TAM for task abstraction.

6. A novel top-down design method based on prescriptive modeling and model checking (Brambilla et al., 2014). The study uses the TAM for task abstraction.

7. An approach to self-organized task allocation in environments that exhibit periodic properties (Castillo-Cagigal et al., 2014). The study uses the TAM for task abstraction.

8. An adaptive algorithm for strategy selection in a task partitioning scenario (Frison et al., 2010; Pini et al., 2013b) and a comparison of this algorithm to established multi-armed bandit algorithms (Pini et al., 2012, 2013b). The study uses the TAM for task abstraction.

9. A new method for self-organized allocation of a swarm of robots to a complex task (Brutschy et al., 2014c). This work provided the inspiration for the novel tools proposed in this dissertation.

## Other scientific contributions

In addition to the contributions made in this dissertation, I contributed to various other scientific studies on a diverse range of topics. In the following, I enumerate these contributions grouped by their topic and list the publications that resulted.

1. Self-organized task allocation:
   a) An approach to self-organized decision-making based on the k-unanimity rule (Scheidler et al., 2014). In the accompanying study, robots have to collectively decide on the shorter of two paths. A peer-reviewed video that illustrates the approach (Brutschy et al., 2012b).
   b) Several ant-inspired organic computing algorithms, inspired by the house-hunting strategies of the ant *Temnothorax*

*albipennis* (Scheidler et al., 2011; Brutschy et al., 2008). One of the studies won the "Best paper" award at ANTS 2012 (Brutschy et al., 2008).

2. Autonomous task partitioning:
   a) A method for autonomous task partitioning in a foraging scenario (Pini et al., 2013a, 2014). The method is based on cost estimations of the available choices: robots rely on these estimates to autonomously partition the foraging task.
   b) A method for reducing physical interference between robots that occurs at shared resources (Pini et al., 2009, 2011a). The method is based on task partitioning.
3. Automatic design of robot controllers:
   a) A novel approach to the automatic design of control software for robot swarms, called AutoMoDe (Francesca et al., 2014b). AutoMoDe is based on injecting bias in the design process by selecting, instantiating and combining preexisting parametric modules.
   b) An objective comparison of multiple design methods: two automatic methods—AutoMoDe and a method based on evolutionary robotics—are compared with swarms manually designed by human experts (Francesca et al., 2014a).
4. The Swarmanoid project:
   a) The *Swarmanoid* itself, a novel distributed robotic system made up of heterogeneous, dynamically connected, small autonomous robots (Dorigo et al., 2013). Additionally, a video demonstrating the Swarmanoid, which won the "Best video" award of the AAAI video competition (Dorigo et al., 2011).
   b) The ARGoS simulation framework, a novel simulator for heterogeneous swarms (Pinciroli et al., 2011, 2012). The focus of ARGoS is to provide a framework for simulating large-scale, heterogeneous swarm robotics systems while being accurate, efficient and flexible.

5. Spatially targeted communication:
   a) A novel communication protocol for spatially targeted communication in robot swarms (Mathews et al., 2014). The protocol relies on cameras and LEDs to provide spatially targeted communication links. It is distributed and scalable, while being independent of external tracking infrastructure and global information.

## Publication summary

**13 Journal papers** (3 as first author – ★, 3 under review – †)

- *Systems, Man, and Cybernetics, Part B: Cybernetics*, 2014 (†)
- *Swarm Intelligence*, 2014 (★, †)
- *Autonomous Robots*, 2014 (†)
- *Swarm Intelligence*, 2014
- *ACM Transactions on Autonomous and Adaptive Systems*, 2014
- *Autonomous Agents and Multi-Agent Systems*, 2014 (★)
- *Artificial Life*, 2014
- *Robotics and Autonomous Systems*, 2014 (★)
- *IEEE Robotics & Automation Magazine*, 2013
- *Adaptive Behavior*, 2013
- *Swarm Intelligence*, 2013
- *Swarm Intelligence*, 2012
- *Swarm Intelligence*, 2011

**2 Book chapters**

- *"Organic Computing – A Paradigm Shift for Complex Systems"*, 2011, Springer, Germany.
- *"Informatics in Control, Automation and Robotics"*, 2011, Springer, Germany.

**8 Peer-reviewed conference papers** (2 as first author – ★)

| | |
|---|---|
| – *ANTS 2014* | 9th International Conference on Swarm Intelligence |
| – *ANTS 2014* | 9th International Conference on Swarm Intelligence |
| – *ANTS 2012* | 8th International Conference on Swarm Intelligence |
| – *TAROS 2011* | 12th Conference Towards Autonomous Robotic Systems (★) |
| – *IROS 2011* | IEEE/RSJ International Conference on Intelligent Robots and Systems |
| – *ANTS 2010* | 7th International Conference on Swarm Intelligence |
| – *ICINCO 2009* | 9th International Conference on Informatics in Control, Automation and Robotics |
| – *ANTS 2008* | 6th International Conference on Swarm Intelligence (★, "Best paper" award) |

**2 Video proceedings** (1 as first author – ★)

| | |
|---|---|
| – *IROS 2012* | IEEE/RSJ International Conference on Intelligent Robots and Systems (★) |
| – *AAAI 2011* | 25th Conference on Artificial Intelligence ("Best video" award) |

## Structure of the dissertation

The main part of this dissertation is structured into seven chapters. Please note that, due to the diversity of the topics of these chapters, rather than discussing the state of the art in a separate chapter, I do so at the beginning of Chapter 3, 4, and 5.

In Chapter , I present some general reflections on the role of abstraction in swarm robotics. Furthermore, I discuss the role of robot experiments for engineering-oriented research in swarm robotics.

In Chapter 3, I present the core concepts of this dissertation and discuss why abstracting tasks in laboratory experiments is advantageous over simply simulating them.

In Chapter 4, I provide a definition of complex tasks and their interrelationships. I use this definition as a basis for my approach to modeling complex tasks.

In Chapter 5, I present the design and implementation of the TAM. I discuss the design goals of the TAM and detail how I attained these goals in the implementation of the TAM.

In Chapter 6, I verify the concept of the TAM and the proposed approach to modeling complex tasks on the basis of two proof-of-concept experiments.

In Chapter 7, I present four examples of scientific studies that are based on the TAM: 1) a study of the costs and benefits of behavioral specialization in swarms of robots, 2) a study on property-driven design for robot swarms, 3) a study on self-organized task allocation, and 4) a study on collective decision-making in the context of task partitioning.

In Chapter 8, I summarize the contributions of this dissertation and discuss possible directions for future research.

Several appendices cover the technical aspects of this dissertation. In Appendix A, I describe the materials I rely on in this dissertation: the robot platform employed and the simulation framework used. In Appendix B, I present the technical details of the TAM.

Chapter 2

# Empirical investigation in swarm robotics research

Swarm robotics is a promising approach to applications that benefit from massive parallel task execution by fault-tolerant, flexible, and scalable systems. As stated in Chapter 1, applications that fall into this category are manifold; swarm robotics is therefore expected to become an applied technology in the future. However, at the current stage of the field, its application is hindered by the lack of mature technologies in many areas—for example, robust robot platforms and reliable engineering principles. Swarm robotics is therefore facing a typical "chicken-and-egg" problem: progress in any of these areas generally depends on the other areas being fully developed. For example, the development of engineering principles and group behaviors depend—in theory—on the availability of the final robot platform for a given application. In order to circumvent this problem and to progress in these areas today, researchers use abstractions: experiments are conducted with abstract representations instead of the final robots, environments, and applications.

Abstraction allows researchers to study tomorrow's problems with today's technologies; it is therefore one of the key elements in swarm robotics research. In this chapter, I explore the role of empirical investigation in swarm robotics research and its dependence on abstraction—as we will see, a dependence that affects the type of problems considered in swarm robotics. Furthermore, I discuss the role of robot experiments in swarm robotics research, and relate this role to that of simulation experiments.

By discussing these aspects of empirical investigation in swarm robotics, I wish to show why swarm robotics research should also be conducted with robots rather than solely in simulation. This is especially relevant if one considers that, due to the aforementioned lack of mature technologies at this stage of the field, robot experiments might be just as abstract as their simulated counterparts. In the context of this dissertation, this leads to the following question: why should we use physical representations of abstract tasks if we can simply simulate everything?

The reasoning that I present in this chapter builds on the "scientific vs. engineering" dichotomy proposed by Dorigo and Birattari (2007). Swarm robotics research can be conveniently classified on the basis of the goals that are pursued: some research is of a scientific and speculative nature, while other research is engineering-oriented and has a marked pragmatical nature.

Speculative research considers questions out of scientific interest rather than practical necessity. As such, speculative research is driven by scientific curiosity—while its results might find application in the long-term, applicability is not its main purpose. An example of speculative research is the work presented by Ferrante et al. (2013b), which considers the question of how many informed robots are required to guide a flock of uninformed ones. While this question is certainly interesting, it is not tied to a given application, practical necessity, or robotic platform.

Engineering-oriented research has the goal to develop fundamental approaches to design and engineer artificial systems—systems that will be employed in some, possibly future, application. As such, engineering-oriented research is markedly pragmatic and goal-oriented. In the context of swarm robotics, engineering-oriented research aims to find approaches to design and engineer future swarms of robots (see, e.g. Hamann and Wörn, 2008; Kazadi et al., 2009; Berman et al., 2011; Brambilla et al., 2014; Francesca et al., 2014b). As stated in Chapter 1, this dissertation positions itself within the engineering-oriented research stream of swarm robotics. Accordingly, the reasoning presented in this chapter is to be considered in this context.

The topics that I discuss in this chapter are of a somewhat philosophical nature as I try to isolate and denominate processes that are, at least in the case of abstraction, rarely discussed explicitly in the swarm robotics literature. Additionally, regarding the topic of simulation experiments versus robot experiments, I touch upon a long-

standing debate between researchers. As a result, the positions that I take on these topics are—at least partly—my personal views rather than the scientific consensus.

This chapter is structured as follows. In Section 2.1, I present some general reflections on the role of abstraction in swarm robotics research—reflections that provide a basis for the discussion of task abstraction in Chapter 3. In Section 2.2, I discuss the role of robot experiments for engineering-oriented research in swarm robotics. In Section 2.3, I provide a summary of this chapter.

## 2.1 Abstraction in swarm robotics research

Abstraction serves two main purposes in swarm robotics research: First, it allows researchers to study future applications with today's technologies. To date, swarm robotics is practically *fundamental research*: swarm robotics systems have not yet been successfully applied to real-world scenarios. As stated above, this is partly due to the limitations of today's technologies; as a result, researchers are forced to consider abstract versions of these problems. The rationale behind this is that studying an abstract version of a given problem contributes to developing technologies for related real-world applications. Hence, abstraction of future applications of swarm robotics is especially relevant in engineering-oriented research. For example, consider de-mining: even though robot swarms that are capable of neutralizing a mine do not exist yet, some of the problems presented by collective de-mining have already been studied in abstract experiments (e.g., Cassinis et al., 1999; Kopacek, 2004; Zafar et al., 2006).

Second, abstraction allows researchers to isolate a generic problem from a specific application. In particular, by studying an abstract problem, researchers can ensure that an approach is sufficiently generic to be applicable to other instances of the same problem. For example, consider the following applications: a swarm of robots has to load a truck with sacks of cement; and a swarm of robots has to retrieve human casualties from a disaster site. A problem that arises in both cases is to collectively transport objects from one location to another. By considering an abstract version of this problem, researchers can ensure that their proposed approach can be applied to either of these applications.

Note that abstraction takes places whether the experiments are con-

ducted using robots or in simulation. The following therefore applies to both types of experiments—see Section 2.2 for a discussion of the relation between robot experiments and simulation experiments.

### 2.1.1 Definition, usage, and implications

Engineering-oriented research intends to find solutions to problems that occur in some real-world application, commonly by abstracting from these applications. But what exactly does *abstraction* mean in this context?

In general terms, abstraction is *"the process of considering something independently of its associations or aspects"* (Stevenson, 2010, def. 4). In the context of research, abstracting a problem means that researchers isolate the problem from the real-world application where it occurs. Abstraction is achieved by identifying certain *aspects* of the application that are essential to the problem and retaining these in an abstract description of the problem. Consequently, any aspects of the application that are inessential to the problem are omitted in the description.

Real-world applications present instances of one or multiple problems. Accordingly, they possess the essential aspects of these problems as well as many inessential aspects. On the other hand, laboratory experiments consider problem instances that are typically abstract insofar as they possess few inessential aspects. This allows researchers to focus on the problem of interest without having to consider many inessential and possibly confounding aspects. In other words, abstraction allows researchers to work on a "distilled" version of the problem. Figure 2.1 illustrates the relationship between abstract problems, instances, and aspects.

Ideally, studying an abstract instance of a given problem allows researchers to draw conclusions about all instances of that problem. However, the validity of these conclusions depend on the steps of abstraction and instantiation having been performed properly. Consequently, this step is critical to the design of a study; it lies within the responsibilities of the researcher to identify and report the essential aspects considered in a study.

In the context of swarm robotics, some aspects of a given problem might be properties of the robots or of the environment in which the robots operate. For example, a collective transport problem might require the robots to be equipped with a manipulator capable of lifting

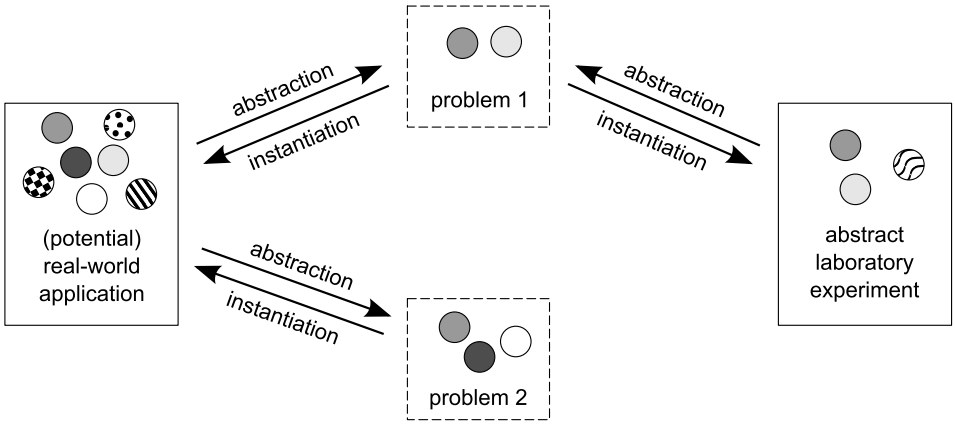



Figure 2.1: Illustration of the process of abstraction as employed in swarm robotics research. Problem instances and abstract problems are represented as boxes with full and dashed borders, respectively. Instances and problems alike possess various aspects, represented as small circles with different colors/patterns. Abstract problems possess only aspects essential to them. Instances such as real-world applications typically possess inessential aspects in addition to the essential aspects of the abstract problems they instantiate. Laboratory experiments are abstract instances in the sense that they typically posses all the essential and only few inessential aspects.

a certain type of object and the environment to contain such objects. In abstract instances such as those studied in laboratory experiments, abstract representations of robots and environment might be used.[1] For example, the aforementioned problem might be studied using an abstract representation of the manipulator-object interaction by using a robot platform equipped with a simplified magnetic gripper and matching objects. In such an experiment, the specific robot platform is considered *in lieu* of a whole class of robots that share the same aspects (i.e., being able to lift a certain type of object). I discuss representations commonly used in the literature for the robots in Section 2.1.2 and for the environment in Section 2.1.3.

[1] I adopt the view that these are two separate but interrelated dynamical systems (see Beer, 1995; Smithers, 1997).

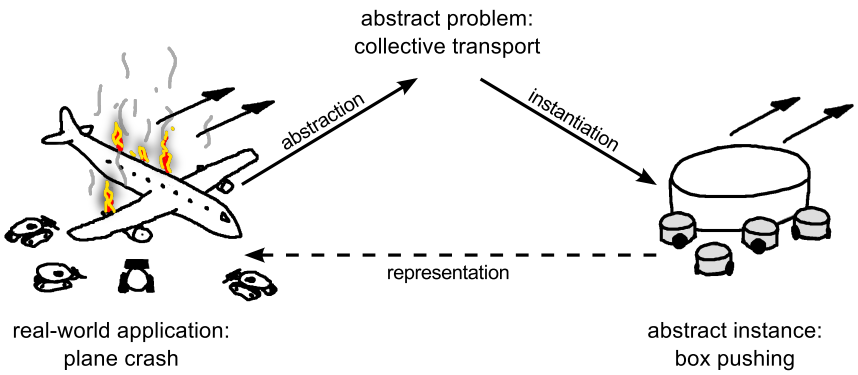



Figure 2.2: Example of an abstraction of a real-world application for study in laboratory experiments. The problem selected for study requires the robots to collectively move the plane wreck (on the left side). The abstract problem is commonly called collective transportation (in the middle). An abstract instance of this problem might employ of a swarm of e-puck robots and a heavy cylinder (on the right side). The e-pucks and the cylinder are abstract representations of some physical aspects of the real-world application.

### Example

Let us consider an example to illustrate the relationship between real-world applications, problems, and abstract representations. Suppose that the real-world application is a disaster scenario—a plane crash. In the scenario, a swarm of autonomous robot bulldozers has been tasked to clear the crash site of the plane wreck, as illustrated in Figure 2.2. In order to address this mission, a large number of problems have to be solved, for example: how to extinguish the burning plane, how to control a robot bulldozer, how to manipulate a plane such that it does not break apart, how to collectively push the heavy plane in order to move it, how to navigate when smoke clouds the sensors, etc. Typically, researchers focus on a subset of the problems that have to be solved for a given application; which problem to study depends on the focus of the researcher and the technologies available for representing the physical aspects of the problem considered in an experiment.

Assume that the problem of choice is a collective behavior to move the plane wreck—a problem commonly called *collective transportation*. The essential aspects of this problem might be that robots can perceive and push an object, and that the object can only be moved by a certain number of robots. Aspects inessential to the problem might

be that a robot bulldozer has a variety of shovel configurations or that there is smoke emitting from the plane. In order to study this problem in a laboratory experiment, researchers would employ abstract representations of the physical aspects considered: the yet-to-be-invented robot bulldozers might be represented with a swarm of e-puck robots (Mondada et al., 2009) and the object might be represented with a cylinder of a specific weight—see Figure 2.2. Exemplary studies that consider problems of this class have been proposed by Kube and Zhang (1993), Campo et al. (2006), Chin et al. (2009), Ferrante et al. (2010), and Rubenstein et al. (2013).

Another problem that arises in the same scenario might be related to how robots manipulate the object to be transported. In this case, the essential aspects of the problem are different from the collective transportation problem: the manipulator of the robots and the geometry of the object in question might be the essential aspects of the problem. Consequently, the abstract representations required to study this problem in the laboratory also differ from the ones used for the collective transportation problem. Exemplary studies that consider problems of this class have been proposed by Donald et al. (1997), Baldassarre et al. (2006), Baldassarre et al. (2007), and Groß and Dorigo (2009).

### Impact of representations and their availability

As the example scenario shows, the study of different problems might require different representations for robots and environments. Consequently, the availability of abstract representations for robots and environments defines the problems that can be studied by the researcher. In other words, there exists a process opposite to the process of abstraction described in Figures 2.1 and 2.2: the availability of certain representations dictates the set of problems that can be studied.

One important effect of this constraining process is that researchers often study the same kind of problems. This is due to the fact that studying problems outside of the aforementioned restricted set of problems is costly as it requires different, possibly custom-made representations. This effect is well-known in case of robots and their capabilities:

> *It is clear that technological constraints have limited the scope of implementations and task domains attempted in multiple-robot research systems.*
>
> (Cao et al., 1997)

However, it is less known in the case of environments where the same effect can be observed: *ad hoc* representations available define the environments—and hence problems—that can be considered in a cost-efficient manner.

This effect creates an issue for the whole of swarm robotics research: the frequency at which a given problem is considered in the literature is strongly influenced by the availability of the required representations rather than by its relevance to any real-world application. For example, foraging and flocking are often referred to as "canonical problems" in the literature (e.g., Østergaard et al., 2001; Martinoli et al., 2004; Campo et al., 2006; Lein, 2010), a term that implies that these problems are of great importance to many real-world applications. However, I speculate that these problems became "canonical" due to the fact that they can be conveniently studied rather than due to their relevance to any real-world application.

### 2.1.2 Abstract robots

Robot platforms used in the swarm robotics literature can be considered abstract representations of classes of future robots. Strictly speaking, researchers do not abstract existing robots as, to date, no robot swarms exist that have been applied to real-world scenarios. Instead, researchers anticipate which aspects of a robot might be required for a given problem and use available robots that possess said aspects—a process that is very close to abstracting an existing robot.

In an experiment, the robot platform employed possesses certain aspects essential to the problem considered. Conclusions drawn on an approach to this problem using this specific robot platform therefore apply to any other existing or future platform that shares these aspects. In other words, by studying an approach to a problem using a specific platform, researchers can demonstrate that the proposed approach works on a whole class of robot platforms.

Unfortunately, which aspects of the employed robot platform are essential to a study is rarely discussed explicitly in the literature. Exceptions include works that consider automatic generation of controllers such as evolutionary robotics. For example, Jakobi (1997) identified a *base set* of robot-environment interactions relevant to the problem at hand, which he uses to develop minimal simulations of robots. Similarly, Francesca et al. (2014b; 2014a) recently proposed to use a *reference model* to specify the aspects of the e-puck platform essential to

| **Sensors/Actuators** | **Variables** |
|---|---|
| Proximity | $prox_i \in [0,1]$, $\angle q_i$, with $i \in \{1,2,\ldots,8\}$ |
| Light | $light_i \in [0,1]$, $\angle q_i$, with $i \in \{1,2,\ldots,8\}$ |
| Ground | $gnd_i \in \{0, 0.5, 1\}$, with $i \in \{1,2,3\}$ |
| Range and bearing | $n \in \mathbb{N}$ and $r_m, \angle b_m$, with $m \in \{1,2,\ldots,n\}$ |
| Wheels | $v_l, v_r \in [-\bar{v}, \bar{v}]$, with $\bar{v} = 0.16\,{}^{\mathrm{m}}/_{\mathrm{s}}$ |

Table 2.1: Aspects of the e-puck platform essential to the problem considered by Francesca et al. (2014b), formally specified in form of a *reference model*: $prox_i$ is the reading of the $i$-th proximity sensor and $\angle q_i$ is the angle at which the $i$-th proximity sensor is positioned with respect to the head of the robot; $light_i$ is the reading of the $i$-th light sensor and $\angle q_i$ is the angle at which the $i$-th light sensor is positioned with respect to the head of the robot; $gnd_i$ is the reading of the $i$-th ground sensor; $n$ is the number of robots in the neighborhood; $r_m$ and $\angle b_m$ are respectively the range and bearing of the $m$-th neighbor; $v_l$ and $v_r$ are respectively the speed of the left and right wheel; and $\bar{v}$ is the maximum speed of the robot. Sensors and actuators are updated with a period of 100ms. For further details on the reference model see Francesca et al. (2014b).

the problem at hand. The proposed reference model formally defines the required capabilities of a robot in order to generalize the proposed approach and compare it to other approaches—Table 2.1 reproduces their reference model.

However, despite the prevalence of using abstract robots for empirical investigation, little work has been devoted to the standardization of robot capabilities and to the creation of tools that provide standardized abstractions (Pinciroli, 2014). Instead, researchers employ widely available platforms such as the e-puck (see Appendix A), which is becoming a *de facto* standard in the robotics community.

Unfortunately, the usage of *de facto* standards for robot platforms has several issues. For one, in absence of a reference model as used by Francesca et al. (2014b), the definition of the aspects essential to the study is implicit and cannot be easily discerned from the technical description of the platform. Another issue is the previously mentioned bias when selecting problems to study: researchers consider problems and approaches that can be studied with robot platforms already available in the laboratory, omitting problems that might re-

quire other robot platforms. Still another issue stems from the fact that ideally, researchers should use different robot platforms that support the same set of aspects in the evaluation of an approach. Studies that omit this step might obtain results that depend also on inessential aspects of the chosen platform rather than exclusively on the essential aspects. This, in turn, might render the proposed approach less transferable to other robot platforms where these inessential aspects are not present (e.g., robots employed in a real-world application). This issue is closely related to the difference observed between simulation and reality, commonly called the "reality gap"—see Section 2.2.

### 2.1.3 Abstract environments

Environments used for laboratory experiments in swarm robotics must satisfy several constraints. First, as discussed above, they must retain the essential aspects of the original problem that they are supposed to abstract. For example, the environment might be required to contain a hill with a certain inclination or a trough that cannot be crossed by a single robot (see, e.g., Christensen et al., 2007; O'Grady et al., 2010; Mathews et al., 2012).

Second, environments must allow for empirical investigation with an available robot platform. In particular, a given environment must accommodate for the limitations of sensors and actuators of the robot platform employed in the experiment. For example, the environment might be required to contain areas with a special floor color or other visual cues to accommodate for the image processing capabilities of the robots (see, e.g., Tuci et al., 2004; Campo et al., 2011; Werfel et al., 2014).

Third, environments must be realizable with the technologies and resources available to the researcher. For example, consider artificial pheromones: even though a large body of works in biology and computer science document the utility of indirect communication by pheromones, few studies investigate their utility for robotic systems. This is most likely due to the fact that the environments and robots required are complex and costly to implement (see, e.g., Payton et al., 2001; Purnamadjaja and Russell, 2007; Garnier et al., 2013; Fujisawa et al., 2014).

Due to these constraints, researchers commonly conduct experiments in highly abstract environments that are tailor-made for a specific study. In other words, environments are abstracted on an *ad hoc*,

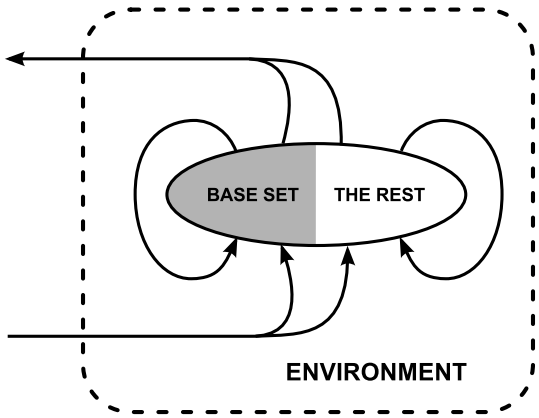



Figure 2.3: Base set of aspects of an environment, defined as those aspects that have interactions with the relevant parts of the system. Reproduced from Jakobi (1998).

per-experiment basis—contrarily to robots, which typically persist in the literature for a few years. As a result, the process of abstracting environments is usually even less structured than the one for abstracting robots.

Notwithstanding this lack of structure, the abstraction of environments has been discussed in the context of simulation (Jakobi, 1993; Pinciroli, 2014). As with any abstraction, the issue lies with the selection of those aspects of the original environment that must be retained in the abstract simulation environment. Again, studies of automated controller generation address this issue more explicitly than other studies—most likely due to the fact that they depend heavily on simulation experiments and thus suffer from the "reality gap" (see Section 2.2). For example, Jakobi (1993, 1998) discussed abstraction and selection of an environment's essential aspects explicitly by proposing an approach for identifying what he calls a "base set" of environmental aspects that must be retained—see Figure 2.3.

In some cases, robots interact with the environment purely by arranging themselves in space, for example, in spatially organizing behaviors such as flocking and navigation behaviors. In other cases, robots interact with the environment through physical objects. For example, in the previously mentioned plane-crash scenario, the robots have to interact with the plane in order to perform the collective transport task. If the physical object is an essential aspect of the problem, the abstract representation used for this object in an experiment is a critical part of the experimental design.

Abstraction and representation of physical objects is related to task abstraction as considered in this dissertation. Task abstraction in general is the process of considering a task independently of those of its

aspects that are not relevant for the study at hand. Task abstraction as considered in this dissertation concerns tasks that involve interactions of the robot swarm with physical objects or their abstract representations. In Chapter 3, I define task abstraction in more detail, discuss the state of the art, and present a novel approach to abstracting and representing tasks in a generic way.

## 2.2 Robot experiments vs. simulation experiments

Since the creation of computer-based simulations of robot systems, researchers have been discussing whether simulations of robot systems are sufficiently accurate to predict the behavior of their counterparts in reality. This discussion is becoming even more relevant due to the fact that today's simulators are significantly more accurate than those available in the nineties when researchers became first aware of the issue. As a result, simulation experiments are not only an integral part of swarm robotics research, but are also starting to replace experiments involving robots entirely. After all, if the studied system is highly abstract as discussed in Section 2.1, why not simulate everything? In this section, I propose my answer to this question, first by discussing simulation experiments and the "reality gap", then by discussing the role of experiments involving robots.

### 2.2.1 Simulation and the "reality gap"

Simulations not only reduce the cost of developing new approaches but can also help to minimize the risk of harming humans and damaging robots (Pinciroli, 2014). As a result, studies of approaches that require many iterations (e.g., automatic controller generation) or large groups of robots depend heavily on simulation experiments. Consequently, simulations are an essential tool for developing swarm robotics systems.

However, notwithstanding all their advantages, simulations have long been criticized for their potential of producing systems that are overly simplified (Brooks, 1987, 1991; Smithers, 1994). Critics commonly question the usefulness of simulations with the argument that systems developed in simulation often do not translate to real-world systems:

> *There is a real danger (in fact, a near certainty) that programs which work well on simulated robots will completely fail on real robots because of the differences in real world sensing and actuation—it is very hard to simulate the actual dynamics of the real world.*
>
> (Brooks, 1992)

Indeed, systems developed in simulation often exhibit bad performance when transferred to reality (Brooks, 1992; Jakobi et al., 1995).

The "reality gap" refers to the difference between simulation and reality (Jakobi, 1998; Francesca et al., 2014b); accordingly, transferring a system developed in simulation to reality is commonly called "crossing" the reality gap. Systems that are developed exclusively in simulation are sensitive to the reality gap—for example, systems developed using approaches based on artificial evolution such as evolutionary robotics (Jakobi et al., 1995; Nolfi and Floreano, 2000). The effect of the reality gap is particularly relevant in complex systems, where small but unavoidable differences between simulation and reality could lead to widely diverging behaviors. Swarm robotics systems are an example of complex systems that are highly sensitive to the reality gap—a sensitivity that, depending on the goals of the researcher, might be a critical factor when evaluating approaches.

### 2.2.2 The role of robot experiments

Arguably, the reality gap is shrinking due to the aforementioned increase in accuracy of today's simulators. For example, modern simulators such as ARGoS (see Pinciroli et al., 2012, and Appendix A) are able to simulate motion dynamics, sensor noise, and other aspects with reasonable accuracy. Thus, considering the capabilities of today's simulators, what is the justification for experiments with robots? Moreover, if—as stated in Section 2.1—all experiments in swarm robotics consider only highly abstract versions of the original problem, why not simulate everything?

The answer to this question is of a philosophical nature, and might be answered differently by different roboticists. In my opinion, there are two main reasons for conducting experiments with robots, both related to the transferability of approaches between simulated and real systems.

The first reason is that conducting both simulation and robot experiments allows researchers to draw conclusions about the transfer-

ability of their approach between robot platforms. This is due to the fact that simulated robots always differs from their real-world counterparts (Brooks, 1992; Pinciroli, 2014). In other words, simulated robots are essentially another robot platform that shares some aspects with the real-world robot platform. As discussed in Section 2.1.2, by studying an approach on two different robot platforms that share a certain set of aspects, researchers can draw conclusions on the applicability of the approach to all platforms with this particular set of aspects. In this context, researchers can not only draw conclusions about the transferability of their approach between simulation and physical systems, but also about its transferability to any robot platform that shares the same set of aspects—existing or future. This process is illustrated by the vertical arrow on the left side of Figure 2.4. Consequently, using both types of experiments for studying the same system has the potential to alleviate the problem mentioned in Section 2.1.2: approaches that are evaluated exclusively on a single robot platform might yield results that depend on inessential aspects of this platform rather than on the essential aspects of the problem.

The second reason is related to the motivation of the research considered. As previously mentioned, speculative research is not tied to a given application, practical necessity, or even robotic system. Consequently, speculative research can, in most cases, be conducted solely in simulation—experiments with robots are typically not a strict requirement. Engineering-oriented research, on the other hand, aims at developing fundamental approaches for designing and engineering artificial systems—systems that will be adopted in (possibly future) applications. I expect that future robot swarms will be engineered following the same principles used in most areas of engineering today: systems are developed using computer-aided simulations and subsequently transferred to the real world—this process is illustrated by the vertical arrow on the right side of Figure 2.4. Another reason to assume that simulations will play a major role in the development of future robot swarms is due to the complexity of robot swarms—it is not to be expected that they could ever be developed using only analytical methods. By following the same engineering principles today, researchers can assess whether an approach can be successfully transferred between simulation and robots (i.e., cross the reality gap). I speculate that approaches robust enough to sustain this transfer are more likely to be transferable to future real-world applications—this process is illustrated by horizontal arrows in Figure 2.4. Accordingly,

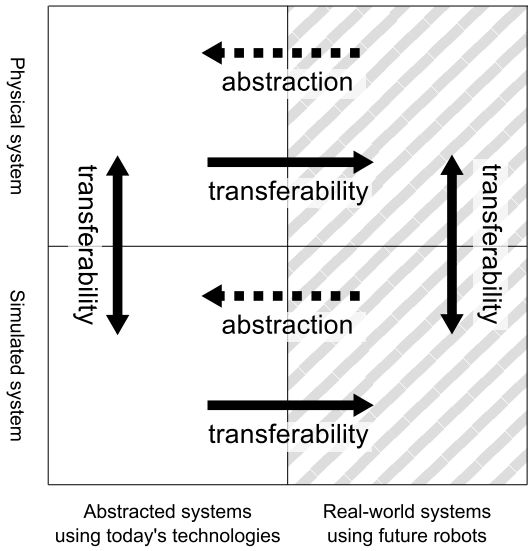



Figure 2.4: Relationship between experiments involving real and simulated versions of today's abstract and future real-world systems. Researchers abstract future systems to ones based on today's technologies. Approaches that are transferable between abstract simulated and real-world systems can be transferred to future systems, provided that the initial abstraction faithfully captured the essential aspects of the problem.

I propose that robot experiments are, together with simulation experiments, a requirement for engineering-oriented research in swarm robotics.

For these two reasons, I consider the assessment of a system using robot experiments essential to any engineering-oriented research in swarm robotics. As a result of the discussion in this section, I can make two conclusions: First, speculative research is free to chose any system that retains the essential aspects of the problem considered, be it simulated or real. Second, simulation experiments are just as essential for engineering-oriented research in swarm robotics as robot experiments.

## 2.3 Summary

In the first part of this chapter, I established that all experiments conducted to date in swarm robotics are heavily abstracted and that the process of defining this abstraction is a crucial but often implicit step in the design of an experiment. I reviewed the few explicit approaches to abstraction found in the literature and discussed their common trait: isolating the aspects of a system that are essential to the problem studied.

The key finding of this chapter is that, to date, little effort has been made to systematically abstract the experimental environment. This is especially relevant considering the fact that the available representations strongly influence the type of problems that can be studied by researchers. In other words, the lack of a systematic approach to abstraction limits the variety of problems studied in swarm robotics. Furthermore, the *ad hoc* representations used to date threaten the *ceteris paribus*[2] assumption: the reproducibility of studies is generally low in swarm robotics.

In the second part of this chapter, I argued that experiments involving robots are essential to any engineering-oriented research in swarm robotics. This also has implications for the type of problems that can be studied: robot experiments are even more constrained in the variety of problems than simulation experiments due to the fact that creating the required physical representations of the environment is difficult and costly.

In the following Chapter 3, I introduce a novel set of tools to alleviate these issues: a generic abstract representation for tasks in robot experiments and a modeling approach that allows researchers to abstract tasks of various complexity.

[2] lat., with other things being the same

Chapter 3

# Task abstraction and the TAM

Task abstraction is prevalent in the swarm robotics literature: as established in Chapter 2, almost every study is conducted in abstract laboratory environments where robots perform highly idealized tasks. While this fact is well understood, to date no effort has been made in the literature to formalize the concept of task abstraction. In this chapter, I introduce a novel set of tools to abstract and represent tasks of various complexity in robot experiments.

This chapter is structured as follows. In Section 3.1, I define task abstraction as used in the context of this dissertation. In Section 3.2, I review the state of the art in task abstraction. More specifically, I discuss the various types of *ad hoc* solutions used for task abstraction in the literature. In Section 3.3, I outline the novel tools for task abstraction that are the main contributions of this dissertation. More specifically, I introduce the TAM: a device that can represent a single-robot stationary task in laboratory experiments. I outline how to abstract complex multi-robot tasks so that a group of TAMs can be used to represent them. Furthermore, I discuss the advantages and limitations of the TAM and describe which problem classes can potentially benefit from it. In Section 3.4, I provide a summary of this chapter.

Note that the terms introduced in this chapter are closely related to further terms introduced in Chapter 4. In order to facilitate understanding, I summarize the key terms used in this dissertation in Table 3.1.

| Term | Definition | Section |
|---|---|---|
| task | A unit of work that has to be performed by the robots. | 3.1.1 |
| stationary task | Tasks that require robots to be busy for a given duration at a specific location and at a specific moment in time. | 3.1.1 |
| single-robot task | A task that can be performed by a single robot in a defined time window. | 3.3.1 |
| stationary single-robot task | A task that is both single-robot and stationary. | 3.3.1 |
| atomic task | A task that cannot be decomposed into subtasks. In the context of this dissertation, atomic task are always stationary single-robot tasks. | 4.2.2 |
| complex task | A task that can be decomposed into atomic subtasks that have interrelationships between them. | 3.3.1, 4.2.2 |
| interrelationship | A mutual relationship between two atomic tasks that defines the conditions of their execution. | 3.3.1, 4.2.2 |
| sequential interrelationship | An interrelationship between subtasks that requires that these subtasks are executed in a given order. | 4.2.2 |
| concurrent interrelationship | An interrelationship between subtasks that requires that these subtasks are executed at the same time. | 4.2.2 |

Table 3.1: Glossary of key terms used in this dissertation. "Section" indicates the section number in which the respective term is defined. Note that I indicate two section numbers for "complex task" and "interrelationship" as these terms are redefined more precisely in Chapter 4.

## 3.1 Task abstraction

As mentioned in Chapter 2, I define *task abstraction* as the process of considering a task independently of those of its aspects that are not relevant for the study at hand.

In this dissertation, I consider abstraction of tasks that involve interactions of the robot swarm with physical objects. The abstraction and representation of the involved physical objects in an experiment is central to the study of any problem that concerns tasks of this kind—for example, studies of task allocation and task partitioning.

### 3.1.1 Stationary tasks

In the context of this dissertation, a *task* is a unit of work that has to be performed by the robot swarm. A *stationary task* is a task that involves interactions of the robot swarm with some stationary physical objects: in order to address a stationary task, robots have to interact with these objects. More precisely, stationary tasks are tasks that require robots to be busy for a given duration at a specific location and at a specific moment in time.

Stationary tasks can be abstracted by isolating and reproducing the aspects that are relevant to the problem to be studied, while disregarding all irrelevant aspects. More specifically, stationary tasks can be studied in an experiment using abstract representations of these tasks that retain the relevant aspects of the tasks at hand (cf. Chapter 2). The specification of these essential aspects is highly domain-dependent; it is therefore the obligation of the researcher to identify and describe these aspects.

Commonly, the essential aspects of a stationary task include, but are not limited to:

- the capabilities of the robots required to perform it,
- the location at which robots have to interact with the stationary task,
- the time at which the task becomes available, and
- the duration of performing the task.

For example, assume two buttons in the environment that can be pressed by the robots. The task of pressing a button can be represented

in an experiment with an abstract object, a specially demarcated area, or a real button—basically anything that retains the essential aspects of the task: capabilities, location, time, and duration. Other essential aspects of a single task can be envisioned, for instance, that it can only be performed by a specific robot or a subset of the available robots.

### 3.1.2 Interrelationships

Continuing with the same example, assume further that the buttons must be pressed at the same moment in time in order to achieve some higher-level goal. Accordingly, the abstract tasks must be executed concurrently—that is to say, there exists an *interrelationship* between those tasks. This interrelationship is an essential aspect to the higher-level goal, which can be seen as a task that includes the two stationary tasks (the buttons) and their interrelationship (concurrency).

For example, consider an extension of the plane-crash scenario of Chapter 2 in which robots first have to dump remaining fuel from the plane, then rescue the remaining victims, and eventually push the wreckage away. The logical relationship between the tasks is that they have to be executed one after the other—an aspect of the problem that must be retained in an experiment if this relationship is essential to the problem.

For further examples, consider a hypothetical swarm-operated assembly line where one group of robots drills holes through several parts and another group subsequently bolts them together; or imagine a swarm of nano-bots that extirpate cancer cells: one group of robots identifies and marks tumors; another group subsequently destroys them.

While task execution is completely different in these examples, the logical relationship between the tasks is the same: the tasks have to be executed one after the other. If the focus of the research is to develop coordination mechanisms that allow a swarm to tackle tasks with this kind of logical relationships, it might be desirable to isolate the logical relationship from the details of task execution and focus on it, rather than spending resources on inessential aspects of the implementation. Consequently, the essential aspects of the problem, and thereby the tasks and their representation in the physical space, include this interrelationship.

Conventionally, stationary tasks without interrelationships are represented in an experimental environment by using special objects or

other robots—see Section 3.2 for a review of the state of the art. However, representing interrelationships between tasks is not trivial due to the fact that the interrelationships do not have readily available representations that lend themselves to experimentation. As a result, researchers either ignore interrelationships or study those that can be represented with little effort in an experiment. This is one of the reasons for the effect discussed in Chapter 2: the task representations available to a researcher limit the problems that this researcher can study.

## 3.2 State of the art

In this section, I discuss the different types of *ad hoc* solutions used in the literature for task abstraction. Furthermore, I discuss the advantages and disadvantages of each type, which provides me with important insights for the conceptual design of the TAM in Section 3.3 and its implementation, discussed in Chapter 5.

### 3.2.1 Classification of the literature by task abstraction used

In the following, I classify the literature depending on the type of *ad hoc* solution used to abstract tasks. I distinguish between these solutions:

- virtual (simulated and/or disembodied tasks);
- passive objects (inanimate physical objects);
- active objects (some form of sensing and/or actuation present);
- robots (a powerful form of active object).

For the sake of brevity, I refer in the following to a given *ad hoc* solution for task abstraction simply as *solution*. Note that not all solutions used in the literature fit this rigid classification scheme and might possess elements of several of the solutions identified by me.

### Virtual

Virtual solutions to task abstraction represent a stationary task in a virtual way: task execution is simulated in the robot's controller rather than leaving an impression on the environment. As such, experiments using this type of solution "blur the line" between robot experiments and simulated experiments—the environments using virtual solutions can be considered an "augmented" reality.

An example for a task abstracted using a virtual solution is an object retrieval task where objects are not physically present. Instead, objects are simulated using a special logic internal to the controller of each robot. In this case, retrieval and deposit of objects are possible when the robot is at specific locations, commonly designated with some environmental cue (e.g., different ground colors for the source and nest). This type of virtual solution is commonly used when studying foraging tasks with robots that do not possess advanced manipulation capabilities, such as works using the Jasmine robot[3] (Kernbach et al., 2012), the e-puck[4] (Acerbi et al., 2007; Alers et al., 2011; Francesca et al., 2014b; Brambilla et al., 2014), or the Kilobot[5] (Rubenstein et al., 2011).

Other works represent the problem of distinguishing between options of different quality using paths between two locations. In such a scenario, the quality of a path is the cost associated with taking this path and different qualities can be modeled by varying the relative lengths of the paths. For example, Scheidler et al. (2014) used an experimental setup similar to the famous "double bridge" experiment (Goss et al., 1989; Deneubourg et al., 1990) to abstract two tasks of differing duration. A similar task has been studied by Campo et al. (2010b), with the difference that the swarm uses pre-established chains of robots between the two locations rather than moving in the environment.

### Passive objects

Passive objects are inanimate physical objects that can be transported or otherwise manipulated by the robots. Passive objects are commonly used in foraging experiments as they evidently retain the essential as-

[3] http://www.swarmrobot.org/
[4] Mondada et al. (2009)
[5] Rubenstein et al. (2012)

pects of various object transportation problems. However, some form of abstraction is commonly present, for example in form of specific colors or shapes to accommodate the limited sensing and manipulation capabilities of the robots (see Brutschy et al., 2012a; Pini et al., 2013a, for an example of objects that have been designed for a specific robot platform).

Most studies that use passive objects consider single-robot foraging tasks (e.g., Parker, 1998; Krieger and Billeter, 2000; Labella et al., 2006) or bucket-brigading tasks (e.g., Fontan and Matarić, 1996; Goldberg and Matarić, 2002; Pini et al., 2013a). In both type of tasks, the manipulation capabilities required are limited to the pickup and deposit of the object. The Swarmanoid project (Dorigo et al., 2013) studied an object retrieval task using a book as the passive object. Here, an essential aspect of the task is that it cannot be executed by all robots of the swarm. Instead, only robots of a certain type are equipped to perform the task by gripping the object.

Similarly to foraging tasks, collective transport tasks have been abstracted with passive objects that have to be transported by the robots (Donald et al., 1997; Kube and Bonabeau, 2000; Groß and Dorigo, 2009). In these works, an essential aspect of the task is that the object cannot be moved by a single robot—the representation used is designed such that a specific number of robots is required to move it, typically by ensuring a specific weight of the representation.

In their seminal stick-pulling experiment, Ijspeert et al. (2001) used passive objects to study a problem where two robots have to collaborate to extract a stick from a hole in the ground. The length of the stick requires robots to act sequentially: first one robot pulls the stick from the hole to approximately half its length; then another robot continues to pull the stick so that it can be fully extracted from the hole. The essential aspect of the problem, other than the sequential interrelationship, is that the overall task fails if the first robot aborts before the second arrives (a "blocking" sequential interrelationship). This aspect is represented by the fact that the stick falls back into the hole if released.

### Active objects

Active objects are (usually custom-built) devices that possess some form of sensing or actuation. Using tailor-made active objects for task abstraction is less common in the literature, most likely due to the

cost of building a custom device.

Matarić et al. (2003) used active objects to represent alarms that have to be attended by the robots. In the study, objects are essentially speakers that sound an alarm that can be sensed by the robots. Additionally, objects are marked with passive color patches for visual recognition. The essential aspect of the problem was that alarms can sound at random times. The use of active objects allowed the authors to represent this aspect—something that would not be possible with passive objects.

Tuci et al. (2006), Nouyan et al. (2009), and Baldassarre et al. (2007) used an active device to study a collaborative transport task. The device, called "s-toy", is designed to work in conjunction with the s-bot robot: it is equipped with LEDs for signaling different task types and a special ring that allows it to be gripped by the s-bot. The essential aspect of the problem retained by this representation is the fact that the object's weight forces the robots to collaborate in order to move it. To this end, the experimenter can regulate the weight of the object so that varying numbers of robots are required for the task. The fact that the objects possess a special ring grippable by the s-bot represents another aspect of the problem: robots can pull objects.

Kube and Bonabeau (2000) used similar, robot-specific objects to represent a collaborative transport task, although with the restriction that robots can only push the object. The problem considered by Kube and Bonabeau (2000) is therefore different from the one studied by Tuci et al. (2006) and Baldassarre et al. (2007).

A class of active objects that is commonly used for task abstraction are research robots. In this case, a subset of the available robots act as tasks (e.g., objects to be acted upon), while the rest of the robots are available for experimentation.

Several works focusing on collective transport used robots for task representation (Ferrante et al., 2013a; Dorigo et al., 2013). Similar to the collective transport tasks discussed above, the essential aspect retained is that a robot can only be moved by several others, either by pushing and pulling.

Brutschy et al. (2014c) used s-bot robots to represent "prey" objects to be harvested and transported by a swarm of s-bot robots. The robots used as prey objects are actively controlled to change the colors of their LEDs, which facilitates sensing by the robots. Furthermore, robots representing objects were mounted on sliding platforms to account for the aspect that objects can be transported by a single robot.

Another essential aspect of the problem is that transfer of objects between two robots is blocking, that is, both robots must grip the object together in order to transfer it. This aspect was represented changing the LEDs of the prey object when both robots gripped the object.

### 3.2.2 Discussion

In the following, I discuss the advantages and disadvantages of the different solutions used in the literature.

Virtual solutions possess the convenience and flexibility of simulated experiments: all parameters are internal to the robot and can be changed with little effort. Furthermore, the development of an abstract task representation can happen alongside with the development of the robot's control software. As a result, researchers using virtual solutions can control the experimental setting in a convenient manner.

Unfortunately, there are many disadvantages of virtual solutions. First, a clear separation between the logic that controls the robot and the logic that controls the tasks is difficult to achieve and can lead to mistakes that threaten the rigor of the experiment. Second, certain aspects of a given task such as congestion or resource limitations cannot be easily represented. Third, task representation is dependent on the hardware of the robots and a malfunction might threaten the integrity of the experimental records. Fourth, some problems might possess task-related aspects that are difficult to implement in a decentralized manner—for example, consider a problem in which the availability of tasks follows complex temporal patters.

The advantage of passive objects is that they are perfectly suitable for simple foraging tasks, as illustrated by the review of the literature. Furthermore, passive objects are widely available and cheap to obtain.

However, representing task interrelationships with passive objects is limited to simple sequential interrelationships as in foraging. More complex foraging tasks can theoretically be conceived, but the complexities of the manipulation behaviors and the resulting cost of the experiment effectively prevent researchers from studying foraging tasks with complex interrelationships. Additionally, the robots have to be equipped to manipulate the passive objects, which is not the case for the vast majority of robots employed in swarm robotics research. Finally, passive objects can neither represent essential aspects of a task such as its duration nor can they represent tasks that require disparate robot capabilities.

