UNIVERSITY OF BREMEN

INSTITUTE FOR
ARTIFICIAL INTELLIGENCE

Actionable Knowledge Graphs

How Daily Activity Applications can Benefit from Embodied Web Knowledge

Michaela Kümpel

Vollständiger Abdruck der vom Fachbereich Mathematik und Informatik der Universität Bremen genehmigten Dissertation zur Erlangung des Grades

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

Vorsitzender:	Prof. Dr. Dieter Hutter *Universität Bremen*
1. Prüfer:	Prof. Dr. hc. Michael Beetz, PhD *Universität Bremen*
2. Prüfer:	Prof. Frank van Harmelen, PhD *Vrije Universiteit Amsterdam*
Beisitzer:	Prof. Dr. Nico Hochgeschwender, *Universität Bremen*

Die Dissertation wurde am 20.12.2023 bei der Universität Bremen eingereicht und durch den Prüfungsausschuss am 19.03.2024 angenommen.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Bremen, den April 12, 2024

Michaela Kümpel

Abstract

Knowledge Graphs have been a research trend. The idea to acquire knowledge from the Web and make it accessible for users to answer queries has resulted in many famous knowledge graphs in industry (e.g. Google, Facebook or Amazon) as well as research (e.g. wikidata, WordNet or FrameNet). Due to their success, one can come to the conclusion that knowledge graphs might also help robots in answering complex queries in the same way that they help humans. In order to access knowledge graphs in robotic applications, one has to initially understand how robot control systems work. A robot can use a range of sensors to perceive its environment and build an internal belief state, a model of the environment, to make sense out of the perceived data. This means that a robot will use one of its sensors (e.g. a camera) to perceive an apple on the table in front of it. In its belief state it will then store the information that there is an apple and a table available in the environment (which are set in relation to the robot by storing exact object positions) but it might add the information that the apple is *on the table*, which more specifically relates the position of the apple to the position of the table as well as the robot itself in its belief state. If the robot is given the task to "cut the apple", the robot will access its belief state to reason about where it might find an apple to then perform the given task by relating the task to the environment information of its belief state. A belief state can also link to a knowledge base with additional knowledge that the robot might need. If we change the given task to "give me the milk" but do not change the environment so that the robot still only detects an apple on a table, the robot would not be able to fulfil this task without additional knowledge. However, if the belief state is linked to a knowledge base that offers the information that milk is a perishable product that usually is stored in a fridge, the robot can access this knowledge base to infer that it must search for a fridge in order to accomplish the given task.

Much of such additional knowledge that is needed to perform tasks is available on the Web. An agent that is enabled to access knowledge graphs containing Web

information can infer that the apple perceived on the table is a pome fruit of a food taxonomy, which has a core that usually is removed before eating. However, in order for a robot to access this information, we have to translate the object knowledge contained in a knowledge graph to environment information perceived by an agent in its belief state. What is more, if we want the robot to know how to hold the apple in order to cut it or how to remove the core of the apple, both the object and environment information needs to also be linked to action information. A knowledge graph that links object to environment and action information *makes the contained knowledge actionable*.

This Thesis proposes a five-step methodology for creating actionable knowledge graphs that follows existing knowledge engineering standards but links object knowledge to environment and action knowledge to enable various applications in daily environments, on different agents. The methodology is exemplary applied in two scenarios with different foci to create a product knowledge graph and a food cutting knowledge graph. The product knowledge graph aims at enabling omni-channel applications in unknown environments. It therefore contains product-related knowledge that is used by different agents such as smartphone, smart glass and robot, which aim at providing shopping assistance in a retail store. In order to provide user assistance like routing a customer to a searched product on different devices such as robot or smartphone, this scenario focuses on accessing relevant Web knowledge about products in a retail store that is linked to precise, reliable and agent-independent environment information. The food cutting knowledge graph aims at enabling robots to execute task variations of cutting actions. Here, the idea is to access Web knowledge to enable a robot to autonomously perform a range of cutting tasks. Therefore, this scenario focuses on how object information can influence action execution, how the needed knowledge can be acquired from the Web and how it can be modelled in a knowledge graph in such a way that a robot can use it to execute tasks. The methodology is validated by showcasing various applications that are enabled by the two exemplary knowledge graphs. The applications range from smartphone applications for shopping assistance that highlight interesting product features or route to a searched product over smart glass applications like shopping assistance and a recipe application to robot applications for shopping assistance and execution of cutting task variations on different fruits and vegetables.

Zusammenfassung

Wissensgraphen sind ein Forschungstrend. Die Idee, Wissen aus dem Internet zu akquirieren und es Nutzern zur Beantwortung von Anfragen zugänglich zu machen, hat zu vielen berühmten Wissensgraphen in der Industrie (z.B. Google, Facebook oder Amazon) und in der Forschung (z.B. wikidata, WordNet oder FrameNet) geführt. Aufgrund ihres Erfolges kann man zu dem Schluss kommen, dass Wissensgraphen auch Robotern bei der Beantwortung komplexer Anfragen helfen könnten. Um Wissensgraphen in Roboteranwendungen nutzen zu können, muss man jedoch zunächst verstehen, wie Robotersteuerungssysteme funktionieren. Ein Roboter kann eine Reihe von Sensoren verwenden, um seine Umgebung wahrzunehmen und ein internes Modell der wahrgenommenen Umgebung aufzubauen, welches die wahrgenommenen Daten in einen Zusammenhang bringt. Das bedeutet, dass ein Roboter mit einem seiner Sensoren (z.B. eine Kamera) einen Apfel auf dem Tisch vor ihm wahrnehmen kann. In seinem internen Modell speichert er dann die Information, dass in der Umgebung ein Apfel und ein Tisch vorhanden sind (die durch die Speicherung der genauen Objektpositionen in Beziehung zum Roboter gesetzt werden), aber er könnte zusätzlich die Information hinzufügen, dass der Apfel *auf* dem Tisch liegt, was die Position des Apfels mit der Position des Tisches sowie des Roboters weiter spezifiziert. Wenn der Roboter nun die Aufgabe "Schneide den Apfel" erhält, greift er auf sein internes Modell zu, um zu überlegen, wo er einen Apfel finden könnte, um dann die gestellte Aufgabe auszuführen, indem er die Aufgabe mit den Umgebungsinformationen seines internen Modells in Beziehung setzt. Das interne Modell kann dabei auch mit einer Wissensbasis verknüpft sein, die zusätzliches Wissen enthält, welches der Roboter möglicherweise benötigt. Wenn wir die gegebene Aufgabe in "Gib mir die Milch" ändern, aber die Umgebung nicht verändern, so dass der Roboter weiterhin nur einen Apfel auf einem Tisch erkennt, wäre er ohne zusätzliches Wissen nicht in der Lage, diese Aufgabe zu erfüllen. Wenn das interne Modell jedoch mit einer Wissensbasis verknüpft ist, die die Information enthält, dass Milch ein verderbliches Produkt ist, das normalerweise in einem Kühlschrank

aufbewahrt wird, kann der Roboter auf diese Wissensbasis zugreifen und daraus ableiten, dass er nach einem Kühlschrank suchen muss, um die gegebene Aufgabe zu erfüllen.

Ein Großteil dieses zusätzlichen Wissens, das zur Erfüllung von Aufgaben benötigt wird, ist im Internet verfügbar. Ein Agent, der in der Lage ist, auf Wissensgraphen mit Web-Informationen zuzugreifen, kann daraus schließen, dass es sich bei dem Apfel auf dem Tisch um ein Kernobst einer Lebensmitteltaxonomie handelt sowie dass Kernobst ein Kerngehäuse hat, das normalerweise vor dem Verzehr entfernt wird. Damit ein Roboter jedoch auf diese Informationen zugreifen kann, muss das in einem Wissensgraphen enthaltene Objektwissen mit dem internen Modell des Agenten verbunden werden. Wenn der Roboter außerdem wissen soll, wie er den Apfel halten muss, um ihn zu schneiden, oder wie er das Kerngehäuse entfernen muss, müssen sowohl die Objekt- als auch die Umgebungsinformationen mit Aktionsinformationen verknüpft werden. Ein Wissensgraph, der Objekt-, Umgebungs- und Handlungsinformationen verknüpft, macht *das enthaltene Wissen handlungsfähig*.

In dieser Arbeit wird eine fünfstufige Methodik zur Erstellung von handlungsfähigen Wissensgraphen vorgeschlagen, die sich an bestehenden Standards der Wissensmodellierung orientiert, dabei aber Objektwissen mit Umgebungs- und Handlungswissen verknüpft, um verschiedene Anwendungen in alltäglichen Umgebungen und auf unterschiedlichen Agenten zu ermöglichen. Die Methodik wird beispielhaft in zwei Szenarien mit unterschiedlichen Schwerpunkten angewandt, um einen Produktwissensgraphen und einen Wissensgraphen zum Schneiden von Lebensmitteln zu erstellen. Der Produktwissensgraph zielt darauf ab, Omnichannel-Anwendungen in unbekannten Umgebungen zu ermöglichen. Er beinhaltet daher Produktinformationen, die von verschiedenen Agenten wie Smartphones, Smart Glasses und Robotern genutzt werden, um Nutzer beim Einkauf in einem Einzelhandelsgeschäft zu unterstützen. Um z.B. den Nutzer auf verschiedenen Geräten wie Roboter oder Smartphone zu einem gesuchten Produkt zu leiten, konzentriert sich dieses Szenario auf den Zugriff auf relevante Produktinformationen aus dem Internet, die mit präzisen, zuverlässigen und agentenunabhängigen Umgebungsinformationen verknüpft ist. Der Wissensgraph für das Schneiden von Lebensmitteln zielt darauf ab, Roboter in die Lage zu versetzen, Aufgabenvariationen von Schneideaktionen auszuführen. Daher konzentriert sich dieses Szenario auf die Frage, wie Objektinformationen die Ausführung von Aktionen beeinflussen können, wie das benötigte Wissen aus dem Internet gewonnen werden kann und wie es in einem Wissensgraphen so modelliert werden kann, dass ein Roboter es zur Ausführung von Aufgaben nutzen kann. Die Methodik wird anhand verschiedener Anwendungen, die durch die beiden beispielhaften Wissensgraphen ermöglicht werden, validiert. Die Anwendungen reichen von Smartphone-Anwendungen zur Einkaufsunterstützung, die interessante Produktmerkmale hervorheben oder zu einem gesuchten Produkt führen, über Smart-Glass-Anwendungen wie Einkaufsunterstützung und eine Rezeptanwendung bis hin zu Roboteranwendungen zur Einkaufsunterstützung und Ausführung von Schneideaufgaben auf verschiedenen Obst- und Gemüsesorten.

Acknowledgements

I am not the type to really read acknowledgements but rather dive into the text. If you do, this is for you. And if you don't know me, even more so. Me, I am a mom of three. Why do I tell you this? Because it still makes a difference. I was naive enough to think that I could just as easily get my PhD as before kids, or as men. Unfortunately, even in the 2020s, it is not easy. You need nerves, and people that support you. If by any chance you are a mother reading this, please feel encouraged to pursue your goals. Also, feel free to contact me. I will try to support you in the ways I can. This already leads me to thank the person that helped me through my journey: Sabine Veit. Without you I most likely would have given up due to university bureaucracy. Thank you so much, I will try my best to be a similar help to others! Next, thank you Michael Beetz for believing in me. I remember the day when you asked me if I really wanted to work towards a PhD (to which I nodded enthusiastically). You could have questioned me, like others, but you believed in me and supported me. Thank you for that. You are an inspiration and I appreciate all the 5 minute talks in front of the coffee machine where you just shared the brightest ideas. Due to both our restricted time schedules I learned to pack all important things into such a 5 minute discussion - this will surely be very helpful in the future as well! Next I want to thank my family, and most importantly my kids. You actually deserve the most gratefulness. Thank you for your patience, for dealing with me being in submission mode and not having time for your problems, for me being late, not watching your sports game but rather working or not preparing lunch, for your support. I love you. Last but not least, I want to thank all my students for their works and my colleagues and outside collaborators for the discussions and cooperation that have led to this Thesis.

Acronyms

API Application Programming Interface

AR Augmented Reality

bfo Basic Formal Ontology

cdno Compositional Dietary Nutrition ontology

ChEBI Chemical Entities of Biological Interest

CRAM Cognitive Robot Abstract Machine

DL Description Logic

doid human disease ontology

DUL DOLCE+DnS Ultralite

ETL extract load transform

FDA Food & Drug Administration

FoodKG Food Knowledge Graph

FoodOn Food Ontology

GTIN Global Trade Identification Number

hpo human phenotype ontology

KnowRob Knowledge Processing for Robots

LLM Large Language Model

NDRF National Drug File

NEEM Narrative enabled episodic memory

OBO Foundry Open Biological and Biomedical Ontology Foundry

OWL Web Ontology Language

QR code Quick response code

RDF Resource Description Framework

RDFS Resource Description Framework Schema

RT-2 Robotics Transformer 2

semDT semantic Digital Twin

SOMA Socio-physical Model of Activities

SPARQL SPARQL Protocol And RDF Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the visions of AI-powered and cognition-enabled robotics are autonomous household robots that can accomplish everyday tasks. Everyday tasks in household environments require robots to perform manipulation actions including pouring, cutting or cleaning in natural contexts. Household robots have the potential to support us in a number of daily activities from meal preparation to elderly care and some robots already are able to clean the breakfast table (Kazhoyan et al., 2021) or prepare drinks (Sung and Jeon, 2020) and pancakes (Beetz et al., 2011b). Yet, the ability of robots to perform new task variations, act in unknown environments or support us in more complex everyday tasks such as cleaning dishes is still very limited. In particular, robots fail in situations where they are challenged by new tasks, new environments or new objects for which they lack knowledge.

If we reduce the scope and look at digital agents[1] such as smartphone applications that support us in daily activities, we can see that digital agents are similarly challenged by new environments and new objects. Digital agents like shopping assistance applications shown in (Davis et al., 2006; Álvarez Márquez and Ziegler, 2017; Mora et al., 2020), codecheck[2] or food recommenders like FoodWiki (Çelik et al., 2015) or Food4healthKG (Fu et al., 2023) have proven to be able to answer

[1]This work distinguishes between a *robotic* and a *digital agent*, where a *robotic agent* is any agent with a sensor and an effector like a robot arm and a *digital agent* is any agent with a sensor but without effector like a smartphone or smart glasses.

[2]codecheck app: https://codecheck-app.com/de/

complex queries and offer appropriate user assistance on websites or in a given environment. These applications usually rely on the recognition of barcodes or images of products. While these applications offer lots of information in a given environment, the transferability of such applications to new environments, particularly to products without barcodes like meat or fruit, is very limited. I claim that if digital agents can be enabled to use semantic environment information to detect objects that are not in the field of view or use environment maps created by different agents like precise camera systems, they can overcome this limitation and be able to answer user requests in new environments.

> Hypotheses 1:
> Enabling (digital) agents to recognise products without a barcode, that change in appearance or are not in the field of view, will enable them to provide information in new environments or on new objects.

Looking at robotic agents that are to autonomously perform tasks in daily environments, we do not only have to consider environment information but also action information. A robot that knows how to cut a slice of bread but is given the task to "Quarter an apple" needs to be able to infer that quartering is a variation of cutting, what tool needs to be used for quartering, and how to move its body to successfully perform the task. Although such knowledge can be implemented manually into a robot knowledge base, it is unfeasible to manually model it for all meal preparation tasks that a robot might need to perform. If we consider how humans would behave in such a situation, the Web as a knowledge source providing information comes into mind. The Web offers plenty of sources that can be accessed to retrieve information. A human that wants to know how to prepare a new recipe might use the Web to search for preparation instructions or instruction videos. I argue that a robot that is enabled to access Web information and reason about it would similarly benefit from it.

> Hypothesis 2:
> If robots can be enabled to access and understand Web information, they can use it for action execution.

Much of the knowledge needed to refine and execute vague tasks such as "Cut the bread" or "Get me lactose-free milk" and transfer them to new task variations is contained in instruction Websites like WikiHow, encyclopedic Websites like Wikipedia and WordNet (Miller, 1995), and many other web-based information sources. The idea to access the Web for an integration of object knowledge into robotic applications has been around since the 2000s (e.g. (Fensel and Bussler, 2002)) and exemplary applied to robotics in 2011 (Tenorth et al., 2011; Waibel et al., 2011). Web knowledge is distributed on many Websites and comes in different formats that often are not machine-readable. What is more, knowledge sources offering general knowledge like encyclopedic knowledge or taxonomies usually are not linked to knowledge sources offering specific knowledge like product data or object properties, thus making it hard to link specific objects at hand to general object knowledge. I argue that if we can link general knowledge about actions and objects to specific knowledge about object properties and tasks, robotic agents can use the knowledge to understand task variations as well as the role of objects in a given task, thus enabling them to access Web information similarly to humans.

Hypothesis 3:
If we can link specific and general object and task knowledge from distributed Web sources, a robot can use the knowledge to understand task variations as well as the role of tools and objects in given tasks.

In the World Wide Web there have been developments towards a Semantic Web (Berners-Lee et al., 2001), an extension of the current Web (Sirin et al., 2003). These developments have made knowledge graphs (see Sec. 2.1.2) popular for knowledge representation ever since Google proclaimed to be using knowledge graphs in 2012 (Blog, 2012). On the one hand, this has led to large web-based knowledge sources like wikidata, FrameNet (Baker et al., 1998) or the LOD cloud (Fernández et al., 2017) and it has been shown that such sources are a powerful tool that can be used to efficiently answer questions and reason about the contained information such as in health specific consumer support applications like recipe recommenders (e.g. (Cantais et al., 2005; Haussmann et al., 2019)). On the other hand, such example applications as well as open sources like wikidata are not applicable in agent applications that link to user environments like providing recipe recommendations for leftover products. I argue that the Semantic Web contains lots of valuable knowledge that is useful for daily applications and especially for robotic applications where robots need to reason about available information. However, if robots are to use knowledge graphs, knowledge graphs need to be linked to the perceived objects of the environment.

Hypothesis 4:
In order to make the existing (Semantic) Web knowledge applicable for agents in daily applications, there is a need to link such knowledge sources to the real world so that agents can relate perceived objects to Web knowledge and reason about the information.

Through this, Web knowledge can enable digital agents like smartphone applications to link recipe recommendations to products at hand or robotic applications to link object information like food properties to an environment map of product locations.

For robotic manipulation applications, we additionally need to consider action information. A robot needs to be able to translate task requests into body movements and reason about object information to distinguish between cutting a slice of bread and quartering an apple. Everyday manipulation tasks, such as "Cut it into pieces" are broadly scoped as they include, for example, cutting a large variety of fruits, vegetables, bakery items or meat, with a variety of tools, for a variety of purposes, in many different contexts. For this, (Web) knowledge needs to additionally be translatable to body movements of a robot, thus making the knowledge *actionable*.

The key question is: How can we make such abstract knowledge *actionable* for agents?

To answer this question, we have to start looking at the perception action loop of agents. Figure 1.1 is based on the perception action loop proposed by Russel

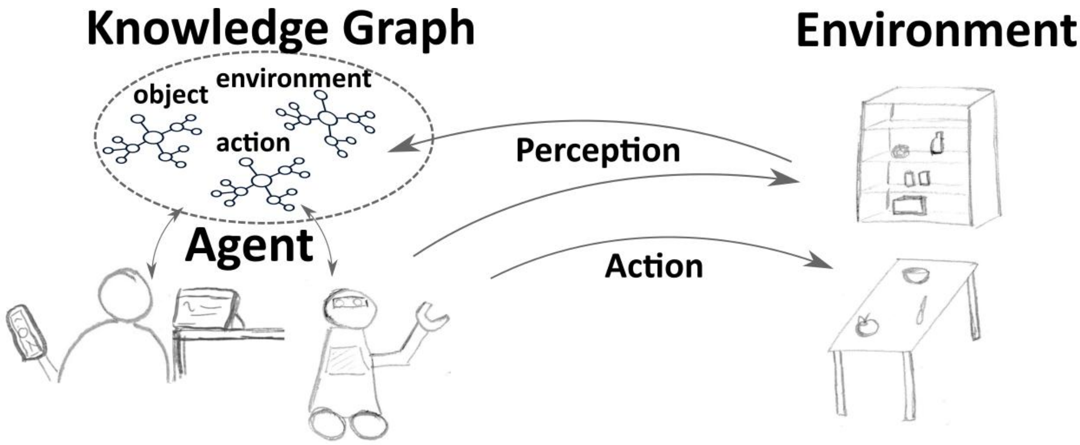

Figure 1.1: Basic perception action loop where an agent perceives its environment to create a belief state, linked to a knowledge graph and action knowledge, to then perform an action.

and Norvig, where an agent is defined as *"anything perceiving its environment through sensors and acting upon that environment through effectors"* (Russell, 2010), p. 31-32. Since digital agents have no effector, they only sense their environments, while the user acts upon it. Fig. 1.1 appends the model of Russel and Norvig by adding **environment**, **object** as well as **action** knowledge as knowledge sources for the agent belief state, which I model as a knowledge graph. A robot uses its sensors to perceive its environment and build an internal belief state, a model of the environment, to make sense out of the perceived data. The **environment** knowledge of the knowledge graph therefore consists of environment information sensed by the agent as in positions of objects perceivable by the agent, which is semantically enhanced and generalised through relations such as number of shelf floors of each shelf or doors of a cupboard, their height, the type of handle, etc. The **object** knowledge consists of general object information like a classification of objects (e.g. a milk is a dairy product, and is usually stored in a fridge) as well as their specific properties (e.g. objects with this barcode are of type milk and come in a glass bottle), which has to be linked to the environment information in order for agents to successfully make the contained knowledge applicable. **Action** knowledge is knowledge about action categories as well as the translation of task requests into body movements. For example, a cutting action will always be broken down into the same body movements, which can be parameterised based on the agent performing the task, available objects, etc.

The idea behind this is that if an agent has access to a knowledge graph with its object information (i.e. “milk is a dairy product”, “milk is usually stored in Tetra Paks”, “a Tetra Pak is a container for liquids”, “milk is usually stored in a fridge”), which can be grounded in real world environments with exact object positions (i.e. “Object A is a fridge”, “Object B with barcode 123 is a milk”) as well as action information (i.e. for the task to “put the milk in the fridge” the agent has to **pick up** object B → **lift** object B → **navigate to** object A → **open** object A → **place** object B **into** object A), the contained knowledge becomes perceivable and actionable to users and robots. I argue that Web knowledge can be made actionable if it is linked to action knowledge that a robot can translate to body movements.

> Hypothesis 5:
> In order to make the existing (Semantic) Web knowledge *actionable* for robotic agents, the knowledge needs to be translatable to body movements of the robot.

In summary, to enable agents to successfully and autonomously perform actions in new environments, on new objects and for task variations, I hypothesise that:

1. Agents that can access semantic environment information are able to make sense of new environments and thus operate in them.

2. Agents that are enabled to access Web information like humans do can similarly use it for action execution.

3. In order for robots to access Web information, distributed sources offering general and specific object knowledge need to be linked.

4. To further make the Web knowledge applicable in daily applications, it needs to be linked to perceived environment information.

5. Web knowledge can be used for execution of actions (and thus task variations) if, and only if, the knowledge is linked to action knowledge that can be translated to robot movements.

1.1 Challenges

Making knowledge graphs actionable entails several challenges. The following section will detail the challenges in creating knowledge graphs for agents that need to contain general and specific knowledge that is interlinked and accessible, in integrating environment information and linking it to object information and lastly, in translating Web knowledge to robot body movements.

1.1.1 Creation of Knowledge Graphs for Agents

There has been lots of research on creation of knowledge graphs. However, no one-fit-all construction approach exists and no methodology for creation of knowledge graphs for agent applications has been proposed. Many existing knowledge graphs focus on properly modelling a domain of interest to offer support to humans. Thus, they provide information that a Granny Smith apple is a pome fruit and a pome fruit has a core with seeds, for example. An agent however does not know what a Granny Smith apple is. It also does not know that humans usually remove the core of an apple before eating or the shape an apple has. Such knowledge is important if the agent is given a task to "Cut the apple".

Therefore, actionable knowledge graphs have to include such common-sense object information that is relevant for action execution and semantically encoded so that agents can reason about it.

> This reveals the challenges to 1) define the knowledge and object properties an agent needs for action execution, 2) acquire the needed data and 3) link it in such a way that an agent can query it.

1.1.2 Integration of Environment Information

Household environments are changing, not standardised and open. Therefore, actionable knowledge graphs need to encode environment information semantically and link it to background information in order to model that milk is usually stored in the fridge but long-life milk can as well be stored at room temperature and thus in a cabinet or storage room, for example.

Additionally, robotic agents use a different reference frame in their environments than digital agents. If both agents are to access environment information, the environment information needs to be translatable so that the different agents can successfully exchange information and route a user to a searched location, for example.

Lastly, even if environment information is available in machine-readable form for different agents, it needs to be linked to object information for an agent to know that an object perceived on the table, which is round and red, is a Granny Smith apple, some other apple, or a cherry.

> This reveals the challenges to 4) find ways to acquire and standardise Web knowledge from distributed sources that offer the information in different formats, 5) encode environment information in such a way that different agents can use it in applications and 6) link perceived objects in the environment to objects of a knowledge graph.

1.1.3 Integration of Action Information

A robot that is to execute task variations needs to know that slicing, dicing and quartering are action verbs that belong to the action category of cutting. It further needs to know that chopping and dicing are similar in their execution, while they are different to slicing and halving.

In addition to such an action classification, an agent needs to know how different action verbs influence action parameters. This means that for a given action category, the factors that influence action execution (e.g. position, speed or angle) need to be defined. What is more, the action plan of the robot needs to be adapted in such a way that it can be parameterised for execution of task variations.

For successful action execution an agent needs to also link action to object information. For the task of "cutting the apple" the agent needs to know if the given object is an apple, if it can be cut and what tool it can use to cut the apple.

This reveals the challenges to 7) create an action classification an agent can use for action execution, 8) determine parameters that influence action execution and provide the robot with ways to use these parameters for an execution of task variations and 9) relate object information to task information that a robot can reason about for action execution.

1.2 Approach for Creating Actionable Knowledge Graphs

The following section describes my approach to solve the given challenges. I propose to create actionable knowledge graphs out of Web information that link object to environment and action information, use semantic Digital Twins for a semantically enhanced environment representation that can link to object information and parameterise robot action plans with actionable knowledge graphs to enable robots to perform task variations.

1.2.1 Gathering and Linking the Needed Knowledge for Actionable Knowledge Graphs

Previous work proposed a methodology for knowledge engineering which is capable of creating ontologies out of Web information that formalise hierarchies of tasks and link them to the respective types of actions to make them actionable for robots (Töberg et al., 2023). The work has shown how challenges 1) - 3) can be solved. This Thesis extends the idea and generalises the methodology for creation of different actionable knowledge graphs that can be used by various agents, in new environments and for execution of new task variations. In particular, this Thesis describes how object knowledge graphs can be created from Web information to then link them to environment and action information.

1.2.2 Using semantic Digital Twin Environments in Actionable Knowledge Graphs

The concept of a *Digital Twin*, *"the digital equivalent to a physical product"* was introduced by Grieves in 2003 (Grieves, 2011). Using Digital Twins to create exact virtual representations of the real world has been a technology trend in digitisation of manufacturing and industry (Augustine, 2020). A *semantic Digital*

Twin (semDT) can be described as the semantic connection of object information into a digital representation of an environment. While a semDT offers detailed environment information, it only offers limited object or product information from connected enterprise management systems. Prior work has demonstrated that semDTs can successfully be used for linking object to environment information in the retail domain for customer support (Kümpel et al., 2021, 2023), thus providing a solution for challenge 6). SemDTs can additionally be used for user support in daily activities on different agents as shown in (Kümpel and Beetz, 2023) and therefore offer a solution for challenge 5). I therefore extend the idea and link semDTs to knowledge graphs of domain-specific object information acquired from Web sources to solve the challenge in 4).

1.2.3 Using Actionable Knowledge Graphs to Parameterise Robot Action Plans for Flexible Manipulation

If robots are to further use knowledge graphs for action execution, knowledge graphs have to not only be connected to environment information but also include action information that can be broken down into body movements. Only then the robot knows how to move its body in order to achieve the goals implied by a request and avoid causing unwanted side effects. An action classification further helps to differentiate tasks. In the case of cutting, action verbs should be classified in regards to their execution. This means that a chopping task will be more closely related to a dicing task since both have result objects in similar amount and shape. In addition to this action classification, task dependencies should be available. A quartering task depends on the execution of a halving task, for example. Modelling such knowledge can solve the challenge in 7).

The body motions of robots can be stated in terms of motion constraints and motion objectives. For example, when picking up a knife, example constraints are to hold the knife by its handle and to keep the blade horizontal, and objectives include generating smooth motions and minimising acceleration to avoid contacting anything other than a cuttable object and its underlying surface. Using online projection (Kazhoyan and Beetz, 2019), actions can be broken down into a small set of body movements with different parameterisation. Consider the action of cutting and the body movements as depicted in Figure 1.2. Each

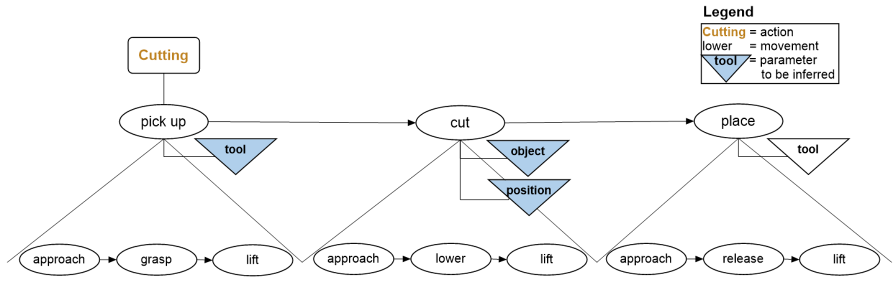

Figure 1.2: Cutting action broken down into body movements of the robot and the parameters that need to be inferred.

cutting action can be broken down into a sequence of tasks: **pick up** → **cut** → **place**. These tasks can then be broken down into body movements. For picking up, they would be: **approach** → **grasp** → **lift**. For cutting, they would be **approach** → **lower** → **lift** and for placing **approach** → **release** → **lift**. Even with this small example it becomes clear that many different actions can be broken down into the same body movements that only generate different behaviour because of their parameterisation. For example, while in the main cutting action for the **approach** body movement the robot has to know to approach a cuttable object in a certain cutting position above it and with the arm holding the tool, at the same time it has to be able to infer that for the picking up task it has to approach the tool that can be used for cutting with an arm it can use for cutting and in a position that voids collision with other objects. To solve the challenge in 8) one therefore has to define the parameters that might influence each of these given body movements and make them available as parameters in the general action plan.

