Graph-based World Modelling for Composable Cognitive Robotics

A First Exploration in a Greenhouse Environment

Herremans, B.R.

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Control Systems Technology

Master Thesis

Graph-based World Modelling for Composable Cognitive Robotics: A First Exploration in a Greenhouse Environment

FlexCRAFT Project

B.R. Herremans

Student number: 0970880
CST number: CST2021.061

Supervisors:
Dr. ir. M.J.G. van de Molengraft
Prof. dr. ir. H.P.J. Bruyninckx

Advisor:
ir. J.P.F. Senden PDEng

Final Version

Eindhoven, September 2021

Abstract

In the agro-food industry the decreasing amount of skilled personnel for crop harvesting, caused by harsh and undesirable working conditions, is the main motivation for automation of harvesting processes. Automation is especially challenging for the task of harvesting tomato trusses in the greenhouse, because it takes place in an open-world environment with many possible variations. In this environment, a system design is not able to foresee all possible variations and prescribe all required robot actions at design time. Nevertheless, most research in harvesting robotics still adopt a design approach which tries to capture these situations at design time.
This work explores a new method which introduces a graph-based world model that captures the semantic relationship between objects. In our approach the robots' course of action is not predefined at design time, but deliberated on real-time. A reasoning methodology is applied on a symbolic representation of the environment, to base this course of action on available knowledge about the current situation. A first investigation of applying such a symbolic world model is done by means of a simulation using Apache Gremlin to construct and query the graph. The results show that decisions can indeed be based on updated information available in the world model. We address our main lessons learned form this explorative implementation, pointing out the need for a more expressive representation of the world model and conditions queried for. This raises the challenge of handling multi-hypothesis of obtained information, inexact sub-graph matching to base decisions on, assessing the origin of information, and investigating integration with automated planning.

Contents

1 Introduction

The amount of skilled personnel for crop harvesting is decreasing significantly over the years due to repetitive, harsh and heavy tasks in undesirable greenhouse climate conditions. These labour shortages are a major problem in the agro-food industry, while food demand is ever-growing. This is the main stimulus for the sector to advance and utilize automation processes. This transition is further driven by major opportunities to increase production efficiency, enhance hygiene standards, increase product quality, positively impact yield margins, and reduce labour costs [1], [2].
Within certain domains of the agro-food industry, automation is implemented successfully. Single harvesting crops such as maize or wheat are harvested by combine harvesters which remove plants from the field and separate the grains. To eliminate the need for human operation, efforts are made to develop full autonomous combine systems [3]–[5]. Such robotic systems operate in partly controlled closed-world environments with little or negligible variations. Moreover, there are a few possible interactions with these objects and the system is assigned with a single repetitive task, causing these processes to be suitable for automation. Most autonomous robots operate in such environments, for instance the car manufacturing industry, where products are moved by robot arms with high precision but repeatedly in the same manner.
However, when variation increases to a more complex open-world environment with a range of tasks and multiple possible interactions, the deployment of autonomous robotics becomes more challenging. Such an open-world environment is encountered in a greenhouse where plants grow with natural diversity causing their fruits and other plant parts to vary in shape, size, position, and reflectance. For example, the tomato plants in figure 1.1 have distributed truss positions, and each truss holds a different amounts of tomatoes which vary from rounded to more stretched shapes. Due to the dense positioning of plants in a greenhouse, surrounding plant parts often cause poor visibility and accessibility of fruits. This requires alternative ways of interacting with the plant to harvest its fruits. Additionally, work in the greenhouse consists of a variety of tasks throughout the season, such as harvesting trusses, storing trusses, de-leafing plants, and positioning plants along the hire-wire.

Figure 1.1: Tomato plants in greenhouse Vereijken [6]

The complexity of this environment and variety of tasks is the main reason autonomous harvesting robots are not yet implemented successfully. Confirming this challenge is the research over the past decades on harvesting robotics for a variety of crops. The review of *Bac et al.* [7] compares research development throughout 30 years, evaluating complete harvesting robotic systems

as well as systems devoted to sole navigation, localization, or manipulation tasks. While several performance indicators are used, the major indicator of system performance is the overall success rate of harvesting fruits. On average, systems reported a success rate of 66%. Later developed harvesting robotic research reported similar success rates[8], [9]. Despite enhancements of robotic hardware and software and an increasing amount of research projects in the field, this success rate has, interestingly enough, hardly increased [7], [10].

This limiting performance and stagnation in development can be addressed to the fact that in the open-world environment a system designer is not able to foresee all possible variations the robot can encounter and prescribe all required actions the system has to perform to accomplish a task, at design time [11]. Autonomous robots operating in such open-world environments, assigned with a range of tasks and multiple possible interactions with their environment need deliberate acting. This means that robot actions are motivated through reasoning concerning the current situation and the objective. Deliberation has to be performed by the robot itself through interpreting the situation and reasoning about its course of action online. This requires robot actions to be composed automatically, which implies a decomposition of the task and how the robot solves the task by itself.
A vast majority of greenhouse robotic applications adopt the *sense-plan-act* paradigm [10] which captures deliberation at design time. This is an open-loop control paradigm where a robot consecutively uses its sensors to detect objects and construct a world model, after which it uses this world model to plan its next action, and finally executes this action. To illustrate this paradigm we consider the example of reaching a truss of tomatoes, to thereafter harvest it. First, the *sense* step is performed where the robot uses a camera to detect the tomato plant. When a truss of tomatoes and other plant parts are successfully located, the second *plan* step is carried out. Here the robot plans a feasible non-colliding path its end-effector can use to reach the truss of tomatoes. Finally, this action is carried out by moving the end-effector along the planned path. The inherent problem with this paradigm is twofold. Due to the series connection of components, each component's performance is highly dependent on the performance of its prior, causing accumulating errors towards the final result. As such, in the example of reaching the truss of tomatoes, the overall success rate is dependent on the multiplication of each individual success rate. This causes overall performance to be lower than the performance of each individual step. In addition, during acting the paradigm does not allow for reactive behaviour to unexpected events. This is required when the environment changes, for instance interaction with the plant during execution can result in dislocation of the truss causing the action to fail.
To resolve the problems faced we relate to how humans approach these kinds of tasks in an open-world environment. Firstly, as humans we continuously integrate our senses to achieve a general belief of the world around us. We do not rely on a single perception as we see, feel, smell and hear at the same time. For instance, when assigned to the task of harvesting this allows us to alternate between vision and touch overcoming possible occlusions of objects in the environment. Secondly, we evaluate our belief of the world with prior knowledge about known variations in objects and expected situations in the environment, such as the general fixed structure of plants. Hence, we can complete our belief of a partially detectable world and become aware of the situation we are in. Thirdly, humans solve a task by interpreting the situation and arguing which of our capabilities can be used, based on know-how and experience. When assigned a new task we deliberate on what course of action we should take to accomplish it, given the current situation. As such, one may use a different technique to harvest a truss depending on our situation, capabilities, and experiences. These three concepts of *multi-sensing*, *situational awareness* and *deliberation* can be related to research in the field of autonomous robotics to potentially improve the performance of harvesting robots. We shortly cover each concept potential.

1.1 Multi-sensing

Within the field of harvesting robotics detection techniques relying on vision-based sensors, i.e. camera's, is undoubtedly the most used and developed [12]. However, robots should not only

rely on a single sensor and benefits on multi-modal sensing to enhance robustness of detection techniques. Senden *et al.* [13] proposes a supplementary sensing method for fruit localization by measuring dynamic responses of tomato plants upon excitation. It shows that upon excitation of the stem, the relative position of the truss can be deduced. By reducing the dependency on sole vision, such fusion of multiple sensor techniques can increase robustness to unexpected variations in objects or occlusions. Sensing capabilities based on different physical phenomena are to be integrated into a generic world model to conclude on multiple sensor information.

1.2 Situational Awareness

A harvesting robot is not required to accurately know the exact geometry and precise orientation of the plant and truss to reason about a possible approach to harvest it. Being aware of the general structure of a plant with its trusses allows the robot to predict at which position along the stem trusses can be expected. It can assume trusses to be ripe at the bottom of the stem and that the amount of ripe tomatoes in a truss is decreasing further up along the stem. Moreover, it knows that trusses are connected to the stem with a peduncle and the same holds for leaves. As a result, when detection of a truss is occluded by leaves, for instance, the robot can reason about an alternative approach where it can track the stem upwards until a side-shoot is found. Identifying these as a truss or a leaf then motivates for de-leafing or harvesting. This approach is referred to as the main-stem paradigm [14]. Such high-level reasoning about the environment based on prior knowledge and predictions is essential to deliberate on the appropriate approach. Therefore, the way a robot constructs its world model must facilitate this deliberation, allowing for logical inferences and predicting on a symbolic level. However, robotics typically adopt high-resolution world models with a specific representation of obtained sensor data to supply semantic information for a specific action or task. As such, these world models lack meaning for other interactions with the environment the robot may need to achieve its goal. Therefore, the obtained semantic information from such models is to be abstracted into a symbolic world model, integrating information from different (prior) sources. This world model accommodates interpretation of the environment, referred to as *situational awareness*: 'the perception of object and events with respect to time or space, the comprehension of their meaning, and the projection of their future status' [15]. The following three aspects form the main fundamental for the comprehension of the meaning of perceived objects in the environment.

In the first place, the aforementioned prior knowledge about objects in the environment has to be available to the robot. The use of symbolic knowledge representations is investigated in other fields too, especially in research on service robots in domestic environments [16]. It improves robustness to incomplete information caused by detection failures since missing information can be completed based on prior knowledge. Comparing knowledge representation models is still an open problem as there is no consensus on the way humans store knowledge to process information either. However, ontologies are considered a useful tool to formalize semantic knowledge by describing objects, properties, and relationships within a domain. An example is KnowRob [17] where ontologies are used to reason about possible object locations in the domestic domain according to their mutual relationships. For example, if a cup is used to hold a drink, and drinks are poured in the kitchen, the cup could be in the kitchen. As such, a series of actions to serve a drink should include navigation to the kitchen to find the cup, motivated by knowledge form the ontology. Thereafter, KnowRob models manipulation actions by coupling robot capabilities to object affordances. However, referring to predefined action models does not allow for reasoning on *how* to perform these manipulations. Take, for example, the affordance of a *graspable* cup modelled as a single relation between the cup and the robot end-effector. The robot's ability to grasp the cup is not always true and is dependent on the situation. Neither is the *way* of grasping fixed, dependent on the possibilities a cup can be grasped by squeezing the cup, holding the handle, or maybe lifting it from underneath will suffice. In an open-world environment where these situations vary each time, arguing which *way* of grasping a cup can be essential to accomplish a task. To maintain such ontologies RoboBrain [18] is an example of an processing engine which learns and shares knowledge from several internet sources. Although this produces a very generic knowledge

base, it showcases maintaining and updating such ontologies by merging and splitting matching pieces of information.
Secondly, closely related to inclusion of prior knowledge with ontologies is semantic mapping. Semantic mapping takes semantic concepts of objects in the environment into account [16]. This includes task-relevant features such as behavioural or constraining properties. In the domestic environment example, robot navigation based on geometry is supplemented with semantic knowledge which benefits autonomy since it allows robots to draw their own conclusion on where to go. Naik *et al.* (2019) [19] proposes a graph-based semantic mapping approach capturing semantic, topological, and geometrical information for indoor structures such as corridors, walls, and doors. In the navigation context, these are behavioural properties of areas such as non-blocking areas in front of doors or semantics similar to traffic systems where taking a right turn requires getting in the most right lane area of a corridor. Although less intuitive, in the greenhouse such behavioural and constraining properties are applicable too. For instance, certain areas of the plant are to be harvested or de-leafed. Trusses of tomatoes are to be handled with care, while pruned leafs may be damaged.
The third essential element to obtain situational awareness is the use of a reasoning methodology to extract needed information from the world model, including the prior semantic knowledge and sensor information. Thereafter, it can conclude on its consequences for the robot. This methodology consist of a set of solving queries to answer to an overall inquiry. For instance, in the harvesting use case the question could be if an truss of ripe tomatoes can be expected behind a collection of leafs.