Active objects combine the flexibility of virtual solutions with the physical tangibility of passive objects—their advantage is that they allow for the flexible definition of a task's essential aspects while being a tangible object in the physical space. The disadvantages of tailor-made active objects are their cost and the effort required in order to create them. Reusable, generic active objects would solve the problem of cost, but would require even more effort to create. As a result, to date, all custom-built active objects are highly specialized and can only be used for certain limited problems.

Begin active objects themselves, robots share the advantages of tailor-made active objects with the difference that robots are readily available. However, depending on the robot platform employed, this choice has several drawbacks. First, many simple robots such as the e-puck have only single-color LEDs and thus cannot represent tasks of different types. Second, robots often lack scalable wireless communication capabilities[6] and therefore can be used neither to represent tasks with complex interrelationships nor to collect reliable statistics in a large experiment. Third, when conducting swarm robotics experiments, robots are typically a scarce resource: swarms are required to be large in order to observe the desired group dynamics. Using some of the available robots to represent tasks in an experiment reduces the size of the swarm that can be studied, and therefore limits the type of studies that can be undertaken with the swarm.

Note that the first and second drawback depend on the capabilities of the robots employed, and might therefore be alleviated by using more complex robots. Unfortunately, this aggravates the third drawback, making this solution only viable if cost is not a limiting factor for the experiment at hand.

In comparison with the other solutions mentioned, robots have the advantage that they possess sensors that can be readily employed to record experimental data. However, robots suffer from some noise and uncertainty in their perception of the environment, caused by imperfect sensors. If the experimental framework relies on the same sensors,

[6] Even though the great majority of robots possess some means of wireless communication, these robots are commonly not suitable for representing a task due to the limited scalability and range of their communication means. For example, Bluetooth is limited to a maximum of 8 short-range, point-to-point serial connections; wireless Ethernet is not well-suited due to scalability issues beyond tens of clients as it uses a centralized network topology rather than a meshed network topology.

and thus suffers the same noise and uncertainty as the robots themselves, the accuracy of the experimental records might be undermined.

## 3.3 The TAM: A novel approach to task abstraction and representation

In this section, I introduce the main contribution of this dissertation: a structured approach to abstracting and representing tasks that enables researchers to represent the logical interrelationships between tasks in a generic and flexible manner. I begin by defining stationary single-robot tasks and continue by describing how to represent such tasks in laboratory experiments using a special device that I call the TAM. I then outline how this device can be used to represent more than just single-robot tasks; more specifically, how the TAM can be used to represent tasks that consist of many interrelated subtasks. Furthermore, I discuss the advantages and limitations of the TAM.

### 3.3.1 Task abstraction and representation

As stated in Section 3.1, I only consider the abstraction of stationary tasks in this dissertation, that is, tasks that involve interactions of the robot swarm with physical objects at a specific location. In the following, I define two types of abstractions that act as the basic "building blocks" for modeling tasks of various complexity.

#### Stationary single-robot tasks

In the context of this dissertation, I call a *single-robot task* a task that can be performed by a single robot in a defined time window. Furthermore, I call a *stationary single-robot task* a task that is both single-robot and stationary. In other words, a stationary single-robot task is an abstraction of any task that requires a robot to be busy for a given amount of time at a specific location and at a specific moment in time. Examples of stationary single-robot tasks are "push a button", "hold a door open for a given amount of time", and "guard a specific location".

Conceptually, the TAM is a device that can represent single-robot stationary tasks, that is, their presence in the physical space. Task representation by the TAM is preceded by a step of task abstraction:

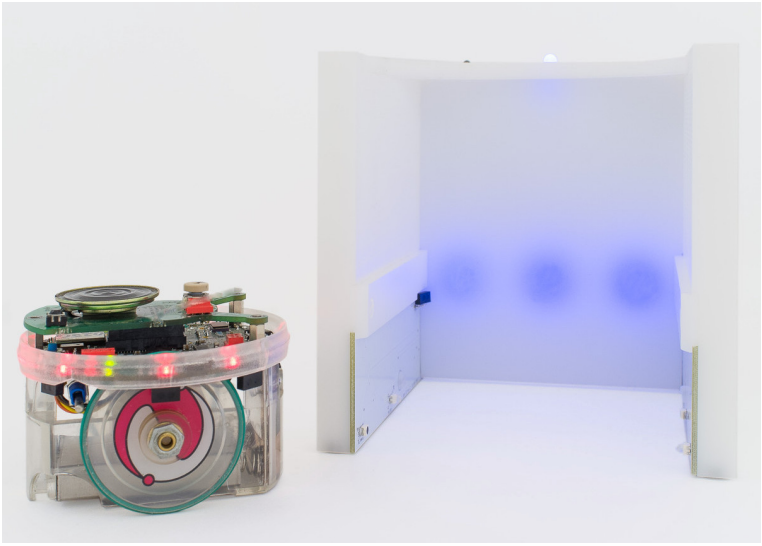


Figure 3.1: A picture of an implementation of the TAM concept for the e-puck robot. The TAM is an abstract representation of a stationary single-robot task. The behavior of multiple TAMs can be coordinated to implement interrelationships between the complex tasks they represent.

the original task is abstracted to those of its aspects essential to the study at hand (see Section 3.1). The TAM can also represent variable aspects of a task, for example, duration and availability in time. In this section, I present the general concept of the TAM. See Chapter 5 for an implementation of this concept intended for use with the e-puck robot (see Appendix A for more information on the e-puck).

Physically, the TAM is an object of roughly cubical shape that can be conveniently placed in an experimental environment. Each TAM represents an instance of a stationary single-robot task. See Figure 3.1 for a picture of an implementation of the TAM for the e-puck robot.

The TAM can announce the presence and availability of the task it represents, for example, by using visual communication. Different types of tasks can be represented by using different signals. The signal emitted by a TAM can be perceived by the robots: if a robot detects a TAM in its proximity, it can decide to work on the associated task by moving to the location of the TAM. Once a robot has moved to the location of the TAM, it is considered to be working on the stationary single-robot task represented by the TAM. The robot has to remain

at the location of the TAM for the duration of the task in order to successfully complete the task.

#### Complex tasks and interrelationships

I call a *complex task* a task that can be abstracted by decomposing it into a collection of stationary single-robot tasks, referred to as subtasks. These single-robot subtasks might require to be performed in a given order or concurrently. I call the logical and hierarchical structure of the subtasks the *interrelationship* between subtasks. The essential aspects of a complex task is the sum of the essential aspects of its subtasks, plus the interrelationships between these subtasks.

Stationary single-robot subtasks might be contiguous in space or might take place at different locations. In the latter case, robots might need to travel from one location to another in order to carry out the complex task. This implies that a complex task that is composed of stationary single-robot subtasks could be non-stationary.

Examples of complex tasks are "harvest an object and store it at a given location", "open a faucet while holding a bucket under it", and "push two buttons at the same time at different locations".

Note that complex tasks are not necessarily multi-robot tasks. For example, a complex task might require a single robot to execute several single-robot subtasks, one after the other. However, due to the nature of the TAM, it is not possible for a single robot to execute two tasks at the same moment in time.

In Chapter 4, I present a novel approach to modeling complex tasks as a set of single-robot subtasks and their interrelationships. In the resulting model, the complexity of the original task lies in the interrelationships between its single-robot subtasks. In order to represent these interrelationships, the behavior of the TAMs representing the single-robot tasks must be coordinated. To this end, I also propose a centralized framework for controlling a group of TAMs such that the behavior of the group implements the model of the complex task.

### 3.3.2 Relation to simulation

As discussed in Chapter 2, simulation is a cost-effective tool for studying robotic systems that runs the risk of over-simplifying said systems. Robot experiments, on the other hand, are costly but essential in order to be able to draw conclusions about the transferability of a given

approach between robot platforms.

The TAM is a device that operates at an intermediate level of abstraction between the task representations as used simulation and robot experiments. The TAM achieves this by mimicking the essential aspects of the task that it abstracts. As such, the TAM can be considered a task emulation similarly to hardware emulators used in integrated circuit design. Accordingly, the advantages of emulating a task through the TAM as opposed to simulating it are similar to those gained by emulating a circuit with special hardware emulators: an approach can be studied *in situ*, that is, in the environment in which it is intended to be used. In the case of the TAM, this means that tasks can be studied in laboratory experiments that use robots rather than in fully abstracted simulations.

### 3.3.3 Advantages of the TAM

The TAM has many advantages for swarm robotics research. First, the TAM facilitates research on complex tasks. To date, the complexity of tasks studied in the literature is primarily limited by two factors: the capabilities of the robots employed, and the type of *ad hoc* task abstraction used—see Section 3.2. The TAM is capable of abstracting and representing tasks of various complexity, which can be studied with relatively simple robots. The TAM therefore enables research on complex tasks that was prohibitively costly before.

Second, the TAM allows researchers to study complex tasks using relatively simple and inexpensive robots. Performing a complex task with robots as simple as the e-puck often requires adding task-specific hardware capabilities (e.g., by adding a specialized gripper). The TAM allows researchers to use generic robots for studying complex tasks, thereby reducing the cost of hardware.

Third, the TAM allows researchers to focus on the aspects that are relevant to a study when designing the software that governs the behavior of the robots. Hence, the TAM enables the development of strategies that are generic, portable and reusable, as it is not encumbered by task-specific details.

Fourth, the TAM provides infrastructure essential for conducting experiments with large swarms. The TAM helps researchers to record, analyze and correlate data of real-robot experiments; activities that are essential to conducting rigorous experiments. This becomes especially valuable when conducting experiments involving large swarms:

while a few robots are easily handled, the cost of handling large swarms becomes quickly prohibitive with an increasing number of robots (Carlson et al., 2004). In order to alleviate this problem, one has to provide automated infrastructure that supports the researcher in conducting experiments (McLurkin et al., 2006).

Fifth, the TAM provides a general and unified approach to task abstraction, which alleviates the problem of reproducibility in robotic experiments.

In summary, the TAM allows researchers to avoid aspects of a study that are:

- very expensive in terms of hardware,
- time consuming to perform,
- problem specific and thus not transferable,
- outside the goals of swarm robotics research.

Note that none of these advantages are exclusive to the TAM. For example, one could devise problem-specific *ad hoc* abstractions representing complex tasks (e.g., Ijspeert et al., 2001), build or extend robots so that they possess the required capabilities (Magnenat et al., 2012), use software development strategies for separating generic behaviors from task-specific ones (e.g., Brooks, 1986), or employ a swarm of graduate students to track every robot's movements. However, all of these solutions incur a cost, often to the point where cost becomes the decisive factor in the design of a study. Furthermore, none of these solutions are generic in the sense that they cannot be easily applied to different problems.

As a result, all of the advantages discussed above can be conceived as a reduction in cost. This leads to the following succinct observation of the TAM's advantages:

> *The advantage of the TAM is cost reduction.*

While this observation is rather humble, one has to realize that cost, monetary or other, is the main limitation when studying artificial systems in general and swarm robotics systems in particular.

### 3.3.4 Limitations of the TAM

As previously mentioned, the TAM is limited to stationary tasks, that is, tasks that concern objects with a fixed location in the environment. The TAM is therefore of little use for tasks that evolve in space, for example, spatially organizing behaviors or navigation behaviors.

Even though the TAM is limited to stationary tasks, object transportation tasks can be represented using the TAM as long as transport occurs from one fixed location to another fixed location. An example of this type of task is a typical foraging scenario where objects are harvested from a source and transported to nest.[7] In such a scenario, the source and the nest can be modeled as a collection of stationary single-robot tasks represented by a group of TAMs, and parameters such as capacity of the central place can be regulated by the number of TAMs used.

Certain object transportation tasks, on the other hand, are not suited to be represented by the TAM. An example is a collective transport task in which robots implicitly communicate their goal direction by analyzing the forces that occur when physically manipulating the object to transport (see, for example, Donald et al., 1997; Baldassarre et al., 2006, 2007).

Arguably, the TAM might suffer from a discrepancy between experiment and application similarly to the reality gap discussed in Chapter 2: in case the TAM does not properly reflect the essential aspects of the problem at hand, conclusions drawn in the experiment might not be valid for the final application. However, this is the case for all abstract task representations—it remains the responsibility of the researcher to identify the essential aspects of a task and to decide if employing the TAM (or any other representation) is justifiable. Swarm robotics usually focuses on group dynamics and collective processes; as stated in Chapter 2, I believe that it is always advantageous to conduct experiments in reality and abstract from inessential aspects of the environment rather than to conduct the entire experiment in simulation.

[7] In the literature, this is often referred to as source/sink setup or central-place foraging (Orians and Pearson, 1979).

### 3.3.5 Classes of problems that benefit from the TAM

In this section, I discuss classes of problems whose study potentially benefits from the TAM. Complementary to this high-level discussion, I discuss concrete examples of tasks studied in the literature that can be represented with the TAM in Chapter 4, Section 4.5.

The TAM is a task representation that is highly abstracted; its intended purpose is to represent tasks in studies where the details of task execution are inessential to the result (see Section 3.3). Furthermore, the TAM allows researchers to conduct experiments using real robots for studies that would otherwise be confined to simulation (see Section 2.2). As a result, the primary beneficiaries of the TAM are studies that:

- focus on group dynamics and collective processes rather than the specifics of task execution,
- involve large numbers of relatively simple robots,
- consider stationary tasks,
- are difficult to conduct using the original task due to the complexity of the task and/or the limitations of the robots.

Generally speaking, studies that benefit from the TAM are all swarm robotics studies that consider stationary tasks that have to be performed by the robots. More specifically speaking, studies that consider collective decision-making behaviors such as self-organized task allocation and task partitioning benefit most from the TAM.

## 3.4 Summary

In this chapter, I discussed task abstraction and my contribution to it. After defining task abstraction and its role in swarm robotics, I reviewed abstract task representations commonly used in the literature and discussed their advantages and drawbacks. I then introduced the TAM: a special device for task representation that is sufficiently generic and flexible to represent tasks of various complexity.

The TAM lies at the core of a novel set of tools for task abstraction that are the main contribution of this dissertation. Alone, the TAM can represent single-robot stationary tasks. In order to represent more

complex tasks, a preceding abstraction-step is required in order to model said tasks as a set of interrelated single-robot stationary tasks—in the following Chapter 4, I present a novel approach to this end.

Chapter 4

# Abstracting and representing complex tasks

In this chapter, I present a novel approach to abstract complex tasks. The primary goal of the approach is to enable uniform physical representations for complex tasks. More specifically, its goal is to enable the representation of a large variety of complex tasks using the same object—the TAM. As such, the approach aims to abstract and model complex tasks as a set of interrelated single-robot subtasks, which can then be represented in an experiment with a group of TAMs.

I propose a two-level approach to abstract complex tasks. The goal of the *high-level model* is to describe the hierarchical structure of a complex task without having to consider all the details of the interrelationships between its subtasks. The high-level model is a convenient high-level description of complex tasks and serves as a basis for classifying and comparing them. The goal of the *low-level model* is to describe the details of the interrelationships between the subtasks of a complex task. The low-level model serves as a "blueprint" for the software that coordinates a group of TAMs so that their behavior replicates these interrelationships.

I use well-known visual modeling languages for both levels; more specifically, I use UML 2.x activity diagrams for the high-level (see Rumbaugh et al., 2004, for an overview) and Petri nets for the low-level (see Petri and Reisig, 2008, for an overview). UML 2.x activity diagrams are appropriate for the high-level model as they are intuitive and convenient to use (Rumbaugh et al., 2004). Petri nets are appropriate for the low-level model because they have well-defined execution

semantics that allows one to simulate them (Störrle, 2000).

This chapter is structured as follows. In Section 4.1, I discuss the state of the art in task modeling and decomposition. In Section 4.2, I present how to model complex tasks at a high level of abstraction: how to decompose tasks into a set of subtasks and how to identify the hierarchical structure of these subtasks in the form of various interrelationships. In Section 4.3, I present how to model complex tasks at a low level of abstraction in order to describe these interrelationships such that they can be represented using a group of TAMs. In Section 4.4, I present a centralized control framework for implementing the coordination software derived from the low-level model. In Section 4.5, I review the swarm robotics literature and apply the proposed modeling approach to the various tasks considered in order to demonstrate its flexibility. In Section 4.6, I provide a summary of this chapter.

## 4.1 State of the art

The modeling approach presented in this chapter is based on task decomposition, a technique frequently used to describe and model tasks (Anderson et al., 2001a; Korsah et al., 2013). Task decomposition typically entails that tasks are broken down into parts that can then be considered separately. In this section, I discuss the state of the art in modeling and classifying tasks by using task decomposition.

There are two different types of approaches that use decomposition for modeling tasks: descriptive approaches, used for classifying and describing existing systems; and prescriptive approaches, used for designing artificial systems. The distinction between those two approaches is not precise; whether an approach is considered descriptive or prescriptive depends on the context in which it is used rather than on intrinsic differences. Consequently, most approaches can be used in both ways. For example, consider the approach presented in this chapter: I construct abstract models of complex tasks (see Section 4.2 to 4.4) and describe, classify and compare complex tasks found in the literature (see Section 4.5) using the same approach. In the following, I will first discuss descriptive approaches, then prescriptive approaches.

Several descriptive approaches use taxonomies based on task decomposition to classify tasks, for example, by complexity. Anderson et al. (2001a) presented a taxonomy for the classification of tasks found in

colonies of social insects such as ants. In their taxonomy, the authors describe tasks using a hierarchical structure of nested tasks that can be of four types: individual, group, partitioned and team tasks. The authors assign a numeric value to each of these task types; the sum of the values of all subtasks is used to measure the complexity of the original task. The authors use the proposed metric to classify studies of insect societies.

In the domain of robotics, Gerkey and Matarić (2004) proposed a taxonomy to classify tasks of multi-robot task allocation problems (MRTA). In their work, the authors classify tasks along three axes: task type (single-robot versus multi-robot), robot capabilities (single-task versus multi-task), and allocation type (time-extended versus instantaneous). The work is limited to what I refer to as *orchestrated* approaches, that is, approaches that rely heavily on communication to negotiate allocations (e.g., market- and auction-based approaches such as those proposed by Goldberg et al., 2003; Kalra and Martinoli, 2006; Dias et al., 2006).

An extension of the work of Gerkey and Matarić has recently been proposed by Korsah et al. (2013). In their work, Korsah et al. present a two-level taxonomy: the first level is used to classify tasks by their interrelationships, and the second level is used to classify tasks as in the taxonomy proposed by Gerkey and Matarić (2004). On the first level, Korsah et al. use task decomposition to identify tasks and their interrelationships, called constraints. Task decomposition is used in a purely descriptive way—contrarily to the work presented in this dissertation, the proposed approach does not yield models that can be simulated, formally analyzed, or implemented in the form of abstract task representations. Analogously to the work proposed by Gerkey and Matarić (2004), the work focuses exclusively on orchestrated approaches to MRTA problems.

In terms of prescriptive approaches, task decomposition is commonly used as a strategy to partition work, following the principle of *divide et impera*:[8] a complex task can be addressed in a more efficient way if it is decomposed into smaller units of work (Jeanne, 1986; Pini, 2013). See Chapter 7, Section 7.3 for a study on task partitioning using the TAM.

In the following, I discuss some exemplary descriptive approaches used for the design of artificial systems. Parallel computing, for ex-

[8] lat., divide and rule

ample, commonly uses task decomposition to distribute tasks between multiple processors (Grama et al., 2002). More precisely, a program can be divided into smaller units that are executed in parallel on different processors of the same system (Cormen et al., 2001). Similarly, an operating system capable of multi-tasking executes several processes in a pseudo-concurrent manner by allocating them to a processor for short periods of time (Tanenbaum, 2007). Task decomposition is also the basis of the common programming pattern of recursive algorithms: a recursive algorithm decomposes a problem into smaller units and executes itself for each of these units, thereby possible decomposing the problem further (Knuth, 1997).

In robotics, there are few works that explicitly use prescriptive approaches—most works are limited to describing and classifying tasks, rather than employing task decomposition for designing systems (for an example, see Korsah, 2011). Zlot (2006) is one of the few that considers task decomposition as an integral part of his approach to planning for multi-robot teams. In the work, the author contends that considering task decomposition concurrently with task allocation can result in more efficient solutions.

## 4.2 High-level model

In order to model the hierarchical structure of a complex task, I employ task decomposition. More specifically, I propose to decompose complex tasks into their constituent subtasks and their interrelationships. In other words, the proposed approach to model tasks is a hierarchical-deconstructionist approach similar to the approach presented by Anderson et al. (2001b). I describe the resulting hierarchical structure visually using UML 2.x activity diagrams.

In Section 4.2.1, I briefly introduce UML 2.x activity diagrams. In Section 4.2.2, I define the basic concepts that are used by the high-level model: the definition of tasks and subtasks as well as different types of interrelationships between tasks. In Section 4.2.3, I discuss the process of task decomposition used to model a complex task in detail. Lastly, I discuss the advantages and limitations of the high-level model in Section 4.2.4.

### 4.2.1 UML 2.x activity diagrams

The Unified Modeling Language (UML) is a graphical general-purpose modeling language (OMG, 2011; Rumbaugh et al., 2004). UML can be used to visually describe process flows. Initially intended for object-oriented software engineering, the language has been extended to various other domains such as business and data modeling. UML defines 14 different diagram types; the current major version of the language is 2.x (OMG, 2011).

In this dissertation, I use the UML 2.x diagram type *activity diagrams* to visually model tasks. Activity diagrams represent a view on the elementary *actions* of a given workflow. An elementary action is a single step within the workflow; it is visually represented using ovals. Elementary actions can be combined to *activities*, which define interrelationships between these actions. Activities are visually represented using rectangles with rounded corners. Arrows represent the work flow between the actions of a given activity. Activity diagrams support concurrency: actions that must be performed concurrently are enclosed by two black bars. Note that the specification of UML 2.x activity diagrams includes further elements such as conditionals or decisions (OMG, 2011). As the high-level model presented in this section does not support conditionals, I do not use these elements.

The semantic of UML 2.x activity diagrams is loosely based on Petri nets; as a result, UML 2.x activity diagrams can intuitively be transformed into Petri nets (Spiteri Staines, 2010). Contrarily to Petri nets, activity diagrams cannot be simulated and analyzed formally.[9] Note that the semantics of UML 1.x activity diagrams differ significantly from those of UML 2.x activity diagrams; I therefore do not consider UML 1.x activity diagrams in this dissertation.

The main reason for using UML 2.x activity diagrams for the high-level model is convenience: activity diagrams can be understood intuitively and are therefore easy to handle. Additionally, a vast body of tools and resources exists due to the fact that activity diagrams are commonly used in various engineering disciplines.

[9] It should be noted that, although the semantics of activity diagrams is loosely based on Petri nets, activity diagrams are unsuitable for simulation because "the rules for activity execution are not clearly explained and defined in the UML specification" (Spiteri Staines, 2010).

### 4.2.2 Definitions

In the following, I define the basic concepts used by the high-level model. Most of these definitions extend on the definitions brought forward in Chapter 3, most notably the concept of complex tasks. I will rely on the following definitions throughout the rest of the dissertation.

#### Tasks and subtasks

As introduced in Chapter 3, a *task* is a unit of work that has to be performed by the swarm. Furthermore, a *complex task* can be decomposed into several subtasks. A *subtask* concerns therefore a fraction of the work of its complex *supertask*. I use UML 2.x *activities* to model complex tasks visually—see Figure 4.1b and 4.1c for an example UML 2.x activity diagram of a complex task.

Tasks that cannot be decomposed are called *atomic tasks*. Atomic tasks are single-robot tasks, that is, tasks that require only a single robot for their completion and allow only a single robot to work on them at a given moment in time. Accordingly, stationary single robot tasks as introduced in Chapter 3 can be modeled as atomic tasks. For example, the task of deactivating an alarm in the environment is an atomic task, if this alarm can be deactivated by a single robot. I use UML 2.x *actions* to model atomic tasks visually. Figure 4.1a shows the UML 2.x activity diagram of an atomic task.

I consider a complex task to be completed once all of its constituent subtasks have been completed. Hence, I re-define a *subtask* as a unit of work that contributes to its complex supertask upon its completion. Note that a subtask can either be an atomic task or a complex task, that is, a subtask of a task can potentially consist of subtasks as well. Therefore, I use the letter $\tau$ to reference tasks and subtasks alike.

As mentioned in Chapter 3, the stationary atomic subtasks of a complex task might be contiguous in space or might take place at different locations. In the latter case, robots might need to travel from one location to the other in order to carry out the complex task; travel between tasks is not modeled explicitly. Hence, a complex task that is composed of stationary atomic subtasks could be non-stationary.

Lastly, I define a *task instance* as a specific realization of a given task. Commonly, the swarm faces many instances of the same type of task, for example, several objects scattered throughout the environment represent each an instance of the same complex task.

Note that the subtasks of a complex task can either be addressed by

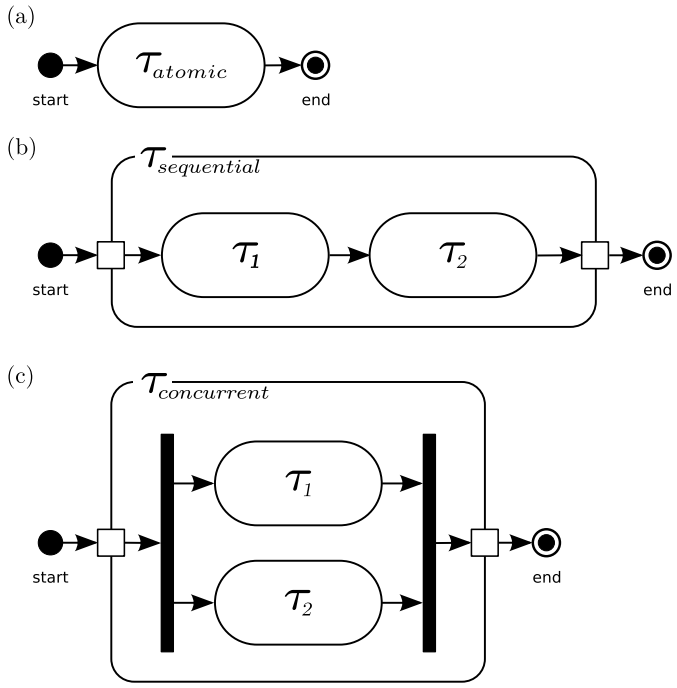



Figure 4.1: High-level models of the basic task types, expressed using UML 2.x activity diagrams. a) An atomic task does not consist of subtasks, b) A complex task that consists of two subtasks with a sequential interrelationship. c) A complex task that consists of two subtasks with a concurrent interrelationship, Atomic tasks are modeled as UML actions; complex tasks are modeled as UML activities.

separate robots or by the same robot—the model imposes no restriction on the robot-to-subtask mapping. For example, assume a complex foraging task whose subtasks are harvesting an object and storing it in the nest. Each of these subtasks could either be performed by the same robot or by a group of collaborating robots.[10]

### Interrelationships

Subtasks of a complex task have *interrelationships* with other subtasks. The hierarchical structure formed by these interrelationships defines the flow of execution of the subtasks. A task that is not a subtask

[10] In the second case, another subtask of transferring the harvested objects between robots would be required.

of a complex task does not have interrelationships with other tasks or subtasks. In the context of this dissertation, I distinguish between two types: *sequential* interrelationships and *concurrent* interrelationships.

A *sequential* interrelationship between subtasks requires that these subtasks are executed in a given order. An example is the "stick-pulling" experiment presented by Ijspeert et al. (2001). In the experiment, robots face a task that consists of two subtasks: the first subtask is to pull a stick from a hole in the ground to approximately half its length. The second subtask is to continue the pulling motion so that the stick is fully extracted from the hole. The subtasks have a sequential interrelationship: the first must be completed before the second can start, and both subtasks have to be completed successfully in order to complete the original task. Figure 4.1b shows the UML 2.x activity diagram of a complex task whose two subtasks have a sequential interrelationship.

A *concurrent* interrelationship between subtasks requires that these subtasks are executed at the same time. An example is an area coverage task as presented by Berman et al. (2009). In the work, the task of the robots is to occupy pre-defined spatially distributed positions in the environment. In order to complete this task, the robots need to occupy the positions at a given moment in time for a given duration. Figure 4.1c shows the UML 2.x activity diagram of a complex task whose two subtasks have a concurrent interrelationship.

### 4.2.3 Task decomposition

As previously mentioned, task decomposition refers to the process of subdividing a task into several units of work which can be subsequently tackled separately. Task decomposition is commonly found in the domains of, among others, artificial intelligence (Durfee and Lesser, 1989) and robotics (Korsah et al., 2013). Task decomposition in robotics is commonly studied in the context of autonomous decomposition strategies for a swarm of robots (e.g., Lein and Vaughan, 2008; Parker and Zhang, 2010; Pini et al., 2014). In this dissertation, I employ task decomposition solely to model complex tasks; as such the act of decomposing a task is part of the experimental design by the researcher rather than a strategy for the robots.

The result of decomposing a complex task is a model consisting of a set of subtasks and the interrelations between them. In the resulting model, the complexity of the original task resides in these interrelation-

ships. The interrelationships are defined by the hierarchical structure of the subtasks, as identified by task decomposition. Decomposition is recursive, that is, subtasks can potentially be decomposed further; a subtask of a task can therefore consist of subtasks as well. Decomposition stops once all decomposable subtasks have been decomposed.

I call the hierarchical structure formed by the subtasks of a complex task its *task relationship graph*. A task relationship graph is a directed acyclic graph: there is a direction in which the graph has to be traversed in order to execute the original task. The nodes of the task relationship graph of a given complex task represent its subtasks. Each of these nodes may be a task relationship graph in itself, that is, the graph may be nested due to the recursive application of task decomposition as explained above.

I refer to the complex task at the top of the task relationship graph as the *overall* task; all nodes of the task relationship graph are subtasks of this overall task. I refer to the maximum number of nested complex tasks as the *nestedness* of a task relationship graph. Furthermore, I refer to the number of recursive decomposition steps required to reach a particular node as its *depth*. In other words, the depth of a task indicates the number of complex task it is nested in. The overall task, lacking a supertask, has a depth of zero. By definition, the nodes with the highest depth of a given task relationship graph are atomic tasks, as any complex task has subtasks with a higher depth. Consequently, the nestedness of a task relationship graph equals the highest depth among its nodes. Note that the concept of depth as used in the context of this dissertation differs considerably from the concept of depth and level as used in non-nested tree data structures (Knuth, 1997)

The simplest task relationship graph is the graph of an atomic task, which has a nestedness of 0 (see Figure 4.1a). Complex tasks that consist only of atomic subtasks have a nestedness of 1 (see Figure 4.1b and 4.1c). As an example, Figure 4.2 shows the high-level model of a complex task with a nestedness of 3.

Note that there might be several ways to decompose a given complex task, that is, a given decomposition is to a certain extent arbitrary. It is the researcher's obligation to pick a decomposition that suits the application.

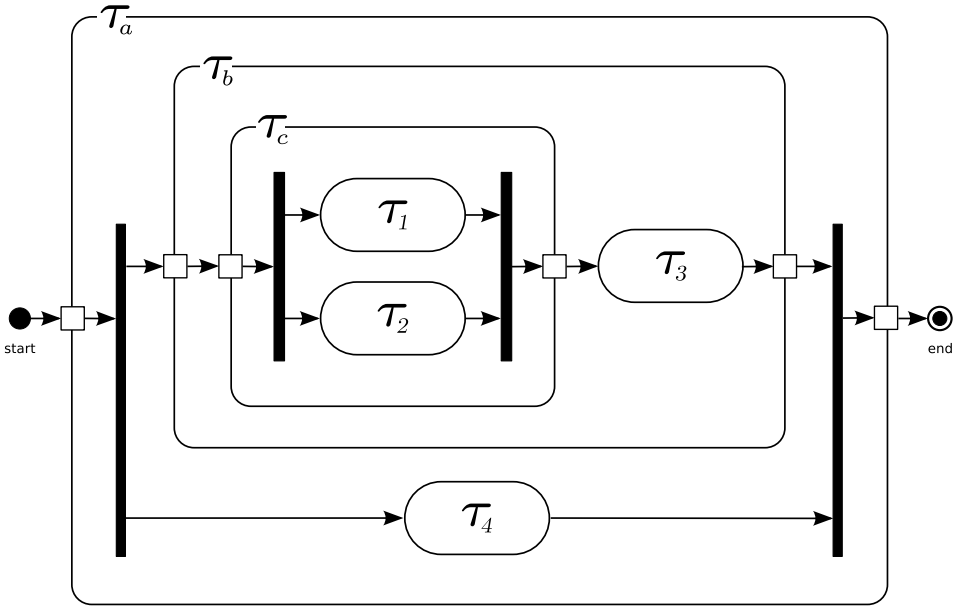



Figure 4.2: High-level model of a complex task with a nestedness of 3. The overall task $\tau_a$ consists of two subtasks, $\tau_b$ and $\tau_4$, that have a concurrent interrelationship. $\tau_b$ has two subtasks, $\tau_c$ and $\tau_3$, that have a sequential interrelationship. $\tau_c$ has two subtasks as well, $\tau_1$ and $\tau_2$, that have a concurrent interrelationship. Tasks indexed with a letter are complex tasks, tasks indexed with a number are atomic tasks.

### 4.2.4 Discussion: Advantages and Limitations

Recursive task decomposition used in conjunction with the described types of task interrelationships yields a powerful yet simple approach for modelling various complex tasks. The distinction between sequential and concurrent interrelationships is commonly used for the decomposition of tasks, for example in parallel computing (Grama et al., 2002). I am therefore confident that the proposed high-level model allows me to describe the majority of tasks that occur in real-world scenarios and that are of interest in swarm robotics research. More specifically, the model is capable of describing all task types identified in the seminal classification of task types and complexities presented by Anderson et al. (2001a). Considering the domain of multi-robot task allocation, the model can be used to describe all classes of task allocation problems covered by the taxonomies proposed by Gerkey and Matarić (2004) and Korsah et al. (2013).

Nevertheless, the presented approach has certain limitations. For example, conditional execution of tasks is not covered by the high-level model as presented. Note that this limitation is imposed by me (rather than being a restriction of UML 2.x) in order to be able to quickly and conveniently model complex tasks. However, the choice of UML 2.x as a modeling language does restrict the utility of the model: it can neither be simulated, nor formally analyzed.

More importantly, the high-level model as presented abstracts from the details of specific task interrelationships. Again, this is a restriction imposed by me for the above-mentioned reason. For example, consider subtasks with a sequential interrelationship: the interrelationship might either be blocking (e.g., Ijspeert et al., 2001) or non-blocking (e.g. Pini et al., 2014), but the high-level model does not differentiate between those two cases. Other examples of interrelationships not captured by the high-level model are subtasks that must be executed by the same robot or subtasks that have to start and/or stop at the same moment in time.

In order to model such aspects of task interrelationships, I propose the Petri net-based low-level model.

## 4.3 Low-level model

The high-level model is convenient to use, but cannot capture all the details of task interrelationships. In order to properly model these details, I propose an approach based on Petri nets.

In the following, I first give a summary of Petri nets in Section 4.3.1. I then present an approach for creating detailed low-level, Petri net-based models of a given high-level model. The approach is a "bottom-up" approach: one starts from the bottom of the nested task relationship graph, that is, with the "deepest" atomic tasks with the maximum depth, iteratively adding interrelationships of complex supertasks and atomic tasks at each depth until reaching depth zero of the graph. In order to explain this "bottom-up" approach, I present in the following models for tasks with increasingly complex task relationship graphs.

In Section 4.3.2, I present a basic low-level model for atomic tasks. In Section 4.3.3, I extend this basic model and use it as a basis to model complex tasks with a nestedness of 1. I present detailed models of atomic tasks with variations of the interrelationships identified in Section 4.2, namely sequential and concurrent interrelationships.

In Section 4.3.4, I generalize these models for tasks whose relationship graph has an arbitrary nestedness. In Section 4.3.5, I propose a generic model for atomic tasks that is based on the results presented in the previous sections. This model serves as a basis for the firmware of the TAM, that is, for the software that controls the behavior of a single TAM—see Section 4.4. In Section 4.3.6, I consider two examples that require modelling of not only the tasks themselves but also of the interaction between these tasks and the swarm. I demonstrate how the low-level model can be extended to describe and analyze these examples. Finally, in Section 4.3.7, I discuss the advantages and limitations of the low-level model.

### 4.3.1 Petri nets

I use Petri nets to describe the subtasks of complex tasks and their interrelationships. In the following, I will give a summary of Petri nets and the specific variant used in this dissertation. For more information, I refer the reader to Petri and Reisig (2008) as well as Reisig (2013).

Petri nets are a mathematical modeling language for the description of discrete distributed systems (Petri and Reisig, 2008). Petri nets offer, just as UML 2.x activity diagrams, a graphical notation for stepwise processes that include sequential and concurrent execution (Störrle, 2000)—in fact, Petri nets inspired the semantic of UML activity diagrams when redesigned for version 2.x. Contrary to UML 2.x activity diagrams, the flow of execution in a Petri net can be simulated and analyzed, which allows researchers to test the model before implementing it (see Section 4.2.1 for a brief discussion of the simulation of UML 2.x activity diagrams).

A Petri net is a directed bipartite graph whose nodes are either *transitions* or *places*. Commonly, transitions represent events and places represent conditions for these events to occur. Visually, transitions are represented with rectangles and places are represented with circles. Directed *arcs* connect pre- and post-conditions to a given transition; arcs never connect a place to a place or a transition to a transition.

Places may contain *tokens*. A mapping of tokens to places is called *marking* of the net. Tokens are modified by transitions: when a transition occurs, which is called *firing*, it consumes tokens from its preconditions and produces tokens in its post-condition. A transition can only fire when its pre-condition contains sufficient tokens to consume. Transition that can fire are *enabled*; firing is nondeterministic, that

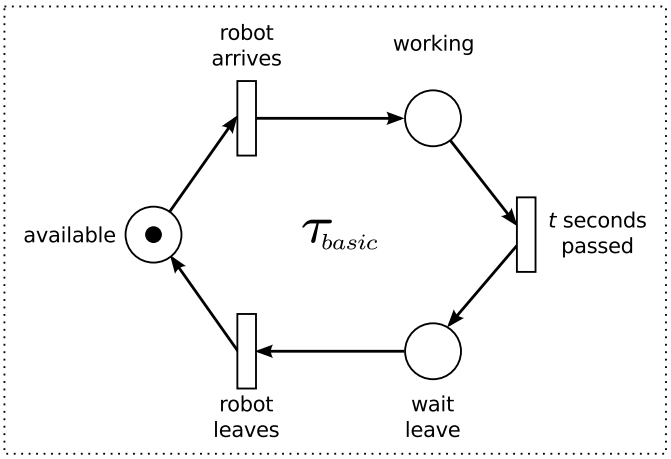



Figure 4.3: Basic low-level Petri net model of an atomic task $\tau_{basic}$ with three states. The state of the task changes depending on external events.

is, if multiple transitions are enabled at the same time, any may fire. The amount of tokens produced or consumed is signified by the *weight* of the connecting arc; weights omitted in the description of a net are assumed to be 1.

I use a variant of Petri nets that is *bounded*: contrarily to the base version of Petri nets, the places of a bounded Petri net are limited in the number of tokens they can hold. This limit is called the *capacity* of a place. If the capacity of a place is reached, a transition that would produce a token in this place cannot fire. Similarly to the weight of arcs, capacities omitted in the description of the net are assumed to be 1.

### 4.3.2 Atomic tasks

I model an atomic task using a type of bounded Petri net called *state machine*. In a state machine, every transition has exactly one pre-condition and one post-condition. Furthermore, all markings only have a single token; there is always a single place that defines the state of the net. Consequently, the state of the resulting Petri net is not distributed; it cannot model concurrency. I refer to a given place of the model as *state*. Modeling an atomic task as a state machine has the distinct advantage that the model can be directly transferred to a TAM—see Section 4.4.

Figure 4.3 shows the basic low-level model of an atomic task, which consists of three states:

1. in the available state, the task is available and a robot can approach and enter into it;
2. in the working state, the task is busy as a robot is currently working on it;
3. in the wait leave state, the task has been completed and the robot has to leave.

An atomic task transitions between its states depending on external events, for example, a robot arrives to work on the task or $t$ seconds have passed since the robot started to work on it.

The model shown in Figure 4.3 is *basic* insofar that there are only few states; other states can be added as needed. For example, a task could become available only some time after a robot has left—which could be modeled by adding a new state between wait leave and available. In the next section, I will show how to model interrelationships between tasks by extending this basic model.

### 4.3.3 Complex tasks consisting only of atomic subtasks

Following the high-level model presented in Section 4.2, I model a complex task as a set of subtasks with interrelationships. In this section, I consider only complex tasks with nestedness 1, that is, tasks that consist exclusively of atomic subtasks. I model the interrelationships between these atomic tasks by adding places between their transitions. The places are additional conditions to the state transitions of a given subtask; I therefore refer to them as *conditions*.

Adding conditions to the state transitions of an atomic task implies that more than one token can be in a given marking. This effectively makes the state machines of the atomic tasks part of a bigger, unrestricted Petri net which has a distributed state and can model concurrency.

By keeping the atomic tasks as distinct *subnets* of the overall Petri net, one can easily distinguish between atomic subtasks and their interrelationships. Furthermore, this allows one to maintain the state machines for the atomic tasks, which helps implement the model to

control a group of TAMs. Note that I extend the basic model of the atomic task presented above with additional states in order model the interrelationships between subtasks.

Without any loss of generality, I assume two atomic subtasks for all following complex tasks. Hence, the marking of the low-level model of a complex task with two atomic subtasks has at least two tokens, one for the state of each atomic task. Additional tokens in the marking are used for modeling the specific interrelationship between the subtasks. In the following, I refer to the two subtasks as $\tau_1$ and $\tau_2$, respectively.

### Sequential interrelationships

A sequential interrelationship between tasks indicates that these tasks have to be executed in sequence—see Section 4.2.2. The high-level model for complex tasks with this type of interrelationship is shown in Figure 4.1b. The model does not capture the various details of sequential interrelationships commonly found in the literature. In the following, I model two common varieties of sequential interrelationships, namely non-blocking and blocking ones.

**Non-blocking:** The simplest form of sequential interrelationship is non-blocking. Robots working on tasks with a non-blocking sequential interrelationship do not have to wait for each other. More specifically, if $\tau_1$ and $\tau_2$ have a non-blocking sequential interrelationship, a robot that finished working on $\tau_1$ can leave after completing its task without having to wait for another robot to start working on $\tau_2$.

In order to model this form of interrelationship, I extend the basic model of the atomic task with a new state called wait available. The task $\tau_2$ has to remain in this state until new work is available, modeled by a new condition work available. Figure 4.4 shows the resulting model.

Examples of non-blocking sequential interrelationships are foraging tasks where robots can deposit objects on the ground instead of handing them over directly. As a result, multiple executions of $\tau_1$ are possible before an execution of $\tau_2$ is mandatory. Object deposits can either happen at a dedicated cache site (Pini et al., 2011b, 2013b) or distributed throughout the environment (Pini et al., 2011b, 2013b; Shell and Matarić, 2006; Lein and Vaughan, 2008). In such a scenario, the condition work available effectively represents the cache site. Accordingly, the condition can have a capacity higher than one in order to model the size of the cache site.

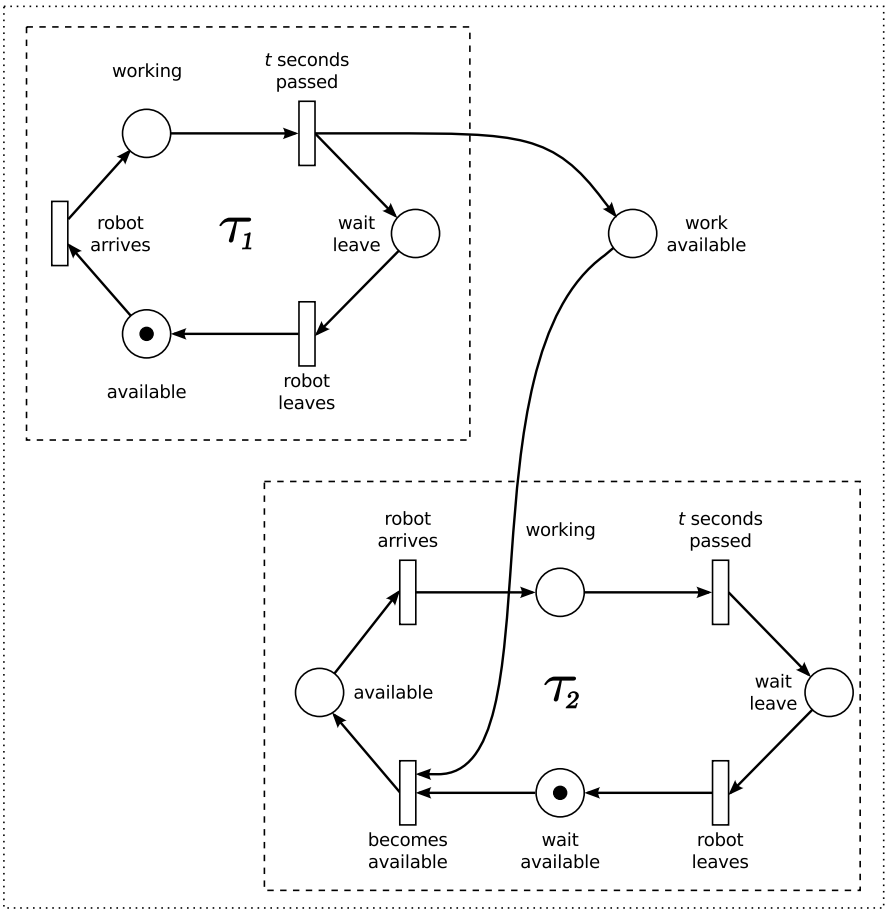



Figure 4.4: Low-level model of a complex task with a non-blocking sequential interrelationship between its atomic subtasks $\tau_1$ and $\tau_2$. Note that the model of the subtasks have been extended differently with respect to the basic model of atomic tasks. The additional condition work available represents the non-blocking sequential interrelationship that links the subtasks. Dashed lines designate the subnets that model subtasks.

**Blocking:** Blocking sequential interrelationships are more complex than non-blocking ones and thus require a more complex model. Robots working on tasks with a blocking sequential interrelationships have to wait for each other. More specifically, if $\tau_1$ and $\tau_2$ have a blocking sequential interrelationship, a robot that finished working on $\tau_1$ must wait after completing its task until another robot starts to work on $\tau_2$.

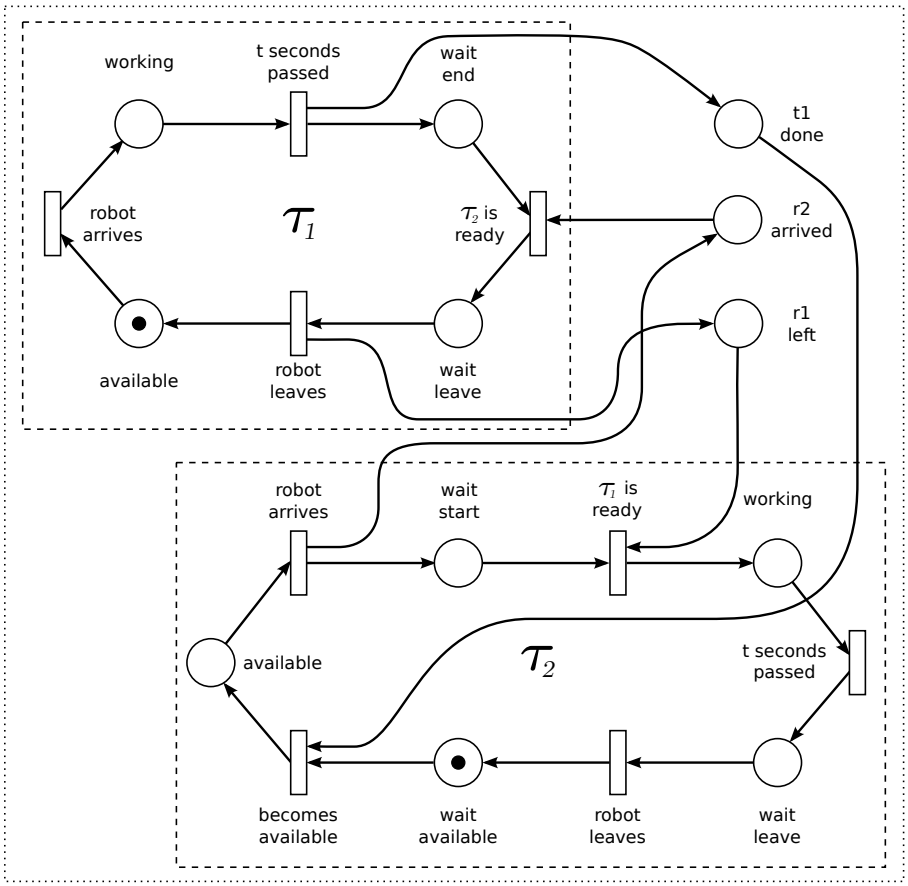



Figure 4.5: Low-level model of a complex task with a blocking sequential interrelationship between its atomic subtasks $\tau_1$ and $\tau_2$. Note that the model of the subtasks have been extended differently with respect to the basic model of atomic tasks. The additional conditions t1 done, r2 arrived, and r1 left model the blocking sequential interrelationship that links the two atomic subtasks. Dashed lines designate the subnets that model subtasks.

In order to model blocking sequential interrelationships between two atomic tasks, I extend the basic model of the atomic task differently for $\tau_1$ and $\tau_2$. I add a new state called wait end to $\tau_1$, in which the task remains until $\tau_2$ is ready. Regarding $\tau_2$, I add a state called wait start, in which the task remains until $\tau_1$ is ready. Furthermore, I add the following conditions that link the two atomic subtasks:

1. t1 done, which is enabled after $\tau_1$ has been completed;

2. r2 arrived, which is enabled after a robot arrived to work on $\tau_2$;

3. r1 left, which is enabled after the robot that completed $\tau_1$ has left.