With such generalised action descriptions, the ultimate reasoning task is to infer the constraints and objectives that the body motion of the robot should satisfy and maximise given a vague task request. For a cutting action, the robot would need to know which tool can be applied to which object. Additionally, object information that changes action execution, such as the existence of a peel that needs to be removed prior to cutting, needs to be available. Integrating the knowledge into the action plan solves the challenge in 9).

1.3 Example Scenarios

This work describes a methodology for creating actionable knowledge graphs and their applications for two example scenarios:

1. **Omni-channel User Assistance** The user assistance scenario aims at supporting users in daily environments like retail stores or at home. A focus lies on omni-channel applications (Taylor et al., 2019) providing a seamless shopping experience that is independent of the used device (i.e. channel) or environment.

   For this, the ProductKG knowledge graph (Kümpel and Beetz, 2023) was created. It contains product information like ingredients, allergens, labels or brands, which was acquired from various websites. The ProductKG knowledge graph links to a semDT created by a robotic agent scanning the store as in (Beetz et al., 2022), which is transformed to environment information that can also be used by digital agents. The ProductKG knowledge graph can be used in various applications such as a product finder smartphone app, a service application for smart glasses to highlight product information according to user preferences or for service robots to route customers to a searched product, amongst others. Due to its modularity, it can just as well be used in household environments to highlight recipes that can be prepared with a product in the fridge, for example.

2. **Robots Executing New Task Variations** In this scenario, the goal is to empower robot agents to autonomously and flexibly execute a range of underspecified task requests for a set of different objects, namely fruits and vegetables, in a given action category, namely cutting.

   In (Töberg et al., 2023), a methodology for creating an ontology for performing new task variations of a cutting action was proposed. Here, a robot that knew how to cut a slice of bread as discussed in (Dhanabalachandran et al., 2021) should be enabled to quarter an apple. This work builds upon that methodology, generalises it and details how robots can use an actionable knowledge graph for executing new task variations on new objects as in quartering an apple or chopping a cucumber. The food cutting knowledge graph links environment to object and action information and can be used by robotic agents for execution of task variations.

1.4 Contribution

This thesis builds upon the mentioned previous works to create knowledge graphs that link object to environment and action information. In particular, I present a methodology for creating actionable knowledge graphs by creating two distinct object knowledge graphs from Web information that link to environment and action information and can be used for various user applications in household environments by digital and robotic agents.

The contributions of this work are:

1. A five step methodology for creating actionable knowledge graphs for the use in different applications, domains and on different agents is proposed.

2. The knowledge acquisition, processing and linking methods needed for creating actionable knowledge graphs are described in detail for two example knowledge graphs with different focus: a product knowledge graph for user applications like recipe recommender and product finder, as well as a food cutting knowledge graph robots can use to execute new task variations of cutting actions.

3. Solutions for representing environment information for digital and robotic agents and connecting it to object information are presented. Therefore, semDTs of environments are created, linked to the object knowledge graphs and then accessed by both digital and robotic agents to show its applicability.

4. A solution for integrating action information in knowledge graphs and linking it to object information for task execution is proposed. While the solution is discussed for task variations of cutting, this Thesis also shows how it can be transferred to other meal preparation tasks like pouring.

5. A new approach for translating action knowledge to parameters that influence body movements of the robot using a knowledge graph that the robot

can reason about is presented.

6. The example knowledge graphs are made publicly available on websites with tutorials and standard SPARQL APIs through triply[3] for accessibility. The ProductKG website including example queries that can be accessed through a grlc API (Meroño-Peñuela and Hoekstra, 2016) is available at `https://michaelakuempel.github.io/ProductKG/`, the ProduktKG application page is available at `http://productkg.informatik.uni-bremen.de`. The food cutting knowledge graph is available at `https://food-ninja.github.io/FoodCutting` with a web interface to query the graph. Jupyter notebooks for querying the knowledge graph and running a robot simulation of the food cutting project are available at `https://moodle.intel4coro.de/course/view.php?id=8`.

The contributions are validated by various applications that support users in daily activities using their smartphone, smart glasses or robot assistants in retail and household environments, some of which have already been shown in prior publications:

Kümpel M, Mueller CA, Beetz M. **Semantic Digital Twins for Retail Logistics**. In *Dynamics in Logistics: Twenty-Five Years of Interdisciplinary Logistics Research in Bremen, Germany* 2021 Sep 22 (pp. 129-153). Cham: Springer International Publishing.

Kümpel M, Dech J, Hawkin A, Beetz M. **Robotic Shopping Assistance for Everyone: Dynamic Query Generation on a Semantic Digital Twin as a Basis for Autonomous Shopping Assistance**. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* 2023 May 30 (pp. 2523-2525).

M. Kümpel and M. Beetz. **ProductKG: A Product Knowledge Graph for User Assistance in Daily Activities**. In *Ontology Showcase and Demonstrations Track, 9th Joint Ontology Workshops (JOWO 2023), co-located with*

[3]The datasets can be found at: `https://krr.triply.cc/mkumpel/`

FOIS 2023, 19-20 July, 2023, Sherbrooke, Québec, Canada, 2023.

M. Kümpel, J.-P. Töberg, V. Hassouna, M. Beetz and P. Cimiano. **Towards a Knowledge Engineering Methodology for Flexible Robot Manipulation in Everyday Tasks**. In *Workshop on Actionable Knowledge Representation and Reasoning for Robots (AKR*^{3}*) at Extended Semantic Web Conference (ESWC 2024)*.

M. Kümpel, V. Hassouna, J.-P. Töberg, P. Cimiano and M. Beetz. **Cut, Chop, Slice or Dice: Parameterising General Action Plans Using Knowledge Graphs**. In *International Conference on Intelligent Robots and Systems (IROS 2024)*. submitted.

A full list of prior publications can be found in Appendix A.

Parts of this work are based on supervised student Theses, which are listed in Appendix B.

1.5 Outline

The remainder of this work is structured as follows: **Chapter 2** is giving an overview over ontologies, knowledge graphs, Linked Data as well as their commonalities and differences. It continues to introduce and compare knowledge representation as well as other approaches for robotic action execution. In **Chapter 3** a five-step methodology for creation of actionable knowledge graphs is presented. The methodology distinguishes between object, environment and action information that needs to be defined, acquired, processed and linked for a given domain and target application. For robotic applications it further needs to be translated to body movements. In **Chapter 4** the methodology is exemplary applied for creation of a product knowledge graph and omni-channel applications, where the focus lies on encoding environment information in a flexible way to allow for different agents to use the contained knowledge in user support applications. After that, in **Chapter 5**, the methodology is applied for creation of a food cutting knowledge graph in order to enable robots to execute new task variations of cutting, where the focus lies on translating the contained knowledge to body movements of the robot. **Chapter 6** shows how different agents can access actionable knowledge graphs as well as various applications of both the product and food cutting knowledge graph. **Chapter 7** concludes.

Chapter 2

Ontologies and Knowledge Graphs for Knowledge Representation and Action Execution

This chapter gives an overview and demarcation of ontologies, knowledge graphs and Linked Data, shows how specifications of ontologies or knowledge graphs can be delineated using Description Logic and lastly points out how knowledge representation or other techniques can be applied for robotic action execution.

2.1 Background: Ontologies, Knowledge Graphs and Linked Data

The discussion on the commonalities and differences between ontologies and knowledge graphs are very controversial. Denny Vrandecic, the founder of wikidata, stated in 2022 *"What has been ontologies, now are knowledge graphs"*, inferring that there is no difference between an ontology and a knowledge graph. Many researchers disagree to this statement. Although popular examples of knowledge graphs like Google, Facebook or Amazon have greatly increased interest in the topic starting in the 2010s, we are still lacking a definition of a knowledge graph in comparison to an ontology.

Nevertheless, the following sections describe the terms ontology and knowledge graph and give a working definition of a knowledge graph for this thesis. To complete the list, the term Linked Data is also introduced. The chapter closes with an overview of related work.

2.1.1 Ontologies

In the field of artificial intelligence, Tom Gruber coined the definition of an ontology in 1993 as *"An explicit specification of a contextualisation"* (Gruber, 1993), which is based on the view of Genesereth and Nielsson who describe conceptualisations of objects that exist in the real world, and their interrelations (Genesereth and Nilsson, 1987). Grubers definition of an ontology was extended in the late 1990s, where the importance of a **shared** conceptualisation was highlighted and discussed (Borst, 1999; Studer et al., 1998). Optimally, such a shared conceptualisation has a formal description and is machine-understandable so that it can be understood, reused and extended.

Ontologies thus formally describe a domain of interest by defining objects and their interrelations. They can be specified using triples that delineate the relation between two objects such as `<milk_A is_a Cows_milk>` or between an object and its properties as in `<milk_A has_nutrient saturated_fatty_acids>`. To represent the contained knowledge, ontologies can use different standards like the Resource Description Framework (RDF), the Resource Description Framework Schema (RDFS) or the Web Ontology Language (OWL) to describe triples of entities and their properties (Hitzler et al., 2007)

`<subject predicate object>`

and thereby formally model the structure of a system by classifying knowledge chunks as subjects and objects, their hierarchy (called taxonomy), and their relations (as predicates) (Guarino et al., 2009). Through such a formal description, ontologies are machine-readable and can easily be reused as well as extended (Borst, 1999). What is more, using an ontology and ontology language enables programs to be fairly independent of the tasks that shall be executed, the robots or users to query the program, and their environments (Olivares-Alarcos et al., 2019) since the knowledge about tasks, agents and environments is stored as general knowledge chunks that are combined by their relations.

Furthermore, ontologies can be divided into different categories (Guarino, 1998):

- *Top-level ontologies* describe basic concepts like time or measurement that are applicable for many domains and use cases.
- *Domain or task ontologies* usually are based on a top-level ontology and describe a domain or task of interest with its concepts. There are domain ontologies for food, products, diseases or illnesses as well as task ontologies for disease diagnosis or recipe recommenders, for example.
- *Application ontologies* include both domain and task concepts and hence can include roles of objects.

2.1.2 Knowledge Graphs

In (Ehrlinger and Wöß, 2016), Ehrlinger and Wöß point out that even the creators of the YAGO2 knowledge graph (Hoffart et al., 2013) refer to their knowledge graph as an ontology (supporting the claim that both actually are the same) and evaluate the use of the term knowledge graph in order to define it. Ehrlinger and Wöß suggest that the difference of a knowledge graph and an ontology could be interpreted as a matter of quantity (i.e. a knowledge graph is a large ontology), extended requirements (i.e. a built-in reasoner makes an ontology a knowledge graph) or integration (i.e. a knowledge graph extends a knowledge-based system with an integration system) and conclude with this definition of a knowledge graph: *"A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge."* (Ehrlinger and Wöß, 2016). Noy et al. highlight the fact that usually large scale data from a variety of sources is integrated into knowledge graphs (Noy et al., 2019).

What these previous definitions are missing is the key aspect that actually made knowledge graphs like Google, Amazon, eBay, Facebook or LinkedIn (see Hogan et al. (2021)) successful: their major constituent being web information (i.e. any data that is available in the World Wide Web). In these knowledge graphs, web information can be textual information based on keywords found on different websites, semi-structured information like price comparisons gathered from different websites as well as structural data from a single source that semantically connects such information to create graphs of befriended people and their properties or products that might be bought together. In this thesis I extend the definition of Ehrlinger and Wöß to include both facts that knowledge graphs integrate web information but also incorporate information from a variety of sources such as Enterprise Resource Planning (ERP) systems or (local) ontologies (that give meaning to the contained facts and allow reasoning on it).

Thus, in contrast to other works, a knowledge graph in this thesis can contain both *assertional box* (ABox) and *terminology box* (TBox) descriptions. While *terminology box* knowledge defines the main objects of a domain of interest and their properties as a common vocabulary like in top-level ontologies that can be described as the ontology layer of the knowledge graph, assertional box data adds facts to such TBoxes by integration of large amounts of examples and therefore can be described as the factual layer of the knowledge graph. I also want to highlight the fact that a knowledge graph usually contains domain information since it is constructed to answer certain queries regarding a domain of interest and thus define a knowledge graph as:

> **Definition 2.1.1 (Knowledge Graph).** A knowledge graph acquires and integrates domain information from the Web, among a variety of sources, contains a factual as well as an ontology layer and can apply a reasoner to derive new knowledge.

2.1.3 Linked Data and Semantic Web

When structured data like ontologies or knowledge graphs are published on the *Semantic Web* and linked to each other, they are referred to as *Linked Data* (Bizer et al., 2011), enabling machines to surf the Web. If the data is openly accessible as well, it is called *Linked Open Data*. The *Semantic Web* (Berners-Lee et al., 2001) aims at structuring content of Web pages by using standardised formats to represent entities, their properties and relations in ontologies that can be understood by robotic or software agents.

2.2 Description Logic for Specification of Ontologies

To specify content of ontologies (or knowledge graphs), one can use Description Logic (DL)(Baader et al., 2017). DL axioms describe ontology content using statements such as in the following:

$$Orange \sqsubseteq Fruit \tag{2.1}$$

$$Orange \sqsubseteq \exists hasDisposition.Cuttability \tag{2.2}$$

$$Orange \sqsubseteq Fruit \ \sqcap \exists hasDisposition.Cuttability \tag{2.3}$$

$$Orange \sqsubseteq \exists hasPart.(Peel \sqcap \exists hasEdibility.MustBeAvoided) \tag{2.4}$$

$$Peelability \sqsubseteq (affordsTrigger.Peeler \sqcup affordsTrigger.Hand) \tag{2.5}$$

$$Orange \sqsubseteq \exists hasPart.(= \ 1 \ Peel) \tag{2.6}$$

$$Slice \ \dot{\cup} \ FoodPart \tag{2.7}$$

Statement (2.1) means that *an orange is a fruit*, where $\sqsubseteq$ denotes a subsumption relation. Axiom (2.2) means that *an orange is part of all things that have a cuttability disposition*, where $\exists$ denotes existence. To connect both statements one can use a $\sqcap$, a logical and as an intersection in statement (2.3), which translates to *an orange is a fruit and has a cuttability disposition.* The axiom in (2.4) connects two statements through a conjunction $\sqcap$ of the subset denoted through the square brackets [], which means *an orange has a peel and the peel must not be eaten.* Statement (2.5) instead connects two statements using a $\sqcup$, a union operator, which means that *the peelability affords both a peeler or a hand as triggers.* Statement (2.6) connects an object with a number restriction =, which in this case means that *an orange has exactly one peel* and statement (2.7) can be used to declare that two classes are disjoint, in this case *a slice is a disjoint class of FoodPart.*

2.3 Using Knowledge Representation for Robotic Action Execution

Robots need a cognitive architecture for successful task and motion planning and execution. Some cognitive architectures for robots are SOAR (Laird, 2019) and iCub (Vernon et al., 2010), but this work is based on the Cognitive Robot Abstract Machine (CRAM) (Beetz et al., 2010), which has successfully been used for robots executing household activities such as cooking (Beetz et al., 2011a) and setting the table (Kazhoyan et al., 2021) as well as for robotic retail store assistants (Cavallo et al., 2022) or even for robots executing chemical experiments (Lisca et al., 2015). CRAM supports extended failure handling such as proposed in (Diab et al., 2020), which is also described in (Dhanabalachandran et al., 2021).

The high-level plan executive of CRAM uses Knowledge Processing for Robots (KnowRob) (Beetz et al., 2018b) as a knowledge processing and reasoning framework that can load ontologies or knowledge graphs for knowledge representation.

Using the same frameworks CRAM and KnowRob, there has been research on plan projection for vaguely defined tasks (Kazhoyan and Beetz, 2019) as well as understanding and grounding natural language instructions (Nyga et al., 2018). For this, a robot that is told to "cut the cucumber", while knowing how to cut a bread and what a cucumber looks like, might actually be able to perform the task with the known action parameters since it calculates the most probable solution. However, if given the different task to "slice the apple" and the robot does not know the meaning of the word "slice" or its most probable search space, it would not be able to perform the task. Also, the robot needs to have knowledge about the apple's features and properties necessary for the "slicing" task to succeed as will be detailed in the following chapters.

Another approach introduced in (Mitrevski et al., 2021) focuses on the generalisation of already executable actions on unknown objects. By employing an object ontology, parameterised execution models for manipulation tasks like grasping or stowing can successfully be transferred to unknown objects.

In (Kümpel et al., 2020) we have presented the idea to use Linked Data to help robots understand product-related actions. The idea is visualised in Figure 2.1. Here, the robot connects its belief state of objects perceived in the environment to an ontology or knowledge graph offering additional product information like a classification similar to the product knowledge graph described in chapter 4 as well as to its action model so that the object knowledge can be used for action execution. The knowledge graph is further connected to outside sources like wikidata or open food facts and therefore can be called Linked Data. The robot

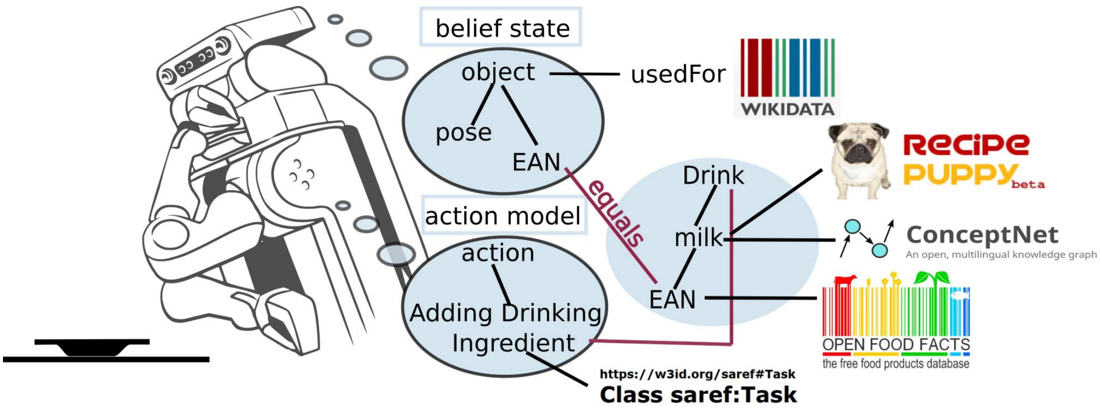

Figure 2.1: Connecting Robots to Linked Data.

accesses the knowledge through a propagation of existing links and, in theory, can answer many different questions. In practice however, using Linked Data poses a major challenge: query requests can not be propagated, i.e. an Application Programming Interface (API) and query service on a knowledge graph only offers to query the contained knowledge. Therefore, even if we create links to wikidata, we cannot obtain further information from wikidata through the knowledge graph API but must send an additional query to the wikidata API. Thus it is more practicable to define the needed knowledge, acquire it and integrate it in the knowledge graph so that wikidata links act more like a connector for modularisation. In that case the knowledge graph can also be used for a different application with another ontology based on wikidata entries.

2.4 Other Approaches for Robotic Action Execution

Correctly executing new task variations is still a big challenge in robotics due to the fact that tasks are often underspecified and assume common-sense knowledge about objects and the environment. There are approaches focusing on detection of task sequences to enable robots to infer the next step, using image recognition (Ramirez-Alpizar et al., 2021) as well as approaches to infer the next action to be performed based on natural language processing (Sera et al., 2021). However, these approaches are based on known actions.

Due to the great success of Large Language Models (LLMs), one idea is to use these language models or the even newer approach of the Robotics Transformer

2 (RT-2) to combine large language with vision models for robotic task execution instead of a knowledge representation approach.

Using Large Language Models for Action Execution The everyday robots project uses LLMs to ground robotic affordances and infer the everyday task to be executed (Ahn et al., 2022). Using this approach, the instructions usually have to be altered to be very specific for the agents to infer the tasks to be executed. What is more, LLMs rely on existing data. One problem with performing new task variations is that it is unclear if the needed data is available. Even then, most robots would have problems in task execution based on the given information. For example, the paper lists the following prompt as a basis for task experiments:

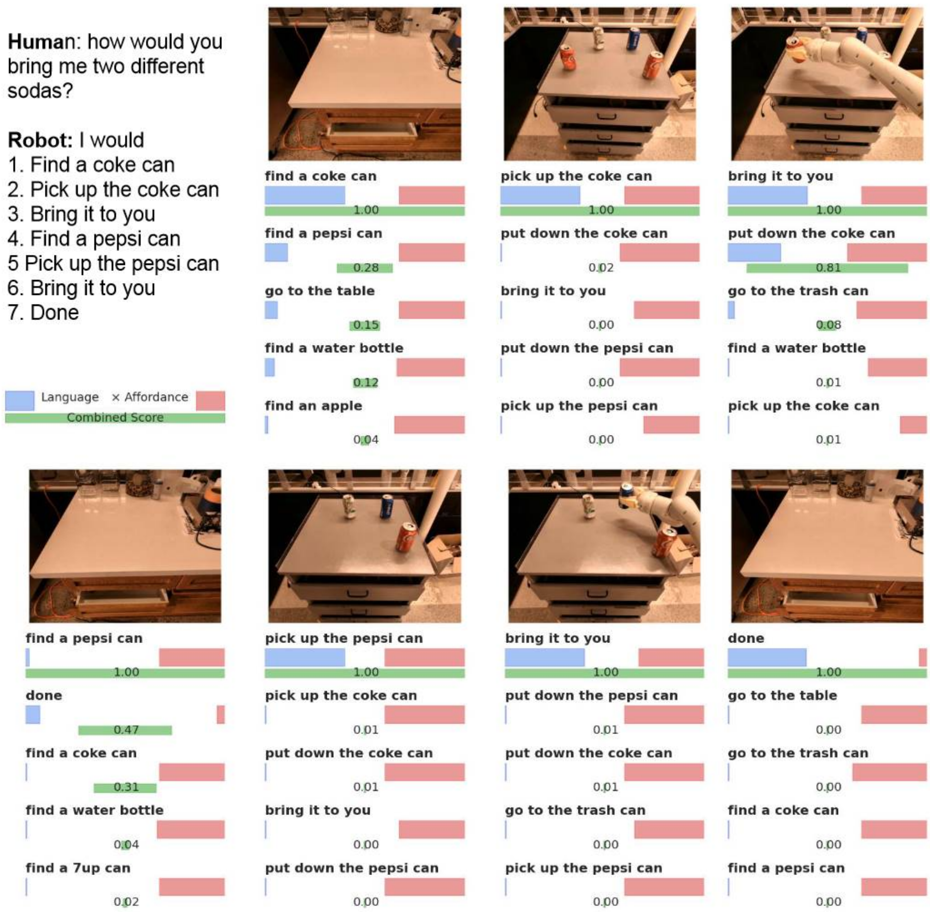

Figure 2.2: Example of robot using LLMs from (Ahn et al., 2022).

The paper shows that the robot used in their experiments was able to perform the given tasks based on the given prompts. Unfortunately, they never show where the robot actually placed the objects, which actually plays an important role. If we look at Figure 2.2, the user asks the robot to bring them two different sodas. Here, the robot randomly chooses to find a coke and a pepsi can. It remains unclear why the robot chose those specific drinks.

Since the robots of the everyday robots project were able to execute actions based on natural language instructions, I decided to try it out myself. For this, I used chatGPT version 3.5 from openAI in the two examples of Listing 2.1 and 2.2 with the goal of retrieving action parameters for a task variation of cutting.

Listing 2.1: Prompt in chatGPT asking how to slice an orange

```
User: For cutting a slice of bread, I
   first pick up a cutting tool by approaching it with my arm,
    grasping it and lifting it.
   Then I cut the bread by approaching it, lowering the cutting
    tool and lifting the cutting tool.
   Finally I place the cutting tool by approaching a placing
    position, releasing the cutting tool and lifting my arm.
   How do I slice an orange?

chatGPT: Slicing an orange involves a similar step-by-step
    process as you described for cutting a slice of bread. Here's
    how you can slice an orange:
 1. Gather Supplies: You'll need a sharp knife and a cutting
    board. Make sure both are clean and in good condition.
 2. Wash the Orange: Rinse the orange thoroughly under cold water
     to remove any dirt or residue from the skin.
 3. Prepare the Work Area: Place the cutting board on a stable
    surface. Ensure it's not wobbly and won't move during the
    cutting process.
 4. Hold the Orange: Hold the orange with one hand. Position it
    on the cutting board so that the stem end and blossom end are
    facing up and down respectively. This will help stabilize the
    orange.
 5. Pick Up the Knife: Approach the knife on the counter with
    your arm, grasp the handle firmly, and lift it.
 6. Cut the Top and Bottom: With the knife in hand, cut a thin
```

```
   slice off the stem end and the blossom end of the orange. This
    will create flat surfaces for stability.
14 7. Place the Cutting Tool: Approach a designated placing
   position for the knife, release the cutting tool, and lift
   your arm.
15 8. Slice the Orange: Hold the orange firmly on one of the flat
   ends you just created. Approach the orange with the knife,
   lowering the cutting tool, and then lift the cutting tool.
   Make a vertical cut from the top to the bottom of the orange,
   following the curvature of the fruit.
16 9. Repeat for Other Slices: Turn the orange slightly and repeat
   the slicing process to create more orange slices. You can
   adjust the thickness of the slices according to your
   preference.
17 10. Place the Cutting Tool: Approach the designated placing
   position for the knife again, release the cutting tool, and
   lift your arm.
18 11. Serve or Store: Your orange slices are now ready to be served
    or used in various dishes. If you're not using them
   immediately, you can store them in an airtight container in
   the refrigerator.
19 Just like with cutting tools and other kitchen tasks, be careful when using a
   knife to slice the orange.  Make sure to keep your fingers and hands away
   from the blade to avoid any accidents.  If you're not comfortable using a
   knife, consider seeking assistance from someone experienced in kitchen
   activities.
```

In Listing 2.1 I prompted chatGPT, detailing how a robot would perform a cutting action on a bread by breaking down cutting into the tasks of picking up, cutting and placing, and further breaking those tasks down into body movements with the parameters that should be inferred. I then asked how an orange should be sliced. The idea behind this specific question was that the answer would be similar to the input. Surprisingly, the response was very different to my expectation and not only mentioned safety measures but also gave a very detailed task description of how to slice an orange including preparation and postprocessing information, beginning with gathering the needed supplies in line 7, over washing the orange and preparing the work area in line 8 and 9 to storing left overs in line 17. Some of the task descriptions do look similar to the given input. Step

5 in line 11 pretty much repeats the input of a picking up task to approach a knife on the counter with an arm, grasp the handle firmly, and lift it. If we look over the fact that apparently the only storage location of knives is the counter top, the command could actually be used for robotic task execution. This most probably is not true for step 6 in line 12. Here, the stem and blossom of the orange should be cut. What is more, the user is expected to know what the stem end and blossom end of the orange are, and how to cut them off. The next step in line 7 suggests to place the knife. Here, the motions could be translated to robotic action plans but the cutting action is not finished so the knife should actually not be placed yet. After placing the knife in step 7, chatGPT suggests to then slice the orange and place the cutting tool again. Overall, the request resulted in a very detailed description. One aspect to note is that the orange was not peeled before slicing.

Since the question in Listing 2.1 resulted in such specific task steps, I decided to ask a more general question with a task variation on how to dice a cucumber as a robot, as shown in Listing 2.2.

Listing 2.2: Prompt in chatGPT asking how to slice an orange

```
User: As a robot, how do I dice a cucumber?

chatGPT: As a robot, dicing a cucumber involves a precise and
    controlled process. Here's how you can do it:
Note: Please ensure that you are programmed to handle objects safely and
    accurately. If you're unsure about your capabilities or safety measures,
    it's best to get assistance from a human operator.
 1. Gather Supplies: Ensure you have a sharp knife, a cutting
    board, and a cucumber.
 2. Prepare the Work Area: Place the cutting board on a stable
    surface where you can work without any obstructions.
 3. Pick Up the Knife: If you have an articulated arm, approach
    the knife on a secure surface, grasp the handle securely, and
    lift it.
 4. Prepare the Cucumber: Hold the cucumber firmly with your
    gripping mechanism. Position it on the cutting board so that
    it's stable and won't roll.
 5. Make the First Cut: Approach the cucumber with the knife,
```

```
    lowering the blade to make a straight cut off one end. This
    creates a flat base for stability.
10  6. Cut into Sections: Approach the cucumber with the knife again
    , lower the blade, and slice the cucumber into sections of the
     desired length. Each section should be roughly the length of
    the diced pieces you want.
11  7. Place the Cutting Tool: Approach a designated placing
    position for the knife, release the cutting tool, and lift
    your arm or gripping mechanism.
12  8. Slice into Sticks: Take one of the cucumber sections and
    approach it with the knife. Make vertical cuts along the
    length of the cucumber, creating thin, long sticks. These
    sticks will form the basis for your diced cucumber.
13  9. Place the Cutting Tool: Approach the designated placing
    position for the knife, release the cutting tool, and lift
    your arm or gripping mechanism.
14 10. Dice the Cucumber Sticks: Take a bunch of cucumber sticks and
     approach them with the knife. Make horizontal cuts across the
     sticks to create evenly sized cubes. The size of the cubes
    can vary based on your desired outcome.
15 11. Place the Cutting Tool: Approach the designated placing
    position for the knife, release the cutting tool, and lift
    your arm or gripping mechanism.
16 12. Serve or Store: Your diced cucumber is now ready to be used
    in salads, dishes, or as a garnish. If not used immediately,
    ensure proper storage to maintain freshness.
17 Remember, safety is of utmost importance. Always ensure that your
    programming, sensors, and mechanics are capable of handling the cutting
    process without causing harm to yourself or others. If you're unsure
    about your capabilities, it's best to get human assistance.
```

Similar to List. 2.1, in List. 2.2 the reply consists of a long list of tasks and safety information. After a similar description of how to prepare the work area in steps 1 to 4 (lines 4-7) as in the previous reply on how to slice an orange, the actual cutting action starts in step 5 and 6, line 8 and 9. This slicing description resembles the input parameters in List. 2.1 and therefore could be understood by a robotic agent. In the following steps however, the knife is being placed three times, in step 7, 9 and 11 and never picked up again to continue cutting, although steps 8 and 10 describe to further slice and dice the cucumber.

Considering the three examples of LLMs, one can conclude that LLMs can be used to teach robots how to execute task variations. However, this heavily depends on chatGPT being able to access all relevant knowledge about task variations and objects to cut, which we can not influence, and requires lots of engineering effort in correctly prompting chatGPT in addition to the translation of results into body movements. If we also consider the implications for task execution especially in regards to human safety when an instruction is misinterpreted, LLMs do not seem to offer an appropriate solution for robotic task execution.

Chapter 3

Methodology for Creating Actionable Knowledge Graphs

I propose a five step methodology for creating actionable knowledge graphs that can be accessed by digital agents for user assistance as well as by robotic agents for action execution. A precondition that has to be met before starting the creation process is to define the domain of interest and the competency questions the knowledge graph should be able to answer. Competency questions should be as specific as possible and belong to one of the following two categories:

1. **Selection questions** that check whether there exist entities in the knowledge graph that satisfy a restriction such as *"Are there any products in this shelf that have ingredients which might cause contact dermatitis?"*.