1.3 Deliberation

To deliberate on the course of action to take, the robot combines the reasoning methodology to be aware of its situation and matches it with its capabilities to conclude on the next action to perform. In robotic systems designed for closed-world environments with a limited amount of tasks and interactions with the environment, this deliberation is performed by the engineer at design time. In this case, the deliberation process is captured in, for example, a Finite State Machine [20], PetriNets [21] or Behaviour Trees [22] with specific situations, tasks and capabilities in mind. Here the desired behavior is captured by describing fixed response actions after the occurrence of a specific monitored event. However, with increasing variations in an open-world environment the amount of possible solutions increases and it becomes harder to foresee all possible scenarios. In this case these approaches are limiting. A main aspect of configuring a course of action online is planning a set of actions to reach the goal. An example of a planning technique is the Stanford Research Institute Problem Solver (STRIPS) [23] which is the basis of planning languages such as the Action Description Language [24] ADL and Planning Domain Definition Language PDDL [25]. These use predefined primitive actions a robot can execute to manipulate the world. These actions are described by pre- and post-conditions and matched to construct a sequence of actions which composes a plan. This plan is the solution which brings the world from the initial state to the desired state. However, this approach requires all robot actions to be deterministic, while in the open-world environment this is not always the case. Here actions can be probabilistic having multiple situation in which they can be applied, multiple ways how they can be used, and multiple possible outcomes. Moreover, these designed deterministic action often lack generality, since they are tailored to a specific task and situation. To obtain a composable deliberate system which uses its knowledge about the task, the situation and its capabilities, the task is to be described separate form the robotic system. This requires robot actions to be modeled without a specific task in mind.
More expressive planning methods rely on symbolic knowledge bases that capture the engineers knowledge about how to compose a successful plan. An example is Hierarchical Task Networks HTN [26] which uses a hierarchical structure to decompose the task into smaller sub-tasks, until a set of executable robot actions is obtained. Although this planning approach allows for refining tasks to a concrete level when demanded online, it is still restricted by the actions and explicit pre-conditions defined during deliberation at design time. The general knowledge the engineer

has about manipulating the world to fulfil a task is still applied to foreseen situations. As such, this implemented domain knowledge is not responsive to unexpected changes in the open world environment and cannot adapt its task composition. Therefore, to create an composable deliberate system the general domain knowledge a system designer would use must be captured and modeled explicitly. With this knowledge available to the robot, independent of task or situation, it can draw its own conclusion on which course of action to take.

To continue on the previously described examples in the harvesting use case, when leafs block the robots vision to detect a truss of tomatoes it has to deliberate about an alternative approach. Rather then prescribing what to do in this situation, the robot has to argue where it would expect a truss to be present and consider which other sensing capabilities it has at its disposal. Moreover, it can reason about other approaches such as the main-stem paradigm. In either of the cases the decision for a course of action is motivated by the situation and robot's capabilities. This motivation is referred to as the explainability of the decision making process.

Realizing such automated planning, acting and monitoring is still an open problem [27]. The open-world environment is never fully observable causing the world model to contain incomplete information, thus never be completely reliable. Therefore, the world model reliability is to be assessed and robots should reckon with the non-deterministic nature of its actions which can have unexpected outcomes during execution [28]. Consequently, planning is an integral part of deliberation, though is outside the scope of this project where we focus on motivating choices.

1.4 Design Process

When the robot is introduced to a new task it has to repeat the aforementioned deliberation process on the situation and its capabilities, but now relative to another objective. The design of a system which can perform this deliberation for a range of tasks can be related to concepts within the field of system engineering. Here numerous models for decomposition of the design process are guidance to the deliberation process of system designers and engineers. An example is the CAFCR model [29] which focuses on the decomposition of the objective defined by the customer and the technical possibilities of the producer. The objective is defined by a set of requirements and the physical properties or limitations of a product defined in system requirements. The design of the product couples the product functionalities to the customer objectives. This is the inspiration for decomposition of the robots capabilities and the task respectively. As the task exist independent of how it is accomplished by the robot, decomposition of the task and robot like in the CARFCR model is essential. This gives flexibility in execution without unintentional restrictions or wrongly predicted approaches.

1.5 Objectives and Outline

Deliberative autonomous harvesting robots require to reason about the situation and motivate possible courses of actions. To this end, we aim to decompose robot and task such that design is focused on modular robot capabilities and not on prescribing when to execute actions by foreseeing the situation and task. To couple the robot capabilities to the task and situation at hand, a reasoning methodology is required. The representation of information available to the robot, the world model, is to be designed such that it accommodates reasoning. Hence, we aim to adopt graph structures to construct the world model on a symbolic level. Here we focus on the relationship between available information and allow the robot to reason about its implications. As such, we set the following objectives:

- Define a graph-based world model and examine which reasoning question are required to motivate robot actions.
- Investigate how this world model and reasoning can be formalized by an exploitative implementation of this method.

- Achieve a method to define system-independent tasks, non-restrictive to how the system should solve it.

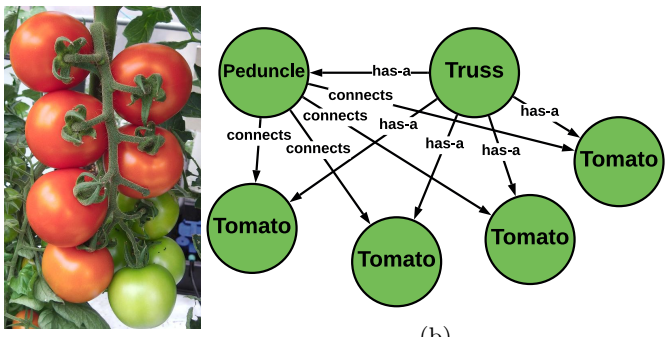
In this document we will first describe how we define the world model utilizing graphs in chapter 2. We investigate how to implement this world model and query it by means of an explorative simulation in chapter 3. With explorative we mean that the goal of the simulation is not to present a working solution but to gain insight in conclusions drawn on query results. From the results we discuss working solutions and limiting problems to support and improve the proposed method in chapter 4. We summarize our findings and recommendations for future research in chapter 5.

2 General Method

In this chapter we propose a general method to utilize graphs as a world model description. We specify how the world model is construed by means of examples and discuss what reasoning over this graph entails.

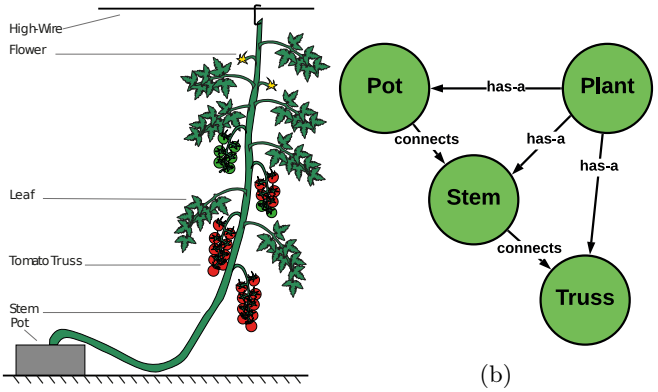
Constructing a symbolic world model includes the objects in the environment with their geometric and physical properties. Properties of objects contain a value and a corresponding unit or other model representation. For example, the mass property of an object can have a value and the unit kilograms. Whereas the shape of an object can be represented by a bounding box or a mesh model of the object. Objects in the environment have relations between them, for instance mereological, topological, topographic, and physical relations. This constructs a semantic web representing the relationship of environmental objects. Mereological relations represent the hierarchical structure of the object in the world since it represents of which the whole consists. The mereological [1] relation is abbreviated to the **has-a** relation, with its inverse being the **part-of** relation. Topological [2] relations represent object connections, denoted by the symmetric **connects** relation. Relations can also refer to a model, such as the topographic relation which refers to a model capturing the spatial distance between objects within a specified reference frame. For instance, a vector represents a distance in a Cartesian coordinate system. A physical relation refers to a model describing the force balance or dynamic behaviour between objects. An example is a free body diagram, describing the force balance between a truss of tomatoes held by a robot gripper, see Appendix. As such, the symbolic world model relations functions as a meta-model of the models used in the robotic system.

Construction of the world model graph is thus concerned with objects, relations and their properties. This is the motivation to choose a Labeled Property Graph (LPG) [30], [31] to construct the world model, using vertices (i.e. nodes) to represent objects and edges to represent relations. This allows both the vertices and the edges to contain properties, as opposed to for example Resource Description Framework (RDF) graphs where edge cannot contain properties [32]. The formulation used in this work is explained in chapter 3.



(a) (b)

Figure 2.1: Picture of a truss of tomatoes (a) represented in a graph (b)



(a) (b)

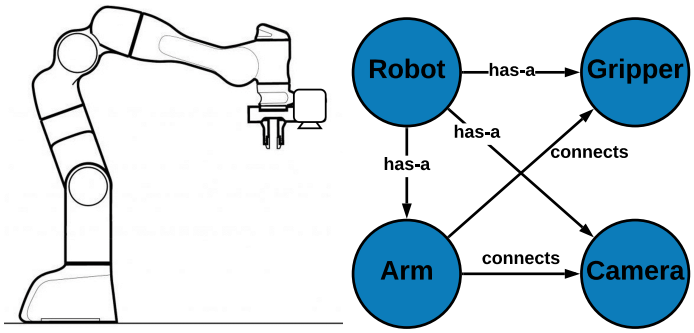
Figure 2.2: Schematic of a tomato plant (a) represented in a graph (b)

To illustrate the use of these relations for representing the world on a symbolic level, we address the following examples. A truss of tomatoes, shown in figure 2.1a, can be modelled in a graph

[1] Mereology concerns the relations between parts and wholes

[2] Topology is concerned with the properties of a geometric object that are preserved under continuous deformations

as depicted in a simplified sketch in figure 2.1b. Here we denote that the peduncle is connected with the tomatoes and that these objects are part of the truss. These relations and objects are part of the prior knowledge we have about objects we expect to encounter in the greenhouse. This truss is part of a tomato plant in a greenhouse as schematically shown in figure 2.2a. This plant is modelled as a graph containing this truss of tomatoes which is connected to a stem, as shown in figure 2.2b. This stem is connected to the pot on the ground and these objects are part of the same plant. As such, the a symbolic world model is constructed containing levels of mereological abstraction. To supplement or verify the world model with the actual instance of the objects in the environment, a robot uses its sensors for detection. For instance, with a camera parts of the plant can be detected and analysed to denote the amount of tomatoes the truss contains and their properties. As such, the world model graph is used to link obtained information about the environment regarding our general knowledge of the objects in the domain.
Beside objects in the environment, robot components are also modelled in the world model using the same type of relations. In figure figure 2.3a a schematic representation of an example robot is shown and its components can be modeled as depicted in figure 2.3b. This robot has a gripper and a camera which are connected to its arm. Here the arm of the robot can be divided into links and joints, similar to dividing the truss into peduncle and tomatoes.



(a) (b)

Figure 2.3: Schematic drawing of a Franka Emika [33] robot with a camera mounted on top (a) represented in a graph (b)

We define a task as the changes to the current state of the world model to a desired state of the world model. For example, when we consider the task of harvesting a truss of tomatoes this involves separation of the truss from the plant after which it is stored in a box. This means that the initial state of the world model must contain a truss which is connected to the stem. This is denoted as the pre-condition. In the desired state the world model no longer has this connection and a connection is obtained between the truss and the box. This is the post-condition. This is a change in the relationships between objects in the environment referred to as the Environment Model (EM) subset of the world model (WM).
The remaining subset of all robot components in the world model is called the *Robot Model* (RM). A skill is defined as the collective of software components that either processes sensor data, generates actuator commands or a combination of the two. As such, it can obtain information about the environment or control actuators to achieve changes in the desired way. The latter, changing relations between system object in the RM, such that object relations in the EM change. We define skills by stating the pre-conditions that must hold to start its execution, and the predicted changes obtain after execution in its post-conditions.
The robot has to reason about which skills can be executed in the current situation to accomplish a task by validating these pre- and post-conditions in the world model graph. In the next chapter we describe an implementation to explore how this is done.