Note that condition r1 left is optional and can be omitted if desired. Figure 4.5 shows the resulting model.

A possible extension of this model is a condition that allows $\tau_1$ to become available anew only after the robot in $\tau_2$ has left. This condition effectively removes any concurrency between two executions of the complex task. More specifically, a new execution of the overall task cannot start before the previous one finished completely.

An example of a blocking sequential interrelationship is a foraging task where robots have to hand over objects without depositing them on the ground (Brutschy et al., 2014c).[11] Another example of a blocking sequential interrelationship is the stick-pulling experiment presented by Ijspeert et al. (2001).

### Concurrent interrelationships

A concurrent interrelationship between tasks indicates that these tasks have to be executed at the same time—see Section 4.2.2. The high-level model for complex tasks with this type of interrelationship is shown in Figure 4.1c. In the following, I model an exemplary concurrent interrelationship in detail: the interrelationship is such that subtasks are synchronized, that is, work must start on both subtasks concurrently. Furthermore, neither subtask can become available anew before all robots have left.

Again, I extend the basic model of atomic subtasks to allow the tasks to synchronize—in this case, I extend the basic model in the same way for both subtasks. In the state wait start, subtasks require robots to wait before starting to work on a task. In the state wait available, subtasks wait until both robots have left before becoming available anew. Additionally, I add four conditions to implement the interrelationships described above:

1. r1 arrived, which is enabled after a robot arrived to work on $\tau_1$;

2. r2 arrived, which is enabled after a robot arrived to work on $\tau_2$;

[11] Blocking handovers are sometimes also described as “direct transfer”, or, if handover coincides with a physical location, as “direct interface” (Pini, 2013).

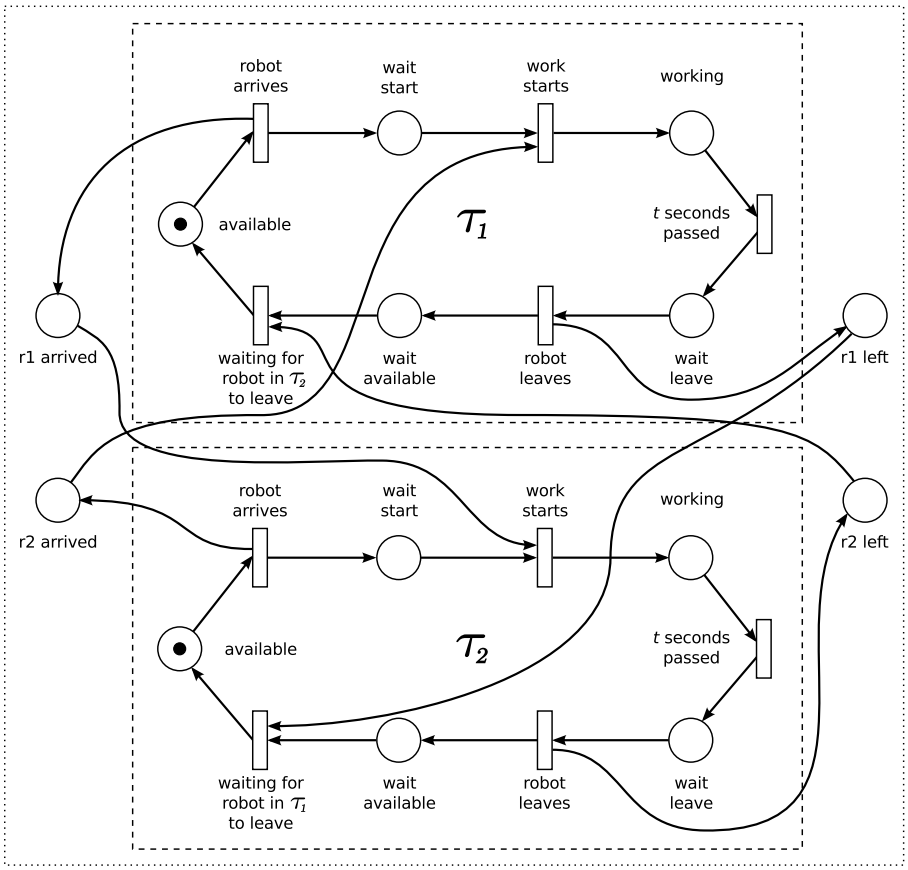



Figure 4.6: Low-level model of a complex task with a concurrent interrelationship among its atomic subtasks $\tau_1$ and $\tau_2$. Note that the model of the subtasks has been extended with respect to the basic model. Dashed lines designate the subnets that model subtasks.

3. r1 left, which is enabled after the robot that completed $\tau_1$ has left;

4. r2 left, which is enabled after the robot that completed $\tau_2$ has left.

Figure 4.6 shows the resulting model.

### 4.3.4 Nested complex tasks

In this section, I demonstrate how to model complex tasks that consist of subtasks that are themselves complex tasks, that is, the overall task

consists of nested complex subtasks. Accordingly, the nestedness of the associated task relationship graph is higher than 1. Note that, as tasks can be arbitrarily nested, the high-level model of such a task can have various shapes.

In general, in order to model a nested complex task one has to follow a "bottom-up" approach starting from the "deepest", most nested tasks of a given task relationship graph identified by the high-level model. More specifically, one starts modeling the atomic tasks at the maximum depth and adds the complex supertask at the next-higher level by modeling the necessary interrelationships. This process is repeated until one reaches the overall supertask at depth zero.

As an example, I model the nested complex task whose high-level model is shown in Figure 4.2. The resulting low-level model is shown in Figure 4.7. In the figure, tasks indexed with a letter are complex whereas tasks indexed with a number are atomic. Arcs denoted using Latin letters model concurrent interrelationships whereas arcs denoted using Greek letters model sequential interrelationships.

The two atomic tasks $\tau_1$ and $\tau_2$ are subtasks of the complex task $\tau_c$ and have a concurrent interrelationship. This interrelationship is modeled as discussed in Section 4.3.3: work on the $\tau_1$ and $\tau_2$ can only start when both robots are present and each robot can leave only after both robots completed their work. In Figure 4.7, I denote the arcs modeling this interrelationship as follows: $a, a'$ = robots arrived to work on $\tau_1$ and $\tau_2$, respectively; $b, b'$ = robots of $\tau_1$ and $\tau_2$ have left, respectively.

The complex task $\tau_c$ and the atomic task $\tau_3$ have a sequential interrelationship, which forms the complex task $\tau_b$. For simplicity, I assume a non-blocking interrelationship, that is, the robots that completed $\tau_c$ do not have to wait for a robot to arrive for $\tau_3$. This interrelationship can be modeled by adding a single condition $\alpha$ that is enabled if $\tau_c$ is completed, that is, the tasks $\tau_1$ and $\tau_2$ are both completed. Note that $\alpha$ has a weight of 2:1 in order to fully consume the output of the concurrent task.

Finally, $\tau_b$ and the atomic subtask $\tau_4$ form the overall task $\tau_a$ by having a concurrent interrelationship. Accordingly, $\tau_b$ and $\tau_4$ cannot start before they are both ready (condition denoted with $c$ and $c'$ in Figure 4.7, 2:1). Furthermore, $\tau_b$ and $\tau_4$ cannot become available anew before the previous execution of $\tau_a$ has been completed by completing $\tau_3$ (condition denoted with $d$ in Figure 4.7, 3:1).

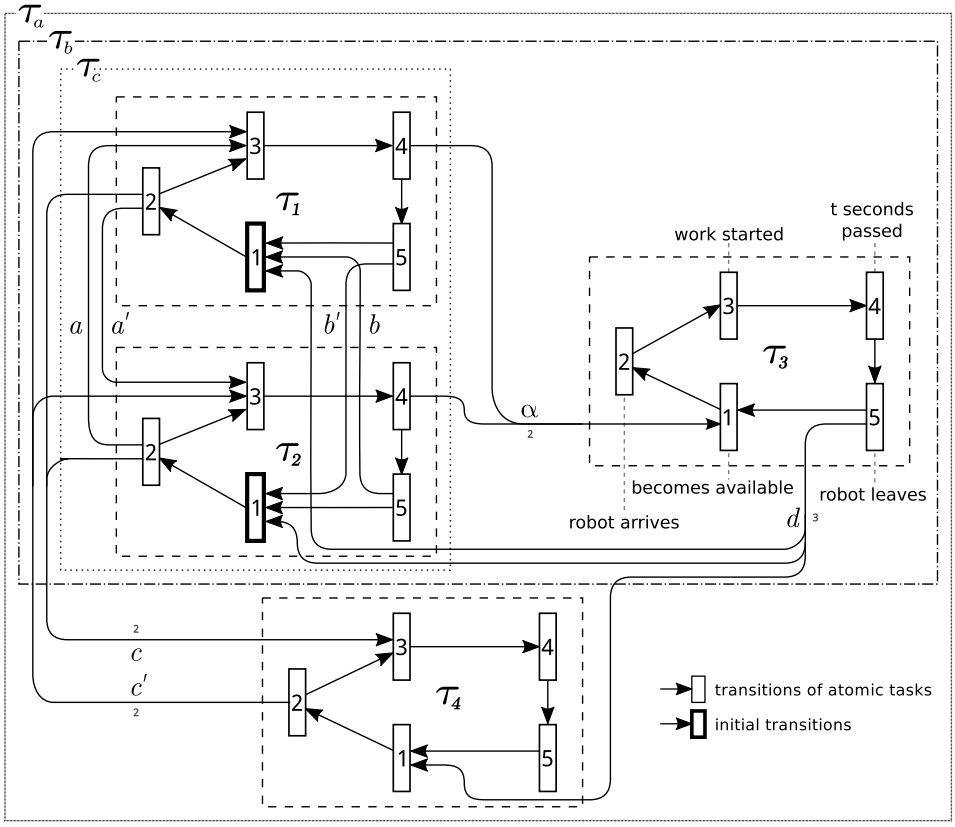



Figure 4.7: Low-level model of a nested complex task with a nestedness of 3 (reduced version without places).[11] This model is the low-level counterpart of the high-level model shown in Figure 4.2. Transitions enabled at the start of the experiment are marked with a bold border. Arcs denoted using Latin letters model concurrent interrelationships and arcs denoted using Greek letters model sequential interrelationships. Tasks indexed with a letter are complex, tasks indexed with a number are atomic. Dotted/dashed lines indicate task boundaries.

Note that Figure 4.7 shows the reduced version of this Petri net.[11] As the initial marking cannot be visualized in the reduced version, I highlight transitions enabled at the start of the experiment with a bold border.

[11] By convention, the places of a Petri net can be omitted in order to better visualize the structure of the net (Petri and Reisig, 2008).

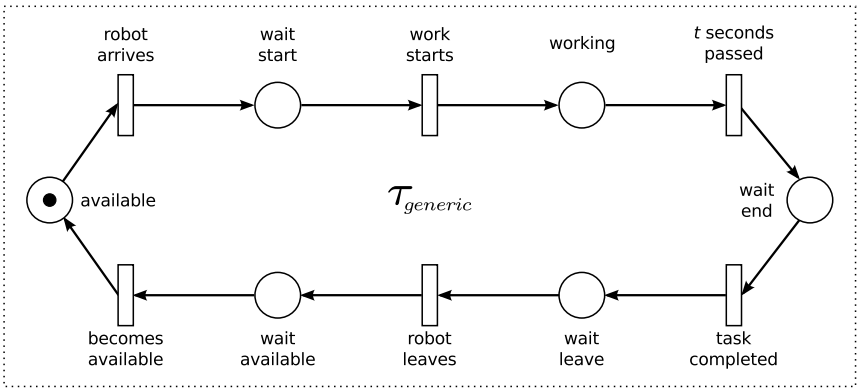



Figure 4.8: Generic low-level model for atomic tasks with six states. This model is an extension of the basic model presented in Section 4.3.2. It incorporates all additional states required to model the interrelationships discussed in Sections 4.3.3 and 4.3.4.

### 4.3.5 A generic model of atomic tasks

In the previous sections, I extended the basic model for atomic tasks several times in order to model the various interrelationships. In the following, I propose a generic model for atomic tasks that incorporates all these extensions—the resulting model is *generic* in the sense that it can be used to model all the interrelationships discussed in Sections 4.3.3 and 4.3.4. Figure 4.8 shows the generic low-level model. Compared with the basic model, the generic model possesses three additional states:

1. in the wait start state, the task has to wait for the start or completion of an interrelated task before its robot can begin to work;
2. in the wait end state, the task has been completed but the robot has to wait before it can leave;
3. in the wait available state, the task has to wait before it can become available anew.

All three states can be used to implement the necessary conditions to model interrelationships with other tasks. As the generic model is a superset of all atomic task models discussed in the previous sections, it can be used without modification when modeling any complex task. Note that this might result in superfluous states through which the task will transition immediately. The generic model serves as basis for

a generic software component that controls the behavior of a single TAMs. Given this software, a researcher has to implement only the necessary interrelationships by adding conditions between the aforementioned states of this generic software, which greatly simplifies the implementation of a complex task's control software—see Section 4.4.

### 4.3.6 Extensions

The low-level model can be extended in order to model various other processes or aspects. The resulting models can be used for simulating and analyzing more complex interactions such as deadlocks or resource contention. However, if these models incorporate other entities than the TAM, they might not be transferable to the physical device.

In the following, I present two example tasks that require modelling of not only the tasks themselves, but also of the interactions between these tasks and the swarm. In the first example, the same robot is required to execute both subtasks of a sequence. In the second example, tasks might not be executable due to a lack of robots.

#### Identity

I assume a complex task with two atomic subtasks that have a non-blocking sequential interrelationship. Additionally, the interrelationship requires the same robot to execute both subtasks of the sequence. I model this interrelationship using two additional conditions:

1. The condition swarm models the swarm; its capacity reflects the number of robots available. It models that a robot is removed from the swarm upon the start of $\tau_1$ and returned to the swarm only after the same robot completed $\tau_2$.

2. The condition t1 started models that $\tau_2$ can only be started by the robot that completed $\tau_1$.

Figure 4.9 shows the extended low-level model for a three-robot swarm, that is, the condition swarm is initially marked with three tokens. Furthermore, the condition work available has a capacity of one: each execution of $\tau_1$ must be directly followed by an execution of $\tau_2$.

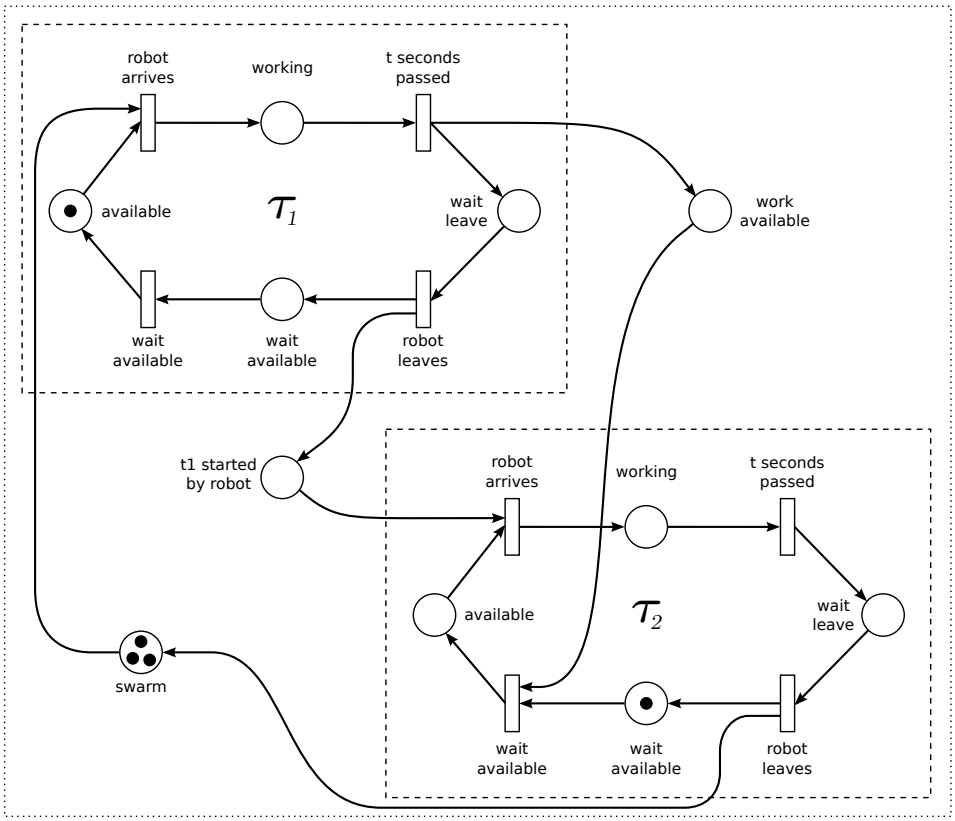



Figure 4.9: Low-level model of a complex task with a "same-robot" sequential interrelationship between its subtasks: the same robot is required to execute both subtasks of the sequence. The number of tokens initially in condition swarm reflects the size of the swarm. Dashed lines designate the subnets that model subtasks.

### Resource contention

I assume three atomic tasks without a sequential or concurrent interrelationship, that is, tasks can be executed in arbitrary order and at any time. Accordingly, task executions might overlap in time. I model the swarm as a condition swarm with a capacity that reflects the number of robots available in the swarm. The condition is initially marked with two tokens, representing the individual robots. Figure 4.10 shows the resulting low-level model.

Upon the execution of a task, a robot is removed from the swarm. It is returned only after the task has been completed. As there are more tasks than robots, this model only allows the concurrent execution of

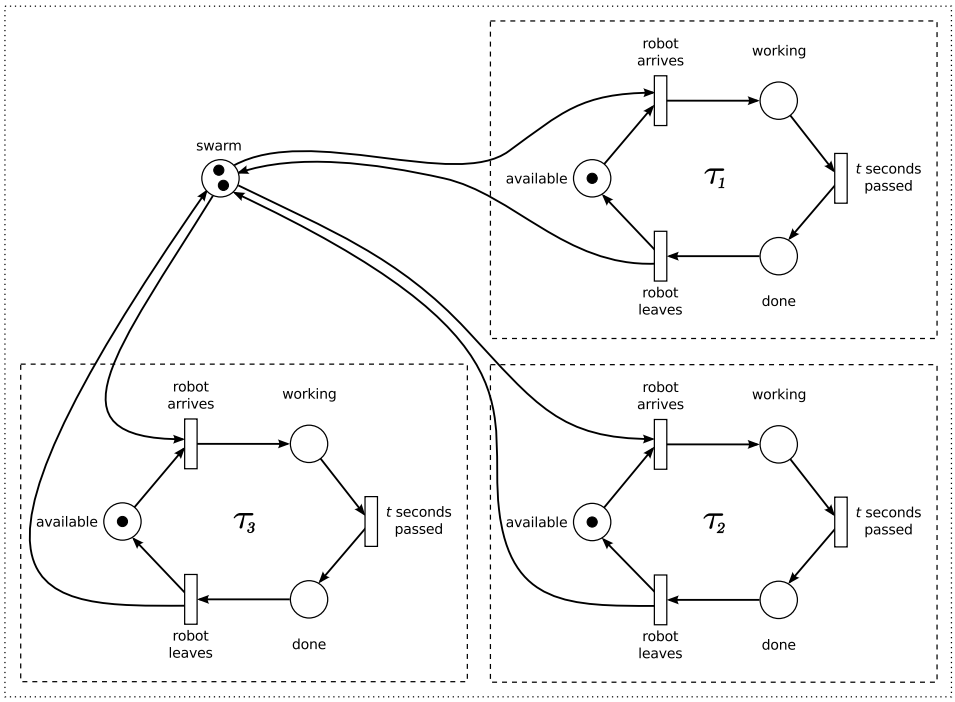



Figure 4.10: Low-level model of three atomic tasks that are not interrelated. The swarm consists of two robots, which means that only two atomic tasks can be executed at a given time. The number of tokens initially in place swarm reflects swarm size. Dashed lines designate the subnets that model the different atomic task.

two tasks. In other words, the model describes resource contention in form of a lack of robots: more tasks could be executed if more robots were available. Models of this kind can help to simulate and formally analyze deadlock situations for more complex tasks and larger swarms.

These examples demonstrate that the presented approach to modeling is sufficiently powerful to express various additional processes and aspects.

### 4.3.7 Discussion: Advantages and Limitations

The previous sections demonstrate that the low-level model is sufficiently powerful to describe a large variety of interrelationships. However, the resulting models are often complex and hard to maintain.

The choice of Petri nets for visually describing a complex task on a

low-level has certain advantages regarding the analysis of the model. First, there are plenty of existing software frameworks for the simulation of Petri nets, which allows researchers to simulate models of complex tasks before implementing them. Second, given the boundedness of the Petri nets, there is only a finite number of possible markings. This makes is possible to prove the liveness of the net, thereby ensuring that the net is deadlock free. Furthermore, one can trivially decide the reachability of a given marking, which can be used to decide whether a given complex task can be completed or not.

The choice of modeling all atomic tasks in a generic way also has certain advantages: maintaining these generic models as a subset in a larger model of a complex task simplifies modeling and analysis. Furthermore, as previously mentioned, the generic model can directly serve as basis for the software that controls the behavior of a single TAM—see Section 4.4.

## 4.4 Control framework

In this section, I present a framework for controlling a group of TAMs in an experiment. More specifically, the framework provides the basis for implementing the low-level model of a task on one or more TAMs.

Conducting experiments that involve solely atomic tasks without interrelationships is relatively easy: each atomic task is represented using a TAM and the behavior of each TAM replicates the state-machine described by the respective low-level model of each task. In other words, the low-level model serves as a direct "blueprint" for the software that controls the behavior of the individual TAM.

Unfortunately, conducting experiments that involve complex tasks is more complicated due to two reasons. First, the low-level model of a complex task possesses a distributed state, that is, its global state depends on the state of all of its atomic subtasks. In the case of simulation experiments, this would not cause any problems as all TAMs live in the same place—the simulator. In case of robot experiments, however, the TAMs are separated in space. Consequently, the state of the complex task is distributed over several, physically separate devices. Second, interrelationships between a set of atomic tasks cannot be attributed to a single TAM, but always involve two TAMs or more. However, the control software that implements these interrelationships has to be implemented on some device.

Arguably, it would be possible to develop a distributed control software that implements a given low-level model on multiple devices. This approach has several drawbacks: a) the required effort for developing such a distributed control software is high, b) each TAM would need to be accessed and programmed individually, and c) each TAM would possibly require a unique control software.

In order to address these difficulties, I propose a framework that allows researchers to implement a given low-level model as a centralized control software. The control framework consists of two main components: the central *coordinator* and the *firmware* running on each TAM—see Figure 4.11 for a graphical representation. The coordinator, which implements the low-level model of the complex task, consists of several components itself (see Chapter 5, Section 5.4 for a detailed discussion of the implementation). State changes of the associated low-level model are triggered by events that happen at the individual TAMs, which may result in commands that change the behavior of one or multiple TAMs. For example, if a TAM reports to the coordinator that its task has been completed, the model switches state, which in turn causes the TAM in question to switch off and other TAMs to become available. Note that the coordinator does not control the robots of the swarm, which remains a fully distributed system.

Events and commands are relayed using a wireless mesh network—see Chapter 5, Section 5.5 for an illustration of an example network topology and the flow of commands and data. The firmware of the TAM reports all events and changes in sensory data to the coordinator, and executes all commands that it receives in return.

The control framework that I propose has several advantages. First, it makes setting up and conducting experiments with TAMs relatively effortless, as changing the behavior of all TAMs requires only modifications at a central point rather than on many devices that are possibly distributed in space. Second, the central design allows for accurate statistics-keeping during experiments: all events can be recorded using a central time, which is required for the consistency of the experimental records. This, in turn, allows researchers to fuse data from multiple TAMs with external sensor data (e.g., from a tracking system).

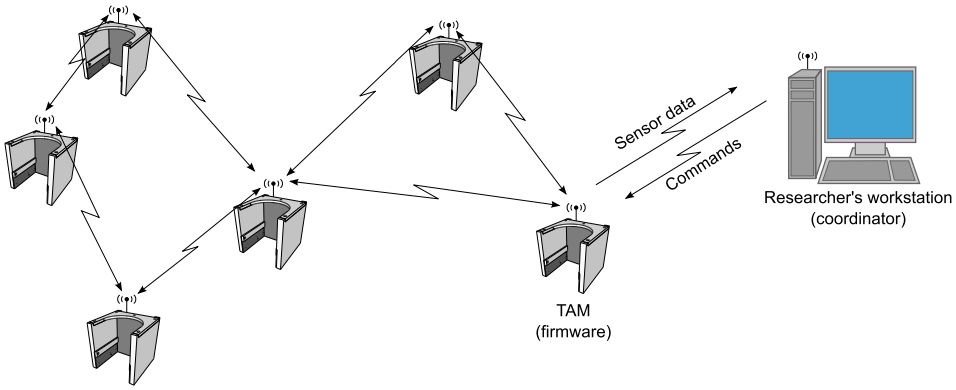



Figure 4.11: The control framework and its components. The coordinator implements the low-level model of a complex task. State changes of this model are triggered by events that happen at the individual TAMs, which may result in commands that change the behavior of one or multiple TAMs. Events and commands are relayed by a wireless mesh network that connects all TAMs and the coordinator.

## 4.5 Tasks studied in the literature

In this section, I review the swarm robotics literature with respect to the tasks considered, which I describe using the high-level model presented in Section 4.2. Furthermore, I group works according to similarities in their high-level model. This review serves two purposes.

First, it allows me to substantiate my claim of Chapter 1: the use of *ad hoc* solutions for task abstraction and representation limits the complexity of the tasks that are addressed in the literature. As I will show, the hierarchical structure of the majority of tasks studied in the literature is relatively simple: tasks are either atomic or consist of a single complex task. However, real-world tasks commonly exhibit a higher complexity as demonstrated by, for example, Dorigo et al. (2013).

Second, by modeling each task using the high-level model, I demonstrate how to use the presented modeling approach to abstract various complex tasks. This, in turn, outlines how these tasks could be represented using the TAM. In the discussion, I refrain from detailing how to represent each single task with the TAM—see Chapter 6 for an example of how to abstract a task using the modeling approach presented and represent the abstract task in an experiment with the TAM.

Third, having a clear classification allows researchers to identify the class of problem they are dealing with, and reuse solutions that have proved to be effective.

Note that I focus on works that consider atomic or complex tasks that can be represented with the TAM. This excludes, for example, many spatially organizing behaviors and navigation behaviors.

### 4.5.1 A note on task complexity

Examining the structure of the component subtasks of a given task can give insights into the group behaviors required to address these tasks (Anderson et al., 2001b). The underlying assumption is that with increasing complexity of this structure—that is, with an increasing number of subtasks and interrelationships—the complexity of the group behavior required to address the task also increases. A metric for measuring task complexity might therefore be a good indicator for the complexity of the required behavior.

Accordingly, several complexity metrics have been proposed in the literature, as previously mentioned in the discussion of the state of the art in Section 4.1. Authors commonly assign "complexity points" to different task topologies (Anderson et al., 2001b) or rate the "degree of interrelatedness" of the tasks by measuring some other quality (Korsah et al., 2013). However, it is difficult to propose a metric that is objective and consistent for the large variety of task types and interrelationships that exist. Furthermore, tasks might have a varying complexity depending on the group behavior used to address them. Therefore, any complexity metric is—to a certain point—subjective and reflects the designer's viewpoint.

In the context of this dissertation, for example, an intuitive choice for a complexity metric is to measure some property of a task's high-level model, such as the nestedness of its task relationship graph. Unfortunately, as such a metric would be based on the high-level model, it could not reflect differences between tasks that can only be described with the low-level model. As a consequence, the metric would be unable to distinguish between, for example, blocking and non-blocking sequential interrelationships—see Section 4.3.3. Using properties of the low-level model for measuring task complexity, on the other hand, poses a different kind of problem: the possibly infinite number of Petri net topologies and interrelationships makes it difficult to find an objective and consistent metric.

Accordingly, I refrain from proposing a complexity metric in the context of this dissertation. Instead, I group works according to similarities in their high-level model without imposing a strict order on their complexity.

### 4.5.2 Atomic tasks

There are several works that study atomic tasks. Atomic tasks cannot be decomposed into subtasks (i.e., the nestedness of their task relationship graph is zero). Atomic tasks can be readily represented using a single TAM. See Figure 4.1a for the high-level model of an atomic task.

Matarić et al. (2003) studied the problem of emergency handling. In their work, robots have to patrol an area and attend to alarms that appear in the environment. Alarms can be attended independently from each other by a single robot and can therefore be modeled as atomic tasks.

Brutschy et al. (2012c) studied behavioral specialization in a swarm of robots. In the work, two types of atomic tasks appear in the environment with a varying distribution. Individual robots have to specialize in one of the tasks by adapting their behavior. On the level of the swarm, the robots have to match the distribution of task types in the environment.

### 4.5.3 Complex tasks consisting only of atomic subtasks

Several works consider complex tasks that consist exclusively of atomic subtasks (i.e., the nestedness of their task relationship graph is 1). Complex tasks of this type can be distinguished according to the interrelationship among their subtasks; tasks that have the same type of interrelationship differ only in the number of these subtasks. Note that, while the exemplary high-level models that I reference in the following have two subtasks, they can be easily extended to a higher number of subtasks.

A complex task whose subtasks have a sequential interrelationship requires that subtasks are executed in a given order—see Figure 4.1b for the high-level model of such a task.

A commonly studied problem that involves complex tasks of this type is foraging for food or energy. In a simple version of this problem, hereunder called *simple foraging*, robots must balance energy consumed by the process of foraging with the energy provided by the collected food items (Krieger and Billeter, 2000; Li et al., 2004; Labella et al., 2006). The subtasks of the simple foraging task exhibit a sequential interrelationship which lies in the fact that robots first have to locate an item in the environment and then transport it to a predefined drop-off location. The same type of complex task has been studied in the form of a "waste cleanup" scenario (see, e.g., Parker, 1998). Note that in simple foraging, robots do not need to collaborate in order to complete a single task instance—instead, a single robot performs all the subtasks in sequence.

Contrary to the various works considering foraging, Ijspeert et al. (2001) studied a non-transportation task with a sequential interrelationship. The goal of the robots is to pull sticks from the ground. The length of the sticks is such that a robot cannot pull it from the ground in a single motion; instead, a second robot has to continue the pulling motion in order to complete the task. Hence, the subtasks have a blocking sequential interrelationship: contrarily to the simple foraging tasks discussed above, this task requires that robots cooperate to complete it.

Parker (1998) studied a hazardous waste cleanup task using real robots. The task of the robots is to collect objects at two "spill" locations, and transport them subsequently to a final destination. This task is essentially a simple foraging task as discussed above, but without the requirement to balance the energy level of the swarm.

A complex task whose subtasks have a concurrent interrelationship requires that subtasks are executed at the same time—see Figure 4.1c for the high-level model of such a task. Commonly studied complex tasks with this type of interrelationship among their atomic subtasks are collective transport tasks (Donald et al., 1997; Kube and Bonabeau, 2000; Groß and Dorigo, 2009).

### 4.5.4 Nested complex tasks

Nested complex tasks are tasks that consist of subtasks that are complex tasks on their own. Examples are the complex task shown in Figure 4.2 and the disaster response task considered in the proof-of-

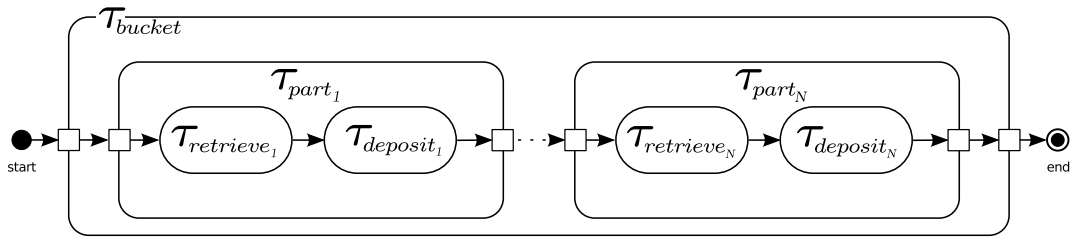



Figure 4.12: Generic high-level model of bucket-brigading tasks with a nestedness of 2: The overall task $\tau_{bucket}$ of transporting an object is partitioned into a sequence of $N$ complex tasks. Each complex task consists of two atomic subtasks for retrieving and depositing the object.

concept experiment presented in Chapter 6. Note that, as tasks can be arbitrarily nested, the task relationship graphs of a nested task can take various shapes and its nestedness is higher than 1.

The majority of the works that study nested complex tasks consider the same type of task: "bucket brigading". Bucket brigading is a special case of the simple foraging task: robots divide the original task into multiple smaller subtasks. Bucket brigading is an instance of the task partitioning problem (Pini, 2013).

Each subtask consists of transporting an object for a limited distance and subsequently transferring it to a robot working on the next subtask. Hence, the overall task is a sequence of foraging tasks, and each foraging task consists of a sequence of two atomic tasks (i.e., the nestedness of the task relationship graph is 2). Figure 4.12 shows the high-level model of a bucket-brigading task. Most works studying bucket brigading consider fixed partition sizes (Fontan and Matarić, 1996; Goldberg and Matarić, 2002). Pini et al. (2013b) studied this problem using TAMs, albeit only in a simulated setup with fixed partition sizes. Brutschy et al. (2014c) studied self-organized allocation to such tasks with two partitions of fixed size. The work published by Pini et al. (2014) is, to the best of my knowledge, the only one that studied a bucket-brigading task with dynamic partition sizes.

Different from bucket-brigading tasks, complex tasks that consist of several nested complex tasks are rarely studied in the literature, most likely due to the complexity and cost of the experiments needed to study them. In the following, I outline two exemplary studies that

considered tasks with several nested complex tasks. I refer the reader to the respective publication for details on each task.

In the context of the Swarm-bots project,[12] Nouyan et al. (2009) studied allocation to a collective transportation task—see Figure 4.13a for the high-level model of this task. The complexity of the task lies in the fact that robots first establish a chain of landmarks between source and nest; subsequently, other robots use this chain to navigate while collectively transporting an object from the source to the nest. The nestedness of the task relationship graph is 3.

One of the most complex tasks found in the swarm robotics literature has been presented by the Swarmanoid project:[13] a swarm collectively explores an environment, identifies an object to retrieve, and uses self-assembly and collective transport to retrieve it (Dorigo et al., 2013)—see Figure 4.13b for the high-level model of this task.[14] The nestedness of the task relationship graph is also 3.

### 4.5.5 Discussion

The review of the literature shows that the vast majority of works consider tasks without or with a limited number of interrelated subtasks. Moreover, to the best of my knowledge, most of the works that consider nested complex tasks tackle tasks that exhibit only one type of interrelationship, namely sequential bucket-brigading tasks. I speculate that the restriction to tasks with a low number of interrelated subtasks is due to the costs involved in studying such tasks: performing real-robot experiments for these tasks requires considerable effort and resources, as illustrated by the two studies that consider multiple nested complex tasks. As a result, swarm robotics systems are to date incapable of tackling complex tasks that consist of a large number of interrelated subtasks—a fact that limits the possible application of swarm robotics to real-world problems.

From this insight, two directions for future research are discernible: one, the development of group behaviors and collective decision processes that allow a swarm to tackle tasks of this kind, and two, the development of robotics hardware that is sufficiently capable, cost-effective, and robust to apply a swarm of robots to such tasks. The

[12] http://www.swarm-bots.org/

[13] http://www.swarmanoid.org/

[14] See http://youtu.be/M2nn1X9Xlps for a video describing the swarm and its task in detail (Dorigo et al., 2011).

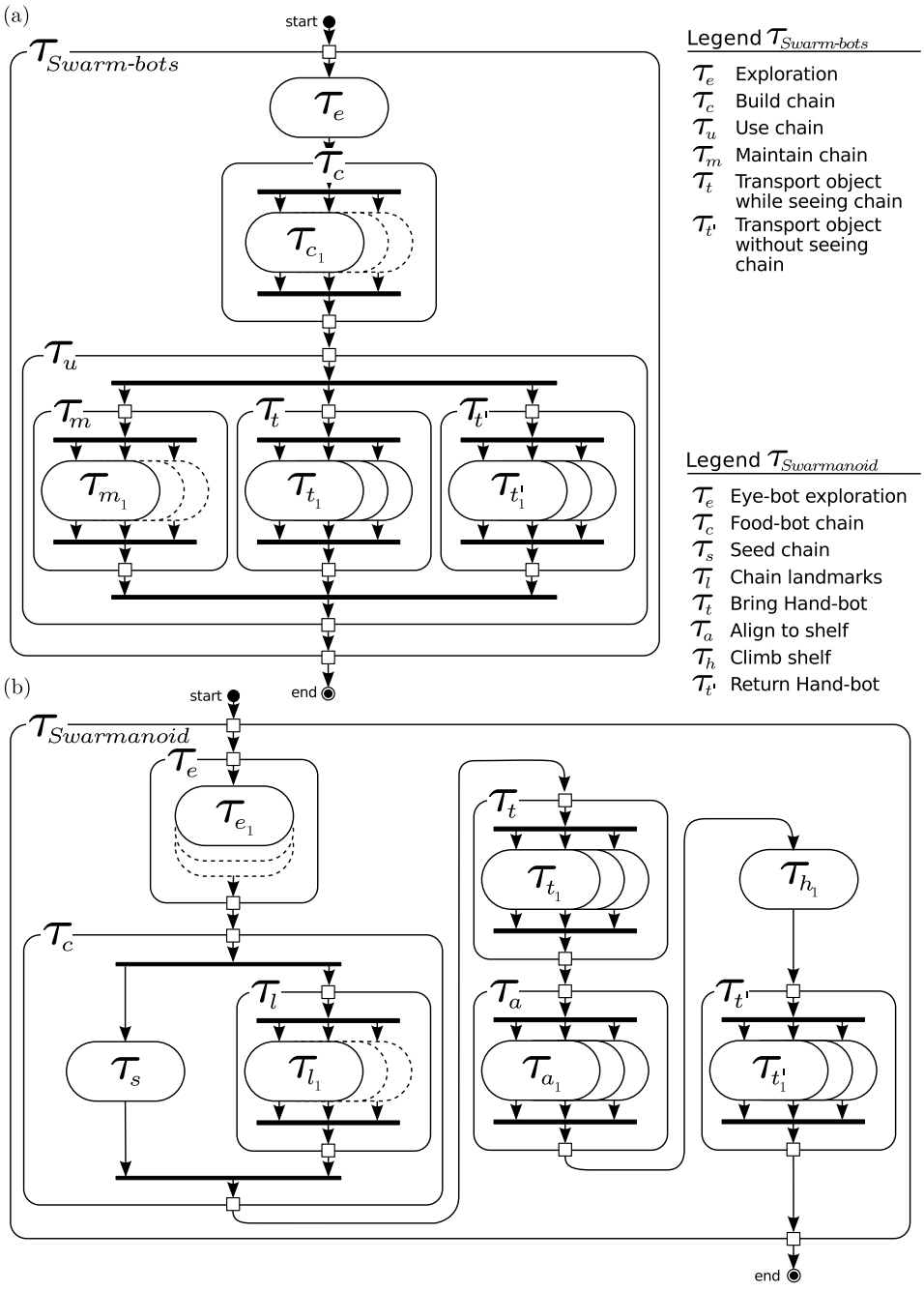



Figure 4.13: High-level UML model of two tasks with multiple nested complex tasks: a) Task studied by Nouyan et al. (2009) in the context of the Swarm-bots project;[12] b) Task studied by the Swarmanoid project (Dorigo et al., 2013).[13] Atomic tasks with dashed lines represent tasks where the number of tasks with the same depth is more than three or can vary. Both tasks have a nestedness of 3.

tools presented in this dissertation enable researchers to work towards the first direction.

## 4.6 Summary

In this chapter, I presented a novel two-level approach to modeling complex tasks. I presented how to use UML 2.x activity diagrams to conveniently model complex tasks at a high level, which permits researchers to isolate the hierarchical structure of the subtasks and to define the type of their interrelationships. Once the hierarchical structure of the subtasks and their interrelations has been isolated, researchers can model the details of these interactions using a low-level model based on Petri nets. The low-level model allows researchers to simulate and analyze a task if desired.

The modeling approach presented is generic and powerful, which I demonstrated by modeling a large variety of different tasks and interrelationships as well as using the approach to describe and classify tasks studied in the swarm robotics literature. The approach is therefore an essential contribution towards the study of complex tasks in swarm robotics research: researchers can abstract tasks of various complexity, which can then be conveniently represented in laboratory experiments using the TAM.

In the following Chapter 5, I describe the design and implementation of the TAM device in detail.

Chapter 5

# Design and implementation of the TAM

In this chapter, I present the design and implementation of the TAM, a custom-built device intended for uniform task representation. In particular, I discuss the design requirements of the TAM that are a consequence of the preceding chapters: the TAM is an active device that represents stationary single-robot tasks in laboratory experiments. Furthermore, a group of TAMs must be able to represent complex tasks that consist of several interrelated subtasks—a requirement that is the primary cause for complexity in the design of the TAM.

This chapter is structured as follows. In Section 5.1, I review the state of the art in robot docking, a problem related to the TAM insofar that similar hardware designs have been proposed in its context. In Section 5.2, I discuss the requirements for the design of the TAM and how I attained them in the implementation. In Section 5.3, I present the hardware implementation of the TAM, explaining the different choices that I made during the design and discussing details such as the type of electronic components employed. In Section 5.4, I present the software implementation of the TAM and its surrounding infrastructure such as the control framework. In Section 5.5, I discuss the mesh network that connects the TAMs used in an experiment with each other and with the researcher's workstation. In Section 5.6, I present two experiments in order to evaluate the reliability of the TAM's hardware and software. In Section 5.7, I outline how the TAM can be reproduced, adapted, and extended by other research groups.

In Section 5.8, I provide a summary of this chapter.

## 5.1 State of the art

Designs similar to the TAM have been proposed in the context of robot docking. Robot docking refers to the problem of perceiving, approaching, and connecting to a stationary object, commonly referred to as the docking station. A typical application is the autonomous recharging of a robot: the robot has to locate the recharging station in the environment, approach it, enter into it, and establish an electrical connection to the charger. Due to the requirement of establishing a connection, much of the literature attributed to robot docking focuses on mechanical connectors between the robot and the recharging station.

In the following, I discuss several approaches to robot docking proposed in the literature. In particular, I focus on aspects relevant to the TAM such as the infrastructure provided to allow robots to perceive the docking station.

Hada and Yuta (2001) were among the first to propose automatic recharging stations for autonomous mobile robots. The authors used a charging station equipped with spring contacts for electrical connectivity. The robot detected and navigated to the station using its ground sensors and environmental cues in the form of reflective tape on the ground.

Silverman et al. (2002) proposed a docking station for recharging a Pioneer 2DX robot. In the work, the authors focused on the design of a docking mechanism that provides a mechanical and electrical connection between the station and the robot. Similar to the TAM's case, a robot perceives the position of the docking station using a camera; contrarily to the TAM's case, the authors relied on passive colors rather than using LEDs—a choice made possible by the fact that robots to not have to discern between docking stations of different types, which means that a single passive color is sufficient to identify a station. The authors tested the proposed solution using only a single station and a single robot.

Cassinis et al. (2005) also presented a charging station for a Pioneer 2DX robot, but focused on the detection of the station and navigation towards it. Similar to the TAM's case, a robot uses its camera to detect and approach the station: the authors proposed a vision-based

system depending on "range lights" as commonly used in nautical navigation to indicate a certain approach vector and distance from the goal. Robots use this system for aligning to and approaching the charging station.

In the field of swarm robotics, Bonani et al. (2010) proposed a distinctly different approach to autonomous recharging: rather than recharging the robot's battery *in situ*, dead batteries can be exchanged "on the fly" with a charged one. This is made possible by a super-capacitor that powers the robot during the exchange. The authors proposed an autonomous charging station for performing the exchange that holds a reservoir of charged batteries and recharges dead batteries in its care. This approach has the advantage that changing batteries does not require an electrical connection and is significantly faster than recharging *in situ*; its disadvantage is the mechanical complexity of the station itself. In the work, the authors do not discuss sensing of and navigation towards the charging station.

McLurkin et al. (2006) published a work on robot-human interaction in large swarms. More precisely, the authors focused on the difficulty of conducting experiments that involve large swarms of SwarmBots.[15] In the work, the authors identify costs related to setup and operation as one of the main difficulties when conducting experiments. Accordingly, the authors proposed a comprehensive solution for conducting experiments—a solution that includes autonomous charging stations for the robots, a physical support infrastructure, and a centralized utility software for development and debugging. The authors initially planned to use global beacons for navigation, but stated that navigation using local communication performs better without requiring additional infrastructure.

## 5.2 Design requirements and overview

I designed the TAM following a number of requirements that I define and discuss in this section. Moreover, I outline how I attained these requirements in the implementation of the TAM.

The design requirements are a consequence of the discussions of the previous chapters. More specifically, the design requirements are

[15] Not to be confused with the *swarm-bot*, a composite entity formed by several *s-bot* robots proposed by the Swarm-bots project (see Mondada et al., 2004). http://www.swarm-bots.org/

driven by the concept presented in Chapter 3 and fulfill the requirements mandated by the modeling approach presented in Chapter 4. Furthermore, I formulated the design requirements under consideration of the advantages and disadvantages of the various existing solutions for task abstraction.The technical design requirements of the TAM are as follows:

1. **Atomic tasks:** The TAM shall be be able to represent stationary single-robot tasks (i.e., atomic tasks).
2. **Complex tasks:** The TAM shall be able to represent complex tasks described using the modeling approach presented in Chapter 4.
3. **Flexibility:** The TAM shall be able to represent different tasks without requiring physical reconfiguration.
4. **Compatibility:** The TAM shall be compatible with the e-puck robot and its various extensions.
5. **Visibility:** The TAM shall be recognizable under difficult light conditions.
6. **Identification:** The TAM shall be able to identify robots that interact with it.
7. **Communication:** The TAM shall be able to exchange information with a robot that has entered into it.
8. **Data recording:** The TAM shall be able to reliably record experimental data.
9. **Autonomy:** The TAM shall be autonomous in the sense that it should allow for several hours of consecutive operation and should be relatively unrestricted in its placement in an experimental setting.
10. **Cost-effectiveness:** The TAM shall be considerably cheaper than the e-puck robot.

In the following, I outline how the design of the TAM meets these requirements.

In the context of the discussion of existing solutions for task abstraction in Chapter 3, Section 3.2, the TAM can be classified as an active

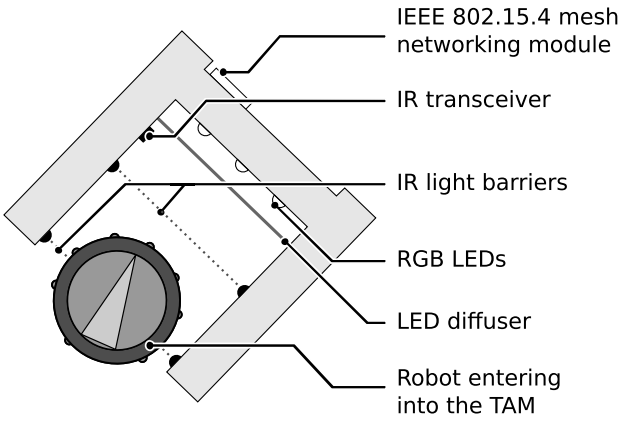



Figure 5.1: Schematic drawing of the TAM. The TAM is a booth with a cubical shape, large enough that an e-puck robot can enter into it. Different types of tasks and different states of task execution can be signaled to the robot using the RGB LEDs of the TAM. By defining interrelationships between multiple TAMs using wireless communication, the researcher can model a large number of tasks.

object: it is a dedicated, custom-built device that can be actively controlled using its electronics. Furthermore, the TAM incorporates some elements of virtual solutions, most notably the capability to freely control most of the parameters of a task it represents. Figure 5.1 shows a schematic drawing of the TAM.

Physically, the TAM is a booth into which an e-puck can enter. In particular, it is sufficiently large that it can accommodate an e-puck equipped with various extensions. The TAM is equipped with RGB LEDs, light barriers, and an infrared transceiver for communication. The TAM can use its RGB LEDs to announce to robots the presence, availability, and state of execution of the single-robot task it represents. The fact that the brightness and color of the RGB LEDs can be individually controlled facilitates perception in various light conditions. Different types of single-robot tasks can be signaled by using different LED colors, which can be perceived by the e-puck robots using their color camera. The light barriers allow the TAM to detect the presence of a robot that entered into the TAM. The infrared transceiver can be used to communicate with an e-puck robot inside the TAM, as well as to identify this robot.

The TAM can be remotely operated from a central computer using wireless communication, which allows a group of TAMs to implement

the low-level model of a complex task described using a Petri net. Furthermore, the TAM can report experimental data to the computer using wireless communication. As it is equipped with a rechargeable battery, the TAM can operate without being physically tethered to the central computer. The TAM relies on cost-effective off-the-shelf hardware components and uses existing infrastructure of the e-puck. As a consequence, the TAM is considerably cheaper than the e-puck robot.

## 5.3 Hardware

The hardware of the TAM consists of several parts that are either active (i.e., electronics) or passive (i.e., plastic body). In the following sections, I explain the implementation of these parts in detail. Figure 5.2 provides an overview of the different parts in form of an exploded view of the TAM.

### 5.3.1 Electronics

The electronic circuit of the TAM is distributed over three printed circuit boards (PCBs). The *main* circuit board, on the back face of the TAM, is flanked by two boards, forming a "U" shape. I refer to the two boards at the side of the TAM as the *left* and *right* circuit board.