2. **Binary questions** that can be answered with yes or no as in *"Is this product vegan?"*.

In (Ren et al., 2014), in addition to these two categories **Counting questions** are proposed. However, for most agent applications that this work aims at, numeric results from counting questions can be disregarded.

After defining competency questions, one can focus on one question at a time and incrementally work through the methodology to create an actionable knowledge graph that is able to answer the questions at hand. The methodology can

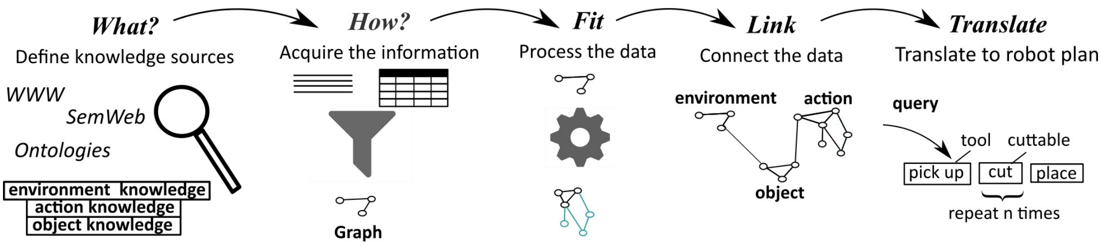

Figure 3.1: The methodology proposed in this work for creating actionable knowledge graphs.

shortly be described by

What? - meaning that first the data to be acquired needs to be defined
How? - to then find the right way to acquire the data
Fit - fit it to standards and process it so that its semantics can be queried
Link - link it to other acquired and existing data and finally
Translate - translate actions to body movements of a robot.

In the end, the created knowledge graph should not only answer the competency questions but also enable agents to access it to translate the contained knowledge into body movements or relate it to the current environment. The steps of the methodology are depicted in Figure 3.1. While steps 1 to 4 are required for creating actionable knowledge graphs for any agent, step 5, the translation of action information to body movements of a robot, is only needed in robotic applications for action execution.

Another aspect to keep in mind when developing a knowledge graph (or ontology), are the FAIR guiding principles for scientific data management and stewardship (Wilkinson et al., 2016). They were proposed in 2016 in order to improve the reusability of data holdings and introduce four principles: Findability, Accessibility, Interoperability, and Reusability:

- Findability: Only if knowledge sources are available to external people, they will inspect and use or even extend the contained knowledge. For this, developed knowledge graphs should be openly accessible, preferably on a website with additional information about the graph. Since website Uniform Resource Locators (URLs) or knowledge graph versions change, one can use persistent URLs[1] that redirect the user to the current storage location/ URL and do not change even if the landing page or source file changes, making it easier for users to find and access the knowledge.

- Accessibility: To help users to inspect, query and reuse the created knowledge graph, it should not only be findable on a website, but also easily accessible. Thus, the knowledge graph storage location should be given, preferably providing a standard API and a tutorial on how to query it. Example queries can be saved and made available as well. One tool that can be used for this is grlc (Meroño-Peñuela and Hoekstra, 2016), where an API can be created for a github query folder. Additionally, Sparklis (Ferré, 2017) can be used to provide an interface for users that are not familiar with SPARQL Protocol And RDF Query Language (SPARQL) or API access to inspect the knowledge graph content.

- Interoperability: When knowledge is findable and easily accessible on the Web, users might want to reuse it in their applications. Interoperability has many benefits and can be increased by reusing existing vocabularies, top-level and domain ontologies.

- Reusability: To further ease reusability of knowledge, one should also focus on documenting the knowledge graph, providing different language labels, comments, links and annotation properties as well as usage instructions. A clearly stated licence that clarifies the terms of use should also be available.

The following sections will explain each step of the proposed methodology in detail. The usability of the methodology for the example scenarios is explained in the next chapters.

[1] Link to the persistent url (purl) administration page: `https://purl.prod.archive.org/`

3.1 Define Necessary Knowledge Sources

Knowledge graphs can be created manually, by accessing structured external (Web) sources and by accessing unstructured external (Web) sources (Heist et al., 2020). Since manual creation of knowledge graphs comes at high cost and lots of information and tools for information retrieval already are available on the Web, one should search for existing knowledge sources that best fit the domain of interest. This includes structured as well as unstructured sources that offer information about 1) general object knowledge, 2) environment structure and 3) action execution. In general, the resulting knowledge graph should consist of at least one taxonomy describing the domain of interest as well as at least one ontology providing additional object information.

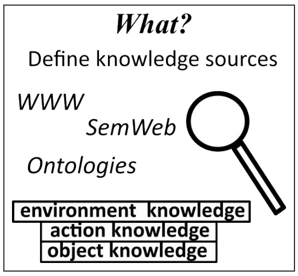

Figure 3.2: Step 1 of the methodology.

> Together with the created competency questions this step can be used to tackle challenge 1) to define the needed knowledge and object properties an agent needs for action execution.

3.1.1 General Object Knowledge

The Web as well as the Semantic Web offer vast amounts of object knowledge, making it hard for users to find sources that best fit their domain of interest. However, there are ways that can support users in finding the right sources. For structured external sources, users can

- search for published ontologies in ontology catalog websites like the Linked Open Vocabularies for Internet of Things (LOV4IoT) [2] or search websites offering open datasets like Google [3], TriplyDB [4] or the LOD cloud [5].

[2] The linked open vocabularies website can be found at `http://lov4iot.appspot.com/`
[3] The Google datasets are available at `https://datasetsearch.research.google.com/`
[4] The TriplyDB datasets are available at `https://triplydb.com/`
[5] The LOD cloud is available at `https://lod-cloud.net`

- use wikidata with its connection to many linked sources as a starting point to investigate the sources offering information about the domain of interest. If one searches for the verb "cutting" in wikidata, for example, in the result list there are many indentifiers from external sources linking to KBPedia, Freebase and WordNet, for example. The search for "apple" results in links to Open Food Facts, USDA, FoodOn or GS1, all offering not only a taxonomy but also different information about the domain.

For unstructured sources, i.e. websites providing information in different formats and in natural language text, it is a good idea to first check the websites' robots file[6] whether automated data retrieval is allowed. In addition to accessibility, trustworthiness of the acquired data should be considered in order for the knowledge graph to be as reliable as possible. Gil and Artz (Gil and Artz, 2007) introduce some factors that influence content trust of websites. For example, website creator expertise and authority, recency and update frequency of the presented information as well as number of websites offering similar information influence content trust and should be considered. Furthermore, many websites with consumer focus already offer standard interfaces to access data.

3.1.2 Environment Structure

Depending on the domain of interest, different types of environment information can be integrated in a knowledge graph. Highly structured environments like retail stores facilitate automated generation of environment information, which is due to

- **the objects available in the environment**. Products often hold a barcode as identifier that can easily and reliably be perceived with cameras. What is more, products usually are stored in shelves or cabinets facing the user, thus increasing the chance of successful image recognition.

- **little environment variation**. In a retail store, all shelves and spaces between shelves are standardised and look similar. Even lighting conditions are standardised for an optimal article presentation.

[6] A robots file is the /robots.txt file located at the root of a website with instructions for web robots on how to crawl the website and which information can be extracted.

- **object presentation**. Very good lighting conditions not only make it easier for customers to find products but also make object perception easier.

- **ease of object identification**. Perceived products can be identified by their barcode or product image, which usually also are available in Web stores and consumer websites. Thus, objects perceived in the environment can reliably be linked to object information from the Web.

- **availability of environment information**. Digital agents can already create environment models of retail stores autonomously (Beetz et al., 2022).

In less structured environments like households, reliable object identification is harder and heavily relies on perception techniques. Still, while a robot routing a customer to the correct product location does need reliable environment information, other applications like recipe recommenders can already support users if they include basic environment information like being able to perceive a product. We can distinguish between four forms of environment information that can be integrated in knowledge graphs:

1. **Identification via barcode or QR code** is the easiest form of integration as detailed before. Barcode information can easily be linked to product information available on the Web. Similarly, objects like shelves can be equipped with QR codes that link to shelf information in the object knowledge graph. Dependent on the application, QR codes can then relate to a fixed reference coordinate for robotic applications as well as be used as world anchors that can be detected in AR applications and visualise digital content in relation to the anchor position.

2. **Identification via Image recognition** is a reliable solution for domains with a relatively small set of objects. While linking of product image information to object information from websites still is fairly easy due to the amount of available information on the Web, image recognition does not work reliably in retail environments with lots of similar products standing next to each other or very small products like toothpaste and toothbrush being placed without spacing.

3. **Identification via 3D object recognition** is a very reliable solution for object recognition but comes at the cost of having 3D models available, which usually have to be manually linked to object knowledge.

4. **Identification via employed semantic Digital Twins** is the most reliable solution for localisation of objects in an environment. It has been shown that robotic agents can autonomously create semDTs of environments (Beetz et al., 2022) that can be used to retrieve precise article positions both by robotic as well as digital agents (Kümpel et al., 2023). SemDTs are semantically enhanced environment models that allow for linking the contained environment information to object information in a knowledge graph. However, reliant on the environment structure, it might not be possible to automate the linking.

Depending on the domain of interest, one approach might be more applicable than others. While users will accept to scan a barcode or lift a product for image recognition to see recommended recipes, they most likely will not accept a routing application giving wrong position information.

3.1.3 Action Execution

For action execution, knowledge sources need to comprise rudimentary concepts necessary for the execution of general manipulation tasks, covering environmental aspects like agents, objects and situations as well as temporal and physical aspects like trajectories or forces. Situations define tasks already available and executable by the robot, thus serving as a foundation for the hierarchical representation of more complex tasks. Due to the availability of approaches in the domain of cognitive robotics, knowledge about robotic movement or manipulation concepts is collected in top-level ontologies like the Socio-physical Model of Activities (SOMA) (Beßler et al., 2022) or the Event-Model-F (Scherp et al., 2012).

In addition to general knowledge about actions and objects, knowledge about their relation is needed. This knowledge consists of object features and properties which are relevant for the current manipulation action. Due to the speciality of this knowledge, the sources need to be specific for the execution domain since general sources only offer a broad coverage that can not be translated into motions. In particular, sources should deliver general information about existing objects and their usage, represented through the concepts *disposition* and *affordance*. In general, a *disposition* describes the property of an object, thereby enabling an agent to perform a certain task (Turvey, 1992a) as in “a knife can be used for cutting”, whereas an *affordance* describes what an object or the environment offers an agent (Bornstein and Gibson, 1980) as in “an apple affords to be cut”. In works like (Beßler et al., 2020a, 2022), both concepts are set in relation by stating that dispositions allow objects to participate in events realising affordances. An application example of how the SOMA top-level ontology can be used is shown in (Pomarlan and Porzel, 2022). However, since the available structured resources are top-level ontologies, they do not cover object dispositions very well, which is why a manual and semi-automated process of collecting data for a knowledge graph is necessary.

Knowledge sources employed for linking action to object information should handle dispositions and affordances as well as describing any other properties or features that are relevant for grounding basic manipulation actions like grasping, holding or transporting to support basic planning. This knowledge supports the robot in understanding and recognising objects and their purpose during task execution. Due to the lack of structured sources available, action knowledge usually has to be created manually or extracted from unstructured sources.

3.2 Acquire the Needed Information

When acquiring Web knowledge, the goal is to transform the extracted data into facts stored in modular ontologies. For example, product information like ingredients should be stored in one ontology while product information like brands should be stored in a separate ontology. Such a modular approach supports reuse of specific parts of the created knowledge graph for different purposes in the future. Knowledge acquisition methods can be categorised based on the knowledge that is to be accessed: 1) unstruc-

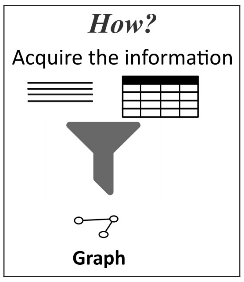

Figure 3.3: Step 2 of the methodology.

tured external (Web) sources, usually textual information,
2) semi-structured external (Web) sources such as tables on a website or relational databases and 3) structured external (Web) sources like ontologies. Available methods are further detailed in the following.

> This step can be used to tackle challenge 2) to acquire the previously defined knowledge and the first half of challenge 4) to acquire Web knowledge in different formats from distributed sources.

3.2.1 Extracting Information from Unstructured Sources

In contrast to Semantic Web sources, websites created for human access often provide textual information. Since machines cannot reason over textual information, it usually is classified as unstructured information. *Information Extraction* describes the process of converting text to structural information like entities and their relations (Etzioni et al., 2008a). To acquire textual information from websites, **Python** and the Beautiful Soup library can be used. The extracted information can then easily be transformed to ontologies with the Owlready library (Lamy, 2017). The research field of Information Extraction has also led to many different tools that can be used for acquiring data. Example tools that can be used for different purposes are listed below.

- For **Taxonomy Creation**, one can use a pattern-based approach for taxonomy induction (Davulcu et al., 2003a). Another approach is to access online stores for taxonomy creation since they are designed in a taxonomy-directed structure that can be used for Information Extraction as shown in (Davulcu et al., 2003b; Tenorth et al., 2011).

- For **Learning Dictionaries** from text, multi-level bootstrapping can be applied (Riloff et al., 1999). Here, only some seed words need to be given as input and the algorithm will learn dictionaries of similar sentences. This is especially useful in text that offers similar information in large varieties, such as recipe preparation instructions where one might use it to learn a dictionary of all pouring actions and their destinations, for example. For other text covering different domains, lots of unwanted relations might be learned that require postprocessing.

- If one wants to **retrieve all relations in a text**, Open IE is the right tool to use (Etzioni et al., 2008b). Here, the KnowItAll sytsem (Etzioni et al., 2005) was the first unsupervised system proposed that only needs a small set of domain-independent extraction patterns to then automatically label and extract relations. Building upon this, the TextRunner system (Yates et al., 2007) was built to be more scalable and general.

- To **retrieve only certain relations** (which are predefined) in a text, tools like Snowball (Agichtein and Gravano, 2000) or DIPRE (Brin, 1998) can be used as explained in more detail in (Nasar et al., 2021).

- To deeper dive into natural language processing and extract **word conjunctions** or **retrieve hyponyms** in texts, the Stanford CoreNLP (Manning et al., 2014) toolkit can be used.

The presented tools can be used to tackle challenge 7) to create an action classification that agents can use.

3.2.2 Extracting Information from Semi-structured Sources

Semi-structured sources such as relational databases storing their data in csv, json or xml format do not need much processing since the data already is available in tabular form. However, the tabular data needs to be transformed to a graph structure by defining relations between entities if machines should be able to interpret it. Protègè, the standard tool for ontology editing, offers the cellfie plugin (Hardi, 2018) that can also be used to convert semi-structural sources to ontologies by defining rules, ranges of data and automatic mapping to relations. Additionally, tabular data can be mapped to RDF format using the R2RML standard (Consortium et al., 2012) and the RML mapping language (Dimou et al., 2014) through statements defining the mapping relation.

Structured sources like ontologies or knowledge graphs published on the Web do not have to be recreated manually. One can query the source to retrieve a subgraph of it using SPARQL CONSTRUCT queries (Kostylev et al., 2015). However, by creating subsets of existing knowledge graphs, updates of the retrieved sources will not be displayed in the extracted source. To address this problem, one can either reconstruct the source from time to time or use a framework for keeping the acquired source up to date using extract load transform (ETL) as proposed in (Bansal, 2014).

3.3 Process the Acquired Data

Web sourced knowledge graphs essentially contain logical statements about entities and their relationships that are acquired from Web sources as described in the previous section. However, Knowledge acquired from the Web can be of poor quality. To solve this, different techniques for knowledge processing exist that focus on "cleaning" of the acquired data to make sure that the statements are correct (i.e. they do not contain contradictions or false information). With this, reasoning about the contained information will also produce conclusions that we as humans would regard as true.

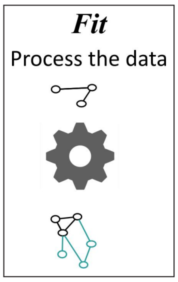

Figure 3.4: Step 3 of the methodology.

Processing of the acquired data thus aims at increasing data quality as well as semantically enhancing the data so that each created source can be used to answer questions about the contained data. To further enrich the acquired data, manual ontology editing is needed. This is especially important for environment and action information, as will be discussed in the following.

> This step can be used to tackle the second half of challenge 4) to find ways to standardise Web knowledge in different formats from distributed sources.

3.3.1 Cleaning, Standardising and Enriching the Acquired Web Data

In order to clean acquired data, especially if it was extracted from text, natural language processing tools can be applied:

- For **Coreference Resolution**, the identification of ambiguous noun phrases as in "Cut *it* into slices", where the object to be cut is unclear and should reference to an explicit object of the ontology, many different tools have been developed. These tools like Sucre (Kobdani and Schütze, 2010) or BART (Versley et al., 2008) are based on the learning approach proposed in (Soon et al., 2001).
- To filter for certain parts of speech like verbs or adjectives, **Part-of-Speech Tagging**(Voutilainen, 2003) can be used. The tools usually rely on hidden Markov models such as in (Banko and Moore, 2004) and can be applied to different languages (e.g. German in (Schmid, 1999)).

The following standardisation of data further helps in increasing the quality of the data and making linking of knowledge chunks easier:

- Conversion of all textual information to all lower case format.
- Strict splitting of textual and numeric information or removal of numeric information.
- Removal of special characters.

Additionally, simple automated data categorisation can increase the encoded semantic knowledge. For example, string matching techniques (Charras and Lecroq, 2004) can be used to classify product ingredients like sunflower or corn oil as oil, which then enables users to search for or filter out all products that contain oil.

3.3.2 Enrichment of Environment Information

Environment knowledge is about data items such as images, sound recordings or coordinate values. These can be attached to logical entities in a knowledge graph but the data itself cannot be used by a reasoner to reach conclusions. Therefore, the focus here should be to accurately model the real environment and link the model to data items in the knowledge graph.

In Sec. 3.1.2 different forms of environments information that can be integrated in knowledge graphs have been presented. If only barcodes of products are used for object identification, the acquired data does not need to be enriched for linking it to object knowledge. However, if any other object identification shall be used in applications, the acquired environment data has to be extended as detailed in the following:

1. If **QR codes** are used to easily identify landmarks in the environment like shelves in a store or a cupboard in a kitchen, the QR codes have to semantically link to the object of interest, which can be integrated by adding relations such as `<QR_code_1 has_QR_code 123>, <QR_code_1 has_depiction image_A>`, which then links image information perceivable by a camera to a code that can ultimately be mapped to object information (which will be discussed in the next section).

2. When using **image recognition** for object identification, the only way of improving success rates for recognition are to provide as many images in different orientations, viewing angles and lighting conditions as possible for training. Manual integration of additional images into the knowledge graph can easily be done by adding more relations to a given object as in `<object_2 has_depiction image_B, has_depiction image_C>`.
   Since the object information most likely was extracted from websites in combination with images, the object information already links to at least one image. Image recognition databases like Vuforia [7] additionally allow upload of multiple images for a single object. The database already is integrated in the Unity development platform, can easily be used for Augmented Reality (AR) applications and allows for assigning the same object identifier for multiple images.

3. **3D object** models can similarly be encoded as environment information as QR codes in the example above by providing a link to the file or a file name for the 3D model: `<object_Z path_to_model link_B>`.

[7] Vuforia image recognition: https://library.vuforia.com/objects/image-targets

4. **SemDTs** already provide semantically enhanced environment information for robotic agents and only need further processing if digital agents are to use the contained information. The positions in the semDT use a fixed reference frame with a given origin, usually set to a randomly chosen corner of the room. All objects perceived by the robot then are set in relation to this origin as in <barcode_2 has_x_position 0.12, has_y_position 1.5>, <QR_code_1 has_x_position 2.2, has_y_position 0.05>. In contrast to robots, Augmented Reality devices use the varying device position when starting an application as origin and display digital content relative to the device position. To solve this discrepancy, all product locations have to be encoded in relation to shelf, cupboard or other object positions that can be perceived in the environment as in <product_1 stored_on shelf_12, stored_in shelf_floor_8>, <shelf_12 has_QR_code QR_code_1>.
Simple relations such as `stored_on` or `stored_in` can easily be created in structured environments like retail stores. In household environments without barcodes or QR codes, detailed inspection of semDT data and its accompanying 3D models is needed to classify an object as product, cupboard or table and then relate it to other object positions.

> The translation of object positions with a set origin to positions with a changing origin can be used to tackle challenge 5) to encode environment information in such a way that different agents can use it in applications.

3.3.3 *Enrichment of Action Information*

Action knowledge is about how to set up an agent to behave in an appropriate way. The focus is on correctly performing given tasks, parameterising them to successfully perform task variations, and avoid unwanted effects. Thus, the important factor here is how to make the knowledge actionable, i.e. we do not only form subgraphs or link data items, but we need actual programs running on robots to make the robot successfully perform a task.

As only top-level action information can be acquired from the Web, the acquired data has to be enriched to model and reason about relevant object features and properties that are important for action execution. For example, the SOMA (Beßler et al., 2022) ontology contains top-level descriptions of how events consist of actions that are performed by agents and include objects that take on roles during action execution. Actions further can execute a number of tasks like navigating, or pouring. They also come with basic descriptions of their dispositions and affordances, such as `<Pouring is_task_afforded_by Pourable>`. Thus, it can be used for a broad range of different tasks already. However, when it comes to more specific tasks like distinguishing between cutting, slicing and dicing, task information needs to be enhanced so that a robot can infer the needed knowledge. Additionally, the tasks still need a formal definition and a connection to the existing objects. This includes knowledge influencing task execution, like e.g. the existence of a core or peel for a fruit but also the creation of task variations as explained in (Kümpel et al., 2024). In particular, parameters that can influence motion execution of the robot have to be integrated, such as

- **the tool being used** for action execution. Some objects might only afford certain tools to be used or have different action parameters based on the tool being used (e.g. imagine using a knife instead of scissors for cutting).

- information about the **object affording the action** to be executed. For example, an orange does not have a core that needs to be removed and therefore should not afford a *core_removal* action.

- **changes in object** appearance, shape or amount have to be modelled if they **influence action execution**. If the goal is to create dices of a cucumber, for example, the base object might first be cut into two objects: a piece and a slice, where the slice will need to be cut again to create dices.

3.4 Link Distributed Knowledge Chunks

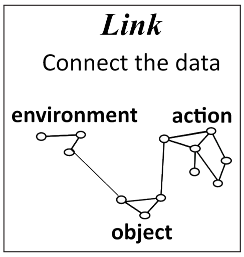

Figure 3.5: Step 4 of the methodology.

The previous steps have defined necessary data sources to then extract information and process the acquired data, leading to knowledge chunks with modular domain knowledge that can be queried. Now the knowledge chunks have to be linked to connect the contained information, allow for more advanced reasoning and ultimately link object to environment and action information to make the knowledge graph actionable. For this, one first has to align the knowledge chunks with top-level ontologies to then connect object, environment and action information following the Linked Data standards set by Bizer, Heath et al. (Bizer et al., 2007), which are:

- Use the RDF data model to publish structured data on the web.
- Use RDF links to interlink data from different data sources. Linkage of the data generates a graph, in which the nodes are Uniform Resource Identifiers (URIs) of the represented entities and edges resemble the relation between two nodes.
- Re-use as many existing URIs as possible and unique identifiers like article numbers as part of the URI.
- Re-use existing terms/vocabularies if possible.
- Use the owl:sameAs property to interlink two data sources.

> This step can be used to tackle challenge 3) to link the acquired Web knowledge in such a way that an agent can query it.

3.4.1 Ontology Alignment

Ontology alignment can be achieved by using an upper ontology like SUMO (Niles and Pease, 2001) or DOLCE+DnS Ultralite (DUL) (Presutti and Gangemi, 2016) and reusing the available relations as well as existing vocabularies. However, one major challenge in ontology alignment of ontologies with different foci is the use of diverse upper ontologies. For example, while the action ontology SOMA is based on the DUL ontology with its descriptions of relations of actions, objects and agents at a given time and space, the Open Biological and Biomedical Ontology Foundry (OBO Foundry) ontologies are based on the Basic Formal Ontology (bfo) (Otte et al., 2022) defining occurrents and continuants over a period of time (Mascardi et al., 2007). Thus, external concepts like processes that are defined as occurents in bfo need to be aligned with the definitions of tasks, actions and processes as events from DUL.

3.4.2 Interlinking Object Information

Since knowledge graphs usually consist of data from many different sources, chances are high that the used sources use different terms to identify similar objects. In simple cases like linking an “apple” from source A to “apples” in source B, string matching techniques can be used. Mapping words to vectors, i.e. learning word embeddings where similar vectors indicate semantically similar words (Bhoir et al., 2017) has proven successful. For word embeddings, Word2Vec can be used to link taxonomies but also to expand the contained knowledge by learning sibling classes (Wohlgenannt and Minic, 2016). Similarly, Node2Vec can be used for graph data with good results. If we want to link quite different ontologies such as a taxonomy to a brand ontology, the domain of interest changes from linking two ”shampoo” classes to linking one ”shampoo” class to ”Elvital Dream Length Super Strengthening Shampoo”, for example. Here, the distinct product name consists of multiple words. One solution for this is to create a class hierarchy. Another solution is to handle both classes as branches that shall be linked, in which case Doc2Vec has been shown to efficiently handle multiple words with context (Lau and Baldwin, 2016).

The links can be created either by using the `owl:sameAs` property, stating that both apples of source A and B are the same, as proposed by Bizer, Heath et al. (Bizer et al., 2007), or using the `oboInOwl:hasDbXref` annotation property to link a logical entity to a data source by stating a cross reference annotation link to an external source, as proposed by the gene ontology (GO) consortium (Consortium, 2019).

Additional linking of ontology terms to wikidata highly increases the knowledge that can be accessed. The automatic linking of ontology terms to wikipedia (or wikidata) is called entity linking in (Lin et al., 2012).

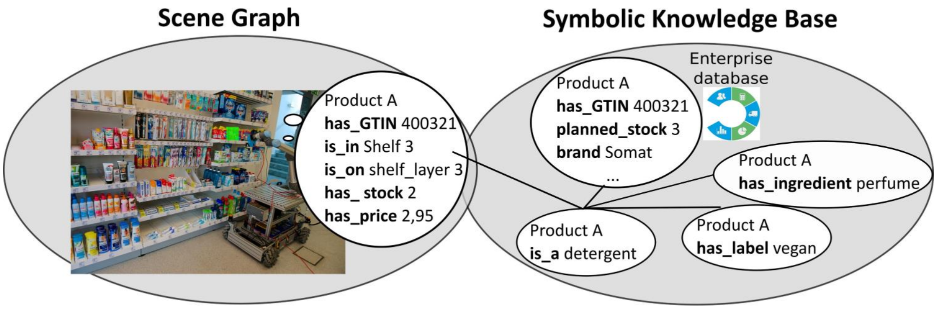

Figure 3.6: Linking scene graph environment information to symbolic object information (published in (Kümpel et al., 2021)).

3.4.3 Linking Object to Environment Information

The effort needed for integrating environment information into an object knowledge graph highly depends on the environment information used. If barcodes or other unique object identifiers are used, environment and object information can easily be linked as explained in prior work (Kümpel et al., 2021) and depicted in Figure 3.6. Here, an agent querying the knowledge graph can use the recognised article number/ barcode number to query for product information. For robots acting in household environments, a perception framework for task aware robot manipulation has been proposed in (Balint-Benczedi et al., 2016). The framework is able to adapt robot perception in accordance with the task at hand. If the perception framework is used to create scene graphs of environments and perceived objects are classified as "apple" or "milk", perception results can easily be linked to object information using the techniques described in the previous Sec. 3.4.2.

When using digital agents, scene graphs seldom are available. Integrating environment information like landmarks with QR codes or 3D models with object information therefore usually requires manual connection of environment object identifier and object as in `<shelf_12 identified_by QR_code_1>`.

This step can be used to tackle challenge 6) to link perceived objects in the environment to objects of a knowledge graph.

3.4.4 Linking Object to Action Information

Once action specific object knowledge like object dispositions and affordances relevant for action execution have been defined, it needs to be connected and integrated into the knowledge graph. To do so, the conditions that are necessary for the different tasks to be performed need to be described using the previously identified object features. For example, if a fruit the robot wants to cut has a peel, the task to remove it needs to be executed before additional cutting can be achieved. Thus, for an orange one would add the dispositions of peelability and cuttability as well as the property that it has a peel. For the peel one would then add the constraint that it must be avoided. Additionally, one should add that the peelability affords the task of Peeling by hand or peeler. This can be formalised as shown in Figure 3.7. The orange in this example then needs to link to an orange of the object knowledge graph. Again, this can be achieved using the previously presented methods (see Sec. 3.4.2)

$$
\begin{aligned}
&Orange \sqsubseteq Fruit \wedge \exists hasDisposition.Cuttability \wedge \exists hasDisposition.Peelability \wedge \\
&\qquad \exists hasPart.[Peel \sqcap \exists hasEdibility.MustBeAvoided] \\
&Peel \sqsubseteq FruitPart \\
&Peelability \sqsubseteq \exists affordsTask.Peeling \sqcap \\
&\qquad\qquad \exists(affordsTrigger.Peeler \vee affordsTrigger.Hand) \\
&Cuttability \sqsubseteq \exists affordsTask.Cutting \sqcap \exists affordsTrigger.Knife
\end{aligned}
$$

Figure 3.7: Representation of task-specific object knowledge that is important for executing cutting actions on an orange.

The enrichment of action information to model object knowledge that influences action execution can be used to tackle challenge 9) to relate object information to task information that a robot can reason about for action execution.

3.5 Translate Knowledge to Robot Plan

If robot agents are to use the knowledge graph for action execution, it needs to be connected to the cognitive architecture employed by the robot for task planning. This allows the robot to query the created ontology to gather important action parameters for execution of manipulation tasks in the current environment.

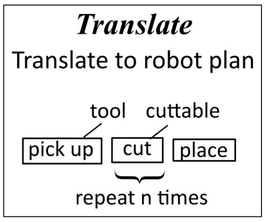

Figure 3.8: Step 5 of the methodology.

Figure 1.2 shows the general cutting action and how it is broken down into tasks and body movements of the robot. Here, the general cutting action already was edited so that it can be parameterised in regards to a cutting position and the object to cut. If these parameters are not set, the robot would try out a position it retrieves from its search space, which was set to possible cutting locations near the end of the object to cut, for example. Thus, the parameterisation is important since it changes the way how the cutting task is executed by a robot. If the robot is to execute more task variations of cutting, the action plan might need to be changed to request even more parameters while the knowledge graph should integrate all information needed for a robot to infer the parameters it needs for successful action execution. For example, if the robot was taught to cut an object as in creating slices of bread, the knowledge graph should contain the information that for other cutting tasks like quartering an apple, a different position in the middle of an object is needed. Since this work is based on KnowRob (Beetz et al., 2018b), which can be accessed by the cognitive architecture CRAM (Beetz et al., 2010) and uses Prolog (Bramer, 2005) for inferring parameters, queries must be available in Prolog in order for a robot to execute them.