3 Implementation

In this chapter we describe a first step toward implementing the world model graph and querying it. The goal is to show how a fixed set of queries is used to base decisions on. Furthermore, it is explored how the proposed world model and this set of queries to traverse the world model graph can be formulated. The decision on which actions to take is based on the results of these queries, validating a set of conditions. This chapter will explain the considered use case, the assumptions made, formulation details and final result analysis.

3.1 Simulation Design

In this section we discuss which part of the harvesting task is considered, which skills are used to complete this task and what disturbances are introduced. We make assumptions and abstractions because the simulation is not designed to represent a complex real world scenario, but to mimic the results of decisions made based on a query result. As such, we can analyse if each decision made is desired.

3.1.1 Task

In order to harvest a truss of tomatoes, the truss is to be reached. Reaching the truss is the subject of this simulation which entail reducing the relative distance between the robot end-effector and the truss towards zero.

3.1.2 Skills

To accomplish the task we consider the following set of simplified skills the robot is able to execute. With the use of a camera a truss of tomatoes can be detected, by processing the images with the use of, for example, deep learning algorithms [10], [34], [35]. We name this the **look** skill which requires a target object as pre-conditions and obtains the relative distance to this object as a post-condition. The robot motion skill named **move** can change the position of the robot end-effector to a point in space. This requires a target position as a pre-condition and can reduce this relative distance as much as possible. Measuring the dynamic response upon excitation of the stem to obtain the distance to the peduncle [13], is abbreviated as the **shake** skill. This skill requires the robot to be at the stem, thus its relative distance to be the zero vector, assuming it makes contact to excite the stem as a pre-condition. Measuring the dynamic response upon excitation obtains the connection of the stem and the peduncle and the relative positioning of the peduncle to the robot, defined as its post-condition. Finally, to **follow** the stem the robot gripper is required to be at the stem and the stem to be connected to the peduncle. The relative location of this peduncle is needed to determine in which direction to follow the stem. After the stem is followed, the distance between the robot and the peduncle is reduced to the zero vector, representing that it has reached the peduncle. An overview of the pre- and post-conditions is given in table 3.1.

Table 3.1: Skill pre- and post-conditions

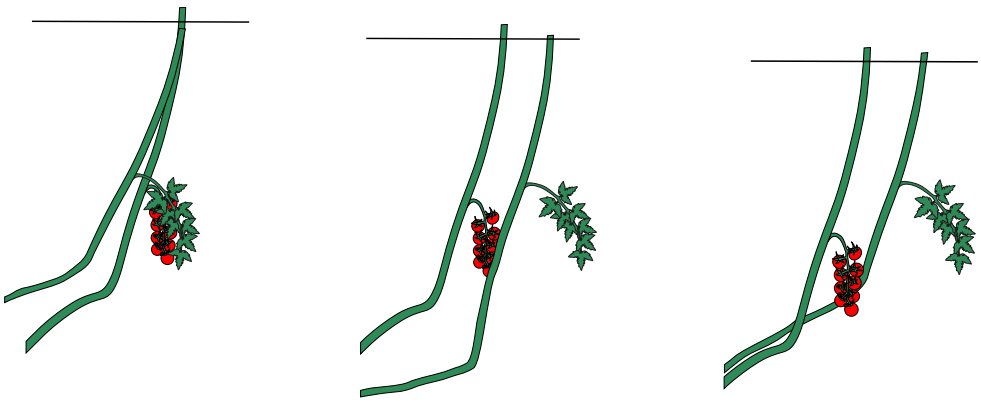
Skill	**Pre-condition**	**Post-condition**
Look	Target object	Distance robot-object
Move	Distance robot-object	Distance robot-object $\{'rel\_pos',\ [0,0,0]\}$
Shake	Distance robot-stem $\{'rel\_pos',\ [0,0,0]\}$	Topological relation stem-peduncle Distance robot-peduncle
Follow	Topological relation stem-peduncle Distance robot-stem $\{'rel\_pos',\ [0,0,0]\}$ Distance robot-peduncle	Topological relation stem-peduncle Distance robot-stem $\{'rel\_pos',\ [0,0,0]\}$ Distance robot-peduncle $\{'rel\_pos',\ [0,0,0]\}$

Given this set of skills there are two possible courses of action the robot can take; either **move** to the truss directly (plan A), or to **move** to the stem **follow** the stem to the peduncle and then **move** to the truss (plan B). Detection skills are not included in these plans, as they are only used when information is needed.

3.1.3 Disturbances

Through execution of the available action plans, supplemented with detection skills when required, the task can be accomplished. However, which detection or action plan should be executed is dependent on the situation the robot is in. In each situation the environment can contain different disturbances, which lead to a different course of action. These disturbances are the difference between the information the robot has available in its own world model and what the real world looks like. Figure 3.1 shows a picture of a greenhouse with tomato plants, from which we define the following disturbances. The first case is the free hanging tomato plant on the right side. This case is considered to have no disturbances. Occlusions are observed on the left side of the picture, where plants are in front of each other or intertwined. We distinguish three types of disturbances which are schematically shown in figure 3.2. The visual occlusion is caused by leaves handing in front of the truss as shown in figure 3.2a. This causes a detection of the truss to fail but is not necessarily blocking the motion path since leafs can move aside. Blocking the motion path to the truss is caused by another stem hanging in front of the truss, as shown in figure 3.2b. Intertwined plants, as shown in figure 3.2c, cause the skill of following of the stem to fail.

Figure 3.1: Greenhouse with tomato plants



(a) (b) (c)

Figure 3.2: Schematic representation of selected disturbances from the greenhouse use case. In (a) the truss is visually occluded by leafs causing truss detection to fail. In (b) the truss is blocked by a stem causing motion to the truss to fail. In (c) two plants are intertwined, causing following of the stem to fail.

Introducing a combination of these disturbances represents different scenarios, which are used to test the robots reasoning ability to motivate which actions to take.

3.1.4 Prior knowledge

When disturbances cause actions to fail, an alternative approach is to be considered. Reasoning about which alternative approach will be suitable is based on information about the environment and the expectation to encounter. For instance, if the vision of the truss is occluded it can be decided to estimate the position of the truss relative to another object that can be localized. Knowledge about the relative distance of the other object to the truss, prompts the decision to detect this related object and derive the desired truss position therefrom. To enable the robot with this reasoning step we provide an estimated positioning of the truss relative to the stem, modelled as a topographic relation. This relation can be used to employ the detection skill **look** on the stem and thereafter derive the resulting vector to the truss. In case the path to the truss is obstructed, a direct motion will fail. With the knowledge the truss is connected to the stem the robot can reason to follow the stem and find the truss instead. This knowledge is implemented as a topological relation describing that the truss is connected to the stem via the peduncle.

3.1.5 Query Machine

With the skills and prior knowledge available, the robot has to reason about which skill to execute to accomplish the task and overcome possible disturbances. This is done by querying the world model with a fixed set of questions. First, it assesses if the next action skill can be executed by finding its pre-conditions in the world model. In this case, both action skills have a pre-condition that requires to find a path over topographic relations from the robot to the object of interest. If there is no match, it checks whether the the missing information can be obtained with a detection skill and if its pre-conditions hold. If so, the detection is executed and its result is updated in the world model. A successful execution returns to the question if the pre-conditions of the action skill hold. If not successful or no detection was available, the world model is questioned if there is a related object which can be used to find an alternative path that represent the same relation. This requires the world model to be traversed [1] starting from the object of interest to an adjacent vertex over, in this case, topographic relations. When such an object is found, it is set to be the temporary object of interest and a candidate for an alternative detection. If none of the related objects can be detected, it concludes that the action skill cannot be executed and the missing information cannot be obtained. Hence, the robot argues if another approach can be used. The available alternative approach of following the stem is based on the connection of objects. Therefore, the world model is assessed for related objects over the topological relation type. If an object is found, a plan regarding this object is selected. We return to the first query and check again if the pre-conditions of the action skill hold. However, when none of the related objects have a plan that can be executed, the task at hand is considered unexecutable or failed and re-planning is required.
The collection of these questions is referred to as a query machine. A schematic overview of the query machine is given in figure 3.3. Here δ_i is the action skill currently assessed. When this skill is executed successfully, the next action skill in the plan δ_{i+1} is selected. Each of the query steps is indicated with a letter for reference and bookkeeping, used for explaining query syntax, result analysis and discussion.

[1] A graph traversal refers to the process of visiting each vertex in a graph

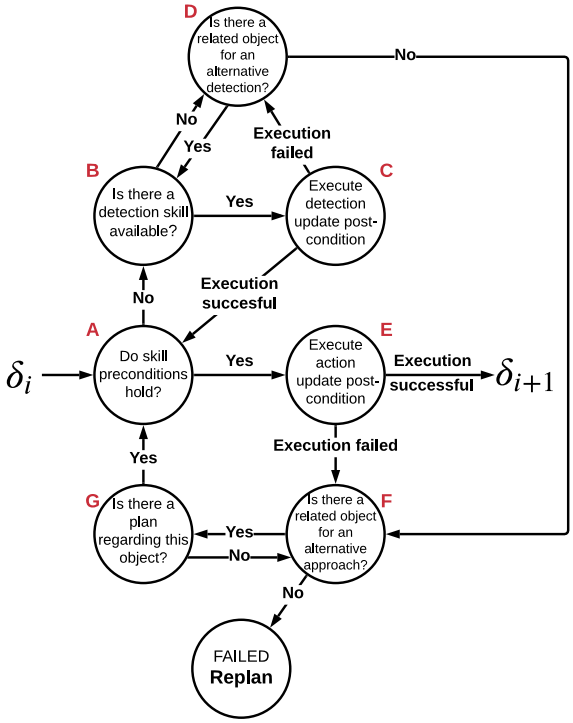


Figure 3.3: Query Machine

3.2 Materials and Methods

To implement this simulation, we use Gremlin-Python [36]. Gremlin is the graph traversal language of Apache TinkerPop, used to run traversals on a Labeled Property Graph. Gremlin traversals are navigational expressions in which patterns can be matched with the graph to navigate paths. A Gremlin traversal is composed of a sequence of steps which each perform an operation on the data. It can either transform objects by mapping, remove objects by filtering or compute statistics about objects. Gremlin is used because of its more natural expression of navigational queries compared to other languages such as SPARQL and Cypher [31], its flexibility in drivers and compilers, and its extensive documentation [37]. To implement Gremlin traversals, it offers its own applications as well as coupling to other drivers and compilers. Here, the Gremlin server compiler is used to host the world model graph. During the iterative process of formulating the queries, the graph is visualized using a Gremlin-Visualizer [38]. The Gremlin-Python Driver is used to construct Gremlin traversals. This allows the query machine to be formulated in python alongside the input variables of the task, skills and disturbances.

3.2.1 Task Implementation

The task is considered to be completed when the skill of motion to the truss is successful. This because the post-condition of this skill obtains the required relative distance between the robot and the truss to be the zero vector.

3.2.2 Skills Implementation

The considered set of skills is entered by stating each of the pre- and post-conditions in variables. When the pre-conditions are evaluated in the world model, the skills can be executed and its post-conditions result updated. The execution of these skills is implemented by evaluating whether

any of the disturbances cause the action to fail, and update its status. As such, successful or unsuccessful events are generated, breaking the continuous time aspect of real robot motions.
As stated in section 3.1.2, there are two possible approaches to consider; either the direct motion to the truss (plan A) or following the stem towards the truss (plan B). Because we have this limited set of options and we focus on querying the world model to motivate the execution of a detection skill or consider an alternative action plan, the construction of these two action plans using a planning algorithm is disregarded. As such, the action planning process of matching the pre- and post-conditions for each of the skills to construct these two action plans is done by hand.