The main circuit board is mechanically joined with the other circuit boards at a 90° angle using two interlocking slots.[16] This connection provides the structural backbone of the TAM: all plastic body parts are attached to one of the three circuit boards. Electrical connections between the circuit boards are provided by solder joints at the point where the circuit boards meet. The main circuit board contains the majority of the circuit, while the left and right circuit boards contain only the part of the circuit related to the sensors of the TAM. Figure 5.3 gives an overview of the functional components of the TAM in form of a block diagram. Appendix B details the circuit schematics and the circuit board layouts in Section B.1.2 and Section B.1.3, respectively.

[16] Also known as a "halved joint".

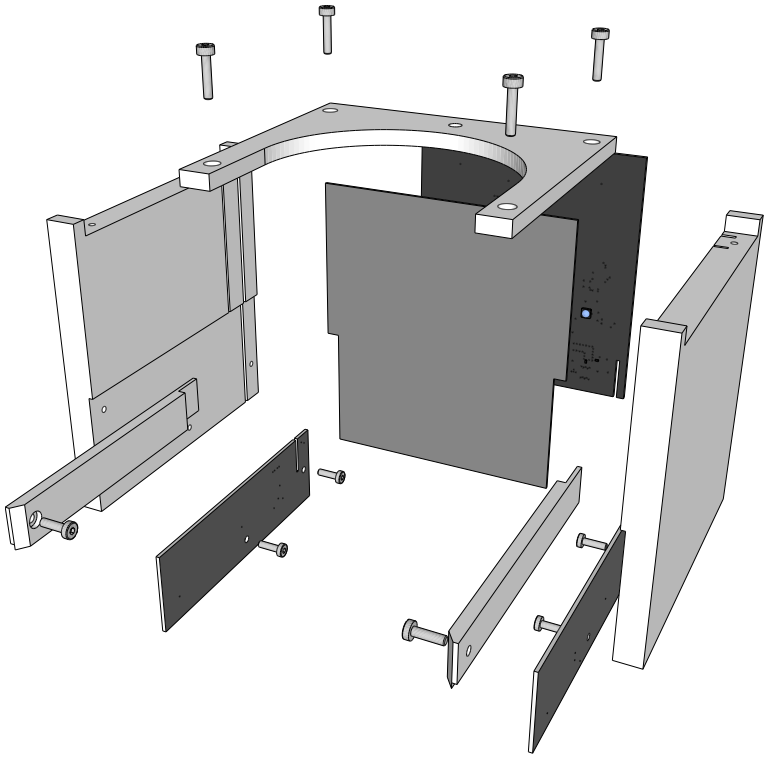


Figure 5.2: Exploded view of the TAM showing its different parts and their relative positioning. There are three electronic parts: the main board and two boards on the left and right side (dark gray). All other parts are passive plastic parts that constitute the body (light gray/white).

The TAM is based on *Arduino*,[17] an open-source embedded electronics platform that uses an Atmel micro-controller of the AVR family as a central processor. Arduino is widely available, supported by a large community, and easier to use than other embedded electronics platforms (Banzi, 2008).

The TAM adopts an 8-bit RISC processor (ATmega-1284p) running at 16 MHz. The processor is equipped with 16 KiB main memory and 128 KiB flash memory. It runs the firmware that controls all the electronic components of the TAM, either directly or, in case of the LEDs and the IEEE 802.15.4 mesh networking module, indirectly. The firmware implements the behavior of the TAM—see Section 5.4.

The TAM features two infrared light barriers, one at the entry of

[17] http://www.arduino.cc/

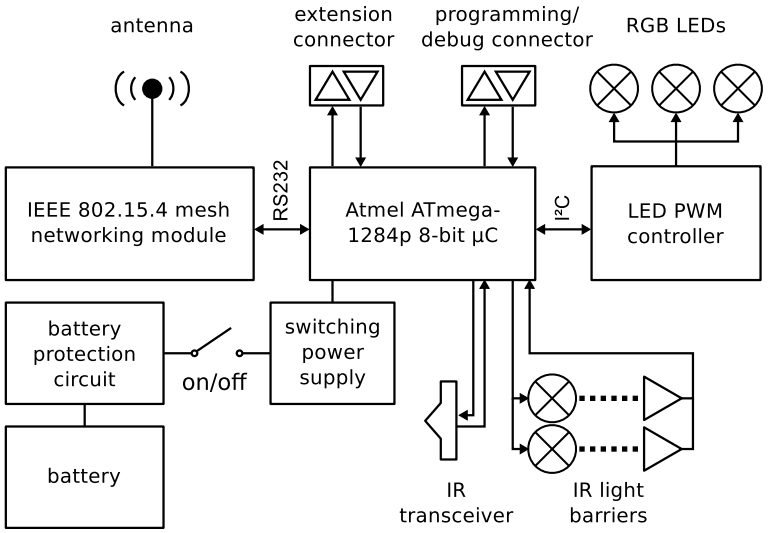



Figure 5.3: Block diagram showing the functional components of the TAM. The Atmel AVR 8-bit RISC processor directly controls the infrared light barriers and the infrared transceiver. The LEDs are controlled indirectly using a separate PWM controller connected via $I^2C$; the mesh networking module is also controlled indirectly using a serial connection.

the booth and one close to its rear wall (see Figure 5.1). Using two light barriers has the advantage that the TAM can distinguish whether a robot is entering or leaving the TAM. The light barriers are implemented using 850 nm infrared emitters and matching photo transistors. The light barriers use a pulsed signal in order to prevent false positives due to noise and ambient light. Furthermore, the emitters and transistors are mounted close to the ground in order to minimize interference with the proximity sensors of the e-puck.

The TAM possesses four RGB LEDs: Three LEDs are mounted on the main board at the rear wall of the booth, which makes them visible to robots approaching the opening of the booth. The LEDs announce available tasks to the robots; different colors signify different tasks or task states. The LEDs are diffused by a sheet of semi-transparent plastic to facilitate detection by the camera of the e-puck. As these LEDs are not visible from every angle, a fourth LED mounted on top of the TAM can provide feedback to the researcher. All LEDs are controlled using pulse-width-modulation (PWM) via a dedicated controller, which is connected to the central processor using an $I^2C$

| Parameter | Value | | | Unit |
|---|---|---|---|---|
| Nominal supply voltage | 1.8–5.5 | | | V |
| Nominal battery voltage | 2.7–4.2 | | | V |
| Nominal operation voltage | 3.3 | | | V |
| Supply current at | 2.7 | 3.3 | 4.2 | V |
| maximum load | 577.5 | 453.9 | 346.4 | mA |
| normal load | 148.4 | 124.6 | 94.5 | mA |
| idle load | 141.0 | 113.9 | 90.8 | mA |

Table 5.1: Electrical characteristics of the TAM. Supply currents reported are averages of 10 measurements taken when running the default firmware. Maximum load: constantly communicating, all channels of all LEDs at maximum brightness; Normal load: occasionally communicating, one channel of all LEDs at 19% of maximum brightness (TAM announces task in a typical experimental setting[18]); Idle load: occasionally communicating, all LEDs off (no task).

bus. PWM provides fine-grained control over the color and brightness of every color-channel of each LED separately. As a result, the TAM supports 24-bit colors that facilitate calibration under various light conditions.

Communication between the TAM and the e-puck is implemented using the *IRcom* protocol. The IRcom protocol has been proposed for inter-robot communication using infrared transceivers—transceivers that are present on most robots in the form of proximity sensors. On the e-puck, a library called *libIrcom* provides IRcom support (see Appendix A, Section A.1.4). The firmware of the TAM supports the IRcom protocol using a reimplementation of libIrcom for the Atmel AVR processor architecture—see Appendix B.2.1, Section B.2.1 for details on this reimplementation. To ensure compatibility, the TAM uses the same infrared transceiver as the e-puck robot.

The autonomy of the TAM is facilitated by a rechargeable lithium-ion battery with a capacity of 5 Wh. This battery is of the same type as the one used for the e-puck robot, which significantly eases charging and handling of the batteries during the course of an experiment. The TAM uses a highly efficient boost/buck switching power supply when powered with said battery. Table 5.1 reports the electrical characteristics of this power supply. Based on the values reported in Table 5.1,

the TAM consumes approximately 0.4 W in a typical experimental setting.[18] As a result, a single battery lasts over 10 hours. Note that this value depends predominantly on the LED brightness used as the power consumption of the TAM's other subsystems remains constant under most experimental conditions.

The TAM possesses a battery protection circuit but no charging circuit. Consequently, charging the TAM's battery requires the same external charger as required for the e-puck. Note that the TAM can also be supplied via the serial connector using an external power supply cable. However, this connector does not support many mating cycles and is therefore not intended to be used on a regular basis.

The TAM is remotely controlled by the *coordinator*, a software package running on a central computer. The coordinator implements the low-level model of the task represented by the TAM, explained in Section 5.4. The communication between the TAM and the coordinator is wireless, implemented using a 2.4 GHz IEEE 802.15.4 mesh networking module—see Section 5.5 for more information on the mesh network.

Figure 5.4 shows the electronics on the back face of the TAM, featuring the IEEE 802.15.4 mesh networking module and the battery.

### 5.3.2 Body

The TAM has a cubical shape with a length of 12 cm in every dimension. Its body is composed of six parts:

- the left and right *side walls*, which are fastened to the left and right circuit boards, respectively;
- the left and right *bumpers*, which act as a rail at the inside of the booth in order to prevent robots from getting stuck on the sensors of the light barriers;
- the *LED diffuser*, which diffuses the light from the RGB LEDs for better perception by the robot's cameras;
- the *top bracket*, which provides structural integrity to the whole assembly.

[18] I assume the proof-of-concept experiment presented in Chapter 6 to be a typical experimental setting.

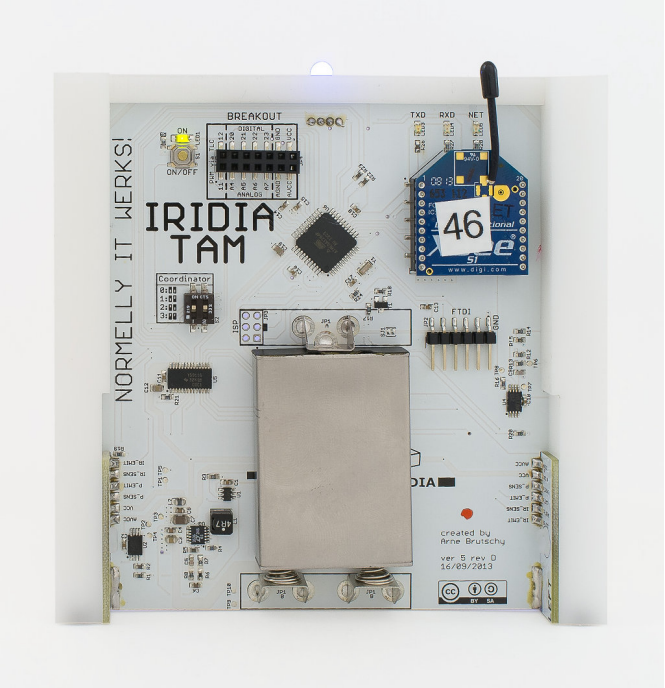



Figure 5.4: Photo of the electronics on the back face of the TAM. At the top right, the 2.4 GHz IEEE 802.15.4 mesh networking module. At the top left, the connector that provides an interface for extending the TAM. At the bottom center, the lithium-ion battery, the same type of battery that is used for the e-puck robot.

Figure 5.2 shows the relative positioning of all parts. In the original design, all parts of the body consist of CNC-machined Polyoxymethylene plastic (POM). However, simpler and cheaper alternatives such as wood can be used if desired. Appendix B, Section B.1.4 provides an overview of the whole assembly, including CAD models for each part with detailed measurements.

I designed the body of the TAM so that an e-puck can enter into the TAM without accidentally moving it. This is achieved by two measures: first, the chosen material for the body (POM) is relatively heavy, and second, I equipped the TAM with six rubber feet that increase friction between the TAM and the floor.

The TAM is compatible with various extensions of the e-puck. Consequently, it can be used with the basic model of the e-puck as shown in Chapter 3, Figure 3.1, or with extended e-pucks such as shown in Figure 5.5. The TAM is wide enough for an e-puck to enter without

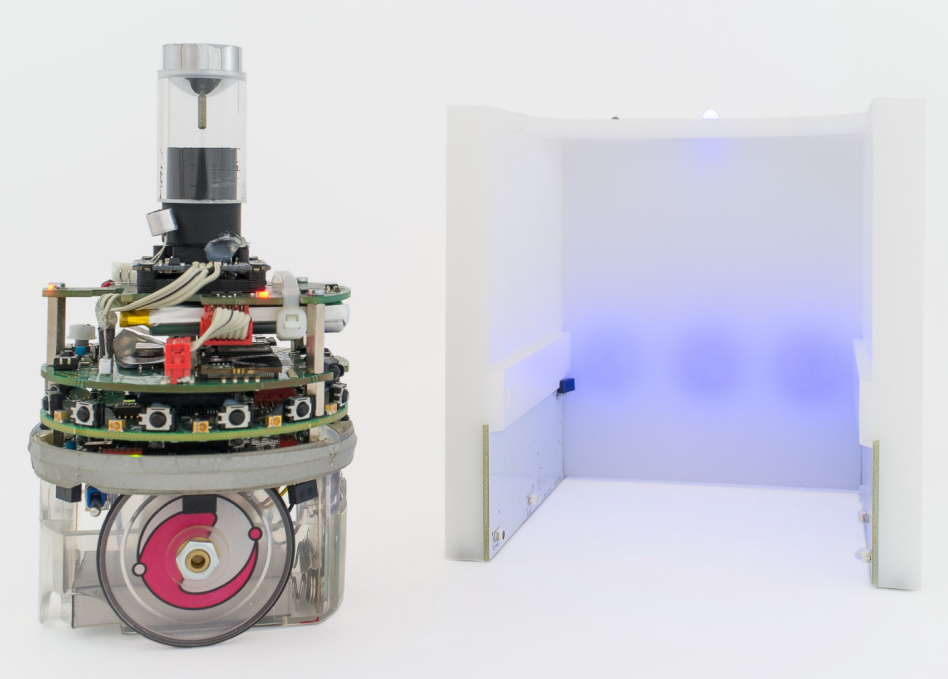


Figure 5.5: The TAM is compatible with various extensions of the e-puck. The e-puck shown here is equipped with several extensions: a range and bearing sensor, an embedded computer running Linux, and an omni-directional camera. In case the omni-directional camera is present, it is used to detect the TAM instead of the forward-facing camera. See Appendix A, Section A.1 for detailed information on the e-puck platform and the available extensions.

problems, while being narrow enough to provide directional information to an approaching robot. This is especially helpful when using the forward-facing color camera of the basic model of the e-puck, as distances cannot be obtained from the captured images. On the other hand, the directional information provided by the TAM is not required if the e-pucks are equipped with an omni-directional camera extension, as robots can compute distances and angles directly from the captured images.

Experiments have shown that e-pucks can detect the TAM when positioned in a conical area that extends up to a distance of 90 cm from the TAM's opening because they can only perceive a TAM's LEDs from an acute angle (see Figure 5.6).

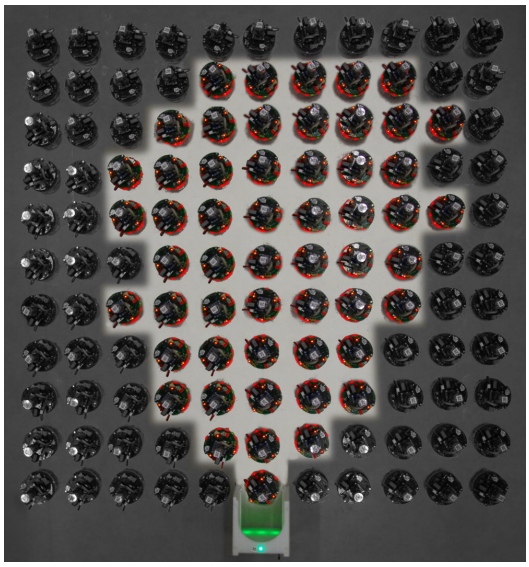


Figure 5.6: Area in which an e-puck is able to perceive the TAM when using the omni-directional camera extension. To obtain this image, an e-puck was placed on a 10 cm by 10 cm grid with random orientations. The e-puck signalled that it was able to perceive the TAM at a given location by setting its LEDs to red. The asymmetry in the perception area results most likely from imperfections in the mirror of the omni-directional camera.

As mentioned above, the TAM announces different task types by using different LED colors. Hence, the number of different tasks that the TAM can announce depends on the capability of the robots to differentiate between LED colors. Experiments have shown that e-pucks equipped with the omni-directional camera extension are able to differentiate between to six tasks. See Appendix A, Section A.1 for detailed information on the e-puck platform and the available extensions.

## 5.4 Software

In an experiment, the behavior of a group of TAMs is controlled by a user-defined software that implements the low-level model of the task at hand—see Chapter 4. In order to facilitate the development of this software, I provide a *control framework* that abstracts from the low-level details of the TAMs and the mesh network that connects them.

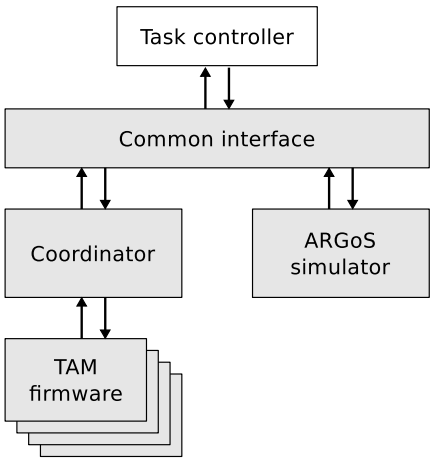



Figure 5.7: Overview of the software components for the TAM (gray components are provided, white ones user-defined). Task controllers can transparently interface with a group of TAMs using the common interface, whether these TAMs are simulated or physical devices.

In the following, I focus on this control framework for conducting experiments involving real robots.

The control framework consists of four components:

1. the *firmware* that locally controls all the TAMs involved in an experiment;
2. the *coordinator* that handles the communication with the TAMs;
3. the *common interface* that provides an abstract interface for the user-defined task controller;
4. the *task controller* that controls the behavior of the TAMs by implementing the low-level model of a task.

Figure 5.7 shows the relationship between these components.

Simulation experiments are, as mentioned in Chapter 2, an essential tool for swarm robotics research. Accordingly, I also provide the software necessary to simulate the TAM with the ARGoS simulation framework. ARGoS is highly modular and extensible; I therefore provide the software required for simulating the TAM and the e-puck as a set of plug-ins—see Appendix A for a details.

The common interface enables the transferability of task controllers between the ARGoS simulation framework and the control framework presented here. By using the common interface, researchers can develop generic task controllers that can be ported without requiring any changes from simulation experiments to robot experiments, and *vice versa*. The first three components and plug-ins for simulating the

TAM using ARGoS are readily provided in the context of this dissertation (gray boxes in Figure 5.7). The task controller is the sole component that must be implemented by the user.

### 5.4.1 Firmware

A TAM is controlled locally by its *firmware*, an embedded software executed by the main processor of the TAM. The firmware is written in the programming language used by Arduino (a dialect of the *C* programming language) and programmed into the flash memory of the main processor. Changing the firmware currently requires a physical connection to the TAM, which is very time-consuming—especially in experiments that involve a large number of TAMs, possibly scattered throughout the environment. In order to alleviate this problem, I designed the firmware so that researchers do not have to change it during normal operation.

This is achieved by exposing all the functionality of the TAM via an API that can be accessed using the wireless mesh network. This API allows researchers to query and control the state of the TAM remotely: it reports all events and changes in sensory data to the coordinator and executes all commands that it receives in return. Control software such as the user-defined task controller can therefore be implemented on the central computer that runs the coordinator. Consequently, the API allows researchers to implement any functionality without any modification to the firmware of the TAM.

Note that the problem of requiring a physical connection to the TAM in order to change the firmware could be solved as follows. Currently, a so-called *boot loader* allows researchers to program the flash memory of the main processor using a USB connection. Without such a boot loader, programming the flash memory requires special programming equipment. Similarly, using a more advanced type of boot loader would allow researchers to program the main processor with data received over the mesh network. However, this solution would require a TAM to reboot and is therefore of little use during an ongoing experiment.

### 5.4.2 Coordinator

The *coordinator* is a centralized software component that handles the interaction between the TAMs in an experiment and the user-specified

task controller. In order to communicate with the TAMs, the coordinator requires an IEEE 802.15.4 radio module. The coordinator performs low-level operations required to query and control a large number of TAMs, for example, node discovery in the mesh network, handling packet loss, and polling the status of TAMs that have not reported a change for a long time. Furthermore, it provides an interface that exposes the state of each TAM and all of its functions—the common interface, discussed below.

The coordinator is a library written in Java; it is therefore platform independent. Internally, the coordinator uses an event-driven framework for handling asynchronous communication with the TAMs. More specifically, it executes two types of events: periodic ones and reactive ones. Periodic events are management tasks such as node discovery and polling. Reactive events are either triggered by incoming data from a TAM (e.g., sensor data) or commands invoked by the task controller (e.g., change LED color). Incoming sensor data is cached by the coordinator; a task controller can therefore transparently access the current state of all TAMs. Commands invoked by a task controller are translated by the coordinator and executed remotely on the respective TAMs using the aforementioned firmware API.

### 5.4.3 Common interface

The *common interface* provides an abstract interface to the TAMs used in an experiment. It allows researchers to query and modify the state of an individual TAM in an abstract fashion; all low-level operations are handled by the coordinator. The interface is common in the sense that the same interface is used for simulation experiments, that is, the common interface is replicated by the plug-in that allows researchers to simulate the TAM using ARGoS. Consequently, task controllers written against the common interface are can be ported from simulation to the robots without requiring an changes. See Appendix B, Section B.2.2 for the complete definition of the common interface.

### 5.4.4 Task controller

The *task controller* is the only component of the control framework that has to be implemented by the researcher. The task controller

interfaces with the TAMs in an experiment though the common interface exposed by the coordinator. The task controller is executed on the central computer that runs the coordinator; in fact, the coordinator is a library used by the task controller. This architecture allows researchers to write task controllers in a very convenient fashion: the researcher neither has to program a low-level device such as the TAM, nor to wrestle with low-level network protocols as used by the mesh network. This makes setting up and conducting experiments with TAMs relatively effortless, as changing the behavior of all TAMs involved in an experiment only requires researchers to change the task controller at the central computer.

The task controller can govern the behavior of an individual TAM as well as a group of TAMs. More specifically, there is a $1 : n$ mapping between task controllers and TAMs. Typically, it implements the low-level model of a complex task as described in Chapter 4, Section 4.3: each state of an atomic task reflects the state of an individual TAM, and the interrelationships between states are implemented as conditions for state transitions in the controller.

Multiple task instances are supported by the coordinator by instantiating multiple copies of the task controller. However, each TAM can only be mapped to a single controller, that is, a given TAM can only represent an atomic subtask of a single complex task. Note that the coordinator is not limited to a single type of task controller: several different task controllers can be used in the same experiment.

Together, the firmware, the central coordinator, and the common interface provide a ready-made environment to implement the low-level model of any complex task. The software package that I provide includes a template controller that implements the generic model of an atomic task as presented in Chapter 4, Section 4.3.5.

Apart from the implementation of controllers, the centralized nature of the coordinator has an additional advantage: all events can be recorded at a central point using the same clock, which enables accurate and consistent statistics-keeping during experiments using a single clock-source. Furthermore, this allows researchers to correlate data from multiple TAMs with external sensor data, for example, data of a tracking system—see Chapter 6 for a demonstration.

## 5.5 Network

Communication between the TAMs and the central coordinator is implemented using a mesh network. Mesh networks are a type of network in which all nodes relay data; typically, mesh networks do not differentiate between routers and clients. Consequently, communication links can be established between any node of the network rather than between a client-node and a router-node as in traditional networks. The resulting network topology is closely meshed (hence the name), approaching a fully connected network with increasing link density—contrarily to networks following a client-router principle, which traditionally use a sparsely connected star or bus topology. See Chapter 4, Figure 4.11 for an example topology of the mesh network between a group of TAMs.

Wireless mesh networks have certain advantages over traditional wireless networks, most importantly in terms of scalability and robustness when paired with a self-healing networking protocol. Due to these advantages, mesh networking enables experiments with a many TAMs scattered over a large area, while reducing the risk of communication failures caused by the broadcast nature of traditional wireless networking.

The TAM uses a 2.4 GHz IEEE 802.15.4 radio module for connecting to the mesh network. The TAM can be configured to work on 4 different wireless channels, which allows up to four experiments to be run in close proximity: TAMs involved in each experiment would operate on the same channel and would not interfere with the TAMs used in the other experiments. The TAM is compatible with IEEE 802.15.4 modules of various manufacturers. Furthermore, the TAM and the coordinator support various mesh-networking protocols, most notably ZigBee 2.0.

In the context of this dissertation, I use XBee Series 1 network modules provided by *Digi Inc.* and the associated proprietary mesh-networking protocol, called *DigiMesh*. DigiMesh is a self-healing, *ad hoc* mesh networking protocol that does not require dedicated router nodes. Furthermore, it can be used with the inexpensive XBee Series 1 network modules, whereas the ZigBee 2.0 protocol requires costly modules and is relatively complex to configure.

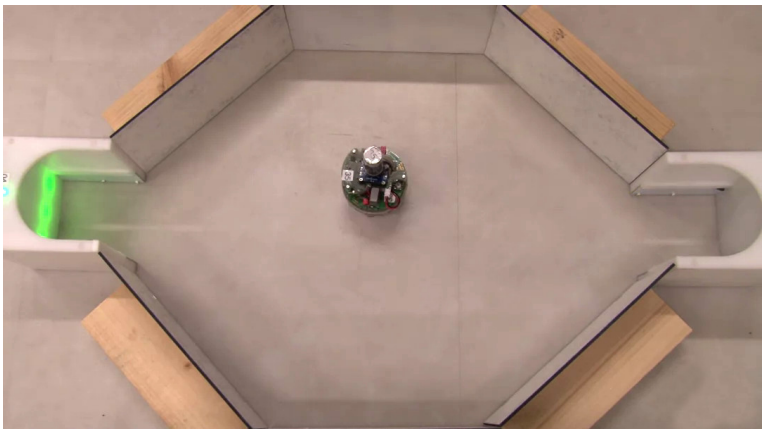


Figure 5.8: Snapshot recorded during the first TAM reliability experiment, where a single e-puck robot has to continuously perform two stationary atomic tasks. In the situation shown, the e-puck has recently completed the blue task represented by the TAM on the right side. The TAM on the left side is now announcing a green task, which is perceived by the e-puck.

## 5.6 Reliability experiments

I conduct two experiments in order to evaluate the reliability of the TAM device, the mesh network, and the control framework. Both experiments have been recorded using a video camera. Additionally, I recorded all the available data from the TAMs using the coordinator. Videos and data are available in the supplementary online material (Brutschy et al., 2014b).

The goal of the first experiment is to measure the reliability and battery life of the TAM. In the experiment, a single e-puck robot has to continuously perform two stationary atomic tasks. The robot has to alternate between tasks; each task is abstracted using a single TAM. Figure 5.9 shows a snapshot that illustrates the arena and the position of the TAMs. The experiment is terminated once the battery of the robot is depleted. I conducted a single trial that terminated after 40 min. During this time, the robot executed a total of 96 single-robot tasks. No failures occurred during the runtime of the experiment.

The goal of the second experiment is to evaluate the mesh network and the control framework in terms of scalability. In the experiment, all 50 TAMs that I produced are used to abstract 50 atomic tasks. Figure 5.9 shows a snapshot that illustrates the arena and the position

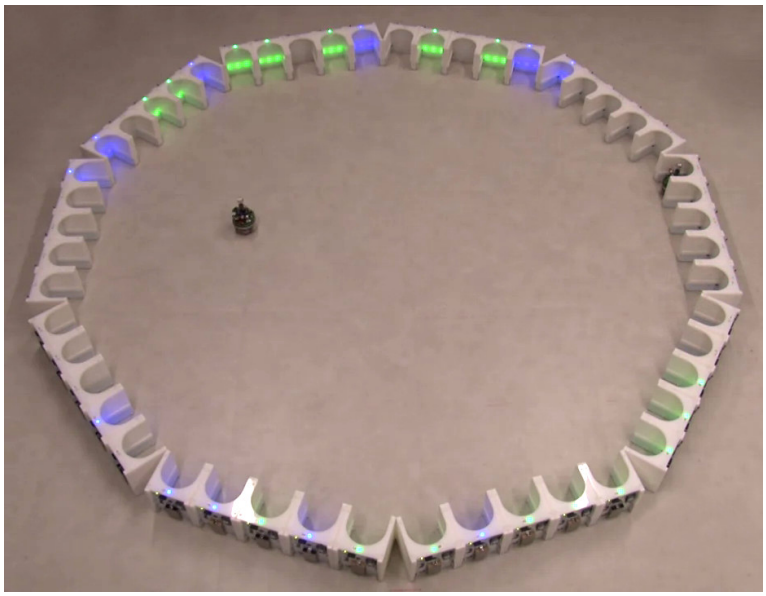


Figure 5.9: Snapshot recorded during the second TAM reliability experiment, where all 50 TAMs that I produced are used to abstract 50 atomic tasks. In the situation shown, the two e-pucks have already completed about half of the tasks available.

of the TAMs. The tasks have to be performed by two e-puck robots. The experiment terminates once each of 50 atomic tasks has been performed exactly once. I conducted a single trial that terminated after 17 min. Again, no failures occurred during the runtime of the experiment.

In addition to these experiments, we can consider the data recorded during the proof-of-concept experiments presented in Chapter 6 in terms of reliability. The data shows that during these demonstrations, the robots performed 35 complex tasks successfully. Two tasks failed because robots abandoned a task due to sensor noise or other technical problems. No failures occurred because of the TAMs or the coordinator.

## 5.7 Use for other research projects

I designed the TAM so that it can be easily reproduced by other research groups and adapted to other robot platforms. The TAM is—similarly to Arduino—easily extensible, which allows students and re-

searchers to extend the capabilities of the TAM without requiring a full redesign. In the following, I describe reproduction, adaptation, and extension of the TAM in more detail.

### 5.7.1 Reproduction

The TAM is open source, including all its components in hardware and software. All components, except the IRcom library, are licensed under the *Creative Commons Attribution-ShareAlike 3.0 Unported License*. The license allows others to modify and reuse the design as long as it remains open source. Additionally, the license grants the rights to anyone to produce the TAM commercially—see Appendix B, Section B.3 for the full license.

The production of the TAM is possible for a moderately equipped laboratory: all electronic components such as chips and discrete devices are common and easily obtainable from distributors all over the world. Furthermore, apart from the electronic components, the printed circuit boards and the body can be produced in-house. However, I advise that the printed circuit boards and the plastic body be produced by a professional service for consistency in quality (e.g., mechanical tolerances).

Even if the printed circuit boards and the plastic body are produced by a professional service, the cost of the TAM is relatively low—provided that the final assembly is done in-house. The total cost of a single TAM is 140 € when producing a quantity of 50 TAMs. Table 5.2 details the cost of each component of the TAM. In the table, I also indicate how costs change when increasing the quantity to produce in moderate numbers, that is, numbers that can still be assembled in-house.[19] Note that, in case the required facilities are available, the printed circuit boards as well as the plastic body can be produced in-house, lowering the cost even further. For example, a 3D-printer can be used to produce the plastic body, possibly even in a single piece.

See Appendix B, Section B.1 for all technical information required to reproduce the hardware of the TAM: the bill-of-materials, circuit schematics, and layouts of the printed circuit boards, as well as CAD drawings of the plastic parts that compose the body of the TAM.

[19] Commonly, electronic components benefit from price breaks only for large quantities of 2000–3000 pieces, depending on the component.

Table 5.2: Costs of the different components of the TAM, indicated for a quantity of 50 TAMs, when produced using a professional service. The column "cost change" indicates how costs change when increasing the quantity to produce in moderate numbers (i.e., numbers that can still be assembled in-house).

| Component | Approx. cost per TAM | Cost change |
|---|---|---|
| Plastic body | 60 € | strongly decreasing |
| PCB production | 8 € | decreasing |
| PCB assembly | 12 € | decreasing |
| Electronic components | 60 € | constant[19] |

### 5.7.2 Adaptation to other robot platforms

Conceptually, the design of the TAM is generic: a booth into which a robot can enter. Accordingly, the TAM can be adapted to work with almost any mobile robot platform. In the following, I outline the steps necessary to adapt the TAM to three exemplary robot platforms common in swarm robotics research: the marXbot, the Khepera III robot, and the Kilobot.

#### MarXbot

The marXbot (Bonani et al., 2010) has been proposed in the context of the Swarmanoid project (Dorigo et al., 2013). The marXbot is similar to the e-puck insofar that it is of round shape, uses the same proximity sensors as the e-puck, and possesses a forward-facing camera as well an omni-directional camera. It is still frequently used in swarm robotics research and is the mobile robot platform of choice for several other research projects.[20,21]

Due to its similarity to the e-puck robot, adapting the design of the TAM to the marXbot would be relatively simple. For example, as the marXbot uses the same proximity sensors as the e-puck, communication could be implemented using the IRcom protocol. Therefore, adapting the TAM to the marXbot would primarily require redesign-

[20] http://www.ascens-ist.eu/
[21] http://www.e-swarm.org/

ing the plastic body—as the marXbot has a diameter of 17 cm, the body of the TAM would need to be considerably larger.

The marXbot possesses a gripper which can be used to connect to other marXbots or objects with a specific ring. This gripper has also been employed to perform foraging tasks using custom-designed passive objects (Brutschy et al., 2012a; Pini et al., 2013a, 2014). Furthermore, the marXbot has been extended with a magnetic gripper module used for autonomous construction task (Magnenat et al., 2012). As the marXbot can already execute a variety of tasks using its gripper, it can be argued that adapting the TAM to it is of questionable utility. However, the grippers are expensive, error prone, fragile, and require behaviors that are not relevant to the core of swarm robotics research. As such, I believe that adapting the TAM to the marXbot would be a worthwhile endeavor.

### Khepera III

The commercially available Khepera III robot[22] is of roughly rectangular shape. It is modular and can be extended with, for example, an additional processor running Linux, a forward-facing camera, and a gripper module.

Adapting the design of the TAM to the Khepera III is possible, even though this poses a few more challenges than for the marXbot. For example, it is unclear whether communication between the TAM and the Khepera III can be implemented using the IRcom protocol due to different proximity sensors. In general, building custom devices that interact with the Khepera III is difficult as the underlying hardware and software has not been released as open source.

Even though there exists a gripper module for the Khepera III, I believe that the adaptation of the TAM for the Khepera III is a worthwhile endeavor, mostly due to the limited availability and the prohibitive pricing of the gripper module.

### Kilobot

The most recent of the three robots discussed in this section, the Kilobot (Rubenstein et al., 2011, 2012) is also the simplest. The robot possesses a differential drive system consisting of three stiff legs, on

[22] http://www.k-team.com/mobile-robotics-products/khepera-iii

which it moves using vibration, following the principle of the common "bristlebots".[23] Additionally, the Kilobot features an infrared transceiver for communication and inter-robot distance sensing.

Adapting the design of the TAM to the Kilobot would require that the TAM announces task availability and type using infrared communication rather than LED color. The same system could also be used for general-purpose bi-directional communication between the TAM and the robot (currently implemented using the IRcom protocol).

Due to its simplicity, the Kilobot would benefit from the TAM insofar that one could study a considerably wider range of tasks with the Kilobot. However, as the governing design principle of the Kilobot was cost reduction, the design of the TAM would need to be optimized such that its cost is comparable to the cost of the Kilobot—currently approximately 120 € when purchased in retail.

### 5.7.3 Extensions

The design of the TAM in terms of extensibility is inspired by Arduino and the e-puck: it features an extension connector that allows researchers and students to easily extend its capabilities without requiring a redesign of the TAM itself. In addition to supplying power, the extension connector provides 10 input/output channels: 2 channels for PWM LED control, 4 digital channels and 4 analog channels. Appendix B, Section B.1.5 details the pinout schematic for the extension connector.

The variety of channels enables a wide range of possible extensions. An example of a possible extension is a light sensor that allows researchers to model the day/night cycle of a colony of social insects by changing the behavior of a group of TAMs depending on the ambient light.

## 5.8 Summary

In this chapter, I presented the design and implementation of a generic task representation: the TAM. A single TAM can represent stationary single-robot tasks in laboratory experiments. Used in conjunction with the modeling approach presented in Chapter 4, the TAM enables research on multi-robot tasks of various complexity.

[23] http://en.wikipedia.org/wiki/Bristlebot

I presented the design requirements of the TAM and how I attained them in its implementation. To this end, I detailed the implementation of the TAM in hardware and software. I evaluated the reliability of the implementation in two experiments using real robots. Furthermore, I discussed how the TAM can be reproduced, adapted, and extended by other research groups.

In the following Chapter 6, I will demonstrate how to represent a complex task with a group of TAMs after the task has been abstracted using the modeling approach presented in Chapter 4.

Chapter 6

# Proof-of-concept experiments

In this chapter, I present two proof-of-concept experiments that demonstrate the usage of the tools proposed in this dissertation. In particular, I show how a complex task can be modeled and abstracted using the approach presented in Chapter 4; how this task can be represented in an experiment using groups of TAMs; and how the proposed control framework can be leveraged to conduct an experiment involving swarms of robots.

The experiments presented in the following serve as a demonstration of the tools presented in this dissertation. As the intended application of these tools are laboratory experiments, the demonstration replicates the experimental setup commonly used in such experiments. For this reason, I collect data of all TAMs in the experiment and report various metrics such as the time a robot has to wait for an interrelated task. Note that this does not serve any other purpose than demonstrating the capabilities of the TAM and the centralized coordinator. In particular, as I do not attempt to study a specific algorithm or approach to solve a specific collaboration problem, the collection of data and its analysis is not meaningful beyond demonstrating how to perform these activities when using the TAM. Accordingly, I only conduct three experimental runs for each experiment, being well-aware of the fact that this would not allow me to make observations of statistical significance if my goal were to empirically study the behavior of the robots and assess their performance.

This chapter is structured as follows. In Section 6.1, I present the task to be addressed by the robots in the experiments. The task is a complex task whose subtasks have sequential and concurrent inter-

relationships. I model the task following the approach presented in Chapter 4, first by decomposing and describing it on a high level of abstraction, then by modeling it on a low level of abstraction. The low-level model serves as a basis for implementing the task controller that governs the behavior of the TAMs in the experiments. In Section 6.2, I present the experimental setup in terms of the arrangement of a group of TAMs to represent the complex task, as well as the robots and their controller. The first experiment, presented in Section 6.3, demonstrates how the proposed control framework can be leveraged to collect detailed data from each TAM. The second experiment, presented in Section 6.4, demonstrates how the TAM can be used for conducting experiments involving large swarms and several task instances. In Section 6.5, I provide a summary of this chapter.

## 6.1 Task

For the following experiments, I assume a fictitious disaster response task as it might occur after a nuclear accident. In particular, the overall task of the robots is to repair something inside a nuclear reactor. The task requires three robots to collaborate: two robots have to open the reactor lock to allow the third robot to enter the reactor chamber and perform the repair.

### 6.1.1 High-level model

In the following, I model the disaster response task on a high level using the approach presented in Chapter 4, Section 4.2.

The overall disaster response task $\tau_{response}$ is a complex task that consists of two subtasks with a sequential interrelationship: 1) $\tau_{open}$, the task of opening the reactor airlock and 2) $\tau_{repair}$, the task of repairing something inside the reactor. The task $\tau_{open}$ requires two robots to act concurrently on the airlock. To this end, each robot executes one of two atomic subtasks $\tau_{left}$ and $\tau_{right}$. After the airlock has been opened, it has to be kept open by the robots until a third robot has entered the reactor chamber. Once the third robot is in the reactor chamber, the robots of $\tau_{open}$ must leave. Once these robots left, the third robot can perform the repair, that is, work on $\tau_{repair}$. The disaster response task $\tau_{response}$ is completed once the reactor has been repaired by completing $\tau_{repair}$.

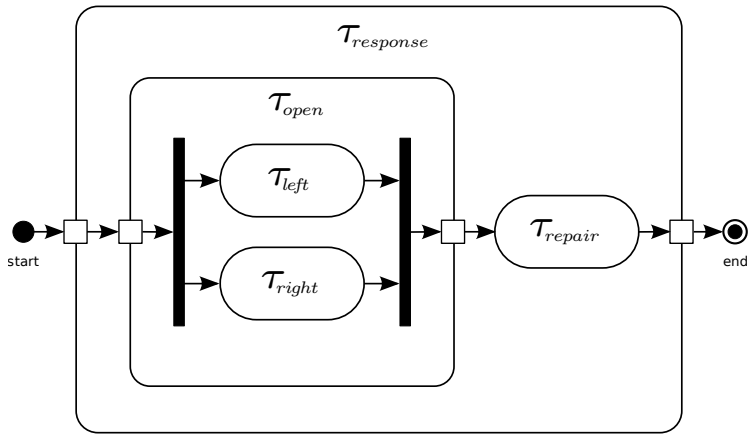



Figure 6.1: High-level model of the example task $\tau_{response}$. The model shows the hierarchical relationship between the atomic subtasks and defines the order of execution: $\tau_{left}$ and $\tau_{right}$ have a concurrent interrelationship forming $\tau_{open}$; $\tau_{open}$ and $\tau_{repair}$ have a sequential interrelationship forming $\tau_{response}$. Each of the three atomic tasks has to be performed by a single robot. The task relationship graph of $\tau_{response}$ has a nestedness of 2.

The task $\tau_{open}$ is a complex task that consists exclusively of atomic subtasks $\tau_{left}$ and $\tau_{right}$ (i.e., it has a nestedness of 1). As $\tau_{open}$ is the sole complex subtask of $\tau_{response}$, the nestedness of $\tau_{response}$ is 2. Figure 6.1 gives a visual representation of its task relationship graph described using UML 2.x activity diagrams.

### 6.1.2 Low-level model

In order to abstract $\tau_{response}$ using a group of TAMs, I have to transform its high-level model into a low-level model. I visually describe the low-level model using Petri nets as discussed in Chapter 4, Section 4.3. I use the resulting low-level model subsequently as a basis for implementing the task controller that governs the behavior of the TAMs.

Figure 6.2 shows the low-level model for $\tau_{response}$, visualized using the reduced version of its Petri net.[24] The transitions of the three

[24] By convention, the places of a Petri net can be omitted in order to visualize better the structure of the net (Petri and Reisig, 2008). The full version of the Petri net and instructions for simulating it can be found in the supplementary online material (Brutschy et al., 2014b).

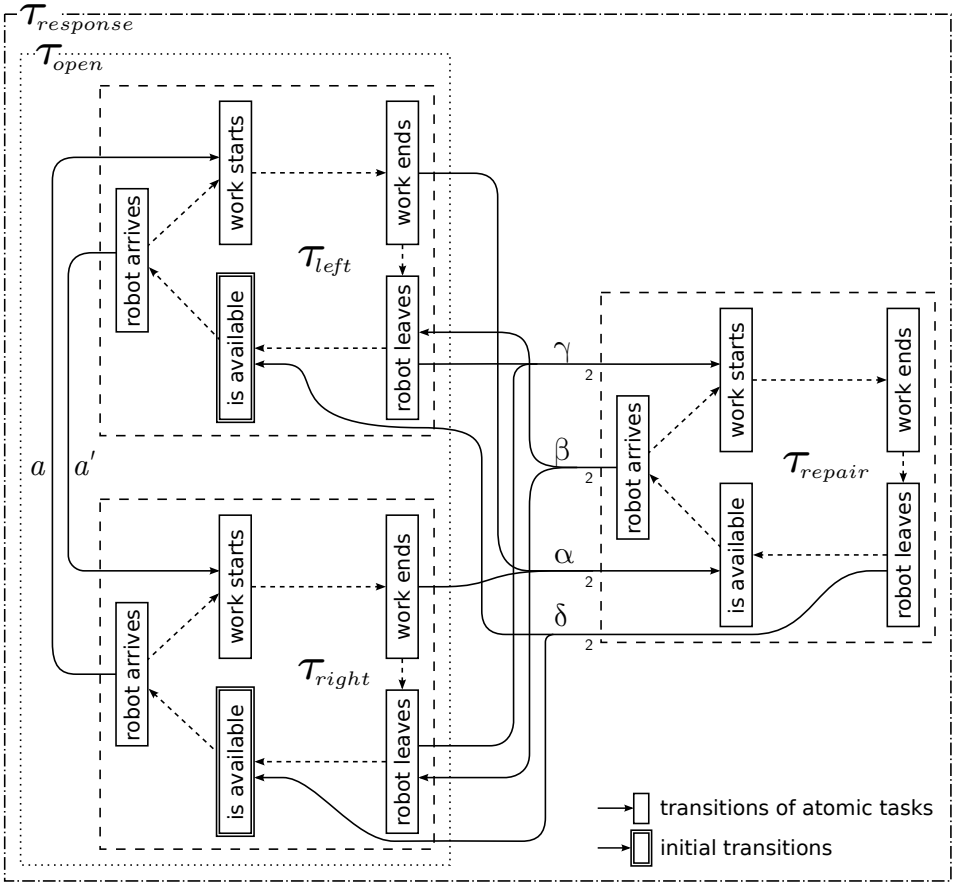



Figure 6.2: Low-level model of the example task $\tau_{response}$ (reduced version without places).[24] Please note that, as the initial marking cannot be visualized in the reduced version, I denote transitions that can fire at the start of the demonstration using a double border. The weight of all edges is 1 unless indicated otherwise. Edges that are internal to the functioning of an individual TAM are represented using dashed lines. Edges labeled $a$ and $a'$ model the concurrent interrelationship: a robot that is ready to work on $\tau_{left}$ can start working once a robot arrives to work on $\tau_{right}$ (and vice versa). Edges labeled using Greek letters model the sequential interrelationship: once $\tau_{left}$ and $\tau_{right}$ have been completed, $\tau_{repair}$ becomes available (edge $\alpha$); a robot that arrives to work on $\tau_{repair}$ allows the robots in $\tau_{right}$ and $\tau_{left}$ to leave (edge $\beta$); once these robots have left, work can start on $\tau_{repair}$ (edge $\gamma$); once work on $\tau_{repair}$ finishes and the robot leaves, $\tau_{right}$ and $\tau_{left}$ can become available anew (edge $\delta$). Dotted/dashed boxes indicate task boundaries.

atomic tasks $\tau_{left}$, $\tau_{right}$, and $\tau_{repair}$ are described in Figure 6.2 (labeled using Arabic numerals). Note that, even though I use the generic model of an atomic task (cf. Chapter 4, Section 4.3.5) as the basis for modeling the three atomic tasks, Figure 6.2 shows only the transitions that possess interrelationships with other tasks.

I model the concurrent and sequential interrelationships identified by the high-level model analogously to the models presented in Chapter 4, Section 4.3.3: the two atomic tasks $\tau_{left}$ and $\tau_{right}$ are subtasks of the complex task $\tau_{open}$ and have a concurrent interrelationship. In particular, I model this concurrent interrelationship so that work on $\tau_{left}$ and $\tau_{right}$ can only start when both robots are present, and each robot can leave only after both robots completed their work (edges labeled using Latin letters in Figure 6.2). Moreover, I model the sequential interrelationship between the complex task $\tau_{open}$ and the atomic subtask $\tau_{repair}$ such that the robots that completed $\tau_{open}$ must wait for a robot to arrive for $\tau_{repair}$ before they can leave—in other words, I assume a blocking sequential interrelationship (edges labeled using Greek letters in Figure 6.2).

Note that the low-level model also describes a condition not previously mentioned: $\tau_{left}$ and $\tau_{right}$ can only become available anew after $\tau_{repair}$ has been completed (edge labeled $\delta$ in Figure 6.2), that is, a second execution of the overall task $\tau_{response}$ can only commence after the first has been fully completed. While this condition does not reflect a mission objective of the particular disaster scenario described, it allows me to study a scenario in which the swarm has to execute many instances of $\tau_{response}$.

## 6.2 Experimental setup

I conduct two experiments: in the first experiment, six e-puck robots have to perform a single instance of $\tau_{response}$; in the second experiment, 20 e-puck robots have to perform six instances of $\tau_{response}$. I use more robots than strictly necessary in order to observe the effect of multiple robots competing for the same task.

In total, the task $\tau_{response}$ consists of three atomic subtasks. Accordingly, I use three TAMs to abstract a single instance of $\tau_{response}$, with each TAM abstracting one of the three atomic subtasks. I use the control framework described Chapter 4, Section 4.4 to implement a task controller that reflects the behavior of the low-level model shown

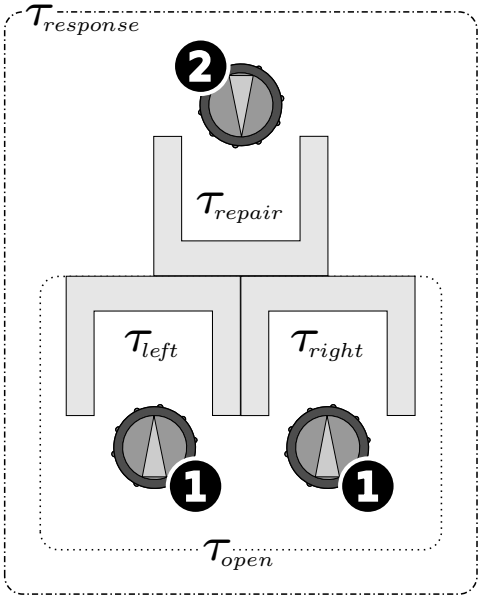



Figure 6.3: I use three TAMs to represent an instance of $\tau_{response}$, with each TAM representing one of the three atomic subtasks $\tau_{left}$, $\tau_{right}$, and $\tau_{repair}$. Dotted/dashed lines indicate task boundaries; white numbers in the black circles designate order of execution.

in Figure 6.2. The task controller governs the behavior of a single instance of $\tau_{response}$ by exchanging data and commands with the three TAMs that represent the three atomic subtasks of this particular instance. In case of the second experiment, six instances of the task controller are launched on the centralized coordinator (cf. Chapter 5, Section 5.4) in order to represent the six instances of $\tau_{response}$ using 18 TAMs. Note that in the following I use the term "TAM $\tau_x$" interchangeably with the term "task $\tau_x$".