For different actions, different action parameters are needed. A pouring action will depend on the viscosity of the liquid to pour and the container size that influence the pouring angle, for example.

> The translation of knowledge to robot body movements can be used to tackle challenge 8) to determine parameters that influence action execution and provide the robot with ways to use the parameters.

3.6 Discussion and Related Work

This chapter has introduced a five-step methodology for creating actionable knowledge graphs that can be used as a guideline for creating knowledge graphs that are grounded in the environment or even in robotic actions. Due to the many different application domains of knowledge graphs and the range of agents accessing knowledge graphs, the methodology describes a general guideline for different domains and agents. Therefore, when using the methodology for creation of actionable knowledge graphs, some steps of the methodology will be more important than others, while some steps might even be left out or need to be extended. The next chapters show how different steps need more attention than others in two example scenarios in different domains and applications.

Although the need for a unified methodology for creating ontologies has already been remarked in 1996 (Uschold, 1996) and there has been lots of research on creation of knowledge graphs, no one-fit-all construction approach exists. There has also been approaches to include spatial and temporal knowledge in the YAGO2 knowledge graph (Hoffart et al., 2013), which resulted in object knowledge that semantically models time spans and regions. However, no methodology for creation of actionable knowledge graphs that link to perceived environments (instead of GPS coordinates) and action information has been proposed. In general, related work in creation of knowledge graphs suggests to first acquire the needed data to then construct a data and concept layer and finally process the acquired information to generate the knowledge graph (e.g. (Ryen et al., 2022; Wang and Yang, 2019)), which is similar to the approach of steps 1-3 in this work without differentiating between object, action and environment knowledge. Many existing works focus on presenting tools for knowledge engineering, but automated approaches or a detailed description of a methodology for knowledge graph creation are still lacking. There has been little research on linking knowledge graphs to environment information or action information. In (Pomarlan and Porzel, 2022), objects are linked to affordances and dispositions in broader contexts than in this work. The robot knowledge graph (R-KG) (Hao et al., 2020) links environment objects and their attributes to support the creation of semantic maps. Hence, the created knowledge graph can be categorised as a semantically enhanced environment model like the semantic Digital Twin used in this work and is different to the actionable knowledge graphs created in this work.

The presented methodology can be applied for creation of different actionable knowledge graphs as will be shown in the following two chapters.

Chapter 4

An Actionable Product Knowledge Graph for Omni-channel User Assistance Applications

This chapter describes how the methodology introduced in the previous chapter can be applied in a user assistance scenario. The scenario aims at supporting users in daily environments like retail stores or households and has a focus on omni-channel applications that are independent of the used device (Taylor et al., 2019).

Although we have seen many interesting consumer applications in the offline retail domain like mobile (Waltner et al., 2015; Davis et al., 2006) or robot shopping assistance (Gross et al., 2008; Chen et al., 2017), robotic depalletising and shelf replenishment (Caccavale et al., 2020; Costanzo et al., 2021) or click-and-collect services and smart shopping carts, they often are too customised, meeting only certain consumer profiles. What is more, the same consumer will demand different product information like its availability and price, ingredients that might cause allergic reactions or harm the environment, recommendations for complementary products as well as convenient return policies, depending on the temporal relation to the actual purchase (before, during or after) (Lockie, 2014) and the users' personal relationship to the device used for information retrieval (Riegger et al., 2022). The same user might want to check product availability on their smartphone before shopping, use a shopping robot to locate the searched product in the store and check store return policies on a website after purchase. In order for applications to turn into omni-channel applications providing a seamless shopping experience that is independent of the used device or environment (Taylor et al., 2019), I suggest that they should: a) share one source of data, b) be modular and generalised so that they can be instantiated for a personalisation that takes into account the temporal relation to purchase and c) include semantically encoded environment information. The methodology proposed in this Thesis can easily be used to provide omni-channel application as has been demonstrated in (Kümpel et al., 2023) and will be described in this chapter.

The next sections will describe in detail how the methodology can be applied for creating a product knowledge graph, the ProductKG (Kümpel and Beetz, 2023) for user assistance in daily scenarios.

Competency Questions The product knowledge graph for user assistance should be able to answer the questions detailed in Table 4.1.

Q1	Where is [product category]?
Q2	Where is [product category] containing [ingredient]?
Q3	Where is [product category] with [any preference]?
Q4	Where is [product category] with [any two preferences]?
Q5	Does this product contain [ingredient]?
Q6	Show me the brand of this product
Q7	Show me labels awarded to this product
Q8	Show hazard information for this product
Q9	What recipes can be prepared with this product?
Q10	Can this product be substituted by other products?
Q11	What are the nutritional values of this product?
Q12	Can the product or ingredients of it cause any disease?
Q13	Can the product or ingredients of it cause any symptom?
Q14	Can the product or ingredients of it ease any symptom?

Table 4.1: Competency Questions the product knowledge graph should be able to answer.

The competency questions Q1-Q8 aim at increasing the shopping experience and assisting customers in a store. While Q1 to Q4 help users in finding product locations with Q1 providing a simple category search, Q2 offering to filter for product ingredients and Q3 and Q4 narrowing down the result set to products that offer another preference besides ingredients (here a preference is a relation that can be filtered for, i.e. brand and labels). Q5 to Q8 can be used to show/ highlight interesting product information like contained ingredients (Q5), its brand (Q6) or labels (Q7) as well as hazard information of the product (Q8). Q9 to Q14 relate the product to different other domains like recipes (Q9), ingredient substitution (Q10), diseases (Q12) and symptoms (Q13 and Q14) as well as nutritional values (Q11). These competency questions will usually be asked before or after shopping.

Since the focus of the product knowledge graph is to create omni-channel applications, both robotic and digital agents should be able to answer the competency questions.

4.1 Define Necessary Knowledge Sources

Before defining specific knowledge chunks that should be integrated into the knowledge graph, the top-level ontology to be used should be specified. What is more, for many domains vocabularies exist and should be reused. Due to the restriction that the knowledge graph should contain action information, I propose to reuse the top-level ontology DUL (Presutti and Gangemi, 2016), as it is reused by the SOMA ontology (Beßler et al., 2020b). For products, the product types ontology[1] can be reused to integrate specific terms that already are linked to wikidata. For shopping related information, the Good Relations web vocabulary for E-Commerce[2] can be reused.

To further define the knowledge sources necessary for creating a product knowledge graph for user assistance, one has to bear in mind the competency questions defined previously in relation to the three different types of knowledge actionable knowledge graphs consist of: object, environment and action knowledge.

[1] The product types ontology is available at http://www.productontology.org/
[2] Good Realtions vocabulary: http://www.heppnetz.de/ontologies/goodrelations/v1

4.1.1 General Object Knowledge

In the Semantic Web, food ontologies have been a research focus for a while, resulting in partially interlinked and publicly available domain ontologies with consumer applications ranging from agricultural over health specific to recipe centered (e.g. (Dooley et al., 2018b; Cantais et al., 2005; Haussmann et al., 2019)). Contrary to this, non-food product information in the Semantic Web has only been mentioned as support for food ontologies so far (Boulos et al., 2015). There has also been research on the creation of product knowledge graphs (e.g. (Lee et al., 2006; Zalmout et al., 2021)), highlighting the benefits of using ontologies or knowledge graphs as databases but lacking publicly available sources.

If we look at the competency questions in Tab. 4.1, we can infer the object knowledge the product knowledge graph should contain:

- Q1: product categorisation. Existing shopping applications try to integrate a product categorisation by remembering previously bought items (Davis et al., 2006) or offering a limited set of product categories that can be chosen from (Thompson et al.). As most knowledge graphs, the product knowledge graph needs a taxonomy. Using a product taxonomy to retrieve information about the product class offers some advantages: there is a broader range of products to be chosen from as well as the possibility to infer parent, sibling and children classes. Hence a digital assistant can find the shelf holding cereal even if the store offers cereal in a broader context like `grain products`. The FoodOn ontology (Dooley et al., 2018b) or the Bremen Ambient Assisted Living (BAALL) ontology (Krieg-Brückner et al., 2021) provide an extensive food taxonomy but lack non-food objects. For product taxonomies, GS1 proposes the Global Product Classification (GPC) standard[3] that could be used as shown in (Allweyer et al., 2021). However, the GPC standard covers products from cars or camping products over clothes and video games to gasoline and medicine and therefore seems too extensive for the given use case. Therefore I chose to access online stores for taxonomy creation. Previous attempts in taxonomy creation from online stores show its applicability but are spamming the used website to call children and

[3]The GPC standard: `https://www.gs1.org/standards/gpc`

sibling websites for Information Extraction (Davulcu et al., 2003b; Tenorth et al., 2011). Since online stores usually offer a sitemap in their robots file[4], I propose to use this sitemap for Information Extraction to avoid spamming of websites and be more time-efficient. Sitemaps of online stores are usually hidden .xml files that consist of a list of Uniform Resource Locators (URLs) of children websites so that crawlers can easily access them.

- Q2: product ingredients. Packaged food usually contains a large amount of different ingredients that often are hard to understand or have ambiguous names. At the same time, ingredient filter for harmful substances or ingredients that might cause allergies are very helpful for making informed decisions. Many online stores offer ingredient information that can be crawled. The classification of ingredients as allergens is available at Drugbank[5] or in the Chemical Entities of Biological Interest (ChEBI)[6] ontology.

- Q6: product brand. Consumers are interested in product brand information and might search for store brands as substitutes for interesting products. They might as well be loyal to a certain brand and want to search for it. To allow for such applications, ProductKG includes a brand ontology, which classifies brands and allows for linking brands to a business entity. The brand ontology reuses the Good Relations vocabulary for E-Commerce[7] and can be constructed by extracting product brand information from product websites.

- Q7: product labels. Labels certify certain quality standards of products, their ingredients, packaging or production processes and play an important role in shopping decision processes. Label criteria can be acquired from the Web and can additionally be categorised into label classes like `vegan` for all labels certifying vegan products such as the `V_label`. Supplementary, the label ontology should be complemented with an image database to be

[4] A robots file is the /robots.txt file located at the root of a website with instructions for web robots on how to crawl the website and which information can be extracted.

[5] Drugbank: https://go.drugbank.com/

[6] Chemical entities of biological interest: https://www.ebi.ac.uk/chebi/

[7] Good Realtions vocabulary: http://www.heppnetz.de/ontologies/goodrelations/v1

used in AR applications. Product label information can be acquired from product websites like Open Food Facts[8] or Codecheck[9].

- Q8: hazard information. Similar to product ingredients or labels, hazardous chemical ingredients influence customer decision processes. Ingredient classification criteria as well as images are available online. For example, the ChEBI ontology or the National Drug File (NDRF) of BioPortal[10] offer information on hazardous product ingredients while hazard images are also available on wikidata.

- Q9: recipes. Many people use either cookbooks or apps and websites to search for recipes and ideas what to cook. The success of Vorwerks' Thermomix or cook boxes such as HelloFresh can be attributed to their simplification of the cooking process as well as their recipe recommendation. A recipe recommender based on ontologies is Food Knowledge Graph (FoodKG) (Haussmann et al., 2019), which links ingredients to the taxonomy of the Food Ontology (FoodOn) but only offers a small set of recipes and ingredients.

- Q10: substitutes. The FoodKG recipe recommender proposes a method to link substitute ingredient information using the Cook's Thesaurus (Haussmann et al., 2019). This approach can be reused with standardised ingredients but I extend the idea by an inclusion of a `purpose` and quantity information. That way a recipe recommender can propose to substitute wheat flour with the same amount of almond or millet flour (amongst others) if the purpose is baking but instead propose buckwheat flour if the purpose is thickening (of a sauce), for example.

- Q11: nutritional values. Nutrition ontologies play an important role in recommender systems as seen in (Espín et al., 2016; Tumnark et al., 2013). The goal is to link products to nutrition properties. The nutrition properties are being used for product-specific nutrition information as in `<Product A has_sugar 1.5 g>`. Unfortunately, such information is not available online.

[8]Open Food Facts: https://world.openfoodfacts.org/
[9]Codecheck website: https://www.codecheck.info/
[10]NDRF at BioPortal: https://bioportal.bioontology.org/ontologies/NDFRT/

Instead, the FoodData central [11] from the U.S. Department of agriculture offers extensive nutritional information for product classes.

- Q12: diseases. Product ingredients can trigger diseases. To model this, ingredient information can be classified according to chemical Websites like ChEBI and then linked to disease information from existing ontologies like the human disease ontology (doid) (Schriml et al., 2012, 2022).

- Q13: symptoms. Diseases trigger symptoms and both symptoms and diseases can sometimes be treated by ingestion of nutrients. An example ontology describing symptoms is the human phenotype ontology (hpo) (Robinson and Mundlos, 2010). There also are websites that offer information on how symptoms can be treated like PatientsLikeMe [12].

4.1.2 Environment structure

In order to reliably answer competency questions Q1-Q4, detailed environment information is needed. Doering et al. (Doering et al., 2015) use a shopping robot to guide customers to a searched product in a home improvement store. For localisation of products, they reuse a database designed for stocktaking of human employees. Unfortunately, they are only able to locate 83% of articles the store holds. Recent research has focused on standardised mapping, in particular of retail environments. The benefits of using standardised maps has been identified in (Davis et al., 2006). In (Brenner and Hummel, 2017) a Digital Twin is successfully used to offer a standardised map with exact real-time article positions. The idea of creating semDTs of retail environments as standardised maps has been proposed in (Kümpel et al., 2021) for shopping assistance or shop-floor assistance for product refilling. It has additionally been shown how semDTs can be used in retail applications to route a customer to a searched product on different devices such as smartphone and robot (Kümpel et al., 2023). In contrast to the approach of Doering et al. (Doering et al., 2015), for example, semDTs are very accurate since they are based on such standardised maps created by robotic agents. Additionally, semDTs enable robots as well as smartphones, smart glasses or other

[11] Food Data central: `https://fdc.nal.usda.gov/`
[12] PatientsLikeMe website: `https://www.patientslikeme.com/`

devices to use current and exact location information. SemDTs are reliable and can be created on a daily basis as the success of Ubica robotics [13] demonstrates. However, a shopping assistant should not be expected to know more than a shopkeeper. For example, a customer should expect to retrieve the usual storage location of a product they are searching for. SemDTs and the shopping assistants introduced in this work should not be able to identify a soap on the desk as a misplaced product and lead a customer (or store employee) to this misplaced product.

In open household environments, creation of semDTs is challenging. Therefore, the knowledge graph should not only contain semDT environment information but also product identifiers like barcodes or product images to enable user assistance in household environments.

4.1.3 Action Execution

The competency questions defined for the product knowledge graph reveal that not much action information is needed. If the SOMA (Beßler et al., 2020b) ontology is integrated, basic tasks, in particular navigation tasks to route a customer to a searched product already are defined and can be performed by a robot. However, if robots are to replenish shelves or perform click & collect orders, additional information needs to be added to the knowledge graph.

4.1.4 Resulting Knowledge Graph Structure

With the given competency questions and their implications, the knowledge graph structure in Figure 4.1 can be created. Here, the main focus is on the product, i.e. the object knowledge in the different ontologies to assist different user needs. The modular ontologies in ProductKG have different foci that can be used to retrieve additional product information for different applications. All ontologies contained in ProductKG with their most relevant properties and their current use in applications as well as the number of axioms are detailed in Table 4.2. In order for the knowledge graph to be actionable, the object knowledge additionally links to action knowledge from the SOMA ontology by defining a product as an object

[13] Ubuca robotics: `https://www.ubica-robotics.eu/`

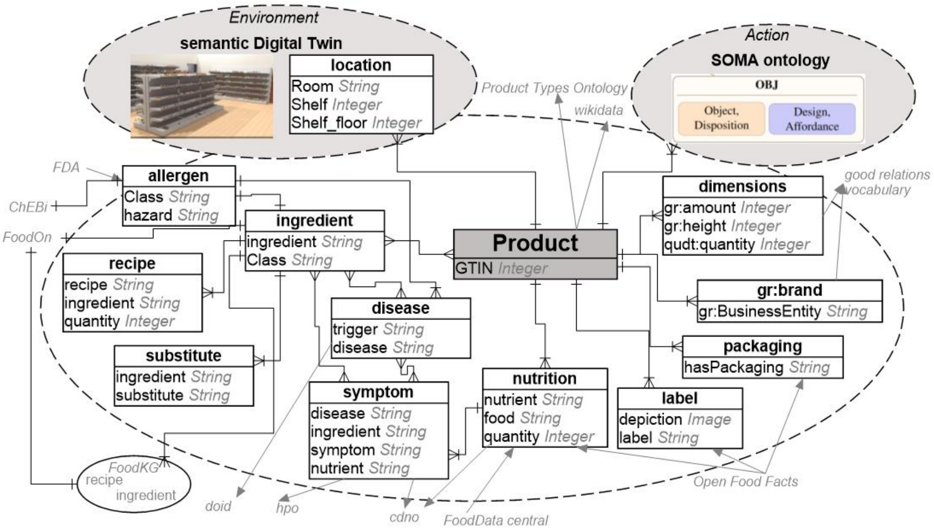

Figure 4.1: Overview of ProductKG ontologies and their links to external sources.

and to a semantic Digital Twin by linking barcode identifiers of products to recognised barcodes in the semDT as well as a location ontology that translates semDT environment information to semantically enhanced environment information that can be used in Augmented Reality applications.

4.2 Acquire the Needed Information

ProductKG consists of modular ontologies that contain structured, semi-structured or unstructured information as described in the following.

4.2.1 Extracting Information from Unstructured Sources

Due to the domain of interest and most needed product information being available in semi-structured or structured HTML documents, no natural language processing needs to be applied.

Table 4.2: Ontologies contained in ProductKG with their most relevant properties and number of contained axioms. *Italic* information sources belong to ProductKG while all other mentioned sources are external.

Ontology	Information Sources	Offered Properties	Axioms
Product Taxonomy	Good Relations vocab wikidata Aldi Nord, dm Food Ontology semDT	gr:hasEAN_UCC-13 wikientry trust:source oboInOwl:hasDbXref pathToCadModel	8,603
Location Ontology	semDT	loc:has_stock loc:stored_in/ stored_on	7,721
Dimension Ontology	QUDT Good Relations vocab	qudt:unit/ hasQuantity gr:depth/ height/ width/ weight	4,312
Brand Ontology	Good Relations vocab	gr:hasBrand/ name has_StoreBrand	5,669
Packaging Ontology		has_PackagingMaterial has_RecyclingProperty	5,435
Label Ontology	Friend of a Friend	foaf:depiction has_label	3,341
Nutrition Ontology	QUDT	qudt:has_quantity has_nutrient	95,192
Symptom Ontology	*Disease* *Nutrient*	has_symptom possible_treatment	1,487
Disease Ontology	*Ingredient* *Symptom*	triggers symptom_of	582
Ingredient Ontology		has_ingredient has_trace	18,050
Recipe Ontology	FoodKG QUDT	food:isRecommendedForCourse hasMealType qudt:has_Quantity	295,231
Substitute Ontology	*Ingredient* QUDT	has_substitute has_purpose qudt:has_Quantity	2,0580
Allergen Ontology	*Disease* FOAF *Ingredient*	has_trigger foaf:has_depiction owl:same_as	1,327

4.2.2 Extracting Information from Semi-structured Sources

For creation of the ProductKG ontologies, semi-structured Web information can be acquired to create structured knowledge sources that machines can query. In order for ProductKG to be usable in various environments, it needs a general product taxonomy that can be used for different domains of daily activities like at home, in drugstores and grocery stores. As mentioned before, I chose a more generalised approach and use Information Extraction techniques on online store sitemaps of different retail domains to automatically create a general product taxonomy.

For this general product classification I create a product class structure from sitemaps of two online stores representing different retail sectors, namely Aldi-Nord[14] (grocery store) and dm[15] (drugstore). I then use Owlready (Lamy, 2017) to generate ontologies out of the extracted information. The resulting product taxonomies are merged into one global product taxonomy that additionally integrates the food product structure of the FoodOn taxonomy.

Online stores usually offer product information like ingredients or brand and product name in semi-structured HTML format. Thus, one can apply Information Extraction techniques to acquire product ingredient information for use in the ingredient ontology and information like brand, awarded labels, weight, filling capacity and country of origin from Codecheck[16], a consumer-oriented product information website.

4.2.3 Extracting Information from Structured Sources

For ProductKG, both structured object and environment information can be acquired as described in the following.

i) Structured Product Information

ProductKG integrates parts of structured Web information like the FoodKG ontologies (Haussmann et al., 2019) for a product-specific recipe recommendation. This is done by reusing the food product classification of the FoodOn

[14] Aldi-Nord sitemap: https://www.aldi-nord.de/.aldi-nord-sitemap.xml
[15] dm sitemap: https://www.dm.de/?view=asSitemap
[16] Codecheck website: https://www.codecheck.info/

ontology (Dooley et al., 2018b) (that has its own import file[17]), which then is intertwined with the product classification in the product taxonomy. With this, products can be identified as ingredients and recipes available in FoodKG can be searched for. ProductKG also integrates nutrition information from the Compositional Dietary Nutrition ontology (cdno) (Andrés-Hernández et al., 2020) by extracting nutrient classes as well as nutritional product information like Nutri-Score, product labels or packaging size for food products from Open Food Facts [18] using Information Extraction techniques in their HTML websites. For food classes, product class nutrition information from the FoodData central is accessed through its API.

ii) Structured Environment Information

In order to generalise applications for the use on different devices and in differing applications, we need to have precise, interchangeable environment information. A standardised map is also beneficial for localisation of products in a store as discussed previously. Thus, I create a location ontology that is linked to precise environment information in a semantic Digital Twin environment model. A location ontology can be generated for any indoor environment following the approach described in (Beetz et al., 2022) and (Kümpel et al., 2021).

The location ontology used in ProductKG is created using the knowledge processing framework KnowRob (Tenorth and Beetz, 2009b; Beetz et al., 2018b) and the semDT environment model created through robot stocktaking. KnowRob can be seen as one of the currently most influential knowledge representation and processing systems in the field of cognitive robots (Olivares-Alarcos et al., 2019; Thosar et al., 2018b). Since KnowRob is an ontology-based system, the semDT is stored in OWL format, allowing for more advanced reasoning on the comprised information. The semDT therefore includes object properties, i.e. relations between individuals, in the following way: <Product_Z has_pose Pose_A>. It also includes data properties, i.e. relations of individuals to data like <Pose_A has_xposition 0.29>.

[17]The food product import file is available at: http://purl.obolibrary.org/obo/foodon/imports/foodon_product_import.owl

[18]Open Food Facts: https://world.openfoodfacts.org/

The process of creating the semDT environment model consists of two major steps: layout detection and store monitoring. During layout detection, rarely changing features of the store (like room size and shelf positions) are captured. This task needs to be performed once for each new store layout. The store monitoring process is the repeating process of stocktaking.

- *Layout Detection:* The layout detection starts with the creation of a 2D map of the store using grid mapping as simultaneous localisation and mapping technique (Grisetti et al., 2007). Afterwards, the robot drives through the store to create a basic scene graph of the environment without product information. Each shelf is detected using a Quick response code (QR code). The position data is added to the semDT in such a way that shelf positions in relation to other shelves or points of interest can be calculated and reasoned about.

- *Store Monitoring:* During store monitoring, frequently changing product positions are detected automatically by the robot. For each shelf, the robot scans the shelf vertically to detect shelf layers and horizontally for each shelf layer to detect price labels and product separators. The stock as the number of products between two product separators is estimated based on detected product features of an RGB-D camera similar to other approaches (Donahue et al., 2014). Each price label contains a barcode that encodes the Global Trade Identification Number (GTIN) of a product, thereby adding product information to the semDT. This product information is continuously updated based on sales data. If the store monitoring process is performed regularly, the semDTs of two different time points can be compared to detect irregularities between calculated inventory and actual inventory that can be reasoned about.

4.3 Process the Acquired Data

For an amplification of semantic information, the acquired data is processed and standardised. As proposed in the methodology, I use string matching to categorise ingredients. All ingredients enclosing the strings `alcohol`, `alcohols` or its german equivalent `alkohol` are categorised as `<alcohol rdf:type Alcohol>`, for example. This is done for all products of the knowledge graph, which means both for food and non-food products. Through such a classification an agent can already reason about the use of products. For example, while most (or all) alcohols can be assumed to be toxic, an agent can infer that if the product contains alcohol and is a food product, it should still be suitable for consumption.

What is more, string matching is used to classify product ingredients as allergens according to information from the U.S. Food & Drug Administration (FDA) website[19]. Besides this, extracted product categories are standardised to lower case words while special characters and numbers are deleted. This allows for an accurate matching of categories of different stores. Furthermore, additional semantic information is added manually as in linking a specific label to a label category.

4.3.1 Enrichment of Environment Information

The positions in the semantic Digital Twin use a fixed reference frame with a given origin, usually set to a randomly chosen corner of the room. In contrast to this, AR devices use the varying device position when starting an application as origin and display its digital content relative to the device position. To solve this discrepancy for a use of the location ontology in both robot and AR applications, all product locations in the location ontology can be encoded relative to shelf, table or other object positions.

Localisation While robots can easily use the semDT to localise themselves even in unknown environments, digital agents need additional services for successful localisation in unknown environments. Using game engine technology to complement knowledge based systems has proven to effectively support decision process especially for object manipulation (Haidu et al., 2018). Game engines can also be used for creation of device-independet applications. Applications developed with the Unity game engine can be used on Smartphones, HoloLens or Magic Leap. Thus, I use the Unity game development platform[20] for creation of the shopping assistant applications.

[19]FDA website: `https://www.fda.gov/`
[20]Unity game engine: https://unity.com/

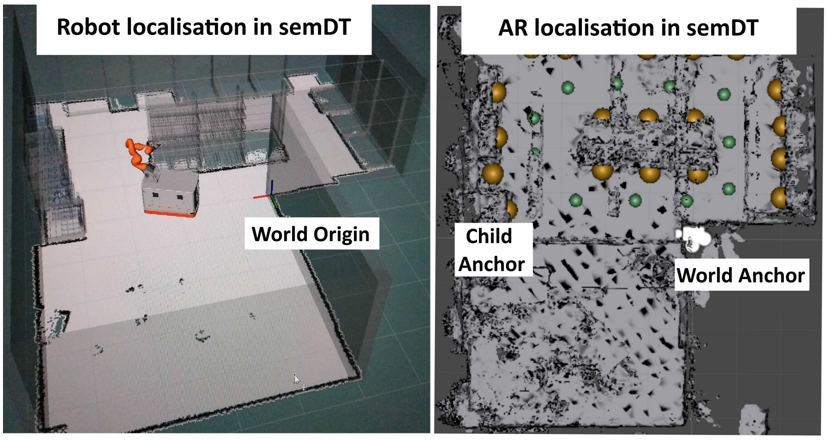

Figure 4.2: Robot map and AR map for localisation of agents.

In the AR shopping applications I use spatial perception and match the world origin of the game to the reference frame origin of the semantic Digital Twin as shown in Fig. 4.2. Here, a world Anchor is used for locking positions in the physical space, a technique that has proven effective in other work (Jakl et al., 2018). World anchors need to be created once and will be loaded in every subsequent run of the application. Child anchors can be set in relation to such a world anchor in the same manner as shelves are set in relation to the world origin of the semDT and can be used as goal positions for routing applications, for example. Thus, product positions can be inferred relative to world anchors and digital content can be displayed.

4.4 Link Distributed Knowledge Chunks

ProductKG links to existing data sources by re-using many available URIs and standard vocabulary as proposed in the Linked Data standards by Bizer, Heath et al. (Bizer et al., 2007). I do not, however, use the `owl:sameAs` property but the `oboInOwl:hasDbXref` annotation property to interlink two data sources, as proposed by the gene ontology (GO) consortium (Consortium, 2019). Both the GO and the FoodOn workgroup are part of the OBO Foundry (Smith et al., 2007), a community focusing on the development of interoperable ontologies for the biological sciences. Thus, I chose to reuse some of the OBO foundry ontologies (besides FoodOn and cdno, ProductKG links to doid (Schriml et al., 2012, 2022) and the human phenotype ontology (Robinson and Mundlos, 2010)) since they are openly accessible, are already following the same upper ontologies and have a strong community.

The product classification is further linked to wikidata whenever possible.

4.4.1 Ontology Alignment

One major challenge in ontology alignment of ontologies with different foci like products, food or actions is the use of diverse upper ontologies. While ProductKG is based on DUL with its descriptions of relations of actions, objects and agents, the OBO foundry ontologies are based on bfo defining occurrents and continuants. Thus, external concepts like processes that are defined as occurents in bfo need to be aligned with the definitions of tasks that classify actions, which are events in DUL. To simplify ontology reuse in ProductKG, I focus on the domain and solely integrate product information. In particular, the main DUL class ProductKG is based upon is *PhysicalObject* - the class of all objects perceivable in an environment. In DUL, a $PhysicalObject \sqsubseteq Object \sqsubseteq Entity$.
In bfo, a $FoodProduct \sqsubseteq FoodMaterial \sqsubseteq MaterialEntity \sqsubseteq Continuant \sqsubseteq Enity$.
In ProductKG, the terms can be aligned by stating that a
$FoodProduct(\text{bfo}) \sqsubseteq groceries \sqsubseteq (ProductOrService(\text{gr}) \sqcap PhysicalObject(\text{DUL}))$
a food product of the Food Ontology is a subclass of groceries in ProductKG, which belongs to the intersection of the Product or Service class of the Good Relations vocabulary and the physical object class of DUL.

4.4.2 Object Entity Linking

Each product instance in the product taxonomy is assigned a Global Trade Identification Number (GTIN), a unique identifier encoded in the price tags of products. The entities of these product instances or the product class are reused in all other ontologies holding product-specific information. Through this, product relations and properties from all ontologies related to the product taxonomy can easily be queried. This is different for the nutrition ontology since it links to FoodOn classes. Therefore, nutrition information is linked to products by creation of a food-nutrition ontology, which links nutrition to the FoodOn classification that already is intertwined with the product taxonomy and hence can be used for food products in the product taxonomy.

Furthermore, ProductKG links to many ontologies by referencing them with the `oboInOwl:hasDbXref` annotation as in `<tax:OatFlour oboInOwl:hasDbXref http://purl.obolibrary.org/obo/FOODON_03301312>`. Since FoodOn provides a much more detailed classification of food products, links them to the plant ontology and other sources, the annotation property provides the link to access such additional information if needed (which in fact is not needed for the domain of the product knowledge graph). For allergens, it links to ChEBI, the NDRF or Drugbank. Symptoms and diseases reference the hpo, cdno and doid ontologies.