3.2.3 Disturbances Implementation

Disturbances cause actions to fail and force the robot to take decisions about alternative approaches. To have a visual representation of the reason these actions fail, the disturbances are defined as planes in the same Cartesian coordinate frame model as the topographic relations. These planes represent objects that block a detection or motion when intersecting with the concerned vector. Figure 3.4 gives a plot of the objects considered in the simulation. The arrows represent the position vectors between them. In this example the visual occlusion of the truss is shown as the plane and the vector between the robot and the truss intersect. Other disturbances are implemented in the same manner.

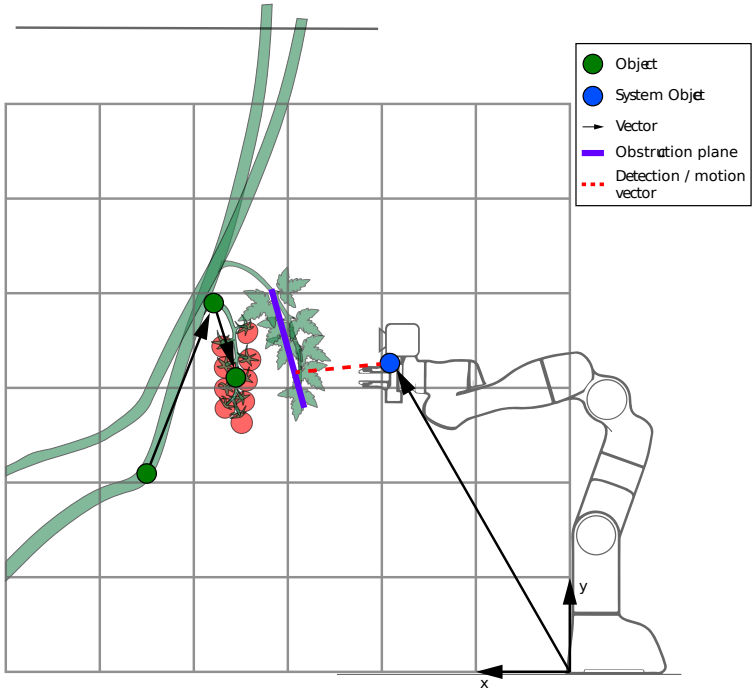


Figure 3.4: Representation of the topographic relations in a coordinate system, with a plain representing the visual occlusion of the truss.

3.2.4 Prior Knowledge Implementation

At the start of the simulation, the world model gremlin graph is initiated with the knowledge available by writing the vertices and edges in the graph. This entails the estimated topographic relations between stem, peduncle and truss and the topological relation between these objects. Also the positioning of the robot end-effector to its base at the origin is known beforehand. Objects a topological relationship are connected and have no further property specification as we do not describe how these objects are connected. Edges with a topographic relation type have

a property describing the distance between the object. The model used behind this relation is a vector in a Cartesian coordinate frame. This initiated world model $WM = (N, E, \mathcal{L}, P, \eta)$ [30], [31] is depicted in figure 3.5, where:

1. N : *a set of vertices* $= \{n_1, n_2, n_3, n_4, n_5\}$;
2. E : *a set of edges* $= \{e_1, e_2, e_12, e_34, e_45\}$;
3. $\mathcal{L}$: *a set of labels* = { Origin, System, Object, Distance, Connects };
4. P : *a set of properties* = {(name, "origin"), (name, "robot"), (name, "stem"), (name, "peduncle"), (name, "truss"), (type, "topographic_relation"), (vector, "[1, 1, 3]"), (vector, "[0, 0 ,4]"), (vector, "[1, -1, -0,5]") }
5. η : *function that associates each edge in E with a pair of vertices in N where the first vertex is its source and the second vertex its target* $= \eta(e_1) = \eta(e_{34}) = (n_3, n_4)$, $\eta(e_2) = \eta(e_{45}) = (n_4, n_5)$, $\eta(e_{12}) = (n_1, n_2)$

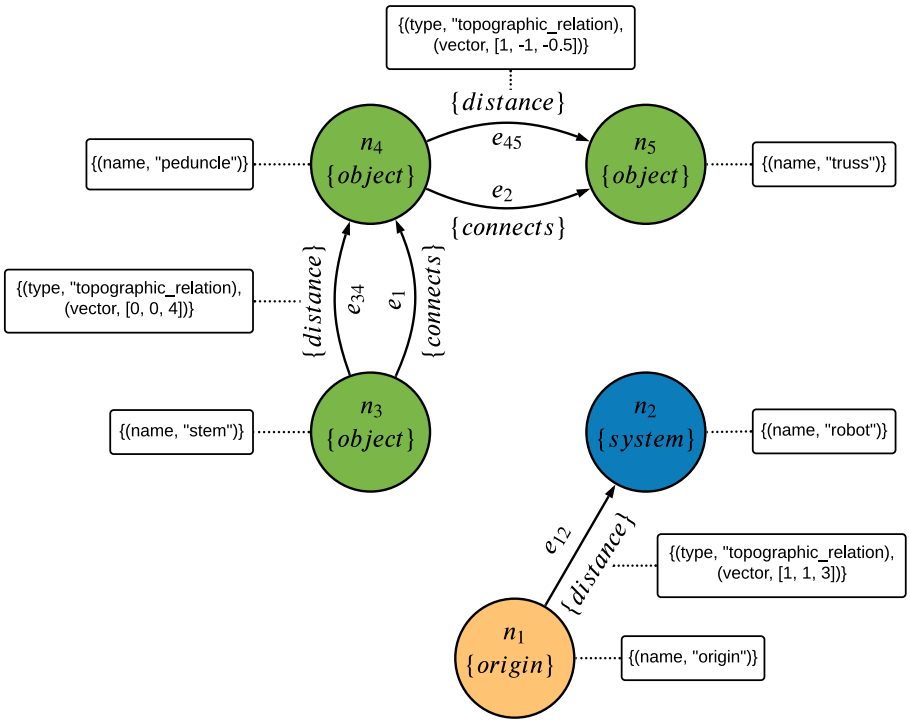


Figure 3.5: Initialized world model in Gremlin

3.2.5 Query machine implementation

The query machine from figure 3.3 is implemented in Pyhton, switching form question to question. In each step queries are run and its results are evaluated to draw a conclusion. In the following section we state the main query formulations used in the query machine with listings to give their syntax.

First the pre-conditions of detection and actions skills are to be evaluated in query step A and B. Within this simulation all skills have a topographic relation as pre-condition, namely between the robot (`current_object`) and an object (`target_object`). Running the first query of listing 1, a vertex with the target object is found in the world model graph and its in- or outgoing edge to

the robot vertex is returned and stored in the variable `e_t`. If this edge is found, the precondition holds and its vector is obtained to configure the action. Since the direction in which the edge is defined determines the direction of the vector, the edge tail is obtained by the second query of listing 1. It is verified if the edge tail vertex is the same as the `current_object` or `target_object`. If the edge is defined toward the robot, the vector is inverted.
The skill **shake** and **follow** contain another precondition, namely that the robot is at the position of the stem. To evaluate this condition we query for the edges between the `current_object` and `target_object`, as stated in the third query of listing 1. This query add the `values()` step and returns the vector value of this edge. If this value is equal to the zero vector, the robot is at the position of the target object and this precondition holds.
If a query cannot find the vertex or relation it is looking for, it returns an empty list or raises an error. Therefore, the queries are run in a python `try` statement to conclude that the preconditions do not hold.

```
1 e_t = g.V().has('name', target_object).bothE('topographic
  ↪ relation').where(otherV().is_(V_current_object)).next()
2 g.E(e_t).outV().next()
3 g.V(V_current_object).bothE('topographic
  ↪ relation').where(otherV().is_(V_target_object)).values('vector').next()
```

Listing 1: Queries to evaluate skill pre-conditions

When a detection skill is executed, the obtained results are written into the world model in query step C. First, it is evaluated if the obtained relation is already present in the world model using the first query of listing 1. Often in case of a detection, the obtained topographic relation does not yet exist and thus a new edge is written in the world model using the first query of listing 2. Here an edge is added from the vertex of the robot (`v_current_object`) to the vertex of the targeted object (`v_target_object`) with its `relation_type`, `new_id`, `name` and the newly obtained `vector`. When the relation already exists, its vector property is updated using the second query of listing 2.

```
1 g.addE('topographic_relation').from_(v_current_object).to(v_target_object).
  ↪ property(id, new_id).property('name', name).property('rel_pos',
  ↪ vector).next()
2 g.V(v_current_object).outE('topographic
  ↪ relation').where(inV().is_(v_target_object)).property('rel_pos',
  ↪ vector).next()
```

Listing 2: Queries to update skill post-conditions

When an action skill is executed in query step E, the obtained post condition is a change in the position of the robot object, which is updated in the world model using the queries of listing 2. However, when updating this position relative to the origin, its position to other objects in the environment changes too. Since the displacement of the robot is known, this displacement is updating in all edges between the robot and connected objects. The first query in listing 3 obtains all topographic relation from the robot object and stores its properties in a list. Thereafter, a `for-loop` is used to correct each vector in the list with the `robot displacement` and update its

property in the world model graph.

```
outEdges = g.V(v_current_object).outE('topographic relation').project('id',
    'outV', 'label', 'inV',
    'rel_pos').by(id_).by(outV()).by(label).by(inV()).by('rel_pos').toList()
for i in outEdges:
    new_vector = i['rel_pos'] - robot_displacement
    e_id = Edge(i['id'], i['outV'], i['label'], i['inV'])
    g.E(e_id).property('rel_pos', new_vector).next()
```

Listing 3: Queries to update robot displacement in each outgoing edge

When a detection of the target object is not possible or fails, a query is used to traverse the graph to find a related object for an alternative detection in query step D. If the action is not possible or fails, a related object is sought for an alternative approach in query step F. In both cases the query of listing 4 is used, where starting from the `current_object_of_interest` in- or outgoing edges are traversed over the topographic or topological `relation_type` respectively. When an vertex is found, its `name` and `id` is projected to a list and set as the new `current_object_of_interest`. When multiple related objects are found, they are assessed successively. When these objects are not sufficient for an alternative detection or approach, the query of finding a related object is run again. However, when doing so, the previously passed vertices are to be excluded from the traversal. This is done by only traversing to vertices which are not present in the `passed_ids` list, using the `without()` step.

```
g.V(current_object_of_interest).bothE(relation_type).otherV().
    hasId(without(passed_ids)).project('name','id').by('name').by(id).toList()
```

Listing 4: Query to find the shortest path to the target object vertex

When an alternative detection is used, the newly obtained relation is not fulfilling the preconditions of the action skill immediately. Therefore, the resulting vector over the topographic relations to the target object is derived. To this end, the query in listing 5 is formulated to find a path between the robot and the target object. It does so by repeating an in- or outgoing edge traversal over the topographic relation type using a `simplePath()` step. This step prevents the traversal path from going in circles. This is repeated until it reached the target object vertex. Using the `limit(1)` step the shortest path found is selected and its vector values are unfolded and stored in a list. Again, to account for the direction of the edges in the path the vectors are inverted if needed. When the resulting vector is calculated, it is written in the world model using the queries of listing 2.

```
g.V(v_current_object).repeat(bothE('topographic
    relation').otherV().simplePath()).until(is_(v_target_object)).path().limit(1).
    unfold().hasLabel('topographic relation').values('rel_pos').toList()
```

Listing 5: Queries to derive the resulting vector over a topographic relations path

3.2.6 Simulation Setup and Analysis

The simulation is set up by initiating the world model and entering the skills with their plans to accomplish the task of reaching the truss. To test the capabilities of the query machine to decide which skill to execute four scenarios are considered, each scenario incremented with a disturbance. An overview of the scenarios is given in table 3.2. For each scenario we document and analyze the steps taken in the query machine. In each successive scenarios we resume the analysis at the step where the conclusion drawn differs from the previous scenario.