Figure 6.3 illustrates how a single instance of $\tau_{response}$ is represented using a group of three interrelated TAMs. Figure 6.4 shows a close-up of the TAMs taken during one of the experiments. As Figure 6.4 shows, I use an extended version of the e-puck robot equipped with a range and bearing sensor, an embedded computer running Linux, and an omni-directional camera (see Appendix A, Figure A.1 for details). Note that the range and bearing sensor extension is not used by the robots in the following experiments, despite being mounted on the robots. All robots use an instance of the same controller: by default, robots perform a random walk. If a robot perceives a TAM with its omni-directional camera, it tries to enter into the TAM in order to start working on the task this TAM represents. The robots follow a simple *greedy* strategy to select tasks, that is, every robot tries to work on any available task it encounters. Upon completion of the task, the robot leaves the TAM and starts to perform a random walk again.

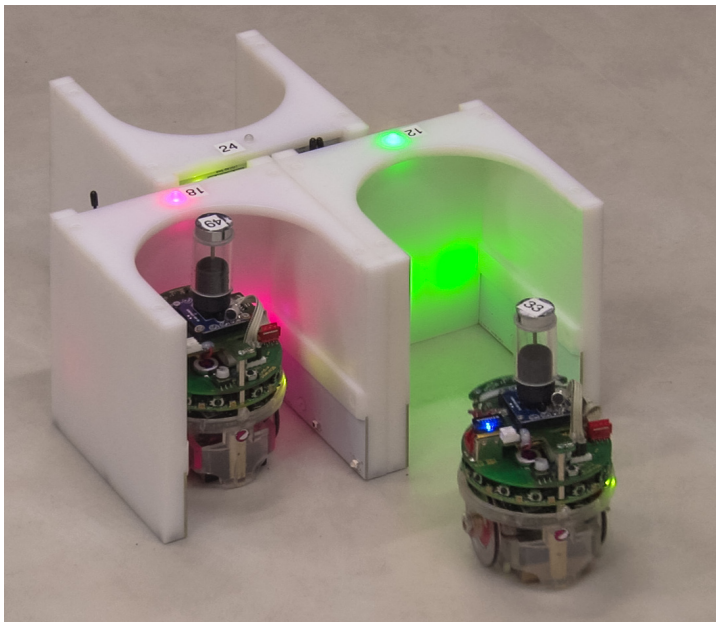


Figure 6.4: Close-up of the TAMs taken during one of the experiments. At the left, a robot has already entered into the TAM that represents task $\tau_{left}$; the TAM signals the robot to wait by changing the color of its LEDs to pink. At the right, the TAM representing $\tau_{right}$ signals the approaching robot that its associated task is available by its green LEDs. At the top, the third TAM, representing $\tau_{repair}$, is still idle as its sequential interrelationship with the complex task $\tau_{open}$ requires that $\tau_{left}$ and $\tau_{right}$ are completed before $\tau_{repair}$ can become available.

Both experiments have been recorded using an overhead camera. Additionally, I recorded all data available from the TAMs using the coordinator. Videos and data are available in the supplementary online material (Brutschy et al., 2014b).

## 6.3 Single-instance experiment

The first experiment illustrates how the centralized design of the control framework can be leveraged to collect detailed data from each TAM. I record various task-related data such as which robot executed which atomic task and the time a robot had to wait for an interrelated task.

In this experiment, I use three TAMs, six e-puck robots, and a $4\,\mathrm{m}^2$

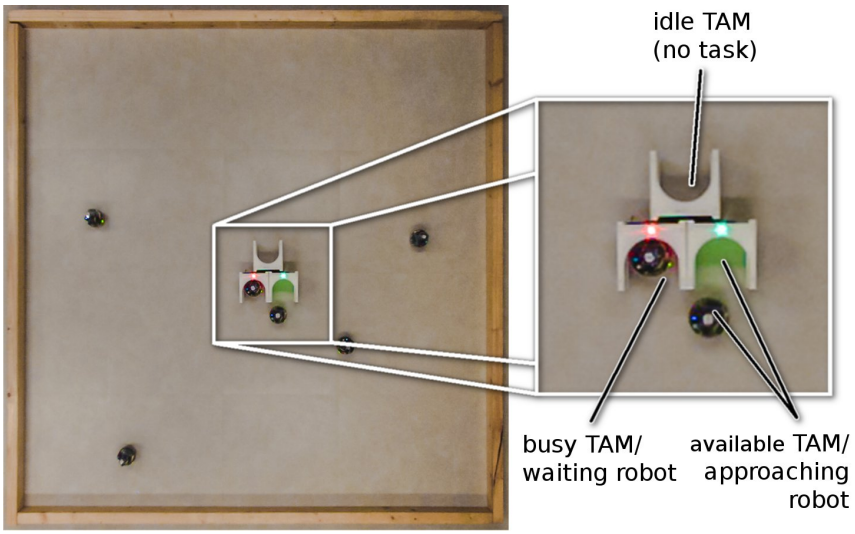



Figure 6.5: Snapshot recorded during the second run of the single-instance experiment, taken with an overhead camera in the same situation as shown in Figure 6.3. The arena is a $4\,m^2$ square. A single instance of $\tau_{response}$, represented using three TAMs, is placed in the center of the arena. I use six e-puck robots, randomly positioned in the arena at the beginning of the experiment.

square arena. The three TAMs are configured as shown in Figure 6.3 and placed at the center of the arena. Figure 6.5 shows a snapshot that illustrates the arena and the position of the TAMs. At the beginning of the experiment, six e-puck robots are positioned arbitrarily in the arena. The experiment terminates as soon as task $\tau_{response}$ has been completed once. Accordingly, the duration of the experimental runs can vary. Figure 6.6 illustrates how $\tau_{response}$ evolves over time.

### 6.3.1 Results

I performed three experimental runs. For each run, I recorded all the events that occurred at the individual TAMs using the centralized coordinator. An example for such an event is when a robot enters into a TAM: the TAM communicates with the robot to obtain its identity and reports it to the coordinator. The centralized recording of all events allows researchers to conveniently analyze and report a large variety of data.

For this experiment, I report the recorded data in the form of the

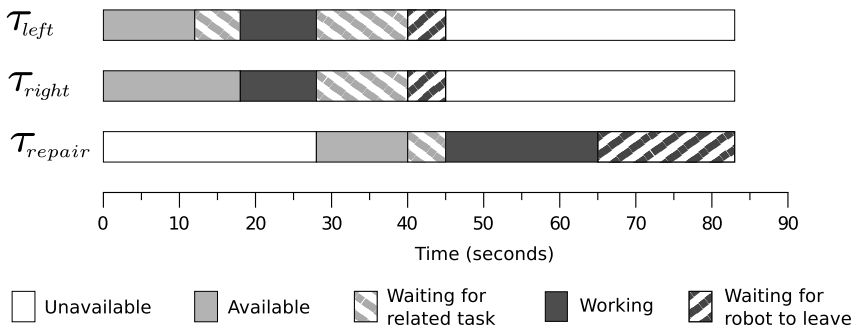



Figure 6.6: Evolution of the task $\tau_{response}$ over time in terms of state changes of its subtasks. The times shown are the result of an exemplary execution of a single instance of $\tau_{response}$ recorded by the coordinator during the second run of the single-instance experiment.

following metrics: $a_x$ denotes the time from the moment the TAM $\tau_x$ signals the availability of a task to the moment a robot is inside the TAM $\tau_x$; $d_x$ denotes the time the robot has to work on task $\tau_x$; I set $d_{left}$ and $d_{right}$ to 10 s and $d_{repair}$ to 20 s.[25] $w_x$ and $w'_x$ denote the time a robot has to wait before and after working, respectively. $t_x$ denotes the total time required to complete task $\tau_x$, that is, $t_x = a_x + w_x + d_x + w'_x$.

The time a robot has to wait before and after working on a task is a result of the interrelationships between TAMs. More specifically, $w_{left}$ and $w_{right}$ are due to the concurrent interrelationship between $\tau_{left}$ and $\tau_{right}$. For example, $w_{left}$ measures the time a robot ready to work on $\tau_{left}$ has to wait for a robot to arrive for $\tau_{right}$. Consequently, either $w_{left} = 0$ or $w_{right} = 0$ (or both, in the rare case that robots arrive for both tasks at the same moment in time). Furthermore, $w'_{left}$, $w'_{right}$, and $w_{repair}$ are due to the sequential interrelationship between $\tau_{open}$ and $\tau_{repair}$. More specifically, the robots working on $\tau_{open}$ have to wait for the arrival of a robot for $\tau_{repair}$. Therefore, $w'_{left} = w'_{right} = a_{repair}$. $w_{repair}$ is due to the fact that a robot ready to work on $\tau_{repair}$ has to wait for the robots of $\tau_{open}$ to leave before it can start working. $w'_{repair}$ is always zero as the robot can leave immediately after the completion of $\tau_{repair}$.

Table 6.1 reports the aforementioned metrics for each of the three experimental runs.

[25] Note that in this experiment, $d_x$ is constant and has been defined *a priori*. Other ways of defining $d_x$ are possible, for example, it could follow a pre-defined sequence of values or be a random variable drawn from a specific distribution.

Table 6.1: Detailed results of the single-instance experiment: $a_x$, time from the moment the TAM $\tau_x$ signals the availability of a task to the moment a robot is inside the TAM; $w_x$ and $w'_x$, time a robot has to wait before and after working on $\tau_x$; $d_x$, time the robot has to work on task $\tau_x$; $t_x$, total time required to complete task $\tau_x$. All times are reported in seconds.

| Run | Task | Robot ID | $a_x$ | $w_x$ | $d_x$ | $w'_x$ | $t_x$ |
|---|---|---|---|---|---|---|---|
| 1 | $\tau_{left}$ | 53 | 12 | 6 | 10 | 12 | 40 |
| | $\tau_{right}$ | 54 | 18 | 0 | 10 | 12 | 40 |
| | $\tau_{repair}$ | 49 | 12 | 5 | 20 | — | 37 |
| 2 | $\tau_{left}$ | 49 | 21 | 6 | 10 | 45 | 82 |
| | $\tau_{right}$ | 33 | 27 | 0 | 10 | 45 | 82 |
| | $\tau_{repair}$ | 53 | 45 | 6 | 20 | — | 71 |
| 3 | $\tau_{left}$ | 33 | 30 | 0 | 10 | 25 | 65 |
| | $\tau_{right}$ | 56 | 30 | 0 | 10 | 25 | 65 |
| | $\tau_{repair}$ | 53 | 25 | 5 | 20 | — | 50 |

In case of the complex tasks $\tau_{open}$ and $\tau_{response}$, I measure only the total duration of the task starting from the moment the task becomes available and ending at the moment the task has been completed. Due to the concurrent interrelationship of its subtasks, the total duration of $\tau_{open}$ equals $t_{left} = t_{right}$ (see Table 6.1). Due to the sequential interrelationship of its subtasks, the total duration of $\tau_{response}$ equals $t_{open} + w_{repair} + d_{repair}$. For the three experimental runs, the value of $t_{response}$ is 65, 108, and 90 s, respectively.

### 6.3.2 Discussion

In terms of data collection, the experiment demonstrates how researchers can record various task-related metrics using the centralized coordinator. More precisely, the results show the level of detail that can be obtained from an experiment. An example of these task-related metrics is the amount of time a robot that is ready to work on a task spends waiting for a partner. The ability to record such task-specific data, which is not trivial to obtain when using *ad hoc* task abstractions, enables the study of algorithms that leverage this data (see, for exam-

ple, Brutschy et al., 2014c)—data that is not trivial to obtain when using *ad hoc* task abstractions. Furthermore, the coordinator records the identity of each robot that enters into a TAM. This feature is of great utility when conducting experiments with large swarms.

In terms of experimental setup, the experiment demonstrates how the TAM broadens the range of tasks that can be represented in laboratory experiments. First, any task-related aspect can be closely controlled and modified during an experiment. For example, the object transport task studied by Pini et al. (2011b) can be completed by the robots in two ways: individually, by traveling through a corridor, or collectively, by partitioning the task and exchanging objects at a cache site. By abstracting the cache site using a set of TAMs, Pini et al. were able to vary the advantage of using the cache site over using the corridor—see Chapter 7, Section 7.3 for an in-depth discussion of this work. Without the TAM, this kind of study would require modifying the length of the corridor, which might incur unintended changes in the environmental parameters—for example, the robot density would change.

Second, the TAM's capability of identifying robots enables tasks that are specific to robots. An example is the task studied by Brutschy et al. (2012c): robots specialize in one of two possible task types. As each robot has its individual level of specialization, the duration of a task might be different for each robot—see Chapter 7, Section 7.1 for an in-depth discussion of this work. A study of this type would not be possible without the TAM's capability of communicating with the robots.

In summary, the experiment demonstrates that the tools for task abstraction presented in this dissertation facilates research on complex tasks in swarm robotics. In particular, the implementation of the TAM and the proposed control framework enable researchers to conduct experiments by allowing them to collect detailed data as well as analyze and report this data in a convenient fashion.

## 6.4 Multi-instance experiment

Expanding on the single-instance experiment, the multi-instance experiment demonstrates that the TAM can be used with several instances of a given task and larger swarms. Again, I use the centralized coordinator to record all events that occur during the experiment. I

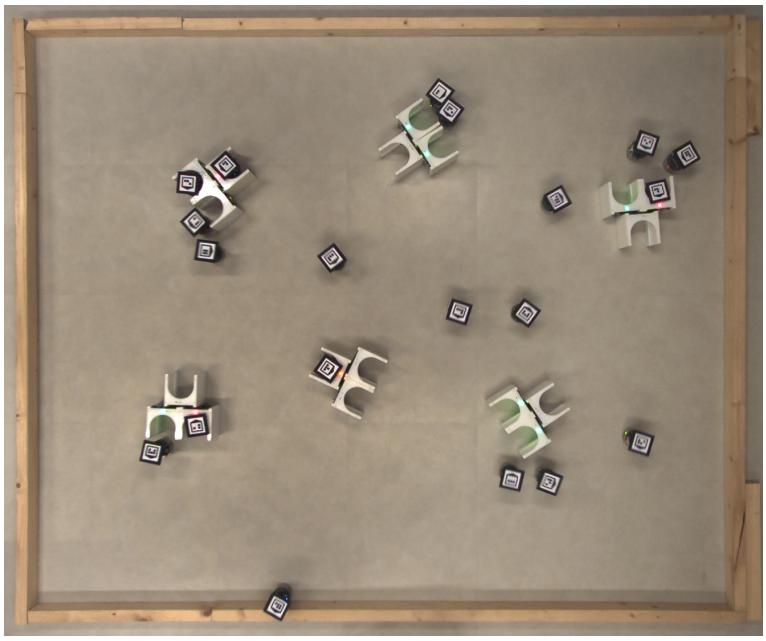


Figure 6.7: Snapshot recorded during the second run of the multi-instance experiment, taken with an overhead camera. The arena has dimensions of 2.7 m×2.2 m. Six instances of $\tau_{response}$, represented by a total of 18 TAMs, have been placed in the arena. I use a swarm of 20 e-puck robots, randomly positioned in the arena at the beginning of the experiment. The black-and-white visual tags on top of the robots are used by a ceiling-mounted tracking system to identify and track individual robots.

use this data to report metrics such the number of successful task executions per atomic subtask. Additionally, I use a ceiling-mounted tracking system to track the trajectories of the robots (Stranieri et al., 2013).

In this experiment, I use 18 TAMs, 20 e-puck robots, and a rectangular arena with dimensions of 2.7 m×2.2 m. Six instances of $\tau_{response}$ are placed in the arena, again configured in groups of three TAMs as shown in Figure 6.3. At the beginning of an experimental run, 20 e-puck robots are randomly positioned in the arena. An experimental run terminates after 5 min. Figure 6.7 shows a snapshot that illustrates the setup of the arena and the position of the TAMs.

Table 6.2: Detailed results of the multi-instance experiment. In this experiment multiple executions of the different instances of $\tau_{response}$ are possible: $s_x$, number of successful executions of $\tau_x$; $f_x$, number of failed executions of $\tau_x$.

| Run | $\tau_{response}$ | | $\tau_{left}$ | | $\tau_{right}$ | | $\tau_{repair}$ | |
|---|---|---|---|---|---|---|---|---|
| | $s_x$ | $f_x$ | $s_x$ | $f_x$ | $s_x$ | $f_x$ | $s_x$ | $f_x$ |
| 1 | 9 | 1 | 13 | 1 | 14 | 0 | 9 | 0 |
| 2 | 12 | 0 | 15 | 0 | 15 | 0 | 12 | 0 |
| 3 | 11 | 1 | 15 | 1 | 16 | 0 | 11 | 0 |

### 6.4.1 Results

I performed three experimental runs. Contrary to the first experiment, multiple executions of the different instances of $\tau_{response}$ are possible in this experiment. Accordingly, I report the following task-related metrics of the recorded data: $s_x$ is the number of successful executions of $\tau_x$, whereas $f_x$ is the number of failed executions. Task failures are due to robots abandoning a task because of sensor noise or other technical problems. Table 6.2 reports these task-related metrics for each of the three experimental runs. The discrepancy between the number of successful tasks $s_{left}$ and $s_{right}$ compared to $s_{repair}$ is due to the fact that some executions of $\tau_{repair}$ were prematurely terminated by the end of the experiment. This effect would be less pronounced in experiments with a longer duration.

Additionally to the metrics related to the success and failure of task execution, I report some of the metrics considered in the single-instance experiment. In particular, I report the metrics $a$, $w$, and $w'$ in Table 6.3. As the underlying distributions of these metrics are not known, I report each metric using a non-parametric approach. More precisely, I report the $1^{st}$, $25^{th}$, $50^{th}$, $75^{th}$, and $99^{th}$ percentile of the distribution of each metric, for all runs combined.

I also recorded the positions of the robots using a ceiling-mounted tracking system (Stranieri et al., 2013). Figure 6.8 shows this data correlated with the data from the TAMs recorded by the centralized coordinator. In the figure, black semi-opaque dots represent robot positions at a given moment in time, whereas the resulting lines show robot trajectories over time. Darker areas are caused by many overlaid

Table 6.3: Detailed results of the multi-instance experiment in the form of a non-parametric analysis of the following metrics: $a$, time from the moment the TAM $\tau_x$ signals the availability of a task to the moment a robot is inside the TAM; $w$ and $w'$, time a robot has to wait before and after working on $\tau_x$. I report the 1st, 25th, 50th, 75th, and 99th percentile of the distribution of each metric, for all runs combined. All times are reported in seconds.

| Task | Metric | 1% | 25% | 50% | 75% | 99% |
|---|---|---|---|---|---|---|
| $\tau_{left}$ | $a$ | 2.7 | 14.7 | 30.0 | 44.2 | 112.4 |
| | $w$ | 1.0 | 5.0 | 16.0 | 28.0 | 64.2 |
| | $w'$ | 4.7 | 12.0 | 21.0 | 39.0 | 70.7 |
| $\tau_{right}$ | $a$ | 7.4 | 20.0 | 29.0 | 40.0 | 89.9 |
| | $w$ | 1.0 | 5.5 | 11.0 | 26.0 | 57.9 |
| | $w'$ | 4.7 | 12.0 | 21.0 | 39.0 | 70.7 |
| $\tau_{repair}$ | $a$ | 4.7 | 12.0 | 21.0 | 39.0 | 70.7 |
| | $w$ | 4.4 | 5.0 | 6.0 | 11.0 | 20.8 |

dots, which represents areas frequently occupied by the robots. Comparing Figure 6.8 with Figure 6.7, we can observe a strong concentration of movement around the positions of the TAMs. Additionally, I visually indicate the success and failure of task execution, as reported by Table 6.2 (visualized using circles at the top and bottom of the plot).

### 6.4.2 Discussion

Extending upon the single-instance experiment, the multi-instance experiment demonstrates how the researcher can leverage the control framework to conduct experiments involving larger swarms.

In terms of data collection, the experiment records the same data, but for many robots and several task instances, thereby illustrating how the researcher can record large amounts of data over several task instances and/or experiments. The experiment also demonstrates how the centralized recording of data allows researchers to correlate data from the TAMs with the data of an external tracking system—if necessary, in real time.

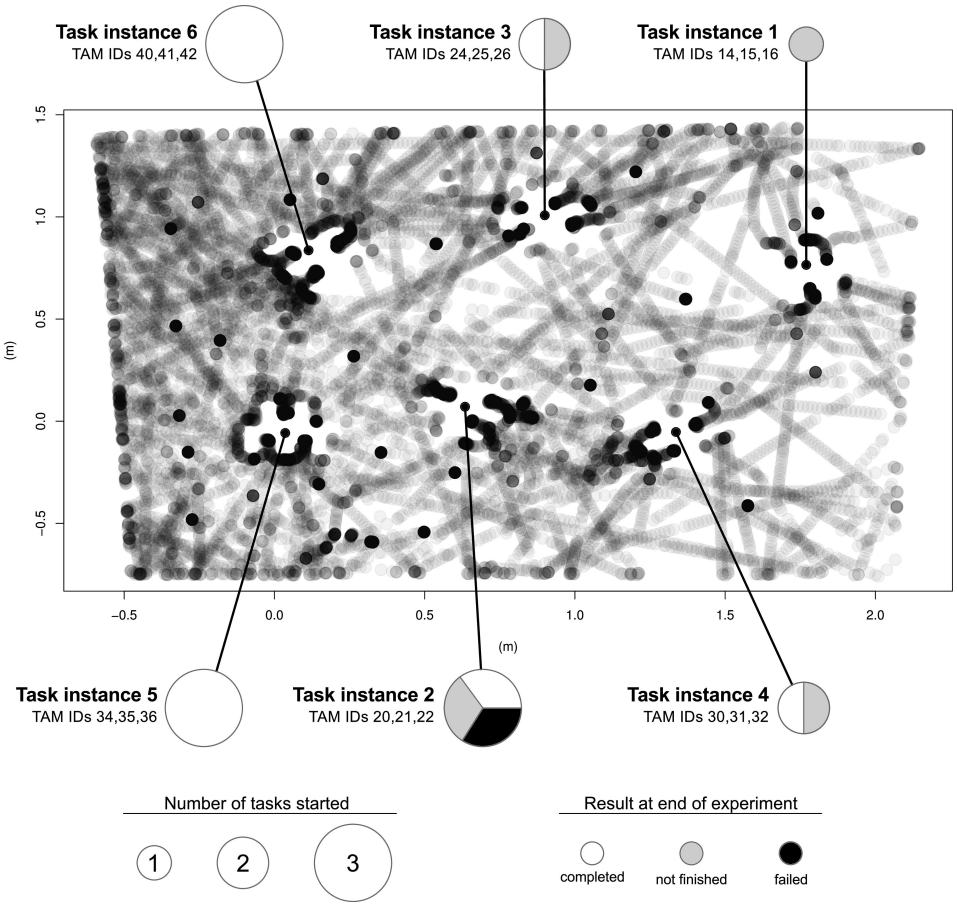



Figure 6.8: Data of a ceiling-mounted tracking system (Stranieri et al., 2013) correlated with data from the TAMs recorded by the centralized coordinator during the first experimental run of the multi-instance experiment. Black semi-opaque dots represent robot positions at a given moment in time. Resulting lines show robot trajectories over time. Darker areas represent areas frequently occupied by robots (e.g., the areas close to the TAMs). Circles represent task executions: their color corresponds to the task executed, while their diameter corresponds to the number of tasks executed over the course of the experiment.

In terms of experimental setup, the experiment demonstrates how to deploy several instances of the same complex task. The experiment also illustrates how the centralized coordinator can control a large number of TAMs in parallel, using the scalable mesh network to communicate with each TAM.

In summary, the experiment shows how the proposed tools for task abstraction can be leveraged to conduct experiments involving larger swarms and many interrelated tasks.

## 6.5 Summary

In this chapter, I demonstrated the usage of the tools proposed in this dissertation. In particular, I showed how a complex task can be modeled and abstracted using the approach presented in Chapter 4, how this task can be represented in an experiment using groups of TAMs, and how the control framework can be leveraged to conduct an experiment involving larger swarms and many interrelated tasks.

The experiments presented were proof-of-concept experiments, that is, I conducted them for the sole purpose of demonstrating the TAM and its capabilities. In the following Chapter 7, I present four scientific studies that rely on the TAM for task abstraction and representation.

Chapter 7

# Exemplary studies conducted using the TAM

In this chapter, I discuss four scientific studies that have been conducted using the tools presented in this dissertation. All four studies have been previously published; in the following description, I focus on the role of the TAM in each study and provide only a summary of the algorithms considered and results obtained. For further information, I refer the reader to the respective publication.

The presented studies have been performed in collaboration with other researchers; I therefore refer in the following to my collaborators and myself using the first person plural. My contribution to these publications ranges from providing the tools for task abstraction to designing and implementing the algorithms, performing the experiments, and writing the publication.

Note that of the four studies presented, the first and the third were conducted during the design phase of the TAM. This provided me with new insights into how tasks are studied in actual experiments—invaluable when designing a novel tool for research involving robots. As the physical TAM device did not exist in sufficient numbers at the time of the experiments, both studies were conducted solely in simulation. Furthermore, the TAM described in the associated publications differs slightly from the version presented in this dissertation. These differences are not significant in the context of the studies presented.

The second and the fourth study, on the other hand, were conducted after design and production of the TAM had been completed. The second study was conducted using physical TAM devices in robot exper-

iments. The fourth study is, at the time of writing, rather recent—as of yet, the experiments presented have been conducted solely in simulation. Robot experiments that involve physical TAM devices are in preparation at the time of writing.

In Section 7.5, I provide a summary of this chapter.

## 7.1 Cost and benefits of behavioral specialization

In this section, I present a study that considers atomic tasks of two different types that appear stochastically in time and space. The focus of the study is behavioral specialization: robots of a swarm have to specialize in one of the two task types in order to work efficiently. The study has been published (Brutschy et al., 2011, 2012c).

Division of labor is a concept that is common when considering the organization of large groups of individuals such as humans or social insects (Beshers and Fewell, 2001; Garnier et al., 2007). In division of labor, as defined for social insects by Beshers and Fewell (2001), "*(a) each worker specializes in a subset of the complete repertoire of task types performed by the colony, and (b) this subset varies across individual workers in the colony*". In artificial systems, a common way to obtain division of labor is to let individuals adapt their behavior so that they predominantly work on a subset of the available task types—this is called *behavioral specialization* (Nitschke et al., 2007). Behavioral specialization is known to increase the overall performance of an individual due to different reasons, one of them being learning. In some types of learning, an individual can acquire experience by repeatedly performing a task, which may improve the efficiency of the individual for tasks of the same type (Ratnieks and Anderson, 1999). The individual can exploit this increased efficiency by adapting its task selection behavior, that is, by selecting with a higher probability tasks of the type for which it has improved its performance.

### 7.1.1 Tasks and learning

We considered an environment in which robots can choose between two task types: blue tasks and green tasks, denoted by $\tau_x$ with $x \in \{b, g\}$. While performing task instances, robots learn. To implement *learning*,

we used a simple model where a robot that repeatedly performs a task of a certain type becomes more efficient in performing other tasks of the same type. The improvement in task performance is a reduction of the task completion time $d_x$, that is, the time it takes to complete a task of type $\tau_x$:

$$d_x(n_x) = \begin{cases} \overline{d}_{std} & \text{if } n_x = 0 \\ \overline{d}_{std} - \dfrac{\overline{d}_{std}}{k\,(1 + e^{-n_x + c})} & \text{if } 0 < n_x \leq n_{max} \end{cases} . \tag{7.1}$$

The meaning and effect of the parameter $k$ and the constant $c$ will be explained in the following. The counter $n_x$ is incremented on the completion of a task of type $\tau_x$, while, at the same time, the opposing counter $n_y$ for task type $\tau_y$, with $y \neq x$, is decremented (this is a form of forgetting; see below). Both counters are limited to the interval $[0, n_{max}]$. For example, if a robot has exclusively worked on tasks of type $\tau_b$ its counters are $n_b = n_{max}$ and $n_g = 0$.

The factor $k$ is used to vary the maximal time gain attainable through learning. This gain of learning is reached after a robot has successively completed $n_{max}$ tasks of the same type. For convenience, we refer to the resulting minimal task completion time attainable by a fully learned robot as $\overline{d}_{min} = d_x(n_{max})$. Note that the parameter $k$ is independent of $\overline{d}_{std}$, for example, $k = 1.25$ always results in a maximal time gain of 80%. The constant $c = n_{max}/2$ renders the function $d_x(n_x)$ point-symmetric on the median of the interval $[0, n_{max}]$, that is, a robot reaches 50% of the time gain attainable through learning after performing $n_x = n_{max}/2$ tasks of type $\tau_x$.

The standard task completion time is denoted with $\overline{d}_{std}$; it is the time a robot takes to perform a task $\tau_x$, when its $n_x = 0$. Figure 7.1 shows an illustration of the learning model.

We also implemented a form of *forgetting*: if a robot has improved its performance on a given task and then either starts to work on another type of task or does not work on any task for some time, it loses part of its performance improvement for the first task type. We implemented forgetting as follows. First, when improving its performance on a certain type of task $\tau_x$ due to learning, a robot forgets what it learned previously about the other task type, that is, upon incrementing $n_x$, we decrement $n_y$, with $y \neq x$. Second, a robot that keeps searching for tasks of a certain type gradually decreases its performance for both task types, that is, upon having travelled for a distance of 3 m,

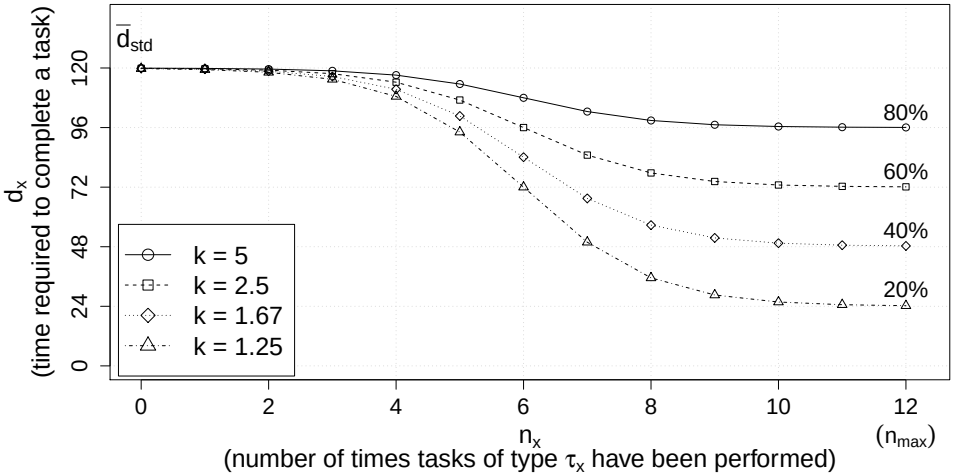



Figure 7.1: The effect of learning on the task completion time $d_x$ for different values of the parameter $k$. Learning takes effect when a robot repeatedly works on the same type of task. The standard task completion time in the initial state $\overline{d}_{std}$ is 120 s. The parameter $k$ influences the time gain of learning. The values $k = \{1.25, 1.67, 2.5, 5\}$ shown correspond to 20%, 40%, 60% and 80% of $\overline{d}_{std}$ at $n_{max}$, respectively.

the counters $n_b$ and $n_g$ are both decremented by 1 (to a minimum of 0). This mechanism causes the robots to return to their initial, non-specialized state over time.

### 7.1.2 Environment and approach

The environment consists of an obstacle-free, hexagonal arena—see Figure 7.2 for an illustration. We represented tasks with TAMs located at the boundaries of the arena, with a total of 24 instances concurrently available. Each TAM stochastically selected which type of task it represents, that is, task instances of either type appeared stochastically in time and space. Once a task instance has been completed, the coordinator assigns a new task of random type to the respective TAM. TAMs signalled the type of task they represented using their LEDs. At the beginning of the experiment, 18 e-puck robots were randomly positioned in the arena.

Robots are able to exploit learning by adapting their task selection behavior, that is, by selecting tasks of the type on which they have improved their performance with higher probability. This adaptation

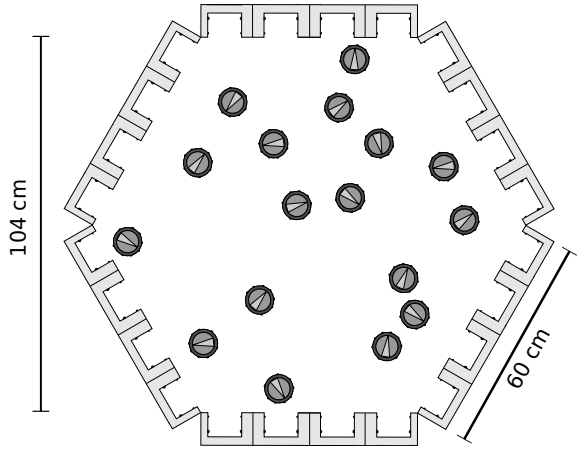



Figure 7.2: Illustration of the arena with e-pucks at random initial positions. Tasks are represented by TAMs located at the boundaries of the arena, with a total of 24 instances concurrently available. Each TAM stochastically selects which type of task it represents, that is, task instances of either type appear stochastically in time and space.

of behavior is called behavioral specialization. We employed a simple stochastic task allocation strategy, called *selective* strategy, which allows a swarm of robots to specialize behaviorally. The selective strategy is fully distributed and requires no communication between robots, as it depends only on the robots' memory of the previously completed tasks. In order to evaluate the influence of learning, we compared the selective strategy to a simple *greedy* strategy. The greedy strategy causes robots to work on any task they encounter; robots using the greedy strategy do not learn.

### 7.1.3 Results and discussion

We studied the influence of different environmental parameters on the performance of the swarm and showed that the swarm can exploit learning successfully (Brutschy et al., 2011, 2012c). In particular, we compared the performance of a swarm using the selective strategy with a swarm using the greedy strategy. Figure 7.3 shows the performance of both strategies when varying two parameters: the minimal task completion time, which affects the performance advantage attainable through learning; and the search speed, which corresponds to changing the size of the environment and therefore the distance between tasks.

The results show that the selective strategy performs better in relatively small environments where learning is advantageous. However, the results also show that spatiality has a major influence on the costs of specialization: robots specialized in a certain task are prone to losing efficiency due to longer search times—an effect that is especially pronounced in large environments. Behavioral specialization is therefore not to be considered in terms of benefits only, as it is affected by external factors such as task availability and the spatial distribution of the tasks, which might lower its benefits considerably.

The study demonstrates how to conduct experiments that involve atomic tasks represented by the TAM. Task availability is stochastic and follows a given distribution—the nature of the TAM allows researchers to closely control and conveniently modify this distribution. Furthermore, the study considers a problem that could not be studied with simple *ad hoc* abstractions: in order to study specialization, tasks have to exhibit completion times that depend on the individual robot. While this is trivial to represent in simulation, representing it in robot experiments is considerably more difficult. Accordingly, the study prompted me to include the required capabilities in the final design of the TAM. Note that the study presented was conducted in simulation only, as the physical TAM device did not exist in sufficient numbers at the time of conducting the experiments.

## 7.2 Property-driven design

In this section, I present a study that considers a same-robot foraging task with nestedness of 1, that is, the overall task is a complex task that consists of two atomic subtasks with a sequential interrelationship. The focus of the study is property-driven design, a novel top-down design method for robot swarms based on prescriptive modeling and model checking. The study has been published (Brambilla et al., 2014).

In swarm robotics, the developer needs to design the behavior of the individual robots so that their interaction will result in the collective-level behavior that is needed to accomplish a certain task. Unfortunately, the design and development of individual-level behaviors to obtain a desired swarm-level goal is, in general, very difficult, as it is difficult to predict and thus design the non-linear interactions of tens

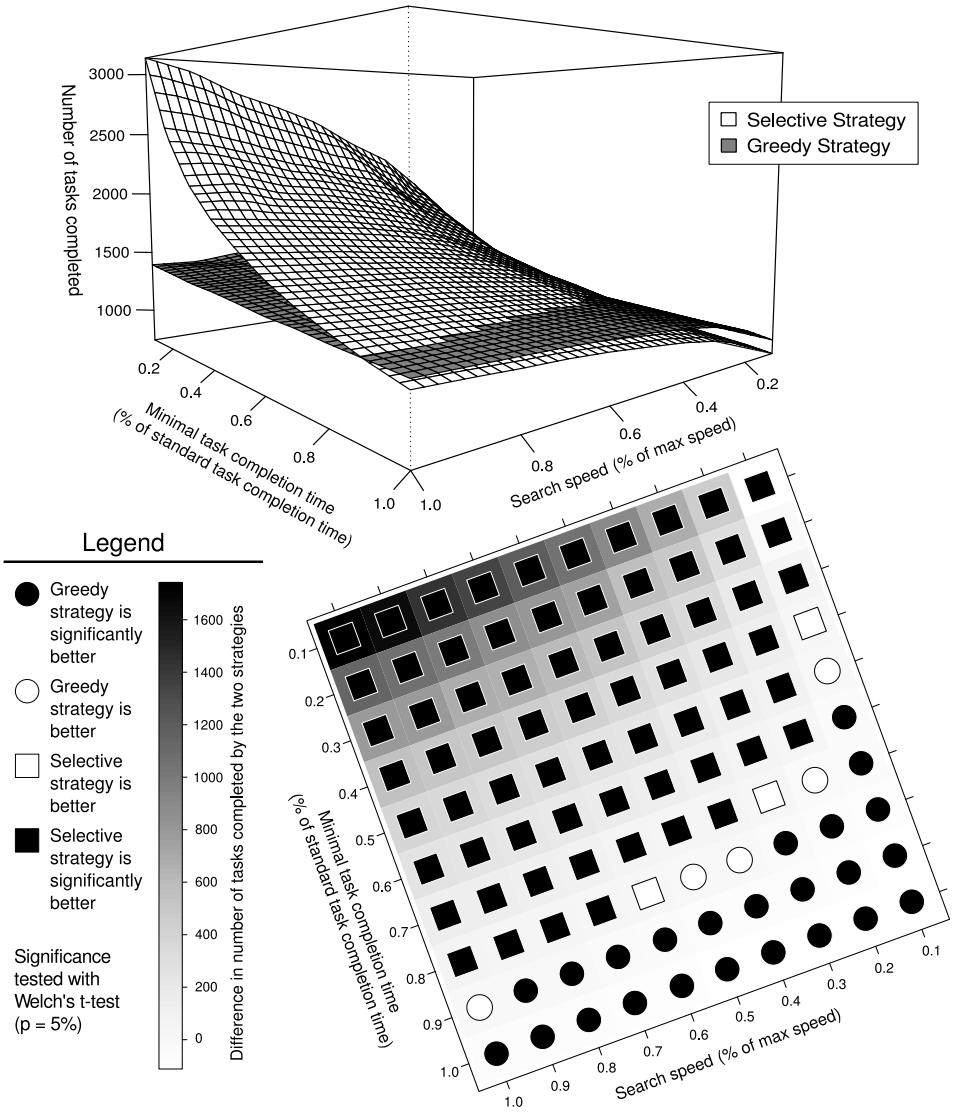



Figure 7.3: Performance for different search speeds and task completion times when fully specialized, collected over 20 simulation runs. At the top, the observed mean of the number of completed tasks for the selective and greedy strategy (white and dark surface, respectively); standard deviation $< 5\%$ for all tested cases (not shown). At the bottom: difference in number of tasks completed by the two strategies (shades of gray), with indication of which strategy is better and whether the difference is statistically significant or not (see symbols in the legend).

or hundreds of individual robots that result in a desired collective behavior. The difficulty in predicting and designing such interactions and the lack of a centralized controller make traditional system engineering approaches ineffective (Wooldridge and Jennings, 1998; Banzhaf and Pillay, 2007).

Existing approaches to the design of robot swarms present limitations (Brambilla et al., 2013); to date, an effective approach to the top-down design of robot swarms is still missing (Brambilla et al., 2014). In Brambilla et al. (2014), we presented property-driven design, a novel top-down design method for robot swarms based on prescriptive modeling and model checking. We evaluated the method in a case study concerning a foraging task represented with TAMs in robot experiments.

### 7.2.1 Model checking and property-driven design

In our approach, the developer creates a prescriptive model of the desired robot swarm and uses it as a blueprint for the implementation and improvement of the swarm. The use of model checking allows the developer to formally verify properties directly on the model, reducing the need for costly evaluations using simulation or robot experiments. In property-driven design, different "views" of the system to be realized are produced, from the most abstract (the properties of the system) to the most concrete (the final robot swarm). This idea takes its inspiration from model-driven design in software engineering, where software is designed through a series of transformations from platform-independent models to executable platform-specific models (Miller and Mukerji, 2003).

Property-driven design addresses the shortcomings of the existing approaches:

- it aims at providing a method that allows researchers to formally specify the requirements of the desired robot swarm;
- it reduces the risk of developing the "wrong" robot swarm, that is, a robot swarm that does not satisfy the requirements;
- it promotes the re-use of available models and tested solutions;
- it can be used to develop platform-independent models that help in identifying the best robotic platform to use;

- it helps to shift the focus of the development process from implementation to design.

Property-driven design is intended to be a step forward in the development of *swarm engineering*: the systematic application of scientific and technical knowledge in order to specify requirements, design, realize, verify, validate, operate and maintain an artificial swarm intelligence system (Brambilla et al., 2013).

Practically, property-driven design is based on prescriptive modeling and model checking. Model checking is a formal method that allows researchers to formally prove that a model satisfies a given property. The idea is that a system can be modeled using a formal mathematical model and then checked against a property defined using a formal logic language. We use Markov chains to define models and probabilistic temporal logics to define properties (Konur et al., 2012; Brambilla et al., 2012). Property-driven design is composed of four phases:

1. the requirements of the robot swarm are formally described in the form of desired properties using probabilistic temporal logics;
2. a prescriptive model of the robot swarm is created using Markov chains;
3. this prescriptive model is used as a blueprint to implement and improve a simulated version of the desired robot swarm;
4. the final robot swarm is implemented.

A schema showing the different phases of property-driven design is presented in Figure 7.4.

### 7.2.2 Case study

In order to evaluate our approach, we considered a case study in which the swarm has to address a foraging task.[26] In particular, we considered a same-robot foraging task: a single robot has to harvest and object and store it in the nest. Figure 4.9 in Chapter 4 shows the low-level model for this type of task.

[26] In the original publication, we also conducted a case study that considers a spatially organizing behavior. Behaviors of this kind cannot be studied using the TAM (cf. Chapter 3); accordingly, this case study has not been included in this dissertation.

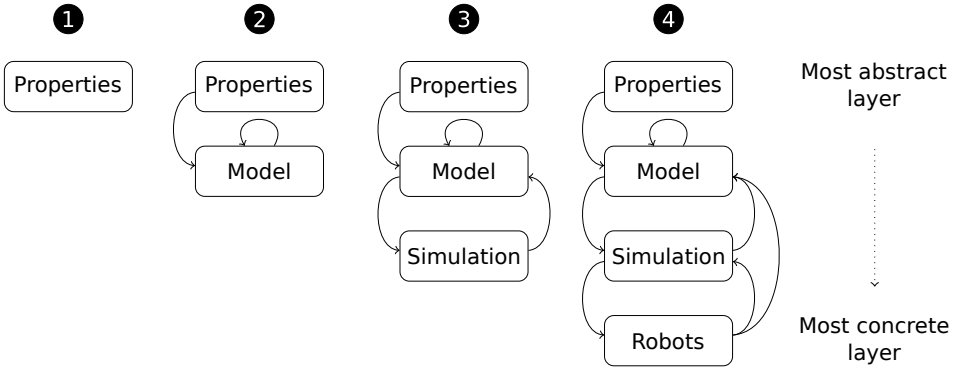



Figure 7.4: The four phases of property-driven design. In each phase, a new layer is added to the system. Layers differ in their level of abstraction: the Properties layer is the most abstract, in which only the goal characteristics of the robot swarm are stated; the Robots layer is the most concrete. The addition of a new layer brings the system closer to its final state.

In the case study, a swarm of robots has to harvest objects from sources and store them in the nest. The arena is comprised of 20 TAMs: 5 TAMs at the upper wall act as the nest, that is, each of these TAMs is a storing location; 15 TAMs at the other walls act as sources, that is, locations where objects can be harvested. The number of available storing locations depends on the number of robots currently storing an object: it can vary from 5, when no storing location is used by any robot, to 0, when all are in use. At any given time, there are $O$ objects available in the arena, that is, a new object appears as soon as one is harvested by a robot.

A TAM encodes its states using its LEDs: green, when it is available for storage; blue, when it has an object available for harvesting; red, when it is busy, that is, a robot is currently harvesting or storing an object in it; off/black, when it is unavailable. The environment is a $2\,\mathrm{m} \times 2\,\mathrm{m}$ square arena. Figure 7.5 shows a snapshot recorded during the experiments that illustrates the layout of the arena.

### 7.2.3 Results and discussion

We applied the four-phase process illustrated in Figure 7.4 in order to generate a controller for the robots. In phase one, we modeled the desired property of the swarm. In foraging, the main requirement is

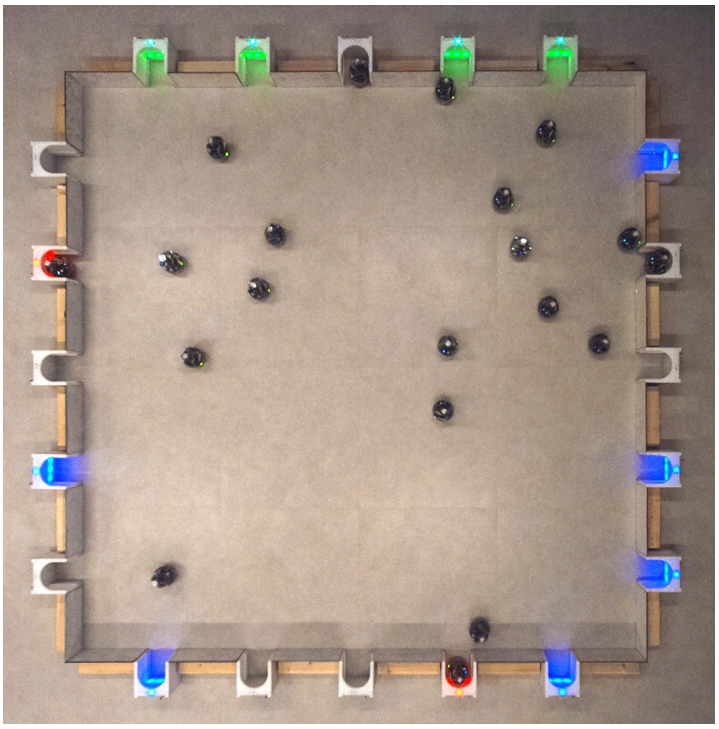


Figure 7.5: Snapshot recorded during an experiment performed with 20 e-puck robots and $O = 6$ available objects. The snapshot was taken with an overhead camera. Green colored TAMs signal storage locations, blue colored TAMs signal objects to be taken, dark TAMs are not available.

that the swarm retrieves at least a certain number of objects within a fixed time. Accordingly, the desired property of the swarm is that the expected number of objects retrieved in less than 600 s is greater than or equal to $k$. The number $k$ of objects that we wish to retrieve depends on the number of robots composing the swarm and on the number of objects $O$ available in the environment at any given time. In phase two, we modeled the swarm using continuous-time Markov chains (Serfozo, 1979), which allowed us to model the duration of actions such as harvesting and storing an object.

In phase three, we performed simulation experiments using a controller that implemented the prescriptive model created in phase two. We considered various swarm sizes $N = 10, 20, 50, 100$ and various numbers $O \in \{2, 4, 6, 8, 10\}$ of available objects, performing 100 experimental runs for every experimental setting. The results show that the

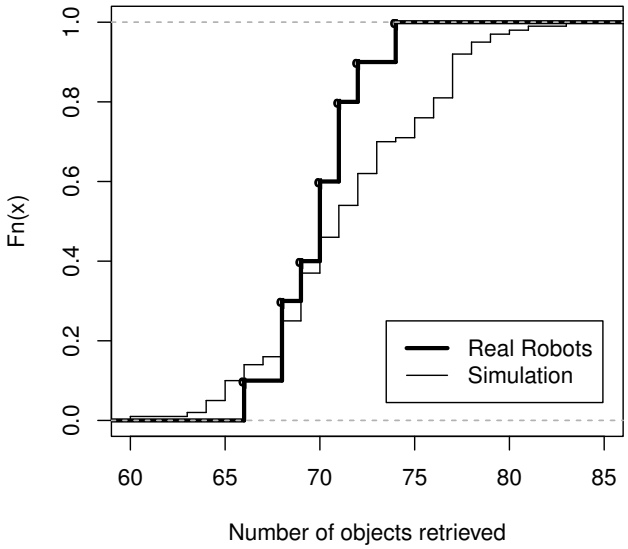



Figure 7.6: A graph showing the empirical cumulative distribution $Fn(x)$ of the number of object retrieved using robots (10 runs) and in simulation (100 runs). Both experiments were performed using 20 e-puck robots and $O = 6$ available objects.

desired property described in phase one is satisfied in all cases (Brambilla et al., 2014). Furthermore, the correspondence between the results obtained from the prescriptive model and the ones obtained from the experiments shows that the model captures the behavior of the robot swarm qualitatively.