4.4.3 Linking Object to Environment Information

To make environment information accessible for digital agents in the location ontology, the semDT environment model can be translated by assigning shelf classes to recognised QR codes (e.g. `<QR_code_123 identifies shelf_1>`) followed by relating all recognised products to the shelf by assigning `<Product_A stored_on shelf_1>` until the next QR code is recognised. To further relate products to the shelf floors they are stored on, the product position can be analysed. If the y position of a recognised barcode changes, it belongs to a different shelf floor than the previously recognised products. Since the robot scans shelf floors from top to bottom, shelf floors can easily be linked to product positions by assigning `<Product_A stored_on shelf_floor_1>`. To further link images to products, one can assign `<Product_A has_depiction image_C>` as described in Chapter 3.

For digital agents using world anchors for localisation, anchor points can be named in accordance to knowledge graph entities for an automated allocation of queried content to anchor positions.

```
(an action
  (type LookingAt)
  (an object
    (type product)))
```

Figure 4.3: Plan description for LookingAt action.

```
(an action
  (type Navigation)
  (to (a location
    (in front of (an object
        (name shelf 2))))))
```

Figure 4.4: Plan description for Navigation action.

4.5 Translate Knowledge to Robot Plan

Action information in KnowRob is stored as episodic memories (Bartels et al., 2019) of performed actions based on general plan descriptions (Koralewski et al., 2019). These general plan descriptions follow an action and task hierarchy where a stocktaking action performed by a robot can be comprised of different tasks like driving, positioning, and scanning. Figure 4.3 shows a general plan description for the action of looking at a product, a subaction of the stocktaking action. Figure 4.4 describes the plan for a navigation action to a destination. With this general plan, the robot knows how to perform a navigation action. Position information for the location `shelf 2` can be inferred from the knowledge graph.

In addition to the action hierarchy, episodic memories can store agent information as participant or performer of an action, assign roles to objects for a certain action (e.g., `shelf 2` in the general plan in Fig. 4.4 would be assigned the role *destination*) and log times and duration of the performed actions.

4.6 Evaluating the Created Graph through Competency Questions

The created product knowledge graph is able to answer all competency questions posed in Table 4.1. To further evaluate the graph I created SPARQL queries for every competency question to retrieve the amount of instances that would be returned in an application. The ProductKG knowledge graph contains 1205 products with a Global Trade Identification Number (GTIN) identifier and 10,000 recipes. The table shows the amount of products/ recipe ingredients that can be retrieved for the different competency questions, which can be used for different applications. For the table, the competency questions were further refined for a differentiated view on the contained knowledge.

	Number of products with a GTIN	1,205
Q1	Number of products with a category assigned	712
Q2	Number of products with a category and ingredients	447
Q3	Number of products with a category and a preference	472
Q4	Number of products with a category and two preferences	454
Q5	Number of products with ingredients	451
Q6	Number of products with a brand	471
Q7	Number of products with labels	293
Q8	Number of products with hazardous ingredients	310
Q9a	Number of ingredients in recipes	401,634
Q9b	Number of products that link to ingredients	58
Q10	Number of ingredients with Substitutes	185
Q11a	Number of food objects with nutritional values	2,707
Q11b	Number of products with nutritional values	62
Q12	Number of products that can cause a disease	334
Q13a	Number of products that can cause a symptom	54
Q13b	Number of ingredients that can cause a symptom	13
Q14a	Number of products that can ease a symptom	451
Q14b	Number of ingredients that can ease a symptom	58

Table 4.3: Amount of results that can be retrieved for the different competency questions.

4.7 Discussion and Related Work

This chapter has shown how the methodology presented in Chapter 3 can be applied for creating a product knowledge graph with omni-channel consumer applications. While the applications will be shown in Chapter 6, this chapter focused on how to create an actionable knowledge graph that can be accessed in different and new environments both by digital and robotic agents to accurately and reliably locate articles and support users in daily shopping activities.

Since localisation of objects is very important for many applications, recent research has focused on standardised mapping, in particular of retail environments. The benefits of standardised maps has been pointed out in (Davis et al., 2006). The idea of creating semantic Digital Twins of retail environments as standardised maps has been proposed in (Kümpel et al., 2021) for shopping or shop-floor assistance. In contrast to other approaches, semDTs are very accurate since they encode environment information perceived by sensor data, which is very accurate in standardised retail environments. It has additionally been shown how semDTs can be used in retail applications to route a customer to a searched product on different devices such as smartphone and robot (Kümpel et al., 2023).

There has also been research on creation of product knowledge graphs (e.g. (Lee et al., 2006; Zalmout et al., 2021)) and food product knowledge graphs, resulting in partially interlinked domain ontologies with consumer applications ranging from agricultural over health specific to recipe centred (e.g. (Dooley et al., 2018b; Cantais et al., 2005; Haussmann et al., 2019)). Although the created ontologies, knowledge graphs and their exemplary applications are impressive, they do not connect the contained object or product information to environment information like the ProductKG does. ProductKG contains different modular ontologies that offer information for different daily applications that can be accessed from different agents. ProductKG therefore is an example of a knowledge graph that contains vast product information to answer complex user queries, which is linked to precise environment information that can be translated to positions on different agents.

It has to be noted that semDTs can not easily be created for all environments. Retail environments are very structured, thus making it easy for robots to perceive the available objects. In contrast, household environments are open and less standardised, making it more difficult to accurately identify all available objects.

Chapter 5

An Actionable Food Cutting Knowledge Graph for Robotic Task Execution

This chapter describes how the methodology introduced in Chapter 3 can be applied to teach a robot new task variations of cutting. In particular, a robot that knows how to cut a slice of bread shall be enabled to quarter an apple or slice an orange by accessing the actionable knowledge graph.

Remember, robots are challenged by situations where they are confronted with new tasks, new environments or new objects for which they lack knowledge. The high success in automation of simpler tasks such as vacuum cleaning or lawn mowing is due to the fact that: i) these tasks show a small variance, ii) are less context-dependent and iii) environments are not of interest other than with respect to the need to avoid obstacles (although most vacuum cleaners don't even avoid walls or other obstacles). These robots are typically pre-programmed or trained for specific tasks only, and can not fulfil new requests. The previous chapter has already shown how semantic Digital Twins help robots to localise and efficiently perform tasks in unknown environments. This chapter will detail how actionable knowledge graphs can be used by robots to learn new task variations on new objects by allowing robots to build upon and reuse existing Web knowledge to infer how to address a new task or carry out known tasks on new objects. Rather than training a robot specifically to perform in every possible context, on every possible object, etc., actionable knowledge graphs can equip a robot with relevant background and common-sense knowledge that allows them to generalise to unseen tasks without having been specifically trained for them. A crucial aspect is that the knowledge can be linked to the actual programs that execute the action on the robot so that the knowledge supports manipulation. For this, one has to bear in mind that the created knowledge graph is not supposed to perfectly model an action domain but shall be used to return query results that can parameterise body movements for different behaviour.

Competency Questions The food cutting knowledge graph should help users in answering the questions detailed in Table 5.1.

Q1	What are task variations of cutting?
Q2	What are the commonly used action verbs in the context of cutting actions?
Q3	What food can be cut/ sliced/ diced?
Q4	What tool can be used for a specific cutting action?
Q5	What position is needed for cutting?
Q6	How often does the cutting action need to be performed?
Q7	Does the given cutting action depend on another (prior) action?

Table 5.1: Competency Questions the food cutting knowledge graph should be able to answer.

While Q1 and Q2 can be used to understand the domain of interest since they aim at properly integrating different kinds of cutting actions such as chopping, dicing, slicing or quartering as well as their part of speech (i.e. cutting (substantive) relates to cut (verb)), Q3 and Q4 aim at properly modelling the dispositions and affordances of objects involved in cutting actions. Q5 to Q7 are needed in order for a robot to be able to actually perform task variations of a cutting action. For the execution of task variations, the robot needs to know the position for cutting (e.g. for halving a different position is needed than for slicing), how often the action needs to be performed (for halving only one cut is needed while slicing a whole cucumber will require many repetitions of the cutting action), if prior actions like peeling or core removal are required as well as a translation to the actual body movements for performing the cutting task.

5.1 Define Necessary Knowledge Sources

Due to the high importance of action information in the food cutting knowledge graph, the DUL top-level ontology (Presutti and Gangemi, 2016) as well as the SOMA action ontology (Beßler et al., 2022) should be reused. The combination of motion and task planning with detailed information from SOMA can help robots to find a fitting plan to their action. Another resource that should be included is an ontology of common failures that helps to diagnose unsuccessful actions and to find alternative paths to recover from them (Diab et al., 2020; Akbari et al., 2019). The failure interpretation ontology in (Diab et al., 2020) covers a wide variety of possible failures that often take place in autonomous planning and execution. The failure ontology includes both the failure classification as well as the reasoning for why the action has failed. Some examples are failures based on location and geometry, direct hardware failures of the robot, as well as failures dependent on faulty reasoning skills and shortcomings in cognitive ability.

For instance, if a robot is not able to grasp an object because it cannot reach it, the ontology diagnoses it as a *reachability failure*. In case the robot was unable to reach the object with a previously applied grasp (e.g. from the side), the failure handling would propose to re-adapt its action plan by trying another grasping pose, such as a from the top. By having access to failure knowledge, robotic agents can reason about their failures to reach the desired goal states of subtasks in the action plan and can re-calibrate their actions to succeed with the plan.

To further define the knowledge sources that should be integrated into the food cutting knowledge graph, one should consider the competency questions and their implications on object, environment and action information that is needed.

5.1.1 General Object Knowledge

To include general object knowledge about fruits and vegetables, one can employ the FoodOn ontology (Dooley et al., 2018a) as a taxonomy. To further define needed object knowledge, let us consider the competency questions in Table 5.1.

- Q3: Object dispositions and affordances. When cutting fruits and vegetables, important object information about the cuttable object can be its shape, size, color, genus or anatomical parts. While some of such information might be available in plant taxonomies like the plant ontology (Jaiswal et al., 2005), most of the needed information is not available in structured or semi-structured sources. One solution is to gather the needed information about which anatomical parts exist and are relevant for the cutting tasks from biological books about fruit anatomy (e.g. (Crang et al., 2018)) or instructional videos from the cooking domain (e.g. (Epicurious, 2019)). For both approaches, the information needs to be modelled manually. It seems that no matter which anatomical parts are present in a fruit or vegetable, the important influence factor for action parameterisation in cooking activities is the parts' *edibility*. On the one hand, both an apple and an orange do have a peel, but peeling is only mandatory for an orange since its peel is inedible[1], while an apple will usually only be peeled if it is specifically stated in the instruction of a recipe. On the other hand, both an apple and an orange do have some form of a core, but since the core of an apple is inedible (i.e. usually removed before eating), we can infer that it has to be removed before eating.

- Q4: Tool dispositions and affordances. The objects involved in cutting food (i.e. bread, knives and cutting board) should also be modelled and set in relation to their dispositional properties. However, the dispositional property of an object varies based on the requirements of the environment. For example, a *CuttingBoard* can be used to provide support to any object which undergoes cutting, with a disposition type *Deposition*. However, the same *CuttingBoard* can also be used as a cover for a container, as in the case in which there is no lid available for a pot, as both have the disposition type of *Coverage* as explained in (Pomarlan and Bateman, 2020). The purpose to model the dispositions of objects is to enable the robot to infer the potential tasks afforded by the objects and what actions are possible given a set of objects, similarly to how a human can reason about the possibilities with particular objects. In reverse, it also makes it possible to reason about the possibility to execute the task given a set of objects.

[1]Although sometimes orange peels are used for cooking, for simplicity it is assumed that they still need to be removed from the rest of the orange.

5.1.2 Environment Structure

To reliably answer the competency questions in Table 5.1, the knowledge graph does not need to encode much environment information since the robot only needs to be enabled to recognise different food objects and tools to be used for cutting. The existing perception framework is able to detect such objects. To further increase recognition success rates, additional object images can easily be acquired from the Web. For food like apple, orange or cucumber, many images are freely available just like online stores offer many images of different knives.

5.1.3 Action Execution

Relevant knowledge for action execution of task variations is not offered in structured or semi-structured sources. One source that offers part of speech information as well as linkage of words with similar meaning is FrameNet (Baker et al., 1998). For the word cutting, FrameNet offers the information *"The Agent is the person cutting the Item into Pieces."*[2], thus linking an agent to a cutting action, a cuttable object as well as result objects similar to the approach in SOMA. However, FrameNet equals cutting to carving, chopping, slicing and dicing and therefore needs adaptions if it is to be used for action execution of different cutting tasks. To further define the necessary object properties, one can use unstructured sources like Biology textbooks (Crang et al., 2018), cooking videos (Epicurious, 2019), large language models or WikiHow instructions. WikiHow articles describe tasks with different levels of granularity, covering high-level ("How to Make an Apple Pie") as well as low-level task instructions necessary to achieve the high-level tasks ("How to Core Apples") (Zhou et al., 2022) and offer information on Websites that can easily be accessed. Therefore, WikiHow is a good source to analyse possible commands as well as task relations.

Furthermore, the competency questions 5-7 have to be considered for action execution:

- Q5: Cutting position. A cutting action in SOMA can be executed by the robot as it translates the cutting task into subtasks of picking up, cutting

[2] FrameNet cutting frame https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Cutting

and placing. From the ontology, the robot can further infer some of the action parameters needed for performing the task: the `object_to_grasp` and `object_to_approach` as described in (Dhanabalachandran et al., 2021). Since the robot knows how to cut a slice of bread but doesn't know any other cutting actions, at least one more important action parameter needs to be added to the ontology, which is the `halving_pose`. Contrary to the `slice_pose`, which is near an end of an object in order to produce a slice, the `halving_pose` is in the center of an object.

- Q6: Number of repetitions. The initial task parametrisation will let a robot cut one slice of bread if it is told to "cut the bread". If we want the robot to be enabled to cut the whole bread or to know how often to perform a cutting action if told to "quarter an apple", the ontology needs to integrate the needed number of repetitions or - since the number of repetitions heavily relies on the type of food being cut and the size of pieces being produced - other information that can be used by a robot to infer the number of repetitions.

- Q7: Task dependencies. As mentioned before, some food affords additional tasks like peeling. What is more, some tasks need to be performed prior to cutting for a cutting action to be successful/ have the intended outcome. Such knowledge needs to be integrated into the ontology. In addition to that, some tasks depend on other tasks. For quartering an apple, for example, one actually has to cut the apple into halves to then again halve the created pieces to create quarters of an apple. Task dependencies thus strongly relate to the number of repetitions mentioned in Q6.

5.1.4 Resulting Knowledge Graph Structure

Considering the sources needed for creating an actionable food cutting knowledge graph as discussed previously, the knowledge graph structure in Figure 5.1 can be created, which connects the three top-level ontologies DUL, SOMA and FoodOn (including the plant ontology) and further defines classes that are necessary for the domain of cutting. Through the integration of SOMA the knowledge graph links object to action as well as environment information, since it uses image

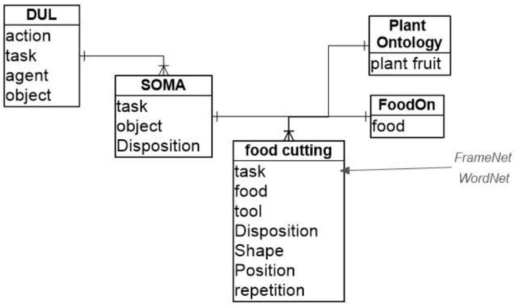

Figure 5.1: Food cutting knowledge graph structure.

recognition to perceive food objects. The resulting knowledge graph contains 480 axioms and offers the properties detailed in Table 5.2.

5.2 Acquire the Needed Information

The food cutting knowledge graph mainly consists of top-level ontology excerpts that are manually extended by knowledge from unstructured sources as defined through the competency questions and explained in the following.

5.2.1 Extracting Information from Unstructured Sources

The food cutting knowledge graph shall contain information on how a cutting action is different to a slicing or halving action. To investigate the relevance of these different commands, one can examine the occurrences of *Cutting* and its hyponyms (that can be found in lexical resources like FrameNet, WordNet or Thesaurus) in WikiHow articles. Through an analysis of WikiHow instructions, relevant cutting hyponyms for the cooking domain can be identified and added to the ontology. The hyponyms with the most occurences in WikiHow articles are detailed in Table 5.3. The given 15 cutting verbs can be grouped into action groups that have similar results and require similar body motions.

Hence, the knowledge graph should contain the action classes cutting, slicing, dicing, julienning, halving and quartering as well as the action class filleting (which actually is a preparation step).

To further include the objects identified as necessary for task execution through the competency questions, the following classes are created:

Information Source	**Offered Properties**	**Usage**
DUL	classifies	a disposition can *classify* a specific tool to use
SOMA	hasDisposition	an object has a *disposition*
	affordsTask	a disposition *affords* a specific *task* to be executed
	affordsTrigger	a disposition *affords* a specific tool as *trigger*
cut	hasPart	a food *has* some food *part*
	hasEdibility	a food part *has* an *edibility* such as it should be avoided
	offersPosition	a cutting task *offers* a slicing or halving *position*
	hasInputObject	a cutting task *has* an *input object*, either a food or a piece of food
	hasResultObject	a cutting task *has* two *result objects*
	requiresPriorTask	a cutting task might *require* a *prior task* to be executed
	repetitions	a cutting task needs a number of *repetitions*

Table 5.2: Information sources contained in the food cutting knowledge graph and their offered properties.

- $Tool, Food \sqsubseteq DUL : PhysicalObject$
- $FoodPart, FOODON\_00001057 \sqsubseteq Food$ (where the FOODON_00001057 ID identifies a "plant fruit food product", *A food product derived from plant fruit.*[3])
- $Knife, Spoon, Peeler, AppleCutter \sqsubseteq Tool$
- $CuttingAction \sqsubseteq DUL : Task$
- $Cutting, Slicing, Julienning, Dicing, Halving, Quartering \sqsubseteq CuttingAction$
- $slicing\_position, halving\_position \sqsubseteq DUL : SpaceRegion$

[3]The FoonOn taxonomy and its FOODON_00001057 ID can be accessed here: http://purl.obolibrary.org/obo/FOODON_00001057

Table 5.3: Analysing the occurrences of 15 different hyponyms for *cut* in the WikiHow data from (Zhang et al., 2020).

Verb	Occurrences	action execution	action group
cut	23486	Creating a slice	cutting
chop	9221	Cutting into pieces	dicing
slice	8200	Creating many slices	slicing
mince	2164	Similar to dicing, but more finely cut	dicing
dice	1631	Creating cubes	dicing
pare	921	Creating a slice	cutting
carve	888	Creating a slice	cutting
cube	516	Creating cubes	dicing
halve	346	Cutting into halves	halving
julienne	189	Creating stripes	julienning
snip	162	Cutting into pieces	slicing
saw	149	Cutting with different motion	cutting
quarter	125	Cutting into quarters	quartering
fillet	91	Preparing food before cutting	filleting
sliver	54	Creating thin slices	slicing

- $Peeling, CoreRemoving, StemRemoving, Filleting \sqsubseteq DUL : Task$
- $Cuttability, Edibility, CoreRemovability, Peelability, StemRemovability, \sqsubseteq SOMA : Disposition$
- $Cube, Slice, Stripe \sqsubseteq Shape$

5.2.2 Extracting Information from Structured Sources

The information extracted from structured sources is restricted to modelling a basic class structure. Therefore, the following classes are extracted:

- $DUL : Task \sqsubseteq DUL : EventType \sqsubseteq DUL : Concept \sqsubseteq DUL : SocialObject \sqsubseteq DUL : Object \sqsubseteq DUL : Entity$
- $Food, Tool \sqsubseteq DUL : PhysicalObject \sqsubseteq DUL : Object \sqsubseteq DUL : Entity$
- $DUL : SpaceRegion \sqsubseteq DUL : Region \sqsubseteq DUL : Abstract \sqsubseteq DUL : Entity$
- $Shape, SOMA : Disposition \sqsubseteq ObjectProperty \sqsubseteq DUL : Quality$

To additionally include plant fruits that can be found as ingredients in recipes such as in the Recipe1M dataset (Marın et al., 2021), all subclasses of the plant fruit food product class of FoodOn can be acquired. To reduce the problem space (since for every food object dispositions and FoodPart relations need to be created), recipe instructions can be analysed in regards to containing plant fruit ingredients, similarly to analysing occurences of tasks in WikiHow descriptions. If all plant fruits that occur in at least 1% of instructions shall be included in the ontology, the following 19 food products can be added:
almond, apple, banana, bean, cherry, citron, coconut, cucumber, kumquat, lemon, lime, olive, orange, pepper, pineapple, pumpkin, strawberry, squash, tomato.

5.3 Process the Acquired Data

Due to the fact that for creation of the food cutting knowledge graph not much information can be automatically extracted from unstructured sources, the data doesn't need to be cleaned. On the contrary, the data needs to be enriched to semantically model task dependencies as well as object affordances and dispositions so that robots can access the graph for inferring parameters for action execution.

5.3.1 Enrichment of Object Information

Following Turvey's definition of dispositions (Turvey, 1992b), for the disposition type *Cuttability* the objects of type *CuttingTool* act as *bearer* and the objects of type *Food* act as *trigger* for the task of cutting. In household environments, the most essential objects include knives that play the role of the *CuttingTool* that afford cutting. To further differentiate between the dispositions of different knives, the following tools are added to the knowledge graph as cutting, core removal or peeling tools:

- *KitchenKnife, BreadKnife* $\sqsubseteq Knife \sqsubseteq CuttingTool$
- *KitchenKnife, Spoon* $\sqsubseteq CoreRemovalTool \sqsubseteq Tool$
- *Peeler, Hand* $\sqsubseteq PeelingTool \sqsubseteq Tool$
- *AppleCutter* $\sqsubseteq CuttingTool$

In addition to such a tool classification, the extracted food can further be classified. The plant ontology (Jaiswal et al., 2005) offers some classes that can be reused to refine the contained food knowledge. Here, the goal is to only include plant classes that help to infer food part edibility and not to best model the plant taxonomy. For example, it is beneficial to define a "citrus fruit" class as a parent class of lemon, lime and orange that all have an inedible peel. Therefore, the following class relations are added to the knowledge graph:

- $cherry,\ olive \sqsubseteq stonefruit$
- $almond,\ coconut \sqsubseteq nutfruit(PO\_0030102)$
- $citron,\ kumquat,\ lemon,\ lime,\ orange \sqsubseteq citrusfruit(FOODON\_03301337)$
- $apple \sqsubseteq pomefruit(PO\_0030110)$
- $cucumber,\ pumpkin,\ squash \sqsubseteq pepofruit(PO\_0030111)$
- $banana,\ pepo\ fruit,\ pepper,\ pineapple,\ pome\ fruit,\ tomato \sqsubseteq berryfruit$ $(PO\_0030108)$

5.3.2 Enrichment of Action Information

The food cutting knowledge graph already contains the different action groups that were analysed in Tab. 5.3. However, if we closely look at how different cutting actions are performed, task dependencies can be identified as visualised in Figure 5.2. Here, three different starting tasks are distinguished: 1) cutting,

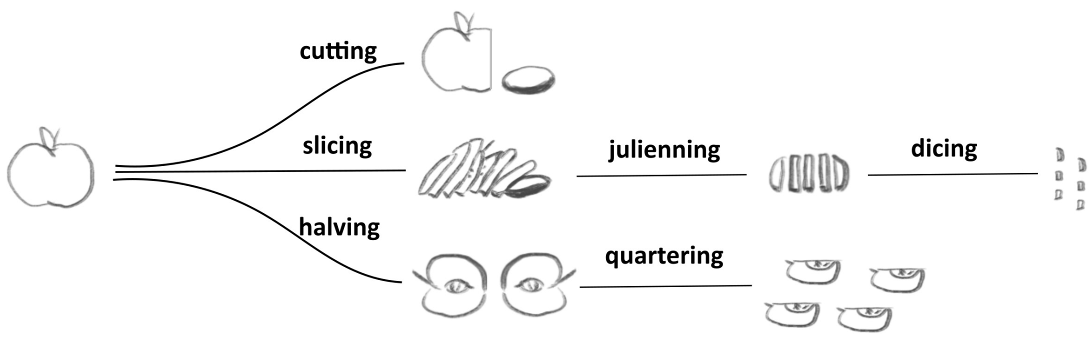

Figure 5.2: Cutting Task dependencies.

$$
\begin{array}{l}
Cutting \sqsubseteq = 1\ repetitions \\
Halving \sqsubseteq = 1\ repetitions \\
Quartering \sqsubseteq = 1\ repetitions \sqcap (requiresPriorTask.Halving) \\
Slicing \sqsubseteq > 1\ repetitions \\
Julienning \sqsubseteq > 1\ repetitions \sqcap (requiresPriorTask.Slicing) \\
Dicing \sqsubseteq > 1\ repetitions \sqcap (requiresPriorTask.Julienning)
\end{array}
$$

Figure 5.3: Representation of task dependencies and number of task repetitions as visualised in Fig. 5.2.

which was defined as cutting off one slice in Tab. 5.3, 2) slicing, which is a repeated cutting action, performed until the whole food is cut into slices of similar thickness and 3) halving, which is a single cutting action at the centre of a food object. The three additional cutting tasks julienning, dicing and quartering actually depend on the tasks of slicing and halving. For 4) quartering, the halves created through halving a food can be halved again to create quarters of the food object, while for 5) julienning the slices created through slicing are taken as input objects to create stripes of food and for 6) dicing the stripes created through julienning are cut into cubes.

These task dependencies can also be modelled in the knowledge graph as shown in Figure 5.3, where the number of repetitions is stated as either being equal to one or more than one. The idea behind this is that the robot can easily calculate an approximate slice/ stripe or cube thickness based on food object size but it does not know whether to cut once or multiple times. To further enable robots to easily query cutting tasks in the knowledge graph for automated retrieval of task information, all action adjectives of Table 5.3 and their corresponding action verbs should be added to the knowledge graph. This can be done by creating subclasses as in the following:

- $Carving,\ Paring,\ Sawing \sqsubseteq Cutting$
- $Slivering,\ Snipping \sqsubseteq Slicing$
- $Chopping,\ Cubing,\ Mincing \sqsubseteq Dicing$

All cutting actions then are assigned an annotation property with the respective action verb as in `<cutting verb cut>`.

5.4 Link Distributed Knowledge Chunks

At this point the food cutting knowledge graph contains food objects, object shape, tools for cutting food, different cutting tasks and cutting positions as well as object dispositions. Now, food objects need to be set in relation to object dispositions, tools and cutting tasks. What is more, task relations should be modelled by integrating food part inedibility.

5.4.1 Ontology Alignment

Since we use the same top-level ontologies as in the previous chapter, the ontologies can similarly be aligned.

In DUL, an action executes a task and can have a physical object as a participant. From SOMA we know that cutting is a task performed on an object. We have to add the information that fruits and vegetables from the FoodOn taxonomy are food. Remember, in DUL, a $PhysicalObject \sqsubseteq Object \sqsubseteq Entity$.
In bfo, a $PlantFruitFoodProduct \sqsubseteq FoodMaterial \sqsubseteq MaterialEntity$
$\in Continuant \in Enity$. In the food cutting knowledge graph, the terms can be aligned by stating that
$PlantFruitFoodProduct(\text{bfo}) \sqsubseteq Food \sqsubseteq PhysicalObject(\text{DUL}) \sqsubseteq Object \sqsubseteq Entity$
all food objects extracted from the FoodOn and plant ontologies, which are subclasses of the plant fruit food product class, are of type food, that belongs to the physical object class of DUL, which again is an object and entity in DUL.

5.4.2 Linking Object to Environment Information

To model the position needed for performing different cutting tasks, the following axioms are added to the knowledge graph:

- $Cutting, Slicing, Julienning, Dicing \sqsubseteq \exists offersPosition.slicing\_position$
- $Halving, Quartering \sqsubseteq \exists offersPosition.halving\_position$

To further link existing food objects and food parts available in the environment to task execution, input and result objects for the different cutting tasks

$$
\begin{array}{l}
Slice \mathbin{\dot\cup} (FoodPart \sqsubseteq Food) \\
Cutting \sqsubseteq \exists hasInputObject.(Food \sqcup FoodPart) \sqcap hasResultObject. \\
\qquad (= 1\ FoodPart) \sqcap hasResultObject.(= 1\ Slice) \\
Slicing \sqsubseteq \exists hasInputObject.(Food \sqcup FoodPart) \sqcap hasResultObject. \\
\qquad (= 1\ FoodPart) \sqcap hasResultObject.(= 1\ Slice) \\
Julienning \sqsubseteq \exists hasInputObject.Slice \sqcap hasResultObject.(= 1\ Slice) \sqcap \\
\qquad hasResultObject.(= 1\ Stripe) \\
Dicing \sqsubseteq \exists hasInputObject.Stripe \sqcap hasResultObject.(= 1\ Stripe) \sqcap \\
\qquad hasResultObject.(= 1\ Cube) \\
Halving \sqsubseteq \exists hasInputObject.Food \sqcap hasResultObject.(= 2\ Halve) \\
Quartering \sqsubseteq \exists hasInputObject.Halve \sqcap hasResultObject. \\
\qquad (= 2\ FoodPart)
\end{array}
$$

Figure 5.4: Representation of environment-specific object knowledge that is important for executing cutting actions in order to identify the object to cut.

need to be defined. This can be done as demonstrated in Figure 5.4. Here, each cutting task is assigned an input object and two resulting objects. For all tasks that do not depend on a prior cutting task to be executed, these input objects are defined either as only *Food* or *FoodPart*. For all tasks that do depend on a prior cutting task to be executed, the result object of the prior task is taken as input object.

5.4.3 Linking Object to Action Information

To model food part edibility, food parts that usually are removed before eating like the core of an apple or cherry, the peel of an orange, the stem of a cucumber and the seeds of a pumpkin are classified as being inedible.

Including important object features such as the ones in Fig. 5.5 in the ontology enables robots to infer that the core of an apple or the peel of an orange needs to be removed during (or prior to) task execution. The statements link object dispositions to food objects, cutting tasks and tools that can be used for the specific tasks. In general, the modelled knowledge distinguishes between edible parts (e.g. apple skin), parts that must be removed before eating due to health or taste reasons (e.g. orange peel) and parts that should be removed but can be eaten if necessary (e.g. apple core). What is more, Fig. 5.5 shows that the knowledge

$$
\begin{aligned}
&Apple \sqsubseteq \exists hasDisposition.(Cuttability \sqcap [\exists affordsTask.CuttingAction] \sqcap \\
&\qquad [\exists affordsTrigger.CuttingTool]) \\
&Apple \sqsubseteq \exists hasDisposition.(Peelability \sqcap [\exists affordsTask.Peeling] \sqcap \\
&\qquad [\exists affordsTrigger.Peeler]) \\
&Apple \sqsubseteq \exists hasDisposition.(CoreRemovability \sqcap \\
&\qquad [\exists affordsTask.CoreCutting] \sqcap [\exists affordsTrigger.CuttingTool]) \\
&Apple \sqsubseteq \exists hasPart.(Core \sqcap hasEdibility.ShouldBeAvoided) \\
\\
&Orange \sqsubseteq \exists hasDisposition.(Cuttability \sqcap [\exists affordsTask.CuttingAction] \sqcap \\
&\qquad [\exists affordsTrigger.Knife]) \\
&Orange \sqsubseteq \exists hasDisposition.(Peelability \sqcap [\exists affordsTask.Peeling] \sqcap \\
&\qquad [\exists affordsTrigger.Hand]) \\
&Orange \sqsubseteq \exists hasPart.(Peel \sqcap hasEdibility.MustBeAvoided) \\
\\
&Core,\ Peel \sqsubseteq FoodPart
\end{aligned}
$$

Figure 5.5: Representation of task-specific object knowledge that is important for executing cutting actions on an apple or orange.

graph also models that an apple affords to be peeled. If a recipe states to peel an apple before cutting, a robot can still use the knowledge graph for action execution since it can infer the correct tool to use for the task. However, since the peel of an apple is edible and does not need to be removed, the additional inedibility statement is not added for the apple peel and therefore the robot will usually cut the apple without peeling it beforehand.