Table 3.2: Overview of Scenarios

Scenario	Disturbances
1	-
2	Visual occlusion Truss
3	Visual occlusion Truss Spacial occlusion Truss
4	Visual occlusion Truss and Peduncle Spacial occlusion Truss and Peduncle Obstruction along Stem

3.3 Results

3.3.1 Scenario 1: No disturbances

In the first scenario, no disturbances are introduced. An overview of proceeding through the query machine is shown in table 3.3. Entering the query machine the pre-condition of moving to the truss does not hold (1). Since the detection skill **look** is available (2), this missing relative position can be obtained and written in the world model (3). As such, the precondition now holds (4) and the truss is reached (5).
In this scenario, no a priori knowledge about the world is consulted.

Table 3.3: Results Scenario 1

Step	Query	Query Result	Conclusion
1	A	Read Topographic relation robot - truss	No
2	B	Detection skill for truss	Yes: **look**
3	C	Write topographic relation robot - truss: 'rel_pos' = [3.0, 3.0, 3.5]	Successful
4	A	Read Topographic relation robot - truss: 'rel_pos' = [3.0, 3.0, 3.5]	Yes
5	E	Write topographic relation robot - truss: 'rel_pos' = [0, 0 ,0]	Successful

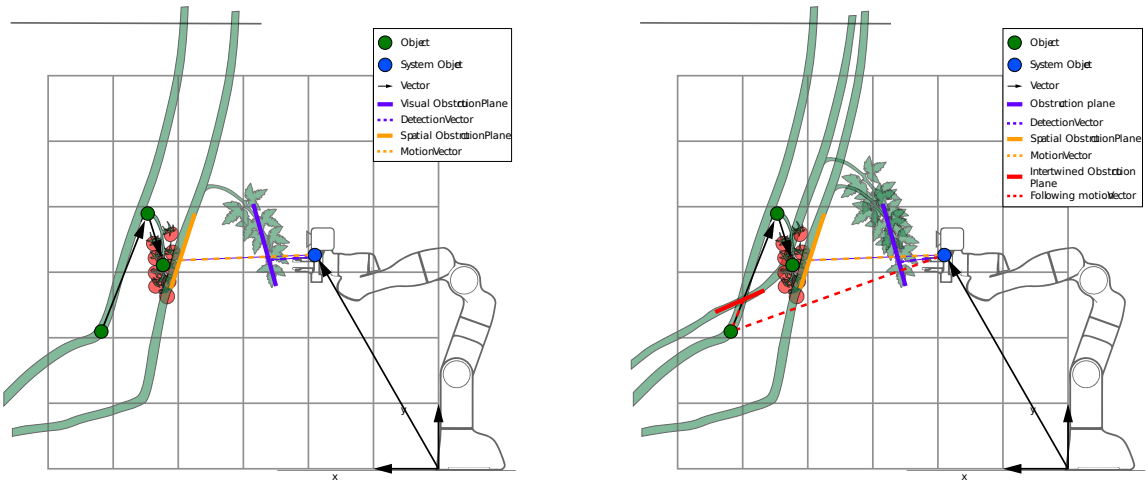
3.3.2 Scenario 2: Visual occlusion

In the second scenario, the truss and peduncle are visually occluded by hanging leaves as previously shown in figure 3.4. Hence, the detection of the truss with the camera from the previous scenario fails at step 3. This requires the robot to find an alternative approach. An overview of the query results is given in table 3.4. It traverses the graph over the topographic relation starting from the truss vertex finding the peduncle as a related object (4). However, since the precondition of the detection **shake** does not hold no detection is available (5). Traversing the graph again, taking the previously passed truss vertex into account, the vertex of the stem is found (6). For this object

the detection skill of **look** is available (7), which is executed and the resulting vector to the truss is derived (8). Now, the precondition of the **move** skill hold again (9) and is executed (10).
In this scenario the a priori knowledge about the estimated distance between the truss, peduncle and stem is used to estimate the positioning of the truss.

Table 3.4: Results Scenario 2

Step	Query	Query Result	Conclusion
3	C	Write topographic relation robot - truss	Failed
4	D	Read Topographic relation truss - peduncle	Yes
5	B	Detection skill for peduncle	No
6	D	Read Topographic relation peduncle - stem	Yes
7	B	Detection skill for stem	Yes: **look**
8	C	Write topographic relation robot - stem and derive robot - truss	Successful
9	A	Read Topographic relation robot - truss: 'rel_pos' = [3.0, 3.0, 3.5]	Yes
10	E	Write topographic relation robot - truss: 'rel_pos' = [0, 0 ,0]	Successful



(a) Scenario 3: addition of spacial obstruction

(b) Scenario 4: addition of obstruction along stem

Figure 3.6: Graphical representation of disturbances in scenarios

3.3.3 Scenario 3: Visual and Spacial occlusion

In the third scenario, the truss and peduncle are visually and spatially blocked by the stem of another plant as depicted in figure 3.6a. As such, the robot can detect the truss to obtain the required pre-condition to initiate the motion via scenario 2, but the motion to the truss fails in stem 10. table 3.5 gives an overview of the step taken from this point. To motive the alternative approach of following the stem it traverses over the topological relations. Starting from the truss it finds the peduncle (11). Since there is no plan available starting from this object (12), the graph is traversed again finding the stem vertex (13). From is object the plan to follow the stem (plan B) is available (14). Here the first action is to move to the stem, of which the pre-condition does already hold since the stem has been detected before (15). As such, detection is only executed when necessary to meet the pre-conditions of an action skill. Thereafter, the motion to the stem is executed and the robot position updated in the world model (16). Now the next action in the plan is selected and the query machine is re-entered. The preconditions of the follow skill entail that the relative distance between the robot and the stem is to be equal to the zero vector and

the relative position to the peduncle is to be known. The latter is not found in the world model, thus the pre-condition does not hold (17). To obtain this relation the detection skill **shake** is available. Its pre-condition of the relative distance between the robot and the stem to be the zero vector holds (18). As such, this detection is executed and its results written in the world model (19). With the results of this detection the pre-condition of the **follow** action hold (20) and it is executed(21). With the robot arm positioned at the peduncle the last step in the action plan is to move to the truss. Re-entering the query machine the pre-conditions of this action hold (22) and it is successfully executed (23).

Table 3.5: Results Scenario 3

Step	Query	Query Result	Conclusion
10	E	Write topographic relation robot - truss	Failed
11	F	Read topological relation truss - peduncle	Yes
12	G	Plan starting from the peduncle	No
13	F	Read topological relation peduncle - stem	Yes
14	G	Plan starting from the stem	Yes: **Plan B**
15	A	Read Topographic relation robot - stem: 'rel_pos' = [x,y,z]	Yes
16	E	Write topographic relation robot - stem: 'rel_pos' = [0, 0 ,0]	Successful
δ_{i+1} next action in plan: **follow stem**			
17	A	Read Topographic relation robot - stem, robot - peduncle	No
18	B	Detection for peduncle	Yes: **shake**
19	C	Write topographic relation robot - peduncle	Successful
20	A	Read Topographic relation robot - peduncle: 'rel_pos' = [0, 0, 4]	Yes
21	E	Write topographic relation robot - peduncle: 'rel_pos' = [0, 0 ,0]	Successful
δ_{i+1} next action in plan: **move to truss**			
22	A	Read Topographic relation robot - truss: 'rel_pos' = [1, -1, -0.5]	Yes
23	E	Write topographic relation robot - truss: 'rel_pos' = [0, 0 ,0]	Successful

3.3.4 Scenario 4: Visual & Spacial occlusion, Obstruction along the stem

The final scenario adds obstruction along the stem by the entanglement of two tomato plants, as depicted in figure 3.6b. In this case the robot proceeds as in scenario 3, but the execution of following the stem fails in step 21. After this failure the graph is traversed over the topological relation until the peduncle is found (22). As there is no plan starting from the peduncle (23), the graph is queried again to find the truss (24). For this object the initial plan (plan A) is available (25), for which the pre-condition holds (26). Since the robot has moved to the stem, the path for the motion to the truss is free again and its execution is successful (27).

Table 3.6: Results Scenario 4

Step	Query	Query Result	Conclusion
21	E	Write topographic relation robot - peduncle	Failed
22	F	Read Topographic relation stem - peduncle	Yes
23	G	Plan starting from the peduncle	No
24	F	Read topological relation peduncle - truss	Yes
25	G	Plan starting from the truss	Yes: **Plan A**
26	A	Read Topographic relation robot - truss: 'rel_pos' = [x,y,z]	Yes
27	E	Write topographic relation robot - truss: 'rel_pos' = [0, 0 ,0]	Successful

4 Discussion

In this chapter we discuss the results of the described scenarios and our observations regarding the query formalization. With the insight we gained from this simulation we discuss challenges we face for a full scale implementation of the proposed method.

4.1 Simulation Results

In the simulated scenarios we show that the query machine indeed decides which action or detection is executed, based on the information available in the world model. As detection of the truss is not possible in scenario 2 (section 3.3.2), the distance relations between the truss and the stem in the world model are consulted to motivate the alternative detection of the stem. Consulting these relations causes the decision to be based on the presence of this information in the world model. When this prior knowledge would not be provided nor obtained over time, detection of the stem would not be considered. Likewise, in scenario 3 (section 3.3.3) the alternative plan B of moving to the stem and following it up to the peduncle is considered because of the topological relation between truss and stem.

In comparison, the classical approach of constructing a Finite State Machine (FSM) could be used to solve these scenarios too. A FSM is often comprehensive and constructed with all disturbances in mind. It is build around the expected scenario and task at hand, and will proceed until the final state is reached. An example of a FSM for scenario 2 is given in figure 4.1a. While constructing this FSM the system designer uses the knowledge he has about the topographic relations between the truss, peduncle, and stem. With this knowledge he prescribes that when the detection of the truss fails, it can detect the stem to thereafter derive the truss position. As such, both detections can be used to obtain the required information to move to the truss. For scenario 3 an example FSM is given in figure 4.1b. Here the knowledge about the topological relation between the truss, peduncle and stem is used to capture the moment when to try the alternative approach of plan B, where the stem is followed up to the truss. As such, when the motion to the truss fails it prescribes the action to move to the stem, shake the stem and thereafter follow it. In both of these FSMs the consult of knowledge is fixed and incorporated into the transitions between states, assuming this will be sufficient for the real scenario the robot encounters. Whereas the query machine consults this knowledge on demand, only when pre-conditions of an action do not hold or actions fail. The query machine is, as it were, constructing its own FSM on the spot using the same, albeit limited, knowledge a system designer would have.

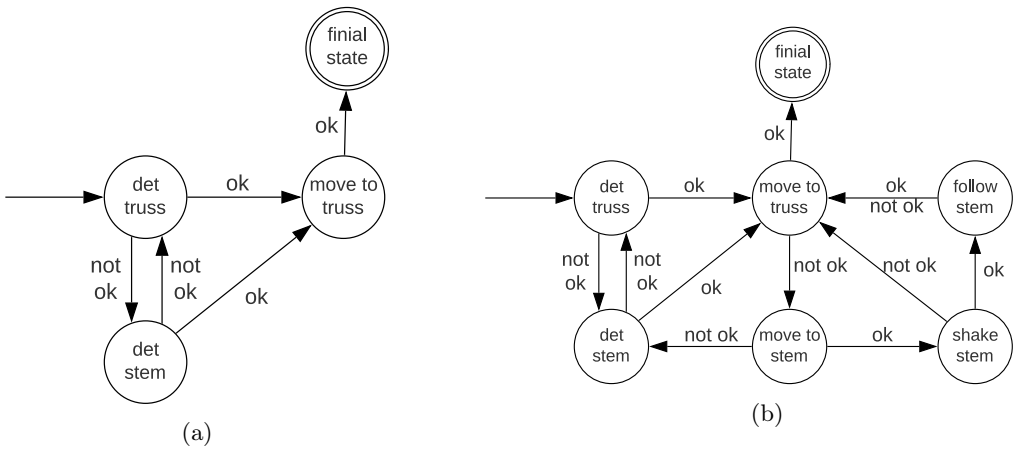


Figure 4.1: Example of a Finite State Machine for scenario 2 (a) and scenario 3 (b). Here, each event is considered an uncontrollable event.