In phase four, we performed an experiment involving 20 e-puck robots and 20 TAM devices. We performed 10 experimental runs; see Figure 7.5 for a snapshot taken with an overhead camera. The results show that the robots satisfied the desired property as well. Figure 7.6 shows that the results obtained with real and simulated robots are quite similar.

In summary, the study shows that using property-driven design provides a well-structured method to develop a swarm that satisfies the desired properties. As such, property-driven design is an effective method for the design and development of robot swarms.

With respect to the TAM, the study illustrates how the TAM can be used to represent one of the most common tasks in the swarm robotics literature: single-robot foraging. In particular, the study shows how the LEDs of the TAM can be used to signal different task types and

their various internal states to the e-puck robots in the environment. Furthermore, the study demonstrates a case in which the coordinator centrally dispatches tasks: as there are $O$ objects in the environment at any given moment, new objects are made available by the coordinator at a random location once an object has been harvested. Of the four studies presented in this chapter, this is the only one that was performed using robot experiments involving physical TAM devices.

## 7.3 Autonomous task partitioning

In this section, I present a study that considers a bucket-brigading foraging task, that is, a complex task with a nestedness of 2. The focus of the study is autonomous task partitioning in a swarm of robots. The study has been published: a first version that focuses on an self-organizing approach (Frison et al., 2010; Pini et al., 2011b), followed by extensions that compare the this approach to established algorithms for the multi-armed bandit problem (Pini et al., 2012) and evaluate the influence of communication on algorithm performance (Pini et al., 2013b).

In biology, the term *task partitioning* refers to situations in which a given task is divided into two or more subtasks that can be performed separately (Jeanne, 1986). As such, task partitioning amounts to decomposing a task into smaller units of work that can be tackled separately. Task partitioning is a subset of the techniques presented in Chapter 4 insofar as the resulting subtasks have exclusively sequential interrelationships. The main body of research on the topic has been carried out in the field of entomology (Fowler and Robinson, 1979; Ratnieks and Anderson, 1999; Anderson and Ratnieks, 2000; Hart and Ratnieks, 2001).

The benefits of task partitioning in robot swarms are many: better exploitation of specialization, increased efficiency, and physical separation of the robots. However, task partitioning also entails costs due to overheads, which can subtract from these benefits. Task partitioning should therefore be used only when the benefits are higher than the costs.

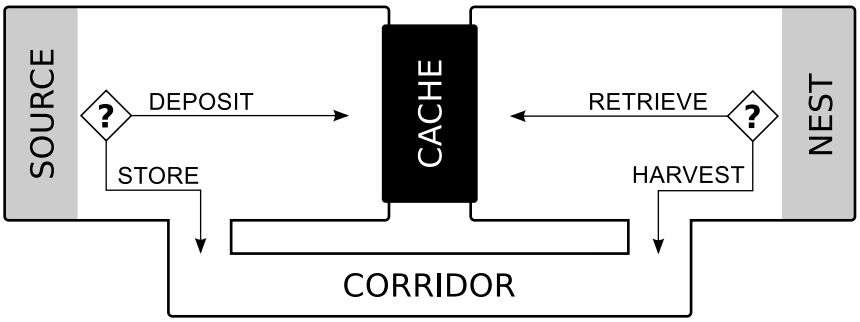



Figure 7.7: Illustration of the task partitioning problem. Robots harvest objects from the source and store them in the nest. The areas containing the source and the nest are physically separated by the cache. In order to travel between areas, robots have to use the corridor. Robots decide whether to use the cache or the corridor in two cases, represented by question marks: after harvesting an object from the source (left), and after storing an object in the nest (right).

### 7.3.1 Problem

In the studies, we concentrated on what we call the *task partitioning problem*: selecting between partitioning a given task into subtasks or performing it as a single piece of work—see Figure 7.7. In particular, the overall task $\tau_{foraging}$ that the swarm faces is a foraging task: robots have to harvest objects from a source and deposit it in the nest. The foraging task is a complex task that can decomposed in two ways.

One way is to decompose $\tau_{foraging}$ into two atomic subtasks with a sequential interrelationship: one subtask that consists in harvesting an object from the source, and one subtask that consists in storing the object in the nest. Similarly to the study presented in Section 7.2, both subtasks have to be performed by the same robot—see Figure 4.9 in Chapter 4 for a low-level model of this type of task. In order to travel from the source to the nest (or *vice versa*), a robot must use the corridor that connects the two areas, as shown in Figure 7.7. As a single robot completes the overall foraging task, we say that the task has not been partitioned, which we denote as $\tau_{nopart}$.

The second way is to decompose $\tau_{foraging}$ into two complex subtasks, again with a sequential interrelationship. The first complex subtask $\tau_{source}$ consists of two atomic subtasks: harvesting an object from the source, and depositing it in a cache. The second complex subtask $\tau_{nest}$ also consists of two atomic subtasks: retrieving an object from

the cache, and storing it in the nest. The overall task is therefore an instance of a bucket-brigading task—see Figure 4.12 in Chapter 4 for a high-level model for this type of task. Note that the complex subtasks have to be performed by different robots that work in isolated areas of the environment. These areas are separated by the cache, as shown in Figure 7.7. If two robots complete the overall foraging task, we say that these robots partition the task, which we denote as $\tau_{part}$.

Robots face a choice: to either partition the task or not, that is, to either perform $\tau_{part}$ or $\tau_{nopart}$. If a robot decides to perform $\tau_{part}$, it typically remains in one area of the environment, collaborating with the robots in the other area in order to complete the overall task. If a robot decides to perform $\tau_{nopart}$, it uses the corridor to travel continuously between source and nest. The two choices have different costs associated with them. The cost of $\tau_{part}$ depends on the cost at which objects can be exchanged at the cache. The cost of $\tau_{nopart}$ depends on the length of the corridor. Both choices are affected by additional overheads, for example, physical interference due to high robot densities.

We represent the source, the nest, and the cache using TAMs. The source and the nest are represented using four TAMs each. The cache consists of four TAMs on each side. Two TAMs, one on each side, form a cache slot: if an object is deposited on the harvest side, the TAM on the store side becomes available after a certain time-cost $\Pi$ by signalling the presence of an object. By modifying $\Pi$, we were able to render $\tau_{part}$ more or less advantageous over $\tau_{nopart}$—a modification that could be conveniently carried out via the centralized coordinator. Figure 7.8 illustrates the setup of the arena and the positioning of the TAMs.

### 7.3.2 Approach

The approach that we proposed relies on *cost estimation*: robots estimate the costs of $\tau_{part}$ and $\tau_{nopart}$ and decide accordingly. The overall goal of the approach is to maximize the number of objects delivered to the nest in a given time, which is equivalent to minimizing the time needed to deliver each object. Therefore, in our approach, we expressed cost as time.

We studied a total of three different algorithms. In Frison et al. (2010) and Pini et al. (2011b), we studied a so called *AdHoc* algorithm. Robots employing the AdHoc algorithm chose between $\tau_{part}$

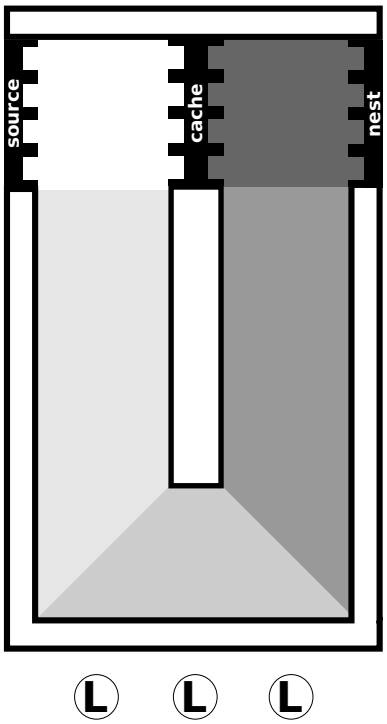



Figure 7.8: Illustration of the arena in which the robots perform foraging. The source, the nest, and each side of the cache are represented using four TAMs. Different areas in the arena are marked with a specific ground color that can be recognized by the robots. Three light sources (each marked with "L") are used by the robots as a directional clue for navigation.

and $\tau_{nopart}$ stochastically. After collecting an object from the source, a robot chooses $\tau_{part}$ and deposits the objects in the cache with a probability $P$. $P$ depends on the cost estimate for $\tau_{part}$ and a parameter that regulates the degree of exploration of the algorithm. Exploration consists in sampling less advantageous solutions in order to detect variations in the environment that possibly make these solutions more advantageous. After delivering an object to the nest, a robot retrieves the next one from the cache with the same probability $P$. Robots employing the AdHoc algorithm chose $\tau_{part}$ with a probability $P$ and $\tau_{nopart}$ with a probability $1 - P$.

In Pini et al. (2012) and Pini et al. (2013b), we compared the AdHoc algorithm with two established algorithms for the multi-armed bandit problem. The *$\epsilon$-Greedy* algorithm is a simple stochastic algorithm that has been applied in many contexts (Sutton and Barto, 1998). Robots employing the $\epsilon$-Greedy algorithm select the choice with the lowest associated cost with probability $1-\epsilon$ and a random choice with probability $\epsilon$. $\epsilon$ is the only parameter of the algorithm and defines the degree of exploration: the higher $\epsilon$, the higher the amount of exploration. The *UCB* algorithm is a heuristic adaptation of the UCB1 policy presented by Auer et al. (2002), which in turn was derived

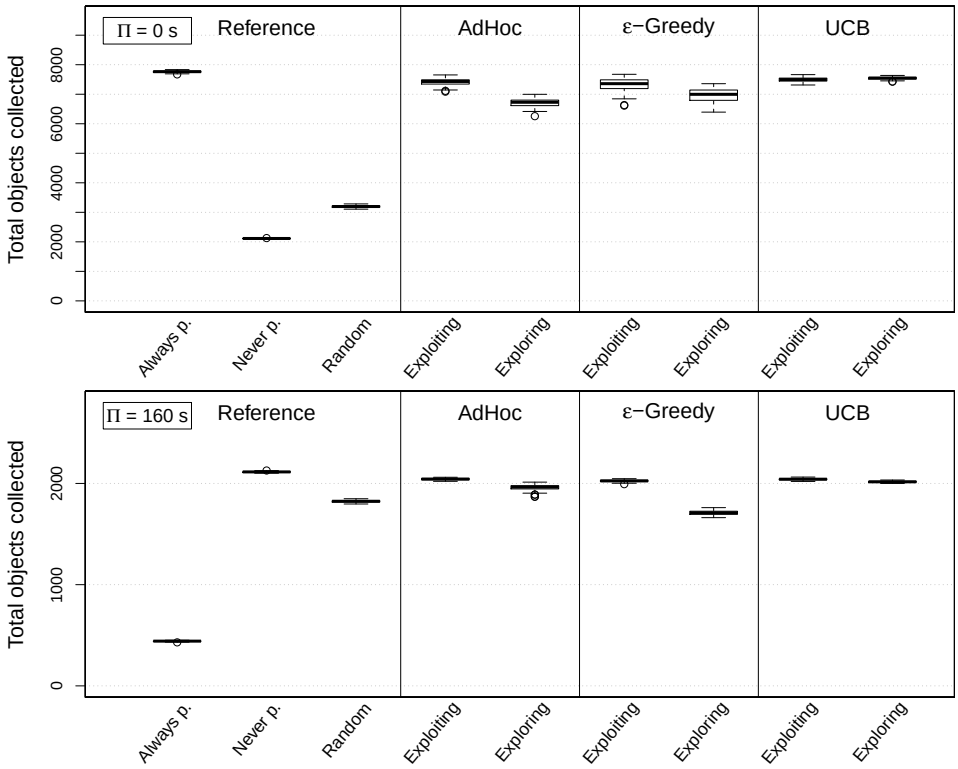



Figure 7.9: Performance of the studied algorithms, measured as objects delivered to the nest by the swarm, based on 15 repetitions per algorithm. At the top, results obtained if $\tau_{part}$ is advantageous ($\Pi = 0\,\mathrm{s}$); at the bottom, results obtained if $\tau_{nopart}$ is advantageous ($\Pi = 160\,\mathrm{s}$).

from an index-based policy proposed by Agrawal (1995). UCB1 is characterized by a rapid convergence because it was originally designed for stationary problems (Auer et al., 2002). Similarly to the algorithms previously mentioned, the degree of exploration in UCB is tunable using a parameter. Robots employing the UCB algorithm select the task to perform deterministically.

Additionally, we compared the algorithms with three reference algorithms: *always-partition*, which always performs $\tau_{part}$, *never-partition*, which always performs $\tau_{nopart}$, and *random*, which randomly switches between $\tau_{part}$ and $\tau_{nopart}$.

### 7.3.3 Results and discussion

Figure 7.9 shows the performance of the six algorithms for two different value for the time-cost $\Pi$ of using the cache: 0 s (top) and 160 s (bottom). We measured the performance of an algorithm as the total number of objects delivered to the nest by the swarm. For each algorithm, we tested two versions: one that is primarily exploiting, and one that is primarily exploring. The results confirm that, for $\Pi = 0$ s, the cache and therefore $\tau_{part}$ are advantageous. Accordingly, the best-performing algorithm was the always-partition algorithm. In case of $\Pi = 160$ s, the corridor and therefore $\tau_{nopart}$ are advantageous. Accordingly, the never-partition algorithm produced the best results. Furthermore, algorithms that are primarily exploiting have an advantage over exploring ones as the environment considered here is time-invariant. We also performed experiments with the aim of assessing the capability of the swarm to react to changes in the environmental conditions (see Pini et al., 2011b) and of measuring whether communicating cost estimates among robots is advantageous (see Pini et al., 2013b).

The study demonstrates the case in which the same overall task $\tau_{foraging}$ can be decomposed in two ways: either as bucket-brigading task $\tau_{part}$ with a nestedness of 2, or a simple foraging task $\tau_{nopart}$ with a nestedness of 1. This is related to the ambiguity of decomposing tasks mentioned in Chapter 4: both decompositions have a justification, depending on the context. It is therefore the responsibility of the researcher—or, in this case, of the robots—to select the appropriate choice for a given problem and environment.

Furthermore, the study demonstrates the flexibility of the TAM: it can represent tasks with varying parameters such as the cost $\Pi$ of using the cache. In particular, the TAM allowed us to closely control $\Pi$ during an experiment—a great advantage over using existing *ad hoc* abstractions, as changing the performance difference between $\tau_{part}$ and $\tau_{nopart}$ would require us to lengthen or shorten the corridor.

## 7.4 Temporal task allocation

In this section, I present a study of a robot swarm that has to perform task allocation in an environment that features periodic properties. The work presented has been published (Castillo-Cagigal et al., 2014).

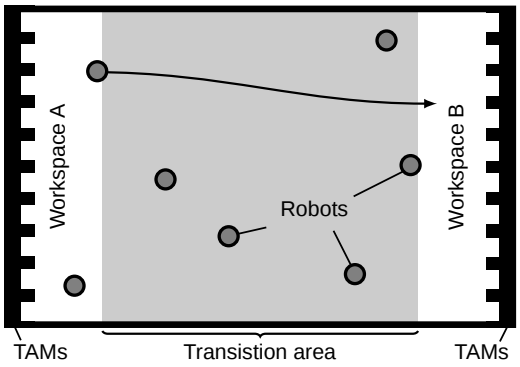



Figure 7.10: Illustration of the arena, with workspaces $A$ and $B$ in white and transition area in gray. 10 TAMs represent 10 single-robot tasks in each workspace.

In dynamical environments, real-time resource allocation commonly involves situations in which events occur periodically and with a certain frequency (Rosu et al., 1997). Periodicity can originate from both natural and artificial phenomena, for example, the rotation and revolution of the earth, tides, cyclic production processes, and customer demands. In artificial systems, the designer typically wishes to allocate resources so as to increase the system performance and achieve predefined goals (Martín H. et al., 2009). To this end, it is paramount that information on the nature of the periodic events involved is available during the design process (Liu and Picard, 1998).

We studied a case in which a robot swarm needs to perform task allocation in an environment with periodic properties. Specifically, the periodicity of the environment lies in the temporal pattern in which new tasks appear. To operate effectively, the swarm needs to reallocate its workforce according to the periodicity of the environment. We call *temporal task allocation* a task allocation that takes into account temporal properties of the environment.

### 7.4.1 Tasks with periodic interrelationships

We considered a rectangular environment divided in three areas: workspace $A$, workspace $B$, and a transition area. Tasks appear either in workspace $A$ or $B$, following a temporal pattern. Figure 7.10 shows an illustration of the environment. Robots have to travel from workspace to workspace to attend to tasks where they appear. The workspaces are separated by the transition area: a robot that moves from one workspace to the other has to cross the transition area. The time spent by a robot to cross the transition area is called *switching cost*.

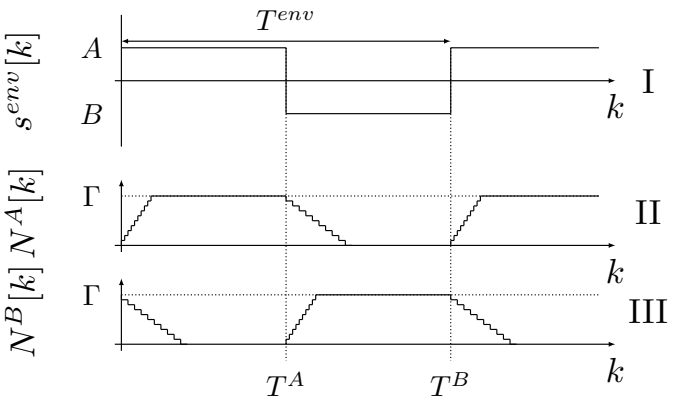



Figure 7.11: Environment period and location of task appearance: I) signal $s^{env}[k]$ of task appearance with period $T^{env}$, II) number $N^A[k]$ of tasks in workspace A, III) number $N^B[k]$ of tasks in workspace B.

Tasks in the environment are stationary single-robot tasks represented using TAMs. Robots spend 0.5 s working on a task. Tasks expire after 5 s: if a task remains unattended for longer than this time, it is removed from the environment. At time $k$, $N^A[k]$ and $N^B[k]$ are the number of tasks present in workspace $A$ and $B$, respectively. The number of tasks in each workspace is limited by the task capacity $\Gamma$, which is $\Gamma = 10$ for both workspaces.

The periodicity of the environment that we consider in this study lies in the temporal pattern with which tasks appear. During a period of time $T^A$, new tasks appear in workspace $A$. After the end of $T^A$, new tasks appear in workspace $B$ for a period of time $T^B$. After the completion of $T^B$, new tasks appear again in workspace $A$, and so on. The full cycle has a period $T^{env} = T^A + T^B$, and we assume $T^A = T^B$. The location of the appearance of tasks in the environment can be described as a square signal denoted by $s^{env}[k]$ that takes a value of either $A$ or $B$. An example of $T^{env}$ and $s^{env}[k]$ is shown in Figure 7.11-I.

Regardless of the workspace, tasks appear in the environment with a certain incoming task rate $\lambda$. If the task capacity $\Gamma$ of a workspace is reached, additional tasks are dismissed. When tasks cease to appear in a workspace, the number of tasks in this workspace decreases as tasks expire. This effect can be observed in Figure 7.11-II and 7.11-III for both workspaces: the number of tasks increases until $\Gamma$ is reached and decreases after new tasks cease to appear.

The robots move in the arena between workspace $A$ and $B$ in order to attend to the tasks. Robots act independently of each other, but are able to exchange simple messages via short-range line-of-sight communication. We call the number of robots in a workspace the *workforce* allocated to this workspace by the swarm. In order to maximize performance, the swarm needs to allocate its complete workforce to the workspace where tasks are available. To achieve this goal, the robots need to switch between workspaces so that their movement is synchronized with the temporal pattern of task appearance, performing a temporal task allocation.

### 7.4.2 Approach

We proposed a novel temporal task allocation algorithm that adapts to the environment. This algorithm is based on concepts that we borrowed from the signal processing and collective synchronization literature.

The goal of the algorithm, called CS for *collective synchronization*, is to synchronize the movement of the robots between workspaces with the appearance of tasks in the environment. In CS, each robot $i$ has an internal timer that governs its transitions between workspaces. This timer increases each time step and resets to zero when it reaches the period $T^i$ of robot $i$, thereby producing a square signal $s^i[k]$ that takes the values $A$ or $B$ as shown in Figure 7.12-I.

CS enables robots to synchronize their internal timer (and thereby their movements) with the environment in two steps. First, each robot $i$ evaluates the extent to which it is synchronized with the environment. This is measured by the fraction of time during which the robot finds tasks its current workspace. Second, each robot $i$ shifts its internal timer such that its square signal $s^i[k]$ matches the work signal $w^i[k]$ of successful task executions, thereby approximating the signal $s^{env}[k]$ of task appearance in the environment. Robot $i$ is fully synchronized if $T^i = T^{env}$ and the time difference $\Delta^i$ between its internal timer and task appearance is zero.

Additionally, CS features a visual communication protocol to speed up the synchronization process and avoid physical interference between robots, as indicated in Figure 7.12.

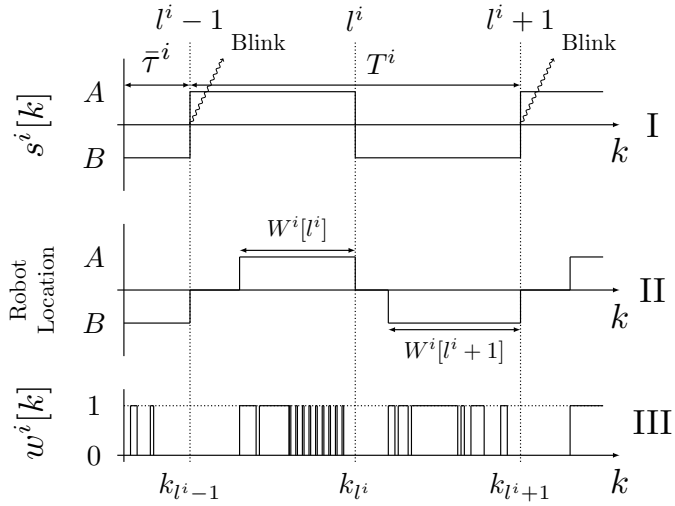



Figure 7.12: Example of robot operation: I) signal $s^i[k]$ of the internal timer of robot $i$; II) robot location in the environment and amount of time $W^i$ spent in a workspace (as opposed to transitioning between workspaces); III) work signal $w^i[k]$, that is, the signal of successful task executions by robot $i$.

### 7.4.3 Results and discussion

We conducted experiments in simulation using 20 TAMs and a swarm of six e-puck robots. Figures 7.13a and 7.13b report the periods and phases of the robots over the course of an experiment when using CS. We can observe that the period $T^i$ of every robot converges to $T^{env}$ and that all $\Delta^i$ converge to zero: the swarm successfully synchronizes with the signal $s^{env}[k]$ of task appearance.

We compared CS to two other algorithms: 1) an algorithm that uses internal timers for switching between areas but does not attempt to synchronize with the environment or with other robots, denoted NS, and 2) an algorithm that does not use internal timers but switches depending on task availability—a "greedy" algorithm, denoted GR. Figures 7.13c and 7.13d report and compare the performance of these three algorithms. From the results, we can conclude that a swarm using our algorithm outperforms the competing algorithms after an initial adaptation period.

The study illustrates how the TAM can be used to represent tasks with interrelationships that are not captured by the modeling approach presented in Chapter 4. In particular, the complex temporal pattern

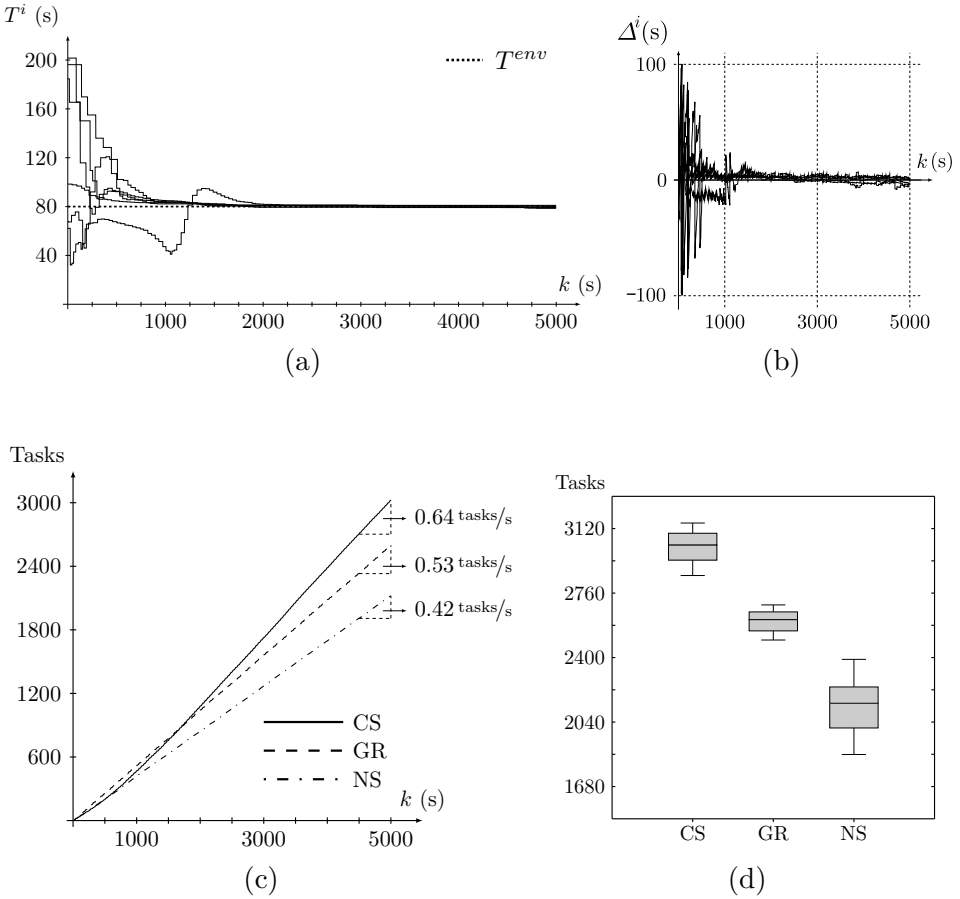



Figure 7.13: a) Periods $T^i$ of the robots compared to the period $T^{env}$ of task appearance; b) Time difference $\Delta^i$ between the internal timers and the task appearance; c) Average number of tasks performed during the experiments, and final performance rate, for all three algorithms; d) Box plot of the number of tasks performed by each algorithm during the experiment, based on 15 repetitions per algorithm. In the plots, *CS* denotes the proposed algorithm, *GR* denotes an algorithm that switches depending on task availability without using an internal timer, and *NS* denotes an algorithm that does not attempt to synchronize its internal timer with the environment.

at which tasks appear in the environment cannot be modeled using the proposed modeling approach. However, the TAM is still capable of representing these single-robot tasks in an experiment: task parameters such as their availability can be conveniently controlled using the centralized coordinator. This allows groups of TAMs to exhibit complex coordinated behavior, for example, as shown here, by following a temporal pattern.

## 7.5 Summary

In this chapter, I presented four scientific studies that have been conducted using the tools presented in this dissertation. The studies serve as a demonstration of the variety of tasks that can be abstracted with the proposed modeling approach and subsequently represented with the TAM. Furthermore, the studies illustrate the benefits of using the centralized control framework for conducting robot experiments. Finally, the studies demonstrate some of the limitations of the proposed tools.

The first study concerned behavioral specialization of robots in one of two task types. The second study concerned property-driven design, a novel top-down design method for robot swarms based on prescriptive modeling and model checking. The third study concerned collective decision-making in a task partitioning scenario. The fourth study concerned a new approach to task allocation in environments that exhibit temporal properties.

None of these studies focused on task execution. Instead, each focused on the group dynamics and collective processes of the swarm. Therefore, these studies could be conducted using an abstract task representation such as the TAM. Conducting them without the TAM would certainly have been possible, but would have required different robots and/or different—possibly tailor-made—task representations. The resulting experimental setup would have incurred a substantial increase in cost without yielding different or qualitatively better results. I speculate that, without the TAM, it would be prohibitively costly to conduct these studies in laboratory experiments using robots.

In the following Chapter 8, I conclude the dissertation by discussing its main contribution and perspectives for future research.

Chapter 8

# Conclusions

In this chapter, I conclude the dissertation by summarizing its main contributions. Furthermore, I briefly discuss perspectives for future research.

## Summary of contributions

The core premise of this dissertation is that existing studies in swarm robotics are limited by the fact that researchers use *ad hoc* solutions for abstracting and representing tasks. As a result, most studies in swarm robotics consider only simple tasks. The primary contribution of this dissertation is to overcome this limitation by providing novel conceptual and practical tools for task abstraction.

On the conceptual side, I first defined task abstraction and its use in swarm robotics research. I reviewed various *ad hoc* solutions used in the literature for task abstraction and representation. Furthermore, I discussed the advantages and disadvantages of these solutions. The lessons that I took from the discussion fueled the design of the tools proposed in this dissertation.

I then presented the tools that lie at the core of this dissertation: a novel approach for abstracting complex tasks and a generic device for representing such tasks in laboratory experiments, called the TAM. The TAM is generic in the sense that it can represent any task that requires a single robot to remain for a given amount of time at a specific location and at a specific moment in time. The TAM can be considered a task emulator that operates at an intermediate level of

abstraction between the task abstractions used in simulation and *ad hoc* task abstractions employed in robot experiments.

A single TAM is limited to representing single-robot tasks; complex tasks first have to be abstracted before a group of TAMs can be used to represent them. To this end, I proposed a novel approach to model complex tasks as a set of single-robot subtasks and their interrelationships. In order to demonstrate the flexibility of the approach, I reviewed the swarm robotics literature and applied the approach to the various tasks considered. This review also allowed me to substantiate the basic premise of this dissertation, namely that the complexity of the tasks considered in the literature is rather limited.

On the practical side, I first presented the design and implementation of the TAM device. In particular, I discussed the goals that stand behind the design and how I attained them in the implementation. Furthermore, I presented a centralized control framework that allows researchers to control a group of TAMs so that they can implement the interrelationships identified by the modeling approach. The control framework also provides a reliable infrastructure for conducting experiments that involve large numbers of robots. By using this infrastructure, researchers can conduct experiments conveniently at a fraction of the cost of experiments that rely on *ad hoc* solutions for task abstraction.

I presented several experiments and studies that employ the proposed tools. In a proof-of-concept experiment, I demonstrated the usage of the proposed tools from beginning to end: how to abstract a complex task using the modeling approach presented, how to represent the resulting model in laboratory experiments using the TAM, how to conduct an experiment using a large swarm of robots by relying on the infrastructure provided by the TAM, and how to collect and analyze the task-related data in experiments involving 20 e-puck robots. Furthermore, I presented four scientific studies that consider a variety of problems ranging from property-driven design for robot swarms to self-organized task allocation and task partitioning. Each of these studies employed the TAM for task abstraction, thereby demonstrating the variety of possible applications for the TAM.

In conclusion, the main contribution of this dissertation are conceptual and practical tools that enable the study of complex tasks with swarms of robots. The primary advantage of these tools is cost re-

duction: as the limitation of swarm robotics experiments to simple tasks is primarily due to cost, these tools enable research on problems that was previously confined to simulation. Finally, I would like to remark that the TAM can be extended with additional functionalities or adapted to other robot platforms. Furthermore, the TAM can be used apart from the modeling approach presented in this dissertation. As such, I believe that the research community will benefit greatly from the TAM.

## Perspectives

Possible directions for future research on the tools presented in this dissertation can be classified, just as the tools themselves, into research on the conceptual level and on the practical level.

Future research on the conceptual level concerns mostly the abstract concept of the TAM device and the presented modelling approach. For instance, the concept of the TAM could be further generalized by making the task representation device itself mobile. This would allow researchers to study complex tasks that exhibit spatial dynamics for example. Regarding the presented modelling approach, a possible direction for future work is the inclusion of further concepts (e.g., conditionals) into the high-level model. Finally, the proposed modelling approach could be related to existing design approaches that work on several levels of complexity. For example, electronic design automation (EDA) uses similar approaches to model the design of integrated circuits on an ever-increasing level of abstraction (Sangiovanni-Vincentelli, 2003).

Future research on the practical level concerns the presented implementations of the TAM and the control framework. Regarding the TAM, possible additional features include a power plug that allows researchers to use the TAM without a battery, a boot loader that enables firmware updates over the mesh network, and other task-specific extensions. Furthermore, a possible direction for future work is to adapt the presented implementation of the TAM to other robot platforms. Generalizing this direction, a future implementation of the TAM could be adapted to multiple robot platform "on the fly" using pluggable sensors. Regarding the control framework, a current limitation is that researchers have to manually describe the mapping between TAMs, atomic subtasks, and the location of the represented task. An useful extension of the control framework would therefore

be the ability to track the location of TAMs in the environment and assign tasks automatically to these TAMs based on their location.

The tools presented in this dissertation enable researchers to study new classes of problems, whose study in physical robot experiments would have presented considerable difficulty using *ad hoc* solutions for task abstraction. Therefore, a large body of work previously studied solely in simulation can now be evaluated in physical experiments. For example, many approaches intended for agent-based systems can now be ported to swarm robotics systems and studied in experiments involving physical robots.

In terms of self-organized methods for task allocation—the topic that I initially set out to study—many opportunities for future research are brought forward by the tools presented in this dissertation. In particular, the question of allocating swarms of robots to tasks with many subtasks and various interrelationships has received very little attention in the swarm robotics literature until now. For example, to date, there are no self-organized methods to allocate a robot swarm to tasks as complex as those presented in Chapter 6 or as those proposed by the Swarm-bots project (e.g., Nouyan et al., 2009) or the Swarmanoid project (Dorigo et al., 2013).

A possible direction for future research is to investigate methods that rely on measuring waiting times at the point at which two subtasks join. The study that I presented in Brutschy et al. (2014c) considers such a method: robots base their decision to switch subtasks on the performance of the group working on this subtask. The study is limited to a problem instance with two subtasks; however, the approach is sufficiently promising to warrant the investigation of its applicability to tasks with several subtasks and/or other interrelationships—thanks to the TAM, these tasks can now be represented in laboratory experiments. Preliminary studies on an agent-based model indicate that the method can indeed be applied to such problems. A first direction for future work is therefore to study an instance of the problem that exhibits many subtasks using physical robots and the TAM.

Methods for different varieties of interrelationships warrant further investigation as well. The method that I presented in Brutschy et al. (2014c) has been tested exclusively on subtasks with a blocking sequential interrelationship, that is, an interrelationship where each subtask has to wait for its predecessor to finish before it can start. A promising direction are methods for task allocation that are sufficiently flexible for tasks with several types of interrelationships, for example, block-

ing ones as well as non-blocking ones, as commonly used in bucket-brigading (see, e.g., Pini et al., 2014).

Other directions for future research outside the domain of task allocation are studies that consider tasks with specific aspects. The tools proposed in this dissertation allow researchers to closely control the aspects of each task in an experiment. This capability enables research on tasks that, for example, are available following certain distributions, exhibit different behaviors for different robots, or provide feedback to the robots. For instance, studies in reinforcement learning can provide positive and negative feedback to the robots directly through the TAM.

Appendix A

# Tools and materials

In this section, I present the tools and materials common to all experiments presented in this dissertation. More specifically, I present the e-puck robot and the ARGoS simulation framework.

## A.1 Robotic hardware: The e-puck

The e-puck (Mondada et al., 2009) is a mobile robot designed for educational and research purposes at EPFL.[27] The e-puck is of cylindrical shape with a diameter of 0.75 cm. The body of the e-puck does not possess any protrusions, which is advantageous when avoiding obstacles. The e-puck is controlled by a dsPIC 30 micro-controller running at 30 MHz, and powered by an exchangeable Lithium-Ion battery. The capabilities of the e-puck can be augmented with various extension boards, some of which are detailed in Section A.1.3. Its small, compact, and extensible nature, and relatively cheap (Mondada et al., 2009), makes the e-puck well suited for swarm robotics research. Figure A.1 illustrates the e-puck in two different configurations: the basic model, equipped with only the sound extension, and a heavily extended one.

### A.1.1 Sensors

The e-puck features the following sensors:

[27] Ecole Polytechnique Fédérale de Lausanne, Switzerland

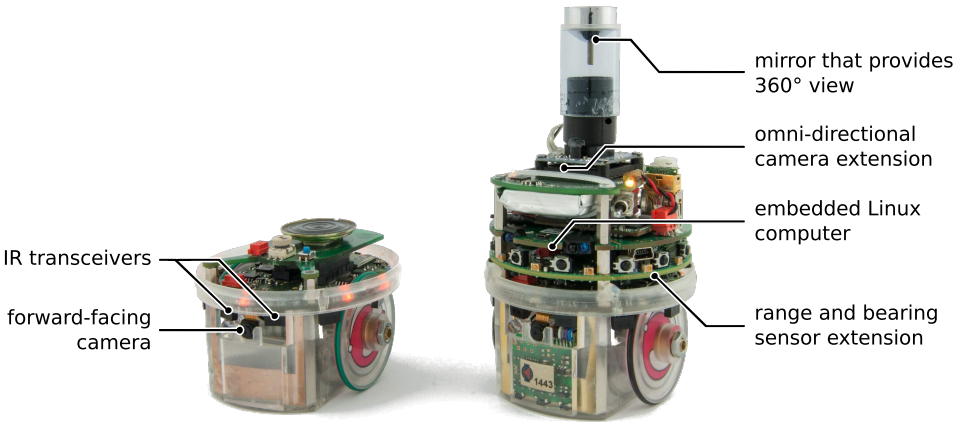



Figure A.1: Two configurations of the e-puck robot. The robot on the left is an e-puck without any extension. The sensors relevant to the TAM are the forward-facing camera for the detection of the TAM and the IR transceiver for communication with the TAM. The robot on the right is an e-puck with several extensions: the range and bearing sensor (Gutiérrez et al., 2008), the embedded computer running Linux, and the omni-directional camera. The TAM is compatible with all these extensions; the omni-directional camera is used to detect the TAM instead of the forward-facing camera, if present.

- *Proximity sensors*: Eight infrared proximity sensors for obstacle avoidance are placed around the body of the e-puck. Note that the sensors are not mounted equidistantly, but with a higher density towards the front of the robot. The proximity sensors also double as ambient light sensors. Furthermore, the proximity sensors can be employed as a communication device by using libIrcom (see Section A.1.4).

- *Forward camera*: A forward-facing color camera with VGA resolution (640×480 pixels) is mounted at the front of the e-puck. Note that the basic model of the e-puck lacks the computing power to analyze a full camera frame in real-time.

- *3D accelerometer*: This sensor measures the proper acceleration of the e-puck in three dimensions.

### A.1.2 Actuators

The e-puck features the following actuators:

- *Wheels*: Each wheel is controlled by a stepper motor. The maximum speed of the e-puck is 18 $^{cm}/_{s}$.
- *LEDs*: Eight red LEDs are distributed equidistantly around the body. The LEDs can be covered by a semi-transparent ring in order to facilitate detection by other robots.
- *Front LED*: A strong LED that projects light into the field-of-view of the forward-facing camera.

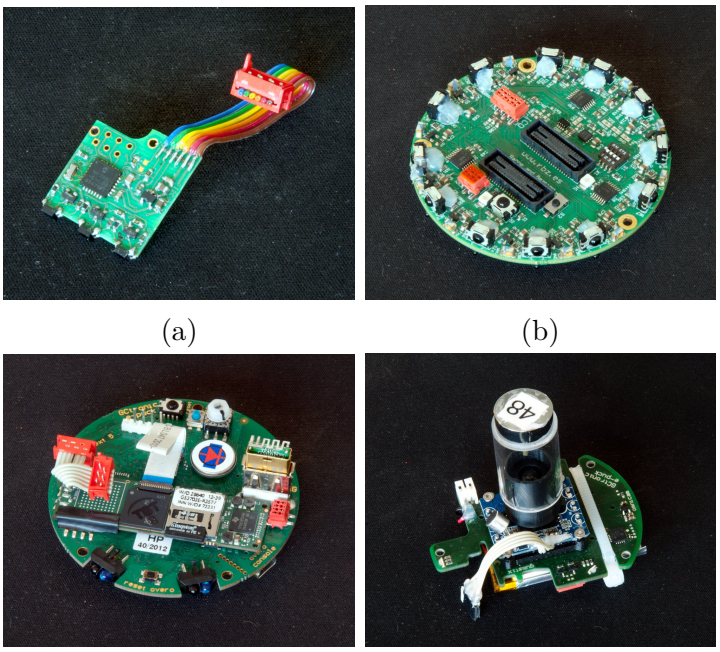
Note that the e-puck does not possess the means for manipulating objects. Furthermore, in its default configuration, the robot does not possess the means for communicating with the researcher in a scalable way.[28]

### A.1.3 Extensions

The capabilities of the e-puck can be augmented using various extensions. In the following, I describe the extensions of the e-puck that I use in the experiments discussed in this dissertation—see also Figure A.2.

- *Sound*: The sound extension features three microphones that allow an e-puck to triangulate sound sources. Additionally, the extension features a speaker for emitting sound. This extension is included in the standard distribution of the e-puck, but is not used in this dissertation—it has been included only for the sake of completeness.
- *Ground sensors*: The ground sensor extension is equipped with three infrared sensors that allow an e-puck to sense the brightness of the ground. Figure A.2a shows a picture of the extension.

[28] The e-puck does possess a Bluetooth communication device. However, Bluetooth is not suitable for communicating between a swarm and the researcher due to its point-to-point nature as well as its limited range.



(a) (b)

(c) (d)

Figure A.2: Various extensions for the e-puck robot: a) Ground sensor extension, b) Range and bearing extension, c) Embedded Linux extension, d) Omni-directional camera extension.

- *Range and Bearing*: The range-and-bearing extension enables an e-puck to sense the range and bearing of neighboring robots (Gutiérrez et al., 2008), provided that they are equipped with such an extension as well. Furthermore, the extension allows low-bandwidth communication between robots. Figure A.2b shows a picture of the extension.

- *Embedded Linux*: The extension features an embedded computer for extending the processing power of the e-puck. The embedded computer is a Gumstix Overo COM module equipped with an ARM Cortex-A8 processor core (600 MHz, 256 MiB RAM). This extension is required for using controllers developed in simulation directly on the e-puck without any changes (for information on the ARGoS simulation framework, see Section A.2). Additionally to the embedded computer, the extension features a WiFi-network module and a long-distance proximity sensor. Figure A.2c shows a picture of the extension.

Figure A.3: Snapshot from the omni-directional camera of an e-puck. Cameras of this type have the advantage that the robot can directly compute distances from the captured image.

- *Omni-directional camera*: The omni-directional camera extension brings 360° vision to the e-puck. The camera is similar to the one used in the s-bot (a robot created during the Swarm-bots project, see Mondada et al., 2004) or the marXbot (a robot created during the Swarmanoid project, see Dorigo et al., 2013). Omni-directional cameras of this type have the advantage that the robot can directly compute distances from the captured image—see Figure A.3 for a snapshot from the omni-directional camera. The extension also features a secondary battery and three top-mounted RGB LEDs. Figure A.2c shows a picture of the extension.

### A.1.4 Inter-robot communication using libIrcom

*libIrcom* is a library for the e-puck robot proposed by Campo et al. (2010a). libIrcom[29] has two functions: first, it allows an e-puck to sense the range and the bearing of other robots (similarly to the dedicated range and bearing extension presented above), and second, it allows inter-robot communication.

In the context of this dissertation, I exclusively use the inter-robot communication functionality of libIrcom for the e-puck and a reimplementation of this functionality for Atmel microprocessors for the TAM (see Appendix B, Section B.2.1).

The communication functionality of libIrcom relies on the IRcom communication protocol. The protocol allows robots to exchange messages at a maximum rate of 30 $^{\text{bytes}}/_{\text{s}}$, including a 2 bit cyclic redundancy check (CRC) that permits them to detect erroneous mes-

[29] http://gna.org/projects/e-puck/

sages (Campo et al., 2010a). Data is encoded using a frequency modulation that permits reliable communication in a wide range of light conditions.

The maximum distance of reliable message reception depends on the infrared transceivers used. Using the proximity sensors of the e-puck, messages can be detected at a distance of up to 25 cm between emitter and receiver. In libIrcom, communication is multiplexed with the proximity sensors and ambient light sensors, so that robots can still perform obstacle avoidance while using libIrcom. Messages are stored in a queue and can be retrieved at any time, unless they are overwritten when the queue is full (e.g., when the controller does not retrieve the messages regularly).

## A.2 Simulation framework: ARGoS

I performed all simulation experiments presented in this dissertation using a simulation framework called ARGoS (Pinciroli et al., 2011, 2012; Pinciroli, 2014). ARGoS was developed during the Swarmanoid project[30] with the specific aim to support the real-time simulation of large swarms of heterogeneous robots (Dorigo et al., 2013). ARGoS is open source and can be freely used for other research projects.[31]

ARGoS is a framework in the sense that it allows users to transfer controllers developed in simulation to the robots without any changes. The framework consists primarily of a simulator and an abstract interface to the hardware of the robots, which I will both outline in the following. Note that, as the simulator is by far the biggest component of the simulation framework, the name ARGoS is usually used interchangeably for the framework and the simulator.

### A.2.1 Common interface

The common interface allows users to access sensors and actuators, whether they are simulated or real. This simple feature allows users to first develop and test controllers in simulation before transferring them to the real robot. Furthermore, transferring controllers typically does not require adaptation of the source code. Various types of robot

[30] http://www.swarmanoid.org/
[31] http://iridia.ulb.ac.be/argos/

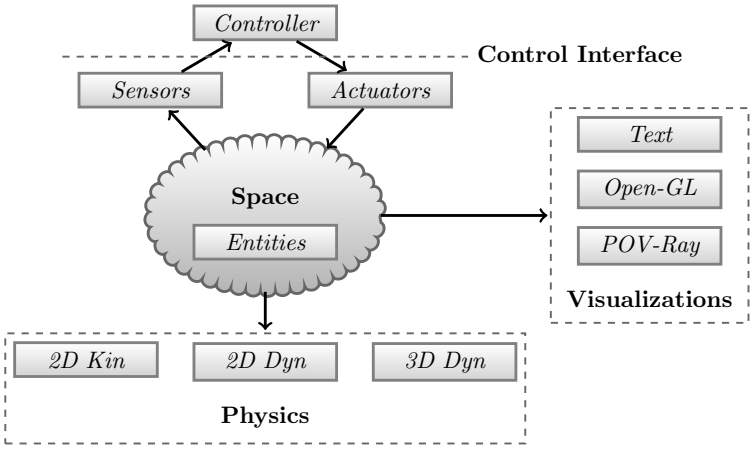



Figure A.4: Architecture of the ARGoS simulator. ARGoS is highly modularized, with the central Space entity being the only part of the simulator that cannot be removed.

platforms are supported by the common interface of ARGoS; due to its modular nature, new robot platforms can easily be added.

### A.2.2 Simulator

The simulator included in the ARGoS framework is a discrete-time, physics-based simulator. The main focus of ARGoS is to provide flexible, accurate, and efficient simulations of large robots swarms. As these are diverging requirements, the simulator accomplishes this by running multiple physics engines in parallel if desired. This feature enables the simulation of large spaces and/or swarms in parallel, making ARGoS the only simulator that can simulate thousands of robots in real-time. Furthermore, interchangeable physics engines allow the user to choose the desired level of detail: simulations can range from simple simulations with few details (e.g., two-dimensional environments governed by kinematic rules) to highly detailed and complex simulations (e.g., three-dimensional environments governed by motion dynamics). The simulator accomplishes this flexibility through its highly modular design (see Figure A.4).

Over the time of my doctoral studies, ARGoS evolved from a prototype to a very stable software.[32] Unless mentioned otherwise, I con-

[32] The current version at the time of writing this is version 3.

ducted all experiments presented in this dissertation using version 2. Furthermore, I employed a kinematics model of the robots simulated in a two-dimensional environment. By default, a uniform random noise of 10% has been added to all sensor readings at each time-step.

### A.2.3 The TAM and e-pucks

ARGoS simulates the whole set of sensors and actuators available on the e-puck. The TAM, including its sensors and actuators, is also simulated in ARGoS from version 2 onwards. In version 2, the TAM is included in the standard distribution of ARGoS.

Starting from version 3, which is highly modularized, the functionality required to simulate the e-puck robots and the TAM has been moved into separate plug-ins, which are available on the homepage of ARGoS.[33] Version 3 also allows the user to simulate the behavior of a group of TAMs using the same task controller as described in Chapter 5: task controllers can be transferred between simulation and reality without any changes. This feature greatly simplifies and speeds up the development of the control code for the coordinator.

Note that e-pucks must be equipped with the embedded Linux extension in order to be fully compatible with the ARGoS framework (see Section A.1.3).

[33] http://iridia.ulb.ac.be/argos/

Appendix B

# The TAM – Technical Details

In this appendix, I present technical details of the implementation of the TAM and its supporting infrastructure such as the control framework. In Section B.1, I provide some hardware-related details of the implementation, while in Section B.2, I provide some software-related details. Section B.3 reproduces the license under which the preceding information in Section B.1 and Section B.2 is released.

## B.1 Hardware

In this section, I report some details required to reproduce the hardware of the TAM. Electronic versions of this data and additional information such as *Gerber*-files are available at `https://github.com/arnuschky/iridia-tam/`. The TAM consists of the electronics, spread over three printed circuit boards (the *main*, *left*, and *right* circuit board), as well as a number of passive parts for the body.

### B.1.1 Bill-of-materials

In this section, I report the bill-of-materials, that is, the full list of electronic components required for producing the electronics of the TAM. Table B.1 reports the bill-of-materials for the main circuit board, which houses the majority of the electronic components. Table B.2 and B.3 reports the bill-of-materials for the left and right circuit boards, respectively. These boards only house the sensors of the TAM; hence the short list of components.