5.5 Translate Knowledge to Robot Plan

The cognitive architecture CRAM is utilised to carry out robot actions in human environments (Beetz et al., 2010). CRAM adopts action descriptions expressed as generalised plans, capable of executing a broad range of variations within a particular action category (Kazhoyan et al., 2021). The highest-level action designator for cutting food is illustrated in Figure 5.6, which functions as a placeholder for information that is yet to be determined. Notably, the cutting task requires the specification of several parameters, including **?repetitions, ?prior_action, ?depends_on_action, ?object_acted_on, ?object, ?arm, ?arm, ?pose** and

?goal, all of which are represented as variables and will be served with information queried from the knowledge graph at runtime based on the context of the task action.

```
(perform
  (an action
     (type :cutting (?repetitions (:boolean)))
     (?prior_action (an action (type :preparing)))
     (?depends_on_action (an action (type :cutting)))
     (?object_acted_on (an object (type :food) (pose ...) ...))
     (?object (an object (type :tool) (pose ...) ...))
     (?arm (a body_part (type :arm) (pose ...) ...))
     (?arm (a body_part (type :arm) (?pose (on :food)) ...))
     (?goal (:success (:cutting (on :food))))))
```

Figure 5.6: CRAM representation of high-level cutting action.

The process of querying data from the knowledge graph not only enables generalisation but also facilitates the development of robust tasks that can be applied to various cutting setups.

A cutting action in the food cutting knowledge graph can be executed by the robot as it translates the cutting task into subtasks of picking up, cutting and placing, where the picking up task is further broken down into approaching, grasping and lifting, the cutting task can be broken down into the body movements of approaching, lowering and lifting and the placing task can be broken down into approaching, releasing and lifting. From the knowledge graph, the robot can infer the action parameters needed for performing the task as visualised in Figure 5.7: additional actions that need to be performed in advance like peel-

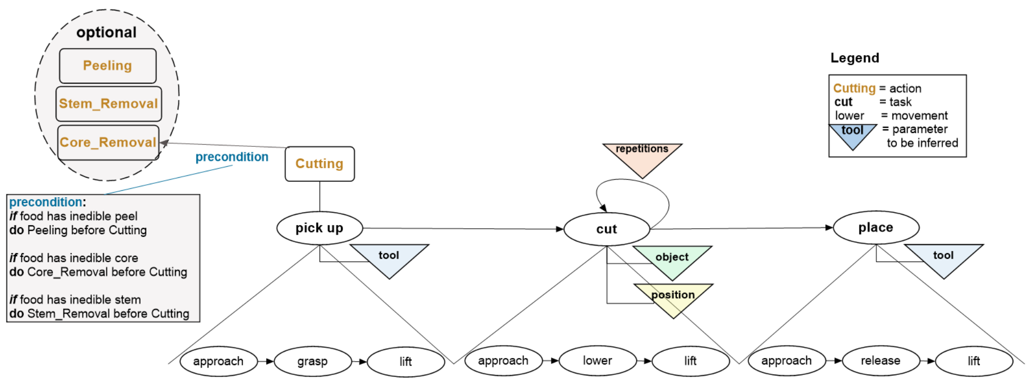

Figure 5.7: Food cutting parametrisation with ontology.

ing can be inferred as **?prior_actions**, cutting tasks that depend on each other like quartering depends on halving as visualised in Figure 5.3 can be queried by **?depends_on_action**. To start execution of the task, the tool to pick up can be inferred as **?object**, for cutting, the cuttable object as **?object_acted_on**, the cutting **?pose** as well as the number of **?repetitions** can be queried and for placing, the tool as object to place can be inferred. In CRAM, the **?goal** is reached when the action was performed successfully, which means that the body parts are at their initial positions and, for the task of cutting, the **?object** has been placed and the **?object_acted_on** has been cut.

The next chapter will show the queries the robot uses to infer action parameters as well as applications of both actionable knowledge graphs presented in this Thesis.

5.6 Evaluating the Created Graph through Competency Questions

The created food cutting knowledge graph can be used to answer all posed competency questions posed in Table 5.1. For an additional evaluation and future extension of the approach to other actions, for each competency question the found solution is given in Table 5.4. The contained knowledge can successfully be used by robotic agents to perform task variations of cutting. What is more, the proposed solution can be extended to other meal preparation tasks like pouring or stirring.

5.7 Discussion and Related Work

This Chapter has shown how the methodology presented in Chapter 3 can be applied for creating a food cutting knowledge graph that can be used by robots to learn and execute new task variations of cutting actions. In particular, I have presented how action groups that represent similar body movements can be created for a specific action category and how to acquire and link object knowledge to object properties. The action groups relate to object dispositions and have affordances that are used to parameterise general action plans. Thus, a robot can execute task variations by querying the actionable knowledge graph with a given task and fruit or vegetable.

This work is based on the knowledge framework KnowRob (Beetz et al., 2018a; Tenorth and Beetz, 2009a), one of the most influential knowledge and reasoning frameworks in the field of cognitive robotics (Olivares-Alarcos et al., 2019; Thosar et al., 2018a). Other robotic knowledge frameworks with similar functionalities are ORO (Lemaignan et al., 2010), OROSU (Gonçalves and Torres, 2015) and PMK (Diab et al., 2019). KnowRob recently got extended by SOMA (Beßler et al., 2022), which defines roles of objects during events, and how they can be used for a more flexible task execution (Beßler et al., 2020a). It has already been shown how the task of cutting bread can be flexibly adapted if a given object cannot be used for task execution in (Dhanabalachandran et al., 2021). Therefore, this work also uses SOMA as an upper ontology for translation of actions into body movements.

Although the proposed methodology can successfully be used by robotic agents to perform task variations of cutting, in the proposed version the knowledge was mostly acquired and added manually. Some of the knowledge acquisition solutions can be automated. Additionally, the created knowledge graph is rather small, containing only 19 food products. However, I the methodology can easily be extended to include all FoodOn food products and a broader range of tasks. Additionally, it can be used to teach robots how to execute task variations of other meal preparation actions.

	solution for cutting	for other tasks
Q1	defining action groups	=
Q2	analysing lexical resources	=
Q3a	modelling food cuttability	modelling object to tool disposition
Q3b	modelling task dependencies	=
Q4	modelling tool dispositions	=
Q5	modelling cutting positions	model positions, speed, angle
Q6	modelling task repetitions	=
Q7	Defining additional actions	=

Table 5.4: Solutions developed in this Thesis to answer the Competency Questions of Table 5.1. = means that the solution for cutting would work for other tasks as well.

Chapter 6

Applications of Actionable Knowledge Graphs

This Chapter shows how both robotic and digital agents can easily query actionable knowledge graphs for knowledge retrieval and shows exemplary applications of the actionable knowledge graphs created in Chapter 4 and 5. The applications use different environment information for assisting users in daily activities and can be deployed on different agents. What is more, object as well as activity knowledge can be visualised in applications for user assistance.

6.1 Knowledge Retrieval in Actionable Knowledge Graphs

One of the benefits of using actionable knowledge graphs is their platform-independency. Both a robotic agent running KnowRob and a digital agent like a smartphone can easily connect to the knowledge graph database by accessing its REST API. Since SPARQL is the standard query language for graph databases (Angles and Gutierrez, 2008), I use it to access the actionable knowledge graph database.

In (Kümpel et al., 2023) we proposed a query template to be used by various agents to dynamically generate queries for shopping assistance. However, for the actionable knowledge graphs proposed in this work the query template can not be used since agents are expected to ask a broader range of queries instead of retrieving only product locations. In actionable knowledge graphs, the agents

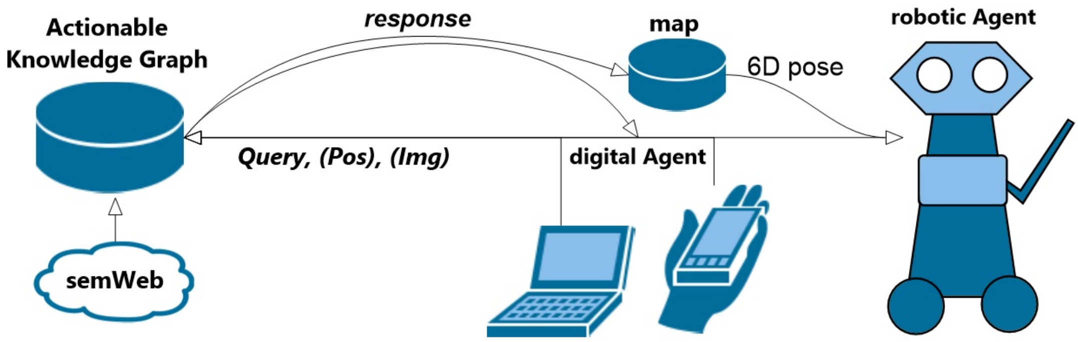

Figure 6.1: Architecture that enables access of both digital and robotic agents.

are expected to send queries based on user input and application type that are accompanied by variable input such as a perceived object, barcode, image or position information as visualised in Fig. 6.1. The query results can be textual information, images (i.e. the URIs of images stored in the actionable knowledge graph) and tabular information. For a routing application, the query would include a starting position and ask for a destination. The actionable knowledge graph would return a shelf ID, which is sufficient for digital agents to guide a user to the shelf when each shelf ID is assigned to an anchor in the routing application. The robot needs to translate the shelf ID to a 6D pose for navigation, which can easily be inferred from the map available within a SemDT. If the query is based on a perceived object like a milk carton in the fridge, the query response might be a recipe text that can be displayed in an application or label IDs that enable visualisation of different image information as well as action parameters a robot needs for action execution.

6.1.1 Robotic Agent

The robots used in this work use the cognitive architecture CRAM and the KnowRob knowledge processing framework. KnowRob can be queried in Prolog, a logic programming language. The Prolog implementation used in KnowRob also includes a SPARQL client library to easily include SPARQL query results but also offers the sparqlprog library that translates prolog queries to SPARQL. Both approaches can successfully be used to translate SPARQL query results into body modevements as will be shown in the following.

```
sparql_query('query', Row,            # call SPARQL query
[endpoint('SPARQL endpoint')]),       # from SPARQL endpoint
get_product_type(?GTIN, P),           # retrieve product to GTIN
has_type(I, P),                       # retrieve item to product
is_at(I, ['map', T, R]).              # retrieve position
```

Figure 6.2: Prolog query to retrieve the article position of a searched product.

```
PREFIX loc:  <http://purl.org/ProductKG/location#>
PREFIX tax:  <http://purl.org/ProductKG/product-taxonomy#>
PREFIX gr:  <http://purl.org/goodrelations/v1#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdf:  <http://w3.org/1999/02/22-rdf-syntax-ns#>

select ?GTIN where {
 ?product rdf:type tax:bodylotion.
 ?product gr:hasEAN_UCC-13 ?GTIN.
 ?product loc:has_flag "NEW".
 ?product loc:has_price ?price.
}  ORDER BY ?price
```

Figure 6.3: SPARQL query asking for the GTIN of the cheapest new body lotion.

Using the SPARQL Client library

Using the SPARQL client library, a regular SPARQL query can be included in a Prolog query, where Prolog will only deal with the returned result. This is a good solution when a single return value is expected and the result can be understood by the robot. When the result consists of more than one value, only the first returned value is taken and processed. Therefore, using the SPARQL client library can only be used in a limited number of applications. It is a very good solution for a routing application, where a single product or shelf is searched for.

A Prolog query to retrieve the article position of a searched product is shown in Figure 6.2. The first two lines call a SPARQL query at a specific SPARQL endpoint. The SPARQL queries used for shopping assistance return the GTIN of all products that match the consumer preferences. The returned GTIN is used as input for the next query in lines 3 to 5. Line 3 and 4 retrieve all articles with the given GTIN from the actionable knowledge graph. For these articles the corresponding position in form of `T` (Transformation) and `R` (Rotation) of the products in world coordinates are returned in line 6.

A SPARQL query asking for the GTIN of the cheapest new body lotion (a routing example) would be formulated as shown in Fig. 6.3. Following the prefix declarations in lines 1 to 4, the query searches for the GTIN of all products that are of type body lotion and are assigned a **NEW** flag. Flags like **NEW** and **SALE** are encoded in the product price tags and can automatically be detected. The Figure shows a query to retrieve the GTIN of a searched product category. In the example query the product category is **tax:body lotion**. All additional preferences can easily be queried, here the agent retrieves all instances that are assigned a **NEW** flag from the location ontology and gets the cheapest product by ordering the result set by ascending price.

Using the sparqlprog library

For robotic applications that require processing of more than one result, the sparqlprog library can be used. For this, SPARQL queries need to be translated to Prolog queries. Let us consider a simple SPARQL query such as in Figure 6.4. Here, for an input parameter `?_task` such as `SOMA:Cutting` or `cut:Halving` as listed in Table 5.3, all restrictions that satisfy the condition `<?_task rdfs subClassOf ?restriction>, ?restriction owl:onProperty cut:requiresPosition>` are retrieved. In the previous Chapter the positions for different cutting actions have been defined as `slicing_position` (which is at the end of an object) and `halving_position` (which is in the middle of an object). Thus, the returned variable *?position* will be one of the two.

This query can be translated to Prolog like shown in the Prolog module in Figure 6.5. To use the sparqlprog library as well as rdfs and owl statements in queries, some additional libraries need to be used by stating `use_module(...)`. The rdf_db library is needed to handle loading of rdf databases. Similar to calling a SPARQL query using the SPARQL client, the SPARQL endpoint needs to be defined. Prefixes also need to be declared in Prolog by using the `rdf_register_prefix(...)`

```
PREFIX cut:  <http://www.ease-crc.org/ont/food_cutting#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>

select ?position where {
 ?_task rdfs:subClassOf ?node.
 ?node owl:onProperty cut:requiresPosition.
 ?node owl:someValuesFrom ?position. }
```

Figure 6.4: SPARQL query asking for the position needed to perform a cutting task.

```
:- module(pose,                 #Define module (name = pose)
[position_to_be_used/2]).     #Define callable routine and no.  of parameters

          #Define outside modules that need to be loaded
:- use_module(library(semweb/rdf_db)),
   use_module(library(sparqlprog)),
   use_module(library(semweb/rdfs)),
   use_module(library(sparqlprog/ontologies/owl)).

          #Define SPARQL endpoint to be accessed
:- sparql_endpoint(fc, 'SPARQL endpoint').

          #Register prefixes for the query
:-rdf_register_prefix(cut, 'http://www.ease-crc.org/ont/food_cutting#')
:-rdf_register_prefix(owl, 'http://www.w3.org/2002/07/owl#')
:-rdf_register_prefix(rdfs, 'http://www.w3.org/2000/01/rdf-schema#')

          #The callable routine with input parameter Action and result parameter
          Pose that accesses the previously defined SPARQL endpoint
position_to_be_used(Action,Pose):- fc ??
 rdfs_subclass_of(Node, Action),
 rdf(Node, owl:'onProperty', cut:'requiresPosition'),
 rdf(Node, owl:'someValuesFrom', Pose).
```

Figure 6.5: Prolog module with query asking for the position needed to perform a cutting task.

statement. Finally the query asking for the cutting position to be used is defined by retrieving triples of `rdf` and `rdfs_subclass_of` statements similar to the SPARQL query in Figure 6.4.

Digital Agent Digital agents can easily access the actionable knowledge graph through its SPARQL REST API. Applications on agents will usually be built in a game engine like the Unity game development platform, which offers a library for making Web requests called UnityWebRequest[1] as well as a JSON Serialization to convert Unity objects to and from JSON format. If the game objects in the agent application are named in accordance to objects in the knowledge graph, query results can easily be processed and visualised.

Figure 6.6 shows a SPARQL query that can be used by a digital agent to retrieve all labels of a given product. If a product is perceived by the agent, the query will return the labels, which then can be visualised in the application.

[1] Unity web requests library: https://docs.unity3d.com/Manual/UnityWebRequest.html

```
PREFIX gr:  <http://purl.org/goodrelations/v1#>
PREFIX lbl:  <http://purl.org/ProductKG/label#>
PREFIX trust:  <http://purl.org/ProductKG/trust#>

select DISTINCT ?label ?lblsource where {
  ?product gr:hasEAN_UCC-13 ?_GTIN.
  ?product lbl:has_label ?label.
  ?label trust:source ?lblsource.}
```

Figure 6.6: SPARQL query asking for the labels of a given product.

App.	device	sources accessed
(i)	smart glass	ProductKG: taxonomy, ingredients, allergens, label
(ii)	smart glass	ProductKG: recipes
(iii)	smartphone	ProductKG: recipes
(iv)	smartphone	ProductKG: recipes, substitutes
(v)	smartphone	ProductKG: taxonomy, label, ingredients, allergens
(vi)	smartphone	ProductKG: taxonomy, label, ingredients, brand, location
(vii)	website	ProductKG: taxonomy, nutrition, disease, symptom, recipes, ingredients
(viii)	smart glass	Narrative enabled episodic memorys (NEEMs)
(ix)	smartphone	NEEM experience data
(x)	robot	ProductKG: taxonomy, label, ingredients, brand, location
(xi)	robot	food cutting graph

Table 6.1: Applications (App.) presented in the following with the devices used and the sources accessed.

6.2 Applications

To demonstrate the applicability of actionable knowledge graphs as one source of data for a range of consumer applications, many user assistance applications for use in daily environments were developed. In omni-channel applications with one source of data, it does not matter if the consumer accesses the information through a Web interface, a smartphone, smart glasses or via interaction with a robot. The following applications show how the same data can be accessed through various channels for different applications. The applications also show how actionable knowledge graphs can use the different techniques described in 3 to link environment, action and object information in applications with different foci.

Table 6.1 shows the devices used and sources accessed for each application.

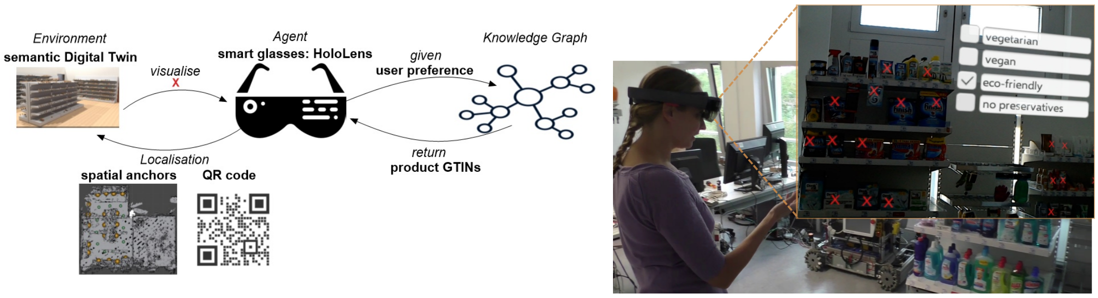

Figure 6.7: Smart Glass Shopping Assistance (Partly published in (Kümpel and Beetz, 2023)).

i) Smart Glass Shopping Assistance

The smart glass shopping assistance application visualised in Figure 6.7 highlights all objects that do not comply with a set preference. A user can specify a preference such as an unwanted ingredient, which is taken as input value to query the knowledge graph. The knowledge graph returns all product GTINs that comply with the given preference so that all products that contain the ingredient get a digital overlay, a red `X`. To localise the HoloLens in the retail laboratory I use spatial perception and match the world origin of the Unity game to the reference frame origin of the robot environment map. To load an environment map, a user has to scan a QR code that loads the semDT map including the parent Anchor. An anchor position has to be set once and can be loaded automatically in subsequent runs of an application. Product positions are inferred relative to this parent anchor using semDT information. Each product in the Unity game is created as a `X` game object named according to the knowledge graph GTINs. Thereby, objects can be set visible or invisible automatically according to the query result.

A consumer in the store will see a menu on the wall, displaying options of preferences like `vegan`, `eco-friendly` or `no alcohol`. Once a preference is chosen, all opposing products will be highlighted as depicted in Figure 6.7.

ii) Smart Glass Recipe Application

The smart glass recipe application for houshold environments, which is visualised in Figure 6.8, to the right, was developed under my supervision by Andreas Keil

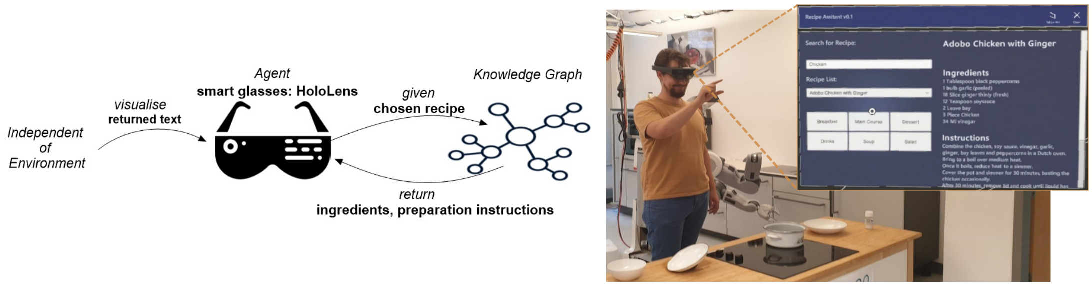

Figure 6.8: Smart Glass Recipe Assistant (Partly published in (Kümpel and Beetz, 2023)).

as part of his Bachelor Thesis *"Development of a hands-free cooking assistant based on a recipe ontology"*. For the recipe application, no semDT map is used. Instead, at application start the recipe application screen will be visible in 2 meter distance right before the user so they can see the application but also the environment around them. The screen can then easily be moved by clicking on the top right corner.

At start, the user is presented an overview screen with dropdown menu, search field and search buttons, all allowing to search for specific recipes or recipes types (such as soups, salads or main dishes). The application offers roughly 10,000 recipes that were extracted from the 1 Mio. recipe dataset (Marın et al., 2021). Although the whole dataset with more than 1 million recipes is available in owl format, its size leads to very long search and load times in the recipe application. Therefore, I decided to only use a subset of randomly chosen 10,000 recipes for the smart glass application. Once a recipe is selected, its ingredients and preparation instructions are returned from the knowledge graph and will be displayed.

iii) Smartphone Recipe Support

The smartphone recipe application visualised in Figure 6.9 recognises a product at hand and recommends recipes that can be prepared with the given product. The application was developed under my supervision by Andrew-Adair Saunders as part of his Bachelor Thesis *"Augmented Reality Linked Data Recipe Finder"*. It uses image recognition to detect product barcodes and therefore only works for packaged products that have a barcode.

If a barcode is detected, the product type (i.e. its classification) is inferred.

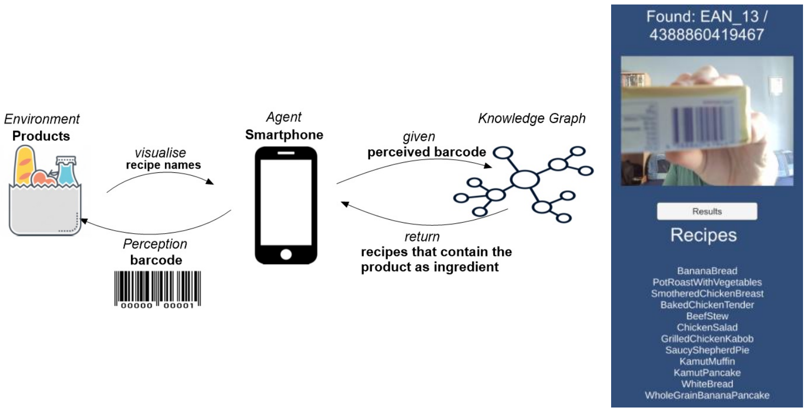

Figure 6.9: Smartphone Recipe Support (Partly published in (Kümpel and Beetz, 2023))

Product types are linked to ingredients from the FoodKG (Haussmann et al., 2019), which then link to recipes, which names are displayed in a result list. Recipes can be selected to open the linked recipe website to see preparation instructions.

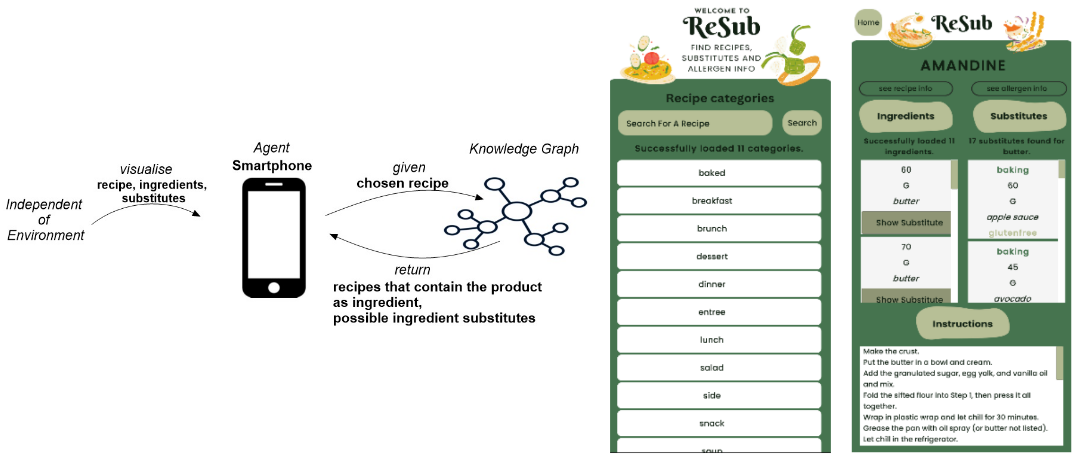

Figure 6.10: Smartphone Ingredient Substitution. (Partly published in (Kümpel and Beetz, 2023))

iv) Smartphone Ingredient Substitution

The recipe ingredient substitution application visualised in Figure 6.10 allows to search for recipes and shows possible ingredient substitutes. The work was developed under my supervision by Vanessa Niemeyer as part of her Bachelor Thesis *"A food substitute ontology with smartphone application"*. The recipe ingredient substitution application is based in the HoloLens recipe application in (ii) as well as the 10,000 recipes from the 1 Mio. recipe dataset (Marın et al., 2021) and similarly allows to search for recipes or filter recipe types. However, this application extends the functionality and integrates an ingredient substitution recommender.

The user can search for a recipe or recipe category to then choose one of the available recipes. Here, not only the needed ingredients and preparation instructions are shown, but also substitute ingredients, their amount and the purpose of substitution. The possibility and amount of substitution highly depend on substitution purpose such as baking, cooking or frying. Therefore, possible ingredient substitutes with the substitute amount for different purposes are displayed.

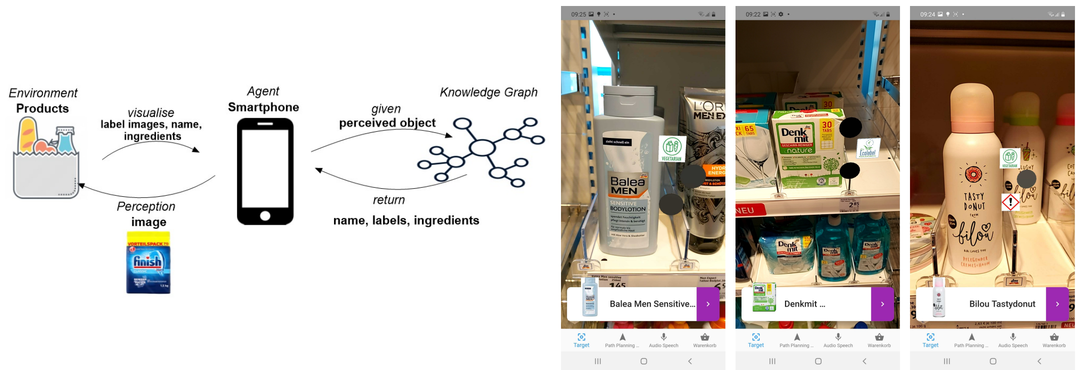

Figure 6.11: Smartphone Shopping Support. (Partly published in (Kümpel and Beetz, 2023))

v) Smartphone Shopping Support

A shopping assistant application that highlights interesting product information such as product labels or ingredients is visualised in Figure 6.11. The shopping assistant application is based on image recognition and recognises images of products in a store.

A consumer using the app can scan a product they are interested in, which will lead to the knowledge graph being queried with the recognised product GTIN to retrieve product name, labels and ingredients so that the user instantly sees interesting product information like which label the recognised product has or if it contains hazardous ingredients as well as an information slider displaying the name of the recognised product on the bottom of the screen. By clicking on or sliding of the product name, additional product information like contained ingredients and allergens are listed. Aside from that, the product can be added to the shopping cart.