Using this knowledge, as opposed to using the predefined transitions of the classical FSM, provides the following advantages. The information in the world model can be changed, supplemented and improved at any given time by updating the world model. This is done by, for example, by the robot's gathering of information, information from other robots or by the designer's new learnings about the environment. When the world model is improved, the robot can make its decisions on knowledge previously not available at design time, using the same set of queries. The FSM, on the other hand, is limited in how well the designer could foresee the environment at the time of designing. Moreover, when new skills are added it can use the same query to identify the pre-conditions. Within this demo for instance, the estimated distance between trusses and leaves hanging on the stem can be added. When adding a leaf detection skill as well, the same query machine concludes to estimate truss positions through detection of the leaf instead of the stem. This is because the queries are not dependent on the object itself but on the relation type between them. In the classical FSM approach, this would require the addition of the state to detect a leaf and define in and out going transitions to the other states, similar to the addition of states in the FSM examples of figure 4.1a and figure 4.1b. By doing this, the designer is trying to foreseeing in which situation this approach may be considered. For instance, each time detection of the truss or stem fails it should consider detecting a leaf. For the use of the topological relation, which is the requirement to follow an object until the end is reached, we could introduce different objects with this relation type. For instance, when a wire connects from the ground to the peduncle, the query machine can argue to follow this wire since it meets the requirement of being topological connected. One could imagine that following this line would eventually lead to finding the truss as well, because of their connected relation. As such, the query machine looks for the relation between objects which motivate an approach, not because approaches were entered beforehand. The formulated query machine suffices also when the world model is extended and more skills are introduced, assuming the additional information and skill conditions are of the same relation types as considered within the simulation. In this way, composition of actions is done online which illustrates the potential of this method.

In scenario 4 we observe undesired behaviour. When the action of following the stem in plan B fails, query step F and G are entered. Here the topological relation of the stem with the truss is traversed, which is used as a motivation to initiate plan A and move to the truss directly. However, knowing that the stem and truss are connected is not necessarily a condition that motivates plan A. Directly moving toward the truss would also be a valid approach not knowing the stem and truss are connected. As these queries were constructed to motivate the choice for plan B, the same condition is used to go back to the previous approach of plan A. As such, in the current implementation the motivation is fixed and not dependent on the plan that is considered. This can be corrected by making this query step dependent on the plan. In this case traversing for the existence of a truss object in the world model could be enough motivation to initiate plan A. However, this points out the difference between motivation for an alternative plan instead of a single action. For a single action, we query for its preconditions to check if execution is possible and for its post-conditions to argue its result is desired. In the simulation this can be seen when querying for a detection skill in query step B and D. However, for a concatenation of actions, querying for the preconditions of the first action is not always sufficient to motivate execution of the whole plan. In the simulation, the connection between truss, peduncle and stem is the condition to motivate the approach of following the stem, not of each individual action in this approach. Currently, the action planning is done by hand, as is defining the motivation for the plan. When automating the planning process, however, it remains to be determined which conditions are motivators for an alternative plan. Besides, we are in no way certain that following the stem will succeed, requiring the need to incorporate uncertainties in plans. Nonetheless, the topological knowledge between a truss and stem is a strong enough motivator for this course of action. This is because we know that we have the ability to follow the stem to the peduncle, and we expect that we can reach the truss from that point because of this connection.

4.2 Query Formalization

As explained in section 3.2.5, some problems are observed when querying the world model. Here we evaluate the used query formulations as a solution to these problems.

When querying for a related object, we encountered the problem of traversing previously passed vertices. In query step D the world model is traversed one edge outward from the object of interest over the topographic relation, to obtain an adjacent object. When it is not possible to detect this object, the query is run again. To prevent it form returning to the object it came from, the query excludes previously passed vertex ids from a list using the `without()`-step. Although this indeed directs the search to new adjacent objects, the formulation of these query steps could be optimized. Using a single query, all possible outgoing paths over the topographic relation type can be obtained. An example of how this query can be formulated is given in listing 6. Thereafter, all the objects in these paths could be assessed for a possible detection, starting with the shortest path. As such, it is not needed to traverse the graph each time to find adjacent objects in an iterative way.

```
1 g.V(v_object).repeat(bothE('topographic
  ↪  relation').otherV().simplePath()).path().toList()
```

Listing 6: Example queries to obtain all outgoing paths from a vertex

In query step C topographic relations are traversed obtaining their vector property. Here the direction of the edges is accounted for by evaluating the in- and outgoing vertices of the edges with respect to the order in which the vertices are traversed. This allows to account for the edge direction by inverting the vector when traversed in opposite direction. Although this is a good way to obtain the edge direction, the implication of traversing over opposite edge direction is not straightforward for all relation types. For a topographic relation the inverse of a vector is a valid solution but, for instance, when using physical relations between objects which refer to a dynamic model this is not necessarily the case. For such relations taking the inverse is not trivial and therefore the implication of traversing in opposite edge directions is to be examined.

4.3 Towards a Full Application

As the results show the potential of this approach, they are limited in providing an answer to the challenges of a real world implementation. Therefore, we discuss the lessons learned and describe our notion on how to improve this implementation in this section. Going towards a full implementation of the method and testing in real world scenarios, the world model is to be expanded and skills are to be designed with more meaningful or expressive pre- and post-conditions. The query formulation and query machine as currently implemented are not generic enough and will not suffice in a real world situation. We foresee the following main problems.

4.3.1 Expanding and Maintaining the World Model

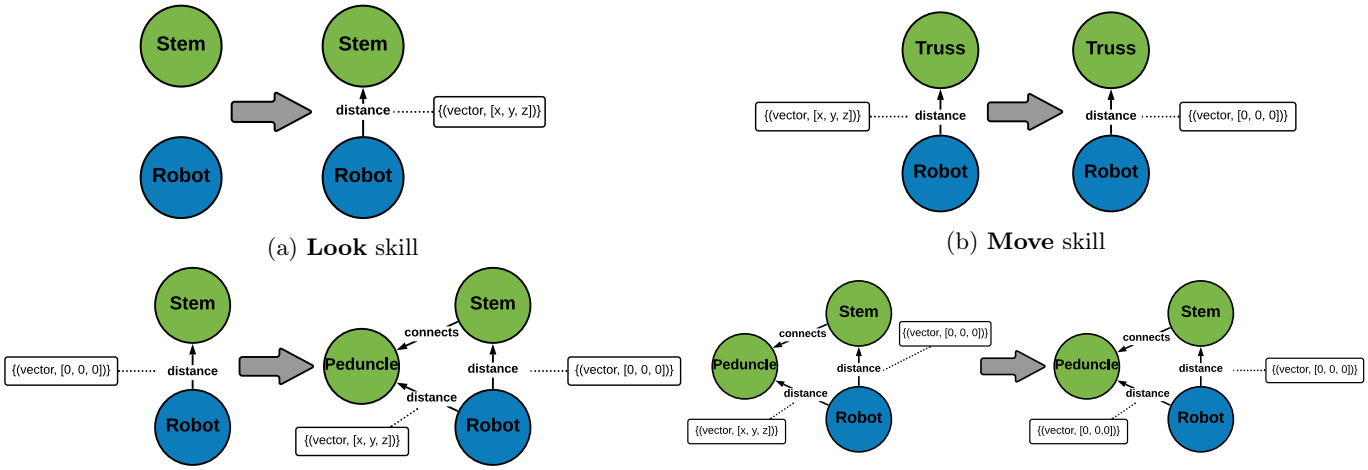
To obtain a more expressive representation of the world, the world model as currently implemented is to be expanded by including more objects, relations and their properties. Newly obtained sensor information is to be linked to the symbolic representation of the environment. However, this introduces a multi hypothesis problem. When a truss of tomatoes is detected it can be added to the world model as a vertex with properties and relations to other objects. However, this truss could be detected multiple times or using different skills. In this case, it is to be assessed if the obtained information regards the same truss as before or that a new truss is found.

As the symbolic world model is a meta-model of the obtained sensor information used for reasoning, it is a design-choice when to update new information. As the world model is used to reason about a possible approach, the update frequency of the world model is dependent on the task at hand and the complexity of the environment. This applies to information about the robot itself too. Within this simulation, the change in position of the robot is updated directly into the world model see listing 3. However, this is a design choice and may only be done when these relations are to be queried. Moreover, handling complementary or contradictory information in multiple hypothesis is not straightforward. Determining update frequency and handling multiple hypothesis warrants further investigation.

Maintaining the world model over time involves more that updating it with newly obtained information from sensors. We construct the world model as a graph with the goal to query its structure of semantic relations between objects. To this end, we want to obtain the relationship between new information and the rest of the world model by logical inferences. For instance, when we come to know that a truss is connected to a stem, we infer that both of these objects are part of the same plant. Within properties of an object these inferences are even more obvious. For example, when volume and density of a tomato are known, we can calculate its mass which is subject to the physical relation is has to other object. A useful tool to derive these quantities are QUDT ontologies [39]. These ontologies specify the relation between quantities which could be used to derive them on demand.
Taking these inferences about relations or properties can also be done by predicting its status over time. Since a plant is growing, we can infer that the positioning of trusses, amount or tomatoes in a truss and the weight of tomatoes will change over time. Investigating the rules to such inferences remains future work.

4.3.2 Expanding Skill Definition

When expanding the expressions of pre- and post-conditions of each skill, we define more conditions that must be evaluated regarding the current state of the world model. In the current implementation this small set of conditions is evaluated step-by-step in different queries where each query results provides a conclusion. However, when this set is growing this is not an efficient approach. Therefore, we propose the skills to be defines as LPGs as well and this graph to be matched to the world model. For the pre- and post-condition of the skills considered in this work this results in the graphs as depicted in figure 4.2. Describing more conditions involves an expansion of these graphs in the amount of vertices, edges and properties that need to be validated. This introduces the problem of graph matching and evaluating property constraints.



(a) **Look** skill

(b) **Move** skill

(c) **Shake** skill

(d) **Follow** skill

Figure 4.2: LPG representation pre-conditions (left) and post-conditions (right) of each skill. To improve readability property specification of the vertices has been omitted

Matching Problem

In scenario 2 (see section 3.3.2), a visual occlusion of the truss causes the detection of the truss to fail. Therefore, the pre-condition for a direct motion to the truss, represented by the topographic relation between the robot and the truss, is not met. To find and alternative detection we query over the topographic relations in the world model graph. For each object found, we evaluate if it occurs in the post-conditions of a detection skill. As such, we match the post-condition of a detection to fulfil the precondition of an action. To illustrate how this queries work, a simplified graphical representation of these graphs is given in figure 4.3. Currently, this problem is implemented step-by-step in query step A, B, C and D (see figure 3.3). The required relations is edge e_{25} from n_2 to n_5, in graph (a). Since the edge e_{25} is not currently available in the world model, represented as graph (b), we check for a possible detection to obtain this relation. This edge e_{25} is neither an obtainable relation from a detection skill, since it does not exist in the graph (c). Therefore, we start querying to find a related object for an alternative detection. As such, we start form vertex n_5 and traverse over the edges in graph (b). For each vertex passed, we check if it exists in the obtainable relation of graph (c). So, first we pass vertex n_4, which does not exist in graph (c). Then we reach vertex n_3, which does exist in graph (c). Now we conclude that the detection skill which obtains the edge e_{23} will be suitable to find the required relation of e_{25}. This because if we write edge e_{23} in graph (b) we can derive the path from n_2 to n_5 over the edges e_{23}, e_{34} and e_{45}. We can use this step by step approach because of the assumption that if the vertex n_3 exists in the obtainable relations graph (c), we will find this alternative path over the vertices.