Table B.1: Bill-of-materials for the main board

| Name | Desc | Package | Manufacturer | Part number | Qty |
|---|---|---|---|---|---|
| Y1 | CERAMIC RESONATOR 16.0MHZ SMD | | Murata Electronics | CSTCE16M0V53-R0 | 1 |
| T1,2 | TRANSISTOR GP NPN AMP SOT-23 | SOT-23 | Fairchild Semiconductors | MMBT3904FSCT | 2 |
| U1 | IC CTLR ON/OFF W/DEB TSOT23-6 | 6-SOT23 | Maxim Integrated | MAX16054AZT+T | 1 |
| U2 | IC OPAMP GP 700KHZ DUAL 8TSSOP | 8-TSSOP | Texas Instruments | LM358PWR | 1 |
| U3 | IC REG BUCK BST SYNC ADJ 12MSOP | 12-MSOP | Linear Technology | LTC3536EMSE#PBF | 1 |
| U4 | IC VOLT COMPARATOR DUAL 8-TSSOP | 8-TSSOP | STMicroelectronics | LM293PT | 1 |
| U5 | IC LED DRIVER LINEAR 28-TSSOP | 28-TSSOP | Texas Instruments | TLC59116IPWR | 1 |
| U6 | IC MCU 8BIT 128KB FLASH 44TQFP | 44-TQFP | Atmel | ATMEGA1284P-AU | 1 |
| U7 | CONN RECEPT 2MM SINGLE SMD 10POS GOLD | 1X10SMD | open/any | | 2 |
| JP1A | CONTACT BATT POSITIVE AA/AAA/N | | Keystone Electronics | 637 | 1 |
| JP1B | CONTACT BATT NEG SPRING AA/AAA/N | | Keystone Electronics | 629 | 2 |
| JP2 | CONN HEADER R/A 6POS 90° GOLD SMD | 1X06SMD/90 | open/any | | 1 |
| JP3 | 2X3POS DIL VERTICAL PIN HEADER | | open/any | | 1 |
| JP4 | CONN FMALE 14POS DL .1" GOLD SMD | 2X07SMD | open/any | | 1 |
| S1 | SWITCH TACTILE SPST-NO 0.05A 12V | | C&K Components | PTS525SM10SMTR LFS | 1 |
| S2 | SWITCH TAPE SEAL 2 POS SMD | CTS-219-02 | CTS Electrocomponents | 219-2MST | 1 |
| L1 | INDUCTOR POWER 4.7UH 1.3A SMD | VLCF5024T | TDK Corporation | VLCF5024T-4R7N1R3-2 | 1 |
| L2 | INDUCTOR MULTILAYER 10UH 0805 | 0805 | TDK Corporation | MLZ2012M100W | 1 |
| LED1 | LED CHIPLED 570NM GREEN 1206 SMD | 1206 | OSRAM Opto Semiconductors | LG N971-KN-1 | 1 |
| LED2 | THROUGH HOLE RGB FULL COLOR | | Hanhua | HH-500CRGBW503 | 1 |
| LED3 | LED YELLOW CLEAR THIN 0805 SMD | 0805 | Lite-On Inc | LTST-C171YKT | 1 |
| LED4 | LED AMBER CLR THIN 0805 SMD | 0805 | Lite-On Inc | LTST-C171AKT | 1 |
| LED5 | LED RED ORAN CLEAR THIN 0805 SMD | 0805 | Lite-On Inc | LTST-C171EKT | 1 |
| RGB_LED_L,R,M | LED MULTILED RGB WHT BINNED 6PLC | 6-PLCC | OSRAM Opto Semiconductors | LRTBG6SF-V2BA-3E7F | 3 |
| R1 | RES 42.2k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R2,22,23 | RES 4.7k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 3 |
| R3,8,16 | RES 1M OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 3 |
| R4 | RES 100K OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R5 | RES 49.9k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R6 | RES 6.49k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R7 | RES 221k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R10,11 | RES 0.0 OHM 1/10W 5% 0603 SMD | 0603 | open/any | | 2 |
| R12,13,19,20,24,25 | RES 10k OHM 1/10W 5% 0603 SMD | 0603 | open/any | | 6 |
| R14 | RES 47k OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R15 | RES 20K OHM 1/10W 1% 0603 SMD | 0603 | open/any | | 1 |
| R17 | RES 39 OHM 1/10W 5% 0603 SMD | 0603 | open/any | | 1 |
| R18,21 | RES 470 OHM 1/10W 5% 0603 SMD | 0603 | open/any | | 2 |
| R9,26,27,28 | RES 68 OHM 1/10W 5% 0603 SMD | 0603 | open/any | | 4 |
| C1,2,10,11,13,14,15,16,17,18 | CAP CER 0.1UF 16V 10% X7R 0603 | 0603 | Taiyo Yuden | EMK107B7104KA-T | 10 |
| C3,12,19 | CAP CER 10UF 10V 10% X5R 0805 | 0805 | Murata Electronics | GRM21BR61A106KE19L | 3 |
| C4,8 | CAP CER 1UF 16V 10% X7R 0603 | 0603 | TDK Corporation | C1608X7R1C105K080AC | 2 |
| C5 | CAP CER 220PF 50V 5% NP0 0603 | 0603 | TDK Corporation | C1608C0G1H221J080AA | 1 |
| C6 | CAP CER 47PF 50V 5% NP0 0603 | 0603 | TDK Corporation | C1608C0G1H470J080AA | 1 |
| C7 | CAP CER 22UF 6.3V 10% X5R 1206 | 1206 | AVX Corporation | 12066D226KAT2A | 1 |
| C9 | CAP CER 100PF 16V 10% X7R 0603 | 0603 | open/any | | 1 |

Table B.2: Bill-of-materials for the left side board

| Name | Desc | Value | Package | Manufacturer | Part number | Qty |
|---|---|---|---|---|---|---|
| IR_LED_L | IR EMITTER | SFH 4258 | 4-PLCC | OSRAM Opto Semiconductors | SFH4258-Z | 1 |
| IR_TRANSISTOR_L | IR PHOTOTRANSISTOR | SFH 320 FA-Z | PLCC-2 | OSRAM Opto Semiconductors | SFH320FA-Z | 1 |
| PROX_L | OPTO TRANS 4MM REFL | TCRT1000 | | Vishay Semiconductors | TCRT1000 | 1 |

Table B.3: Bill-of-materials for the right side board

| Name | Desc | Value | Package | Manufacturer | Part number | Qty |
|---|---|---|---|---|---|---|
| IR_LED_R | IR EMITTER | SFH 4258 | 4-PLCC | OSRAM Opto Semiconductors | SFH4258-Z | 1 |
| IR_TRANSISTOR_R | IR PHOTOTRANSISTOR | SFH 320 FA-Z | PLCC-2 | OSRAM Opto Semiconductors | SFH320FA-Z | 1 |

### B.1.2 Schematics

In this section, I reproduce the circuit schematics of the TAM. The overall circuit is distributed over the main, left, and right printed circuit boards. Electrical connections between the different printed circuit boards are are provided by solder joints at the location where the circuit boards join.

Figure B.1 shows the circuit schematics for the power supply and the battery support circuits (e.g., soft on/off button handling and the battery protection circuit). The power supply of the TAM is based on the Linear LTC3536, a highly efficient boost/buck DC/DC regulator. The regulator provides a stable operating voltage of 3.3 V for the whole range of input voltage delivered by the Lithium-Ion battery (typically, 2.7 V - 4.2 V).

Figure B.2 shows the circuit schematics for the connectors on the main circuit board towards the left and right circuit board, as well as the filtering circuit used for infrared communication via IRcom (see Section B.2.1).

Figure B.3 shows the circuit schematics for the PWM LED controller and the RGB LEDs. The controller, a TLC5911 constant current sink driver, supports 24-bit colors on 16 channels. It is connected to the main processor using an $I^2C$ bus.

Figure B.4 shows the circuit schematics around the main processor of the TAM, an Atmel AVR ATmega-1284p running at 16 MHz with 16 KiB main memory and 128 KiB flash memory. The circuit shown replicates the base circuit of Arduino (Banzi, 2008).

Figure B.5 shows the support circuit for the 2.4 GHz IEEE 802.15.4 radio module used for mesh-networking. The TAM is compatible, on the protocol level, with IEEE 802.15.4 modules of various manufacturers; however, different modules might require a different socket. The TAM can be configured to work on 4 different wireless channels, which allows for up to four experiments to be run in parallel in close proximity.

Figure B.6 shows the circuit schematics for the light barriers on the left side of the TAM and the infrared transceiver for communication with the robots.

Figure B.7 shows the circuit schematics for the light barriers on the right side of the TAM.

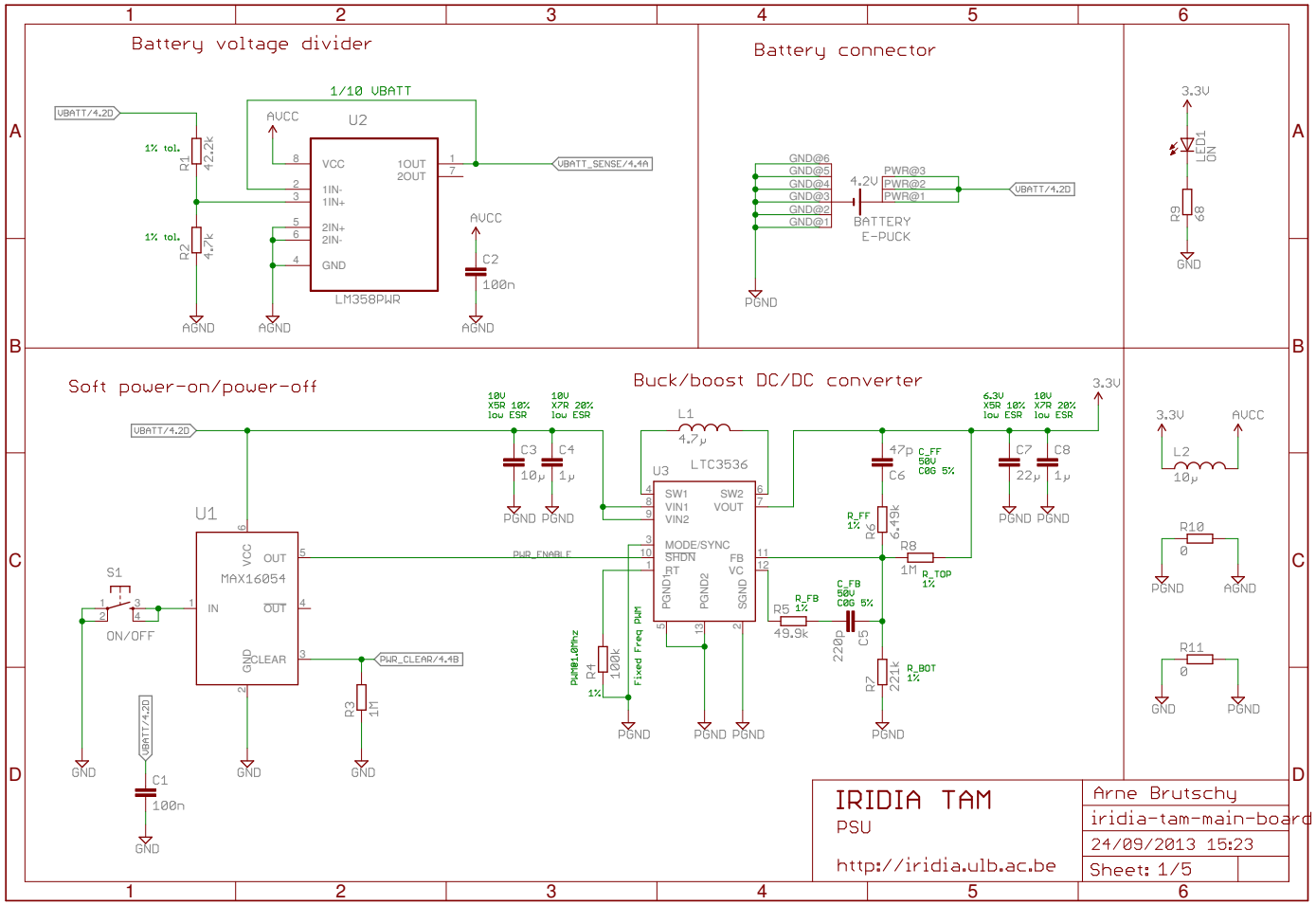



Figure B.1: Circuit schematics for the power supply and battery support circuits

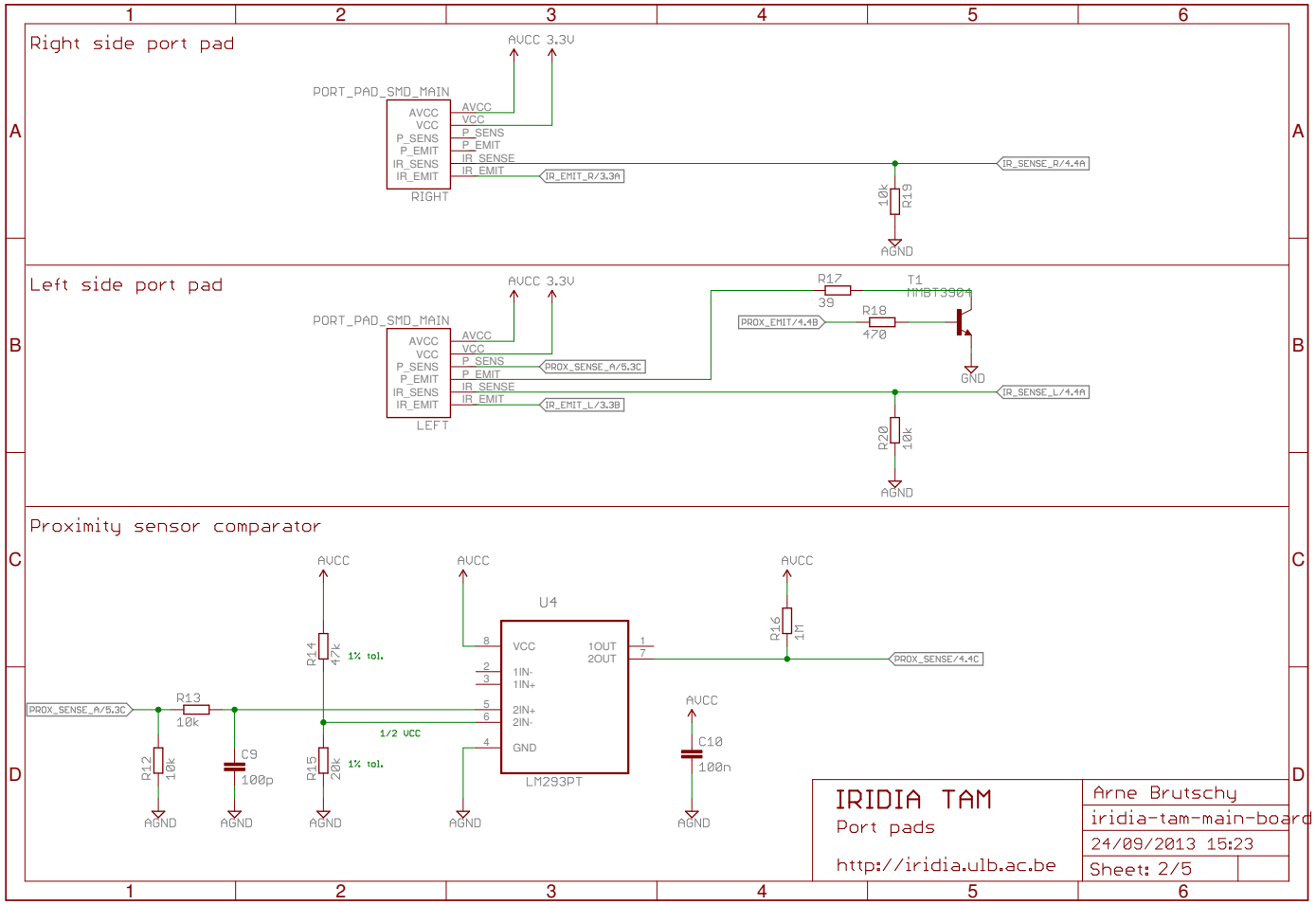



Figure B.2: Circuit schematics for the connectors for the side boards and infrared transceiver

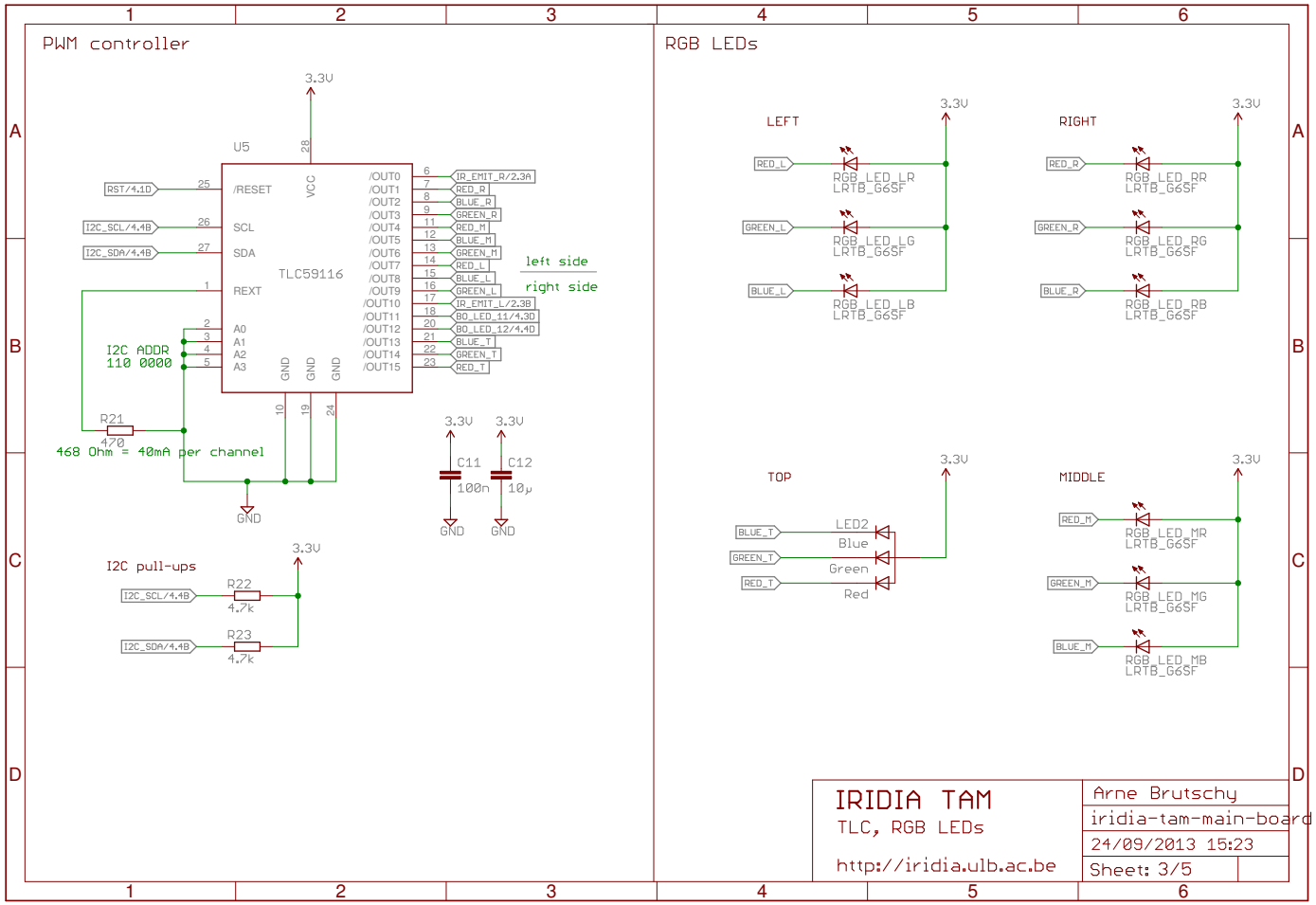



Figure B.3: Circuit schematics for the LEDs and the PWM controller

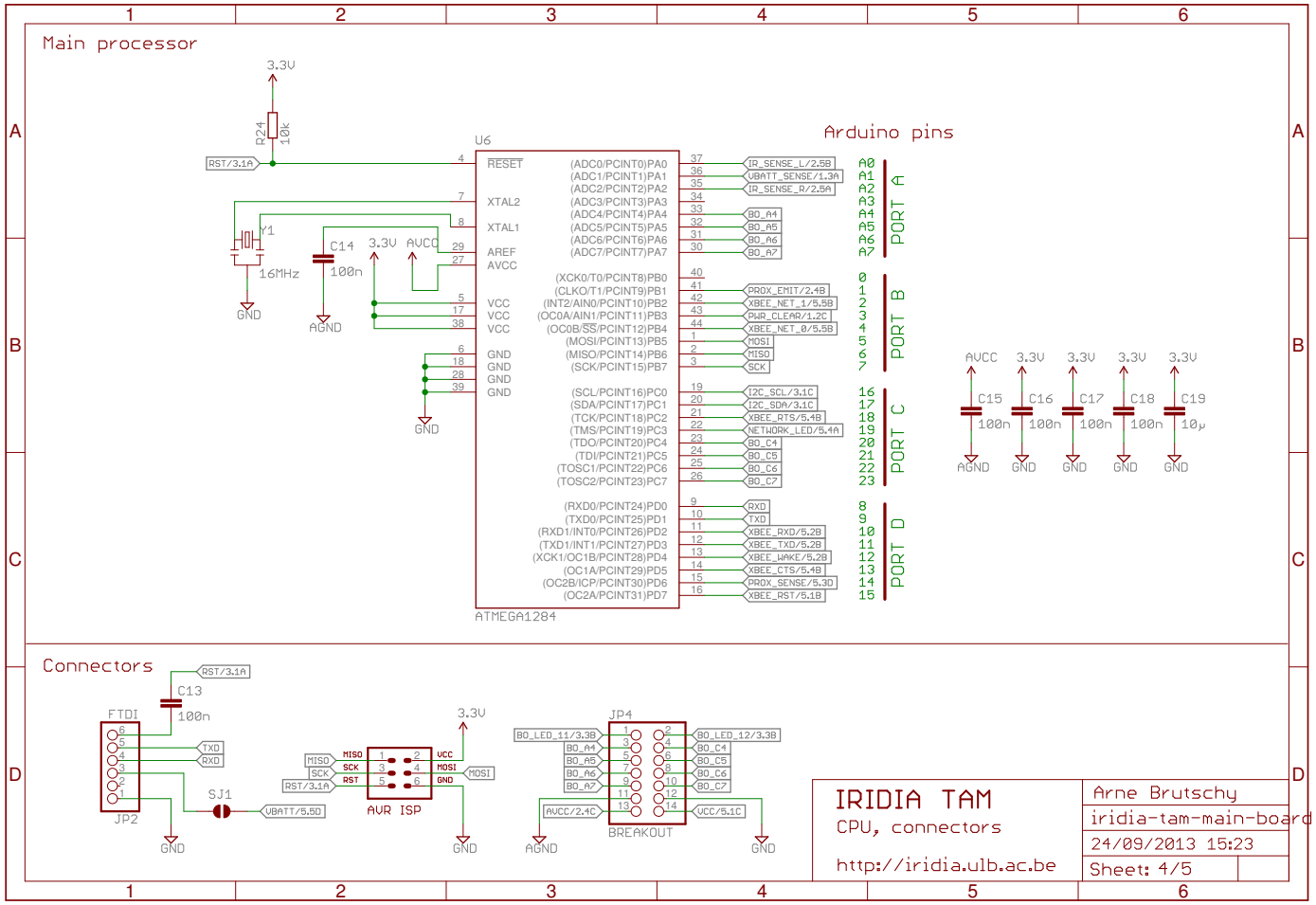



Figure B.4: Circuit schematics for the main processor and supporting connectors

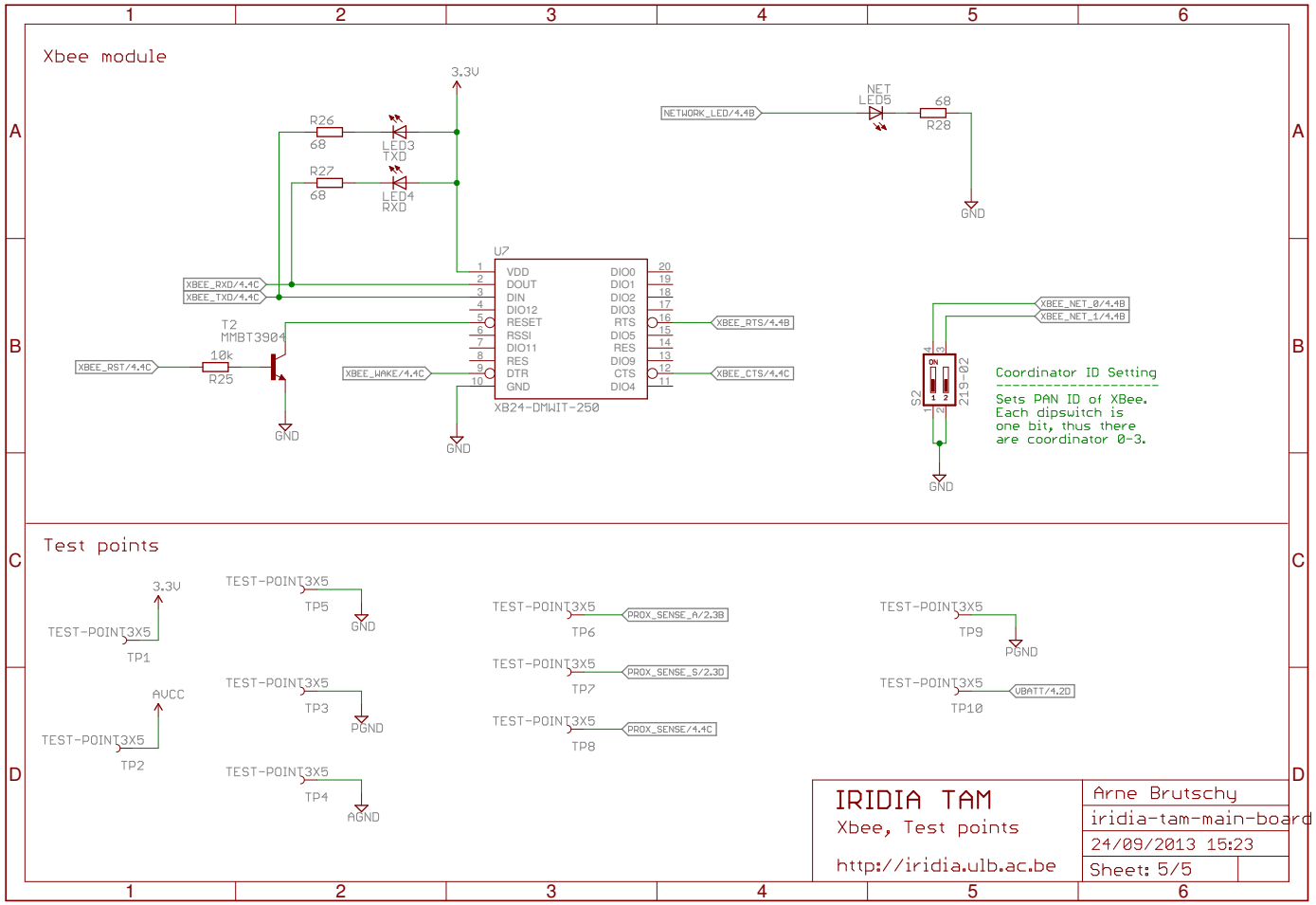



Figure B.5: Circuit schematics for the mesh-networking radio module

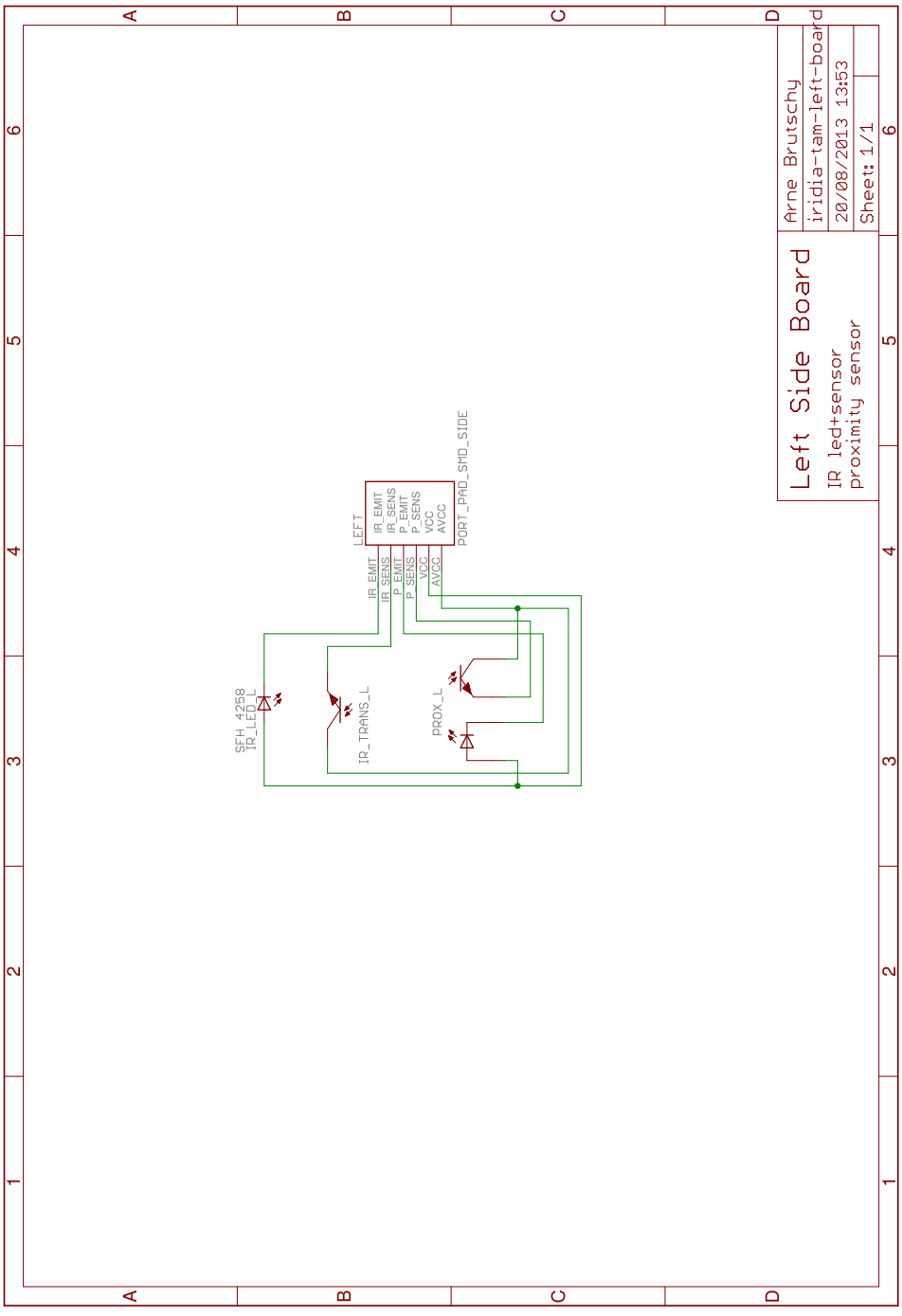



Figure B.6: Circuit schematics for the left-side light barriers and infrared transceiver for communication

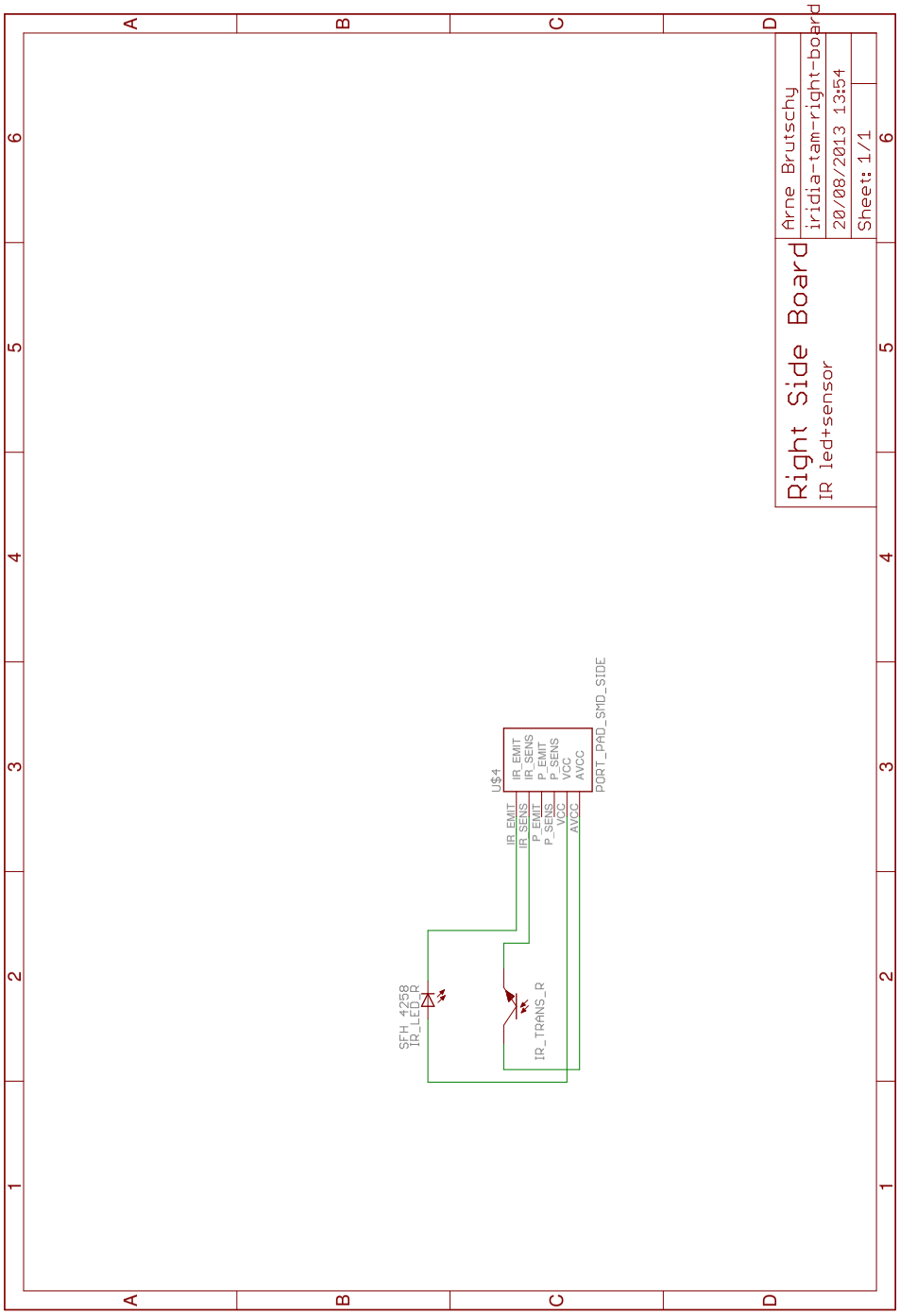



Figure B.7: Circuit schematics for the right-side light barriers

### B.1.3 Circuit boards

In this section, I reproduce the layouts of the printed circuit boards that compose the active parts of the TAM.

The electronics of the TAM consist of three printed circuit boards: The *main* circuit board, at the back of the TAM, is flanked by two boards, forming a "U" shape. The two boards at the side of the TAM are called the *left* and *right* circuit board. The main circuit board is mechanically joined with the other circuit boards at a 90° angle using two interlocking slots.

The circuit boards were produced by a professional service. The main board is a 2-layer FR-4 board, the left and right board are single-layer FR-4 boards. All boards have a white soldermask so that the TAM can be sensed without issues by the proximity sensors of the e-puck. Applying a silkscreen is optional, but advisable on the main board as it provides instructions for the selection of radio channels as well as the extension connector pinout—see Section B.1.5.

Figure B.8 shows the circuit board layout of the main board. At the center, the main processor; above that on the left, the extension connector. At the bottom center, the battery connectors. At the lower left, the power supply; above that, the PWM LED controller. At the right, the filtering circuit for the infrared transceiver; above that, the connector for the IEEE 802.15.4 radio module. At the sides, the slots into which the side boards slide; just above it, the pads for the solder bridges that connect to the other circuit boards.

Figure B.9 shows the circuit board layout of the left circuit board. On the top right, the slot into which the main board slides; just next to it, the solder-pads for the solder bridges that connect the board to the main board. At the top, the infrared transceiver for communication with the robot. At the bottom, the 850 nm infrared emitters and matching photo transistors used for the light barriers.

Figure B.10 shows the circuit board layout of the right circuit board. On the top left, the slot into which the main board slides; just next to it, the solder-pads for the solder bridges that connect the board to the main board. At the bottom, the 850 nm infrared emitters and matching photo transistors used for the light barriers.

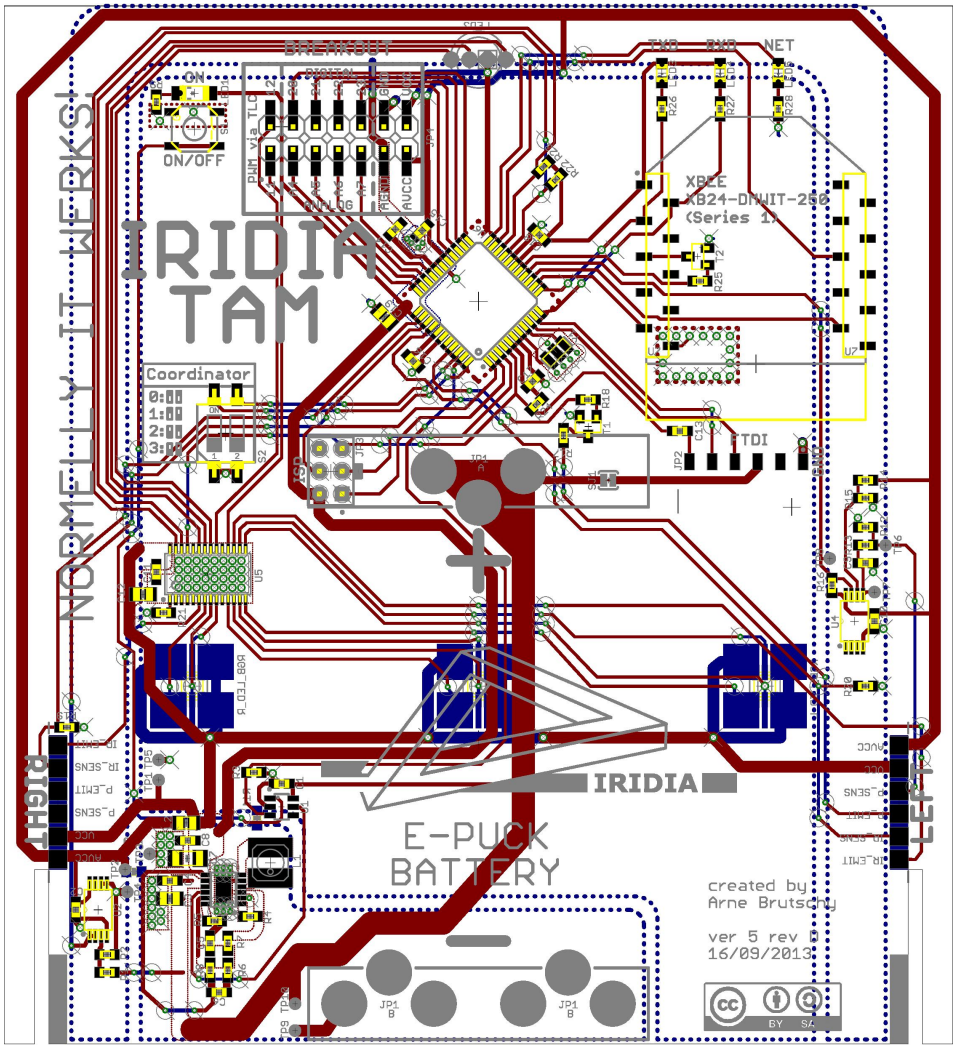



Figure B.8: Layout of the main circuit board. At the center, the main processor; above that on the left, the extension connector. At the bottom center, the battery connectors. At the lower left, the power supply; above that, the PWM LED controller. At the right, the filtering circuit for the infrared transceiver; above that, the connector for the IEEE 802.15.4 radio module. At the sides, the slots into which the side boards slide; just above it, the pads for the solder bridges that connect to the other circuit boards.

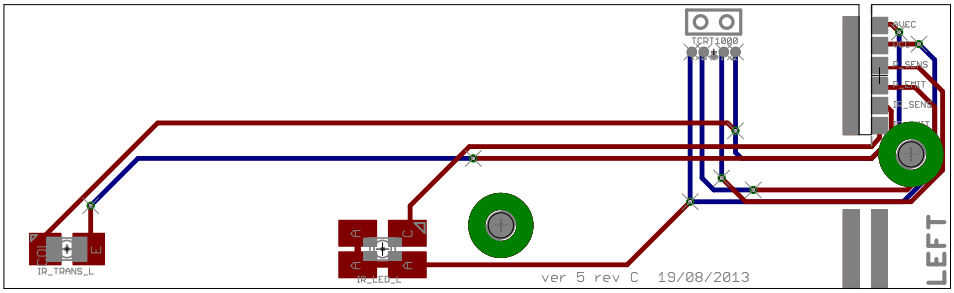



Figure B.9: Layout of the left circuit board. On the top right, the slot into which the main board slides; just next to it, the solder-pads for the solder bridges that connect the board to the main board. At the top, the infrared transceiver for communication with the robot. At the bottom, the 850 nm infrared emitters and matching photo transistors used for the light barriers.

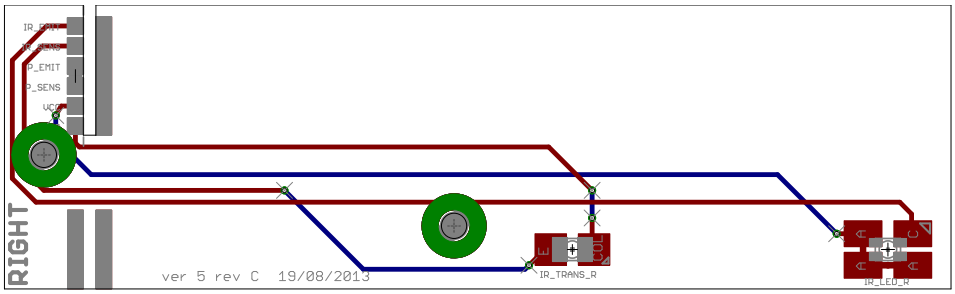



Figure B.10: Layout of the right circuit board. On the top left, the slot into which the main board slides; just next to it, the solder-pads for the solder bridges that connect the board to the main board. At the bottom, the 850 nm infrared emitters and matching photo transistors used for the light barriers.