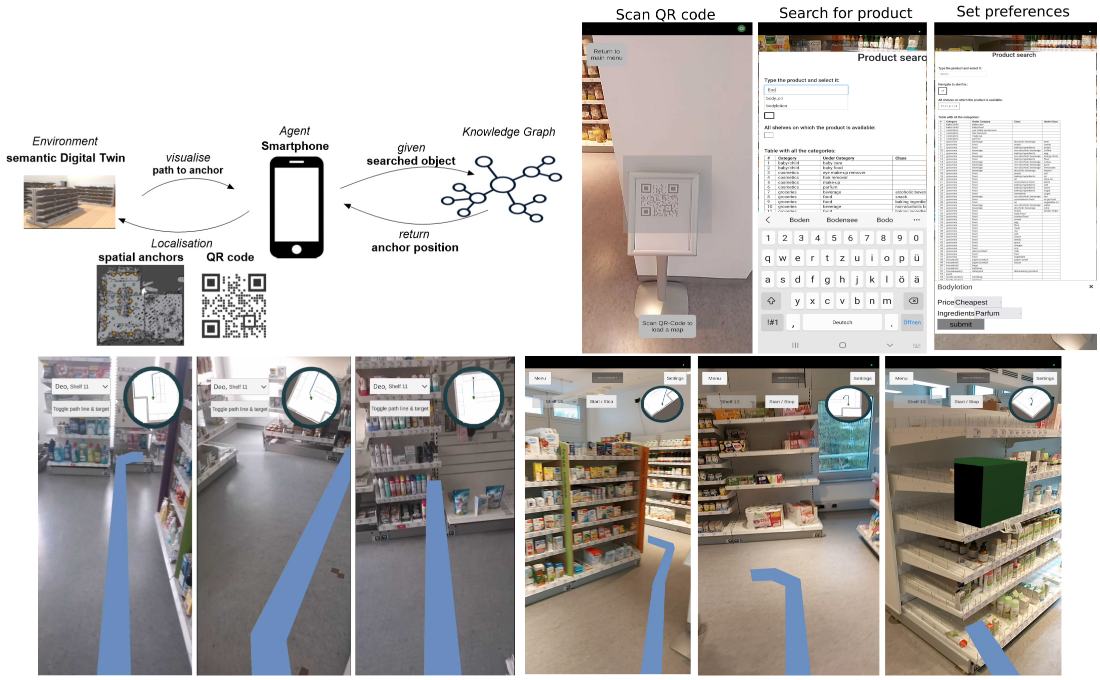

Figure 6.12: Smartphone Routing Application.

vi) Smartphone Routing Application

The routing application visualised in Figure 6.12 helps users in retail stores to find searched products. It was developed under my supervision by Toni Aleksandrov Kozarev as part of his Bachelor Thesis *"Developing a Mobile Application for Product Finding in Digital Twins of Retail Stores using Augmented Reality"*. The routing application uses the semDT map and loads a parent anchor at the reference frame origin of the robot map by scanning a QR code.

After scanning the QR code to start the application, the user can search for a product category and set preferences like cheapest price or filtering out ingredients to then start the routing to the product destination. Upon clicking the start routing button, the app will query the knowledge graph with the searched product category and retrieve the shelf position where the product category can be found. The shelf position then is translated to an anchor point in the routing application and the shortest path to the destination (which is marked by a green cube) is calculated. Using an obstacle mesh, shelves are treated as obstacles and thus avoided. The application will display the camera view and augment a path (in blue) to the product destination. The application uses the odometry sensors of the Smartphone to constantly detect changes in direction and will calculate a new path if needed. Figure 6.12 shows how a user would first start the app and search for a product in the first line. In the second line, the app is used to route to two different article locations.

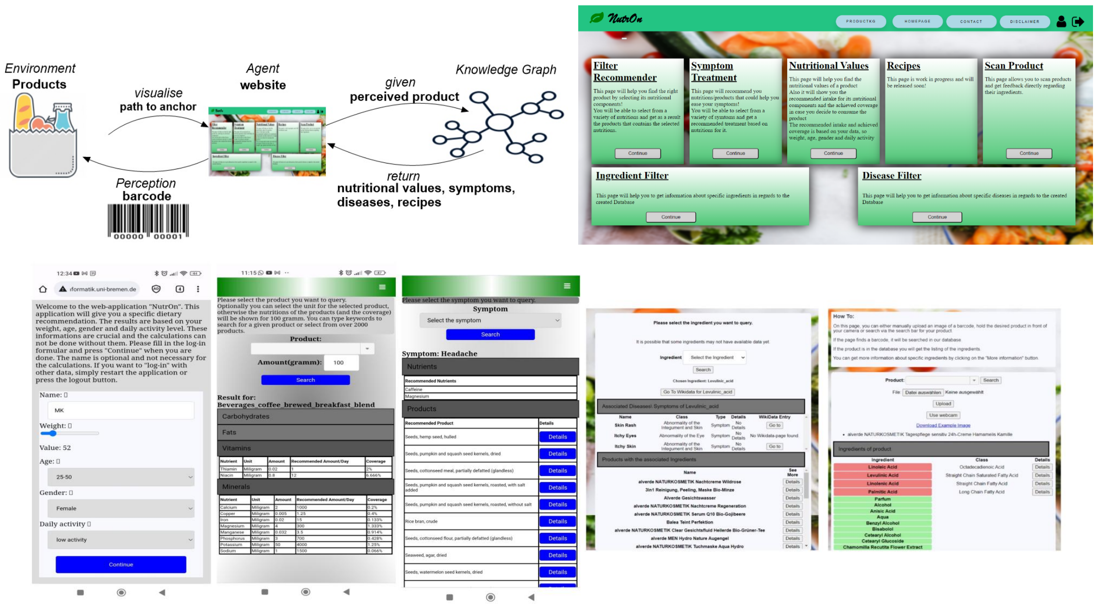

Figure 6.13: Nutrition recommender website that offers to scan a product at hand to get product information.

vii) Nutrition Recommender Website

Figure 6.13 shows an example nutrition recommender application that has different features that allow a user to scan a product barcode to get product information, filter products based on nutritional preferences, get information on how nutrients can treat symptoms of diseases as well as information on diseases and how symptoms, diseases and nutrients relate to product ingredients as well as a recipe page which integrates the data of the smartphone recipe application described previously. The website is a joint effort of my students and me and is based on developments towards Bachelor Theses from Sorin Arion and Naser Azizi *"A nutrition ontology with web interface for customer specific dietary recommendations"* and Meike Wienholt *"A chemical- and medical-based ontology for the extension of a web application for the detection of possible harmful substances in products"*. It is available at `http://productkg.informatik.uni-bremen.de/`.

The website is based on a user profile that contains recommended nutritional intake values for different user groups. A user can (but does not need to) enter personal information like weight, age, gender and daily activity level. The data is not saved, but the user is instantiated based on the given input values. In (Kümpel and Beetz, 2023), I have shown how a user can use the website to search for products that are rich in vitamin D, find nutritional values of coffee and nutrients and products that might treat a headache. Based on this, the user can proceed to see detailed product information or what recipes could be prepared with it. If openly accessible semDTs are connected to the underlying knowledge graph, users can also see where a searched product is available or compare prices.

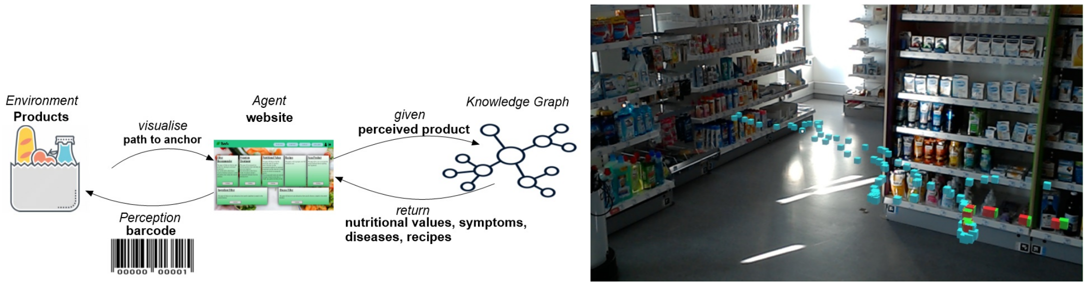

Figure 6.14: Visualisation of performed actions in a smart glass.

viii) Visualising Shopping Actions on Smart Glasses

KnowRob can be used for accessing narrative-enabled episodic memories (NEEMs) (Bozcuoglu, 2019), semantically enhanced experiments that store information about actions performed by robots or other agents, involved objects and agents using the SOMA ontology. Through the use of KnowRob and SOMA, content from actionable knowledge graphs can easily be aligned to experiment data. Thus, movements can be augmented into a given environment. This is especially useful for analysing movements, finding an optimal path or failure handling. In retail environments, experiments can be queried to augment user movements when performing shopping actions. The smart glass application depicted in Figure 6.14 was developed under my supervision by Klaus Prüger in the context of his Bachelor Thesis *"Visualization of virtual memories in Augmented Reality"*. It shows how shopping actions can be visualised in a retail environment.

The application uses the QR code visualised in Fig. 6.12 to localise the device at the frame origin of the NEEM experiments. It then queries for all actions performed in an experiment and visualises them by augmenting a box at specified time intervals of the action. Here, the AR objects of one colour belong to one action performed by a user, such as taking a specific product out of a shelf. Such a visualisation can be useful for analysing routes of customers through the store, for example.

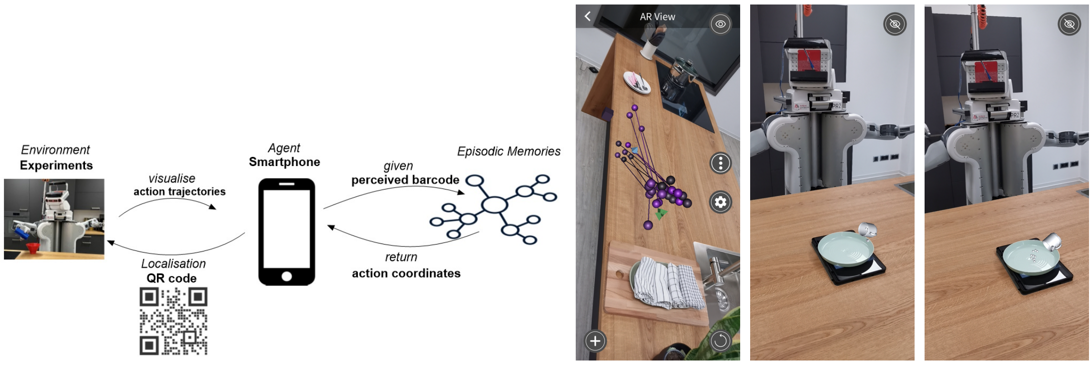

Figure 6.15: Visualisation of performed actions in a smartphone application.

ix) Visualising Robot Experiments on Smartphone

In household environments, NEEMs can be queried to augment robot movements during execution of experiments like pouring liquids or setting the table. This application also was developed under my supervision by Timo Hoheisel and Dennis Riemer in the context of their Bachelor Thesis *"LAB AR: "Development of a Mobile App for Seamless Integration and Visual Representation of Recorded Robot Experiment Data to Enhance Traceability Using AR Objects.""*. The smartphone application in Figure 6.15 can visualise experiments performed by a robot such as pouring liquids into a bowl. Since the robot experiments have different reference frame origins, different experiments need different QR codes or different placement of a QR code to show action information at the correct position.

On application start, all available NEEMs are listed so that one NEEM can be selected. Again, actions are displayed by coloured objects that are augmented onto the environment. Initially, each colour represents an object and its trajectory, but object colours can be changed manually. Object position changes during the experiment are visualised by lines between the boxes. In addition to this visualisation of experiment data, the application allows for a change of view to an animation mode, where the user can scan a QR code at a goal position to retrieve and visualise the optimal position for performing the experiment task, which can be simulated as shown in Fig. 6.15.

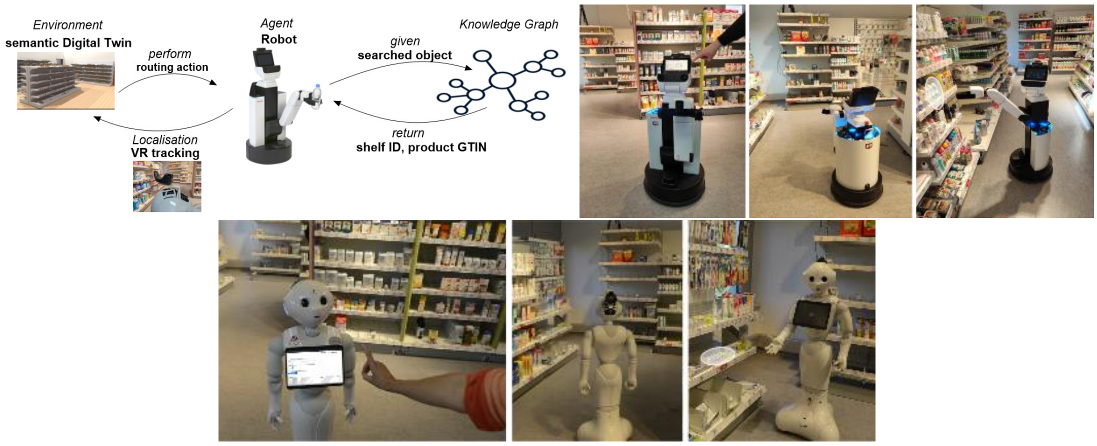

Figure 6.16: Shopping Assistants enabled by actionable knowledge graphs (Partly published in (Kümpel et al., 2023)).

x) Robot Shopping Assistant

Robots can use semDTs for shopping assistance as shown in (Kümpel et al., 2023), which is visualised in Figure 6.16. If the robots can use semDTs for shopping assistance and actionable knowledge graphs can contain semDTs, actionable knowledge graphs can just as well be used for shopping assistants. The application uses the same framework and query template as the smartphone routing application in (vi).

A user can start interaction by searching for a product category on the website displayed on the shopping robots' tablet. Additionally, preferences like price, ingredients, labels or brand can be specified to narrow the search. The robot queries the knowledge graph with the searched product category and retrieves the shelf ID and product GTINs, which can be translated to coordinates as shown in Figure 6.3 and 6.2. To optimise localisation and navigation performance, the service robots use VR tracking for localisation in the semDT map.

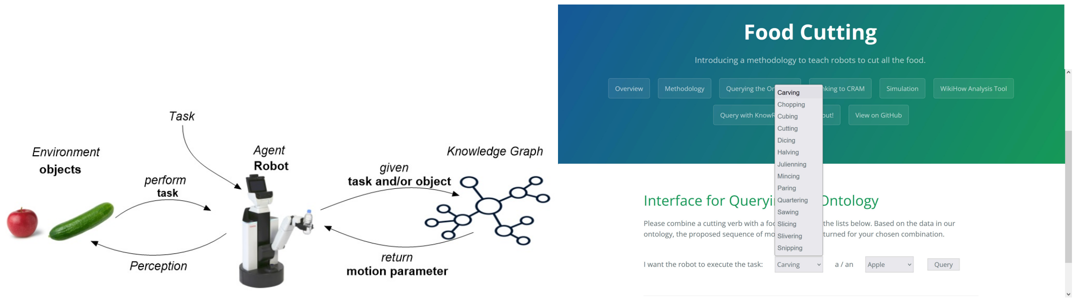

Figure 6.17: Food Cutting Robots: Web Interface to the food cutting knowledge graph where users can query for the parameters needed for task execution.

xi) Robot Performing Task Variations of Cutting

Robots that autonomously perform meal preparation tasks are one of the visions of AI. In previous work (see (Dhanabalachandran et al., 2021)) we have shown in a robot simulation how a robot can use the SOMA action ontology in combination with a failure ontology to adapt to unforeseen situations. In the example use case, the robot was able to adapt its plan in case there are two knives available and it first grasps a dull knife. The simulated robot handled the situation, recognised a failure and adapted its plan to take the other knife for cutting. The simulation is depicted in Figure 6.18.

Based on the success of that work, we extended the idea and let the simulated robot access an actionable knowledge graph, the food cutting knowledge graph, for action execution. Now, the robot is given the task to "Quarter an apple" and is given an apple and a knife in its environment. The robot queries the knowledge graph with the given task and perceived object to retrieve the motion parameters needed to perform a cutting action. We created a website that interfaces to the knowledge graph so users can try out how a robot would query the knowledge

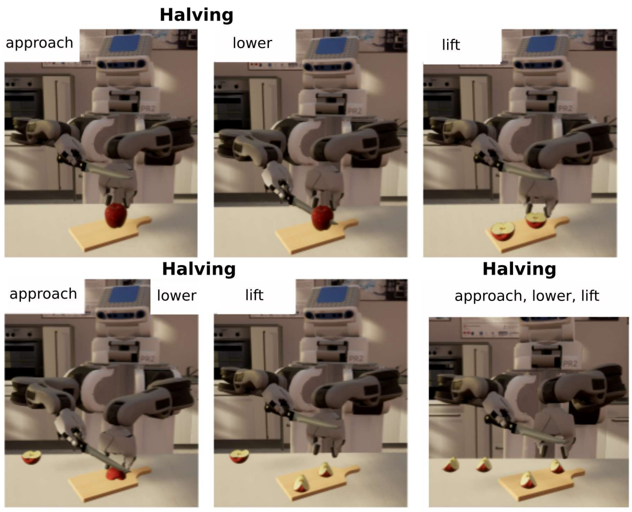

Figure 6.18: A simulated robot executing the necessary body motions needed to "Quarter an apple".

graph. It is depicted in Figure 6.17, on the right. Through the knowledge graph, the robot successfully infers the parameters needed to perform task variations of cutting actions. For quartering the apple it knows that it first has to take the knife to then **approach, lower** and **lift** the knife to first halve the apple. The robot further infers that it needs to cut both halves again for quartering, thus taking the apple halves for cutting them again by **approaching** the apple halves and **lowering** and **lifting** the knife. To make the contained knowledge and the translation of task requests into body movements understandable for users, we created the website with a web interface to the actionable knowledge graph where a user can choose between one out of the 15 available tasks and one of the 19 available food objects currently available in the knowledge graph and query for the body movements and objects needed for a robotic agent to successfully perform the selected cutting action. What is more, we created jupyter notebooks where users can query the knowledge graph for the query parameters that are taken as input for a robot simulation. The simulation is using the same frameworks as the real robot and thus can be used for testing. In Figure 6.19 the simulated robot is depicted in three example tasks:

- *cutting*: The basic parameterisation of cutting is to cut off one slice at the

Figure 6.19: Robot Simulation of performing task variations of cutting.

end of the food object.

- *halving*: For halving, the robot adapts the cutting position to be at the center of an object.

- *slicing*: Slicing requires the robot to perform multiple cutting actions in order to cut the whole object into slices.

The jupyter notebooks of the experiments can be found at: `https://moodle.intel4coro.de/course/view.php?id=8`. Here, you can check out the web interface, inspect the knowledge graph and run the simulation.

6.3 Discussion

This Chapter has shown how agents can use different interfaces and pose queries to retrieve knowledge graph content. While digital agents will usually directly access the graph database API and pose SPARQL queries, robotic agents can either pose SPARQL queries (which reduces flexibility since it has to be known when to access the graph database and pose SPARQL queries beforehand) or use either the SPARQL Client library or the sparqlprog library of Prolog to query the graph database in Prolog modules. This allows for a more flexible approach to accessing knowledge graph content.

I proceeded to show different applications of the two actionable knowledge graphs introduced in Chapter 4 and 5. While some of the applications show how knowledge graph information can be augmented into the environment on digital agents, others are based on precise environment information that enable routing applications both on robot and digital agent and even others are used for robotic action execution. The applications can be seen as a proof of concept that actionable knowledge graphs manage to link object information from the Web to current environment information as well as to action information that can be translated to robot body movements.

Chapter 7

Conclusion

In this Thesis I presented a five-step methodology for creating actionable knowledge graphs from Web information that follows existing knowledge engineering standards but also shows how to link object knowledge to environment and action knowledge to enable various user support applications in daily environments, on different agents. The methodology is exemplary applied in two scenarios with different foci to create a product knowledge graph and a food cutting knowledge graph.

The product knowledge graph shows how actionable knowledge graphs can be used for omni-channel applications in unknown environments. It therefore contains relevant product-related Web knowledge in modular ontologies that is successfully used by different agents such as smartphone, smart glass and robot for shopping assistance in a retail store. To achieve this, the agents access a standardised map created by a robot performing stocktaking in a retail store. I present a solution for representing environment information in a standardised map for digital and robotic agents and how it can be connected to object information. In this regard, semDTs of environments are created, linked to the object knowledge graph and then accessed by both digital and robotic agents to show its applicability. The product knowledge graph and its shopping applications support Hypotheses 1 that i stated.

> Agents that are enabled to locate objects that are not in the field of view can execute actions in new environments or on new objects.

The product knowledge graph with its semDT environment knowledge can be used for routing a customer to a searched product on smartphone or robot even when it is not in sight. What is more, the agents do not have to create a map of the environment themselves but access an environment map created by another agent.

The smartphone and smart glass shopping applications that highlight interesting product information based on semDT information support Hypothesis 4 that I stated.

> If object knowledge from the Web is linked to perceived objects at hand, agents can reason about the information.

By linking object to environment knowledge in the actionable knowledge graph, the agents can relate perceived objects to knowledge graph information in the shopping applications. This enables them to make sense of the perceived objects (e.g. "this is a shampoo") to then augment interesting product information into the environment and support users. They can further access connected knowledge and reason about the brand of the shampoo or if it is dermatologically tested, for example.

The food cutting knowledge graph shows how robots can access actionable knowledge graphs to execute task variations of cutting actions. I present a new approach for integrating action information in knowledge graphs and linking it to object information for task execution. This is shown for the execution of cutting task variations on different fruit, which can be queried in a web interface, and simulated. This supports Hypothesis 2 that I stated.

> Robots that are enabled to access Web information can use it for action execution.

However, this is only true for simple tasks like shown in the robot shopping assistance application. Here, the robot performs the simple task of navigating to a queried location, which it retrieves from the actionable knowledge graph. For more complex actions like meal preparation actions that require a sequence of tasks and body movements to be executed, the Web knowledge needs to further be translatable to body movements of the robot.

Thus, I use actionable knowledge graphs to parameterise general action plans to translate the knowledge of the actionable knowledge graph into body movements of a robot, which supports Hypothesis 5 that I stated.

> A robot can use an actionable knowledge graph to translate the contained knowledge into body movements.

The acquired Web knowledge in the food cutting knowledge graph can be used to execute 14 distinct cutting tasks on 19 different fruits and vegetables.

For the food cutting knowledge graph, the importance of creating action groups that lead to similar robot movements is highlighted. It is further analysed how object information can influence action execution, how the needed knowledge can be acquired from the Web and how it can be modelled in a knowledge graph in such a way that a robot can use it to parameterise and execute different cutting tasks. This supports Hypothesis 3 that I stated.

> A robot can use an actionable knowledge graph to understand task variations as well as the role of tools and objects in a given task.

Using the food cutting knowledge graph, the robot understands task variations in such a way that it knows the specific body movement that is needed to successfully perform the task variation and reach the desired result. It also knows how specific object features such as a peel of an orange change the action execution, namely to first peel the orange before cutting it. It also knows what tools can be used to perform the given task.

The methodology is validated by showcasing various applications that are enabled by the two exemplary knowledge graphs. The applications have different foci and range from smartphone applications for shopping assistance that highlight interesting product features or route to a searched product over smart glass applications like shopping assistance and a recipe application to robot applications for shopping assistance and execution of cutting task variations on different fruits and vegetables.

The presented methodology can be used to create actionable knowledge graphs but can be seen as a guideline and not as a complete toolbox. The proposed tools, techniques and software are examples that can be applied for different domains and applications but as new tools and techniques are constantly being developed, the list is not exhaustive. Instead, I focus on a detailed description of the knowledge acquisition, processing and linking methods needed to create actionable knowledge graphs and show how it can be applied for two example knowledge graphs with different focus: a product knowledge graph for user applications like recipe recommender and product finder, as well as a food cutting knowledge graph for robot applications and teaching robots new task variations of cutting.

I hope that this Thesis sparks interest in the creation of actionable knowledge graphs and that it will be used by researchers to extend the methodology but also to create more actionable knowledge graphs and especially enable more agent applications using Web knowledge.

Appendix **A**

Prior Publications

M. Kümpel, V. Hassouna, J.-P. Töberg, P. Cimiano and M. Beetz. **Cut, Chop, Slice or Dice: Parameterising General Action Plans Using Knowledge Graphs**. In *International Conference on Intelligent Robots and Systems (IROS 2024)*. submitted.

M. Kümpel, J.-P. Töberg, V. Hassouna, M. Beetz and P. Cimiano. **Towards a Knowledge Engineering Methodology for Flexible Robot Manipulation in Everyday Tasks**. In *Workshop on Actionable Knowledge Representation and Reasoning for Robots (AKR^3) at Extended Semantic Web Conference (ESWC 2024)*.

M. Kümpel and M. Beetz. **ProductKG: A Product Knowledge Graph for User Assistance in Daily Activities**. In *Ontology Showcase and Demonstrations Track, 9th Joint Ontology Workshops (JOWO 2023), co-located with FOIS 2023,* 19-20 July, 2023, Sherbrooke, Québec, Canada, 2023.

Kümpel M, Dech J, Hawkin A, Beetz M. **Robotic Shopping Assistance for Everyone: Dynamic Query Generation on a Semantic Digital Twin as a Basis for Autonomous Shopping Assistance**. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* 2023 May 30 (pp. 2523-2525).

Kümpel, M., Dech, J., Hawkin, A. and Beetz, Michael, **Evaluation of Autonomous Shopping Assistants Using Semantic Digital Twin Stores**, In *AIC'23: 9th workshop on Artificial Intelligence and Cognition*, 2023.

Kümpel M, Mueller CA, Beetz M. **Semantic Digital Dwins for Retail Logistics**. In *Dynamics in Logistics: Twenty-Five Years of Interdisciplinary Logistics Research in Bremen, Germany* 2021 Sep 22 (pp. 129-153). Cham: Springer International Publishing.

Kümpel, M., Buchinger, M., Sellami, M., Balta, D. and Beetz, Michael, **Trust, But Verify: Towards Trustworthiness in Digital Assistants Based on Verifiable Claims in Knowledge Graphs**, In *AIC'23: 9th workshop on Artificial Intelligence and Cognition*, 2023.

Krieg-Brückner, Bernd, Nolte, Mark Robin, Pomarlan, Mihai and Kümpel, Michaela, **The Downgrading Axioms Challenge for Qualitative Composition of Food Ingredients**, In *SemREC 2022, 2nd Semantic Reasoning Evaluation Challenge 2022*, vol. 3337, pp. 6-15, 2022.

Michaela Kümpel and Michael Beetz, **Realizing Trustworthiness in Linked Data Applications Based on Individual Data Source Trust Assessment. Extended Abstract**, In *JOWO* 2021.

Kümpel, M., de Groot, A., Tiddi, I. and Beetz, M., 2020, October. **Using Linked Data to Help Robots Understand Product-related Actions**. In *JOWO 2020, The Joint Ontology Workshops*, vol. 2708, 2020.

Appendix B

Supervised Student Works

2.1 Student Works that Contributed to this Thesis

The following student works resulted in ontologies that are part of the Knowledge Graphs presented in this work:

- Bachelor Thesis of Daniel Schmidt: **Standardisierung eines im Internet gescrapten Rezeptsatzes in eine Ontologie** (Standardizing a web-scraped recipe dataset into an ontology), 2023
- Bachelor Thesis of Meike Wienholt: **Eine chemisch- und medizinisch-basierte Ontologie zur Erweiterung einer Web-Anwendung für die Erkennung von möglichen Gefahrstoffen in Produkten** (A chemical- and medical-based ontology for the extension of a web application for the detection of possible harmful substances in products.), 2023
- Bachelor Thesis of Vanessa Niemeyer: **Eine Ontologie für Lebensmittelersatzprodukte mit Smartphone-Anwendung** (A food substitute ontology with smartphone application), 2023
- Bachelor Thesis of Karin Doberstein: **Eine Lebensmittelallergene Ontologie zur Erweiterung einer Mobilgeräte-Applikation mit Zweck der Erkennung enthaltener Allergene in Produkten** (A food allergen ontology to extend a mobile application for detecting allergens in products), 2022

- Bachelor Thesis of Naser Azizi and Sorin Arion: **Eine Nährwert Ontologie mit Web-Anwendungen für benutzerspezifische empfohlene Nährwertzufuhr** (A nutrition ontology with web interface for customer specific dietary recommendations), 2022

The following student works were also supervised by me and contributed towards this Thesis:

- Bachelor Thesis of Timo Hoheisel and Dennis Riemer: **LAB AR: "Entwicklung einer mobilen App für die nahtlose Integration und visuelle Darstellung aufgezeichneter Roboter-Experimentdaten zur Verbesserung der Nachvollziehbarkeit mittels AR-Objekten"** (LAB AR: "Development of a Mobile App for Seamless Integration and Visual Representation of Recorded Robot Experiment Data to Enhance Traceability Using AR Objects."), 2023

- Bachelor Thesis of Fabian Jänecke: **Produktinformationen in einer Kunden-Assistenzanwendung basierend auf Ontologien** (Product information in a customer assistance application based on ontologies), 2023

- Bachelor Thesis of Andreas Keil: **Entwicklung eines hands-free cooking assistant auf Grundlage einer Rezeptontologie** (Development of a hands-free cooking assistant based on a recipe ontology), 2023

- Bachelor Thesis of Fabian Rosenstock: **Comparing the Usability of Internet and Web of Things Ontologies for Different Everyday Scenarios**, 2023

- Bachelor Thesis of Henri Bührmann: **Entwurf und Entwicklung eines Frameworks zur automatischen Klassifizierung von Produktstammdaten des Einzelhandels aus unterschiedlichen Datenquellen**, 2023

- Bachelor Thesis of Toni Aleksandrov Kozarev: **Developing a Mobile Application for Product Finding in Digital Twins of Retail Stores using Augmented Reality**, 2022

- Bachelor Thesis of Andrew-Adair Saunders: **Augmented Reality Linked Data Recipe Finder**, 2022

- Bachelor Thesis of Nazmi Ahmad: **Generierung von Episodic Memories in einem Virtual-Reality-Modell** (Generation of Episodic Memories in a Virtual-Reality-Model), 2022
- Bachelor Thesis of Callum Jänicke: **The Wikidata Use Case: Analyzing the usability of Wikidata Knowledge for Software Agents**, 2022
- Bachelor Thesis of Jan-Andree Pangritz: **Erzeugung einer Ontologie für eine Anwendung in der Trennung von Produktverpackungen** (Ontology creation and application for package separation), 2022
- Bachelor Thesis of Klaus Prüger: **Visualisierung von virtuellen Gedächtnissen in Augmented Reality** (Visualization of virtual memories in Augmented Reality), 2022
- Master Thesis of Jonas Dech: **A Knowledge Based Shopping Assistant in a Retail Environment using VR Tracking**, 2022
- Bachelor Thesis of Luca Barre: **Ontologie-basierte Sprachauswahl eines AR-Shopping Assistenten** (Ontology-based speech selection in Augmented Reality Shopping Assistance), 2021
- Bachelor Thesis of Maik Daniel Klause: **Evaluation of Linked Data in Cognitive Robot Control for kitchen scenarios**, 2021
- Master Thesis of Anna de Groot (collaboration with VU Amsterdam): **Knowledge Graph Enabled Virtual Reality in the Kitchen Domain**, 2020

References

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, 2000.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, August 2022.

Aliakbar Akbari, Fabien Lagriffoul, and Jan Rosell. Combined heuristic task and motion planning for bi-manual robots. *Autonomous robots*, 43(6):1575–1590, 2019.

Oliver Allweyer, Christian Schorr, Andreas Mohr, and Rolf Krieger. Product classification using partially abbreviated product names, brands and dimensions. In *Data Management Technologies and Applications: 9th International Conference, DATA 2020, Virtual Event, July 7–9, 2020, Revised Selected Papers 9*, pages 223–244. Springer, 2021.

Jesús Omar Álvarez Márquez and Jürgen Ziegler. Improving the shopping experience with an augmented reality-enhanced shelf. *Mensch und Computer 2017-Workshopband*, 2017.

Liliana Andrés-Hernández, Abdul Baten, Razlin Azman Halimi, Ramona Walls, and Graham J King. Knowledge representation and data sharing to unlock crop variation for nutritional food security. *Crop Science*, 60(2):516–529, 2020.

Renzo Angles and Claudio Gutierrez. The expressive power of sparql. In *International Semantic Web Conference*, pages 114–129. Springer, 2008.

Peter Augustine. The industry use cases for the digital twin idea. In *Advances in Computers*, volume 117, pages 79–105. Elsevier, 2020.

Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *Introduction to description logic*. Cambridge University Press, 2017.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

Ferenc Balint-Benczedi, Patrick Mania, and Michael Beetz. Scaling perception towards autonomous object manipulation — in knowledge lies the power. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7487801`.

Michele Banko and Robert C Moore. Part-of-speech tagging in context. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 556–561, 2004.

Srividya K Bansal. Towards a semantic extract-transform-load (etl) framework for big data integration. In *2014 IEEE International Congress on Big Data*, pages 522–529. IEEE, 2014.

Georg Bartels, Daniel Beßler, and Michael Beetz. Episodic memories for safety-aware robots: Knowledge representation and reasoning for robots that safely interact with human co-workers. *KI-Künstliche Intelligenz*, 33:123–130, 2019.

Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In Ren C. Luo and Hajime Asama, editors, *Proceedings of the 2nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 1012–1017, Taipei, Taiwan, 2010. IEEE. ISBN 978-1-4244-6674-0. doi: 10.1109/IROS.2010.5650146.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic Roommates Making Pancakes. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 529–536, Bled, Slovenia, October 2011a. IEEE. ISBN 978-1-61284-866-2 978-1-61284-868-6. doi: 10.1109/Humanoids.2011.6100855.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 529–536. IEEE, 2011b.

Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoglu, and Georg Bartels. KnowRob 2.0 - A 2nd Generation Knowledge Processing Framework for Cognition-enabled Robotic Agents. In Alex Zelinsky and Frank Park, editors, *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pages 512–519, Brisbane, Australia, 2018a. IEEE. ISBN 978-1-5386-3081-5. doi: 10.1109/ICRA.2018.8460964.

Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu, and Georg Bartels. Knowrob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2018b.

Michael Beetz, Simon Stelter, Daniel Beßler, Kaviya Dhanabalachandran, Michael Neumann, Patrick Mania, and Andrei Haidu. *Robots Collecting Data: Modelling Stores*, pages 41–64. Springer International Publishing, Cham, 2022. ISBN 978-3-031-06078-6. doi: 10.1007/978-3-031-06078-6_2.

Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

Daniel Beßler, Robert Porzel, Mihai Pomarlan, Michael Beetz, Rainer Malaka, and John Bateman. A Formal Model of Affordances for Flexible Robotic Task Execution. In *Frontiers in Artificial Intelligence and Applications - Volume 325: ECAI 2020*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2425–2432, Santiago de Compostela, Spain, 2020a. IOS Press. doi: 10.3233/FAIA200374.

Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the socio-physical model of activities (soma) for autonomous robotic agents. *arXiv preprint arXiv:2011.11972*, 2020b.

Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents. In *Formal Ontology in Information Systems*, volume 344 of *Frontiers in Artificial Intelligence and Applications*, pages 159–174. IOS Press, 2022. URL https://ebooks.iospress.nl/doi/10.3233/FAIA210379.

Snehal Bhoir, Tushar Ghorpade, and Vanita Mane. Comparative analysis of different word embedding models. In *2017 International conference on advances in computing, communication and Control (ICAC3)*, pages 1–4. IEEE, 2017.

Chris Bizer, Richard Cyganiak, Tom Heath, et al. How to publish linked data on the web. 2007.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global, 2011.

GO Blog. Introducing the knowledge graph: thing, not strings. *Introducing the Knowledge Graph: things, not strings*, 2012.

Marc H. Bornstein and James J. Gibson. The Ecological Approach to Visual Perception. *The Journal of Aesthetics and Art Criticism*, 39(2):203, 1980. ISSN 00218529. doi: 10.2307/429816.

Willem Nico Borst. Construction of engineering ontologies for knowledge sharing and reuse. 1999.

Maged N Kamel Boulos, Abdulslam Yassine, Shervin Shirmohammadi, Chakkrit Snae Namahoot, and Michael Brückner. Towards an "internet of food": food ontologies for the internet of things. *Future Internet*, 7(4):372–392, 2015.

Asil Kaan Bozcuoglu. *Fast Robot Learning using Prospection and Experimental Knowledge: A Cognitive Approach with Narrative-Enabled Episodic Memories and Symbolic Knowledge*. PhD thesis, Universität Bremen, 2019.

Max A Bramer. *Logic programming with Prolog*, volume 9. Springer, 2005.

Beate Brenner and Vera Hummel. Digital twin as enabler for an innovative digital shopfloor management system in the esb logistics learning factory at reutlingen-university. *Procedia Manufacturing*, 9:198–205, 2017.

Sergey Brin. Extracting patterns and relations from the world wide web. In *International workshop on the world wide web and databases*, pages 172–183. Springer, 1998.

Riccardo Caccavale, Pierluigi Arpenti, Gianmarco Paduano, Andrea Fontanellli, Vincenzo Lippiello, Luigi Villani, and Bruno Siciliano. A flexible robotic depalletizing system for supermarket logistics. *IEEE Robotics and Automation Letters*, 5(3):4471–4476, 2020.

Jaime Cantais, David Dominguez, Valeria Gigante, Loredana Laera, and Valentina Tamma. An example of food ontology for diabetes control. In *Proceedings of the International Semantic Web Conference 2005 workshop on Ontology Patterns for the Semantic Web*, pages 1–9, 2005.

Alberto Cavallo, Marco Costanzo, Giuseppe De Maria, Ciro Natale, Salvatore Pirozzi, Simon Stelter, Gayane Kazhoyan, Sebastian Koralewski, and Michael

Beetz. Robotic Clerks: Autonomous Shelf Refilling. In Luigi Villani, Ciro Natale, Michael Beetz, and Bruno Siciliano, editors, *Robotics for Intralogistics in Supermarkets and Retail Stores*, volume 148, pages 137–170. Springer International Publishing, Cham, 2022. ISBN 978-3-031-06077-9 978-3-031-06078-6. doi: 10.1007/978-3-031-06078-6_6.

Duygu Çelik et al. Foodwiki: Ontology-driven mobile safe food consumption system. *The scientific World journal*, 2015, 2015.

Christian Charras and Thierry Lecroq. Handbook of exact string matching algorithms. 2004.

Yingfeng Chen, Feng Wu, Wei Shuai, and Xiaoping Chen. Robots serve humans in public places—kejia robot as a shopping assistant. *International Journal of Advanced Robotic Systems*, 14(3):1729881417703569, 2017.

Gene Ontology Consortium. The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1):D330–D338, 2019.

World Wide Web Consortium et al. R2rml: Rdb to rdf mapping language. 2012.

Marco Costanzo, Giuseppe De Maria, Gaetano Lettera, and Ciro Natale. Can robots refill a supermarket shelf?: Motion planning and grasp control. *IEEE Robotics & Automation Magazine*, 28(2):61–73, 2021.

Richard Crang, Sheila Lyons-Sobaski, and Robert Wise. Fruits, Seeds, and Seedlings. In *Plant Anatomy : A Concept-Based Approach to the Structure of Seed Plants*, pages 649–678. Springer International Publishing Imprint: Springer, Cham, 2018.

Zachary Davis, Michael Hu, Shreyas Prasad, Michael Schuricht, PM Melliar-Smith, and Louise E Moser. A personal handheld multi-modal shopping assistant. In *International conference on Networking and Services (ICNS'06)*, pages 117–117. IEEE, 2006.

Hasan Davulcu, Sukumar Koduri, and Saravanakumar Nagarajan. Datarover: a taxonomy based crawler for automated data extraction from data-intensive websites. In *Proceedings of the 5th ACM international workshop on Web information and data management*, pages 9–14, 2003a.

Hasan Davulcu, S. Vadrevu, S. Nagarajan, and Iv Ramakrishnan. Ontominer: Bootstrapping and populating ontologies from domain-specific web sites. *Intelligent Systems, IEEE*, 18:24 – 33, 10 2003b. doi: 10.1109/MIS.2003.1234766.

Kaviya Dhanabalachandran, Vanessa Hassouna, Maria M. Hedblom, Michaela Kümpel, Nils Leusmann, and Michael Beetz. Cutting Events: Towards Autonomous Plan Adaption by Robotic Agents through Image-Schematic Event Segmentation. In *Proceedings of the 11th on Knowledge Capture Conference*, K-CAP '21, pages 25–32, New York, NY, USA, December 2021. Association for Computing Machinery. ISBN 978-1-4503-8457-5. doi: 10.1145/3460210.3493585.

Mohammed Diab, Aliakbar Akbari, Muhayy Ud Din, and Jan Rosell. PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation. *Sensors*, 19(5):1166, March 2019. ISSN 1424-8220. doi: 10.3390/s19051166.

Mohammed Diab, Mihai Pomarlan, Daniel Beßler, Aliakbar Akbari, Jan Rosell, John Bateman, and Michael Beetz. An Ontology for Failure Interpretation in Automated Planning and Execution. In Manuel F. Silva, José Luís Lima, Luís Paulo Reis, Alberto Sanfeliu, and Danilo Tardioli, editors, *Robot 2019: Fourth Iberian Robotics Conference*, volume 1092, pages 381–390. Springer International Publishing, Cham, 2020. ISBN 978-3-030-35989-8 978-3-030-35990-4. doi: 10.1007/978-3-030-35990-4_31.

Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow*, 1184, 2014.

Nicola Doering, Sandra Poeschl, Horst-Michael Gross, Andreas Bley, Christian Martin, and Hans-Joachim Boehme. User-centered design and evaluation of a mobile shopping robot. *International Journal of Social Robotics*, 7(2):203–225, 2015.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

Damion M. Dooley, Emma J. Griffiths, Gurinder S. Gosal, Pier L. Buttigieg, Robert Hoehndorf, Matthew C. Lange, Lynn M. Schriml, Fiona S. L. Brinkman, and William W. L. Hsiao. FoodOn: A harmonized food ontology to increase global food traceability, quality control and data integration. *npj Sci Food*, 2(1):23, December 2018a. ISSN 2396-8370. doi: 10.1038/s41538-018-0032-6.

Damion M Dooley, Emma J Griffiths, Gurinder S Gosal, Pier L Buttigieg, Robert Hoehndorf, Matthew C Lange, Lynn M Schriml, Fiona SL Brinkman, and William WL Hsiao. Foodon: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2 (1):1–10, 2018b.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2, 2016.

Epicurious. How To Slice Every Fruit — Method Mastery — Epicurious (Youtube), November 2019. URL `https://youtu.be/VjINuQX4hbM`.

Vanesa Espín, María V Hurtado, and Manuel Noguera. Nutrition for elder care: a nutritional semantic recommender system for the elderly. *Expert Systems*, 33 (2):201–210, 2016.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008a. ISSN 0001-0782. doi: 10.1145/1409360.1409378.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12): 68–74, 2008b.

Dieter Fensel and Christoph Bussler. Semantic web enabled web services. In *Advances in artificial intelligence: 25th Annual German Conference on AI*, volume 2479, page 316. Springer, 2002.

Javier D Fernández, Wouter Beek, Miguel A Martínez-Prieto, and Mario Arias. Lod-a-lot: A queryable dump of the lod cloud. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 75–83. Springer, 2017.

Sébastien Ferré. Sparklis: An expressive query builder for sparql endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418, 2017.

Chengcheng Fu, Zhisheng Huang, Frank van Harmelen, Tingting He, and Xingpeng Jiang. Food4healthkg: Knowledge graphs for food recommendations based on gut microbiota and mental health. *Artificial Intelligence in Medicine*, 145:102677, 2023. ISSN 0933-3657. doi: https://doi.org/10.1016/j.artmed.2023.102677. URL https://www.sciencedirect.com/science/article/pii/S0933365723001914.

Michael R Genesereth and Nils J Nilsson. Logical foundations of. *Artificial Intelligence. New York: Morgan Kaufmann Publishers*, 1987.

Yolanda Gil and Donovan Artz. Towards content trust of web resources. *Journal of Web Semantics*, 5(4):227–239, 2007.

Paulo J.S. Gonçalves and Pedro M.B. Torres. Knowledge representation applied to robotic orthopedic surgery. *Robotics and Computer-Integrated Manufacturing*, 33:90–99, June 2015. ISSN 07365845. doi: 10.1016/j.rcim.2014.08.014.

Michael Grieves. *Virtually perfect: driving innovative and lean products through product lifecycle management*. Space Coast Press, 2011.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

H-M Gross, H-J Boehme, Christof Schröter, Steffen Müller, Alexander König, Ch Martin, Matthias Merten, and Andreas Bley. Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 3471–3478. IEEE, 2008.

Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

Nicola Guarino. *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. IOS press, 1998.

Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? *Handbook on ontologies*, pages 1–17, 2009.

Andrei Haidu, Daniel Beßler, Asil Kaan Bozcuoglu, and Michael Beetz. Knowrob-sim — game engine-enabled knowledge processing for cognition-enabled robot control. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE. doi: 10.1109/IROS.2018.8593935.

Wu Hao, Jiao Menglin, Tian Guohui, Ma Qing, and Liu Guoliang. R-kg: a novel method for implementing a robot intelligent service. *AI*, 1(1):6, 2020.

J Hardi. Cellfie plugin, 2018.

Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. Foodkg: a semantics-driven knowledge graph for food recommendation. In *International Semantic Web Conference*, pages 146–162. Springer, 2019.

Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. Knowledge graphs on the web–an overview. *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, pages 3–22, 2020.

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web: Grundlagen*. Springer-Verlag, 2007.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial intelligence*, 194:28–61, 2013.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

Pankaj Jaiswal, Shulamit Avraham, Katica Ilic, Elizabeth A Kellogg, Susan McCouch, Anuradha Pujar, Leonore Reiser, Seung Y Rhee, Martin M Sachs, Mary Schaeffer, et al. Plant ontology (po): a controlled vocabulary of plant structures and growth stages. *Comparative and functional genomics*, 6(7-8): 388–397, 2005.

Andreas Jakl, Lucas Schöffer, Matthias Husinsky, and Markus Wagner. Augmented reality for industry 4.0: Architecture and user experience. In *FMT*, pages 38–42, 2018.

Gayane Kazhoyan and Michael Beetz. Executing Underspecified Actions in Real World Based on Online Projection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5156–5163, November 2019. doi: 10.1109/IROS40897.2019.8967867.

Gayane Kazhoyan, Simon Stelter, Franklin Kenghagho Kenfack, Sebastian Koralewski, and Michael Beetz. The Robot Household Marathon Experiment. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9382–9388, May 2021. doi: 10.1109/ICRA48506.2021.9560774.

Hamidreza Kobdani and Hinrich Schütze. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 92–95, 2010.

Sebastian Koralewski, Gayane Kazhoyan, and Michael Beetz. Self-specialization of general robot plans based on experience. *IEEE Robotics and Automation Letters*, 4(4):3766–3773, 2019.

E Kostylev, J Reutter, and Martín Ugarte. Construct queries in sparql. In *18th International Conference on Database Theory (ICDT 2015)*, 2015.

Bernd Krieg-Brückner, Serge Autexier, and Mihai Pomarlan. The baall ontology-configuration of service robots, food, and diet. In *JOWO*, 2021.

Michaela Kümpel, Anna de Groot, Ilaria Tiddi, and Michael Beetz. Using linked data to help robots understand product-related actions. In *JOWO*, 2020.

Michaela Kümpel and Michael Beetz. Productkg: A product knowledge graph for user assistance in daily activities. In *Ontology Showcase and Demonstrations Track, 9th Joint Ontology Workshops (JOWO 2023), co-located with FOIS 2023, 19-20 July, 2023, Sherbrooke, Québec, Canada.*, 2023.

Michaela Kümpel, Christian A. Mueller, and Michael Beetz. Semantic digital twins for retail logistics. In Michael Freitag, Herbert Kotzab, and Nicole Megow, editors, *Dynamics in Logistics: Twenty-Five Years of Interdisciplinary Logistics Research in Bremen, Germany*, pages 129–153. Springer International Publishing, Cham, 2021. ISBN 978-3-030-88662-2. doi: 10.1007/978-3-030-88662-2_7.

Michaela Kümpel, Jonas Dech, Alina Hawkin, and Michael Beetz. Robotic shopping assistance for everyone: Dynamic query generation on a semantic digital twin as a basis for autonomous shopping assistance. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*, pages 2523–2525, London, United Kingdom, 2023. doi: 10.5555/3545946.3598989.

Michaela Kümpel, Vanessa Hassouna, Jan-Phillip Töberg, Phillip Cimiano, and Michael Beetz. Cut, chop, slice or dice: Teaching robots new task variations using web knowledge. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2024. submitted.

John E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2019. ISBN 978-0-262-53853-4.

Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017.

Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. 2016.

Taehee Lee, Ig-hoon Lee, Suekyung Lee, Sang-goo Lee, Dongkyu Kim, Jonghoon Chun, Hyunja Lee, and Junho Shim. Building an operational product ontology system. *Electronic Commerce Research and Applications*, 5(1):16–28, 2006.

Séverin Lemaignan, Raquel Ros, Lorenz Mösenlechner, Rachid Alami, and Michael Beetz. ORO, a knowledge management platform for cognitive architectures in robotics. In Ren C. Luo and Hajime Asama, editors, *Proceedings of the 2nd IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553, Taipei, Taiwan, 2010. IEEE. ISBN 978-1-4244-6674-0. doi: 10.1109/IROS.2010.5649547.

Thomas Lin, Oren Etzioni, et al. Entity linking at web scale. In *Proceedings of the joint workshop on automatic knowledge base construction and web-scale knowledge extraction (AKBC-WEKEX)*, pages 84–88, 2012.

Gheorghe Lisca, Daniel Nyga, Ferenc Balint-Benczedi, Hagen Langer, and Michael Beetz. Towards Robots Conducting Chemical Experiments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5202–5208, Hamburg, Germany, September 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7354110.

Will Lockie. Delivering an effective click-and-collect strategy: A retailer case study. *Journal of Digital & Social Media Marketing*, 2(2):139–152, 2014.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

Javier Marın, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):187–203, 2021.

Viviana Mascardi, Valentina Cordì, Paolo Rosso, et al. A comparison of upper ontologies. In *Woa*, volume 2007, pages 55–64, 2007.

Albert Meroño-Peñuela and Rinke Hoekstra. grlc Makes GitHub Taste Like Linked Data APIs. In *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016*, pages 342–353. Springer, 2016. ISBN 978-3-319-47602-5. doi: 10.1007/978-3-319-47602-5_48.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Alex Mitrevski, Paul G. Plöger, and Gerhard Lakemeyer. Ontology-Assisted Generalisation of Robot Action Execution Knowledge. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6763–6770, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636791.

Daniel Mora, Robert Zimmermann, Douglas Cirqueira, Marija Bezbradica, Markus Helfert, Andreas Auinger, and Dirk Werth. Who wants to use an augmented reality shopping assistant application? 2020.

Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)*, 54(1):1–39, 2021.

Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems -*

FOIS '01, volume 2001, pages 2–9, Ogunquit, Maine, USA, 2001. ACM Press. ISBN 978-1-58113-377-6. doi: 10.1145/505168.505170.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done. *Queue*, 17(2):48–75, 2019.

Daniel Nyga, Subhro Roy, Rohan Paul, Daehyung Park, Mihai Pomarlan, Michael Beetz, and Nicholas Roy. Grounding Robot Plans from Natural Language Instructions with Incomplete World Knowledge. In *Proceedings of The 2nd Conference on Robot Learning*, pages 714–723. PMLR, 2018. URL https://proceedings.mlr.press/v87/nyga18a.html.

Alberto Olivares-Alarcos, Daniel Beßler, Alaa Khamis, Paulo Goncalves, Maki K. Habib, Julita Bermejo-Alonso, Marcos Barreto, Mohammed Diab, Jan Rosell, João Quintas, Joanna Olszewska, Hirenkumar Nakawala, Edison Pignaton, Amelie Gyrard, Stefano Borgo, Guillem Alenyà, Michael Beetz, and Howard Li. A Review and Comparison of Ontology-based Approaches to Robot Autonomy. *The Knowledge Engineering Review*, 34(29):1–38, 2019. ISSN 0269-8889. doi: 10.1017/S0269888919000237.

Alberto Olivares-Alarcos, Daniel Beßler, Alaa Khamis, Paulo Goncalves, Maki K Habib, Julita Bermejo-Alonso, Marcos Barreto, Mohammed Diab, Jan Rosell, Joao Quintas, et al. A review and comparison of ontology-based approaches to robot autonomy. 2019.

J Neil Otte, John Beverley, and Alan Ruttenberg. Bfo: Basic formal ontology. *Applied ontology*, 17(1):17–43, 2022.

Mihai Pomarlan and John A Bateman. Embodied functional relations: A formal account combining abstract logical theory with grounding in simulation. In *FOIS*, pages 155–168, 2020.

Mihai Pomarlan and Robert Porzel. Narrative objects. 2022.

Valentina Presutti and Aldo Gangemi. Dolce+ D&S Ultralite and its main ontology design patterns. In Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, number 25 in Studies on the Semantic Web, pages 81–103. AKA GmbH Berlin, 2016. ISBN 978-3-89939-715-6.

Ixchel G. Ramirez-Alpizar, Ryosuke Hiraki, and Kensuke Harada. Cooking Actions Inference based on Ingredient's Physical Features. In *2021 IEEE/SICE International Symposium on System Integration (SII)*, pages 195–200, Iwaki, Fukushima, Japan, January 2021. IEEE. ISBN 978-1-72817-658-1. doi: 10.1109/IEEECONF49454.2021.9382721.

Yuan Ren, Artemis Parvizi, Chris Mellish, Jeff Z Pan, Kees Van Deemter, and Robert Stevens. Towards competency question-driven ontology authoring. In *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11*, pages 752–767. Springer, 2014.

Anne-Sophie Riegger, Katrin Merfeld, Jan F Klein, and Sven Henkel. Technology-enabled personalization: Impact of smart technology choice on consumer shopping behavior. *Technological Forecasting and Social Change*, 181:121752, 2022.

Ellen Riloff, Rosie Jones, et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.

Peter N Robinson and S Mundlos. The human phenotype ontology. *Clinical genetics*, 77(6):525–534, 2010.

Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

Vetle Ryen, Ahmet Soylu, and Dumitru Roman. Building semantic knowledge graphs from (semi-) structured data: a review. *Future Internet*, 14(5):129, 2022.

Ansgar Scherp, Thomas Franz, Carsten Saathoff, and Steffen Staab. A core ontology on events for representing occurrences in the real world. *Multimedia Tools and Applications*, 58:293–331, 2012.

Helmut Schmid. Improvements in part-of-speech tagging with an application to german. In *Natural language processing using very large corpora*, pages 13–25. Springer, 1999.

Lynn M Schriml, James B Munro, Mike Schor, Dustin Olley, Carrie McCracken, Victor Felix, J Allen Baron, Rebecca Jackson, Susan M Bello, Cynthia Bearer, et al. The human disease ontology 2022 update. *Nucleic acids research*, 50 (D1):D1255–D1261, 2022.

Lynn Marie Schriml, Cesar Arze, Suvarna Nadendla, Yu-Wei Wayne Chang, Mark Mazaitis, Victor Felix, Gang Feng, and Warren Alden Kibbe. Disease ontology: a backbone for disease semantic integration. *Nucleic acids research*, 40(D1): D940–D946, 2012.

Issei Sera, Natsuki Yamanobe, Ixchel G. Ramirez-Alpizar, Zhenting Wang, Weiwei Wan, and Kensuke Harada. Assembly Planning by Recognizing a Graphical Instruction Manual. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3138–3145, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636041.

Evren Sirin, James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *1st Workshop on Web Services: Modeling, Architecture and Infrastructure*, pages 17–24, 2003.

Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544, 2001.

Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

Hye Jin Sung and Hyeon Mo Jeon. Untact: Customer's acceptance intention toward robot barista in coffee shop. *Sustainability*, 12(20):8598, 2020.

Daniel Taylor, Sebastian Brockhaus, A Michael Knemeyer, and Paul Murphy. Omnichannel fulfillment strategies: defining the concept and building an agenda for future inquiry. *The International Journal of Logistics Management*, 2019.

M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *IEEE Robotics Automation Magazine*, 18(2):58–68, June 2011. ISSN 1558-223X. doi: 10.1109/MRA.2011.940993.

Mori Tenorth and Michael Beetz. KNOWROB - Knowledge Processing for Autonomous Personal Robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, St. Louis, MO, USA, October 2009a. IEEE. ISBN 978-1-4244-3803-7. doi: 10.1109/IROS.2009.5354602.

Moritz Tenorth and Michael Beetz. Knowrob—knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ international conference on intelligent robots and systems*, pages 4261–4266. IEEE, 2009b.

Moritz Tenorth, Ulrich Klank, Dejan Pangercic, and Michael Beetz. Web-enabled robots. *IEEE Robotics & Automation Magazine*, 18(2):58–68, 2011.

Christopher Thompson, Haris Khan, Daniel Dworakowski, Kobe Harrigan, and Goldie Nejat. An autonomous shopping assistance robot for grocery stores. In *Workshop on Robotic Co-workers 4.0: Human Safety and Comfort in Human-Robot Interactive Social Environments 2018 IEEE/RSJ*.

Madhura Thosar, Sebastain Zug, Alpha Mary Skaria, and Akshay Jain. A Review of Knowledge Bases for Service Robots in Household Environments. In *Proceedings of the 6th International Workshop on Artificial Intelligence and Cognition*, pages 98–110, Palermo, Italy, 2018a.

Madhura Thosar, Sebastian Zug, Alpha Mary Skaria, and Akshay Jain. A review of knowledge bases for service robots in household environments. In *AIC*, pages 98–110, 2018b.

Piyaporn Tumnark, Filipe Almeida da Conceição, João Paulo Vilas-Boas, Leandro Oliveira, Paulo Cardoso, Jorge Cabral, and Nonchai Santibutr. Ontology-based personalized dietary recommendation for weightlifting. 2013.

Michael T. Turvey. Ecological foundations of cognition: Invariants of perception and action. In Herbert L. Pick, Paulus Willem van den Broek, and David C. Knill, editors, *Cognition: Conceptual and Methodological Issues.*, pages 85–117. American Psychological Association, Washington, 1992a. ISBN 978-1-55798-165-3. doi: 10.1037/10564-004.

MT Turvey. Ecological foundations of cognition: Invariants of perception and action. 1992b.

Jan-Phillip Töberg, Michaela Kümpel, Vanessa Hassouna, Michael Beetz, and Phillip Cimiano. Towards a knowledge engineering methodology for flexible robot manipulation in everyday tasks. In *European Semantic Web Conference*, 2023. submitted.

Mike Uschold. Building ontologies: Towards a uni ed methodology. In *Proceedings of 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems*. Citeseer, 1996.

David Vernon, Claes von Hofsten, and Luciano Fadiga. *The iCub Cognitive Architecture*, volume 11, pages 121–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-16903-8 978-3-642-16904-5. doi: 10.1007/978-3-642-16904-5_7.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. Bart: A modular toolkit for coreference resolution. In *Proceedings of the ACL-08: HLT Demo Session*, pages 9–12, 2008.

Atro Voutilainen. *Part-of-speech tagging*, volume 219. The Oxford handbook of computational linguistics, 2003.

Markus Waibel, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, JMM Montiel, Alexander Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2): 69–82, 2011.

Georg Waltner, Michael Schwarz, Stefan Ladstätter, Anna Weber, Patrick Luley, Horst Bischof, Meinrad Lindschinger, Irene Schmid, and Lucas Paletta. Mango-mobile augmented reality with functional eating guidance and food awareness. In *International Conference on Image Analysis and Processing*, pages 425–432. Springer, 2015.

XiuQing Wang and ShunKun Yang. A tutorial and survey on fault knowledge graph. In *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health: International 2019 Cyberspace Congress, CyberDI and CyberLife, Beijing, China, December 16–18, 2019, Proceedings, Part II 3*, pages 256–271. Springer, 2019.

Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

Gerhard Wohlgenannt and Filip Minic. Using word2vec to build a simple ontology learning system. In *ISWC (Posters & Demos)*, 2016.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, 2007.

Nasser Zalmout, Chenwei Zhang, Xian Li, Yan Liang, and Xin Luna Dong. All you need to know to build a product knowledge graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4090–4091, 2021.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about Goals, Steps, and Temporal Ordering with WikiHow. In *Proceedings of the 2020 Conference* *on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.374.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. Show Me More Details: Discovering Hierarchies of Procedures from Semi-structured Web Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. arXiv, March 2022. doi: 10.48550/arXiv.2203.07264.

Glossary

Anchor An anchor, also called world anchor, is used as a reference point for localisation in Augmented Reality applications. A world anchor is a digital (not necessarily visible) object at a specific coordinate which can also be attached to some real world object like a table or shelf that can be detected in an application..

AR In Augmented Reality (AR), digital content like holograms is projected onto the real environment. On Smartphones, the camera view is used and augmented with digital objects while smart glasses have a see-through display to augment digital objects onto the environment..

CRAM The cognitive robot abstract machine (CRAM) is a cognitive architecture that provides an entry point for implementing and deploying software on autonomous robots. It can be seen as the connection of the multiple software frameworks needed in robotics: Planning, Perception, Manipulation as well as Reasoning. In CRAM, actions are implemented as general plans that can be parameterised..

KnowRob KnowRob is a knowledge processing framework that provides a robot with knowledge representation and reasoning while being grounded in a robots' physical system. Its logic programming language Prolog enables reasoning over various sources..

Prolog Prolog is a logic programming language that enables declarative programming, which can be used to highlight the problem at hand instead of the way how to solve it. Different sources can be integrated by resolving them to variables..

SemDT *A semantic Digital Twin (semDT) is a symbolic representation of robots, human beings, and their environment as physical elements connected to complementary non-physical entities as well as their properties and interrelations, represented by data structures of Virtual Reality scene graphs. Thereby abstract information associated with the entity of interest can be inferred, reasoned about, and visualised through a variety of media to predict current or future conditions. Particularly, actions can be simulated, and hypothetical scenes can be rendered to support and enhance decision-making.* (Kümpel et al., 2021).

SOMA The socio-physical model of activities (SOMA) is a top-level ontology based on DUL that describes agents and objects as well as the roles they play during events. This is done by using the concepts disposition and affordance, that can be used to set objects in relation..

URI An Uniform Resource Identifier (URI) identifies a resource. In contrast to a Uniform Resource Locator (URL), which usually identifies a website, an URI can also identify a person, an object or a file which does not need to be resolved to a website address..