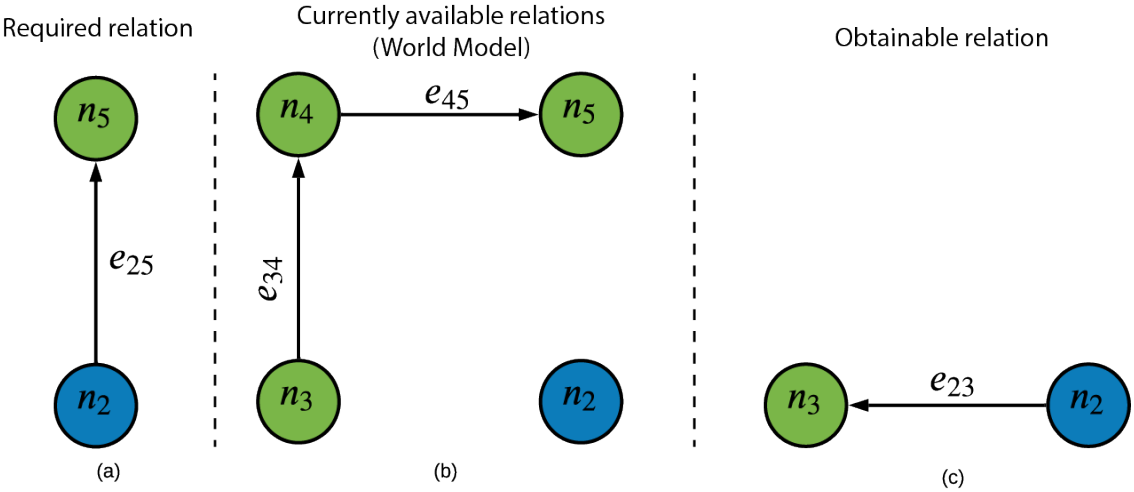


Figure 4.3: Abstracted representation of the pre-condition of an action skill which denotes the required relations (a), the world model which denotes the current available relations (b), and post-condition of a detection skill which denotes the relations which can be obtained (c).

However, when extending the world model and designing more expressive pre- and post-condition, the amount of vertices and edges in the graphs (a), (b) and (c) will increase. In this case, the assumption made to find an alternative detection does not hold. Any vertex passed in the traversal over the graph (b) may be found in graph (c), but will not guarantee that this newly obtained relation will obtain the required relation of graph (a) over an alternative path. For instance, if the vertex n_3 occurs in graph (c) but does not contain e_{23} this path cannot be found. This warrants a more generic way of solving this problem. A query must be formulated which projects graph (c) on (b) such that the condition of (a) holds. In this example, projecting (c) on (b) is constructing a temporary predicted state of the world model (b'), in which condition (a) can be validated by finding a path over the vertices n_2, n_3, n_4 and n_5. In this way we find a match between graphs (a), (b) and (c) without going through all the successive steps as described in the step-by-step approach. Finding this match between the graphs (a), (b) and (c) includes matching multiple

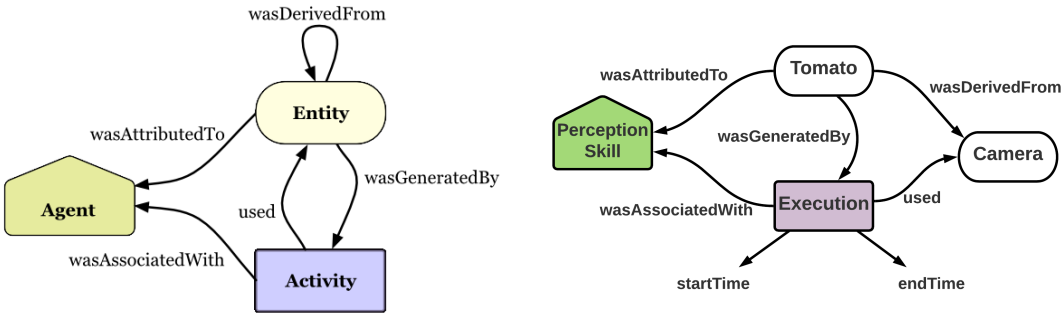
vertices and edges. Validating if the exact vertices and edges of a graph are present in another graph refers to the sub-graph isomorphism, or exact graph matching problem [40]. In Gremlin the `match()`-step can be used to match graph patterns to a graph as done in [41]. However, we also allow alternative path between vertices to be a match. As described in this example, the result of projecting graph (c) on (b) suffices for a match with graph (a) even if e_{25} itself is not present. This because there exists a path between vertex n_2 and n_5. In this case a graph isomorphism is not possible, introducing a problem referred to as the inexact graph matching problem [40], [42]. Here matching aims to find the best non-bijective[1] correspondence between graphs used in, for example, image recognition [43]. It is to be researched how this graph matching can be obtained in Gremlin. Combining the `match()` step with the path finding queries used in our simulation could be a way to validate a inexact graph match. However, this raises the question how exact a graph match has to be to conclude whether a pre-condition holds. When every relation in the pre-condition is found in the world model graph through one or more alternative paths, can we still draw a single conclusion? Is the intended description of constraints of the situation, in which the skill is to be executed, still valid?
We believe that this inexact matching of designed pre- and post-conditions with the current state of the world model, is an interesting field to investigate. When using inexact matching, the design of pre- and post-conditions can be less restrictive. Because the conditions are projected on the world model, reasoning about which actions could be taken is supplemented with awareness about the situation the robot is in. As such, designing these conditions is less focused on foreseeing all possible situations and prescribing corresponding actions.

Include Object and Relation Properties

Beside the increasing amount of relations that must hold, a more expressive description of pre- and post-conditions specifies constraints on the properties of objects and relations. For instance, the pre-condition of moving to a position in space requires the distance vector of the topographic relation to be within the reachable space of the robot. The property of this relation is thus to be assessed, as well as its accuracy and reliability. When we derive the distance to the truss based on its relative position to the stem, the accuracy of the estimated distances between the stem and the truss used is important. The accuracy determines to what extent we can rely on this derived distance and if the pre-condition of the motion skill holds. For a rough initial motion to a point in space this may require the target position to be less accurate opposed to a precise motion over a short distance.
Within the simulation, relations can be labelled as nominal, derived or measured using properties of properties, referred to as meta-properties. These labels indicate the origin of information from either prior information, a combination of detected and prior information, or detected information. However, stating the origin of information is not indicating how accurate it is and its trustworthiness changes over time. Hence, to assess the reliability of the information in the world model, the processes that obtained the information are to be tracked. To this end, a provenance model can be a useful tool. A provenance model contains information about entities, activities, and agents involved in producing a piece of information, which is used to assess its quality, reliability or trustworthiness. The W3C standardized format of a provenance model is called PROV [44], with the goal to enable provenance on the web and other information systems and assess its publications. It represents provenance by means of a Resource Description Framework (RDF) $P = (V, E)$. Where V is a set of vertices representing entities, activities or agents. $E \subset \{(v, w)|(v, w) \in V^2\}$ a set of directed edges which are order pairs of vertices representing a fixed set of relation types: wasDerivedFrom, wasGeneratedBy, used, wasAttributedTo, wasAssociatedWith, as shown in figure 4.4a. The provenance of the obtained information when detecting a tomato can be represented as shown in figure 4.4b. Here the tomato properties are generated by executing the perception skill. This skill used the camera as a source, from which the information is thus derived.

[1] A Bijection is a function between the elements of two sets, where each element of one set is paired with exactly one element of the other set, and vise versa.

Provenance of information in the world model can also be used to assess which of multiple query results to base a conclusion on. The utilization of a provenance model along the world model remains yet to be investigated.



(a) PROV-model [44]

(b) Example Provenance Tomato Perception

Figure 4.4: Provenance Models

4.3.3 Expanding the Task Set

Improving to a full application requires expanding the task set to more than a single task as regarded in the simulation. Moreover, the task definition itself has to be more expressive than a single conditions that must be successfully executed. Alike with skills the task description can be represented as a graph. For example, we consider the task of harvesting a truss of tomatoes, this involves separation of the truss from the plant after which they are to be stored in box. As such, in the initial state the truss is connected to the stem via a peduncle. In the desired state this connection is lost and a connection is obtained between the truss and the box. An example of this task in graphs is shown in figure 4.5.
In some cases a user wants to further specify the task. Alike with skills, property requirements and behavioral constraints are introduced. For example, the harvesting area in which the robot may harvest trusses, which is mapped on the topographic relations between the stem and the truss. Another example is the maximum allowed pressure that may be applied to a tomato to prevent the robot from squashing it. This is mapped on the physical relation between the tomato and whichever part of the robot it makes contact with. Behavioural constraints are to further specify how the task may be accomplished. We can prohibit certain relations in the world model to be changed, defined as a per-condition. For instance in the harvesting task, lifting the box to the truss in order to first make the connection between the box and the truss and thereafter separate it from the stem is not desired. In this case the connection between the box in the ground is defined as a per-condition and is thus to be maintained between the pre- and post-conditions.

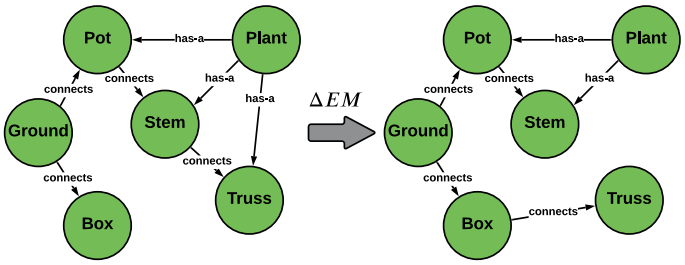


Figure 4.5: Task of harvesting a truss. Here pre-conditions define the initial state (left) and the post condition define the desired state (right)

Task description in graph representation introduces the same challenges on matching and validation of element properties in the world model. Moreover, introducing multiple tasks requires automated action planning, opposed to the predefined action plans used in the current simulation. It is to be investigated how motivation through reasoning on a symbolic level as researched in this work can be used for to direct steps in the planning process.

5 | Conclusions and Future Work

In this work we investigated an alternative design approach for a composable deliberate system faced with the challenge of operating autonomously in an open-world greenhouse environment. This approach uses a graph based world model to capture the semantic relations between objects in the environment. This world representation is used to reason about which course of action to consider, to accomplish a task in the current situation. To explore the implications of this approach, a simulation is made in Gremlin-Python where the world model is implemented as a Labeled Property Graph and a set of queries is formulated to traverse it. The results show that the motivator to execute skills, can be based on the current information available in the world model graph, rather then on information available at design time. As such, upon expansion of this information alternative actions are considered. Additionally, it allow the robot to choose an alternative course of action based on available knowledge instead of following a scheme of steps.