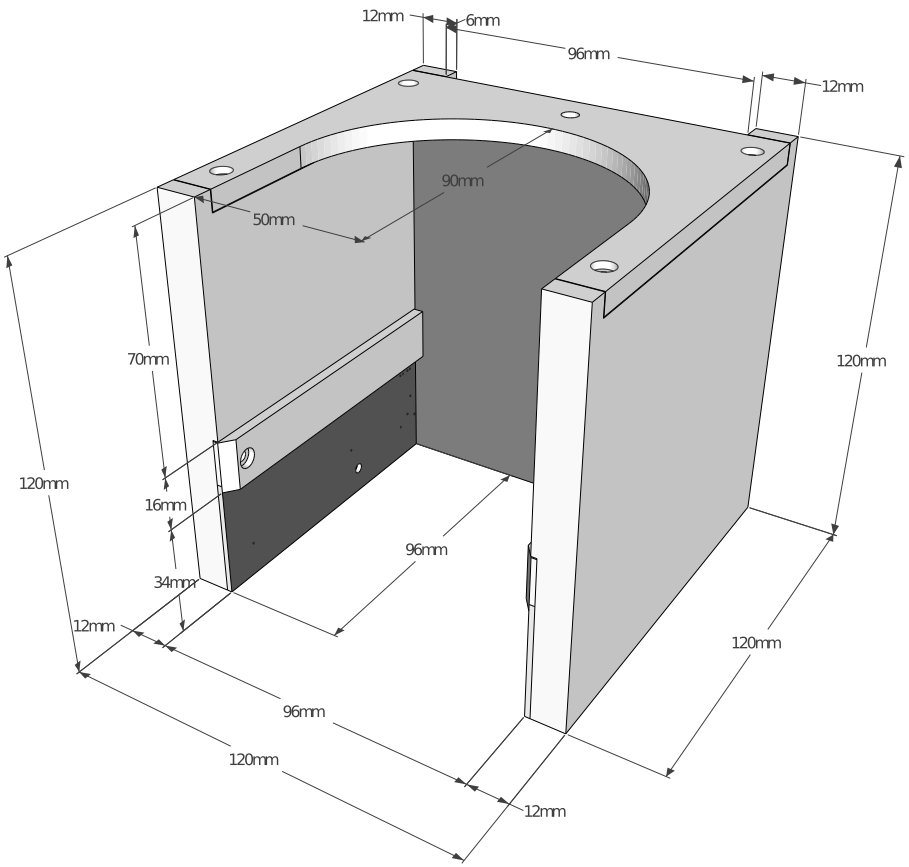
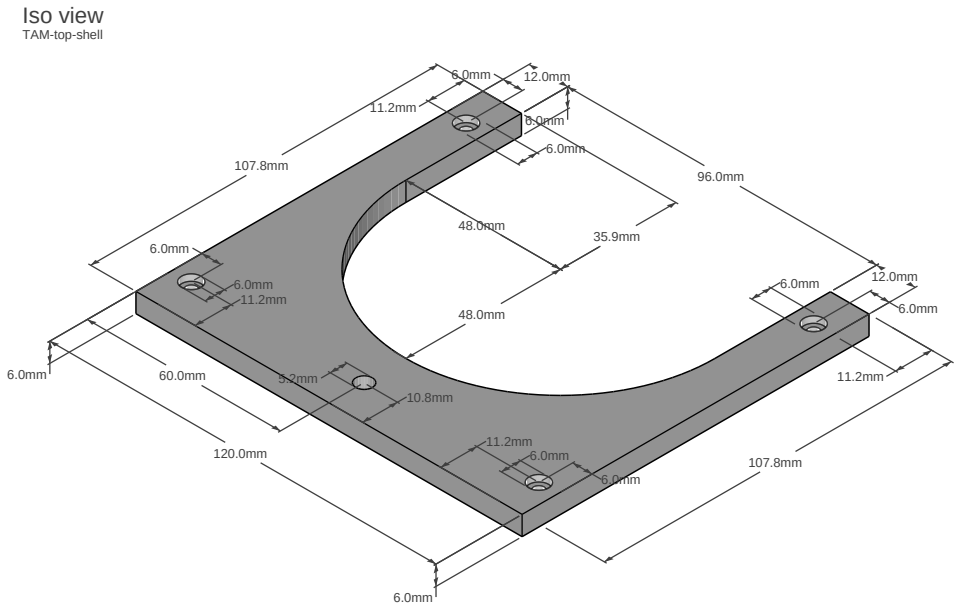


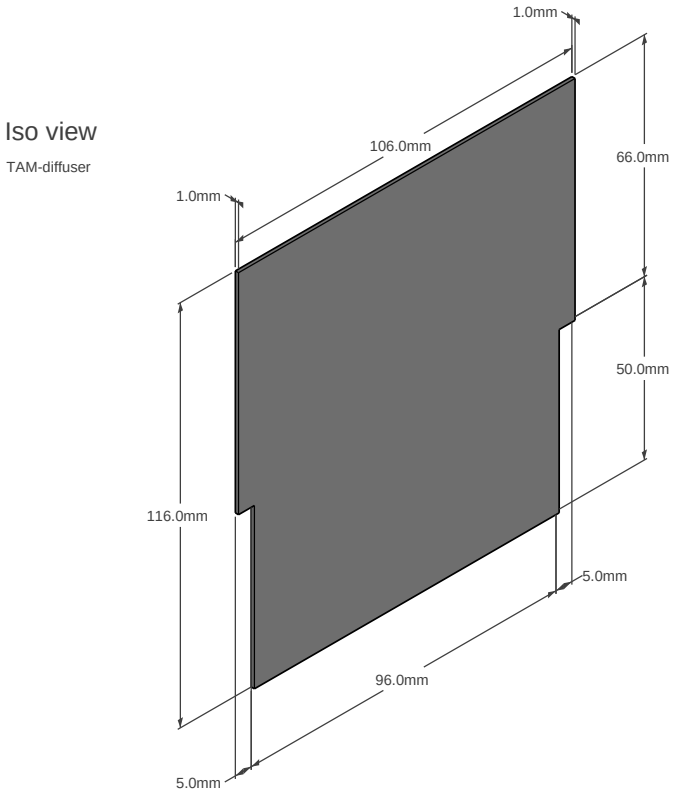

Figure B.11: 3D CAD model of the TAM including measurements

### B.1.4 Plastic body

The TAM has a cubical shape with a length of 12 cm in every dimension. Figure B.11 shows a 3D model including measurements. In the following, I present technical details on its plastic body, which is composed of six parts: the *top bracket*, which provides structural integrity to the whole assembly (Figure B.12a); the *LED diffuser*, which diffuses the light from the RGB LEDs for better perception by the cameras of the robots (Figure B.12b); the left and right *side walls*, which are fastened to the left and right circuit boards, respectively (Figure B.13a and Figure B.14a); and the left and right *bumpers*, which act as a rail at the inside of the booth in order to prevent robots from getting stuck on the sensors of the light barrier (Figure B.13b and Figure B.14b). 2D-plans suitable for production by a professional service are available at `https://github.com/arnuschky/iridia-tam/`

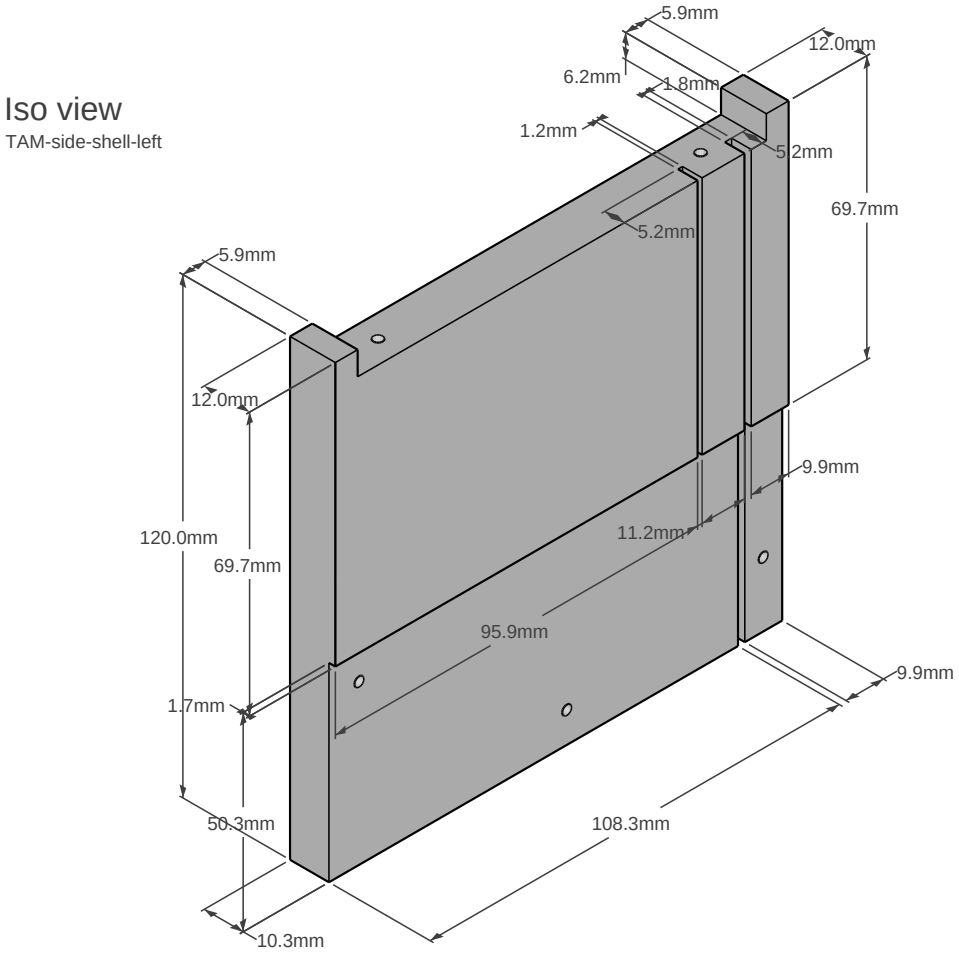



(a) 3D CAD model of the top bracket

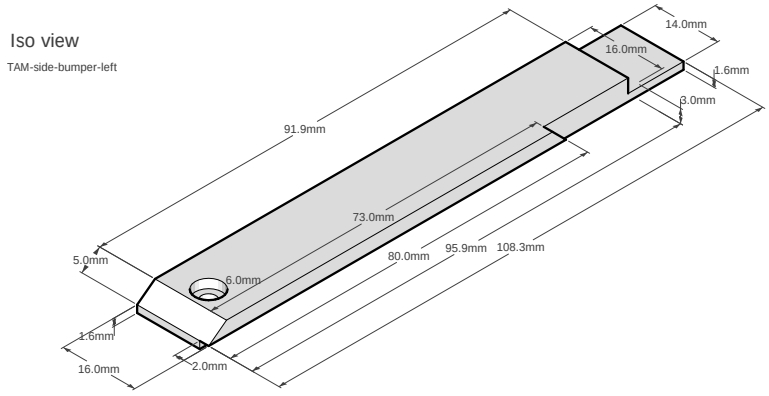



(b) 3D CAD model of the LED diffuser

Figure B.12: 3D CAD models of the plastic body, center parts

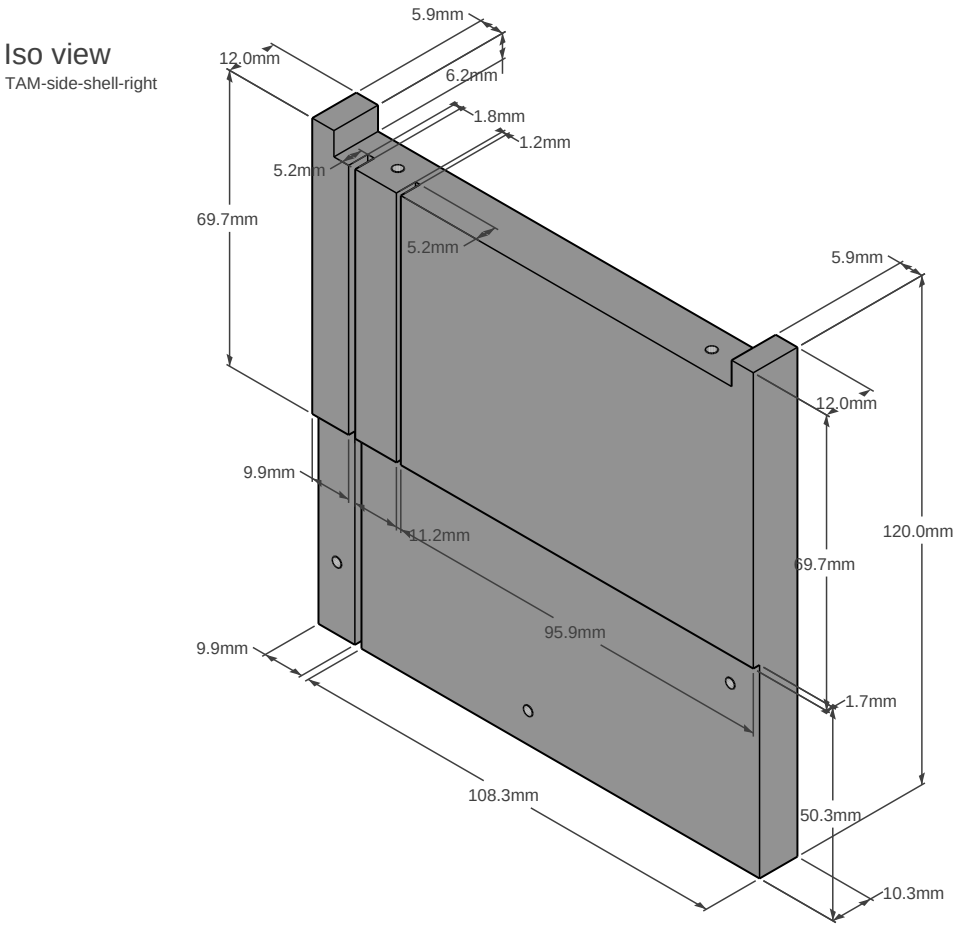



(a) 3D CAD model of the left side wall

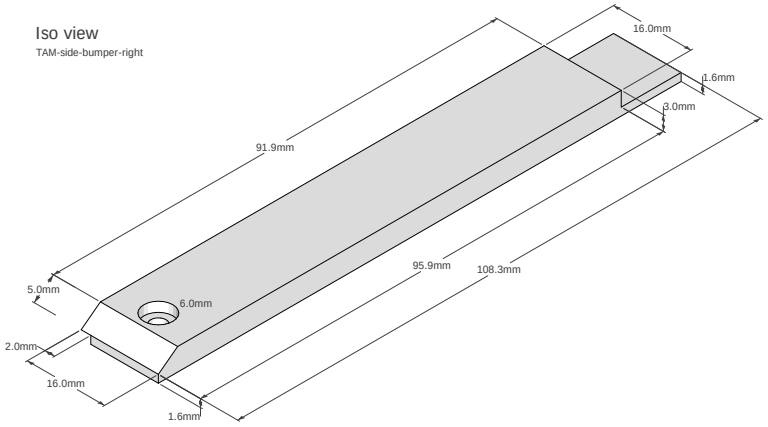



(b) 3D CAD model of the left bumper

Figure B.13: 3D CAD models of the plastic body, left-side parts




(a) 3D CAD model of the right side wall




(b) 3D CAD model of the right bumper

Figure B.14: 3D CAD models of the plastic body, right-side parts

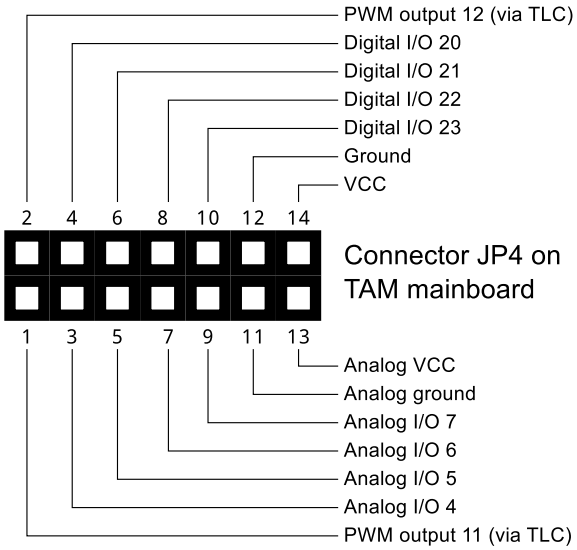



Figure B.15: Pinout of the extension connector

### B.1.5 Extensions

The philosophy of the TAM follows Arduino and the e-puck in terms of extensibility: the TAM features an extension connector that allows students and researchers to easily extend its capabilities without requiring a modification of the TAM itself. The pinout of this connector is shown in Figure B.15; it is also reproduced on the circuit board of the TAM, next to the extension connector.

Additionally to supplying power, the extension connector provides 10 input/output channels: 2 channels for PWM LED control, 4 digital channels and 4 analog channels. The PWM LED control channels are connected to the PWM controller and can be controlled by using an $I^2C$ library for communication with the PWM controller. This library is supplied with the firmware of the TAM. The digital and analog channels are directly connected to the I/O pins of the main processor; they therefore do not require an any additional library.

## B.2 Software

In this section, I report some details of the software for controlling the TAM. The full control framework, including the source code of the firmware, the coordinator, the common interface, and a task controller template are available at https://github.com/arnuschky/iridia-tam/.

### B.2.1 IRcom support

*IRcom* is a communication protocol for communicating using short-range infrared transceivers. It has been initially proposed by Campo et al. (2010a) for exclusive use on the e-puck robots in form of a library called *libIrCom*—see Appendix A, Section A.1.4 for details on libIrcom. However, due to the pervasiveness of infrared transceivers in robotics (e.g., for the purpose of obstacle avoidance), IRcom can potentially be ported to many robot platforms.

The firmware of the TAM supports IRcom to communicate with the e-puck. To this end, the original libIrcom has been ported and modified to work on Atmel microprocessors. The implementation for the TAM deviates from the original libIrcom primarily by its inability to measure the range and bearing of a sending robot. In other words, the TAM only supports communication via IRcom.

The implementation of IRcom for the TAM deviates further from the original implementation in two details:

1. Hardware filtering: as the communication protocol only needs to distinguish between ones and zeros, rather than measuring the intensity of a signal as required for range and bearing sensing, the TAM implements a simplified and efficient hardware-based signal filter;

2. Hardware interrupts: as the TAM features only a single sensor, the implementation uses a hardware interrupt that is executed upon a change in the signal, rather than continuously polling all available sensors.

As a result, the implementation used on the TAM is more efficient than libIrcom used on the e-puck robots—a necessity, as the microprocessor used on the TAM is considerably less powerful than the microprocessor of the e-puck.

## B.2.2 Common interface

The *common interface* provides an abstract interface to the TAMs used in an experiment. It allows researchers to query and modify the state of an individual TAM in an abstract fashion; all low-level operations are handled by the coordinator. The interface is common in the sense that the same interface is used for simulation experiments and robot experiment: the common interface is replicated by the plug-in used for simulating the TAM with ARGoS. Consequently, task controllers written against the common interface can be ported from simulation to reality without requiring any modifications. In practice, the common interface consists of three components:

- the TAM interface, which provides access to data and functionality of the TAMs;
- the task controller interface, which is an abstract interface that every task controller must implement; and
- the experiment interface, which defines a mapping of task controllers to task instances (and therefore TAMs) in an experiment.

Below, I will reproduce these three components implemented using Java interfaces.

### TAM interface

```
package be.ac.ulb.iridia.tam.common;

/**
 * This interface implements all methods required to manipulate and
 * query the state of the TAM.
 *
 * @author Arne Brutschy
 */
public interface TAMInterface
{
    /**
     * Returns the color of the RGB LEDs of the TAM as currently known.
     * @return LedColor object reflecting the 24bit color, or null if TAM didn't
           report status yet
     */
    LedColor getLedColor();

    /**
     * Sets a new LED color.
     * Update is ignored if LED color does not change.
     * @param ledColor  LedColor object reflecting the 24bit color
     */
```

```
    void setLedColor(LedColor ledColor);

    /**
     * Returns true if there is currently a robot in the TAM.
     * @return true if there is currently a robot in the TAM
     */
    boolean isRobotPresent();

    /**
     * Returns the id of the TAM.
     *  - it's TAMXX with XX being a 2-digit unique integer; or
     *  - it's the last 5 characters of the 64bit address if the id hasn't been
          resolved yet
     * @return id of TAM as String 5 characters long
     */
    String getId();

    /**
     * Sets a value for the robotData.
     * Update is ignored if data did not change
     * @param robotData new robotData value
     */
    void setRobotDataToSend(int robotData);

    /**
     * Returns the data received from the robot currently in the TAM.
     * @return data received from the robot currently in the TAM, as int
     */
    int getRobotDataReceived();

    /**
     * Returns the user-defined controller of the TAM.
     * @return controller of the TAM
     */
    ControllerInterface getController();

    /**
     * Sets the user-defined controller of the TAM.
     * @see be.ac.ulb.iridia.tam.common.ControllerInterface
     * @param controller  user-defined controller of the TAM
     */
    void setController(ControllerInterface controller);
}
```

## Controller interface

```
package be.ac.ulb.iridia.tam.common;

/**
 * Interface for a user-defined controller that can be attached to a one
 * or multiple TAMs. The controller can set the LED color of the TAM, check
 * for the presence of a robot and read/write data from the robot using IRcom.
 * @see be.ac.ulb.iridia.tam.common.TAMInterface
 *
 * @author Arne Brutschy
 */
public interface ControllerInterface
```

```
{
    /**
     * Resets the controller.
     */
    public void reset();

    /**
     * Step function of the controller.
     * Called every Coordinator.STEP_INTERVAL milliseconds.
     */
    public void step();
}
```

## Experiment interface

```
package be.ac.ulb.iridia.tam.common;

/**
 * Interface for a "TAM experiment". This should be implemented by your main
     class.
 *
 * The coordinator uses the attachTAMController() function to attach controllers
      to newly discovered TAMs.
 * You can use the id of the TAM to attach specific controllers to specific TAMs
     ,
 * thereby giving them the different functionality.
 *
 * Your main() function should look similar to this:
 *
 *   Coordinator coordinator = new Coordinator();
 *   ExperimentInterface experiment = new YourExperimentClass();
 *   experiment.init(System.currentTimeMillis());
 *   coordinator.setExperiment(experiment);
 *   coordinator.start();
 *
 * @author Arne Brutschy
 */
public interface ExperimentInterface
{
    /**
     * Initializes experiment.
     * @param randomSeed  seed for the prng, set either constant or use System.
         currentTimeMillis()
     */
    public void init(long randomSeed);

    /**
     * Resets the experiment.
     */
    public void reset();

    /**
     * Called by the coordinator to attach controllers to newly discovered TAMs.
     * You can use the id of the TAM to attach specific controllers
     * to specific TAMs, thereby giving them the different functionality.
     * @param tam  TAM the coordinator requests a coordinator for
     */
```

```
    public void attachTAMController(TAMInterface tam);

    /**
     * Checks whether the experiment is ready to start.
     * @return true if ready to start
     */
    public boolean isReady();

    /**
     * Checks whether the experiment should finish.
     * @return true if should finish
     */
    public boolean isFinished();

    /**
     * Called by the coordinator on regular intervals.
     * Can be used for management of TAMs etc.
     */
    public void step();
}
```

## B.3 License

The TAM, including all its components in hard- and software, is open source. All components, except the IRcom library for the TAM,[34] are licensed under the *Creative Commons Attribution-ShareAlike 3.0 Unported License*, included hereunder.

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. **Definitions**

   a) **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with

[34] As the IRcom library for the TAM has been ported from the e-puck, it retains the licensing of the original libIrcom library, which is licensed under the GNU GENERAL PUBLIC LICENSE, Version 3.

a moving image ("synching") will be considered an Adaptation for the purpose of this License.

b) **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

c) **"Distribute"** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

d) **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

e) **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

f) **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

g) **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

h) **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

i) **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. **Fair Dealing Rights.** Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. **License Grant.** Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a) to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

   b) to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

   c) to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

   d) to Distribute and Publicly Perform Adaptations.

   e) For the avoidance of doubt:

      i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

      ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

      iii. **Voluntary License Schemes.** The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. **Restrictions** The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a) You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(b), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(b), as requested.

   b) If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor

institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4 (b) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

c) Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. **Representations, Warranties and Disclaimer**

   UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. **Limitation on Liability.** EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. **Termination**

   a) This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b) Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under

the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. **Miscellaneous**

a) Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b) Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c) If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d) No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e) This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

f) The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

# List of Tables

# List of Figures

# List of Publications Resulting from Work Presented in This Dissertation


M. Brambilla, **A. Brutschy**, M. Dorigo, and M. Birattari (2014). Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems*. In press.

**A. Brutschy**, L. Garattoni, M. Brambilla, G. Francesca, G. Pini, M. Dorigo, and M. Birattari (2014a). The TAM: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*. Under review.

**A. Brutschy**, G. Pini, and A. Decugnière (2012a). Grippable objects for the foot-bot. Technical Report TR/IRIDIA/2012-001, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

**A. Brutschy**, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo (2014b). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 28(1):101–125.

**A. Brutschy**, A. Scheidler, E. Ferrante, M. Dorigo, and M. Birattari (2012b). Can ants inspire robots? In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), Video Proceedings*. IEEE Press, Piscataway, NJ.

**A. Brutschy**, A. Scheidler, D. Merkle, and M. Middendorf (2008). Learning from house-hunting ants: Collective decision-making in

organic computing systems. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, editors, *Swarm Intelligence, Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence, (ANTS'08)*, volume 5217 of *Lecture Notes in Computer Science*, pages 96–107. Springer, Berlin/Heidelberg, Germany.

**A. Brutschy**, N.-L. Tran, N. Baiboun, M. Frison, G. Pini, A. Roli, M. Dorigo, and M. Birattari (2011). Costs and benefits of behavioral specialization. In *Proceedings of the 12th Annual Conference on Towards Autonomous Robotic Systems (TAROS'11)*, volume 6856 of *Lecture Notes in Computer Science*, pages 90–101. Springer, Berlin/Heidelberg, Germany.

**A. Brutschy**, N.-L. Tran, N. Baiboun, M. Frison, G. Pini, A. Roli, M. Dorigo, and M. Birattari (2012c). Costs and benefits of behavioral specialization. *Robotics and Autonomous Systems*, 60(11):1408–1420.

M. Castillo-Cagigal, **A. Brutschy**, Á. Gutiérrez, and M. Birattari (2014). Temporal task allocation in periodic environments: An approach based on synchronization. In *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)*, volume 8667 of *Lecture Notes in Computer Science*, pages 182–193. Springer, Berlin/Heidelberg, Germany.

M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, **A. Brutschy**, D. Burnier, A. Campo, A. L. Christensen, A. Decugnière, G. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. Guzzi, V. Longchamp, S. Magnenat, J. Martinez Gonzales, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rétornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71.

M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, **A. Brutschy**, D. Burnier, A. Campo, A. L. Christensen, A. Decugnière, G. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. M. Gonzales,

J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. M. de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rétornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard (2011). Swarmanoid, the movie. In *AAAI-11 Video Proceedings*. AAAI Press, San Francisco, CA. Winner of the "Best Video" award.

G. Francesca, M. Brambilla, **A. Brutschy**, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, V. Trianni, and M. Birattari (2014a). An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts. In M. Dorigo, M. Birattari, S. Garnier, H. H. M. M. de Oca, C. Solnon, and T. Stützle, editors, *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)*, volume 8667 of *Lecture Notes in Computer Science*, pages 25–37. Springer, Berlin/Heidelberg, Germany.

G. Francesca, M. Brambilla, **A. Brutschy**, V. Trianni, and M. Birattari (2014b). AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112.

M. Frison, N.-L. Tran, N. Baiboun, **A. Brutschy**, G. Pini, A. Roli, M. Dorigo, and M. Birattari (2010). Self-organized task partitioning in a swarm of robots. In M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, R. Groß, E. Şahin, H. Sayama, and T. Stützle, editors, *Proceedings of the 7th International Conference on Swarm Intelligence (ANTS'10)*, volume 6234 of *Lecture Notes in Computer Science*, pages 287–298. Springer, Berlin/Heidelberg, Germany.

N. Mathews, G. Valentini, A. L. Christensen, R. O'Grady, **A. Brutschy**, and M. Dorigo (2014). Spatially targeted communication in decentralized multirobot systems. *Autonomous Robots*. Under review.

C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, **A. Brutschy**, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo (2012). ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, **A. Brutschy**, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, T. Stirling, Á. Gutiérrez, L. M. Gambardella, and M. Dorigo (2011). ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 5027–5034. IEEE Computer Society Press, Los Alamitos, CA.

G. Pini, **A. Brutschy**, M. Birattari, and M. Dorigo (2009). Interference reduction through task partitioning in a robotic swarm. In J. Filipe, J. Andrade-Cetto, and J.-L. Ferrier, editors, *Proceedings of the Sixth International Conference on Informatics in Control, Automation and Robotics (ICINCO'09)*, pages 52–59. INSTICC Press, Setùbal, Portugal.

G. Pini, **A. Brutschy**, M. Birattari, and M. Dorigo (2011a). Task partitioning in swarms of robots: reducing performance losses due to interference at shared resources. In J. A. Cetto, J. Filipe, and J.-L. Ferrier, editors, *Informatics in Control, Automation and Robotics*, volume 85 of *Lecture Notes in Electrical Engineering*, pages 217–228. Springer, Berlin/Heidelberg, Germany. Part 3.

G. Pini, **A. Brutschy**, G. Francesca, M. Dorigo, and M. Birattari (2012). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, and T. Stützle, editors, *Proceedings of the 8th International Conference on Swarm Intelligence (ANTS'12)*, volume 7461 of *Lecture Notes in Computer Science*, pages 109–120. Springer, Berlin/Heidelberg, Germany.

G. Pini, **A. Brutschy**, M. Frison, A. Roli, M. Birattari, and M. Dorigo (2011b). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3–4):283–304.

G. Pini, **A. Brutschy**, C. Pinciroli, M. Dorigo, and M. Birattari (2013a). Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive Behavior*, 21(2):117–135.

G. Pini, M. Gagliolo, **A. Brutschy**, M. Dorigo, and M. Birattari (2013b). Task partitioning in a robot swarm: a study on the effect of communication. *Swarm Intelligence*, 7(2–3):173–199.

A. Scheidler, **A. Brutschy**, K. Diwold, D. Merkle, and M. Middendorf (2011). *Ant Inspired Methods for Organic Computing*, volume 1, pages 95–109. Springer, Berlin/Heidelberg, Germany. Part 1.

A. Scheidler, **A. Brutschy**, E. Ferrante, and M. Dorigo (2014). The k-unanimity rule for self-organized decision making in swarms of robots. *Systems, Man, and Cybernetics, Part B: Cybernetics*. Under review.

# Bibliography

Acerbi, A., Marocco, D., and Nolfi, S. (2007). Social facilitation on the development of foraging behaviors in a population of autonomous robots. In Almeida e Costa, F., Rocha, L., Costa, E., Harvey, I., and Coutinho, A., editors, *Advances in Artificial Life*, volume 4648 of *Lecture Notes in Computer Science*, pages 625–634. Springer, Berlin/Heidelberg, Germany.

Agrawal, R. (1995). Sample mean based index policies with $O(\log n)$ regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27:1054–1078.

Alers, S., Bloembergen, D., Hennes, D., de Jong, S., Kaisers, M., Lemmens, N., Tuyls, K., and Weiss, G. (2011). Bee-inspired foraging in an embodied swarm (demonstration). In *Proceedings of the 10th international joint conference on Autonomous agents and multiagent systems (AAMAS'11)*, pages 1311–1312. ACM Press, New York, NJ.

Anderson, C., Franks, N. R., and McShea, D. W. (2001a). The complexity and hierarchical structure of tasks in insect societies. *Animal Behaviour*, 62(4):643–651.

Anderson, C., Franks, N. R., and McShea, D. W. (2001b). The complexity and hierarchical structure of tasks in insect societies. *Animal Behaviour*, 62(4):643–651.

Anderson, C. and Ratnieks, F. L. W. (2000). Task partitioning in insect societies: Novel situations. *Insectes Sociaux*, 47(2):198–199.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis

of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Baldassarre, G., Parisi, D., and Nolfi, S. (2006). Distributed coordination of simulated robots based on self-organization. *Artificial Life*, 3(12):289–311.

Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007). Self-organized coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 37(1):224–239.

Banzhaf, W. and Pillay, N. (2007). Why complex systems engineering needs biological development. *Complexity*, 13(2):12–21.

Banzi, M. (2008). *Getting Started with Arduino*. O'Reilly Media, Sebastopol, CA.

Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1–2):173–215.

Beni, G. (2005). From swarm intelligence to swarm robotics. In Şahin, E. and Spears, W. M., editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer, Berlin/Heidelberg, Germany.

Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, volume 102, Tuscany, Italy. NATO Scientific Affairs Division.

Berman, S., Halász, Á., Hsieh, M. A., and Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937.

Berman, S., V., K., and R., N. (2011). Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'11)*, pages 378–385. IEEE Press, Pitscataway, NJ.

Beshers, S. N. and Fewell, J. H. (2001). Models of division of labor in social insects. *Annual Review of Entomology*, 46:413–440.

Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., and Mondada, F. (2010). The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pages 4187–4193. IEEE Press, Piscataway, NJ.

Brambilla, M., Brutschy, A., Dorigo, M., and Birattari, M. (2014). Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems*. In press.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Brambilla, M., Pinciroli, C., Birattari, M., and Dorigo, M. (2012). Property-driven design for swarm robotics. In *Proceedings of the 11th international joint conference on Autonomous agents and multiagent systems (AAMAS'12)*, pages 139–146. ACM Press, New York, NJ.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Robotics & Automation Magazine*, 2(1):14–23.

Brooks, R. A. (1987). Intelligence with representation. *Artificial Intelligence*, 47(1991):139–159.

Brooks, R. A. (1991). Intelligence without reason. In *Proceedings of 12th International Joint Conference on Artificial Intelligence*, pages 569–595. Morgan Kaufmann, Burlington, MA.

Brooks, R. A. (1992). Artificial life and real robots. In Varela, F. J. and Bourgine, P., editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 3–10. MIT Press, Cambridge, MA.

Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., and Birattari, M. (2014a). The TAM: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*. Under review.

Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., and Birattari, M. (2014b). The TAM: abstracting complex tasks in swarm robotics research. http://iridia.ulb.ac.be/supp/IridiaSupp2012-002/.

Brutschy, A., Pini, G., and Decugnière, A. (2012a). Grippable objects for the foot-bot. Technical Report TR/IRIDIA/2012-001, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., and Dorigo, M. (2014c). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 28(1):101–125.

Brutschy, A., Scheidler, A., Ferrante, E., Dorigo, M., and Birattari, M. (2012b). Can ants inspire robots? In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), Video Proceedings*. IEEE Press, Piscataway, NJ.

Brutschy, A., Scheidler, A., Merkle, D., and Middendorf, M. (2008). Learning from house-hunting ants: Collective decision-making in organic computing systems. In Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., and Winfield, A., editors, *Swarm Intelligence, Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence, (ANTS'08)*, volume 5217 of *Lecture Notes in Computer Science*, pages 96–107. Springer, Berlin/Heidelberg, Germany.

Brutschy, A., Tran, N.-L., Baiboun, N., Frison, M., Pini, G., Roli, A., Dorigo, M., and Birattari, M. (2011). Costs and benefits of behavioral specialization. In *Proceedings of the 12th Annual Conference on Towards Autonomous Robotic Systems (TAROS'11)*, volume 6856 of *Lecture Notes in Computer Science*, pages 90–101. Springer, Berlin/Heidelberg, Germany.

Brutschy, A., Tran, N.-L., Baiboun, N., Frison, M., Pini, G., Roli, A., Dorigo, M., and Birattari, M. (2012c). Costs and benefits of behavioral specialization. *Robotics and Autonomous Systems*, 60(11):1408–1420.

Campo, A., Garnier, S., Dédriche, O., Zekkri, M., and Dorigo, M. (2011). Self-organized discrimination of resources. *PLoS ONE*, 6(5):e19888.

Campo, A., Gutiérrez, Á., and Longchamp, V. (2010a). libIrcom: a library for inter-robot communication for the e-puck. http://gna.org/projects/e-puck/. Open-source software project.

Campo, A., Gutiérrez, Á., Nouyan, S., Pinciroli, C., Longchamp, V., Garnier, S., and Dorigo, M. (2010b). Artificial pheromone for path selection by a foraging swarm of robots. *Biological cybernetics*, 103(5):339–52.

Campo, A., Nouyan, S., Birattari, M., Groß, R., and Dorigo, M. (2006). Negotiation of goal direction for cooperative transport. In Dorigo, M., Gambardella, L., Birattari, M., Martinoli, A., Poli, R., and Stützle, T., editors, *Ant Colony Optimization and Swarm Intelligence*, volume 4150 of *Lecture Notes in Computer Science*, pages 191–202. Springer, Berlin/Heidelberg, Germany.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226–234.

Carlson, J., Murphy, R., and Nelson, A. (2004). Follow-up analysis of mobile robot failures. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, volume 5, pages 4987–4994. IEEE Press, Pitscataway, NJ.

Cassinis, R., Bianco, G., Cavagnini, A., and Ransenigo, P. (1999). Strategies for navigation of robot swarms to be used in landmines detection. In *Third European Workshop on Advanced Mobile Robots (Eurobot'99)*, pages 211–218. IEEE Press, Pitscataway, NJ.

Cassinis, R., Tampalini, F., Bartolini, P., and Fedrigotti, R. (2005). Docking and charging system for autonomous mobile robots. Technical report, Università degli Studi di Brescia, Brescia, Italy.

Castillo-Cagigal, M., Brutschy, A., Gutiérrez, Á., and Birattari, M. (2014). Temporal task allocation in periodic environments: An approach based on synchronization. In *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)*, volume 8667 of *Lecture Notes in Computer Science*, pages 182–193. Springer, Berlin/Heidelberg, Germany.

Chin, K. O., Teo, J., and Azali, S. (2009). Automatic generation of swarm robotic behaviors using multi-objective evolution. In

*Proceedings of the International Conference on Man-Machine Systems (ICoMMS'09)*, pages 1–6. Universiti Malaysia Perlis, Perlis, Malaysia.

Christensen, A. L., O'Grady, R., and Dorigo, M. (2007). Morphology control in multirobot system. *IEEE Robotics & Automation Magazine*, 14(4):18–25.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition*. MIT Press, Cambridge, MA.

Deneubourg, J.-L., Aron, S., Goss, S., and Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. *Insect Behavior*, 3:159–168.

Dias, M. B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270.

Donald, B. R., Jennings, J., and Rus, D. (1997). Information invariants for distributed manipulation. *Robotics Research*, 16(5):673–702.

Dorigo, M. and Birattari, M. (2007). Swarm intelligence. *Scholarpedia*, 2(9):1462.

Dorigo, M., Birattari, M., and Brambilla, M. (2014). Swarm robotics. *Scholarpedia*, 9(1):1463.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugnière, A., Di Caro, G., Ducatelle, F., Ferrante, E., Förster, A., Guzzi, J., Longchamp, V., Magnenat, S., Martinez Gonzales, J., Mathews, N., Montes de Oca, M., O'Grady, R., Pinciroli, C., Pini, G., Rétornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stützle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugnière, A., Di

Caro, G., Ducatelle, F., Ferrante, E., Förster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., de Oca, M. M., O'Grady, R., Pinciroli, C., Pini, G., Rétornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stützle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2011). Swarmanoid, the movie. In *AAAI-11 Video Proceedings*. AAAI Press, San Francisco, CA. Winner of the "Best Video" award.

Durfee, E. H. and Lesser, V. R. (1989). Negotiating task decomposition and allocation using partial global planning. In Huhns, M., editor, *Distributed Artificial Intelligence*, volume 2, pages 229–243. Morgan Kaufmann Publishers, San Francisco, CA.

Ferrante, E., Brambilla, M., Birattari, M., and Dorigo, M. (2013a). Socially-mediated negotiation for obstacle avoidance in collective transport. In Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M. A., Parker, L. E., and Støy, K., editors, *Proceedings of the International Symposium on Distributed Autonomous Robotics Systems (DARS'10)*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 571–583. Springer, Berlin/Heidelberg, Germany.

Ferrante, E., Turgut, A. E., Mathews, N., Birattari, M., and Dorigo, M. (2010). Flocking in stationary and non-stationary environments: a novel communication strategy for heading alignment. In *Parallel problem solving from nature: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN XI)*, volume 6239 of *Lecture Notes in Computer Science*, pages 331–340. Springer, Berlin/Heidelberg, Germany.

Ferrante, E., Turgut, A. E., Stranieri, A., Pinciroli, C., Birattari, M., and Dorigo, M. (2013b). A self-adaptive communication strategy for flocking in stationary and non-stationary environments. *Natural Computing*, 13:225–245.

Fontan, M. S. and Matarić, M. J. (1996). A study of territoriality: The role of critical mass in adaptive task division. In Maes, P., Matarić, M. J., Meyer, J.-A., Pollack, J., and Wilson, S., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pages 553–561. MIT Press, Cambridge, MA.

Fowler, H. H. and Robinson, S. W. (1979). Foraging by *Atta sexdens* (Formicidae: Attini): Seasonal patterns, caste and efficiency. *Ecological Entomology*, 4(3):239–247.

Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., Reina, A., Soleymani, T., Salvaro, M., Pinciroli, C., Trianni, V., and Birattari, M. (2014a). An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts. In Dorigo, M., Birattari, M., Garnier, S., de Oca, H. H. M. M., Solnon, C., and Stützle, T., editors, *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)*, volume 8667 of *Lecture Notes in Computer Science*, pages 25–37. Springer, Berlin/Heidelberg, Germany.

Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., and Birattari, M. (2014b). AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112.

Frison, M., Tran, N.-L., Baiboun, N., Brutschy, A., Pini, G., Roli, A., Dorigo, M., and Birattari, M. (2010). Self-organized task partitioning in a swarm of robots. In Dorigo, M., Birattari, M., Di Caro, G. A., Doursat, R., Engelbrecht, A. P., Floreano, D., Gambardella, L. M., Groß, R., Şahin, E., Sayama, H., and Stützle, T., editors, *Proceedings of the 7th International Conference on Swarm Intelligence (ANTS'10)*, volume 6234 of *Lecture Notes in Computer Science*, pages 287–298. Springer, Berlin/Heidelberg, Germany.

Fujisawa, R., Dobata, S., Sugawara, K., and Matsuno, F. (2014). Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246.

Garnier, S., Combe, M., Jost, C., and Theraulaz, G. (2013). Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test bed. *PLoS Computational Biolology*, 9(3):e1002903.

Garnier, S., Gautrais, J., and Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31.

Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *Robotics Research*, 23(9):939–954.

Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., and Stentz, A. (2003). Market-based multi-robot planning in a distributed layered architecture. In Shultz, A. C., Parker, L. E., and Schneider, F. E., editors, *From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, pages 27–38. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Goldberg, D. and Matarić, M. J. (2002). Design and evaluation of robust behavior-based controllers. In Balch, T. and Parker, L. E., editors, *Robot Teams: From Diversity to Polymorphism*, pages 315–344. A. K. Peters, Natick, MA.

Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76:579–581.

Grama, A., Karypis, G., Kumar, V., and Gupta, A. (2002). *Introduction to Parallel Computing*. Addison-Wesley, Boston, MA, 2nd edition.

Groß, R. and Dorigo, M. (2009). Towards group transport by swarms of robots. *Bio-Inspired Computation*, 1(1–2):1–13.

Gutiérrez, Á., Campo, A., Dorigo, M., Amor, D., Magdalena, L., and Monasterio-Huelin, F. (2008). An open localisation and local communication embodied sensor. *Sensors*, 11(8):7545–7563.

Hada, Y. and Yuta, S. (2001). A first-stage experiment of long term activity of autonomous mobile robot – result of repetitive base-docking over a week. In Rus, D. and Singh, S., editors, *Experimental Robotics VII*, volume 271 of *Lecture Notes in Control and Information Sciences*, pages 229–238. Springer, Berlin/Heidelberg, Germany.

Hamann, H. and Wörn, H. (2008). A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2):209–239.

Hart, A. G. and Ratnieks, F. L. W. (2001). Task partitioning, division of labour and nest compartmentalisation collectively isolate hazardous waste in the leafcutting *Atta cephalotes*. *Behavioral Ecology and Sociobiology*, 49:387–392.

Ijspeert, A. J., Martinoli, A., Billard, A., and Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171.

Jakobi, N. (1993). Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 348–357. MIT Press, Cambridge, MA.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behaviour*, 6(2):325–368.

Jakobi, N. (1998). *Minimal Simulations For Evolutionary Robotics*. PhD thesis, University of Sussex, Brighton, UK.

Jakobi, N., A. Husbands, P., and A. Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In Morán, F., Moreno, A., Merelo, J. J., and Chacón, P., editors, *Swarm Robotics*, volume 929 of *Advances in Artificial Life*, pages 704–720. Springer, Berlin/Heidelberg, Germany.

Jeanne, R. L. (1986). The evolution of the organization of work in social insects. *Monitore zoologico italiano*, 20:119–133.

Kalra, N. and Martinoli, A. (2006). A comparative study of market-based and threshold-based task allocation. In *Distributed Autonomous Robotic Systems 7*, pages 91–102. Springer, Berlin/Heidelberg, Germany.

Kazadi, S. (2000). *Swarm engineering*. PhD thesis, California Institute of Technology, Pasadena, CA.

Kazadi, S., Lee, J. R., and Lee, J. (2009). Model independence in swarm robotics. *International Journal of Intelligent Computing and Cybernetics*, 2(4):672–694.

Kernbach, S., Nepomnyashchikh, V., Kancheva, T., and Kernbach, O. (2012). Specialization and generalization of robotic behavior in swarm energy foraging. *Mathematical and Computer Modelling of Dynamical Systems*, 18:131–152.

Knuth, D. (1997). *The art of computer programming: Fundamental Algorithms*, volume 1. Addison-Wesley, Boston, MA, 3rd edition.

Konur, S., Dixon, C., and Fisher, M. (2012). Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2):199–213.

Kopacek, P. (2004). Robots for humanitarian demining. In Voicu, M., editor, *Advances in Automatic Control*, volume 754 of *The Springer International Series in Engineering and Computer Science*, pages 159–172. Springer, Berlin/Heidelberg, Germany.

Korsah, G. A. (2011). *Exploring bounded optimal coordination for heterogeneous teams with cross-schedule dependencies*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Korsah, G. A., Stentz, A., and Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *Robotics Research*, 32(12):1495–1512.

Krieger, M. J. B. and Billeter, J.-B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84.

Kube, C. and Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1-2):85–101.

Kube, C. R. and Zhang, H. (1993). Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–219.

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labour in a group of robots inspired by ants' foraging behaviour. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25.

Lein, A. (2010). *Adaptive foraging in robotic swarms*. PhD thesis, Simon Fraser University, Burnaby Mountain, Canada.

Lein, A. and Vaughan, R. (2008). Adaptive multi-robot bucket brigade foraging. In *Proceedings of the Eleventh International Conference on Artificial Life (ALife XI)*, pages 337–342. MIT Press, Cambridge, MA.

Li, L., Martinoli, A., and Abu-Mostafa, Y. S. (2004). Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior*, 12(3-4):199–212.

Liu, F. and Picard, R. W. (1998). Finding periodicity in space and time. In *Proceedings of the 6th International Conference on Computer Vision*, pages 376–383. IEEE Computer Society, Los Alamitos, CA.

Magnenat, S., Philippsen, R., and Mondada, F. (2012). Autonomous construction using scarce resources in unknown environments - Ingredients for an intelligent robotic interaction with the physical world. *Autonomous Robots*, 33:467–485.

Martín H., J. A., de Lope, J., and Maravall, D. (2009). Adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature. *Natural Computing*, 8(4):757–775.

Martinoli, A., Easton, K., and Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *Robotics Research*, 23:415–436.

Matarić, M. J., Sukhatme, G. S., and Østergaard, E. H. (2003). Multi-robot task allocation in uncertain environments. *Autonomous Robots*, 14:255–263.

Mathews, N., Stranieri, A., Scheidler, A., and Dorigo, M. (2012). Supervised morphogenesis – Morphology control of ground-based self-assembling robots by aerial robots. In Conitzer, Winikoff, P. and van der Hoek, editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, pages 97–104. ACM Press, New York, NJ.

Mathews, N., Valentini, G., Christensen, A. L., O'Grady, R., Brutschy, A., and Dorigo, M. (2014). Spatially targeted communication in decentralized multirobot systems. *Autonomous Robots*. Under review.

McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., and Schmidt, B. (2006). Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75. Association for the Advancement of Artificial Intelligence, Menlo Park, CA.

Miller, J. and Mukerji, J. (2003). Model driven architecture: guide. http://www.omg.org. Version 1.0.1.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J. C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In Gonçalves, P. J. S., Torres, P. J. D., and Alves, C. M. O., editors, *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65. IPCB: Instituto Politècnico de Castelo Branco, Castelo Branco, Portugal.

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. V., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). SWARM-BOT: A new distributed robotic concept. *Autonomous Robots*, 17(2–3):193–221.

Nitschke, G., Schut, M., and Eiben, A. (2007). Emergent specialization in biologically inspired collective behavior systems. In *Intelligent Complex Adaptive Systems*, chapter 8, pages 215–253. IGI Publishing, New York, NJ.

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics*. MIT Press, Cambridge, MA.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711.

OMG (2011). Object Management Group Unified Modeling Language (OMG UML), Superstructure. http://www.omg.org. Version 2.4.1.

Orians, G. H. and Pearson, N. E. (1979). *On the theory of central place foraging*, pages 155–177. Ohio State University Press, Columbus, OH.

Østergaard, E., Sukhatme, G., and Matarić, M. J. (2001). Emergent bucket brigading. *Autonomous Agents*, 37:2219–2223.

O'Grady, R., Groß, R., Christensen, A., and Dorigo, M. (2010). Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28(4):439–455.

Parker, C. A. C. and Zhang, H. (2010). Collective unary decision-making by decentralized multiple-robot systems applied to the task-sequencing problem. *Swarm Intelligence*, 4(3):199–220.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240.

Payton, D., Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone robotics. *Autonomous Robots*, 11(3):319–324.

Petri, C. A. and Reisig, W. (2008). Petri net. *Scholarpedia*, 3(4):6477.

Pinciroli, C. (2014). *On the Design and Implementation of an Accurate, Efficient, and Flexible Simulator for Heterogeneous Swarm Robotics Systems*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L. M., and Dorigo, M. (2012). ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Stirling, T., Gutiérrez, Á., Gambardella, L. M., and Dorigo, M. (2011). ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 5027–5034. IEEE Computer Society Press, Los Alamitos, CA.

Pini, G. (2013). *Towards Autonomous Task Partitioning in Swarm Robotics - Experiments with Foraging Robots*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Interference reduction through task partitioning in a robotic swarm. In Filipe, J., Andrade-Cetto, J., and Ferrier, J.-L., editors, *Proceedings of the Sixth International Conference on Informatics in Control, Automation and Robotics (ICINCO'09)*, pages 52–59. INSTICC Press, Setùbal, Portugal.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2011a). Task partitioning in swarms of robots: reducing performance losses due to interference at shared resources. In Cetto, J. A., Filipe, J., and Ferrier, J.-L., editors, *Informatics in Control, Automation and Robotics*, volume 85 of *Lecture Notes in Electrical Engineering*, pages 217–228. Springer, Berlin/Heidelberg, Germany. Part 3.

Pini, G., Brutschy, A., Francesca, G., Dorigo, M., and Birattari, M. (2012). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In Dorigo, M., Birattari, M., Blum, C., Christensen, A. L., Engelbrecht, A. P., Groß, R., and Stützle, T., editors, *Proceedings of the 8th International Conference on Swarm Intelligence (ANTS'12)*, volume 7461 of *Lecture Notes in Computer Science*, pages 109–120. Springer, Berlin/Heidelberg, Germany.

Pini, G., Brutschy, A., Frison, M., Roli, A., Birattari, M., and Dorigo, M. (2011b). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3–4):283–304.

Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., and Birattari, M. (2013a). Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive Behavior*, 21(2):117–135.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., and Birattari, M. (2014). Task partitioning in a robot swarm: Retrieving objects by transferring them directly between sequential sub-tasks. *Artifical Life*, 20(3):291–317.

Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., and Birattari, M. (2013b). Task partitioning in a robot swarm: a study on the effect of communication. *Swarm Intelligence*, 7(2–3):173–199.

Purnamadjaja, A. H. and Russell, R. A. (2007). Guiding robots' behaviors using pheromone communication. *Autonomous Robots*, 23(2):113–130.

Ratnieks, F. L. W. and Anderson, C. (1999). Task partitioning in insect societies. *Insectes sociaux*, 46(2):95–108.

Reisig, W. (2013). *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Springer, Berlin/Heidelberg, Germany.

Rosu, D., Schwan, K., Yalamanchili, S., and Jha, R. (1997). On adaptive resource allocation for complex real time applications. In *Proceedings of the 18th Real-Time System Symposium*, pages 320–329. IEEE Computer Society, Los Alamitos, CA.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'12)*, pages 3293–3298. IEEE Press, Pitscataway, NJ.

Rubenstein, M., Cabrera, A., Werfel, J., Habibi, G., McLurkin, J., and Nagpal, R. (2013). Collective transport of complex objects by simple robots: Theory and experiments. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, pages 47–54. ACM Press, New York, NJ.

Rubenstein, M., Hoff, N., and Nagpal, R. (2011). Kilobot: A low cost scalable robot system for collective behaviors. Technical Report TR-06-11, Computer Science Group, Harvard University, Cambridge, MA.

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, Upper Saddle River, NJ.

Sangiovanni-Vincentelli, A. (2003). The tides of eda. *IEEE Design & Test of Computers*, 20(6):59–75.

Scheidler, A., Brutschy, A., Diwold, K., Merkle, D., and Middendorf, M. (2011). *Ant Inspired Methods for Organic Computing*, volume 1, pages 95–109. Springer, Berlin/Heidelberg, Germany. Part 1.

Scheidler, A., Brutschy, A., Ferrante, E., and Dorigo, M. (2014). The k-unanimity rule for self-organized decision making in swarms of robots. *Systems, Man, and Cybernetics, Part B: Cybernetics*. Under review.

Serfozo, R. F. (1979). An equivalence between continuous and discrete time markov decision processes. *Operations Research*, 27(3):616–620.

Shell, D. and Matarić, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, pages 2717–2723. IEEE Press, Pitscataway, NJ.

Silverman, M. C., Nies, D., Jung, B., and Sukhatme, G. S. (2002). Staying alive: a docking station for autonomous robot recharging. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pages 1050–1055. IEEE Press, Piscataway, NJ.

Smithers, T. (1994). On why better robots make it harder. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats (SAB'94)*, pages 64–72. MIT Press, Cambridge, MA.

Smithers, T. (1997). Autonomy in robots and other agents. *Brain and Cognition*, 34(1):88–107.

Spiteri Staines, A. (2010). *Petri Nets Applications*, chapter Intuitive Transformation of UML2 Activities into Fundamental Modeling Concept Petri Nets and Colored Petri Nets, pages 673–694. InTech Europe, Rijeka, Croatia.

Stevenson, A., editor (2010). *Oxford Dictionary of English*. Oxford University Press, Oxford, UK, 3rd edition.

Störrle, H. (2000). *Models of Software Architecture - Design and Analysis with UML and Petri-Nets*. PhD thesis, LMU München, Institut für Informatik, Munich, Germany.

Stranieri, A., Turgut, A. E., Francesca, G., Reina, A., Dorigo, M., and Birattari, M. (2013). IRIDIA's Arena Tracking System. Technical Report TR/IRIDIA/2013-013, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning, an Introduction*. MIT Press, Cambridge, MA.

Tanenbaum, A. S. (2007). *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, 3rd edition.

Tuci, E., Groß, R., Trianni, V., Mondada, F., Bonani, M., and Dorigo, M. (2006). Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems*, 1(2):115–150.

Tuci, E., Trianni, V., and Dorigo, M. (2004). 'Feeling' the flow of time through sensory-motor coordination. *Connection Science*, 4(16):301–324.

Werfel, J., Petersen, K., and Nagpal, R. (2014). Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758.

Wooldridge, M. and Jennings, N. R. (1998). Pitfalls of agent-oriented development. In *Proceedings of the second international conference on Autonomous agents*, pages 385–391. ACM Press, New York, NJ.

Zafar, K., Qazi, S., and Baig, A. R. (2006). Mine detection and route planning in military warfare using multi agent system. In *30th Annual International Computer Software and Applications Conference (COMPSAC '06)*, volume 2, pages 327–332. IEEE Press, Pitscataway, NJ.

Zlot, R. M. (2006). *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.