The explorative simulation, albeit seemingly simple, gives us great insight in our newly proposed method. Our first impression is that it has promising potential to become a scabable framework, though there are many challenges ahead before it can become applicable to a real world scenario. To obtain a more expressive representation of a real world scenario, the symbolic world models is to be expanded and maintained over time regarding newly obtained information. It is to be investigated how to automate this process and which inferences or rules apply. For example, when a truss and its tomatoes is recognized corresponding vertices with the detected properties can be introduced to the world model. In this research the handling of supplementary and contradictory information about objects as well as the frequency in which the world model is updated is essential, while focusing on accommodating the reasoning process.
We pointed out that to be applicable to real world a more expressive description of skill pre- and post-conditions is required. To this end, we propose to define these conditions in a labeled property graph as well. This introduces the challenge of inexact sub-graph matching where not single relations but sub-graph conditions are matched to the world model. We believe that this inexact matching of designed pre- and post-conditions with the current state of the world model, introduces reasoning about which course of action to take, supplemented with awareness about the situation the robot is in. Investigation of graph matching techniques and to which degree this match must be found to come to the correct conclusions, remains future work. Moreover, properties of relations and objects in the world model graph are to be assessed regarding required constraints. To this end, we propose the utilization of a provenance model to maintain the origin of information in the world model and allow for the assessment of their accuracy and trustworthiness. Upon expansion of the task set, the need for automated action planning becomes evident. In future research it is to be investigated whether steps taken in this planning process can be directed by reasoning about the world model graph as done in this work.

Regarding our objectives, we examined the use of a graph-based world model and the reasoning questions required. We implemented this method in a simplified simulation to obtain insights about this formulation as stated above. To formulating a task independent of the system that accomplishes it, we proposed defining the task as the changes to the environment objects in the world model graph. However, validation of this method and its application is yet to be investigated.

Bibliography

[1] P. Y. Chua, T. Ilschner, and D. G. Caldwell, "Robotic manipulation of food products - A review," *Industrial Robot*, vol. 30, no. 4, pp. 345–354, 2003, ISSN: 0143991X. DOI: 10.1108/01439910310479612.

[2] G. Ahmad Nayik, "Robotics and Food Technology: A Mini Review," *Journal of Nutrition & Food Sciences*, vol. 05, no. 04, 2015. DOI: 10.4172/2155-9600.1000384.

[3] J. Choi, X. Yin, L. Yang, and N. Noguchi, "Development of a laser scanner-based navigation system for a combine harvester," *Engineering in Agriculture, Environment and Food*, vol. 7, no. 1, pp. 7–13, Feb. 2014, ISSN: 18818366. DOI: 10.1016/j.eaef.2013.12.002.

[4] C. Zhang and N. Noguchi, "Development of a multi-robot tractor system for agriculture field work," *Computers and Electronics in Agriculture*, vol. 142, pp. 79–90, Nov. 2017, ISSN: 01681699. DOI: 10.1016/j.compag.2017.08.017.

[5] *Home | AgXeed - We provide autonomy.* [Online]. Available: https://agxeed.com/.

[6] *Vereijken Kwekerijen.* [Online]. Available: https://vereijkenkwekerijen.nl/.

[7] E. J. van Henten, C. W. Bac, J. Hemming, and Y. Edan, "Harvesting Robots for High-value Crops: State-of-the-art Review and Challenges Ahead," *Journal of Field Robotics*, vol. 31, pp. 888–911, 2014. DOI: 10.1002/rob. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/rob.21514/abstract.

[8] B. Arad, J. Balendonck, R. Barth, O. Ben-Shahar, Y. Edan, T. Hellström, J. Hemming, P. Kurtser, O. Ringdahl, T. Tielen, and B. van Tuijl, "Development of a sweet pepper harvesting robot," *Journal of Field Robotics*, vol. 37, no. 6, pp. 1027–1039, 2020, ISSN: 15564967. DOI: 10.1002/rob.21937.

[9] Q. Feng, W. Zou, P. Fan, C. Zhang, and X. Wang, "Design and test of robotic harvesting system for cherry tomato," *International Journal of Agricultural and Biological Engineering*, vol. 11, no. 1, pp. 96–100, 2018, ISSN: 19346352. DOI: 10.25165/j.ijabe.20181101.2853.

[10] R. Barth, *Vision principles for harvest robotics : sowing artificial intelligence in agriculture.* 2018, p. 325, ISBN: 9789463433181. [Online]. Available: https://library.wur.nl/WebQuery/wda/2246673%0Ahttps://research.wur.nl/en/publications/6571bb58-1f30-4b14-9cd9-0c716cac8852.

[11] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017, ISSN: 00043702. DOI: 10.1016/j.artint.2014.11.003. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2014.11.003.

[12] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Ben-Shahar, "Computer vision for fruit harvesting robots - State of the art and challenges ahead," *International Journal of Computational Vision and Robotics*, vol. 3, no. 1-2, pp. 4–34, 2012, ISSN: 1752914X. DOI: 10.1504/IJCVR.2012.046419.

[13] J. Senden, L. Janssen, B. Kool, R. van der Kruk, H. Bruyninckx, and R. Van De Molengraft, "Expoiting Plant Dynamics in Robotic Fruit Localization," 2021.

[14] R. Soetens, *Towards Automatic Harvesting of Truss Tomato in a High-Wire Cultivation System (MSc Thesis).* 2015.

[15] M. R. N. A. Endsley, "DESIGN AND EVALUATION FOR SITUATION AWARENESS ENHANCEMENT," *October*, pp. 97–101, 1988.

[16] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic information for robot navigation: A survey," *Applied Sciences (Switzerland)*, vol. 10, no. 2, 2020, ISSN: 20763417. DOI: 10.3390/app10020497.

[17] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013, ISSN: 02783649. DOI: 10.1177/0278364913481635.

[18] *Robo Brain*, 2015. [Online]. Available: http://robobrain.me/.

[19] L. Naik, S. Blumenthal, N. Huebel, H. Bruyninckx, and E. Prassler, "Semantic mapping extension for OpenStreetMap applied to indoor robot navigation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3839–3845, 2019, ISSN: 10504729. DOI: 10.1109/ICRA.2019.8793641.

[20] R. Balogh and D. Obdržálek, "Using Finite State Machines in Introductory Robotics," *Advances in Intelligent Systems and Computing*, vol. 829, pp. 85–91, 2019, ISSN: 21945357. DOI: 10.1007/978-3-319-97085-1{_}9.

[21] M. Figat and C. Zielinski, "Robotic System Specification Methodology Based on Hierarchical Petri Nets," *IEEE Access*, vol. 8, pp. 71 617–71 627, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2987099.

[22] M. Colledanchise and P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," 2017. DOI: 10.1201/9780429489105. [Online]. Available: http://arxiv.org/abs/1709.00084%0Ahttp://dx.doi.org/10.1201/9780429489105.

[23] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971, ISSN: 00043702. DOI: 10.1016/0004-3702(71)90010-5.

[24] J. Lobo, G. Mendez, and S. R. Taylor, "Knowledge And The Action Description Language A," no. March 2001, 2004. [Online]. Available: http://arxiv.org/abs/cs/0404051.

[25] D. McDermott, "The Formal Semantics of Processes in PDDL," *Proceedings of the Workshop on PDDL at the 13th International Conference on Automated Planning & Scheduling (ICAPS)*, no. Robinson, p. 87, 2003. [Online]. Available: http://www.academia.edu/download/38503906/PDDL-ICAPS03.pdf#page=95.

[26] I. Georgievski and M. Aiello, "An Overview of Hierarchical Task Network Planning," pp. 1–45, 2014. [Online]. Available: http://arxiv.org/abs/1403.7426.

[27] M. Ghallab, D. Nau, and P. Traverso, "The actor's view of automated planning and acting: A position paper," *Artificial Intelligence*, vol. 208, no. 1, pp. 1–17, 2014, ISSN: 00043702. DOI: 10.1016/j.artint.2013.11.002. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2013.11.002.

[28] P. van Doren, *Planning problem formulation for a domestic environment (MSc Thesis)*. 2020.

[29] G. Muller, "CAFCR: A multi-view method for embedded systems architecting," *PhD thesis*, pp. 1–266, 2004, ISSN: 02779536. [Online]. Available: https://www.gaudisite.nl/ThesisBook.pdf.

[30] R. Angles, "The property graph database model," *CEUR Workshop Proceedings*, vol. 2100, no. Section 2, 2018, ISSN: 16130073.

[31] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. Vrgoč, "Foundations of modern query languages for graph databases," *ACM Computing Surveys*, vol. 50, no. 5, 2017, ISSN: 15577341. DOI: 10.1145/3104031.

[32] Jesús Barrasa, *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?* 2017. [Online]. Available: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/.

[33] *Franka Emika - The Robotics Company*. [Online]. Available: https://www.franka.de/.

[34] M. Afonso, H. Fonteijn, F. S. Fiorentin, D. Lensink, M. Mooij, N. Faber, G. Polder, and R. Wehrens, "Tomato Fruit Detection and Counting in Greenhouses Using Deep Learning," *Frontiers in Plant Science*, vol. 11, no. November, pp. 1–12, 2020, ISSN: 1664462X. DOI: 10.3389/fpls.2020.571299.

[35] M. Benavides, M. Cantón-Garbín, J. A. Sánchez-Molina, and F. Rodríguez, "Automatic tomato and peduncle location system based on computer vision for use in robotized harvesting," *Applied Sciences (Switzerland)*, vol. 10, no. 17, 2020, ISSN: 20763417. DOI: 10.3390/app10175887.

[36] *Apache TinkerPop Glemin*. [Online]. Available: https://tinkerpop.apache.org/gremlin.html.

[37] *TinkerPop Documentation*. [Online]. Available: http://tinkerpop.apache.org/docs/current/reference/#_tinkerpop_documentation.

[38] Umesh Prabushitha Jayasinghe, *gremlin-visualizer: visualize the graph network corresponding to a gremlin query*, 2021. [Online]. Available: https://github.com/prabushitha/gremlin-visualizer.

[39] World Wide Web Consortium (W3C), *QUDT*. [Online]. Available: http://qudt.org/.

[40] J. X. Yu and J. Cheng, *Managing and Mining Graph Data*. 2010, vol. 40, Chapter 7, p 217, ISBN: 978-1-4419-6044-3. [Online]. Available: http://www.springerlink.com/index/10.1007/978-1-4419-6045-0.

[41] H. Thakkar, S. Auer, and M. E. Vidal, "Formalizing Gremlin pattern matching traversals in an integrated graph Algebra," *CEUR Workshop Proceedings*, vol. 2599, pp. 1–20, 2019, ISSN: 16130073.

[42] T. K. Tu, J. D. Moorman, D. Yang, Q. Chen, and A. L. Bertozzi, "Inexact Attributed Subgraph Matching," *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*, pp. 2575–2582, 2020. DOI: 10.1109/BigData50022.2020.9377872.

[43] E. Bengoetxea, "Inexact graph matching using estimation of distribution algorithms," Ph.D. dissertation, 2002, p. 259. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.4803&rep=rep1&type=pdf.

[44] *PROV-Overview*. [Online]. Available: https://www.w3.org/TR/prov-overview/.

Appendix

Physical Relation referring to a Free Body Diagram

An example of a physical relation between a robot gripper and a truss of tomatoes is the force balance between them. A common used model to represent such force balance is a free body diagram as depicted in figure 5.1. Here the pressure applied by the gripper imposes a upward friction force in opposite direction of the gravitation force on the tomato truss. In that way, the gripper is constraining the truss.

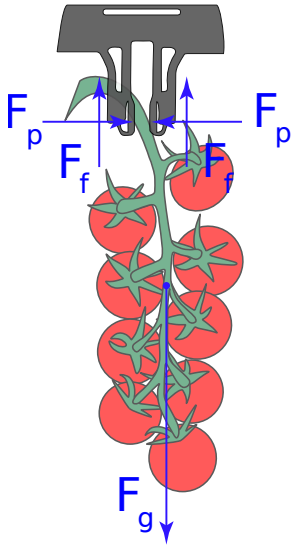


Figure 5.1: Free Body Diagram of a tomato truss held by a gripper

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

…27 september 2021……..……..

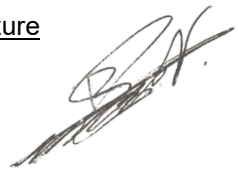
Name

…B.R. Herremans……..……..

ID-number

…0970880……..……..

Signature



……..……..

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU