TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DISSERTATION

Personal Geographic Information Management

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften under der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Andrew U. Frank
E120-2
Department of Geodesy and Geoinformation
Research Group Geoinformation

eingereicht an der Technischen Universität Wien
Fakultät fuer Mathematik und Geoinformation

von

Amin Abdalla, Msc.
Matr. Nr. 0626298
Wulzendorfstr. 77/E/155
1220 Wien

Wien, am

بسم الله الرحمن الرحيم

والحمد والشكر لله رب العالمين

Acknowledgements

Viele Menschen haben in der einen oder anderen Form dazu beigetragen diese Arbeit zu Ende zu bringen und Ihnen allen möchte ich hiermit danken.

Besonderer Dank gilt meiner Frau Sarah, für ihre Geduld und moralische bzw. finanzielle Unterstützung, welche vor allem gegen Ende ausschlaggebend war. Meinen Kindern Yumna und Habiba, welche mir meine geistige Gesundeheit erhielten. Meiner Schwester Salma, für Rat und Mitgefühl. Meiner Schwester Mona und Ihrem Mann, ohne deren Gastfreundschaft ich diese Chance nie ergreifen hätte können. Meinem Vater, dafür dass er mir den Wert der Bildung näher brachte. Meinen Freunden Mario und Hisao, dafür dass Sie mich nicht den Blick für das Wesentliche verlieren ließen. Meinem Mitstreiter Paul Weiser für seinen Sarkasmus und Unterstützung.

Allen Arbeitskollegen, für deren Beistand und Hilfe.

Zuguterletzt natürlich Prof. Frank und Prof. Janowicz für Betreuung und Rat.

Abstract

Looking back at early geographic information systems, it becomes obvious that technologies nowadays are significantly different. With the emergence of spatially aware mobile computers, every day citizens became producers of spatial information. Due to these developments a new form of geographic information, namely, personal geographic information has emerged. While there are similarities to traditional geographic information; considerable differences are present too. This work explores the issues a personal assistant application needs to address in order to effectively handle and manage personal geographic information. It builds on the assumption that spatial aspects and semantics of personal geographic information are tightly bound to activities. The main focus, therefore, lies on the development of a model that allows to represents human activities from a personal perspective. The model provides a multi-granular representation of activities in place and time as well as other components, such as conceptual relations and requirements. Two use case scenarios will demonstrate its applicability. The work shows that an understanding of a user's future, present and past activities, are essential for contextually sensitive structuring and management of personal information.

Keywords PIM, PGI, Activities, Personal Information, Calendars, Schedules, Granularity, Information Management, Space-Time Path, Time Geography

Contents

List of Figures

Chapter 1

Introduction

Looking back at early geographic information systems, it becomes obvious that technologies nowadays are significantly different. Due to the developments in the web and the emergence of spatially aware smart phones, new forms of digital information is produced. In this thesis I argue that we witness the introduction of a new form of geographic information, namely, personal geographic information.

The history of Geographic Information Systems can be traced back into the 1960s, that of its antecedents even before that [Mark et al., 1997]. The first tools for geo-spatial data manipulation were developed by several different institutions, like Harvard Laboratory for Computer Graphics (LCG), the Canada Geographic Information System (CGIS) or the Environmental Systems Research Institute (ESRI) [Coppock and Rhind, 1991].

The motivation behind the development of Geographic Information System (GIS) at that time was, according to Coppock and Rhind [1991], on the one hand academic curiosity and on the other the need of greater speed and efficiency in the manipulation of geospatial data. GIS –until very recent– was a topic disjoint from civil society and mostly applied for administrative, planning or scientific purposes. This was perfectly reasonable, since only these kinds of institutions were producing vast amounts of geographic data, like the US Census for example. Thus it was them who were in desperate need of more efficient ways to maintain, process and analyze geographic data.

It was only recently that GIS became a commercial product available for the broad public and non-professionals, mainly due to a revolution taking place in the 1990s. At the time the first interactive *web mapping services* emerged, like the Xerox PARC Map Viewer [Putz, 1994] or MapQuest[1]. These services, for the first time, allowed

[1] http://www.mapquest.com/

non-professionals to access and query geo-spatial data for their very own purposes. Purposes which were not only related to the fields mentioned before.

The trend further developed and was boosted by the introduction of Google Maps[1]. In those early days such services were of quite limited functionality (e.g. address search or simple map printing), by now, dozens of new functionalities were added [Schmidt and Weiser, 2012].

Users can find routes from A to B or view entire cities in 3D. Suddenly, GIS technologies became widely accessible to civilian and non-professional users who did not have any prior education in the field. This phenomena was referred to as *democratization* of GIS [Butler, 2006]. About the same time other commercial applications of geographic information were brought to the market. In 1998, the US government issued an executive order resulting in the Global Positioning System (GPS) signal for civilian use to be as accurate and reliable as for the military, by the year 2000. Following this new development, car navigation systems became widely available for purchase, and with the introduction of smart phones and their incorporation of GPS receivers, the number of applications using this new technology, so-called Location Based Services (LBS), skyrocketed.

Another crucial development in the web mapping domain, is the ability to populate/create maps by non-expert people. People nowadays are able to share geographic information *they* produced. The data can be captured by GPS-enabled devices, or by vectorizing areal photographs on the desktop for example. This phenomenon defined the notion of *Neogeography* or *New Geography* [Turner, 2006], referring to the fact that non-experts are allegedly acting as geographers. Goodchild [2009] argued against the term and therefore introduced the more accurate term of Volunteered Geographic Information (VGI), described as "...the widespread engagement of large numbers of private citizens, often with little in the way of formal qualifications, in the creation of geographic information..." [Goodchild, 2009]. The Open Street Map (OSM) project[2] is probably the most famous example of VGI.

An important aspect of VGI is, that it is captured intentionally in order to be shared with the wide public. Qualitative studies looked at the reasons why people map and share information and found various motivations among them: altruism, intellectual stimulation, social rewards, pride or simple interest [Coleman et al., 2009]. Thus, the data is (1) produced intentionally and (2) in the majority of cases not produced for the sake of private use.

[1] http://www.google.com/maps
[2] www.openstreetmap.org

However, there is a whole other range of *spatial* data produced by the general public, for other purposes and intentions. Users maintain digital contact-lists containing addresses, digital calendars containing spatially located events, have GPS modules incorporated into their mobile devices (i.e., mobile phones, digital cameras, tablet-computers); all of which feeds into a collection of personal *spatial* data. For the lack of another terminus this sort of information will be referred to as *personal geographic information* in the remainder of this work. Personal geographic information differs in purpose and nature to the information people produce for projects like OSM. It is produced by a user for private purposes only (e.g.: geo-tagged tasks or events, images, etc...), but some forms of it can be gathered without a users conscious knowing or willing[1]. Personal geographic information is not intended to be shared in general, except with a chosen set of persons (e.g., family, friends, colleagues). Further, it is mostly useless to the general public, due to its close relation to a person's personal context, although commercial enterprises can benefit greatly from it (i.e, targeted advertisements, to study customer behavior, etc...).

Personal geographic information is in most cases not geographic per se, since it is not always a representation of real world entities[2]. Rather, its spatial properties are in most cases a result of our own acting in space, i.e., by creating a document in an office it receives a creation date and location, which is equal to the point in space and time at which the creator conducted the activity. The document is not a traditional geographic entity, but can be related to the physical location in which the causing action took place. At the same time, the document can *be about* or *contain* spatial information. It is argued that there is a need to differentiate between the two. Apart from, documents, pictures and the like, digital representations of tasks and calendar events are part of the realm of personal information. It can be information about future events, plans or activities, which in turn can be the cause for the production of new personal information (e.g., pictures). The complete set of digitally manifested personal information is referred to as personal information space [Jones and Teevan, 2007].

Researchers in the field of Personal Information Management (PIM) put much effort in understanding how people organize their digital and virtual desktops (i.e., their personal information space). In PIM literature one of the core activities is the: "...maintenance of a mapping between information and need" [Jones and Teevan, 2007, p.15]. Thus, the goal of PIM-tools is to find the appropriate information in a person's personal information space. Two questions emerge: (1) How to structure a person's personal

[1] http://www.theguardian.com/technology/2011/apr/20/iphone-tracking-prompts-privacy-fears
[2] A general rule in OSM is that only *physically existent* things ought to be mapped.

information space; (2) How a *need* is to be inferred?

The argumentation in this work follows the idea that an understanding of the spatio-temporal dimension of user activities, can provide a structuring principle for the personal information space and give necessary context to infer user needs in certain situations.

The hypothesis of this work is:

> The spatio-temporal structure of our activities is reflected in our personal information space, and therefore essential for its effective organization.

The eminent research question is:

> Can a formal, spatio-temporal and multi-granular representation of human activities give the necessary structure to provide novel ways to organize, manage and retrieve personal information?

This work puts an emphasis on the representation of activities, especially of those to be conducted in future. It will highlight the role of aggregation for describing activities, something essential when coping with multi-granularity. Therefore, work in the field of PIM as well as perspectives on human activity from geography, cognitive science and psychology will be presented and discussed throughout this book.

The thesis is structured as follows: Chapter 2 starts with a general introduction and discussion of PIM and its relation to Geography. Chapter 4 introduces example scenarios and draws a rough sketch of the capabilities a personal geographic information management tool can provide. Chapter 3 concerns itself with the representation of plans and intended activities. Outcomes of a user study build the basis for a discussion of some of the shortcomings of current PIM-tools. In chapter 5 a notion of granularity is defined, that is used in chapter 6, which presents an ontology of intended actions. Chapter 7 presents the operations and inferences possible by using the proposed activity ontology. Chapter 8 evaluates the model by applying it to a use case. Finally, a conclusion and future work section can be found in chapter 9

Chapter 2

Personal Geographic Information Management

This chapter discusses the field of Personal Information Management and the role geographic space takes in it. In particular, the notion of *personal geographic information* and its distinguishing properties from personal information, geographic information and volunteered geographic information are in the focus of attention. The chapter is partially based on earlier work [Abdalla and Frank, 2011], in which an outline of a *personal geographic information management* tool is drawn.

2.1 What is Personal Information?

In order to define personal information, it is useful to start with its constituting components, information items:

> "An information item is a packaging of information in a persistent form that can be acquired, created, viewed, stored, grouped (with other items), moved, given a name and other properties, copied, distributed, moved, deleted, and otherwise manipulated."[Jones and Teevan, 2007, p.7]

An important point in the above quotation is the phrase *packaging of information in a persistent form*, that in this context refers to information items that take the form of an external physical manifestations. Thus, memories about a meeting in a person's head do not fall into the above category of information items. An address scribbled on a piece of paper, or an email saved on your computer, on the other hand, does.

In the wake of ubiquitous and mobile computing, information items become increasingly digital. A *digital information item* exhibits a certain *type* or *form*, that, together with an appropriate application, determines the operations possible with it [Jones and Teevan, 2007]. For example, an email in conjunction with an email-client allows at least for creating, deleting, saving or sending information items of type *email*. A word-processor allows for creating, saving, deleting and changing a document of the form *text-document*. In a more general sense, an information item of type T in conjunction with an application that provides a set of operations O forms an abstract algebra $< T, O >$ [Gill, 1976].

Finding a sharp delineation of what sort of information falls into the category of *personal information* is a non-trivial task. In an attempt to clarify the term, Jones and Teevan [2007, p.9] list the following points:

1. information a person keeps,
2. information about a person but kept by and under control of others,
3. information experienced by a person but not necessarily in the person's control, and
4. information directed to a person.

The above points support the statement that personal information comes into being in the process of a person's activity. Whether it is the information collected by companies about a person's shopping behavior, email conversations at work, the browsing history or photo collections of the last holidays. In all cases, the information is related to a person, by the activity that produced it.

The sum of all this information (produced by a person's activity) forms a personal information space [Jones and Teevan, 2007]. A personal information space carries at its center information a person is aware and in control of (e.g., documents, emails, etc...), and at its periphery information that might not be under a person's control (e.g., search history, bank-account data). A personal information collection is a subset of the personal information space consisting of the information a user actively stores, organizes and manages [Jones and Teevan, 2007]. A folder containing favorite music, but also paper-documents filed in an office, form such personal information collections. Creating them is in essence part of the activity of Personal Information Management (PIM), as will be explained in the next section.

2.2 Personal Information Management (PIM)

The field of research studying the way we manage and handle personal information, is referred to as Personal Information Management (PIM). Jones and Teevan [2007, p.3] define PIM as:

> "...the practice and the study of activities people perform to acquire, organize, maintain, retrieve, use, and control the distribution of information items such as documents, Web pages, email messages for everyday use to complete tasks (work-related and not) and to fulfill a person's various roles (as parent, employee, friend, member of community, etc...)."

Traditionally, PIM-research has focused on the investigation of how people *store*, *organize* and *retrieve* information [Barreau and Nardi, 1995; Jones, 2004]. Finding and re-finding (i.e., retrieval) of information items has been thoroughly investigated [Bruce et al., 2004; Capra and Pérez-Quiñones, 2003; Capra III et al., 2005]. Capra III [2006], for example, found that people exhibited similar patterns for re-finding as for searching. Participants were asked to search for information in the web, and after approximately a week, asked to re-find it. Many simply repeated the strategy utilized for the initial finding task. However, people who put effort in *keeping* information, i.e., actively organizing the information such that it can be found for future use, do not have to repeat the initial search patterns. Unfortunately, this is not as trivial as it may sound, since it is very hard to foresee the future value of information. Often, only after the information was encountered, it becomes clear that it was useful. A phenomena termed *post-valued recall* [Wen, 2003].

Related to this phenomenon, is the difficulty of categorization. Malone [1983]'s seminal study showed that people shied away from filing documents, caused by the fear that once it is categorized for a specific reason, it will be unavailable if retrieval is needed for other reasons. In many cases this actually led to the avoidance of *filing* information and to the practice of *piling*. Malone concluded that "some information is stored in *files* and some in *piles*" [Malone, 1983, p.110] due to four forces: (1) mechanical difficulty of creating labels, (2) cognitive difficulty of creating appropriate categories, (3) desire to be reminded of tasks, and (4) desire to have frequently used information accessible. His suggestion was that computerized systems can tackle the four problems by: "...providing intelligent aids for categorizing and retrieving information and for reminding about things to be done." [Malone, 1983, p.111] .

In a deeper analysis about the psychological ongoings behind the scenes of Malone's observations, Lansdale [1988] attempted to explain some of the reasons for the behavior. The core argument is that human memory is not storing content, but meaning. Henceforth, categorizing documents does not help in the retrieval process. He cites the classic study of Chase and Simon [1973] who tested the recall capabilities of chess positions. In their investigation professional chess players were asked to remember positions from a chess board. Comparing their performance to novice players, it turned out that they were able to reconstruct the configurations much better than novices when the positions were taken from a real play. Nevertheless, when the figures were placed arbitrarily, their memory did not considerably outperform that of the novices. Thus, what the professional chess players remembered was not the positions of the pieces, but the structure and meaning of the positions.

It explains that in Malone [1983]'s study people with tidier desks and less problems of information retrieval had more proceduralized jobs (e.g., purchasing agents); those who had messier desks tended to occupy more flexible jobs (e.g., research scientists). Applied to information management, Lansdale [1988][p.58] argues that the "process of information management should be a well-defined event in which choices made fall into a clear pattern of organisation...".

Another point stressed by Landsdale is that *recall* in humans is highly influenced by its context. As Lansdale [1988] put it: "the ability to recall information depends upon a critical relationship between how the information is held in memory and what we are thinking about when we are trying to retrieve it". For example, Tulving and Thomson [1973] showed that people who *store* a word along with a specific cue word, are not performing well in recognizing the word when it is produced out of associations with another cue word. The following is a short description of Tulving and Thomson [1973]'s experiment in the words of Lansdale [1988][p.58]:

> Subjects were shown words that had to be remembered, such as *jam*, in the context of another word, such as *traffic*. Later, subjects were given words, such as *marmalade* and asked to produced words closely related to them, of which *jam* might be one. These words were chosen so as to be likely to produce as responses words which where in the original to-be-remembered list. Later, the subjects were asked to recognize which, if any, of the words they generated were words they had been originally asked to remember. Finally, they were given the words which appeared with the target words as prompts for recall –eg, traffic – ? What happens in this

> experiment is that words such as *jam* may not be recognized in the context of the closely related words such as *marmalade*.

It follows that *meaning* is more important than the word itself, thus *marmalade* does not trigger recall of *jam* because the meaning is different to the initial context *traffic*. What can be taken from Landsdales analysis is that (1) people shy away from categorizing things under one label; (2) the right cues are crucial for recall; and (3) people memorize meaning or structure rather than raw information.

While Malone [1983]'s studies were conducted in a real world office environment, younger PIM research started to investigate PIM-behavior in virtual environments. For example, Barreau and Nardi [1995] came to the conclusion that even on personal computers, there is a preference for location-based file search (as opposed to text-search) and file-placement as a reminding function (similar to the piling-behavior Malone reported). Nevertheless, PIM studies at that time were mainly conducted on systems using a *desktop-metaphor* (i.e., iOS, MS Windows) and thus did not go beyond the idea of folder-structures. Fertig et al. [1996], therefore, claim that some of the conclusions are artifacts of the software environments employed. They propose alternative metaphors, such as a time-ordered stream of documents that allows extrapolation to create reminders [Freeman and Gelernter, 1996].

As mobile computing became more feasible, PIM-research moved beyond the realm of desktops. Mobile devices suddenly provided the opportunity to gather information about *physical* activities, as opposed to *virtual* activities on the desktop. Soon, efforts were made to capture and store as much information as possible about a person; such as locations, pictures, sound and so on. Wearable devices (e.g., SenseCam [Hodges et al., 2006]) or personal logging services (e.g., reQuall[1]) were developed.

The MyLifeBits-project [Gemmell et al., 2006, 2002] is probably the best example for this sort of data gathering, also referred to as *lifelogging*. The system attempts to capture as much information about a user as possible without considering what sort –or of how much value– it might be. Sellen and Whittaker [2010] distinguished two forms of lifelogging (1) total capture and (2) situation-specific capture. They question the practice of the former one in particular; claiming that benefits of the approach are relatively ill defined. A study conducted in 2007 [Sellen et al., 2007] assessed the utility of lifelogs for memory support and concluded that: "...at least some kinds of cues captured by life-logging technologies, in this case SenseCam images, can be shown

[1] http://www.reqall.com

to provide effective links to events in people‘s personal past.” On the other hand, they stated that ”...the study raises the possibility that the potency of images as cues for remembering might not be effective in the very long term.” [Sellen et al., 2007].

Given the ambiguity in the real value of capturing *everything* (which clearly is impossible) the problems associated to it (e.g., storage memory, additional sensor requirements) gain significance. Sellen and Whittaker [2010] therefore argue that efforts should be focused on data people find more valuable and issues they find more problematic. They highlighted, for example, the role PIM-systems can play in support of our prospective memory [Graf and Uttl, 2001], in contrast to the current bias towards *retrospective memory* [Bell et al., 2009; Dumais et al., 2003; Kalnikaite et al., 2010]. prospective memory may be defined as ”remembering to remember” [Winograd, 1988]. Although research on prospective memory acknowledges *place* as a factor Sellen et al. [1997], it does not play a considerable role in personal information management.

Prospective memory, is crucial for the successful achievement of intended activities and plans. Failure in prospective remembering can lead to annoying situations (e.g., forgotten keys or documents) or even tragic events, as in the case of a father who forgot to drop-off his child at the nursery and locked it in the car[1]. To support prospective remembering Sellen and Whittaker [2010], call for the investigation of *effective* reminders, i.e., timely cues dynamically generated from appropriate contexts.

To recapitulate, personal information management research faces three major challenges that are somewhat related to each other.

Keeping/Storing What should be stored and what not? If ”total capture” is not desirable, how is it to be determined what information is of value to the user in future, and what is dispensable?

Finding What is the information, what type or category is it of? What are the features that are most likely to be remembered, so that it can be searched for?

Reminding Reminders were prominent features in the seminal studies that shaped the field of PIM (i.e., [Barreau and Nardi, 1995; Malone, 1983]). The question is how to create *effective* reminders as described by Sellen and Whittaker [2010].

[1]

2.3 The Case for Task-Centered Information Management (TIM)

Boardman and Sasse [2004] stated that "Many definitions of PIM draw from a traditional information perspective–that information is stored so that it can be retrieved at a later date." There are voices, though, that argue for the pursuit of *task-centered* information management [Catarci et al., 2007; Dix et al., 2007; Katifori et al., 2008; Lepouras et al., 2006]. The vision of task-centered information management, as opposed to the traditial PIM view, is not only that personal data should be arranged around the activities that produce or demand it, but also about task-inference and prediction [Catarci et al., 2007, 2006]. Katifori et al. [2008][p.1] claim that "Users should not have to focus on managing their information but rather on performing tasks this information is to be used for."

Mechanisms have been proposed and prototypical implementations deployed, that illustrate the possibilities of task-centered information management systems or Personal Interaction Management Systems [Catarci et al., 2007, 2006]. Such a system aims to link information sources to actions through a (1) recognizer and (2) a personal ontology [Catarci et al., 2007]. Dix et al. [2007], for example, show how *intelligent assistants* can help to semi-automatically fill out web-forms by inferring facts from a knowledge base. Given a form with fields like: First Name, Last Name, Address City etc., a user input of the first name, could result in automatically filling out the rest of the form.

The work on task-centered information management highlights the importance of a rich semantic description of not only the types of things and objects that are dealt with on a computer, but also of a task-description language [Catarci et al., 2006] that is able to describe the activities a user conducts. The papers cited so far, are concerned about information on a desktop computer, and actions are modeled as independent entities not related to any external real-world activity the user is engaged in.

While the main argument of task-centered information management, namely information has to be organized and modeled around tasks, is valid, the assertion in this thesis is that looking at desktop-tasks alone, does not allow for a full understanding of the users activity. Especially, in the age of *ubiquitous* and *mobile* computing, the question in what context information was produced (e.g., in Vienna while attending a conference) does promise a more accurate response, than asking on what device it was produced. And since space and time are the two most salient dimensions in human life, they are fundamental to structuring our knowledge [Janowicz, 2010]; therefore core to describe activities.

2.4 Human Conceptualization of Space and Time

2.4.1 Space and Places

Human geographers [Tuan, 1975], GIScientists [Jordan et al., 1998], urbanists [Alexander, 1979; Lynch, 1960] or philosophers [Schatzki, 2010] have looked into the notion of place. Winter et al. [2009] argued that it is time to put place-models on the agenda for GIScience. The ontological discussions consent about places as being part of, but distinct from space. Otherwise, it is a very ambiguous term as the following four definitions illustrate[1]:

1. a particular position, point, or area in space; a location,
2. a portion of space designated or available for being used by someone
3. a position in a sequence or series, typically one ordered on the basis of merit
4. [in place names] a square or short street.

Each definition typically lists several sub-specifications, e.g., "a building or area used for a specified purpose or activity" in case of the first. Similarly, the second suggests a strong link between *activities* or *usage* and the concept. In contrast, the current way of storing and retrieving places in GIS are based on the view of places as a relation between location and objective properties. Schatzki [1991][p.2] in his account about *social reality* states:

> My conception, on the other hand, takes off from Heidegger's idea that human existence always constitutes its "there." This means that human life automatically opens (in Heidegger's language *erschliesst*) a nexus of places where it itself occurs. Of course, since social reality is interrelated, rather than individual lives, a person always proceeds through a nexus of places, that a plurality of human lives have opened together.

The assertion that the spatiality of social reality stems from the spatiality of human activity makes *place* and *human activity* inseparable. He suggests two forms of places (1) *places to* do X (conduct a certain activity) and (2) *paths*, that is, a place from A to B (i.e., it brings you from one location to another). These two forms of places are then

[1] http://oxforddictionaries.com/definition/english/place

set together into larger entities, like settings, locales and regions. While, places are anchored in physical objects that relates them to absolute space, settings are anchored in configurations of objects. The social space, according to Schatzki [1991], differs from absolute space in two ways: (a) a nexus of places is inhomogeneous, overlapping, and sometimes discontinuous; (b) places do not exist independent of human lives.

Schatzki [1991]'s account is somewhat in line with Tuan [1975]'s assertion "...place is a center of meaning constructed by experience."; in the sense that experience is closely linked to activity, i.e., we cannot experience without doing (even if it is only passively perceiving). Important is the fact that spatial structure is often a result of activities. For example, the arrangement of a laboratory is so to support the activities taking place there.

This idea resembles the notion of *affordance* introduced by Gibson [1979] in his investigation of how humans and animals perceive the environment. He suggested that the environment is perceived on the basis of its affordances, that is, the possibilities of interaction, rather than the discrimination of properties or qualities.
Jordan et al. [1998] stated the need for an affordance-based model of place as a more accurate alternative to common GIS representations. In their view places comprise of the following 6 aspects:

- Physical features
- Actions
- Narrative
- Symbolic representations/Names
- Socioeconomic and cultural factors
- Typologies/Categorizations

Schatzki [1991]'s notion of places in *social space*, though, does not ascribe affordances to a place itself, rather it is the physical configuration of objects that allows (affords) for certain activities, which then give way to the emergence of a place. A place in social reality, is only produced in hindsight after the affordance was perceived and an action taken.

The object configuration that the place is anchored in, do propagate their affordances to the place it is forming, making the place exhibit those affordances, as suggested by Jordan et al. [1998].

To conclude, interrelated human activity creates (social) places that at the same time mediate activity. Such places can be experienced by a single individual, even though their creation is an effect of interrelated lives. Only a subset of those places are relevant to an individual, depending on what activities are intended.

2.4.2 Calendars and Schedules

Calendars are a social institution developed out of the need for a shared reference frame to locate temporal occurrences of collective activities. They are a social construction and social scientists as Zerubavel, therefore, explained the emergence of calendars as an attempt to define social groups by, on the one hand synchronizing activities (e.g., religious festivals or events), and on the other, segregating them from other groups, by separating activities in time. Examples are the temporal segregation of resting days by Jews, Christians and Muslims (i.e., Saturday, Sunday, Friday) [Zerubavel, 1985].

Another point is the symbolism calendars bear, revealing itself in attempts to change calendar systems coinciding with social, political or religious reforms. A radical example is the alternative system introduced after the French revolution in 1793. This new calendar was designed to embody four themes representing the values of the revolution, i.e., secularism (by eradicating any form of christian occasions), rationalism (by basing it on a decimal system), naturalism (anchoring it to celestial movements) and nationalism (by naming certain concepts after famous revolutionary figures). Obviously, the system did not succeed [1]. Calendars are fundamental to communicate, coordinate or describe events and activities on a coarse (societal) granularity (i.e., days, weeks, months), and date back thousands of years.

In contrast, the invention of the *schedule* as a regulatory institution is relatively young and dates back to the medieval Benedictine monasteries. The purpose was to achieve the complete *ordering* of Christian life [Zerubavel, 1985], by introducing hours of irregular length[2], that allowed to divide the day into parts. The need for a schedule stemmed from the coarse resolution of calendars, not usable to regulate individual's lives on a hourly basis. In fact some say that "*...a society so complex as ours proba-*

[1] In a similar vein the adoption of calendar systems of one culture by another can be understood as a symbolic representation of the power exerted by one over the other (e.g., the Gregorian calendar system is nowadays de-facto a world-wide acknowledged institution, a fact that points to western supremacy over the vast rest of the world).

[2] The early schedules followed the Egyptian-Roman method of time recockning and divided the day into 12 equal parts of daylight and 12 of nighttime, depending on the time in the year, the length of these parts varied. They were referred to as "horae temporales" [Zerubavel, 1985, p.37].

bly could not function without relatively rigid time scheduling." [Parsons, 1964, p.203] (quoted by [Zerubavel, 1985]).

Arguably, calendars and schedules are to time, what places are to space. They allow human beings to reason, locate and communicate about abstract pehenomen (i.e., space and time) in an imprecise manner. They are fundamental to human understanding of space and time and therefore crucial for describing human activities.

2.5 Our Personal Geography

Traditional personal information management focused on the activity of information management and retrieval, but did not put much effort in investigating the *motivating external* activity. People arguing for task-centered information management (see section 2.3) emphasize the role these motivating activities play in information management. Including the context of an ongoing real-world activity can give a rich semantic understanding of the user's intentions. To do so, i.e., relating the representation of a real-world activity to the virtual information space, an understanding of human *spatial behavior* is required.

There are several fields that investigated spatial behavior and decision making extensively. A core issue in transport planning, for example, is the question of service demand. The dominant approaches for modeling it are: (1) recording of spatial behavior, or (2) examining the decision-making and choice processes that result in spatial behavior [Golledge and Gärling, 2001]. These are referred to as behavioral and structural models. While structural models represent the aggregate movement activities of populations, behavioral approaches try to take the uniqueness of each individual into account.

It led to the investigation of wayfinding as well as cognitive mapping and its impact on spatial behavior. It revealed that individuals build a mental map of their unique perception of the world [Downs et al., 1977]. Behavioural geographers in the 1960s and 1970s coined the notion of action space to describe "...an individual's total interaction with and response to, his or her environment"[Golledge, 1997]. An action space draws, according to Jakle et al. [1976], attention to the relation between an individual and her social and spatial environment. Thus, people acquire information about their environment and ascribe a subjective place utility [Golledge and Gärling, 2001, p.227], that determines an individual's willingness to spend time, effort or money to interact with a certain place. An important component of the action space is the activity space,

defined as "...the subset of all locations within which an individual has direct contact as a result of his or her day-to-day activities" [Golledge and Gärling, 2001, p.279].

In brief, people's *spatial behavior* or activity space is determined by, on the one hand, physical constraints, and on the other, socio-economical limitations and *knowledge* about the environment. Since geographers were mainly interested in societal or economic problems and their relation to space, the concept of activity space was primarily meant to be the space of movement from a gods-eye perspective. In this context, though, the aim is the development of a model that represents activity from an egocentric perspective; where movement is only one of the interesting dimensions.

personal geography can be understood as a *personal interpretation* of the environment, based on the information we relate to it. This is not to confuse with a mental map [Gould and White, 2012], which is mainly about individual perception of the physical world. personal geography is about the meaning we see in places, defined by our knowledge about it, it is therefore very close to the notion of action space [Jakle et al., 1976] as explained before. For example, a supermarket close to a persons home: the person will have a picture of it in mind, some idea of what she can buy there, what the price level is, the opening hours and how she can get there; maybe there are tasks attached to it (e.g. "I have to buy milk."). Based on that information she can infer things like: The bread there is fresher than at other places; or that you have to be there before a certain point time to be able to buy something. Some of the information may be objective and even available from other sources (e.g. opening hours) others are purely personal judgment based on past experience.

The information we relate to places play a great role on how humans behave and interact with their environment. To cope with the amount of information, people utilize external tools that help organize and manage it. People use post-its, todolists, documents and increasingly electronic devices to handle the vast amount of information important for their daily life. Thus, this personal information is becoming digital information that in many cases can be related to spatial locations, since it is closely related to our *action space* or our personal geography.

2.6 Personal Geographic Information

Research in PIM does not put much focus on *place* as a factor for organizing, managing or assessing the value of personal information. Even though, it can in the majority of cases be related to a geographic location. Some may have implicit spatial relations

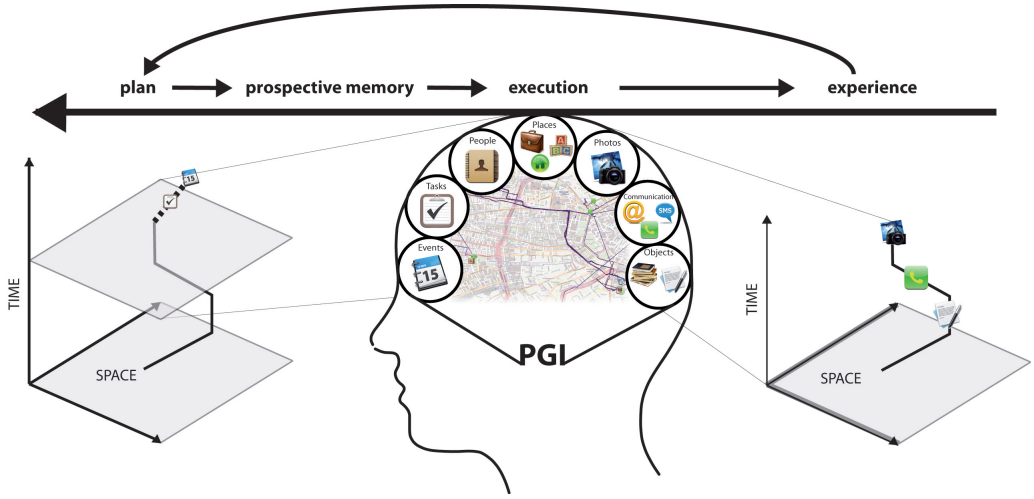

Figure 2.1: Illustration of how personal geographic information is produced and utilized. People project intended activities into the future, have information about their environment, and produce information along their activities. Figure first published in Abdalla and Frank [2011].

like a picture taken at a specific location or explicit ones, like a document talking about a place. Though the tools utilized by us are barely taking advantage of that fact. For proper planning of daily tasks, both temporal and spatial aspects must be considered. Therefore, time-geography [Hagerstrand, 1970] and its concepts (space-time path, space-time prism, etc...) is fundamental. The essential point is that on a physical level, movement-speed constraints a person's *accessibility* [Gregory et al., 2009; Miller, 1991], i.e., the degree of access to services or goods. Space, therefore, plays a role in scheduling or calendar applications, and prototypes as well as theoretic research has been published that take this fact into account [Abdalla, 2012; Raubal et al., 2004, 2007]. Personal information about our past is produced by our activities in space and time, henceforth they can be related to our space-time path by their temporal attributes. Using our spatio-temporal activities as an indexing principle allows to answer questions like: Where was I when I received the email/call? What notes did I take in that lecture? What pictures did I take on that trip?.

2.6.1 The Production of Personal Geographic Information

Longley et al. [2001] stated that "...all human activities require knowledge about the earth - past, present, or future". Figure 2.1 attempts to illustrate the inter dependencies existent between a person's space-time path (i.e., the conduction of activities) and the

information produced along it. On the right side the graphic shows a person's past space-time path and the information we attached to it along our way. For example, a picture that was taken, but also a phone call made or a document received. By conducting activities experience is accumulated, and knowledge about places, objects, people etc..., is formed. Based on such knowledge, plans are formed that allow to produce estimations of a person's future space-time path (left side). Having such a plan, actions which need to be executed in order to achieve the goals are derived, and are stored in a person's *prospective memory* [Graf and Uttl, 2001; Roedinger, 1996]. Prospective remembering requires monitoring of the environment for situations that call for actions that lead to the intended outcomes (e.g., watching the the hour, to catch the train.).

Figure 2.1 essentially visualizes the process of how *personal geographic information* is produced and how it is used in our daily lives. In brief, our personal geography is the physical environment in our own and personal context, which is not only about the past but also about the future. Looking at the graphic, it becomes apparent that space and time are fundamental concepts of knowledge representation, as argued by Janowicz [2010].

The *spatial* link, that personal geographic information introduces, is often omitted or underrepresented in the discussion of personal information. A person who takes photos at a conference, can matched them to the activities stored in a system, based on the time they were introduced into the personal information collection. The photo example does not express the full potential of linking personal information to user activities. Photos do exhibit, what what will be referred to as *explicit* spatial information. That is, they contain information that represents physical or real phenomena, located in space (e.g., a mountain, a restaurant, etc...). For structuring personal information, there is implicit spatial information produced by our activities in time and space. A traditional database storing temporal information about a phenomena, records two temporal properties: (1) The time standing for the phenomena; (2) A timestamp representing the time the observation was entered into the database. While both can have the same datatype (e.g., POSIX Time), the semantic information differs. The first attempts to model a real world (spatial) phenomena; the second tells about a user activity. In the context of PIM, it can be said that documents can (but not necessarily have to) represent spatial objects/phenomena (e.g., a map of the surrounding area of my hotel in pdf-format) and therefore have some *explicit spatial information* attached to them. Further, every piece in a person's personal information space is put there through an activity and can be related to some point in space-time (i.e., *implicit spatial information*). Figure 2.2

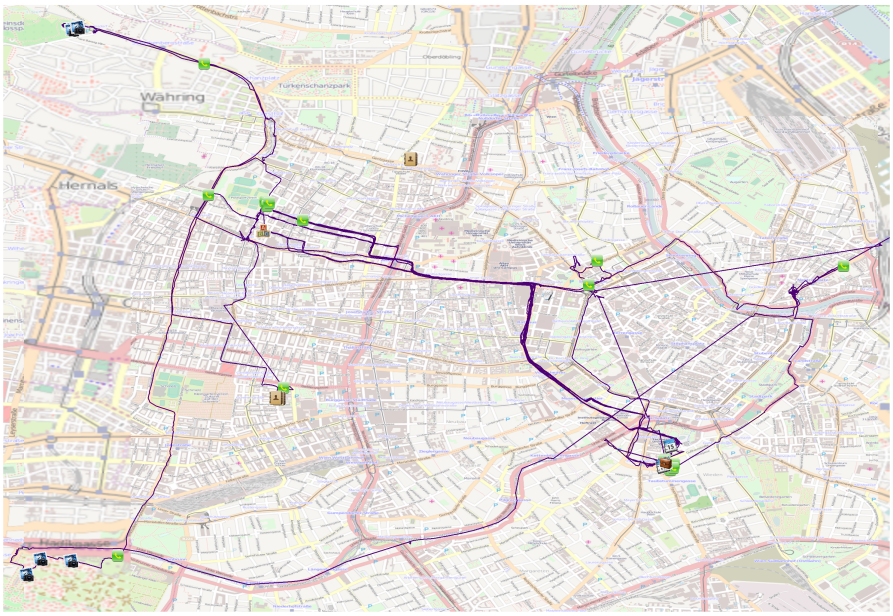

Figure 2.2: Personal information joined to a GPS track by time, hencforth related to space.

is an example of how data that has no explicit spatial information can be related to space by means of GPS-logging.

Using Figure 2.1 as a conceptual basis, personal geographic information is defined as personal information that is related to space-time through an activity or exhibits explicit spatial properties (i.e., a contact with an address line, a schedule entry with a location attached).

2.6.2 Distinguishing Personal Geographic Information from traditional Geographic Information

There are factors that contribute to the distinction of personal geographic information from professional- and Volunteered Geographic Information (VGI). Things like the purpose of use, the motivation behind the production or the quality. Table 2.1 lossely formulates the differences between the forms of geographic information.

Type	Production	Legal Issues	Targeted Users	Quality
GI	Usually produced by organizations of the public or private sector	Well defined licenses that restrict terms and use of the data	Target users are mostly companies, municipalities or other groups that are interested in large scale information sources	The data in most cases adhere to a certain standard and the mapping processes are often well documented. Henceforth, meta-data is available.
VGI	Produced by private individuals and groups.	Well defined licenses that make the terms of use clear.	No preferred user groups. Open to everybody.	Quality of VGI highly disputed. No formal procedures, quality assurance a community effort.
PGI	Actively or passively produced by individuals.	Legal issues of personal information are a hot topic. Until now there are no clear regulations of who can use the data.	The data is usually produced for private purposes and not intended to be shared, except for certain people (e.g., friends, family, colleagues).	In the context of personal information quality is a highly relative attribute, it depends on the specific purposes the information is intended to be used for.

Table 2.1: A informal comparison of the varying forms of geographic information

2.7 Summary

This chapter introduced the notion of personal information space as an abstract space that contains all the information relevant to a person's life. A personal information collection is a subset of the personal information space that consists of the information that is actively manged by the user. Personal Information Management (PIM) is often seen as a subfield of information retrieval, eventhough *reminding* has been acknowledged as an important factor [Malone, 1983]. The argument of task-centered information management is that *activities* should be the core structuring principle for personal information. Extending the task-centered information management argument, a pledge for the inclusion of physical activities was made. It was stated that personal information can be linked to space in two ways: (1) by inherent spatial information (i.e., spatial information that is contained in a document); (2) and by linking it to the activity that produced it (i.e., a trip that on which pictures where taken). A personal information collection can be structured by spatio-temporal activities. It can improve a whole range of PIM tasks, i.e., the planning of future events, the monitoring of present tasks and retrieval of stored information. To do so, a model of human activities is essential. It allows to link non-spatial data to a spatial context and provides valuable information about user needs.

Chapter 3

Planning and Representation of Intended Actions in Space and Time

The main goal of this work is to present a model representing intended activities close to the manner humans conceptualize it. A sound representation of our activities and intentions is vital for (1) the organization of personal information and (2) reminding functions of personal assistant tools. Therefore, in the following, an informal discussion of the relevant terminology and concepts will be given. Starting with the notions of *intention* and *planning*, *prospective remembering* will be introduced. Further, a synthesis of the findings from a qualitative user study that investigated human trip planning behavior [Abdalla et al., 2013] will be presented. It will illustrate gaps current tools exhibit when it comes to the representation of future activities or plans.

3.1 Planning for Intended Actions

3.1.1 Intentions

Investigating the motivational antecedents of plans, Kreitler and Kreitler [1987b] use the term *'Behavioral Intentions'*, referring to a concept that represents the answer to the question of "What will I do?". A behavioral intention leads to the selection of an appropriate pre-defined *behavioral program* (similar to the notion of *scripts* by Schank and Abelson [1975]). In case such a behavioral program is not found, a plan is formed. Intentions, therefore, play a crucial role for the formation of plans.

Table 3.1: Corresponding examples of planned activities or errands for goal- and implementation intentions

Goal Intentions	**Implementation Intentions**
Return the book to the library before the deadline	Attend lecture from 9:00 am to 11:00 am
Be at home before 8:00 pm	Meet me there at 1:00pm
I want to attend the conference	Take the bus at 10:00 to arrive at 11:00

Gollwitzer [1993] investigated the question of implementing intentions. He distinguished between *goal intentions* and *implementation intentions* (see Table 3.1). *Goal intentions* come in the form of "I want to reach x"; *implementation intentions* specify the situation related to the behavior leading to the goal, such as "When time = x, then do y (to reach z)". In that sense, implementation intentions are specifications of the former goal intentions. The distinction between goal and implementation intentions, though, is not crisp and the distance between goal and implementation does vary (e.g., "I want to be rich" vs. "I want to buy X").

It was demonstrated that goal intentions are more likely to be achieved when they are augmented with implementation intentions [Gollwitzer, 1993]. An attempted explanation is that by *'passing the control of one's behavior to the environment'* [Gollwitzer, 1993, p.173] people can be controlled by situational cues and thus automatize action initiation. To transform *goal intentions* into *implementation intentions*, *planning* is necessary.

3.1.2 Human Planning Behavior

A variety of research areas, including Artificial Intelligence, Cognitive Science and Psychology, have spent time and effort to understand the underlying processes of human *planning*. Despite numerous attempts, researchers have not reached the point where planning can be explained by a single definition or theory [Scholnick and Friedman, 1987]. One can explain planning from various foci, thus building definitions based on different aspects of planning (e.g: goal-definition, plan-formulation, monitoring, plan-adaption, etc...) [Scholnick and Friedman, 1987].

First some of the prerequisites of planning are outlined. At the bottom there is a need of *intelligence*, that is, in the words of Piaget [1960]: "a form of symbolic representation

that allows to *evoke absent realities*". Psychologists investigated how planning skills emerge in children, [Kreitler and Kreitler, 1987b, p.2] wrote:

> "It is unlikely that an individual could plan adequately without having developed notions of sequential ordering, hierarchical integration, time, and place, or without being able to think of causes, consequences, alternatives, and so on."

Kreitler and Kreitler [1987a] showed that children at a young age (approx. 5 years old) were prone to forgetting facts about a given planning task, concerned themselves with irrelevant facts or simply confused fact with fantasy. As they learn better conceptual tools by the years, e.g., general labeling or referencing by domains, the cognitive workload becomes easier and they are able to approach the task more systematic. Kreitler and Kreitler [1987a, p.255] state that:

> "...planning improves when the child intensifies meaning elaboration of the target with the result that the irrelevant aspects are sifted out and the relevant ones are ordered into functional groups."

So, as opposed to having to think about how to get milk, cereals and a bowl; to pour both of the former into the latter; we can simply talk about 'preparing breakfast' and group several processes into a single descriptor. This phenomena points to a *knowledge base* as a crucial part for the ability to plan. Thus, it follows that people who are planning for a familiar task in a familiar environment have significantly less effort than those who are in a completely unfamiliar situation. Besides basic cognitive abilities, planning does require a certain will to do so. Kreitler and Kreitler [1987b] also showed that children who *value* the need for *planning* higher, in general, have also developed better skills.

De Lisi [1987] pointed out that *planning* has two major connotations in common language. The first is used to describe a drawing or a diagram as used by town-planners or engineers. The second stands for a sequence of actions, or program for achieving something. He explains that the first understanding puts emphasis on the representational aspect, while the latter underlines the functional-behavioral aspects. Nevertheless, both carry the implicit notion of goal attainment. "Thus, goal attainment is a salient feature of a plan regardless of whether the functional or the representational meaning is emphasized" [De Lisi, 1987, p.6].

In general, literature agrees on the above claim that planning expresses goal-directed behavior [see Hayes-Roth and Hayes-Roth, 1979; Sacerdoti, 1975b; Scholnick and Friedman, 1987], distinguishing it from the unconscious *behavior* of a physical object. Miller et al.'s [1986] definition, commonly used in psychology, describes plans as a hierarchical process controlling the performance order of a sequence of operations. The definition seems limiting, as it imposes a hierarchical and sequential structure, that might not always be the case. Sacerdoti [1975a] for example, points to the non-linear nature plans can exhibit. Further, Hayes-Roth and Hayes-Roth [1979] shaped the term *opportunistic planning*, describing the fact that people often recognize an opportunity in a plan to conduct a related or unrelated task. Plans do evolve and often incorporate additional goals along the way.

In cognitive science, a plan is often put in the context of -or equated with- the more general problem-solving process. Mayer [1990] defines problem solving as "...cognitive processing directed at transforming a given situation into a goal situation when no obvious method of solution is available to the problem solver". Hayes-Roth and Hayes-Roth [1979, p.3] provide the following definition of problem solving:

> ...the predetermination of a course of action aimed at achieving some goal. It is the first stage of a two-stage problem solving process. The second stage entails monitoring and guiding the execution of the plan to successful conclusion. We refer to these two stages as planning and control.

The main aspects to take from the above definition is that planning is more specific than problem solving. In the words of Kreitler and Kreitler [1987a, p.208]: "...planning is the cognitive activity that produces plans". Problem solving then includes not only producing the plan, but executing and monitoring it as well. This understanding is somewhat contrary to the view of Artificial Intelligence, where the foremost aim of planning research is to enable automated planning. There, it is conceived as a response to the combinatorial explosion of earlier problem solving approaches [Russell and Norvig, 2010, p.366] that use atomic representations to search for a solution. Henceforth, it is a generalization of problem solving, that reduces the state-space in which a solution is sought.

The current work is inclined to the former [i.e., Hayes-Roth and Hayes-Roth, 1979] definition of problem solving that includes monitoring and execution. It does, to a certain extend, conform with the process depicted in Figure 2.1. A plan standing for the future and *control* happening in the present. So far it was determined that a plan

is first formed to achieve some goal, to be executed and monitored afterwards. To use the terminology of section 3.1.1 goals can be understood as *goal intentions*, while a plan is a set of *implementation intentions*.

3.1.3 Prospective Remembering

The monitoring of *plans* or implementation intention, does take place in our prospective memory. The transition from implementation intention to prospective remembering is seamless. Prospective memory is involved in remembering to perform a planned action or intention at the appropriate time. "What is unique about prospective memory tasks is that they require identifying or recognizing cues as telltale signs of previously formed plans and intentions..." [Graf and Uttl, 2001, p. 442].

Traditionally, there were two (somewhat contradictionary) lines of explaining prospective memory. One is based on the assumption that some attentional resources are deployed for monitoring or bringing the intentions to mind [McDaniel and Einstein, 2000], i.e., a form of periodically reminding oneself of something ought to be done. The second follows the idea that there is no such strategic or conscious reminding, rather the intended action is triggered by the encountered target situation [McDaniel and Einstein, 2000]. Both explanations are grounded in empirical evidence. This led to the assumption that, depending on the *type* of task, both of the mechanisms (i.e., strategic or automated) are actually employed [McDaniel and Einstein, 2000].

Therefore, there are two *types* of prospective remembering, referred to as: focal or non-focal [McDaniel and Einstein, 2011]. Focal remembering can make use of cues in the environment, so that no active remembering has to take place until the cue is encountered. The non-focal type of tasks demands a person to actively search for the situational cue to not forget the action, e.g., looking at the watch every now and then to be on time for an appointment. The neuroscientific model of McDaniel and Einstein [2000] separates the process of focal prospective remembering into a **planning/encoding** (i.e., determination of situations that trigger the action) and **retrieval** (i.e., detecting the situations and retrieving the intended actions from memory) part. By translating non-focal tasks (i.e., temporal evaluation) into focal-tasks (i.e., remembering triggered by an alert) the resources allocated for constant monitoring can be freed. In brief, a personal assistant tool can, by having an appropriate representation of the user's intention and state, detect the situational cues necessary and take over common prospective remembering tasks (e.g., meetings, acquiring/returning an object, etc..).

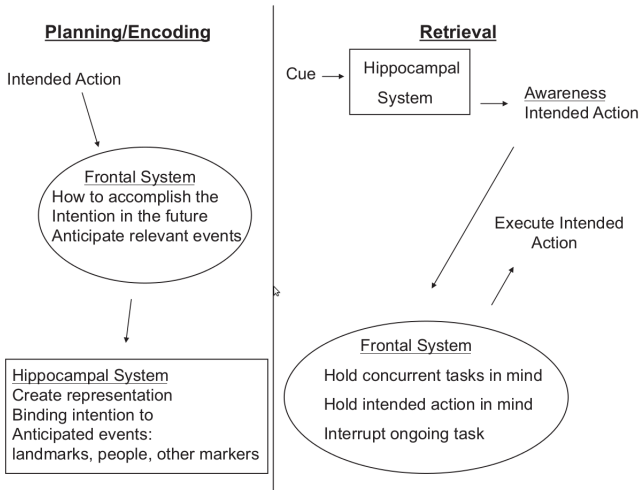

Figure 3.1: A graphical representation of prospective memory consisting of two main components: Planning/Encoding and Retrieval. Source: McDaniel and Einstein [2000].

3.2 Trip Planning: A User Study

In a user-study the planning behavior of 10 Vienna based individualswas investigated. The task was to plan and make arrangements for attending a conference in a foreign country (Tartu, Estonia), using the web. The study had two parts, the first is discussed in Section 3.2.1 which forms a subset of the findings described in Abdalla et al. [2013]. The second and smaller part is presented in section 3.2.2 where the participants were asked to represent the arrangements made in a common PIM tool.

3.2.1 Planning the Trip

For the analysis of the web-based planning process, the participants where filmed throughout their trip-planning activity. The acquired video material was *coded*, i.e., segmented into certain activities. Figure 3.2 shows the nine activity types that were looked for. For a detailed account of the coding process, the reader is referred to Abdalla et al. [2013]. The prevalence of certain activities varied from subject to subject, although some seemed to be consistently prominent. Those are *network understanding* and *querying*. Thus, a lot of effort was put into understanding the transportation network that serves as a basis for the trip. The second important activity was querying databases to seek information that provides the foundation for the various *understanding activities* (i.e., Network-, Event- and Place-Understanding). In contrast, there are activities that are sometimes not present at all, such as event understanding. As the

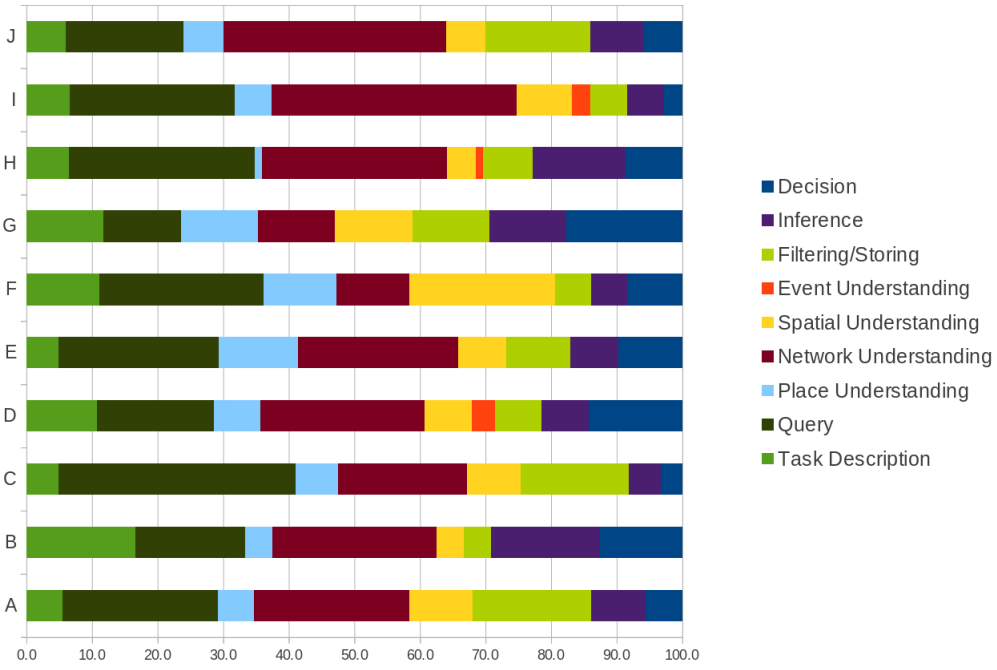

Figure 3.2: The amount of time spent on selected activities in relation to the complete process for each participant

study was interested in the spatial aspects of the planning process it looked at the number of times people employed a map. The *Looking at a map* activity was segmented into the *spatial-understanding* activity. Figure 3.3, though, splits it into large scale and small scale, i.e., looking at the map on a regional or a urban scale. The fact that makes Figure 3.3 interesting is that those participants who erred in the geographic aspects of their planning, where among those who never looked at a map on a regional scale. For example, one participant did not realize until after the experiment that Tallin and Tartu were two different cities hundreds of kilometres apart. It lead to the assumption that a taxi from Tallin airport will be enough to reach the hotel in Tartu.

Apart form the analysis of the data, the observations itself revealed interesting insights into the process. The following are a subset of the points described in Abdalla et al. [2013]:

- **Information Transfer**: The most striking cognitive activity was the extraction and feeding of information from different sources into various query interfaces. It was also one of the most error prone activities. 4 out of the 10 participants did

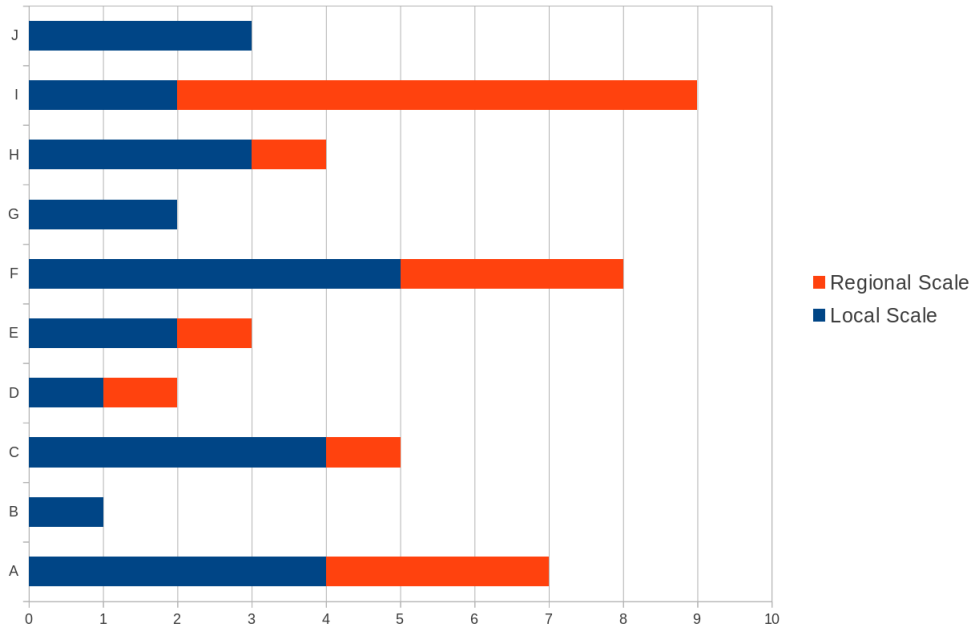

Figure 3.3: The absolute number of people utilizing a map to acquire spatial knowledge; distinguished by regional and urban scale. (First published in [Abdalla et al., 2013])

err in the querying and booking process of flights, caused by wrongly put date information.

- **Geographic Knowledge**: It was mentioned that some subjects did not build up a geographic representation of greater region. This lack of spatial knowledge resulted in a narrowed approach to the search for possible connections to Tartu. Almost all of those who had looked at a large scale map recognized that Riga is as close to Tartu as Tallin. Thus, they were able to look for solutions involving a flight to Riga and a connecting bus/train to Tartu. Those who did not have such knowledge were not able to make that inference.

- **Opportunities**: In our study it appeared that some subjects recognized opportunities. In one case, it was noticed that there is a flight going via Brussels, what could have been an opportunity to meet friends who live there. Consequently, such opportunities played a role in their weighing of a solution in comparison to others. It points to Hayes-Roth and Hayes-Roth [1979]'s findings on *opportunistic* planning behavior.

- **Assumptions**: They build the basis for a lot of queries in the planning process. When people were unsure about things, they made assumptions. Facing a query interface of a flight search engine the following was stated: "...normally Tallin is cheaper, the capital is always cheaper." Since the subject was not sure what city to select, an assumption was made.

- **Postponement**: When explaining the participants what to do, it was stated that they should plan the trip until **they** feel the plan is sufficiently laid out. It occurred that for things which are still part of the trip and might have needed some prior preparation, people tended not to take care of it until very shortly before departure. Most of the subjects were satisfied by having a flight and hotel booked. They were not concerned on how to exactly get from the hotel to the conference venue, or how to go to the airport in their hometown.

The observations point to several important aspects that can help for the design of personal assistant applications. **Information transfer** does highlight the sequential nature of planning, e.g., first choose a flight, then check the bus schedule. Task centered information management (see Section 2.3) does concern itself with such questions, and proposes the use of ontologies to auto-fill forms with information from other places [see Catarci et al., 2007; Katifori et al., 2008]. Secondly, a geographic knowledge base that stores relations or distance estimations between places (over multiple granularities), is essential for forming a plan that is unfolding in space and time. **Opportunity recognition** means that there needs to be a link between the objects involved in a plan and unrelated tasks or goals. **Assumptions** point to heuristics that are employed to reduce the search space and **postponment** implies that plans need to be stored partially or filled with *behavioral programs* or *scripts* for the parts that do not require advanced planning.

3.2.2 Representation of Plans in PIM-tools

Section 3.2 showed how people plan activities and what spatio-temporal aspects contribute to it. This section is going to explore the current state of PIM-tools, that store such plans. A short introduction of available tools and standards is given and followed by a qualitative examination of the second part of the user study.

3.2.2.1 Schedules and Calendars

Calendars or todo-lists are tools developed with the aim to store information externally, in order to ease the cognitive workload for remembering everything. Norman [1993] termed such tools *cognitive artefacts* and their main purpose is to extend our cognitive abilities. In times of mobile and ubiquitous computing scheduling tools became digitally available. Still, a study showed the acceptance of digital calendars is not high [Tomitsch et al., 2006]. In fact most people prefer traditional paper tools. One reason (amongst various others) for this, might be the fact that current digital calendar tools are in general *horseless carriages*[1], not really offering more capabilities than their analogue predecessors.

A schedule allows us to arrange and locate activities in an abstract time continuum, hence, functions as a reference system. Thus, calendar-tools[2] resemble topographic maps in their assistance to locate and navigate through activities, not in space, but in time. A major advantage of schedules is its reliance on a shared reference system (i.e., a calendar-system), supporting group coordination [see Crabtree et al., 2003; Hutchinson et al., 2003]. It is acting as a platform to locate activities in future to assist planning and prospective remembering. Common digital calendar applications usually allow for an alert to be set at specific points in time. A spatio-temporal context or dynamic agent behavior are not taken into account, thus the tools do barely exploit the sensing and computational potential of modern mobile devices.

The iCalendar-standard (Internet Calendaring and Scheduling Core Object Specification) specification is the most prominent format for storing or exchanging calendar and scheduling information across the internet. It nests within its top-component (VCALENDAR) its core calendar components:

- Events (VEVENT): contains properties describing an event
- To-do item (VTODO): contains properties describing an action or task
- Journal entry (VJOURNAL): contains a textual description for a calendar date, intended to contain reports about activities etc...

Possible properties for the description of events or to-do's, contain temporal information (in form of intervals, due-time or durations), priority values and alert-settings.

[1] http://en.wikipedia.org/wiki/BrassEraCar

[2] Calendars in this context refer to the paper or digital calendar-tools, and are not to be confused with calendar-systems, such as the Gregorian- or Islamic-calendar.

The nesting of components is not allowed, although a 'RELATED-TO' property can be used to depict relationships between different calendar components. Even so, it becomes obvious that the main purpose of the specification is group-scheduling and sharing of calendar information. Thus, the conceptual model of an activity itself, in terms of spatio-temporal granular representations, part-of relations between activities or requirements are not included. Further, the specification does not tell anything about how the data is to be used and it is up to the applications to develop additional *logic* to make sense of it. Therefore, often only a subset of the possible components and properties are implemented (e.g., the Google Calendar application does not support VJOUNRAL types).

3.2.2.2 Todo-Lists

Todo-Lists are very common tools used to enhance performance in respect to goal-achievement. It has been shown that todo-lists are kept in all sorts of places (i.e., emails, sheets of paper, etc...) and are mainly held in the form of textual cues taking the role of reminders, rather than formal descriptions [Bellotti et al., 2004]. It appears that such cues, if time management is involved, make their way into calendar-tools as well [Bellotti et al., 2004].

With the emergence of GPS-enabled smart phones, location based reminders were incorporated into various task management applications (e.g., RTM[1]). These alerts are mostly set off, when the user is close to, enters or leaves a certain area and are referred to as *Geo-fences*.

3.2.3 Shortcomings of Current PIM-Tools

To investigate the usability of PIM-tools, a second part of the experiment (3.2) looked at the way people represented the plans they formed in a common scheduling tool was examined. Thus, the participants were asked to store the information that they found necessary for their trip in a common calendar tool[2]. A short introduction to the tool ensured that every user knows all the possibilities it has to offer. There were no minimum or maximum requirements set by the experiment. Note, that the analysis here is completely qualitative; no quantitative data was produced or processed for this part. The observations that have been made form a major motivation behind the coming sections about how to represent future activities.

[1] www.rememberthemilk.com

[2] In this case the Google calendar application was used: http://google.com/calendar

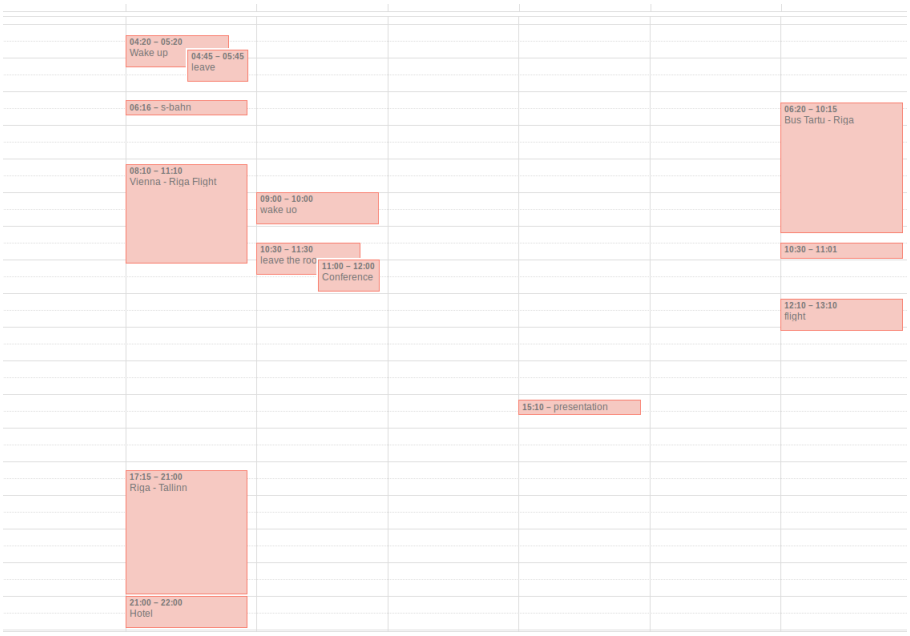

Figure 3.4: An example that includes flight times as well as platial information in the title fields.

What stands out, when looking at the outcomes is the arbitrariness of it. There is no definite way to represent activities. Depending on what aspect of an activity people found to be relevant, the form of representation was determined. For example, some stored a flight as an activity spanning from departure to arrival time (see Fig. 3.4) while others used two separate events standing for departure and arrival time (see Fig. 3.5) with no attention payed to the duration of the events (in that case a default of one hour is assumed by the application).

Several participants (mis)used the title field to include *place* information, like "Vienna-Tallin" or "Department of Geography". While there is an extra field for *place* in the event description of the calendar, it seemed to be more important to see where things happen on the visual front-end of the calendar.

Further, people put information of sub-events in textual form into the notes-field in the event description. For example, one participant added an event that represented the first day by bounding it with the registration start and end of the last session on that day. In the textual field of that event entry, though, information of the opening session was put (i.e., start-time, duration and location). Another person, in a more sophisticated use of the text-field, stored web-links into it. These links were leading

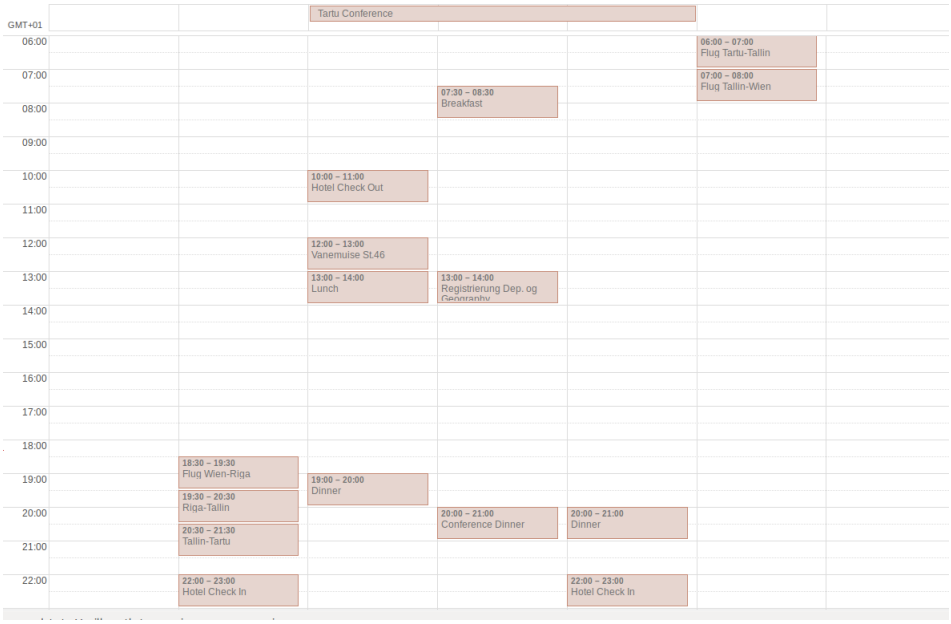

Figure 3.5: An example in which a flight is represented by two separate entities.

to another web-based PIM application (e.g., Evernote[1]) in which corresponding pdf-documents were stored (e.g., Booking-confirmations, Route-Descriptions, Conference Program).

Comparing Figure 3.5 with 3.6 reveals that calendar applications are in most cases an electronic reproduction of their analogue predecessors, save an alert or the adding of a location or place that can potentially link to a map. The potential lying in their use on sensor enabled mobile devices is not reflected in the functions or specifications (except for synchronisation of varying devices as for the ical-standard).

Concluding, from the small sample of people investigated it seemed that the calendar was mostly conceived as a platform to create visual mnemonics that helped to retain information about the planned activities. Very few participants went beyond that by conceptualizing the activities as a sort of *container* that is capable of carrying more information about sub-activities or other sources of information (i.e., documents, route descriptions). Only one participant used the alert functionality of the calendar in mobile devices for waking up in the morning.

[1] https://evernote.com/

3.3 Requirements of Future Personal Assistent Applications

This section lists main aspects a future personal assistant application should be able to handle. The listing is derived from literature and the findings of the reported user study in Section 3.2.1 and 3.2.2.

Unifying Events and Todo's

Using the terminology of section 3.1.1, tasks (as handled in todo-lists) are of the goal intention type, whereas events recorded in calendar-tools can be viewed as implementation intention (since they include answers to when and often where). A general problem of PIM-tools is *information fragmentation* [Jones, 2004] and refers to the increasing number of specialized solutions dealing with different forms of information. As opposed to paper-calendars where formal representations are not required (see Figure 3.6), current digital tools do not sufficiently tackle this issue, rather they tend to focus on one type or another (e.g., Google Calendar,Outlook for schedules, RememberTheMilk for tasks, TripIt for travel, etc...). The errand of *buy bread for dinner* is more flexible than the calendar event *attend a lecture from 1:00 pm to 3:00 pm.* This can lead to confusion about where and how to represent or store them.

Space, Place and Time as a Structuring Principle

Space and time are important structuring principles for human knowledge representation [Janowicz, 2010] and as prior work [Abdalla, 2012; Raubal et al., 2004, 2007] has shown, the abilities of task-planning applications can be enhanced by contextualizing tasks and events by their spatio-temporal dimensions. Time geographic concepts, such as space-time stations, space-time paths, trip-chains or space-time prisms play a significant role for reasoning about people's schedules and their translation into spatio-temporal movement [Raubal et al., 2004]. Many suggest [Alexander, 1979; Tuan, 1974, 1975; Whyte, 2012] that people conceptualize space into places ,i.e., discretize space. Just as calendar entities (i.e., July, Monday, 16th of March, etc...) are used to communicate about temporal concepts, places allow to communicate about space (e.g., meet me at the University). Therefore, places have to be in focus rather than space, leading towards a *Place-time Geography.*

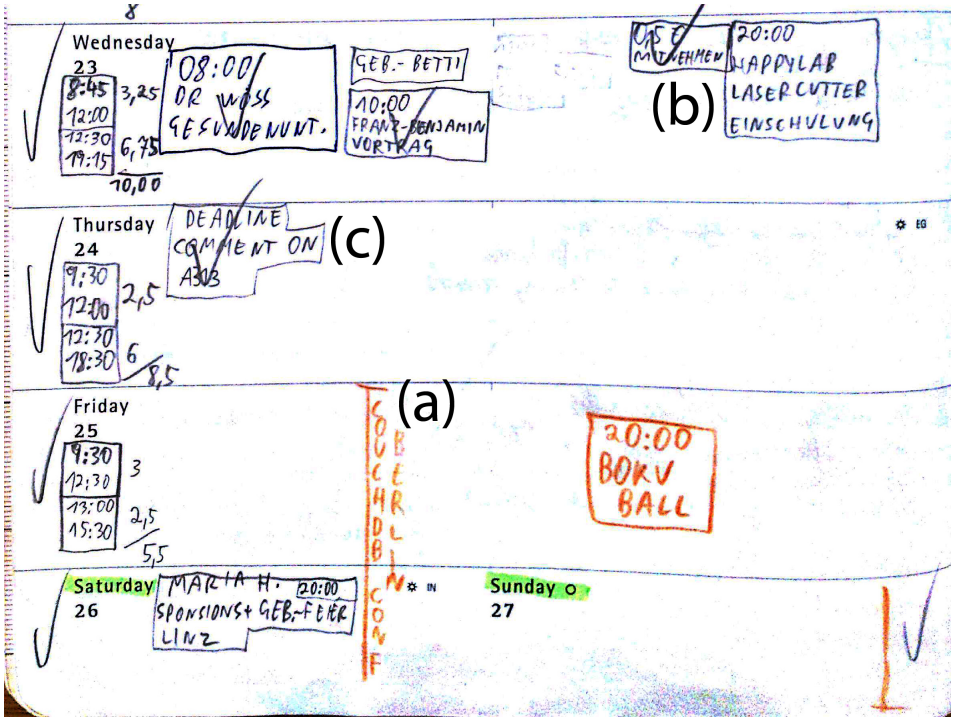

Figure 3.6: An example of a paper-calendar including a variety of entries. (a) A temporal interval, (b) An appointment with object-requirements, (c) A deadline. (The authors likes to acknowledge Markus Mayer for providing the above example of his personal schedule.)

Representation over Multiple Granularities

Plans are formed on different levels of detail [Timpf et al., 1992]. Participants often postponed the planning of lower level activities. It is therefore essential to give a user the possibility to store future intentions on coarser granularities, that allow for the recognition of left out lower level activities, to remind users about it if necessary. For example, planning to go for a hike in the mountains on the weekend, while a definite hiking trail and time has not been determined.

Activity Composition

The problem of aggregation is another issue, not addressed in common calendar tools. Activities are often composed of sub-activities, e.g., a conference is composed out of several conference sessions. Thus, part-of relationships do exist between activities that are ought to be made explicit in an accurate model.

Non Deterministic Nature of Intended Activities

A point worth discussing is the distinction between activities and intentions. Activities modelled in current tools are mostly assumed to be relatively well defined. *Intended Activities*, though, are part of a formed plan not always definite in their properties. It means that there might be several places or time slots that allow for the activity to be conducted and the user might not have determined yet where the activity is going to be undertaken.

Requirements

An essential criteria for inferring facts about activity sequences or constellations are the requirements that need to be met for an activity to start or terminate. A system capable of making automated inferences or compute solutions needs to provide information of pre- and post-conditions of activities.

Small Scale Objects

Many of our personal tasks involves the movement of small objects, like groceries, laptops, books and the like; "objects of human scale" [Reitsma and Bittner, 2003]. Even so, they are completely omitted from current formal task representations. In a recent paper, Goodchild [2014] suggested that *"...it will be possible to know where everything is, at all times"*. Considering RFID sensors or object recognition capabilities of future wearable devices[1][2] this assumptions is not too far fetched. Research in PIM also investigated the possibility of tracking real world objects [Câmara et al., 2008]. Hence, future systems might be able to incorporate object requirements into task representations.

Intelligent Alerts

Current alerts are statically set and do not take the spatio-temporal context into account. Consider the following (actually experienced) scenario. You scheduled a doctor's appointments after work. Because the doctor is only 10 min. walking distance from your office you set a reminder to go off 15 minutes before the appointment. Unfortunately, due to some circumstances, at the day of the appointment, you decide to leave work earlier and head home. In the afternoon, while at home, the alert triggers and you

[1] http://www.google.com/atap/projecttango/
[2] http://www.google.com/glass/start/

realize that you forgot about the appointment. Unfortunately, because you are not in the office any more it will take you at least 40 minutes to the doctor, thus rendering the task of reaching the doctor in time impossible. An alert taking movement and location into account, would have triggered at a time where the achievement of the appointment was still feasible. While a purely spatio-temporal alert is not very complicated to implement [see Abdalla, 2012], the question of how to deal with additional dimensions, like required objects for example is more challenging [see Abdalla and Frank, 2014].

The aspects listed are by no means exhaustive. Nevertheless, they pose a good starting point for the development of a next generation personal assistant application.

3.4 Summary

This chapter concerned itself with the question of how activities are motivated and planned. A user study investigated the properties and representations relevant for planning future activities. It highlighted the importance of spatio-temporal knowledge. It was found that common scheduling tools (i.e., that use the ical-standard format) do not offer sufficient formal representation capabilities to meet the demands of their users, which lead to the use of text fields to represent location changes, sub-activities or links to relevant data. The chapter concluded with a list of requirements a next generation personal assistant application should offer.

Chapter 4

Outlines of a Personal Assistant Application

This chapter introduces two scenarios, which are used to highlight the features a tool to manage personal geographic information can provide. The features listed are not meant to describe an exhaustive set of capabilities, rather it is intended to give a general idea of the possibilities that can be gained by merging virtual and real-world activities. For illustration purposes, screenshots of a prototype application are used.

4.1 Use Case Scenarios

4.1.1 Scenario A

In the scenario a person based in Vienna (Austria) is attending a conference abroad to give a presentation there. The Agile 2012 (Leuven, Belgium) conference[1], will serve as an example. In the scenario it is assumed that the person has planned out the main parts of the trip. A flight from Vienna to Brussel and a Hotel close the main train station in Leuven, was booked. The person has already registered for the conference. Some details, though, have not been laid out yet. For example, it was not determined how to reach Leuven from Brussel Airport. Hence, *gaps* in the process are present. Unrelated to the conference, there are two errands that need to be taken care of. First a book has to be returned to the library before a certain deadline, which lies within the period of the conference. Secondly, a souvenir is intended to be bought from Leuven. While

[1] http://www.agile-online.org/index.php/conference/conference-2014, last retrieval March, 2015

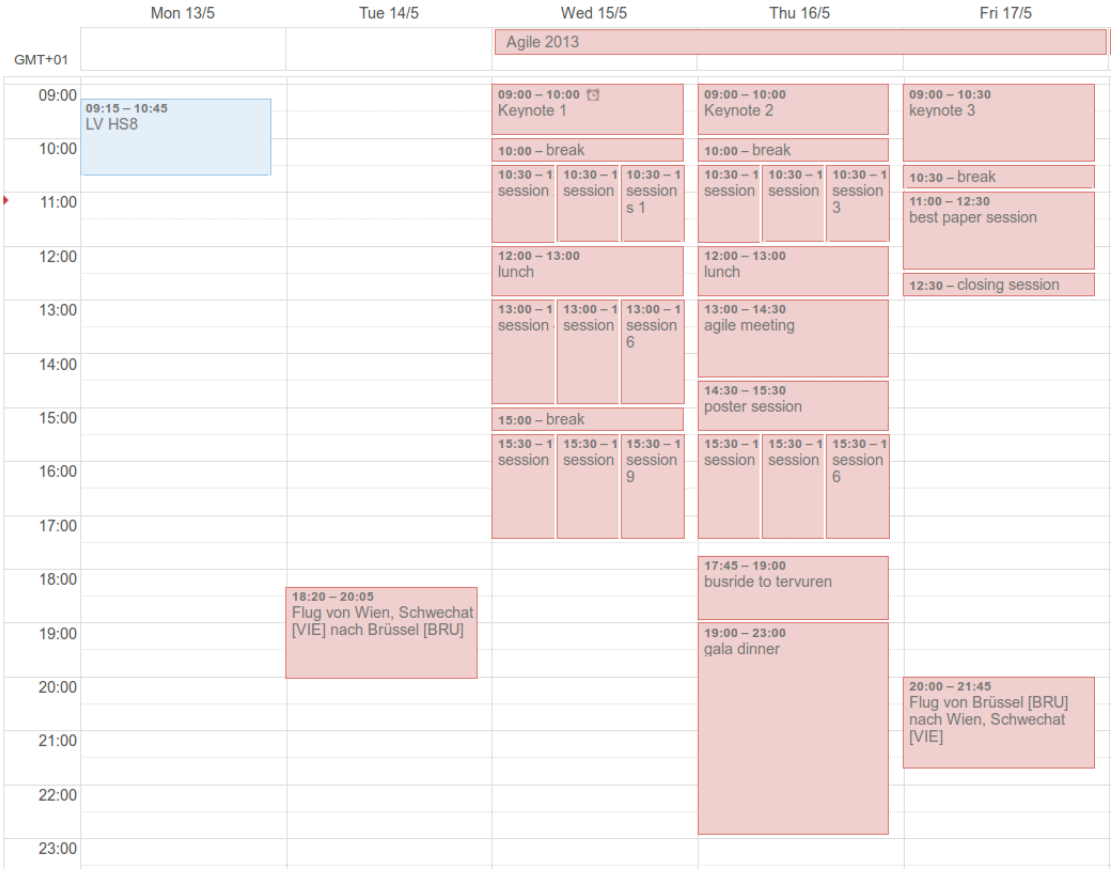

Figure 4.1: The program of the Agile 2013 conference represented in a common calendar application. Events arranged in parallel are spatially disjoint.

conducting the trip, the person produces personal information, i.e., taking pictures and notes at the conference. Figure 4.2 shows a common representation of the conference and flight, in a web based calendar application.

4.1.2 Scenario B

In this scenario a student has several events and errands mapped out for the day. It includes the attendance of several lectures, two in the morning and one in the afternoon. At noon the student meets a friend to return a DVD. In the afternoon, the student has a presentation, for which a laptop is required. Since the student ran out of bread, some has to be bought before returning home in the evening.

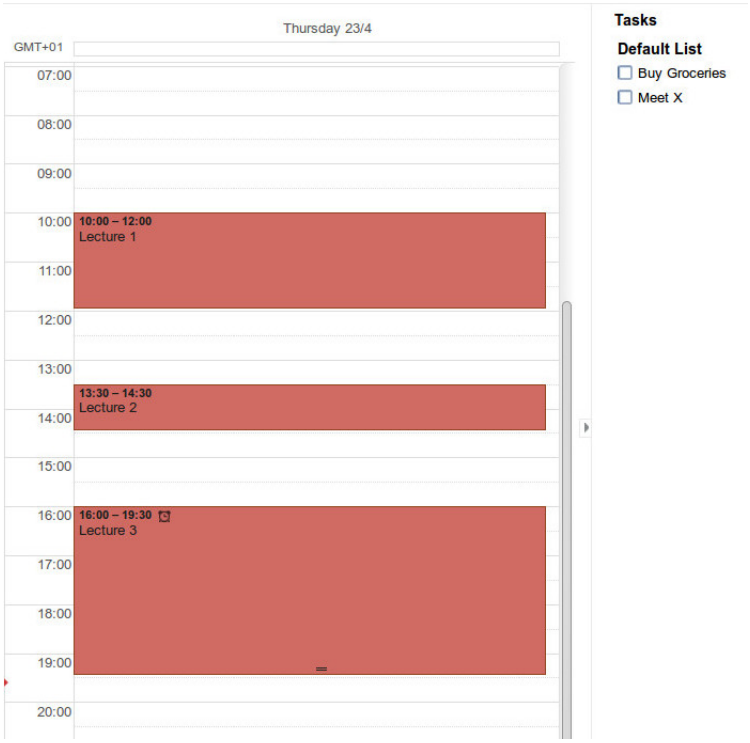

Figure 4.2: The three lectures and two tasks of Scenario B, represented in a common calendar application.

4.2 A General Picture of a Spatio-Temporal Personal Assistant Application

A personal assistant application has to take a holistic view on a user's life and be able to support her in various ways. It includes the management of personal information, but also contains support in planning or monitoring actions in the real world. A user's reality involves a location and movement in space-time, therefore different contexts and restrictions arise. Utilizing spatio-temporal constraints Raubal et al. [2004] suggested a theory for a decision support system which essentially gives the user instructions about how and in what order to finish the tasks set for a day. Contextual reminders were proposed that monitor user actions and trigger in *critical* situations [Abdalla and Frank, 2014; Prelipcean et al., 2015]. Hu et al. [2013] showed the usefulness of organizing personal information by events.

The mentioned work builds on a spatio-temporal perspective of activities, but there are many more properties that play into a conceptual model of activities. For instance, the purpose or meaning of activities that group them together to form hierarchies. In Scenario A (4.1.1) , for example, the conference is made out of several conference days that are made of presentation sessions. Activities are often requirements for others, like the registration before a conference. Activities are complex and cannot be captured in their entirety. What is of interest in the current thesis, are their projections into space and time, as well as into a personal information collection and their semantic relations (see Figure 4.3).

The aspects a personal (geographic) assistant application is ought to handle, spans from the support in planning activities for the *future*, to monitor the actions in the *present* and organize data produced by already conducted activities of the *past*. At the core lies a representation of activities that allows for a more holistic view on activities.

Figure 4.4 shows a graphical user interface of a prototype application that was developed to illustrate the possibilities gained by the utilization of space and time.

4.2.1 A Multi-Granular and Spatio-Temporal Representation of Activities

Places play an important role in human understanding of space, a personal assistant application has to be able to deal with the places relevant to a user. Gazetteers translate geographic coordinates into a textual description, i.e., names. A *personal* Gazetteer,

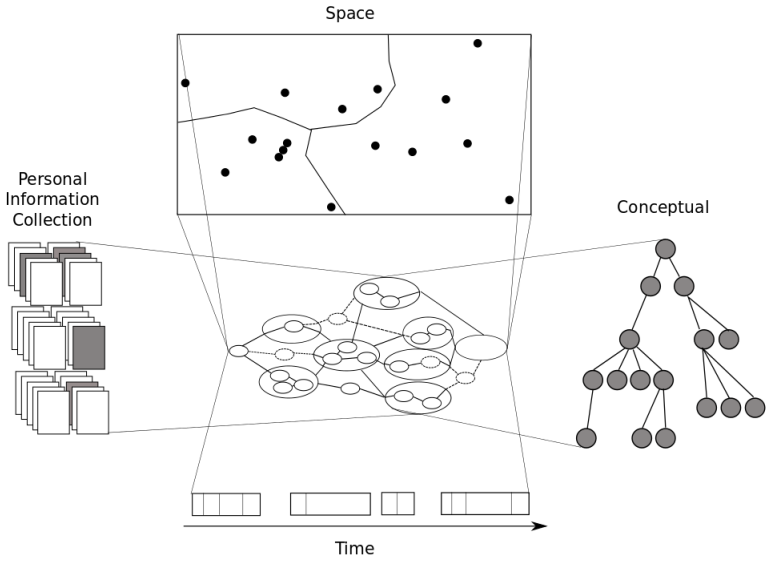

Figure 4.3: An abstract representation of activities (middle) allows to project properties into time and space, as well as a mapping into a conceptual hierarchy (right) and the personal information collection (left).

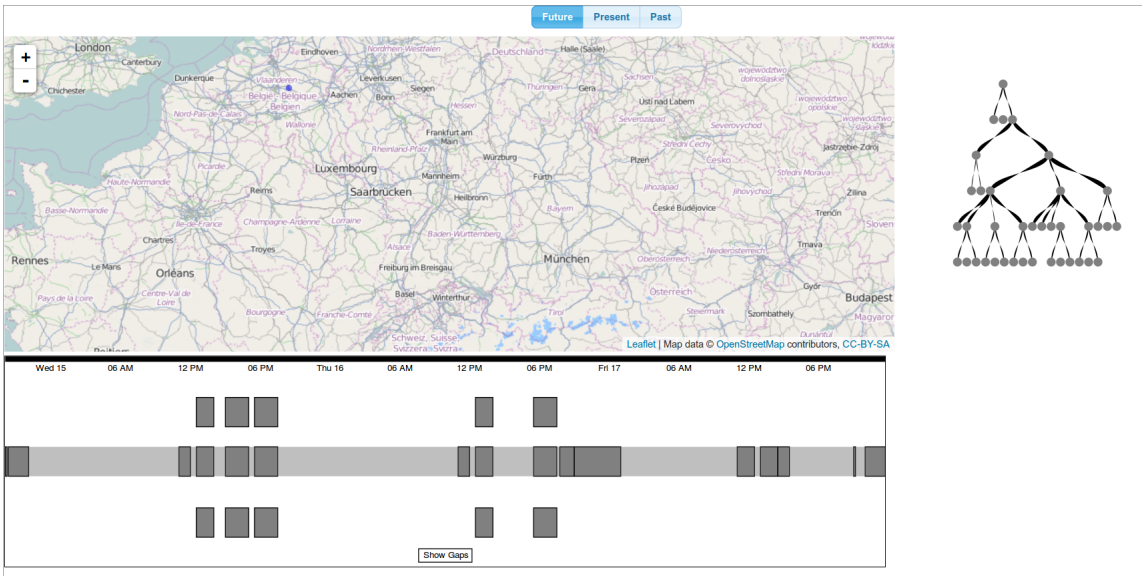

Figure 4.4: The GUI shows three perspectives of the same complex of activities, a spatial (the map), a temporal (a gantt-chart) and a conceptual (the tree shows how the activities and their sub-activities are grouped together).

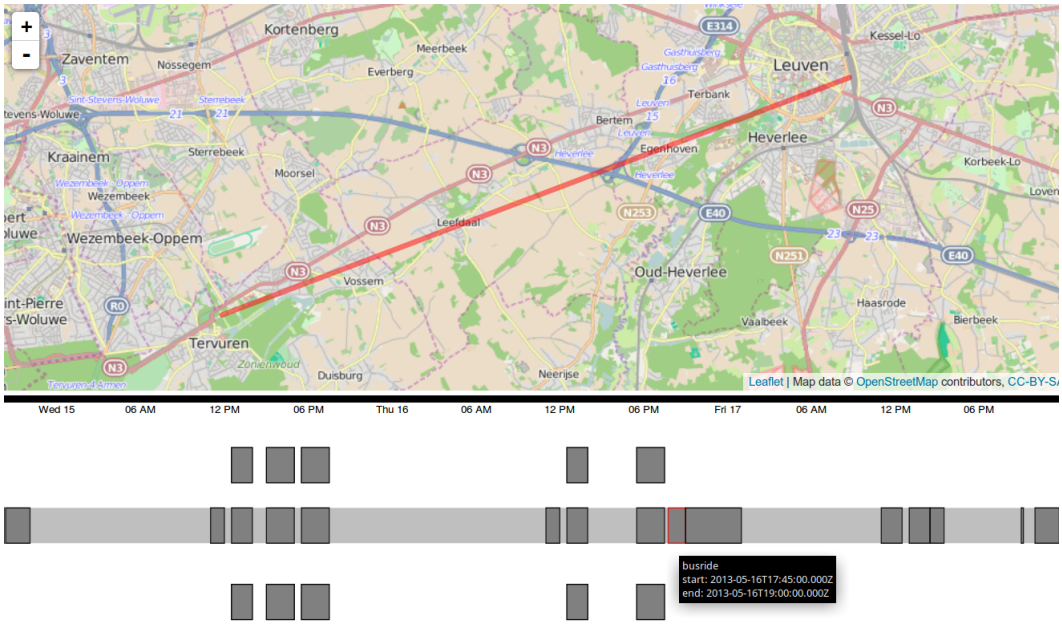

Figure 4.5: The conference in Scenario A (4.1.1) involves a bus ride from Leuven to Tervuren. It is critical information that current schedules do not allow to store.

can be build by asking the user to assign names to places, or by using clustering algorithms applied to the GPS-trajectories of a person [Zhou et al., 2004]. A place database is essential to describe activities in a way a user can understand and share the description. A useful representation of activities has to allow for flexible place configurations. Activities that involve movement can exhibit differing start- and end-places (see Figure 4.5). Errands can often be conducted at multiple locations (e.g., buying milk) (see Figure 4.6).

Research suggests that humans conceptualize space hierarchically [Montello, 1993], and places can therefore be of varying levels of detail. The activity of attending a conference (Scenario A (4.1.1)) involves several places that are of varying granularity. At the same time the activity itself exhibits a hierarchical structure, for example, there are several sessions included in a conference day, and several days make up the conference.

While humans can make these inferences easily, current applications do not. Including a granular representation of place in a personal assistant application, allows to represent activities imprecisely (see Figure 4.7, 4.8 and 4.9). For Scenario B (4.1.2) it allows the user to arrange a meeting at noon *around* the University (i.e., Neighbourhood level), which can be narrowed down later. In Scenario A (4.1.1) a multi-granular representation allows the system to support planning by first finding a route on country level, then on

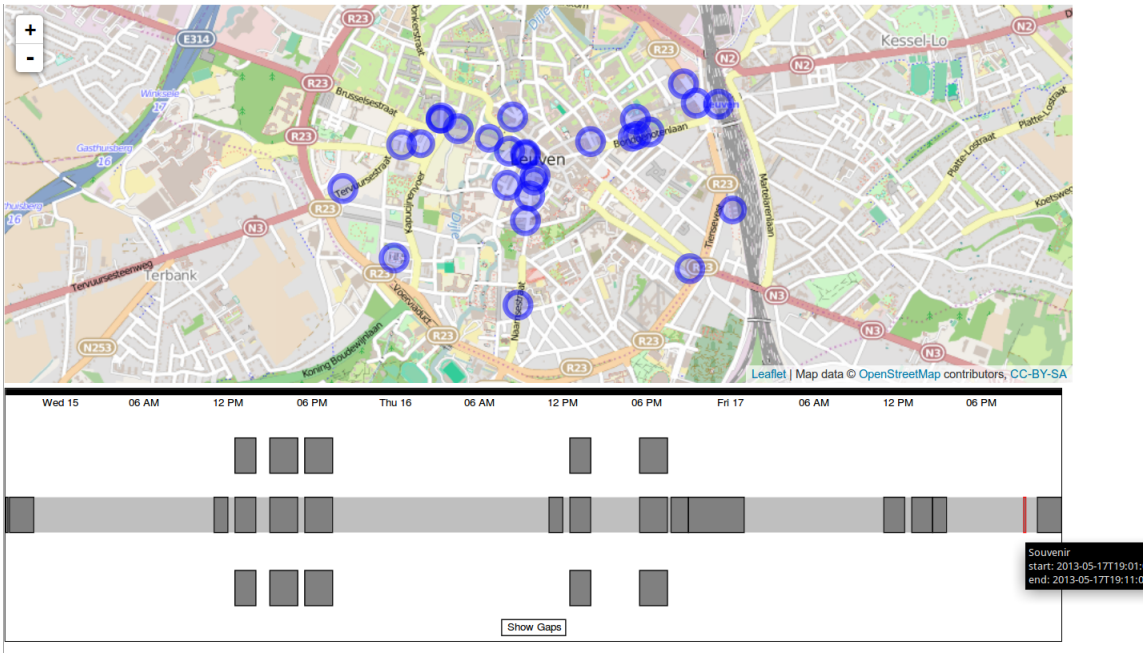

Figure 4.6: The errand of *buying a souvenir*, as depicted in Scenario A (4.1.1) is possible at multiple locations. By knowing the locations, a system can automatically fit the errand into a activity gap that fulfills the spatio-temporal constraints necessary.

city level and then within cities.

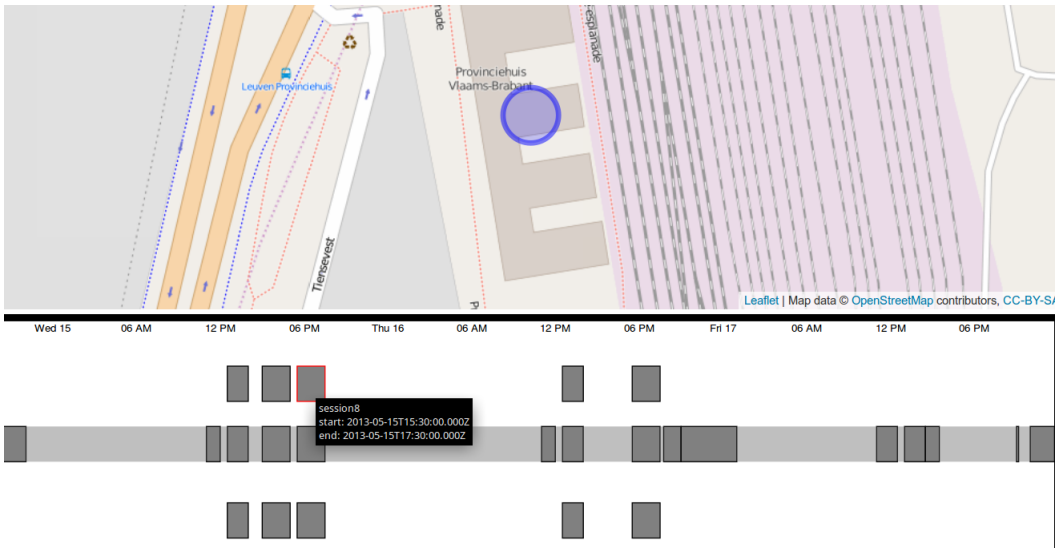

Figure 4.7: An individual session of the conference program depicted in Scenario A (4.1.1) . The place of the activity is given at room granularity.

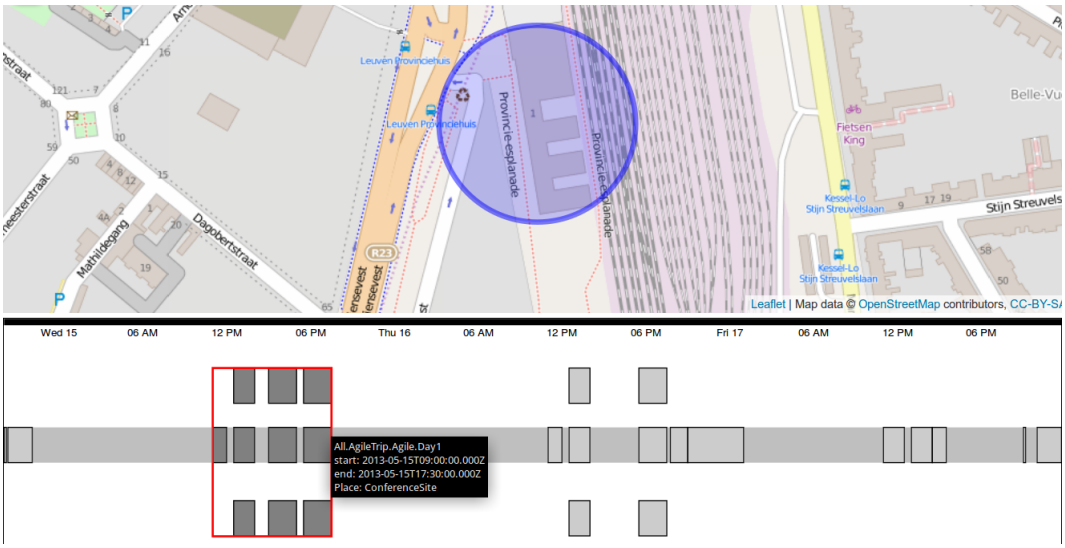

Figure 4.8: Looking at the complete first conference day of Scenario A (4.1.1) , the place that contains all places involved is the building.

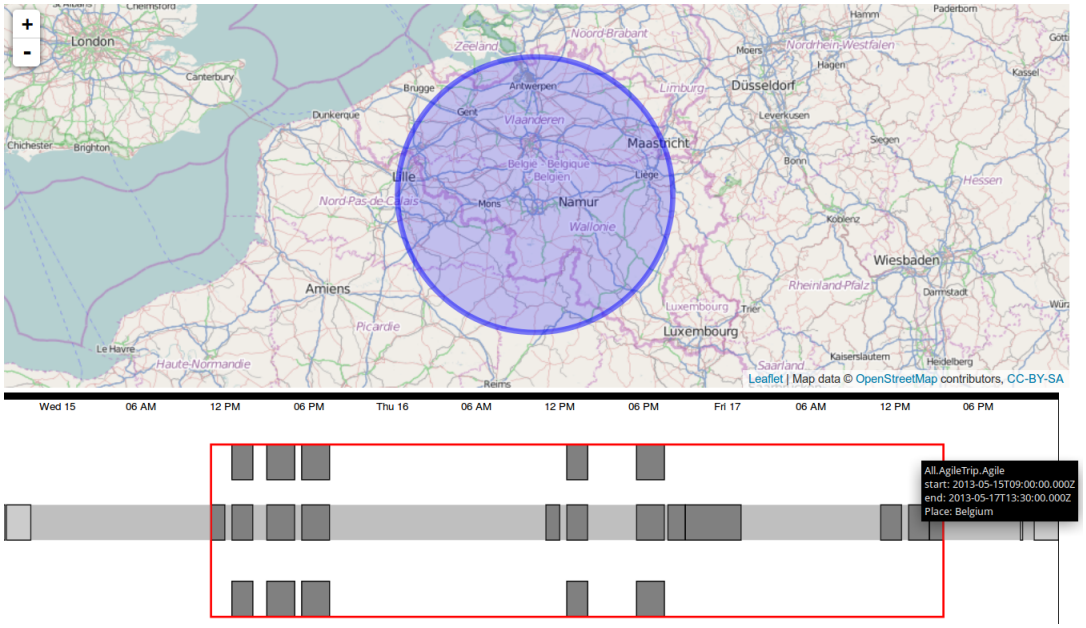

Figure 4.9: The complete conference takes place in Belgium, since some of the activities that are part of it happen in different cities (Leuven and Trevuren). Their containing place (depending on information in the system) is Belgium

4.3 Inferences about Composites of Activities

Schedules store intended activities sorted by time. The common schedule does not allow, though, to ask questions about a complex of activities, like the next week as a whole, or the "attending a conference" activity in its sum. This section discusses how a structured model of (intended) activities does provide the basis to infer useful facts for personal information management. Depending on the information interested, the assumptions that have to be made about user activities differ.

4.3.1 Interval and Point-based Assumptions

Bettini et al. [2000], in their discussion about temporal databases, distinguish between *point-based* and *interval-based* assumptions. The former is related to assumptions that are made at the same temporal granularity, for instance, one assumes that the amount in a bank account *persists*. It implies that if I check my account at 16:00 and the last transaction took place at 13:42, the amount returned will be the the same as by 13:42. It is therefore assumed that the amount *persists* unless explicit transactions are made that change it.

If one looks at an agent location property from the data stored in a schedule, for example, *persistence* is not a reasonable assumption. Asking for the position of an agent at 16:00 after an activity (e.g., attending a lecture) that ended at 15:00 pm, it is not reasonable to assume that the position is the same as the position of the prior activity. The range of possible positions though, is restricted to a subset of all locations by physical constraints.

Interval based assumptions, are being made when moving from one temporal granularity (e.g., hours) to another (e.g., days). Thus, the result is computed out of a number of entities that are in a certain relationship (e.g., containment) [Bettini et al., 2000]. A similar issue occurs in spatial domains, such as geography. Moving from a certain spatial grid containing sampling information, to a coarser one, demands a decision on how to assign aggregate values to the containing grid components (i.e., sum, average, etc...). For the case of modeling moving objects, Hornsby and Cole [2007] show that the aggregate value of potential moving spaces is dependent on assumptions about speed and other variables.

For the same data model, several assumptions can be made that define the outcome of a computation. Subsequently, when aggregating activities such *semantic assumptions* are essential. For instance, returning all the places involved in an activity sequence can be achieved by a simple union over all the activities at hand. All the places that are reachable within a given activity sequence, do require the computation of potential moving spaces and therefore is most likely a subset of the complete place set.

Looking at the objects required before an activity sequence is more complex. For instance, just because I need a book somewhere in the middle of the week does not mean I need it at the beginning of the week.

4.3.2 Inferences about Future Activities

The realm of personal information management includes information we have about our future, e.g. calendar, events, errands, appointments (see Figure 2.1). This kind of information is essentially produced as soon as we start planning. For proper planning of tasks, a spatial location adds constraints. Using current PIM-tools, one often finds that meetings are arranged properly in time, but the time for movement between the meeting locations had not been taken into account (not to speak of the difficulties to deal with arrangements which span multiple time zones). Integrating such constraints can be valuable when multiple people try to arrange meetings or appointments as shown

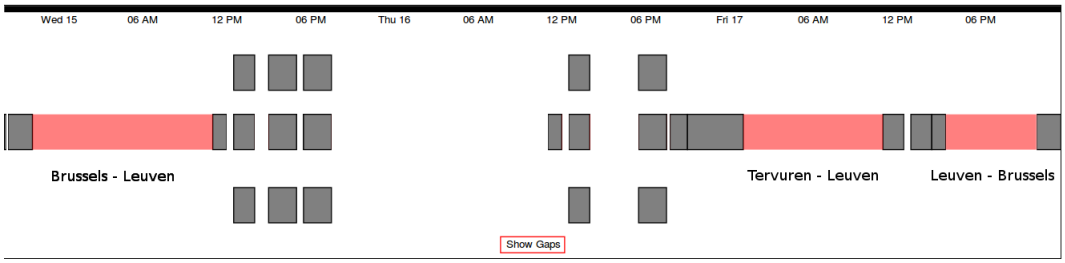

Figure 4.10: Each gray block stands for the timespan of a planned activity. By moving from the beginning to the end and checking whether a change of location is needed potential gaps in a sequence can be recognized. The red blocks stand for locational changes on city level, that are recognized by the system.

in the work of Espeter and Raubal [2009]. The list of features presented in section 3.3, included the determination of gaps in plans as well as the recognition of opportunities.

4.3.2.1 Determining Gaps

The determiniation of gaps in a plan can be partially achived by assuming persistence of location. In this case agent locations do only change when explicitly stated by an activity block. Having a sequence of activities the agent location can then be simulated by running through each block and see whether the transformed agent location does align to the starting location of the subsequent activity block. If the locations do not coincide, it can be inferred that a step is missing in the plan. A system can then fill such transformations by itself (e.g., by calculating a route) or remind the user to take care of it. Note that the familiarity and nature of the location change plays an important role. A location change from home to work, does usually not require a lot of planning and should therefore be filled automatically. A location change from one country to another usually involves the booking of a ticket or if taking a car the planning of the route, maybe the acquisition of a toll-ticket. These issues have to be dealt with when attempting to put such a system into practice.

In the context of Scenario A (4.1.1) a personal (geographic) assistant application can understand the spatial transformations that happen by the activities and detect the gap that exists between the arrival at Brussel-Airport and the conference in Leuven. Routes can be pre-fetched or a reminder to arrange some transport mean, given (see Figure 4.10).

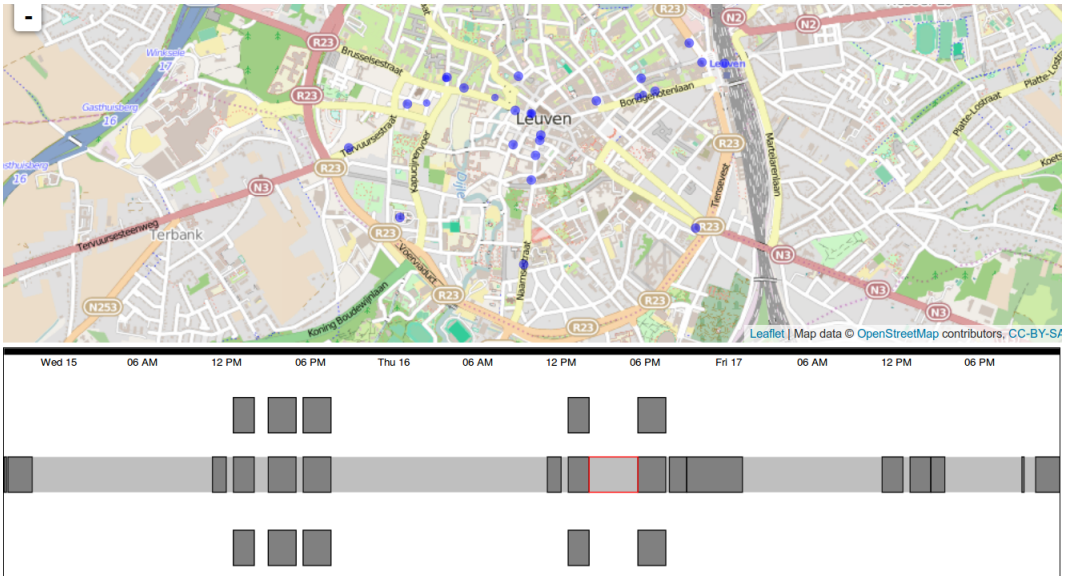

Figure 4.11: The potential places of type shops (blue dots) that a user can spend 10 minutes at and be reached by foot, between two actvities (red box).

4.3.2.2 Opportunities

The empty spaces mentioned between the activities in Figure 4.10 can be seen as gaps in a plan **or** be interpreted as spaces of potential activities (a.k.a freetime). A spatio-temporal representation of activities does allow to project the potential into a set of known places and look for other objects that are within the potential place-set. It allows to check whether errands involving objects can be achieved. The first necessary condition is that the intersection of the potential places with the places required for the errand is non-empty. Secondly, it needs to be checked whether one of the found places can be stayed at for at least as long as the errand takes.

Further, by using the activity information, it can be linked with spatial information stored in a person's information collection, therefore recognized as an opportunity to meet the person. All contacts that live in, or close to, Leuven can be retrieved. Even the combination of personal geographic information and spatial data acquired from outside sources is possible. The user can ask questions like: What is the weather forecast for that event?

4.3.3 Inferences About Present Situations

Sellen and Whittaker [2010] argue that too little effort is put on how PIM can help to support people with their prospective memory, that is remembering the things to do in future. Scheduling tools in most cases allow to set reminders, hence giving the user a hint about when things are going to start or end. More sophisticated reminder systems have been proposed. For example, a punctuality alert [Abdalla, 2012], which alerts the user when it is time to leave in order to arrive at a planned event in time[1]. Having spatial context an application can give a user hints about what needs to be considered before going to a specific place.

For example, it could remind the user to take into account parking restrictions, when going by car to areas in which such restrictions apply. Further, the system, given the necessary technical capabilities, can remind the user of objects need to be taken or brought to a place. A vision not too far fetched, given the current developments in RFID tagging and available applications (e.g., stickNfind[2]). In Scenario B (4.1.2) for example, the student would be reminded of taking the laptop in the morning, when leaving home in order to have it at the afternoon lecture. In Scenario A (4.1.1) the person is reminded to return the book to the library before going to the conference, since the deadline to return it is within the temporal bounds of the conference. Such reminders require extensive spatio-temporal information. Not only about the user's plans and tasks, but also about the environment in which the user is acting. Nevertheless, since additional information is becoming more and more easy to access (e.g.: API's, feeds, etc...) and computers become ubiquitous, as well as equipped with various sensors, it is a technologically feasible.

4.3.3.1 Supporting Prospective Remembering

Having an agent context, the possibility of conducting a specific intended activity can be monitored by looking at the time it is supposed to be undertaken and the current position of a user. Therefore, a *punctuality alert* [Abdalla, 2012] can be set. Such an alert is dynamically generated according to the position of an agent in space and time, as demonstrated in [Abdalla, 2012; Prelipcean et al., 2015].

The spatio-temporal perspective employed by Abdalla [2012] does require to monitor a user's spatio-temporal position in relation to the next intended event. Thus, the

[1] A function that is by 2014 part of the GoogleNow-application (https://www.google.com/landing/now/)

[2] https://www.sticknfind.com/sticknfind.aspx

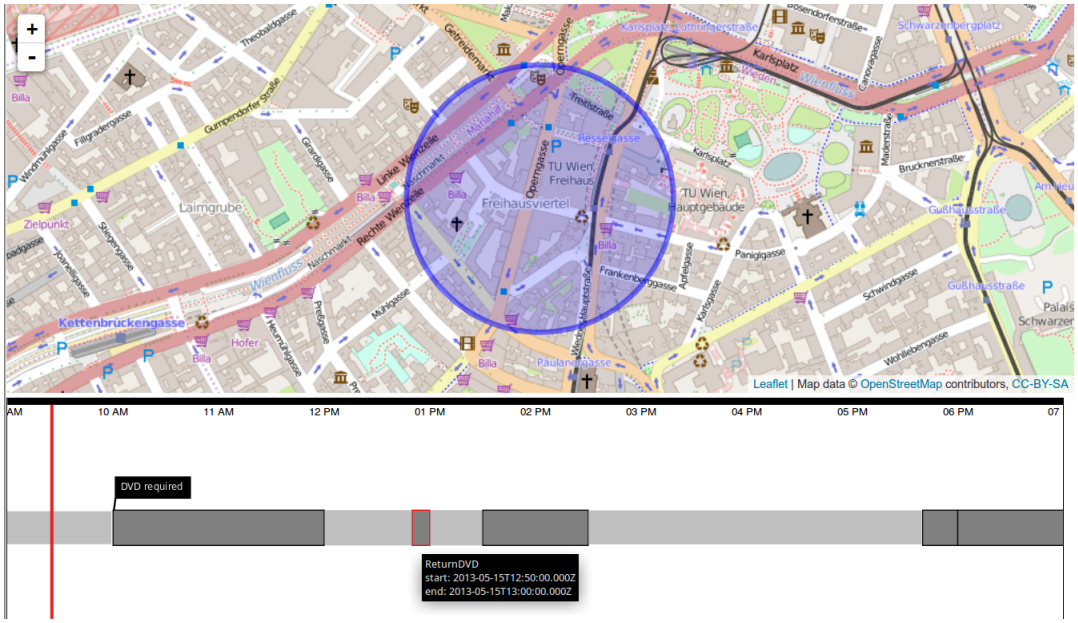

Figure 4.12: Because an object is required in the middle of the day, and the time to move from the activity before it to acquire the object is not sufficient, the requirement is propagated to the prior one. It allows to set context sensitive reminders.

aggregate of activities is not relevant. Adding object requirements renders the approach unfeasible. For instance, in Scenario B (4.1.2) the student has to return a DVD to a friend, after a lecture was attended. Given the DVD is located at home, and assuming that it is impossible to get the DVD from home and move to the meeting place after the lecture, the requirement of the meeting (i.e., the DVD) has to propagate to the earlier event (i.e., the lecture) (see Figure 4.13). To achieve meaningful propagation of objects, additional information, telling what will be done with an object, is required. If an object has to be *dropped off* it will not be with the agent after the activity was completed. If it is *picked up* it does not have to be with the user before, but will be in the users possession afterwards. An object that is *maintained*, has to be brought to the activity and remains with the user after it. This information can be utilized to produce intelligent and context sensitive reminders. The student in Scenario B (4.1.2) can be alerted in case an object that is required in the afternoon is forgotten to be picked up in the morning. Such an alert is only possible if the schedule as a whole is taken into consideration Abdalla and Frank [see 2014].

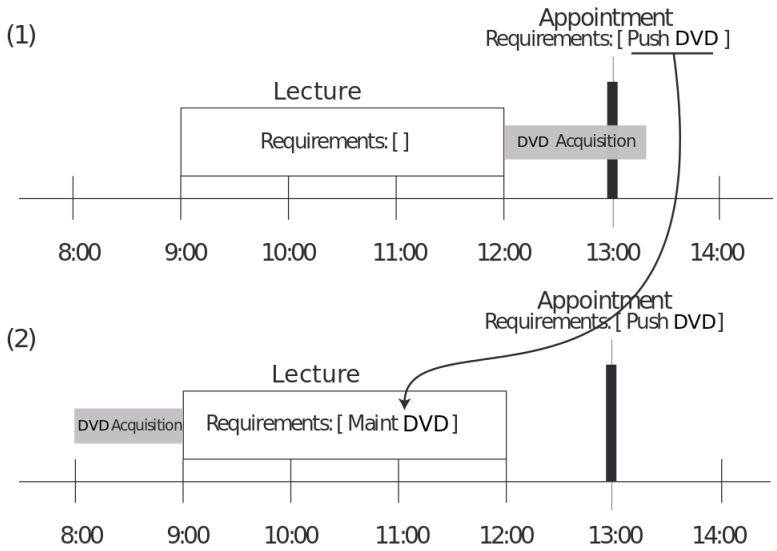

Figure 4.13: (1) The time it takes to pick up the DVD after the lecture does exceed the time available. (2) A "Maint DVD" is added to the requirements of the first event, meaning that the DVD has to be acquired before and maintained until the end of the lecture. (Source: [Abdalla and Frank, 2014])

4.3.4 Activity Driven Personal Information Retrieval

A main question addressed in PIM, is that of information retrieval, or the process of *finding, keeping, re-finding*, i.e., how do we acquire information and how do we store it best to easily retrieve it at need later on [Barreau and Nardi, 1995; Jones and Teevan, 2007] (see Section 2.2).

PIM researchers offer differing solutions to the problem of re-finding, among them are: *searching everything* (through the use of tags), *structuring everything* (by the means of database schemas), or *unifying everything* (using RDF). The search approach, inspired by the success of web search engines, requires the documents to be tagged with as many cues as possible, in order to tackle the problem of contextual recall; while the question of how to represent the meaning of the document is unclear.

Spatial data, as opposed to ordinary data, offers the ability to join data by their spatial proximity. Thus, things that otherwise would not have been join-able are so, through their spatial properties. Current Geographic Information System (GIS) products offer a vast number of operations for performing spatial analysis. Questions such operations can answer are, for example, of topological nature, such as point set-topologies [Egen-

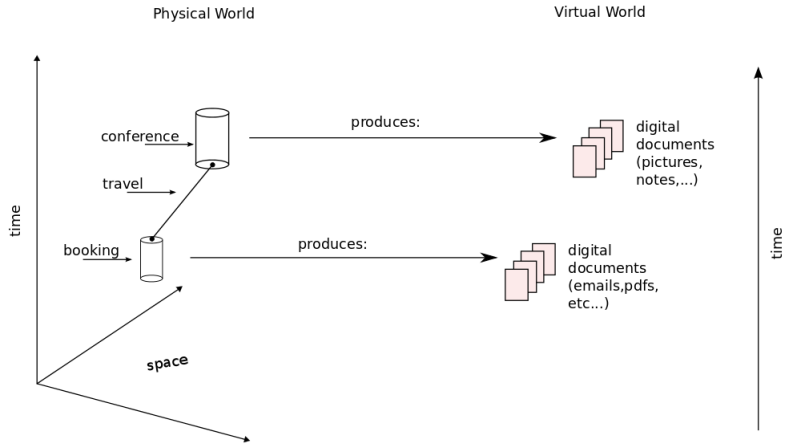

Figure 4.14: The booking activity has to happen in a range of space-time states that allow to reach the conference states. Therefore the same must hold for the spatio-temporal timestamps of the digital documents produced by the activities.

hofer and Franzosa, 1991]. A personal geographic assistant application would enable a user to ask such questions about his own data, i.e., what are the documents that are contained in this country?

The core argument made in this work is that the structure of our activities is reflected in our personal information space. Therefore, structuring our personal information along those activities does give additional cues that can help retrieving information or even extracting knowledge about it. Every piece of digital information produced or received in virtual activities, can be related to space, since it is part of or driven by a physical outer activity. The spatio-temporal structure of the physical activity is reflected in the structure of the documents used and produced by the virtual. For example, to attend a conference abroad, some travel arrangements are usually necessary. These arrangements are (nowadays) mostly done online and produce documents, such as travel itineraries or booking confirmations. Because the flight has to happen before the conference the travel itinerary document of the flight cannot become part of the personal information collection at a spatio-temporal point during or while the conference activity (see Figure 4.14).

4.3.4.1 Inferences About our Personal Information Collection

By having a representation of intended activities, the actual movement of an agent can be matched to such models and be given a meaningful interpretation.

The conference attendee in Scenario A (4.1.1) can retrieve the information produced by taking notes in a session or taking pictures on the trip, based on either the activity or the places the information was produced in. It basically can provide map-based and/or time based query abilities to re-find personal information (see Figure 4.15 and Figure 4.16). Another form of organizing principal would be similar to a project undertaken by [Buckland and Lancaster, 2004] which provided location-based queries for library content and proved to be beneficial. It was achieved by geo-referencing the information according to it's content. The project proved that map-based searches can in some cases improve the finding of information.

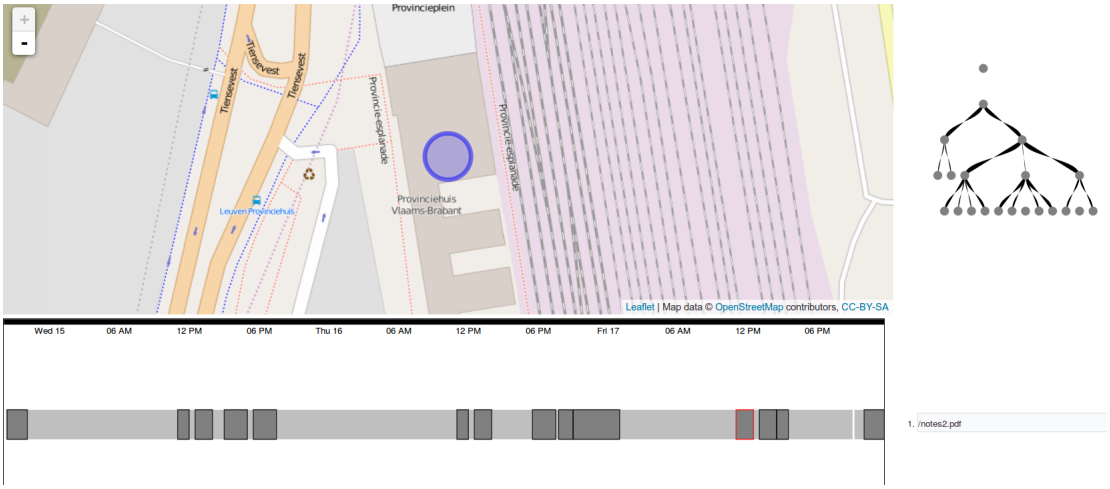

Figure 4.15: Past actvities in a schedule can be used to structure and retrieve documents related to them. The notes (bottom right) that were taken are highlighted based on the activity (red).

Having information about the semantic structure of activities are a key asset in finding data in a personal information collection. For example, all the data produced throughout the activity of attending the conference can be retrieved by aggregating them over the sequence of every single sub-activity (see Figure 4.16).

Thus, the spatio-temporal constraints of physical space do impose a logic on the spatio-temporal occurrence of personal information objects. In the age of mobile computing it will be essential to structure the information around places as well as time, so *space*-time-stamps will be common ways to enrich meta information of documents on operating systems.

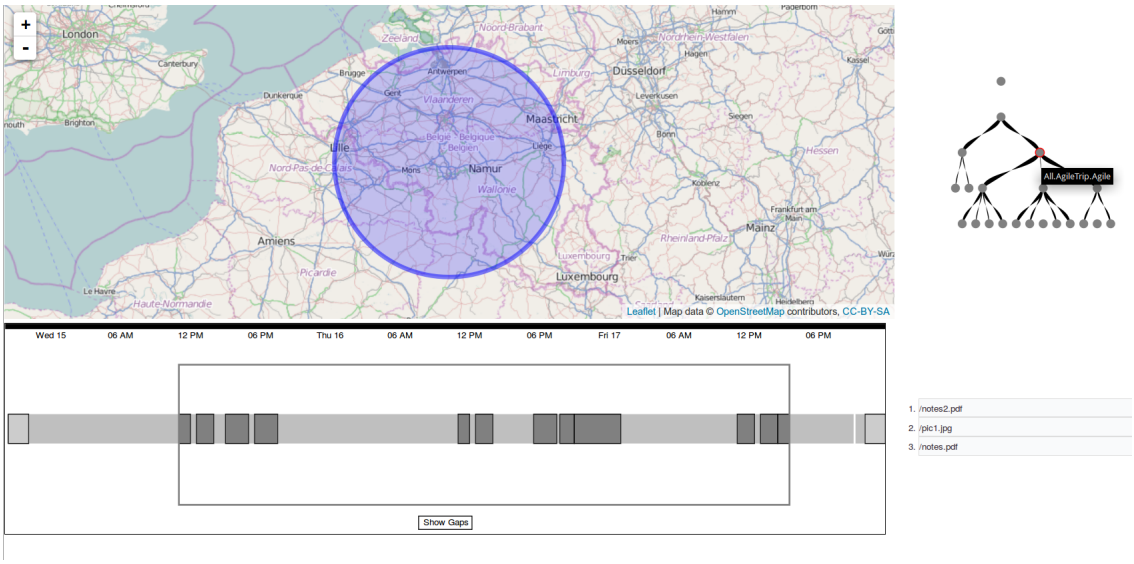

Figure 4.16: Having a hierarchical structure of activities and their sub-activities allows to retrieve agrregates of information produced by the particular sub-activities (bottom right).

4.4 Summary

The current chapter introduced some of the capabilities a personal assistant application can gain by utilizing place and time as a basis to describe activities. Two use case scenarios were introduced and a prototype application (described in Appendix 9.2) illustrated the functionalities. The core capabilities described were:

- Place-based representation of activities, that allow for the expression of movement, or multiple places;
- A Multi-granular activity representation that involves spatial, temporal and conceptual granularity;
- Spatio-temporal consistency checking of plans;
- Recognition of opportunities by integration potential action spaces;
- Merging of errands and events by consideration of spatio-temporal constraints;
- Context sensitive reminding, by inferring requirements from a set of activities;
- Activity driven information retrieval, linking personal information to space.

The list has to be understood as non-exhaustive and represents only some of the possibilities that arise from the integration of spatio-temporal activity representations.

Chapter 5

Granularity in Place, Time and Tabletop-Objects

Section 2.4.2 and 2.4.1 discussed the role *calendars* and *places* play in human conceptualization of space and time. A crucial point in those discussions are the use of hierarchical orderings in both domains. Chapter 3 emphasized, among other points, the need for a multi-granular model to represent activities. Therefore, this chapter is dedicated to the notion of *granularity* in general and gives a formal definition of a system of granularities. Further, system of granularities for *place* , *time*, *table-top objects* and *activities* are discussed.

5.1 Granularity

Section 2.4.1 mentioned Schatzki's [1991] assumption that places are part of larger entities, and the discussion of calendars (Section 2.4.2) showed that mankind constructed *groupings* of temporal entities to conceptualize time. Such groupings, point to notions of *scale*, *level of detail* or *granularity*. They play into a plethora of fields including AI and spatial information theory [Frank, 1997; Hobbs, 1985; Reitsma and Bittner, 2003; Sacerdoti, 1974; Timpf et al., 1992]. Furthermore, granularity was recently proposed as a *core concept* of spatial information [Kuhn, 2012].

Hierarchical conceptualizations are inevitable in order to tackle the complexity of a practically infinite number of objects existent in the world [Hobbs, 1985]. Computer science, for example, exploits this fact in a number of data-structures to reduce computational effort for retrieving or storing data [Comer, 1979; Samet, 1984]. Section 3.3

showed that conceptualization and planning of intended activities does happen over multiple *levels of detail* or *granularities*. Thus, granular representations are an essential part of human cognition.

Hobbs [1985] defined *granularity* by using an *indistinguishability relation* $\sim$, such that if $x \sim y$, then x and y are indistinguishable in regard to a set of properties describing them. It basically partitions the world into equivalence classes according to a domain of interpretation. For example, dishes on a restaurant menu can be grouped into a *granularity* by simply looking at their containment of meat or fish, distinguishing the dishes into two groups: *vegetarian* and *non-vegetarian*. In that case a lamb-curry and spaghetti-bolognese are two granules of the same granularity (non-vegetarian), i.e., are equivalent in respect to the $\sim$ relation. Introducing further properties, like *spicy* or *non-spicy*, subgroups of the two prior groups can be created, that are of a *finer* granularity. At that level, the equivalence between the lamb-curry and the spaghetti-bolognese does not hold anymore. In other words more complex theories about objects, lead to finer granularities.

A considerable challenge lies in the shift between granularities, hence adding or removing relevant properties for the distinction. Moving up in *grain size* (or LoD) is expressed by Hobbs [1985] in form of a mapping K that transforms a complex theory into a simpler (coarser) one. So it relates all the classes on a certain granularity into the equivalence classes on a coarser one. A crucial aspect in the above description of granularity is *partitioning*. Bittner and Smith [2001] claim that *granular partitioning* is a universal tool employed to categorize, divide up, or sort the world. They are often used to divide continuous domains into discrete units [Bittner and Smith, 2001].

To Bittner and Smith [2001] granular partition, are *cells*, possibly nested and of different grain size, with *objects* located inside. Those cells are arranged in a certain structure that is determined by the logic of the partitioning. The structure can be flat (e.g., a simple listing of rooms and their occupants) or hierarchical (e.g., administrative geographic units). Granular partitions are employed by humans and therefore can reflect bona-fide (objects existent independent of human activity) as well as fiat objects (objects mentally constructed by humans) [Smith, 1995]. The domain of interest determines the structure of the partitioning (i.e., geographic administrative units or biological species classification).

Based on the assertions mentioned above, we define *Granularity* and a *System of Granularities* as follows:

Definition 5.1.1: Granularity

A granularity $\mathcal{G}$ is a set of non-decomposable entities indexed by a set I (e.g., real numbers, strings, etc...), $\mathcal{G} = \{g_i | i \in I\}$. An element $g_i \in \mathcal{G}$ is called a *granule*. Each granule maps to a domain $\mathcal{D}$.

A granularity is a set of things (called granules) that are indexed. Each of these granules has some correspondence in a domain $\mathcal{D}$. What the domain $\mathcal{D}$ is, depends on the phenomena modeled (e.g., space or time).

Definition 5.1.2: System of Granularities

Let $P(\mathcal{G})$ be the set of all partitions of $\mathcal{G}$. A *system of granularities* $\mathbb{SG}$ is defined to be a subset of $P(\mathcal{G})$, with an ordering-relation $\leq$ defined upon it. Further, there exists a greatest upper bound $\mathbb{SG}^{Top}$ and a greatest lower bound $\mathbb{SG}^{Bottom}$, and for every partition $\pi_1, \pi_2 \in \mathbb{SG}$ it holds that: $\pi_1 \leq \pi_2 \iff (\pi_1$ is a refinement of $\pi 2)$. Transitivity holds for the $\leq$ relation. It forms a lattice of refinement partitions.

In words, a *system of granularities* as defined in this work, is a set of *granularities* that form a mathematical *lattice of partitions* (compare to definition in [Gill, 1976, p.151]) under an ordering relation $\leq$. In practice each granularity can have a label assigned to describe it (e.g., City, Building, Hour, Week). Defining a *system of granularities* in such a manner ensures several capabilities:

- **ordering**: given two granularities $\mathcal{G}, \mathcal{H} \in SG$, the ordering relation $\mathcal{G} \leq \mathcal{H}$ allows to discern whether one is finer/coarser than the other;
- **containment**: Because, the granularities in $\mathbb{SG}$ form refinements, there exists a containment relation $\trianglelefteq$ that tells whether a granule $g \in \mathcal{G}$ is contained by a granule $h \in \mathcal{H}$ where $\mathcal{H}$ is a coarser granularity ($\mathcal{G} \leq \mathcal{H}$).
- **coarsening**: given a granule $g \in \mathcal{G}$, a coarsening operation $\lambda :: \mathcal{G} \to \mathcal{G}$ can be implemented that returns the containing granule(s) of coarser level(s);
- **shared upper granule**: given two granules $g \in \mathcal{G}$ and $h \in \mathcal{H}$, a coarser granule $k \in \mathcal{K}$ can be inferred, such that $g \trianglelefteq k$ and $g \trianglelefteq k$ (in worst case k is the element of the top-granularity);

The next two sections will interpret the definitions in the context of space and time.

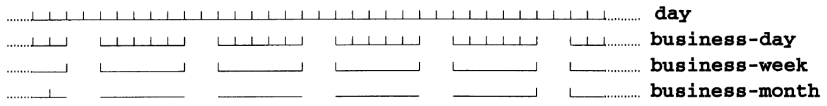

Figure 5.1: An illustration of how contiguous and non-contiguous granularities are build. (Graphic borrowed from [Bettini et al., 2000, p.7])

$(T; \leq)$	Time Domain
G	Granularity
$G(i)$	Granule of G with index (or label) i
$G \trianglelefteq H$	G *groups into* H
$G \trianglelefteq H$	G is *finer than* H
$G \sqsubseteq H$	G is a *subgranularity* of H
$G \tilde{\subseteq} H$	G is *covered by* H
$G \leftrightarrows H$	G is *shift equivalent* to H

Figure 5.2: Possible relations that can hold between granules (from [Bettini et al., 2000, p.18])

5.2 A Temporal Granularity Systems

Calendars are socially constructed [Searle, 1995, see] *granular partitions* of time. The following introduces basic notions of Bettini et al. [1998]'s algebraically constructed temporal granularity system. A temporal granularity system, according to them, consists of a *temporal domain* T, i.e., a non-empty set of time instants, with a total order defined upon it $\langle T, \leq \rangle$. It forms the basis for the construction of *granules*.

Temporal granules are nondecomposable entities that contain *time instants*. An index i preserves the ordering of the temporal domain. A granularity can be defined by a single instant, an interval or non-contiguous instants. A school-term, for example, consists of school-weeks, that are made of five rather than seven days. Thus, the mapping, which maps from the integers i into the time domain T is injective. Figure 5.1 gives a visual illustration of it.

An *algebraic* granularity system requires constructing functions that define the mappings between granularities. Bettini et al. [2000] distinguish between *group-oriented* and *granular-oriented* functions. While the former describe how granularities are built out of grouping together *finer* granularities (e.g., Weeks out of Days); the latter define how granularities are constructed out of the same granules (e.g., Business Weeks from Weeks). For example, the $Group_{60}(second)$ operation defines the *minute* granularity since it groups together each 60 second intervals into minute entities. An example on how to build a calendar then is:

- minute = $Group_{60}(second)$
- hour = $Group_{60}(minute)$
- day = $Group_{24}(hour)$
- week = $Group_{7}(day)$
- Monday = $Select - down_{7}^{1}(day, week)$

The example omitted the *month* granularity, since the altering length of months and leap years introduces complexity (accommodated by the *altering-tick* operation) that exceeds the introductionary scope of this section.

In brief, an *algebraic* granularity system as proposed by Bettini et al. [2000] can be used to produce descriptions of general calendar systems, but also to create other granularities, like *school terms*, or *business weeks*.

5.2.1 Temporal Granularity for a Personal Assistant Application

For describing activities, it is enough to assume that a granular calendar system exists and refinement relations are defined upon it (Figure 5.2 lists the possible relations). Note that the definition of calendars by Bettini et al. [2000] is very flexible and allows for structures that do not adhere to the restrictions imposed by the definition of a system of granularities in section 5.1.2. For example, the introduction of a *week*-granularity breaks the role of *years* as the *top* granularity, since the set of years do not form a partition of the set of weeks (i.e., there exist weeks that do not entirely fall into a single year).

For the rest of this work the `xsd:dateTime` [1] description that can be decomposed into the common (gregorian) calendar granules (see Fig. 5.3 (b)), will be utilized. This format imposes a total ordering on the granularity system, by leaving the week granularity aside.

5.3 Spatial Granularity

Hierarchies play a fundamental role in human conceptualization of space. It was shown that people employ hierarchical reasoning when inferring facts about spatial entities,

[1] http://www.datypic.com/sc/xsd/t-xsd$_{d}ateTime.html$

Figure 5.3: (a) The structure of a spatial multilevel categorization, illustrated with an address description of a room at Tu Vienna. (The granularity classification is that of [Richter et al., 2013]) (b) A *datetime* description split along the common calendar granules.

Type	Description
Figurative Space	Smaller than the body and can be perceived from one position without considerable locomotion required. It is the space of small objects, pictures or distant landmarks.
Vista Space	A space that is larger than the human body. Perceivable by standing still in one place. It includes rooms, small town squares, etc...
Environmental Space	Larger than the human body and surrounds it. It requires movement and accumulation of information over time to create a mental image of it. Buildings, neighborhoods or cities.
Geographical Space	Is considerably larger than the body. It cannot be comprehended easily and has to be learned through symbolic representation (e.g., maps). It includes states, countries but also the solar system.

Table 5.1: One possible distinction of scale-levels based on perceptual cognitive properties, defined by [Montello, 1993].

even when it distorts them [Stevens and Coupe, 1978]. Lakoff and Johnson [2008] provided evidence that humans commonly use a *container* metaphor when talking about space, thus leading to an inclusion-relation that, due to its transitivity, can produce hierarchies [Frank, 1996].

The exact number of *spatial levels* used by humans is disputed, and claims in literature range from four to eleven [Couclelis and Gale, 1986; Freundschuh and Egenhofer, 1997; Kolars et al., 1975; Montello, 1993; Richter et al., 2013]. Table 5.1 lists an example of a four leveled distinction proposed by Montello [1993], based on cognitive perception. Regardless of how many and what the actual levels are, a cognitively sound representation of activities over places, has to be able to handle spatial granularities.

Because humans employ a multi-leveled perspective on space, formal systems allowing to communicate spatial information over multiple granularities were developed (e.g., address systems, administrative boundaries). Arguably, there are similarities between temporal and spatial granularity systems as illustrated in Figure 5.3, which compares two common-place temporal and spatial multi-granular representations, i.e., system of granularities. The main difference, is the lack of an ordering in the space, that is, the euclidean space (i.e., R^2 or R^3). An example for 2D-space, are (thematic) maps and what is referred to as *categorical coverages*, which can be interpreted as (possibly) multi-granular representations of space. Chrisman [1982] (quoted by [Frank

et al., 1997]) defines a categorical coverage as "an exhaustive partitioning of a two-dimensional space into arbitrarily shaped zones which are defined by membership in a particular category of a classification scheme". Frank et al. [1997] show that, by aggregating or refining categories, *families of categorical coverages* can be produced.
The important point is that the spatial areas of categorical coverages are defined by the partitioning of thematic attributes. The aggregation or refinement of the attributes results in an aggregation or refinement of the spatial zones. In contrast, a chloropleth map does start from the spatial zone, and attribute values are produced from the values contained in a zone, for example, statistical units. Families of categorical coverages form mathematically proper partitions and refinements and are therefore exhaustive, meaning that each aggregated category is the union of its refinement. Spatial subdivisions as found in administrative units (e.g., Political Districts - Counties - Country) are often engineered to adhere to the axioms of refinement partitions. Determining whether a set of granules is a refinement of a coarser granule, is in that case a function of *spatial* containment.
Address systems, in comparison, are spatial granularity systems that are socially constructed and do not have *one* general rule to determine what granularity is finer than the other. For instance, the street-granularity is coarser than the streetnumber-granularity, which does not mean that the spatial extent of a street contains the extent of the entities (i.e., buildings or parks) associated to the streetnumbers (see. 5.3(a)). Thus, in such systems containment- or inclusion-relations do not always follow an obvious generic rule and are often defined *by hand*.
Moreover, human conceptualizations of space defy the mathematical properties of refinement partitions. A person does not have an exhaustive spatial representation of a city in mind. Only places of *relevance* are memorized. It means that, space itself is not so much of importance, rather discrete chunks of space forming *places*, are the objects of inquiry. Still, humans are capable to discern that the city itself is more than the simple union of all the places known.
Earlier in the section, a system of granularities was defined as a lattice-structure with a top and bottom element (see Definition 5.1.2). The temporal granularity system defined by Bettini et al. [2000] used *seconds* as an example bottom granularity. All other granularites can be traced back to this granularity. For a spatial granularity system that is based on the domain D (being euclidean space), *cells* can be seen as the equivalent bottom granularity. Henceforth, all other granularities are sets of groupings of cells [1].

[1] A difference is that seconds are well defined and standardized entities, while there is no such

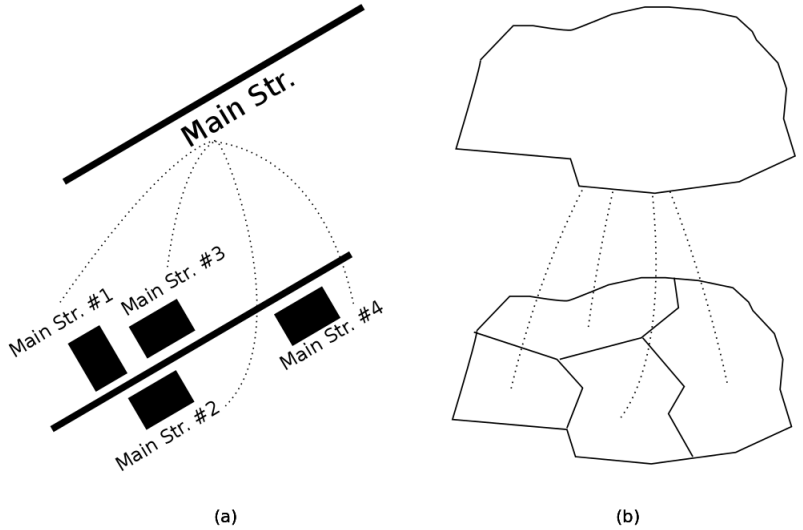

Figure 5.4: (a): The conceptual containment relation as used in an address system. (b): The spatial containment relation as used in administrative granular representations.

5.3.1 Spatial Granularity for a Personal Assistant Application

One of the features of a personal assistant application, as envisioned in Section 4, is the management of a personal gazetteer. It is assumed that the amount of places stored in such a personal gazetteer is finite, and that it forms a spatial system of granularities. It behaves similar to an address-system (see Fig. 5.3), in which each higher granularity is made up of the sum of the lower granules. While such a system is instantiated over a finite set of places; it provides the capability to add new places. The spatial extent of each granularity, though, is not determined by the lower granularities (i.e., the extent of a street is not given by the extent of all the buildings on that street)(see Figure 5.4). It means that the structuring in such a system does not (necessarily) happen through geometric means, but through predefined relationships, usually correlated to spatial relationships, but not determined by a single spatial containment rule. Because the relationships are independent of spatial containment, not all bottom granules (i.e., cells) will fall into an equivalence class on a coarser granularity. It therefore gives the user freedom in assigning *conceptual* containment relationships between places.

To ensure a personal gazetteer does combine into a lattice of refinements, an extra

standard for cells. Nevertheless, just because there is no such standard established it does not mean that they are not conceptually similar.

equivalence class, a *remainder*, has to be introduced that includes all the granules of the finer granularity that are not covered by the rest of the granules.

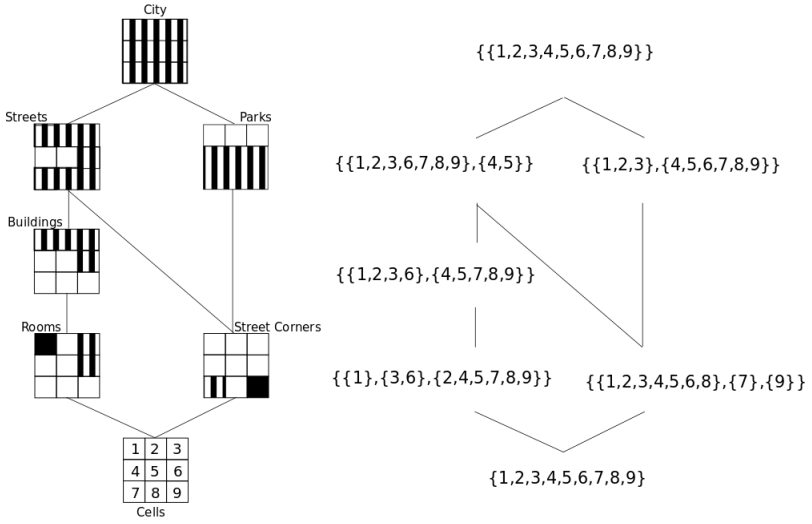

Figure 5.5: Left: A graphical illustration of a hypothetical system of spatial granularities that form a lattice structure of refinements as demanded in the original definition. The bottom granularity is a set of 9 cells which are sequentially grouped into equivalence classes in the coarser granularity, the white cells are grouped into a *remainder*-class. Each grid represents a *granularity*. Right: a corresponding lattice of refinements using integer values to represent the cells.

5.4 Tabletop-Objects and their Granularity

In the previous sections it was shown that space and time can be structured into a granularity system, by essentially defining how things relate to each other in terms of their temporal, spatial or conceptual containment. The same logic can be applied to table-top objects in *figurative space* [Montello, 1993], in which objects are generally smaller than the human body. This scale is in general not of interest to Geography or GIScience, but they play a considerable role in day to day human activities and therefore are of interest here.

Table-top objects are spatial objects and therefore can be embedded into the spatial granularity system discussed previously. Rooms contain table-top objects and therefore can be considered a coarser granularity than table-top objects. But there are other

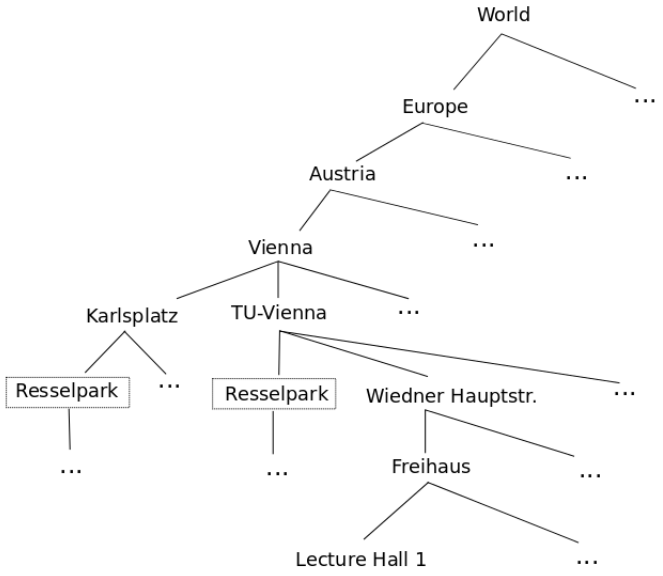

Figure 5.6: An implementation of a granularity system in the form of a tree. The granularities are not necessarily distinguished by the level of the tree, since it is possible to find two differing granularities at the same level (in this case Resselpark of granularity *Park* and *WiednerHptstr.* of granularity *Street*). Further, overlaps are allowed, as in the *Neighbourhood* granularity, where both *Karlsplatz* and *TU-Vienna* contain *Resselpark*. Hypothetically, each node would have to have all the cells stored that are not part of the containing granules. In practice this is not necessary, since each node (granule) can have a geometric representation associated to it, which implies what cells (represented by coordinates of a certain precision) are part of it and which not.

features that distinguish them from the entities organized in the previously discussed spatial system of granularities.

In relation to human lifespans most of the places stored in a spatial granularity system stay the same and are therefore useful to communicate location. Table-top objects do not share this property. A spatial or temporal system of granularities is a relatively fixed schemes in which granules are placed to discretize a continuous phenomena. The schemes are either well established, as in the case of calendars, or at least investigated, as for the case of space [Couclelis and Gale, 1986; Freundschuh and Egenhofer, 1997; Kolars et al., 1975; Montello, 1993; Richter et al., 2013]. There are no comparable schemata to structure table-top objects in the way geographic objects are.

The notion of granularity for such objects are either related to the type or to containment of the objects. Objects can, for example, be abstracted into categories, like groceries, clothes or stationary. Or, due to their spatial properties, into hierarchy of containment. A wallet usually contains money, bankcards or IDs. A jacket can contain a wallet, a back-pack contains many other smaller things, and so on.

5.4.1 Table-top Objects in Personal Assistant Application

The role of table-top objects in an activity descriptions, is not to communicate location, rather it is *requirements* (e.g., Keys, Money, Documents, etc...). Requirements are, in most cases, given at a very specific level, like the exact cost of something, or the exact document you need. Due to the dynamic nature of table-top objects, a notion of containment for table-top objects is not a useful for describing requirements. For example, adding a back-pack as a requirement to an activity does not give valuable information about the objects that need to be taken to the activity.

A hierarchical typology of objects, though, can be useful to restrict the set of objects to be considered, i.e., adding *groceries* to an activity requirement does exclude many other objects. The information can be used to narrow down places relevant for acquiring *groceries*.

5.5 Activities and Granularity

The goal of this chapter is to develop notions of place-, time- and table-top object-granularities, that can be used to describe activities in a personal assistant application. It allows to describe the activities at different levels of detail, and move between them.

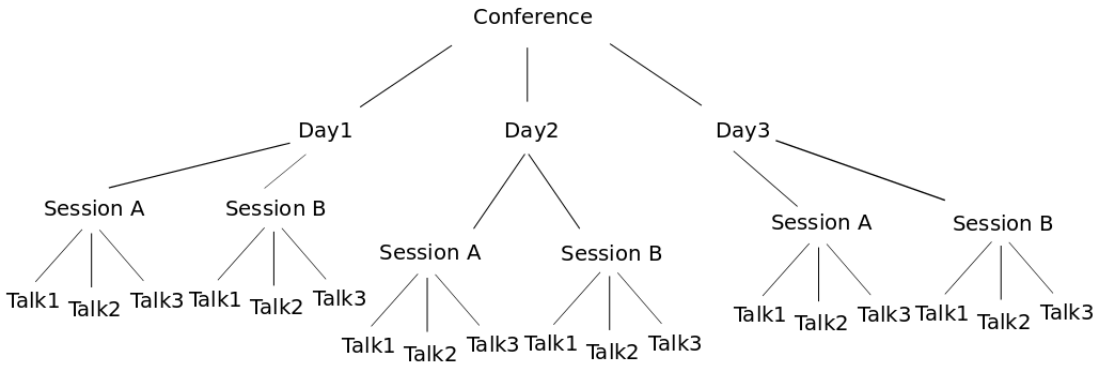

Figure 5.7: A common structure of a conference program represented in a tree.

Section 3.2 showed that activities are composed of sub-activities, and therefore form a hierarchical structure. The conference in Scenario A (4.1.1) , for example, is composed of keynotes, sessions and social events (see Figure 5.7).

Activities can form a hierarchy in which nodes are *semantically* disjoint from each other, but spatio-temporally intermingled. For example, meeting a friend in the lunch break of work, is not related to the activity of work, but temporally falls in between it.

5.5.1 Activity Granularity in a Personal Assistant Application

A system that wants to keep track of activities has to separate the spatio-temporal structure of activities from the conceptual one. While the spatio-temporal arrangement of activities can be automatically inferred by the system (due to physical constraints). The interpretation of the activities is up to the user and forms a layer on top of the spatio-temporal structure. A mapping between the two has to exist, so that the corresponding places to a certain activity, or the activities related to a place or time can be retrieved.

5.6 Summary

The current chapter shed light on the notion of granularity in general, and discussed its meaning in four domains: space, time, table-top objects and activities. It stated that system of granularities are social constructs and subject to change and variability, depending on the culture or society producing it. To allow a user flexibility in the choice of a system of granularities, a formal definition was given (i.e., a lattice of refinements).

It ensures that crucial operations are possible, independent of the detailed make-up of the system of granularities.

Chapter 6

Describing Activities in Place and Time

The past developments in mobile computing led to the use of time geography as a framework to model and reason about human activities from a user-centered perspective [Raubal et al., 2004]. It allows to answer questions like: Can I do this after that? Modeling activities in a *space-time cube* is useful to describe the spatio-temporal structure of activities, but is not suitable to represent activities in a personal assistant application. Its description of activities in continuous space and time does not correspond to the way humans conceptualize and communicate about it. Thus, additional layers that abstracts the continuous view of time-geography into a semantically more meaningful model is suggested. This will allow to represent *intended* or planned activities, essential concepts that need to be stored and reasoned about in personal assistant tools.

In the following a general framework for the definition of activities in *place* and *time* will be outlined.

6.1 Ontological Commitments

Reality is complex, and in order to build theories and models about it, the number of factors taken into account have to be constrained. This work concentrates on agents, places, temporal concepts and small scale objects. These are the core concepts from which intended activities will be described. Thus the ontology commits itself to certain facts:

- An Agent has a state, i.e., a location in space and time and can carry a set of objects;
- An action taken by the agent has immediate consequences, i.e., dropping an object or moving, does change the world instantaneously;
- There exist a common temporal granularity system that has an ordering;
- There exists a finite set of places that are structured in terms of spatial containment. They are located in space and contain objects that can be picked up or stored.

6.2 Agent Movement in Place and Time

The aim of this work is to propose a formal definition of an *intended activity*. Therefore, a multi-granular model of place and time is developed that forms the basis for a description of activities from an agent perspective. An agent-state a representing an agent that moves in space and time is build on the basis of a set of places and a set of temporal granules:

Definition 6.2.1: Spatio-temporal Agent

A spatio-temporal agent is represented by a tuple $a \in \mathcal{A}$, in which $\mathcal{A}$ has at least 2 dimensions: $\mathcal{A} = (\mathcal{P} \times \mathcal{T})$. $\mathcal{P}$ represents the set of place granules known to the agent and $\mathcal{T}$ a set of temporal granules.

An agent-state $a \in \mathcal{A}$ is a pair of place and time, similar to the model of spatial lifelines Hornsby and Cole [2007]. It emphasizes the *user-centred* perspective the model confines itself to. Nevertheless, there are two key features that distinguish the proposed model to that of spatial lifelines: (1) the use of places; (2) the use of granularities. More precisely:

Definition 6.2.2: Agent State-Space

Let $\mathcal{P}$ be a set of all the places structured in a spatial system of granularities $\mathcal{SG}^{Place}$, and $\mathcal{T}$ all the temporal granules structured in a given temporal system of granularities $\mathcal{SG}^{Calendar}$. A (spatio-temporal) agent-state-space $\mathcal{A}$ is the set of all possible states an agent can take by building the cross product $\mathcal{P} \times \mathcal{T}$.

The following notation will be used: $time(a)$ to return time $t \in \mathcal{T}$ and $place(a)$ to get a place $p \in \mathcal{P}$. Further, an ordering relation $\preceq$ that stands for "is possible before" and its converse $\succeq$ "is possible after" are introduced.

Definition 6.2.3: isPossibleBefore-Relation

A relation $\preceq$ holds true for two agent-states $a_i, a_j \in \mathcal{A}$ if a transition from a_i to a_j is possible. For $s, p, e \in \mathcal{A}$ it holds that:

- $s \preceq s$ and $s \succeq s$ (Reflexivity);
- $(s \preceq p, p \preceq s) \Rightarrow (p = s)$ and $(s \succeq p, p \succeq s) \Rightarrow (p = s)$ (Antisymmetry);
- $(s \preceq p, p \preceq e) \Rightarrow (s \preceq e)$ and $(s \succeq p, p \succeq e) \Rightarrow (s \succeq e)$ (Transitivity);
- s is comparable to p, if and only if $time(s) \neq time(p)$ or $s = p$

Thus, $\preceq$ defines a partial ordering over the set $\mathcal{A}$ and therefore forms a poset $\langle \mathcal{A}, \preceq \rangle$. In practice, such a relation requires an environment representation that contains information about travel times between places and (later on) objects that can be obtained at them (e.g., a graph that contains object-sets in its nodes).

Note that for the place-set $\mathcal{P}$, only a subset, namely the granules that represent entities of the granularity type is used. It means that the cells that are put together into the *remainder*-partition (see Section 5.3) are not part of the set $\mathcal{P}$. This is to avoid the situation that a state transition from a certain place to another place is always true, since the *remainder* usually neighbors every other place. So in the model an agent can only be at a place that is known as such.

Excluding the remainder partition and the use of non-decomposable temporal granules, results in a break of continuity. It means that, by using the relation $\preceq$ *place*-time paths can be constructed, that skip certain time-steps. While in a field based view, the extend of the potential path area, i.e., the 2D projection of a space-time prism, grows by a continuous expansion, in a discrete (e.g., graph based representation) environment there are holes in space. Thus, it is possible to reach places at time-step t without crossing places at time-step $t-1$. Vice-versa it means, there are places at time-step t that cannot be reached from time-step $t-n$, while they could from $t-m$ with $m > n$. Figure 6.2 provides a schematic illustration of the above statements.

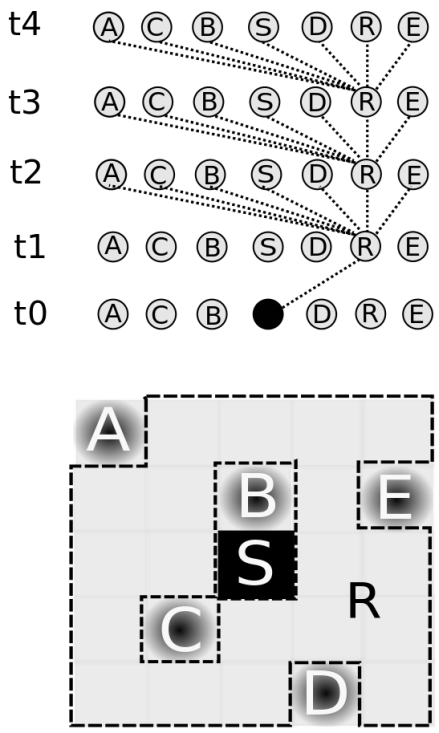

Figure 6.1: The remainder R was introduced to maintain the mathematical properties of a lattice of refinements. Keeping it in a place-time model to simulate agent movement, can potentially lead to the creation of state-transitions that have no assertive value, since they are always possible. In the example above, it is possible to move from S to every other place in only two time steps by passing through R, since it R is non-decomposable.

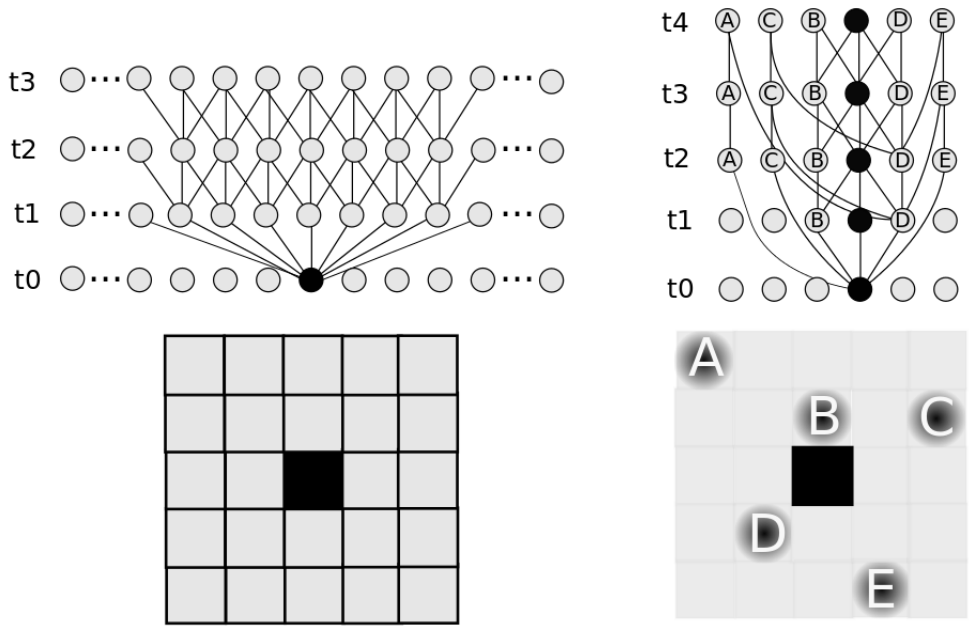

Figure 6.2: A schematic visualization of the state-space created by two differing views on the environment. (a) The state-space in homogeneous and isotropic space. For an agent to reach any point at t_n a point at t_{n-1} has to be passed. (b) In a "place" based model relations between non-contiguous time steps can exist.

6.2.1 Describing Activities as Agent Movement or Movement Potential

Having an agent representation and a state-space in which an agent can be located, describing an agent activity is a matter of extracting subsets of the agent-state space $\mathcal{A}$, that are ordered under the relation $\preceq$. Such descriptions can be viewed as equivalents to time geographic concepts, only in the discrete set of states. The first concept is a place-time path (see Figure 6.3 (a)), that is, a sequence of agent-states that describe the transitions an agent went through.

Definition 6.2.4: Place Time Path

A place time path $\mathcal{PTP}$ is a non-empty set $\mathcal{PTP} \subset \mathcal{A}$, that is totally ordered $\langle \mathcal{PTP}, \preceq \rangle$, and contains a lower bound $\mathcal{PTP}^{bottom}$. It holds that:

$$\forall a \in \mathcal{PTP} : \mathcal{PTP}^{bottom} \preceq p.$$

Further, the temporal granularity $\mathcal{G}^{temporal}$ used in the agent-states is constant:

$$\forall a \in \mathcal{PTP} : time(a) \in \mathcal{G}^{temporal}.$$

A place time path $\mathcal{PTP}$ describes the transitions an agent went through and, therefore, is a description of a past activity. If activities in the future are of interest, not every state transition is known. Therefore, a place-time lattice (see Figure 6.3 (b)) is introduced. It gives the possibility to define a range of agent-states that fall within a pair of states.

Definition 6.2.5: Place Time Lattice

A place-time lattice $\mathcal{PTL} \subset \mathcal{A}$ is a lattice structure $\langle \mathcal{PTL}, \wedge, \vee \rangle$ with a lower bound of $a^{bottom} \in \mathcal{PTL}$, an upper bound of $a^{top} \in \mathcal{PTL}$, for which $time(a^{bottom}) \leq time(a^{top})$ is true. It holds that $\forall p_1, p_2 \in PTL$:

$$p_1 \wedge p_2 = a^{top} \text{ and } p_1 \vee p_2 = a^{bottom}$$

Note that *half* $\mathcal{PTL}$'s (i.e., semi-lattices) can be produced by giving an upper or lower bound and move down- or upwards to include all states reachable under a given number of timesteps.

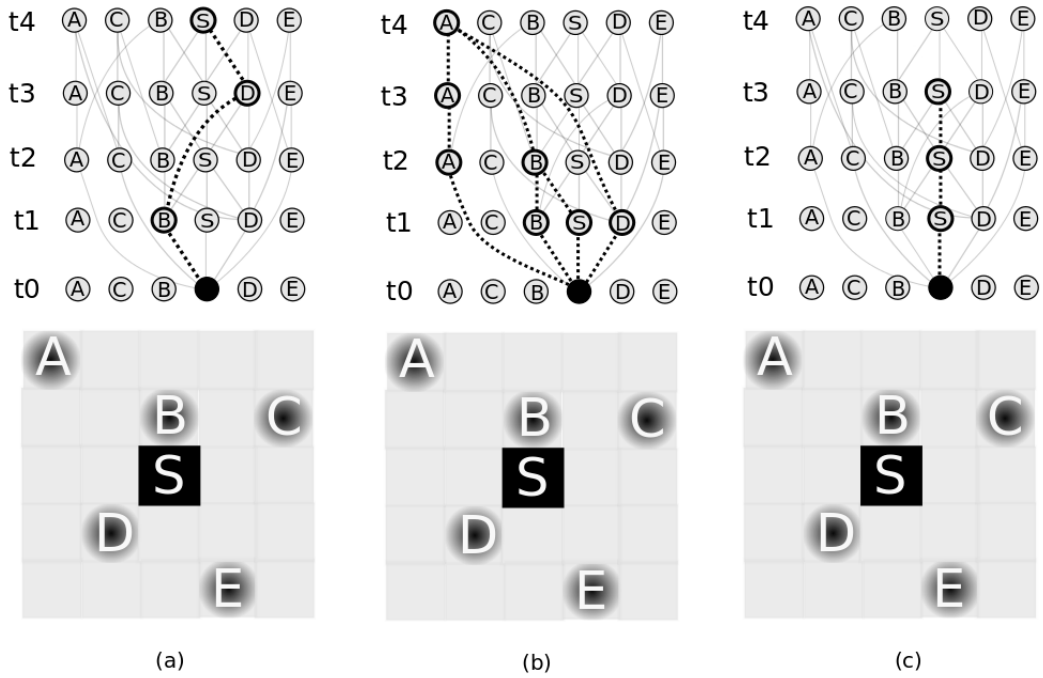

Figure 6.3: (a) A place-time path is a sequence of agent-states. (b) A place-time lattice between the agent-states (S,t0) and (A,t4). (c) A place-time station is a sequence of contiguous agent-states sharing the same place.

A special case of a place-time lattice is a place-time station (see Figure 6.3 (c)), in which the pair of states bounding the activity do not differ in their place, i.e., the agent does stay in the same place over time.

Definition 6.2.6: Place Time Station

A Place-time station (PTS) forms a totally ordered set under the $\preceq$ relation, in which states share the same place, such that $PTS \subset \mathcal{A}$ and:

$$\forall a \in PTS : (a^{bottom} \preceq a) \ \wedge \ (a \preceq a^{top}) \ \wedge \ (place(a) = a^{bottom} = a^{top})$$

Besides reflexivity, antisymmety and transitivity it is total:

$$\forall s, e \in PTS \text{ either } s \preceq e \text{ or } e \preceq s$$

The three concepts place-time path, place-time station and place time lattice, are very similar to the concepts of *space*-time-stations, *space*-time paths and *space*-time prisms.

It will change now as an additional factor that plays into an agent-state description is introduced.

6.3 Adding Requirements

The ultimate goal is to develop a framework capable of representing activities from an agent perspective by considering more factors than space and time. Think of a mobile task planning application that can sense the user's current place, time and table-top sized objects carried. An activity from the perspective of such a system is not only a change in place or time, but can potentially be the dropping or acquisition of an object by an agent. Therefore, a set $\mathcal{O}$ is introduced that represents the complete set of (table-top) objects available to an agent in a given world. The power-set $\mathcal{P}(\mathcal{O})$ therefore produces all subsets that can occur in an agent representation, i.e., all the combinations of objects, an agent can possibly carry. Thus we extend the agent-state-space $\mathcal{A}$ defined in def. 6.2.2 to:

Definition 6.3.1: Extended Agent State

The agent state representation now becomes an element of the set:

$$\mathcal{A}^{ext} = (\mathcal{P} \times \mathcal{T} \times \mathcal{P}(\mathcal{O})).$$

So the extended agent state is now: $a^{ext} \in \mathcal{A}^{ext}$.

There are physical restrictions on how many objects an agent can carry, so the complete power-set $\mathcal{P}(\mathcal{O})$ does include object combinations that are practically not feasible. To return the object set $o \in \mathcal{P}(\mathcal{O})$ of a given *extended* agent-state $a \in \mathcal{A}^{ext}$ (6.3.1) we write $obj(a)$. Note that the "isPossibleBefore"-relation ($\preceq$) is not defined upon the set of *extended* agent-states, since the notion of *possible* expressed by the $\preceq$ relation has changed. While before it was a question of "Can I reach this point/place in a given amount of time-steps?"; now the object dimension added complexity to that question. Answering the question "Can I be there with *something*?" might involve dependencies between objects, such as the need for money before I can buy something (see Abdalla and Frank [2012] for a discussion of this problem).

The additional factor introduces new actions (i.e.,a "pick-up" or "drop" activity). Thus, it has to be assumed that a change in the agent-state does result in a change of the

external environment $\mathcal{E}$. It leads to a break of the traditional time geographic assumption in which the environment does not change by the action of an agent. Subsequently, defining the relation $\preceq$ between the agent states, as depicted previously (see def. 6.2.3), is not possible any more without keeping track of the changes in the environment $\mathcal{E}$. Henceforth, it is necessary to take the environment-state along the state transitions, effectively tracking the changes that happened in the environment. Only this way, feasible future states can be determined.

So another extension of the agent representation has to be made:

Definition 6.3.2: Complete Agent State

The agent state a^{cmp} is an element of the set $\mathcal{A}^{cmp}$, which now becomes:

$$\mathcal{A}^{cmp} = (\mathcal{P} \times \mathcal{T} \times \mathcal{P}(\mathcal{O}) \times \mathcal{E})$$

$\mathcal{E}$ is a set of all the possible states of the environment. Together with the "isPossibleBefore" relation ($\preceq$) it forms the poset structure $\langle \mathcal{A}^{cmp}, \preceq \rangle$

An agent-state $a^{cmp} \in \mathcal{A}^{cmp}$ is represented by a place, a temporal granule, a set of objects and an environment-state $e \in \mathcal{E}$. When planning towards goal, the environment state e can be interpreted as the agent's *mental model of the environment*, i.e., an internal image of the environment at state $a \in^{ext}$. If the agent simulates actions from a state to another, by picking up an object for example, the mental model of the world should be updated accordingly.

By replacing $\mathcal{A}$ by $\mathcal{A}^{cmp}$ the definitions in 6.2.5 (place-time lattice) and 6.2.4(place-time path) and (6.2.6) (place-time station) are still valid, since the $\preceq$ relation retains the structure.

In summary, a poset structure $\langle \mathcal{A}^{cmp}, \preceq \rangle$ that contains all agent-states and corresponding world states possible was defined. It was achieved by using the $\preceq$ relation that captures the logic of the transitions from one agent-state to another. It ensures that requirements and constraints that exceed spatio-temporal factors are incorporated into the model (e.g., Can I get from home to work in three time-steps and have my laptop with me?). The functions *time*, *place*, *obj* and *world* return time, place, objects and the corresponding environment state of $a^{cmp} \in \mathcal{A}^{cmp}$.

6.4 Intended Activity Representation in a Multi-valued State Space

Intentions can translate into projections of future desired states. In section 3.1.1 a distinction between *implementation* and *goal* intentions was made. The two concepts subsume the entities usually stored in scheduling or task planning applications. Implementation intentions correspond to planned activities, i.e., activities that are clearly delineated by geographic and temporal boundaries. Goal intentions are far more general. Gollwitzer [1993] pointed out that the distinction between the two is not crisp, so categorizing intentions into these two types is hard.

In the following a formal definition is given that narrows down the meaning of the terms in the context of this work. Having such a formal description is a first step towards the integration of varying planned/intended activities (such as trips, lectures or errands), one of the points mentioned in section 3.3 that are not implemented in common PIM tools.

6.4.1 Implementation Intention

This work confines the term implementation intention to the following definition:

Definition 6.4.1: Implementation Intention

An implementation intention is any pair (a_{start}, a_{end}) with $a_{start}, a_{end} \in \mathcal{A}^{ext}$ or $a_{start}, a_{end} \in \mathcal{A}$ for which $(a_{start}) \preceq time(a_{end})$ holds true.

Thus, an implementation intention is described by a start-agent-state and an end-agent-state. Thus, is describes the state at the beginning of an implementation intention and the state at the end of it. Note that the states are element of the $\mathcal{A}^{ext}$, meaning that the world state before and after the intention is not included in the definition. Subsequently, because there is no "isPossibleBefore" relation defined for the $\mathcal{A}^{ext}$ set the intermediate states of the pair cannot be inferred.

The goal, therefore, is to translate an implementation intention into a corresponding set of states that are elements of the complete state set $\mathcal{A}^{cmp}$ (i.e., including the environment state). These states form the *space* the agent has to *move* through in order to realize the implementation intention.

To do so, a semantic ambiguity inherent in the implementation intention representation has to be resolved. Consider a start-state description for which an agent has to be at some place at some point in time carrying a specific object. Obviously, carrying another object does not render the achievement of the implementation intention unfeasible.

To solve this problems an equivalence relation '$\sim$' is introduced, defining equivalence between states in $\mathcal{A}^{ext}$ and $\mathcal{A}^{complete}$.

Definition 6.4.2: Equivalence for a^{ext} and a^{cmp}

For $a^{ext} \in \mathcal{A}^{ext}$ and $a^{cmp} \in \mathcal{A}^{cmp}$, $a^{ext} \sim a^{cmp}$ if and only if:

$$(\text{place(a }^{ext}) = place(a^{cmp})) \wedge (time(a^{ext}) = time(a^{cmp})) \wedge (obj(a^{ext}) \supseteq obj(a^{cmp}))$$

In words, a^{ext} *'is equivalent to'* a^{cmp} if they share the same place, time and if the object-set in a^{ext} is a super-set of the object-set defined in a^{cmp}. Note that the relation is antisymmetric $(a^{ext} \sim a^{cmp}) \neq (a^{cmp} \sim a^{ext})$. The case where certain objects are prohibited from being brought (e.g., groceries into a theater), is excluded.

Using this relation and a given agent start- and end-state all the states from the complete state space of $\mathcal{A}^{cmp}$ can be inferred. It builds the basis to define an implementation intention state space, that is, all the states an agent can take while conducting an implementation intention.

Definition 6.4.3: Implementation Intention State Space

The set $\mathcal{I}_{impl} \in \mathcal{A}^{cmp}$, describing an implementation intention (s_{start}, s_{end}), is made out of the valid start- and end-states:

$$SS = \{\ a \in \mathcal{A}^{cmp} \mid a \sim s_{start}\ \}$$

$$ES = \{\ a \in AS^{complete} \mid a \sim s_{end}\}$$

and the intermediate states, given by:

$$\forall i \in \mathcal{I} : \exists ss \in SS \wedge es \in ES \text{ such that } ss \preceq i \preceq es$$

Since $\mathcal{I} \in \mathcal{A}^{cmp}$ it forms a poset under the relation $(\preceq)$.

Thus, an implementation intention can be described by a pair of an agent start- and end-state that constrain the possible world states to a set in which an agent has to operate in order to successfully realize the implementation intention.

6.4.2 Goal Intention

The great challenge in formalizing goal intentions is their generality. As in table 1 illustrated, examples include things like: "I want to be successful!". They can be seen as future desired states, possibly independent of the actual agent state, i.e., a desired world state. The incorporation of a general notion of goal intention is very hard. Therefore only a specific subset of goal intentions will be modeled.

For the purpose of this work, only those goal intentions are considered, that can be expressed in terms of place, time and objects, and are achievable by any of the three activities: moving, picking and dropping. This way, a goal intention can be understood as a conjunction of constraints imposed on the agent-state-set $\mathcal{A}^{ext}$. It means that here, goal intentions are expressed as all the agent-states that are only one transition away from the production of a desired world state. This is possible because only a single agent is present in this model. Further, these states can again be mapped into the complete set $\mathcal{A}^{cmp}$, that represent the operational space of the intention. In other words, goal intentions can be represented as a collection of agent-states that can lead to a desired world state.

Thus, the goal intention "I want the book to be back at the library", for example, can be translated into all the states where the agent carries the book and is at the library (within the opening hours). It can be formally expressed by:

Definition 6.4.4: Goal Intention State Space

A set of states in $\mathcal{I}_{goal} \in \mathcal{A}^{cmp}$ that represent the required states to achieve a a goal intention is produced as follows:

$\forall \mathcal{I}_{goal} = a \in \mathcal{A}^{cmp} : time(a) \text{"before"} closinghour \wedge book \subseteq object(a)$

What is achieved are two sets $\mathcal{I}_{impl}$ and $\mathcal{I}_{goal}$ that express implementation- and goal-intentions, that are part of the same set $\mathcal{A}^{cmp}$. These two types of intentions can be combined and dealt with in a single framework.

6.5 Summary

This chapter developed a model to represent activities on the basis of agent representations by place and time. It moved from the continuous view of space in time geography into discrete space with a finite set of places. The main innovation is that place and time are of a certain granularity, allowing for a multi-granular representation of spatio-temporal activities. It was observed that by adding requirements of small scale objects, to agent state descriptions, does make the description of activities by the means of agent-states more complex. The initial assumption of a static environment unaffected by agent movement, does not hold for small scale objects, since the operations an agent can perform (i.e., picking-up or dropping) to case a change in state, do affect the state of the environment. To compute the states possible from a specific state the changes in the environment have to be tracked. The question of whether one activity is possible after another, becomes a relatively complex planning task. Lastly, it was shown how formal definitions of implementation intention and goal intention can lead to the handling of fixed and flexible tasks in a single framework.

Chapter 7

Activity Composition and Granular Transformation

The previous chapter dealt with the idea of representing two kinds of *intentions* in the form of structured sets of agent-states. It highlighted the changes that occur when introducing actions that change the overall world state. The definitions in the previous chapters assumed that states are always of the same granularity. This chapter will discuss the implications of adding several intentions together as well as what happens if the level of granularity is changed. Section 5.4 stated that there is no notion of granularity defined for table-top objects. Due to the dynamic nature of table-top sized objects it does not make sense to include granular representations, since they cannot always be assumed to be true for future states. In this chapter the discussion will omit object-requirements and focus on spatio-temporal principles. Hence, the discussion is based on the most primitive states of the set $\mathcal{A}$ (see Def. 6.2.1), that are made of a place-granule and a temporal-granule. After the operations to coarsen and combine those basic states are established, it will be shown how to integrate object requirements by using a *current context*.

7.1 Variant and Invariant Conditions when Coarsening States

Section 6.2.1 introduced basic constructs that can describe past and future activities, by bounding them with bottom and top agent-states (i.e. place-time lattice, -path and -station). Each agent-state is made out of a place- and temporal granule. These granules

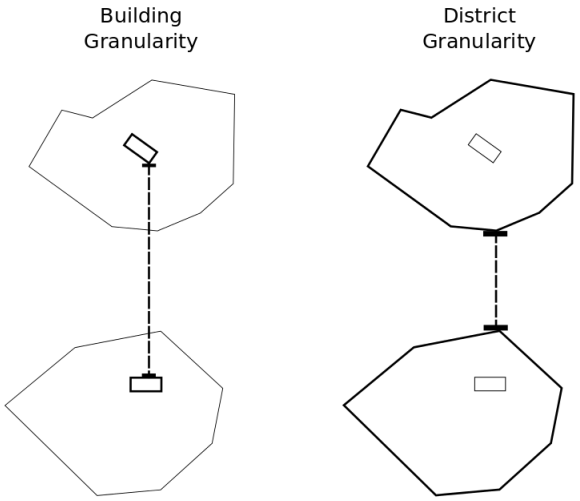

Figure 7.1: To ensure an underestimation the minimum distance of the district level should be used.

can be coarsened and still maintaining the structure given by the finer granularity. Coarsening in this case can happen on either place, time or both of the dimensions. In general, moving from a finer spatial granularity to a coarser one, while keeping the temporal granularity, the amount of *places* are either reduced or stay the same. Moving to a coarser granularity in time, while retaining the place-granularity, the number of places reachable per time-step will be increased (under the assumption of fixed speed) (see Figure 7.2 for a graphical illustration). One difficulty lies in determining which places can be reached and which not. Earlier (Def. 6.2.3) a $\preceq$ relation was introduced that holds if a state is reachable after another. The truth value of the relation is determined by the travel time necessary for moving from one state to another. Unfortunately, a coarser spatial granularity introduces an uncertainty in estimating when a place is reached, since its spatial extend is larger.

Consider the transformation from building to district level. Moving to every part of a building, in general, is a matter of minutes, while moving to every part in a district might take up to an hour or more. To assert that state s_2 is reachable from state s_1 within an hour time, assumes, not only a certain velocity, but also a certain distance between the places. Figure 7.1 illustrates the problem. To ensure that the truth value of the $\preceq$ relation between two states remains invariant after they were coarsened, the minimum time has to be taken. It ensures that no place at a finer granularity, contained

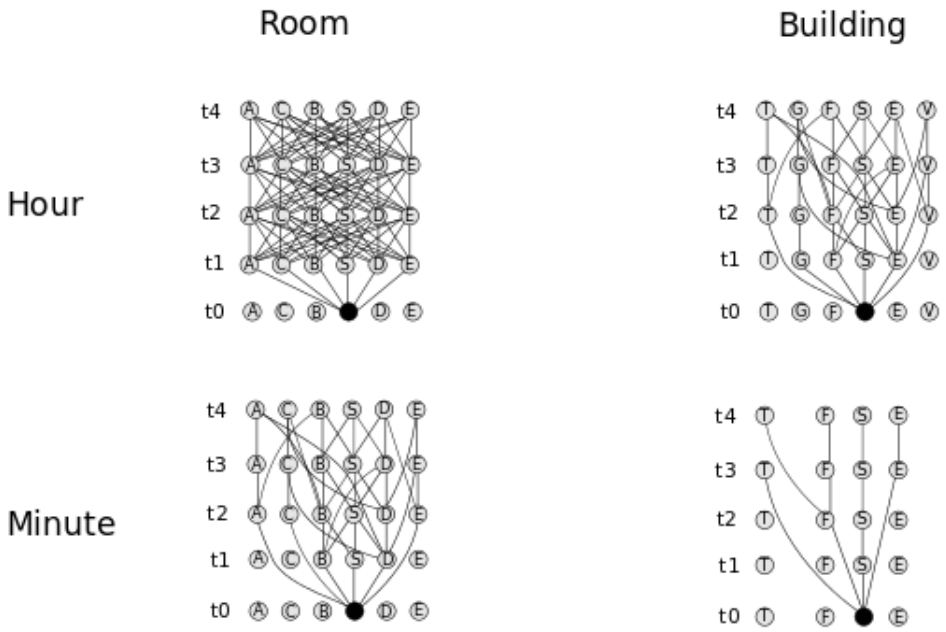

Figure 7.2: Depending on the granularity used for the temporal or platial description of an agent state, the possible transitions are increased or decreased. For example, at a given position S, at room-minute granularity there are less or equal transitions possible in one time step than at room-hour granularity.

in the district, will be excluded.

Definition 7.1.1: Coarsening Operation

Formally, a *coarsening* function $\lambda :: \mathcal{A} \to \mathcal{A}$ mapping from a state of a certain spatial and temporal granularity to a state a coarser granularity it holds that for $a_1, a_2 \in \mathcal{A}$:

$$\text{If } a_1 \preceq a_2 \text{ then } \lambda(a_1) \preceq \lambda(a_2)$$

In words, if a state a_1 is possible before a_2 at a certain granularity, coarsening the resolution of them states maintains the truth value of the $\preceq$ relation.

7.2 Grouping Activities

Next intended activities are combined into into an complex of activities, such as a trip to a conference for example. Wang and Cheng [2001] proposed a model, which distinguished between *stay* and *travel* periods of human activity, that can then be sequenced into an *activity pattern*. The approach taken here is similar. Only an activity pattern is called a *block*. A block $\mathcal{B}$ contains a set of implementation activities (see def. 6.4.1) of the form *place-time station* or *place-time lattice* (Figure 6.3), or goal intentions (6.4.4):

To compose blocks together, a binary composing operation $\circ$ that takes two *blocks* and returns yet again a block, is needed. The operation is defined as $\mathcal{B} \times \mathcal{B} \to \mathcal{B}$. Further, an identity element $e \in \mathcal{B}$ such that for every $b \in \mathcal{B}$, $e \circ b = b \circ e$ holds. Formally, it is an empty block , but can be interpreted as *no intended activity*, combining **no** activity with an activity results in the activity.

$\mathcal{B}$, in principle is the powerset of all intended activity descriptions $\mathcal{I} = \mathcal{I}_{impl} \cup \mathcal{I}_{goal}$ (i.e., implementation and goal intentions) with an additional element e, formally:

$$\mathcal{B} = \mathcal{P}(\mathcal{I}) \cup \{e\}$$

A temporal perspective on the operation $\circ$ assumes that it is associative and commutative. This is because the temporal (total) ordering inherent in the activities force a single constellation of the complex. For instance, given the activities a, b, c, with a being

temporally located before b and b before c. It can be assumed that $(a \circ b) \circ c = a \circ (b \circ c)$ and $a \circ b = b \circ a$, since all of those computations will produce an activity complex that puts the activities in a sequence according to the temporal ordering relation. Introducing a spatial dimension, though, renders the total ordering induced by time invalid. It means that a partial ordering relation $\preceq$ is defined over the contents of a block $\langle \mathcal{B}, \preceq \rangle$. The ordering relation $\preceq$ again stands for a *possibility*, i.e., activity block $a \preceq b$ holds if a is (hypothetically) possible before b.

The aim is to be able to compose blocks that are based on state descriptions that include more than spatio-temporal factors. Therefore a general way to compose blocks independent of the underlying description is described.

7.2.1 Ordering Blocks

To define an ordering between blocks (i.e, groupings of activities), an operation *entry* and *exit* will be defined. Both operations map from the domain of blocks $\mathcal{B}$ into the state co-domain $\mathcal{A}$. They return the states that define the entry- and exit-spaces of the block.

Definition 7.2.1: Ordering Blocks

Block A, with a set of exit-states $S_{exit} = exit(A)$ is said to be *possible before* ($\preceq$) a block B with a set of entry-states $S_{entry} = entry(b)$, if there exists a state $s_{end} \in S_{exit}$ and $s_{start} \in S_{entry}$ such that $s_{end} \preceq s_{start}$.

There has to be at least one state of the potential exit states from which it is possible to reach one of the entry states of the next block, to be able to say that a block is (hypothetically) possible before another.

7.2.2 Blocks of Alternatives

As opposed to Wang and Cheng's [2001] model, in which past activities were the main focus, the goal here is to design a model capable to reason over *intended activities* as stored in common scheduling applications. It implies that some of those activities are of uncertain nature, but are kept to allow for a decision later on. To accommodate for this, Wang and Cheng's [2001] model of *stay* and *move* activity patterns has to be extended to not only include *sequences* of patterns but also *alternatives*. Representing a conference program, for example, requires the storage of parallel sessions. The

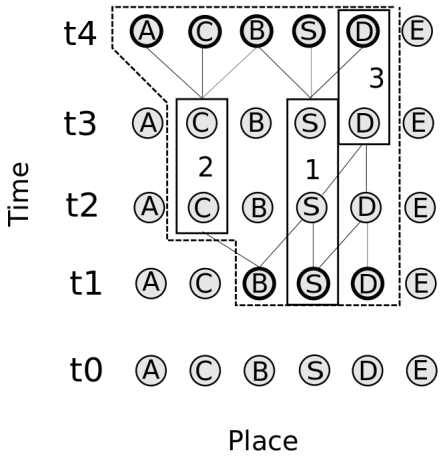

Figure 7.3: An illustration of how entry- and exit-spaces of `Alt`-blocks are defined. At t1 all the places that allow to reach every start-state of the the containing activities are selected. For the exit point all places reachable from the end states of the containing activities are taken. Note that place C at t1 is not part of the entry point since the start-state of block C is not reachable from it.

structure $\langle B, \preceq \rangle$ serves as the basis for distinguishing blocks that contain sequences or alternatives. Given three activities 1,2,3 grouped into a block of alternatives. 1 starts before 2 and 2 before 3. 3 ends after 1 and 2. The entry-states for the containing block is determined by the set of places that allow to reach the starting points of 2 and 3 by the starting time of 1. The exit point consists of all the places reachable from activity 2 and 3 by the end of activity 3 (See Figure 7.3 for a graphical illustration of the example). Following this methodology ensures that placing a block before a block of alternatives is only possible if every alternative of the group is reachable from the prior block (see Figure 7.4).

The underlying principle of spatio-temporal constraints, as mentioned before, is very similar to the conceptual and formal models in time geography. There is, though, an additional fact that needs to be considered. Namely, the varying granularities of state descriptions allowed for in the model.

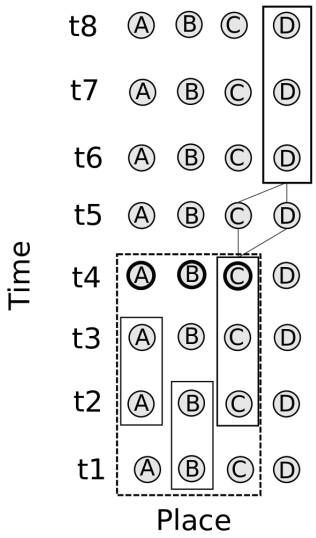

Figure 7.4: An illustration of an `Alt`-block A that extends from t1-t4 and a `Seq`-block B from t6-t8, that is inferred to be possible before B, due to the reachability of state (D,6) from (C,4).

7.2.3 Combining Blocks of Varying Granularity

From a spatial perspective, granularity can be dealt with quite easily. If a block does end at a granularity that contains the granularity of the succeeding block (e.g., first ends in Vienna, second starts in Vienna Central Station), then this block is hypothetically possible. The same holds for the opposite way, i.e., a block that ends at a granularity that is contained by the start of the succeeding. For the temporal granularity so far we assumed activities to be described at the same level. Also we considered blocks that consist of a single activity to be opaque, thus if they temporally overlap they are mutually exclusive. The challenge here is that if there exist two blocks one of which is described on a coarse granularity it might happen that one contains the other. Introducing the block of finer granularity does refine the information available about the coarser one. At the same time, it is necessary to keep the temporal information of the coarser block.

It is necessary to introduce placeholders that ensure the temporal boundaries of the compiled block are the same as the ones of the coarser block. Such place holders are in essence *place-time lattices*, cut in half (see Figure 7.5). The counterpart to space-time cones in a time geography. Finally, a sequence block can be produced that exhibits an entry- and exit-space in consideration of the block within it.

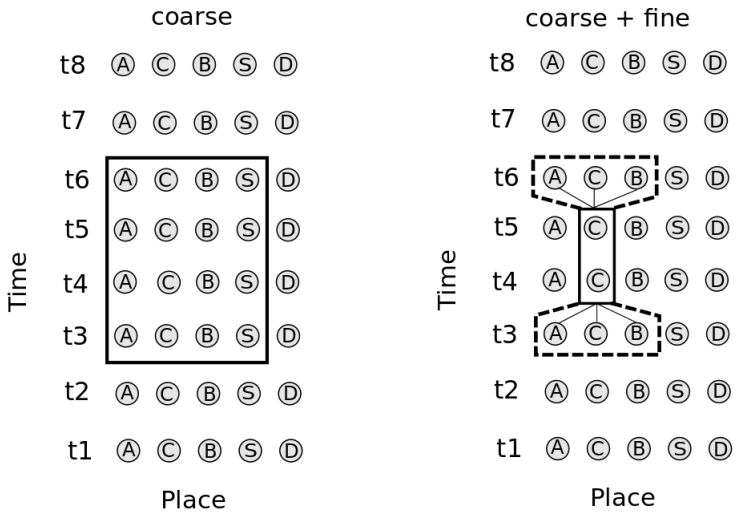

Figure 7.5: On the left: A block that delineates the states of the coarse representation block. On the right: A block of finer granularity is introduced. Placeholder underneath and above this block are needed to maintain the temporal bounds of the coarser block, while simultaneously reducing the state-set due to the new information available.

The approach allows to iteratively refine information about a planned activity. For example, consider an activity (e.g., a picnick with friends) which is intended to be conducted on a Saturday, even though the exact time and place is yet to be decided. The intended activity, therefore, is added to the schedule on a day and city granularity (i.e., Saturday,Vienna). Later on, another activity on a finer granularity, e.g., hour and building. It can therefore narrow down the possibilities to undertake the coarser activity (See Figure 7.4).

7.2.4 Block Composition under the Consideration of Table-top Object Requirements

Looking at a state-set containing object requirements as suggested in the previous sections, the computation of whether an activity is achievable after another becomes more complex. The major problem is that, without additional context, it is impossible to decide whether an activity that requires an object can be achieved after another. As mentioned in Section 5.4 a crucial difference between human conception of geographic objects, like places, and table-top objects is *persistence*. While, places are in general assumed to be stable in space, table-top objects are dynamic and can change their location frequently.

At a general level, it makes sense to say that an activity is impossible due to the underlying spatio-temporal constraints, that are assumed to be stable. For planning activities, first the spatio-temporal structure needs to be dealt with, the object requirements are an additional layer that can be added and looked at in the context of a *current* situation.

In other words, because the composition of blocks is dealing with hypothetical possibilities based on spatio-temporal constraints that are assumed to be relatively stable (places are assumed to not change), it is impossible to make a statement about the possibility of one activity to be conducted before another that requires a certain object, since there is no knowledge about the geographical distribution of the objects in future. Therefore the future world-state is impossible to predict. It is, though, possible to assert things about the feasibility of activities in terms of the present world-state.

7.3 Summary

The set of blocks $\mathcal{B}$ in conjunction with the binary operation $\circ$ do form an algebraic structure, namely a commutative monoid. It gives the ability to combine blocks together and form an *activity complex*. It represents a set of partially ordered activities. The chapter explained the ordering of blocks can be defined, such that a spatio-temporally consistent structure is maintained.

Chapter 8

Implementation of a Computational Model

In the previous sections a rather general framework for activity representations was introduced. It helped to understand the underlying principles and to narrow down the meaning of some of the terminology. This chapter is dedicated to present a computational and applicable model of a system using place and time as the fundamental structure to describe activities.

8.1 Methodology

For modeling the scenario classes of abstract algebras, implemented as Haskell [Jones, 2003] type classes, are used. The instances of such classes define models implemented for particular datatypes. As illustrated by others before [Kuhn, 2009; Raubal, 2001; Raubal and Kuhn, 2004] it allows for ontological models that can be tested for consistency, as well as implemented and executed. The interested reader is referred to Frank and Kuhn's [1999] discussion of the benefits of functional languages as specification languages. The main point is that type classes group together operations to describe and observe behavior of concepts.

Classes can be understood as theories of concepts. The operations defined by the classes can change the state of a concept individual or answer questions about it. Type-classes in Haskell are parameterized polymorphic, meaning that the they can be instantiated and parameterized by various datatypes [Jones, 2003]. Hence, a change or replacement of the datatypes does not render the model invalid, as long as appropriate instantiations (i.e., the necessary operations) of the type-classes for the alternative types are given.

8.2 Implementation

8.2.1 Granularity and Containment

Chapter 5 supported the view that system of granularities are primarily social constructs and there are similarities between spatial and temporal system of granularities. Therefore, it is reasonable to provide a general notion of granularity in form of a typeclass:

```
1 class Granularity granule where
2
3         coarsen :: granule -> granule
4         commonUpperGranule ::
5         (Eq granule) => granule -> granule -> granule
6         isFinerThan :: granule -> granule -> Bool
7         equals :: (Eq granule) => granule -> granule -> Bool
8         contains :: granule -> granule -> Bool
```

The operation `coarsen` takes a granule of a certain granularity and moves to the parent granule (e.g., Vienna → Austria). The `commonUpperGranule` function takes two granules and returns the granule containing both of them (e.g., Vienna, Salzburg → Austria).

The `contains` relation checks whether one granule is contained by another. To avoid confusion, the reader should reconsider the difference between *granularity* and *granule*. While a *granularity* stands for the whole level, e.g., all countries. A *granule* is one instance of the set, i.e., Austria. Therefore, just because Paris is part of a lower granularity than Austria, does not mean it is contained by it. Henceforth, a separate operation is needed (i.e., `contains` vs. `isFinerThan`).

8.2.2 Time

For the temporal representation the `xsd:dateTime` definition is modeled and a `TempGranularity`-type is added to it.

```
1 data Date = Date { y :: Int,
2                    m :: Int,
3                    d :: Int,
4                    hh :: Int,
5                    mm :: Int ,
6                    granularity :: TempGranules } deriving (Eq,Show,Read)
7
8
```

```
9  data TempGranules = YearG   |
10                    MonthG  |
11                    DayG    |
12                    HourG   |
13                    MinuteG deriving (Eq,Enum,Ord,Show,Read)
```

The *Date*-datatype is depicted in record-syntax, meaning that the names on the left side of the '::' stand for operations that, when applied to an instance of type `Date` return the value of the property. The `TempGranularity`-type has an ordering defined upon it. This way the granularity of the information can be derived. `Date` implements the *granularity* class (see Appendix B, Module 9).

```
1  instance Granularity Date where
```

It allows to coarsen the granularity, ask whether one date `isFinerThan` another or is contained in another. For example, the date "2-09-2014" of *month* granularity is contained in the date 2014 of granularity *year*.

Note, that in the model proposed by Bettini et al. [2000], granules of a date system can include more concepts than the usual POSIX date-time (Year-Month-Day-Hour-Minute-Second) granularity ordering. Bettini et al. [2000]'s model can potentially include Year-Week-Hour or Year-Weekend-Day models. It therefore defines lattice structures rather than strict hierarchies. Nevertheless, for reasons of simplicity (and its widespread use), a hierarchical model is implemented. The complete code can be found in Appendix B, Module 9.

8.2.3 Places

A meaningful description of activities requires a formal notion of *place*:

```
1  data Place = Place {placeID :: Int,
2                      placeName :: String,
3                      placeLoc :: Geometry,
4                      granularity :: SpatialGranularity,
5                      getObjs :: [Object]
6                      }
```

The first two types need no further explanation; the `Geometry` type stands for a spatial representation not further detailed here. `Object` represents a tabletop object that can be contained in the place. The `spatialGranules` type is implemented here as a simple enumeration with an ordering defined upon it.

```
data SpatialGranularity = MeetingPoint |
                          Room |
                          Building |
                          Park |
                          Neighbourhood |
                          City |
                          Country |
                          World

instance Ord SpatialGranules where
```

The ordering relation defined over the `SpatialGranules` type, is a partial ordering. It states that `City` is of finer granularity than `Country` and that `MeetingPoint` and `Building` cannot be compared.

The goal for the `Place`-type is to instantiate the *granularity class*. It would allow to coarsen a place (i.e., moving to the upper granule), check whether a place is contained by another and ask whether one place is of finer granularity than another.

Unlike temporal granules, spatial granules cannot be derived from the inherent structure of the place description. To know what place is contained by another, additional knowledge has to be incorporated. Thus, before `Place` can implement the `Granularity`-typeclass, another typeclass is introduced. The complete code for the `Place`-type can be found in Appendix B, Module 2.

8.2.4 The Geography

Now that a `Place`-type is established, a *knowledge base* that integrates and relates is introduced. As stated in section 3.1.2, a knowledge base is essential for the formation of plans. In this context it can be understood as a database that stores the multi-granular structure of places as well as their relations to tabletop objects. It represents the knowledge about the environment or *geography* as it is termed here.

```
class Geography placeDB where

        addPlace :: Place -> placeDB -> IO ()
        updatePlace :: (Place -> Place) -> Place -> placeDB -> IO ()
        parentPlace :: Place -> placeDB -> IO (Maybe Place)
        containingPlaces :: placeDB -> Place -> IO [Place]
        placePerGranule :: SpatialGranules -> placeDB -> IO [Place]
        closePlaces :: Place -> placeDB -> IO [Place]
        queryObj :: placeDB -> TableObject -> [Place]
        queryObjAt :: placeDB -> nodeID -> TableObject
        queryAllObj :: placeDB -> TableObject
```

A geography, therefore allows to add, remove or update places, query for all the places contained in another `Place` of coarser granularity (e.g, all the cities inside Austria) or answer spatial queries, such as what are the places close to another, or how long does it take, given a transport mode, to move from one place to another, what is the euclidean distance between two places. It also allows to answer questions about *objects*, like the question of where to find an object or an object type. The complete code for the `Geography`-class is found in Appendix B, Module 1. Finally, the `geography`-class allows to implement the `Granularity`-typeclass for the `Place` type:

```
1 instance Granularity Place where
2
3    coarsen p' = parentPlace p
4    isFinerThan p1 p2 = (granularity p1) < granularity p2
```

8.2.5 A Planning Facility

The previous `Geography`-class allows to answer question about the environment. They are vital for the following class which allows to plan.

```
1 class (Geography world) => Planner state world | state -> world where
2
3         achievable :: world -> state -> state -> Bool
4         availableTime :: state -> state -> Double
5         requiredTime :: state -> state -> IO Double
6         accessiblePlaces :: world -> state -> state -> IO [Place]
7         reachableInTime :: world -> state ->  IO [Place]
```

The `Planner`-class functions as a facility that allows to ask questions about the reachability of one state and another.

8.2.6 A Block Model to Represent Intended Activities

Finally, a description of *intentions*[1] can be introduced. In section 3.1.1 we distinguished between goal and implementation intentions. The fact that errands (close to goal intentions) and events (implementation intentions) are separated in in common scheduling applications was stated as a challenge to the development of a next generation personal assistant application.

This work describes intentions by ranges of prior- and post-agent states. Those bounding ranges restrict the states possible in between them. Thus, an implementation

[1] For brevity the term intentions, standing for intended activities, will be used

intention like, a lecture from 10:00 am to 11:00 in lecture-hall 3, does require an agent state of `(10:00, lecture-hall 3)` as a prior state and results in a post state of `(11:00,lecture-hall 3)`. Depending on the temporal granularity used, all intermediate states are in the range of 10:00-11:00 am, at lecture-hall 3 (i.e., a *place-time* station as in figure 6.3 (c)). The mentioned "buy milk for home" example, can be expressed by a range of post states, e.g.: `( < 8:00 ,Home, [DropOff Milk])`. The prior agent states are thus, all the states that allow for a transition to one of those post (goal) states. The main distinction made, is that one type has fixed properties, while the other type (goal intentions) can have some of its properties modified.

In the following the model will be developed by considering *implementation intentions* only. Once established, flexible *goal intention* like entities will be added. We begin with a datatype that represents an activity in place and time.

```
1 data Activity = PTS ID (Date,Date) Place [Action]
2               | PTL ID (Date,Date) (Place,Place) [Action]
3                 deriving (Eq,Show,Read)
```

The above sum-type is able to represent common concepts found in calendars, such as lectures, meetings or trips. In addition, a list of type `Action` is added. It defines one or more object related actions necessary to conduct the activity. The actions are explained in 4.3.3.1 and defined as:

```
1 data Action = Maint Object |
2               Drop Object  |
3               PickUp Object
```

It enables to express how an object proceeds through an activity. `Maint` means that an object is required in the beginning and will still be with the agent in the end. `Drop` means that the object is required in the beginning but will be dropped while at the activity and therefore not be included in the exit-space of it. `PickUP` means that the object is not required before the activity but will be existent in the exit-states of the activity.

In section 3.3 it was stated that activities need to be related together. To represent `intended activities`, it is desirable to keep information about alternatives. For instance, to represent a conference program it is necessary to have parallel sessions stored, in order to allow an inquiry about choices available. For this reason *blocks* are introduced that are able to combine activities into aggregates that differentiate between sequences and alternatives.

```
data Block = Single Activity |
             Seq [Block]     |
             Alt [Block]
```

Three types of blocks are defined, the `Single`-block does wrap an `Activity` type into a block-type; the `Seq`-block groups blocks into a sequence; the `Alt`-block into a set of alternatives. For a graphical illustration refer to Figure 8.2.

While the datatype by itself does not carry a lot of information, the actual behavior of the block is defined by the `Blocks` class defined in the following:

```
class Blocks block where
    startTime :: block -> Date
    endTime :: block -> Date
    startPlace :: block -> [Place]
    endPlace :: block -> [Place]
    getRequirements :: block -> [Action]
    temporalProjection :: block -> (Date,Date)
    placeProjection :: block -> [Place]
    potential :: block -> [Potentials]
    possibleBefore :: block -> block -> Bool
    possibleAfter :: block -> block -> Bool
    possibleWithin :: block -> block -> Bool
```

The main operations are found in the first two operation-pairs. They represent the core difference of the model, that moves from calender events to time geography like concepts of *place-time stations* or *lattices*. Therefore, each block has an entry- and exit-state-set that can be determined by invoking the operations. It is a main necessity for making the blocks composable. Note, that a general level of composition only `Place` and `Date` play a role in the composition-logic.

For example, a lecture starts at a room R at time T and ends at time T' so the entry space is (R,T) and exit space is (R,T'). For the `Single`-block the entry and exit points are simply the entry and exit spaces of the activities underneath them. In case of the `Seq`-type, taking the entry/exit spaces of the first/last block in the sequence is sufficient.

The three operations `possibleBefore`,`possibleAfter` and `possibleWithin` check whether a block can be sequentially aligned, merged, or wrapped into a block of alternatives.

Every block has a projection into *place* and *time*. The `placeProjection` returns the set of places, that are part of the *intended activity*, in other words it is the union over all places involved in every sub-block. The temporal projection returns the containing interval, i.e., the outer temporal boundaries of it (e.g., for the conference 8:00 am to 6

pm) (see Figure 8.1). While the `placeProjection` is implemented similarly for every type of block, the `temporalProjection` does follow a different logic when applied to `Alt`-blocks, as explained in section 7.2 .

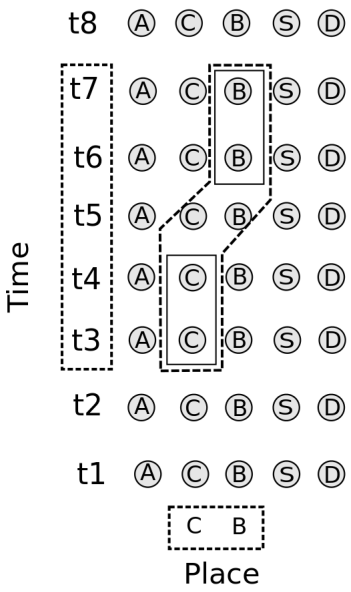

Figure 8.1: An illustration of temporal and platial projections of a sequence block (dotted line) made out of two place-time station activities.

Apart from the place-time projections, the *potential*-operation returns a set of place-time lattices, or place-time prisms. They represent *freetime*, i.e., place-time spaces that can potentially be visited and will be of particular interest when intending to check whether errands can be run within a block.

```
1 data Potentials = Full Interval ([Place],[Place]) |
2                   Half Interval [Place]
```

Figure 8.3 illustrates the representation of (part of) a conference schedule using the suggested block structure.

8.2.6.1 Aggregation of Blocks

In section 7.2 we claimed that the composition of activities follow the rules of a monoid structure (in fact even a commutative monoid). Since the Haskell programming lan-

guage provides a `Monoid` type-class that we can instantiate, there is no need to introduce an *aggregate* operation in the `Blocks` type-class.

The `Monoid` typeclass is described as follows:

```
1 class Monoid a where
2     mempty :: a
3     mappend :: a -> a -> a
4     mconcat :: [a] -> a
5     mconcat = foldr mappend mempty
```

The operation `mconcat` is universally defined as a structural recursion function (`foldr`) that uses the combination operation and the identity operator to aggregate over a list of a implemented type. Thus, implementing a type over a Monoid does imply the possibility of aggregation.

Using the operations `possibleBefore`, `possibleAfter` and `possibleWithin`), the `mappend` operation can be implemented. Subsequently, the `mconcat` operation can be used to build a complex block structure from a list of blocks. The `mappend` operation *merges* blocks, i.e., it does change the internal structure of blocks if necessary.

The block structure does not consider conceptual relations that exist between activities. It can mix activities that are conceptually not part of each other, e.g., a lunch with a friend in the break of the conference is not part of the conference activity itself, but will be represented in the same block. The blocks do merely represent spatio-temporal structure. To distinguish between the block's purposes a separate indexing needs to be implemented, as will be explained in 8.2.6.3.

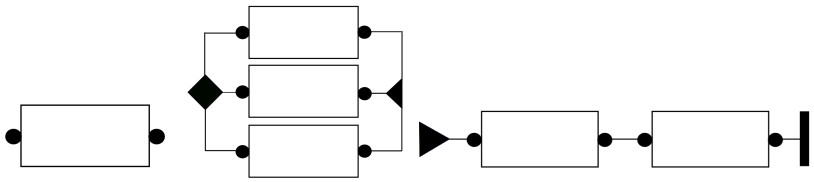

Figure 8.2: Graphical representation of block types. From left to right: The `Free` (upper one) and *Single* block. The `Alt`-block and the `Seq`-block that recursively groups several blocks

The current state of the computational model, does allow to represent the fixed events of Scenario A (4.1.1) in a structured way (see Figure 8.4). The `Alt`-blocks allow to model parallel sessions as alternatives. All blocks form one `Seq`-block that is populated with `Single`- and `Alt`-blocks.

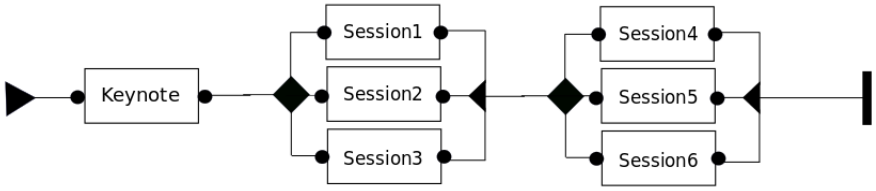

Figure 8.3: A composition of blocks that represent part of a conference program.

8.2.6.2 Goal Intentions as Flexible Blocks

This model treats goal intention as errand like tasks, namely they differ by being loosely defined in terms of their spatio-temporal properties. For example, the errand of returning a book to the library does not say when to do it, but is usually bound to a deadline. As already mentioned, such an errand can be translated into a implementation intention by assuming a duration of the activity. Subtracting the duration from the deadline gives that start-time of the last possible instance of the activity.

Such an activity only represents a single instance of many possible manifestations. The block has to be distinguishable from the rest. Therefore the model is extended by introducing a flexible block.

```
1 data Block = Single Activity |
2              Flex Activity   |
3              Seq [Block]     |
4              Alt [Block]     |
```

A flexible block has the distinguishing property, of being shift-able in time. So its behavior when merging it with other blocks differs to the rest of the blocks. While putting a `Flex`-block *in front of* or *behind* another follows the same logic as with other blocks; it will not be wrapped into an `Alt`-block when it is not serialize-able. In such a case it will be shifted forward in time (See Figure **??**) until a position is found where it satisfies the spatio-temporal constraints. The final position of the block represents the last possible instance of the activity.

Figure 8.5 shows how the flexible block (framed in a dotted line) is placed into the structure of rigid blocks. To review the complete code, refer to Appendix B, Module 11.

8.2.6.3 Conceptual Block Distinction

A final addition to the block model will allow to distinguish between conceptual relationships of activities. While activities are spatio-temporally intermingled, their *relations*

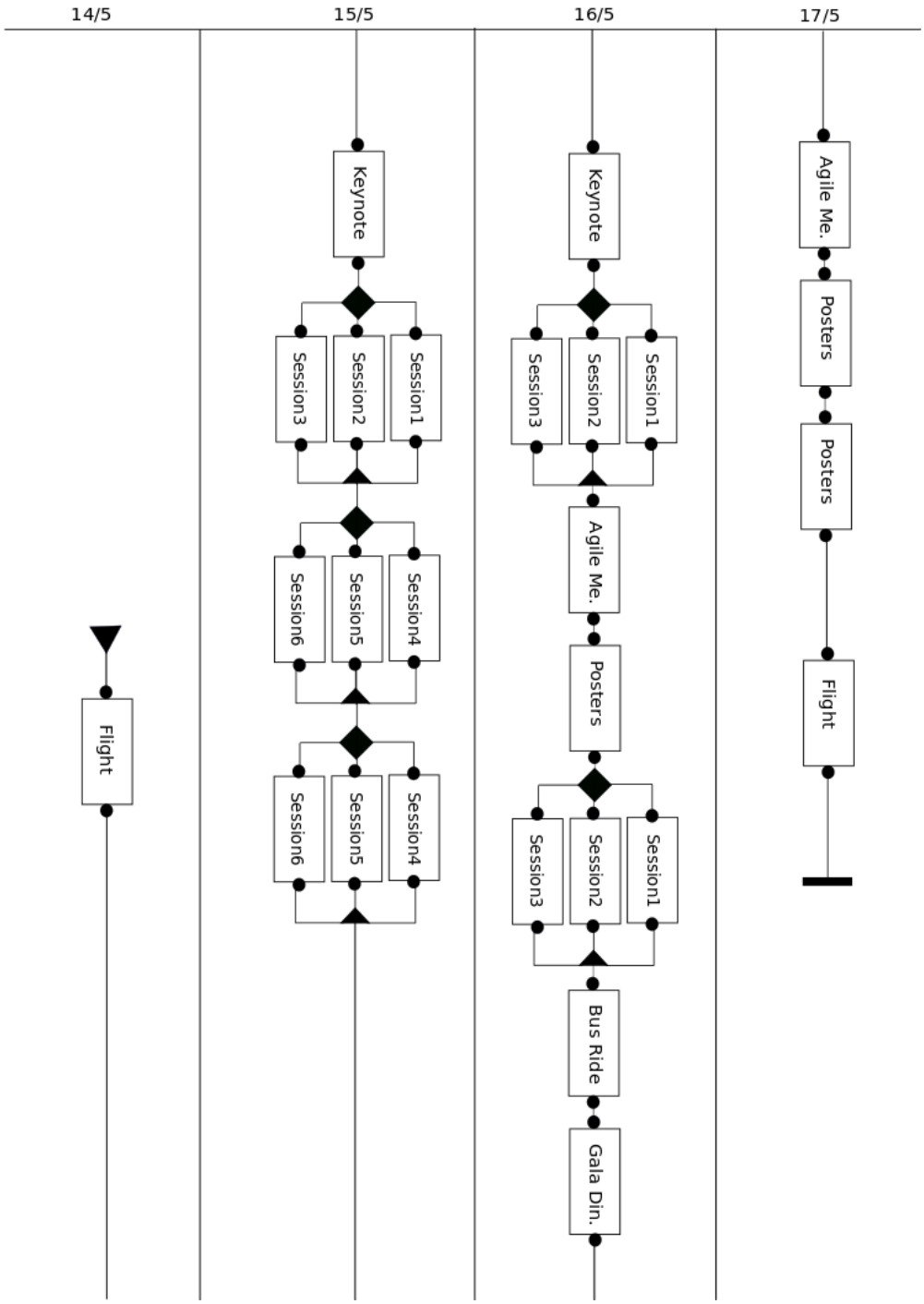

Figure 8.4: The use case scenario depicted in figure 4.2 represented in the block model.

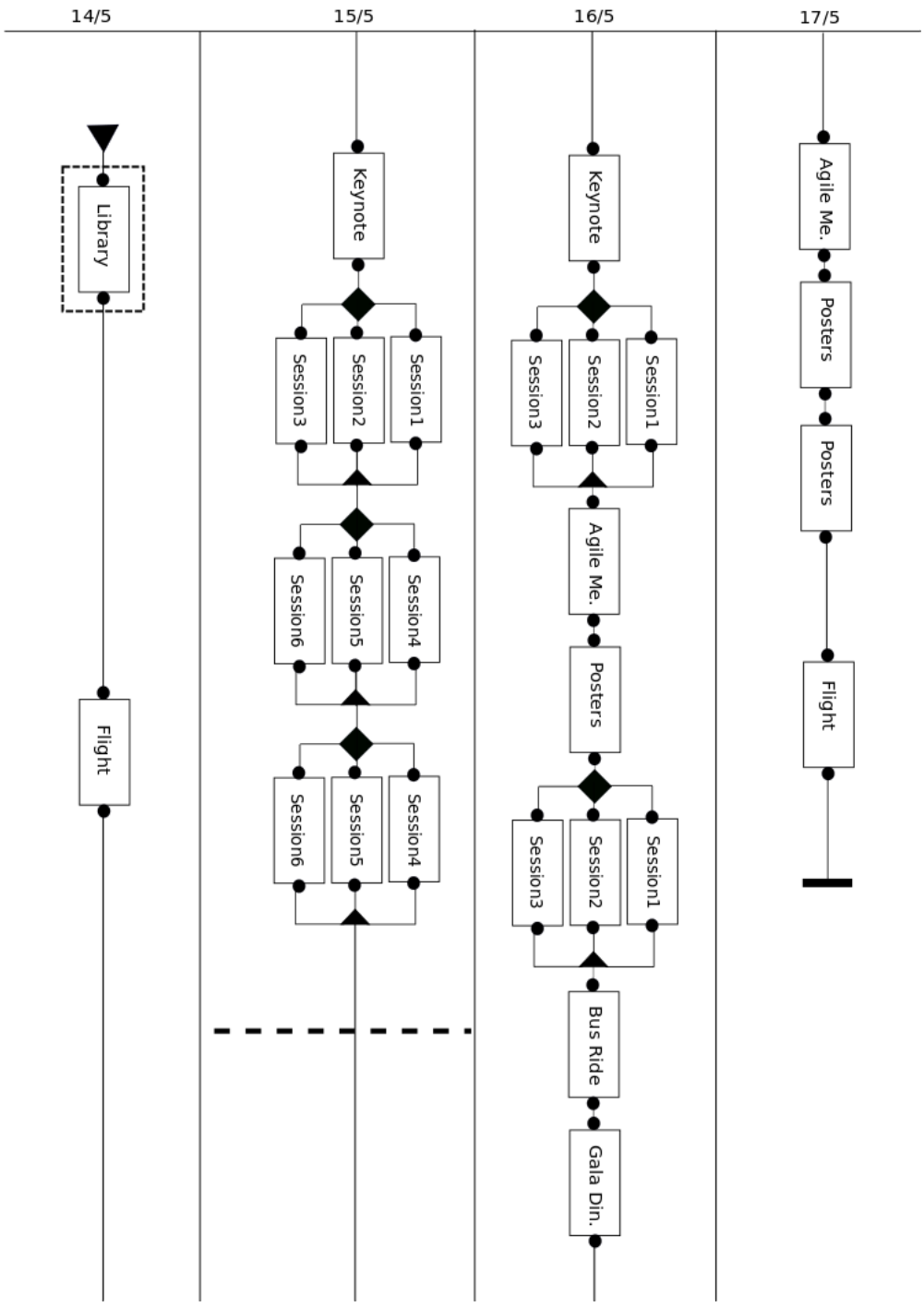

Figure 8.5: The flexible block is put at the first place where it can be achieved, starting from the last possible occurrence depicted by the dashed line.

can be of completely different nature. Common calendar-tools allow to *tag* activities to be distinguishable from each other. The problem is that such activities are often hierarchically structured. So tagging alone, does not suffice. In the implementation a separate index that keeps track of the *conceptual* relationships between the blocks is constructed. Therefore, a new typeclass `IntentionStructures` is introduced:

```
1  newtype Intention = Intention { name :: String } deriving (Eq,Show,Read)
2
3  class (Container intention, Eq intention) => Groups intention where
4
5          type Hierarchy :: *
6          insertIntention :: intention -> intention -> Hierarchy -> Hierarchy
7          getSuperIntention :: intention -> Maybe intention
8          getSubIntentions :: intention -> Hierarchy -> [intention]
9          level :: intention -> Int
10
11 type Identifier = String
12 type Group = [Intention]
13 type IntentionStructure = (Block,Group)
14
15 class IntentionStructures intention where
16
17         insertGroupInto :: intention -> Identifier -> intention -> intention
18         groupInto :: [intention] -> Identifier -> intention
19         addTo :: intention -> intention -> intention
20         returnIntentionById :: Identifier -> intention -> intention
21         returnIntentionByName :: String -> intention -> intention
```

The `groups` class ensures that an *intention* hierarchically groups together. The *Intention*-type (for simplicity reasons represented as a String), implements the `Groups`-class. The `IntentionStructures`-class groups *activity* structures, that is, a tuple consisting of the block structure and the conceptual (intended) activity structure (i.e., `IntentionStructure`) represented as a list of hierarchies.

The operation `groupInto` takes a list of tuples and an `Identifier` and groups the together. For example, three blocks (session1,session2,session3) can be combined to form the *MorningSession*. While the left part of the tuple represents the spatio-temporal structure of it (i.e., a `Alt`-block), the right side is a tree structure with *MorningSession* at the root and three child-nodes. Further, a mapping between the tree nodes and the blocks is required, so that to every name a corresponding block can be retrieved (i.e., `returnIntentionById`-operation). While the block structure keeps track of spatio-temporal interactions, the list of trees separates the activities according to their super-activity.

Blocks, can now be hierarchically structured and retrieved on varying levels. For instance, `Agile.Day1.MorningSession.Session1` would return the corresponding block. Additional operations that deal with updating and inserting blocks in the hierarchy are necessary as well. The complete implementation can be found in Appendix B, Module 13 and Appendix B, Module 12.

8.2.7 The Agent

First a notion of a *present situation* has to be given, so that a block structure can be situated and split into the three conceptual parts: Future, Present and Past. Thus an agent representation is introduced. The agent type follows the formalism of section 6.3.1 and is represented as:

```
type AgentState = ( Place, Date, [Object] )
```

Thus, an agent-state has a place, a time and a set of objects carried.

An agent is represented by the following datatype:

```
data Agent = { agentstate :: AgentState,
               history :: [AgentState] }
```

Thus, it includes a current agent state and a history, that is, a temporally sorted list of previous agent states. Such a history allows to query for a state of an agent at a particular time. It is assumed that an agent-state will be given at the highest resolution possible. The temporal value of the agent representation can serve as a reference point to contextualize blocks as past, present or future blocks. The complete code is found in Appendix B, Module 8.

8.2.8 Personal Information Collection

In Scenario A (4.1.1) the user produces information like pictures and notes. Section 4.3.4 argued that the structure activities is reflected in the personal information collection. To model the personal information collection, personal information objects are defined:

```
data PIO = PIO { id :: String,
                 type :: PIOType,
                 time :: Date}

```

```
class PIS pis where

   getPIitemByActivity :: Identifier -> IntentionStructure -> pis -> pis
   getPIitemByPlace :: Place -> Agent -> pis -> pis
```

The `PIO` datatype stands for an information object that has a name, a type (e.g., .jpg, .pdf, email, contacts etc...) and a *time-stamp* that represents the point in time it was introduced into the personal information collection (i.e., the set of all PIOs). The time property allows to link the information objects to an activity and give it spatial context. The `PIS`-class defines the operations that allow to retrieve personal information objects based on activities, or places.

8.3 Evaluation

This section evaluates the model by describing the use case of Scenario A (4.1.1) in terms of the types and classes previously (see Appendix B, Module 16). It shows how they are used to implement the features presented in section 4. For the evaluation a set of places were stored in a tree (see Appendix B, Module 15) to represent the containment relations, between them. A prototype application (described in Appendix 1), was developed to test the capabilities of the model.

8.3.1 Activity Composition

Section 7.2 described how activities are grouped together. It is implemented by instantiating the `Monoid`-class (section 8.2.6.1) over the `Block`-type. The activities are described as blocks of type `Single` or `Flexible` and are aggregated to form complexes of activities using the `mconcat` operation. The `day1` function returns a block structure that contains the blocks defined before it.

```
flight = Single
        (PTL "Vienna-Brussel"
        (fromString2Date "2013-05-14-18-20",fromString2Date "2013-05-14-20-05")
        (vienna,brussel) [])

bookReturn = Flexible
        (GI "Book return"
        (fromString2Date "2013-05-16-19-00") 10 [tuwien] [])

keynote = Single
         (PTS "keynote"
         (fromString2Date "2013-05-15-09-00",fromString2Date "2013-05-15-10-00")
```

```
13          agileRoomA [])
14
15 session1 = Single
16           (PTS "session1"
17           (fromString2Date "2013-05-15-10-30",fromString2Date "2013-05-15-12-00")
18            agileRoomA [])
19 session2 = Single
20            (PTS "session2"
21            (fromString2Date "2013-05-15-10-30",fromString2Date "2013-05-15-12-00")
22             agileRoomB [])
23 ...
24 day1 = mconcat [keynote,session1,session2,...]
```

While composing the activities, the inability to return the book after the flight is recognized, and the errand of returning a book to the library is automatically shifted to before the flight (see Figure 8.6).

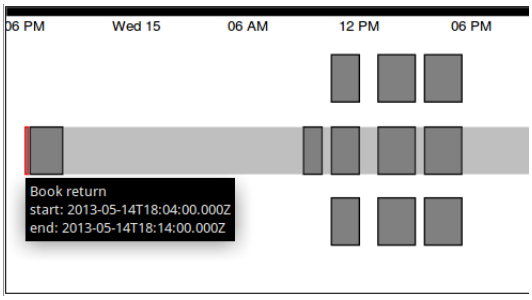

Figure 8.6: The flexible errand of returning a book is not possible to be conducted before the deadline once the flight was taken. The deadline is therefore shifted to the front.

8.3.2 Multi-Granularity

A core component of the system is the ability to represent activities on varying levels of detail. The following operations allow to derive the containing Place of a complex of activities:

```
1 containingPlace :: Block -> Place
2 containingPlace = commonParent . nub $ placeProjection
3
4 commonParent :: [Place] -> Place
5 commonParent (p:ps) =  foldl
6                        (\granule nplace -> commonUpperGranule granule nplace ) p ps
```

These operations, in conjunction with the `returnIntentionById`-operation found in the `IntentionStructure`-class can find the containing places of conceptually related activities, e.g., the conference or the trip as a whole (see Figure 4.9).

8.3.3 Planning Support

The first operation is already defined in the `Block` type-class, i.e., *potential*. It returns place-time lattices or stations that can be used to find opportunities.

```
potential :: block -> [Potentials]

data Potentials = Full Interval ([Place],[Place]) |
                  Half Interval [Place]
```

Potentials are either of the type `Full` or `Half`. The former type is bound by an interval and an start- and end-point, the latter does only have a start-point. By checking, whether places of personal interest are inside the set of places reachable, opportunities can be recognized (for example: are persons from my address book inside the set?). To find all the places that can be reached within a `Potential` of the `Full`-type, the `accessiblePlaces` operation of the `Planner`-class is used (see Figure 4.12).

One problem observed in the user study presented in Section 3.2, was that people often had to pro-actively look for missing steps in the plan. It lead to some users, forgetting to inform themselves about a lag in the travel (e.g., the bus from the airport to the hotel). Others simply had a wrong conceptual model about the spatial relations of the places and therefore simply overlooked the need to travel from one place to another. Using the block model, a system can understand facts about a state of a plan. Therefore the operation `findGaps` (see Appendix B, Module 16) is defined:

```
findGaps :: Block -> SpatialGranules -> [Potentials]
```

It is implemented by interpreting the potentials that have different start- and end-places as *gaps*. By passing a spatial granularity the type of change relevant can be specified, for example only changes on city levels are considered (see Figure 4.10).

8.3.4 Monitoring

To monitor the *present*, a ternary relation has to be constructed, between agent, world and intended activities. One of the benefits of a structured representation that involves tabletop objects is the production of *effective reminders* [Sellen and Whittaker, 2010].

The `block`-class defines the operation `possibleBefore`, which basically implements the $\leq$ ordering relation in the set of blocks (Def. 7.2.1). It was stated that without a notion of the current situation it would not make sense to combine blocks together by

considering requirements. Therefore, the `possibleBefore` function considers spatio-temporal constraints only. Inferences about whether an activity can be achieved before another including required objects, can only be done by adding a current world- and agent-state.
Thus, the `Planner`-class (Section 8.2.5) has to be instantiated by a tuple that consists of an agent-state description and a world description:

```
instance Planner (Date,Place,Equipment) (IO (CT.CTree Place)) where
```

The `Planner`-class uses the agent state description of section 8.2.7, takes the spatio-temporal structure and the underlying requirement-logic into account. The it can therefore answer the question of how long it takes to reach a state from another.
Having these operations and a current world-state in place, the requirements of the next upcoming activity in relation to the activities that are set out for after it, can be computed (see section 4.3.3.1), by using structural recursion, as proposed by [Abdalla and Frank, 2014]. In the case example of Scenario B (4.1.2) it means that the requirement of a *laptop DVD* the student needs for the activities in the afternoon, will be propagated to the morning lecture, meaning that the student needs to pickup the laptop and DVD, before moving to university in the morning. It is therefore an example of a requirement that is dependent on an aggregate of activities rather than simply looking at the next upcoming (see Figure 8.7). It is implemented in the `getRequirement`-operation of the `Block`-class.

```
getRequirements :: block -> [Action]
```

The implemented Scenario B (4.1.2) can be found in Appendix B, Module 16 and seen in Figure 4.13.
Knowing what the requirements for the next upcoming task is and having the `Planner`-class implemented, an alert method can be defined that triggers in case the agent reaches a state where the successful conduction of an intended activity is threatend.

```
alert :: Agent -> World -> Block -> Date
```

This method can be utilized to support prospective remembering as was shown by Abdalla and Frank [2014].

8.3.5 Structuring the Past

By having a history of the agent behavior the actual path that was taken when moving through the block structure (i.e., by *intersecting* the place-time path with the block

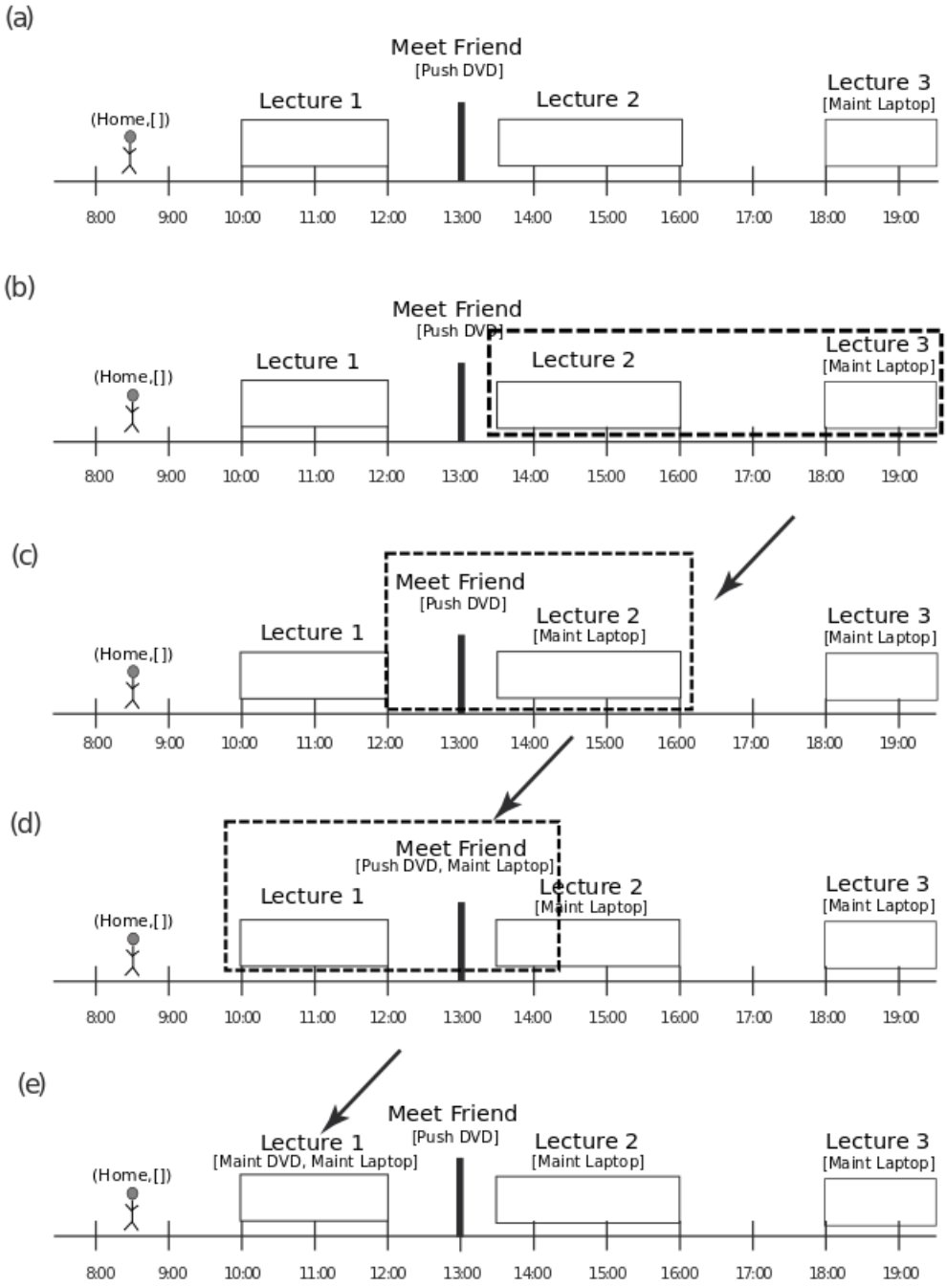

Figure 8.7: The process of propagation illustrated. The structural recusrion starts from the right side and evaluates a pair of two. If there is a need to acquire a requirement of a latter before the former, it is propagated to the prior task.

structure) can be inferred. Thus, the intersection of the actual path and a structure of activities delivers a semantically annotated space-time path. There are two possible applications that are made possible through this approach: (1) patterns in the activity-sequences can be recognized that than allow to automatically fill out missing parts for future activities; (2) retrieval of personal information that was produced throughout the activities. In case of retrieval, checking for spatio-temporal containment the information can be related to the activities conducted. The following operation does allow to retrieve personal information on the basis of an activity:

```
1 class PIS pis where
2
3     getPIitemByActivity :: Identifier -> IntentionStructure -> pis -> pis
4     getPIitemByPlace :: Place -> Agent -> pis -> pis
```

The operation `getPIitemByActivity` takes an identifier (i.e., `Name`), a block and tree structure, as well as the set of `PIO` types as an input and produces the set of `PIO` types that correspond to the activity. For example, the digital notes and pictures the user in Scenario B (4.1.2) took while at the conference (see Figure 4.16). By having a granular representation of space and time as well as of the activity itself (i.e., the whole conference vs. a session), it allows to ask for personal information at different levels (see Figure 4.15 and Figure 4.16).

Chapter 9

Conclusion and Future Work

This chapter gives a summary of the findings. Several points that need further investigation are listed, in order to achieve a next generation personal geographic assistant application.

9.1 Conclusion

The previous pages have taken a comprehensive look at how human activities are determined and represented. It was shown that by using an adequate model it is possible to not only help forming plans, but also to monitor existent ones, as well as to structure the *personal information space* by the information about conducted activities.

Therefore, a multi-granular and extendable model (e.g., by requirements) that uses place and time as a fundamental structuring principle was proposed. Having such a structured model of human activity improves the entire spectrum of personal information management. Calendar and task management applications can use it to have a better understanding of the intended activities and recognize gaps or opportunities. Intelligent reminders can be triggered in case people forgot about things to support prospective remembering. Personal information produced on different devices can be indexed and retrieved, based on the information given in an activity model.

It was found that a useful representation of activities has to deal with granular representations, on a spatio-temporal as well as conceptual level. These system of granularities have to offer a degree of flexibility so that users can define their own granular representations and relations. The addition of object requirements to activities rendered the initial assumptions of a static environment invalid. To simulate the conduction

of future activities under the assumption of requirements, the environmental changes need to be considered. Therefore, it makes no sense to structure activities based on requirements for the future, since the dynamic nature of table-top sized objects can render the ordering invalid. The solution proposed is the inclusion of requirements for a *current* situation, based on the underlying spatio-temporal structure of the activities.

To reach a point where an operational system can actually perform the above mentioned capabilities, many technical as well as theoretic obstacles have to be resolved first. In the following, some of those issues are listed in order to motivate further research in the area.

9.2 Future Work

Ontology Integration The main goal of this work is to bridge the gap between the physical and virtual world. While the model presented, focused on the description of a physical activity, research in PIM is in most cases concerned about virtual activities (i.e., activities preformed on the computer desktop). The question that follows is how to combine these, seemingly interdependent, forms of activities. For example, a conference requires a registration, which nowadays is (mostly) an online procedure. Thus, ontologies that describe *desktop*-activities,a s for example the one presented by a task-description language [Catarci et al., 2006], have to be integrated with ontologies that describe real world activities. Hypothetically it should be possible to simply extend the agent-state description by a factor, representing virtual requirements, such as an confirmation email, that would then afford activities that change this dimension of the agent-representation.

Feedback recognition An essential feature, is *feedback-recognition*, i.e., the system has to be able to recognize the connection between a requirement and *the meeting of the requirement*. This is true for the physical, as well as for the virtual world. The conference registration example, can involve a payment and does finish with a confirmation, usually received by email. Similarly, an object requirement is met when the object is in the possession of the agent at a specified time and place. So both, the confirmation email as well as the acquisition of the physical object, should be recognized as *the meeting of a requirement*.

In the case of virtual activities the developments in the semantic desktop *NEPOMUK*[1]

[1] https://userbase.kde.org/Nepomuk

or its recent replacement *Baloo*[1] are promising starting points. These frameworks allow to annotate and link between files or applications, to categorize and enable new ways of search for documents. Links can potentially be drawn between the schedule entries and the emails confirming a registration.

As for the recognition of possessing objects, different ways of tracking the objects are possible of, e.g., using RFID-tags to read them, image recognition to register them or simply a system of QR-codes to scan them. Which one to use, is determined by a trade-off between usability and reliability.

Pattern recognition and learning In Section 3.1.2 it was stated that planning skills become more advanced, once several smaller procedures are packed into a single descriptor. Many of the activities we do are routines, i.e., regularly repeated. Consequently, the system should recognize typical activities, so that the user can put a keyword, and the system understands the rest. A learning system can acquire information about the activities, like, when they are mostly conducted, where they happen, what they require; and subsequently free the user from giving every detail. For example, a typical visit of the grandparents, that takes 2-3 hours and is preceded by the acquisition of some flowers. The pattern of such an activity can be extracted and stored, so that when the user enters the visit the flowers will be added. Another example is the visit to the gym after work that requires a bag. Having basic information about several routine activities, can improve recommendation services as to when is a good place to do so, as well as usability of it, by reducing the need to give details about every activity.

Activity extraction Tasks and activities are communicated in textual form, i.e., emails, text messages, or phone calls. Parsing those textual description and transforming it into a task is desirable, since it can automatically detect a meeting that was arranged on the phone or by text message. It is here were a multi-granular model is necessary, since such textual descriptions often meetings are often imprecise and are gradually refined (e.g., lets meet in downtown) as more details start to emerge.

Sharing One of the factors that will determine the success of an activity model, is its share-ability. The most useful applications of an activity model, will be the ability to describe activity structures (such as conferences, or festival schedules, etc...) and make

[1] https://community.kde.org/Baloo

the available to the public or a given audience. Activity structures will ideally have requirements as well as places incorporated, to allow a personal assistant application to semi-automatically fill in the steps needed for a specific user context. Ultimately, such activities can be searched or explored in a central database that can either be crowd sourced or mined from various datasources.

Privacy The last point, barely mentioned throughout this work, but deemed to be important for future work, is the question of privacy. A good understanding of a user's intentions and activities does certainly help to support daily life, but at the same time opens potential for abuse. It is in the best case a source for targeted advertisement and in the worst a source to prosecute people for participation in *illegal activities*, a concept that has always been subject to interpretation. It is up to researchers to find suitable system architectures, that avoid central storage of data, which makes it easy prey.

Appdx A

For the evaluation a prototype application was developed. It consists of a yesod[1] web-server back-end and a web-based front end. Visualizations were implmented in D3[2] and Leaflet[3]. The GUI allows to switch between three different views: Future (see Figure 1), Present (see Figure 2) and Past (see Figure 3).

[1] www.yesodweb.com
[2] d3js.org
[3] leafletjs.com

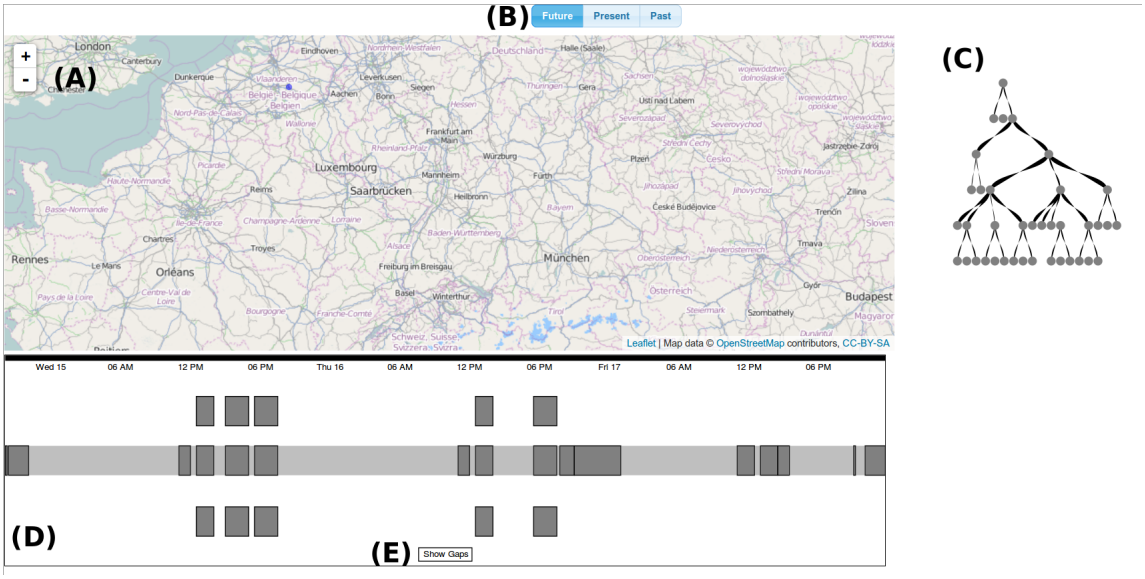

Figure 1: The starting perspective is the Future. (A) A map that shows the corresponding places to the activities. (B) Allows to switch between the prespectives. (C) A tree representation of the conceptual relations between the activities. Clicking on the nodes highlights the groups of activities. (D) The time map visualizes the planned activities. Activities in parallel stand for alternative choices. (E) Shows the gaps that are recognized by the system. Clicking the light gray bars (potentials) retrieves the places reachable between two bounding activities.

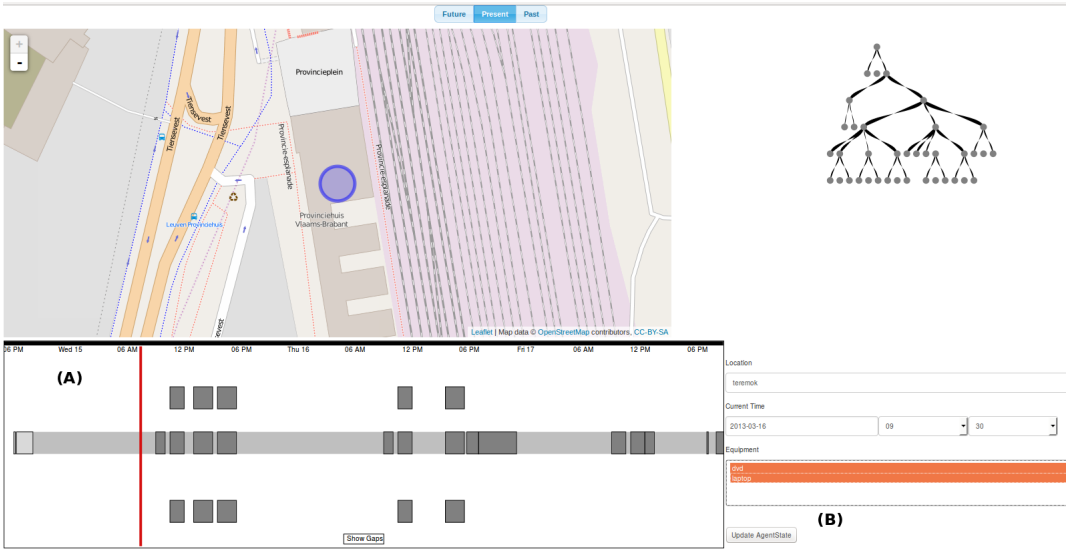

Figure 2: The present perspective adds an agent context and grays out past activities (A). (B) gives an interface to update the agent state. Accordingly, reminders can be computed.

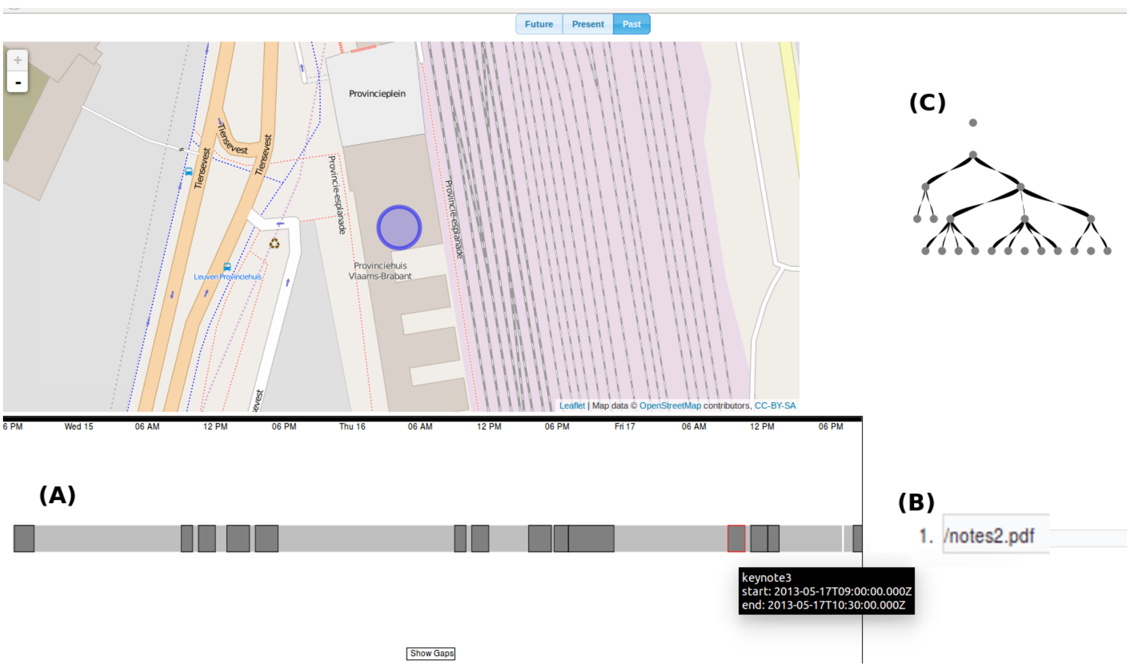

Figure 3: The past perspective shows only activities that are stored in the agent's history. By moving over an activity (A), the corresponding place and personal information objects are retrieved (B).

Appdx B

In Appendix B the complete set of modules and their containing code are listed.

Listing 1: Geography

```
module BackendModules.Geography where

import Prelude
import Data.Tree
import System.Directory
import System.IO.Unsafe (unsafePerformIO)
import Control.Applicative
import BackendModules.Equipment
import BackendModules.Granularity
import BackendModules.Place
import qualified BackendModules.TimeHandle as Time
import BackendModules.ContainmentTree

-- ************************ Type Declarations ************************ --

type KB = CTree Place

class Geography placeDB where
        -- Place related operations and questions
        addPlace :: Place -> placeDB -> IO ()
        updatePlace :: (Place -> Place) -> Place -> placeDB -> IO ()
        parentPlace :: Place -> placeDB -> IO (Maybe Place)
        containingPlaces :: placeDB -> Place -> IO [Place]
        placePerGranule :: SpatialGranules -> placeDB -> IO [Place]
        closePlaces :: Place -> placeDB -> IO [Place]

        -- Object Related operations
        queryObj :: placeDB -> TableObject -> [Place]
        queryObjAt :: placeDB -> nodeID -> TableObject
        queryAllObj :: placeDB -> TableObject

-- ***************************** Instances ***************************** --

instance Geography (IO KB) where
```

```
35
36    parentPlace place kb | granularity place == World =
37                                          return $ Just place
38                         | otherwise = do
39                                        kb'<- kb
40                                        return $
41                                        node2Elem <$>
42                                         getParent spatiallyContains kb' place
43
44    containingPlaces kb place = do
45                    kb' <- kb
46                    return $
47                    map node2Elem $
48                    getChildren spatiallyContains kb' place
49
50     containingPlacesPerType place t kb =
51                   do
52                    kb' <- kb
53                    return
54                    [node2Elem p | p <- getChildren spatiallyContains kb' place,
55                    placeType (node2Elem p) == t]
56
57    placePerGranule g kb = do
58                     kb' <- kb
59                     return $
60                     preorderWith (\place -> granularity place == g) kb'
61
62    closePlaces place kb = do
63              kb' <- kb
64              let quicksort [] = []
65                  quicksort (p:xs) =
66                            quicksort (lesser p xs)
67                                      ++ [p]
68                                      ++ quicksort (greater p xs)
69                   lesser p xs =
70                        [ a | a <- xs , euclideanDist a place < euclideanDist a p]
71                   greater p xs =
72                     [ a | a <- xs , euclideanDist a place >= euclideanDist a p]
73               return . (\x -> if x == Nothing then [] else (\(Just y) -> y) x ) $
74                         tail . quicksort <$>
75                         preorderWith (\p -> granularity p == granularity place) <$>
76                         getParent spatiallyContains kb' place
77
78
79  instance Granularity Place where
80
81    coarsen p' = parentPlace p
82
83    isFinerThan p1 p2 = (granularity p1) < granularity p2
84
85  -- The dbase contains a granular representation of places
86  dbase = read <$> (readFile "BackendModules/Dbase/treefile.txt") :: IO (CTree Place)
```

Listing 2: Place

```haskell
module BackendModules.Place where

import Prelude
import GHC.Generics
import Control.Monad (liftM2)
import Control.Applicative
import BackendModules.Equipment
import BackendModules.Container
import BackendModules.Granularity
import BackendModules.SpatialRelations

-- ************************* Type Declarations ************************* --

type Loc = (Double,Double)
data TravelMode = Foot |
                  Car |
                  Bike
                  deriving (Show,Eq,Enum,Bounded,Read,Generic)

data SpatialGranules = MeetingPoint |
                       Room |
                       Building |
                    Park |
                       Neighbourhood |
                       City |
                       Country |
                       Continent |
                       World
                       deriving (Show,Eq,Read,Bounded,Enum,Generic)

data Geometry = BBOX {
              n :: Double,
              w :: Double,
              s :: Double,
              e :: Double }
              | ABS {point :: Loc}
              deriving (Show,Read,Eq,Generic)

data Place = Place {
                  placeID :: Int,
                  placeName :: Text,
                  placeLoc :: Geometry,
                  granularity :: SpatialGranules,
                  getObjs :: [TableObject]
                     }
                  deriving (Show,Generic,Read)

class Places place where

   spatiallyContains :: place -> Place -> Bool
   transportDist :: TravelMode -> Place -> place -> IO Double
```

```
52    euclideanDist :: Place -> place -> Double
53    estimatedTravelTime :: place -> place -> Double
54    pointRepresentation :: place -> Loc
55     containsObject :: place -> TableObject -> Bool
56
57  -- **************************** Instances ***************************** --
58
59  instance Places Place where
60
61    pointRepresentation = centre . placeLoc
62             where
63             centre (ABS (x,y)) = (y,x)
64             centre (BBOX n' w' s' e') =
65                          (n'- ((n' - s') / 2),e' - ((e' - w') / 2))
66
67     euclideanDist p1 p2 =
68              dist' (pointRepresentation p1) (pointRepresentation p2)
69               where
70              dist' (x1,y1) (x2,y2) =
71                           sqrt $ (x2 - x1)^2 + (y2 - y1)^2
72
73    spatiallyContains p1 p2 = placeLoc p1 `contains'` placeLoc p2
74                          where
75                          contains' (ABS a) (ABS b) = a == b
76                          contains' (ABS _) _ = False
77                          contains' bbx (ABS b) =
78                             w bbx <= fst b &&
79                               e bbx >= fst b &&
80                              n bbx >= snd b &&
81                                s bbx <= snd b
82                          contains' bbx (BBOX n' w' s' e') =
83                                  contains' bbx (ABS (w',n')) &&
84                                  contains' bbx (ABS (e',s'))
85
86    containsObject place obj = elem obj $ getObjs place
87
88    estimatedTravelTime p1 p2 | meter <= 200 = (meter / foottravelspeed) / 60
89                              | otherwise = (meter / cartravelspeed) / 60
90                                 where
91                                 meter = (euclideanDist p1 p2) * 100000
92                                 cartravelspeed = 10.2
93                                 foottravelspeed = 0.5
94
95  instance Ord SpatialGranules where
96
97    compare Room MeetingPoint = EQ
98    compare Room _ = LT
99
100    compare Building MeetingPoint = EQ
101    compare Building Room = GT
102    compare Building _ = LT
103
104   compare MeetingPoint Room = EQ
```

```
105     compare MeetingPoint Building = EQ
106     compare MeetingPoint _ = LT
107
108     compare Park Country = LT
109     compare Park Neighbourhood = LT
110     compare Park City = LT
111     compare Park World = LT
112     compare Park _ = GT
113
114     compare Neighbourhood City = LT
115     compare Neighbourhood Country = LT
116     compare Neighbourhood Continent = LT
117     compare Neighbourhood World = LT
118     compare Neighbourhood _ = GT
119
120     compare City Country = LT
121     compare City Continent = LT
122     compare City World = LT
123     compare City _ = GT
124
125     compare Country Continent = LT
126     compare Country World = LT
127     compare Country _ = GT
128
129     compare Continent World = LT
130     compare Continent _ = GT
131
132      compare World _ = GT
133
134 instance Eq Place where
135
136    p == p1 = (placeID p) == placeID p1
137
138 instance Container Place where
139
140    contains p1 p2 = spatiallyContains p1 p2
141
142 instance SpatialRelations Place where
143
144     spatialRel p1 p2 | contains p1 p2 = SContains
145                      | contains p1 p2 = SContainedBy
146                      | euclideanDist p1 p2 <= 0.2 = SOverlaps
147                      | euclideanDist p1 p2 <= 0.2 = STouches
148                      | otherwise = Disjoint
149
150 instance SpatialRelations [Place] where
151
152     spatialRel p1 p2  | contains p1 p2 = SContains
153                       | contains p2 p1 = SContainedBy
154                       | or $ liftM2 contains p1 p2 = STouches
155                       | otherwise = Disjoint
156
157 instance Container [Place] where
```

```
158
159         contains p1 p2 = Prelude.foldl isContained True p2
160                 where
161                 isContained True p = or $ [ contains p' p | p' <- p1]
162                 isContained False _ =  False
```

Listing 3: Equipment

```
1   module BackendModules.Equipment where
2
3   type Name = String
4   type Equipment = [TableObject]
5   type Conditions = Equipment
6
7   data TableObject = Obj {objName :: String}
8                      deriving (Show,Read,Eq,Generic)
9
10  data Action = Pull TableObject |
11                Push TableObject |
12                Maint TableObject
13                deriving (Eq,Show,Read,Generic)
```

Listing 4: SpatialRelations

```
1   module BackendModules.SpatialRelations where
2
3   import Prelude
4
5   data SpatialRelation = SContains |
6                          SContainedBy |
7                          STouches |
8                          SOverlaps |
9                          Disjoint
10                         deriving (Eq,Show,Enum)
11
12  class SpatialRelations object where
13
14          spatialRel :: object -> object -> object
```

Listing 5: Granularity

```
1   module BackendModules.Granularity where
2
3   import Prelude
4
5   class Granularity granule where
6
7           coarsen :: granule -> granule
8           commonUpperGranule :: (Eq granule) => granule -> granule -> granule
```

```
9           commonUpperGranule g1 g2 | g1 `equals` g2 = g1
10                                   | g1 `isFinerThan` g2 = commonUpperGranule (coarsen g1
    ) g2
11                                   | g2 `isFinerThan` g1 = commonUpperGranule g1 (coarsen
     g2)
12                                   | coarsen g1 == coarsen g2 = coarsen g1
13                                   | otherwise = commonUpperGranule (coarsen g1) (coarsen
     g2)
14          isFinerThan :: granule -> granule -> Bool
15          equals :: (Eq granule) => granule -> granule -> Bool
16          equals a b = a == b
```

Listing 6: Container

```
1  module BackendModules.Container where
2
3  import Prelude
4
5  class Container x where
6
7    contains :: x -> x -> Bool
```

Listing 7: AllenRelations

```
1  module BackendModules.AllenRelations where
2
3  import Prelude
4
5  data TemporalRelation = Before |
6                          After |
7                          During |
8                          Equal |
9                          Contains |
10                         Overlaps |
11                         Overlapped |
12                         Meets |
13                         MetBy  |
14                         Starts |
15                         StartedBy |
16                         Finishes |
17                         FinishedBy
18                         deriving (Eq,Enum,Show)
19
20 class TemporalRelations event where
21
22     relation :: event -> event -> TemporalRelation
23     availableRelations :: event -> event -> [TemporalRelation]
24
25 inverse Before = After
26 inverse After = Before
27 inverse During = Contains
```

```
inverse Contains = During
inverse Overlaps = Overlapped
inverse Overlapped = Overlaps
inverse Meets = MetBy
inverse MetBy = Meets
inverse Starts = StartedBy
inverse Finishes = FinishedBy
inverse FinishedBy = Finishes
inverse Equal = Equal
```

Listing 8: Agent

```
module BackendModules.Agent where

import Prelude
import Data.Time
import BackendModules.Place
import BackendModules.Equipment
import BackendModules.TimeHandle

data Agent = Agent { agentLoc :: Place,
                     agentTime :: UTCTime,
                     agentEqu :: [TableObject]} deriving (Eq,Show,Read)
```

Listing 9: TimeHandle

```
module BackendModules.TimeHandle where
import Data.List.Split
import Data.List
import Data.Time
import Data.Time.Format
import Data.Time.Clock
import Data.Typeable
import Control.Applicative
import Data.Time.Clock.POSIX
import BackendModules.Container
import BackendModules.Granularity
import Prelude


-- ************************* Type Declarations ************************* --

type Minutes = Int

data TempGranules = YearG |
                    MonthG |
                    DayG |
                    HourG |
                    MinuteG
                    deriving (Eq,Enum,Ord,Show,Read)

```

```
26 data Date = Date { y :: Int ,
27                    m :: Int,
28                    d :: Int,
29                    hh :: Int,
30                    mm :: Int ,
31                    tgranularity :: TempGranules }
32                           deriving (Eq,Show,Read)
33
34 -- ***************************** Instances ***************************** --
35
36 instance Ord Date where
37
38     d1 < d2 | (y d1) < (y d2) = True
39             | (y d1) == (y d2) && (m d1) < (m d2) && tgranularity d1 == YearG = False
40             | (y d1) == (y d2) && (m d1) < (m d2) = True
41             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) < (d d2) && tgranularity
     d1 == MonthG = False
42             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) < (d d2) = True
43             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) == (d d2) && (hh d1) < (hh
      d2) && tgranularity d1 == DayG = False
44             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) == (d d2) && (hh d1) < (hh
      d2) = True
45             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) == (d d2) && (hh d1) == (
     hh d2) && (mm d1) < (mm d2) && tgranularity d1 == HourG = False
46             | (y d1) == (y d2) && (m d1) == (m d2) && (d d1) == (d d2) && (hh d1) == (
     hh d2) && (mm d1) < (mm d2) = True
47             | tgranularity d1 == tgranularity d2 = False
48             | otherwise = True
49
50     d1 > d2 = d2 < d1
51
52     d1 <= d2 = d1 < d2 || d1 == d2
53
54     d1 >= d2 = d1 > d2 || d1 == d2
55
56 instance Container Date where
57
58  contains date1 date2 | date1 == date2 = True
59                       | date1 `isFinerThan` date2 = False
60                       | otherwise = contains date1 (coarsen date2)
61
62
63 instance Granularity Date where
64
65    coarsen date | tgranularity date == YearG = date
66             | tgranularity date == MonthG = date {m = 0, tgranularity = YearG}
67             | tgranularity date == DayG = date {d = 0, tgranularity = MonthG}
68             | tgranularity date == HourG = date {hh = 0, tgranularity = DayG}
69             | otherwise = date {mm = 0, tgranularity = HourG}
70
71    isFinerThan d1 d2 = d1 <= d2
72
73 -- ************************ Utilities ************************ --
```

```
74
75  adding :: Minutes -> UTCTime -> UTCTime
76  adding y t = addUTCTime (((fromInteger . toInteger) y :: POSIXTime)*60) t
77
78  subtractUTCTime :: NominalDiffTime -> UTCTime -> UTCTime
79  subtractUTCTime x t = posixSecondsToUTCTime ((utcTimeToPOSIXSeconds t) - x)
80
81  subtracting :: Minutes -> UTCTime -> UTCTime
82  subtracting y t =  subtractUTCTime (((fromInteger . toInteger) y :: POSIXTime)*60) t
83
84  -- takes a String in the format YYYY-MM-DD-HH-MM and transforms it into a Date type, !!
        no checking for syntax validity !!
85  fromString2Date str = let splittedList = splitOn "-" str
86                        in makeDate splittedList 0
87                        where
88                        makeDate [] _ = error "No valid String"
89                        makeDate ls@(x:xs) position = if xs == [] then Date (read x ::
        Int) 0 0 0 0 YearG else
90                                                      if length ls == 2 then Date (read x ::
        Int) (read (head xs) :: Int) 0 0 0 MonthG else
91                                                    if length ls == 3 then Date (read x :: Int) (
        read (head xs) :: Int) (read (xs !! 1) :: Int) 0 0 DayG else
92                                                     if length ls == 4 then Date (read x :: Int
        ) (read (xs !! 0) :: Int) (read (xs !! 1) :: Int) (read (xs !! 2) :: Int) 0 HourG
        else
93                                                   Date (read x :: Int) (read (xs !! 0) :: Int)
         (read (xs !! 1) :: Int) (read (xs !! 2) :: Int) (read (xs !! 3) :: Int) MinuteG
94
95
96  fromDate2UTC d' | (tgranularity d') == YearG = UTCTime (fromGregorian (toInteger $ y d
        ') 0 0) (timeOfDayToTime $TimeOfDay 0 0 0)
97                  | (tgranularity d') == MonthG = UTCTime (fromGregorian (toInteger $ y d
        ') (m d') 0) (timeOfDayToTime $TimeOfDay 0 0 0)
98                  | (tgranularity d') == DayG = UTCTime (fromGregorian (toInteger $ y d')
         (m d') (d d')) (timeOfDayToTime $TimeOfDay 0 0 0)
99                  | (tgranularity d') == HourG = UTCTime (fromGregorian (toInteger $ y d
        ') (m d') (d d')) (timeOfDayToTime $TimeOfDay (hh d') 0 0)
100                 | otherwise = UTCTime (fromGregorian (toInteger $ y d') (m d') (d d'))
        (timeOfDayToTime $ TimeOfDay (hh d') (mm d') 0)
101
102 fromUTC2Date d = let string = filter (\x -> x /= "" && x /= "UTC") (splitOneOf "-: " (
        show d))
103                  in fromString2Date $ intercalate "-" string
104
105 absTimeDiff :: Date -> Date -> NominalDiffTime
106 absTimeDiff et st = diffUTCTime (fromDate2UTC et) (fromDate2UTC st)
```

Listing 10: Planner Module

```
1  module BackendModules.Planner where
2
3
```

```
4  import System.IO.Unsafe
5  import BackendModules.TimeHandle
6  import BackendModules.Place
7  import BackendModules.Geography
8  import BackendModules.Equipment
9  import BackendModules.Granularity
10 import qualified BackendModules.Container as Container
11 import qualified BackendModules.ContainmentTree as CT (CTree)
12 import Control.Applicative ((<$>))
13 import Data.Monoid (mappend)
14 import Prelude
15 import Data.List
16
17 class (Geography world) => Planner intention world | intention -> world where
18
19         achievable :: world -> intention -> intention -> Bool
20         availableTime :: intention -> intention -> Double
21         requiredTime :: intention -> intention -> IO Double
22         accessiblePlaces :: world -> intention -> intention -> IO [Place]
23         reachableInTime :: world -> intention ->  IO [Place]
24
25 instance Planner (Date,Place) (IO (CT.CTree Place)) where
26
27    achievable _ (d,p) (d',p')
28       | d > d' = False
29       | Container.contains p p' && d < d'= True
30       | Container.contains p' p && d' < d = True
31        otherwise =  unsafePerformIO $ do
32                           requiredTime <- requiredTime (d,p) (d',p')
33                           return $ (availableTime (d,p) (d',p')) >= requiredTime
34
35   availableTime (d,p) (d',p') =
36           (/60) . fromRational . toRational $ (absTimeDiff d' d) :: Double
37
38   requiredTime (d,p) (d',p') = return $ estimatedTravelTime p p'
39
40   accessiblePlaces kb (sd,sp) (ed,ep) =
41                         filter accessible <$> allPs
42                         where
43                         allPs = do
44                                 cps <- closePlacesOfParents
45                                 cps' <- sequence $ map (containingPlaces dbase) cps
46                                 return $ (concat cps') ++ cps
47
48  closePlacesOfParents = closePlaces (getParentPlace sp) dbase
49
50  getParentPlace p = unsafePerformIO $
51                       (\(Just x)->x) <$> parentPlace p kb
52
53  accessible p = ((timeGo p) + (timeLeave p)+ 5) <= availableTime (sd,sp) (ed,ep)
54                                          timeGo p = estimatedTravelTime sp p
55                                          timeLeave p = estimatedTravelTime p ep
56
```

```haskell
instance Planner (Date,Place,Equipment) (IO (CT.CTree Place)) where

  achievable _ t1@(d,p,objs) t2@(d',p',objs')
           | d > d' = False
           | objs == objs' = achievable dbase (d,p) (d',p')
           | otherwise =  unsafePerformIO $
                       do
                       requiredTime <- requiredTime t1 t2
                       return $ (availableTime t1 t2) >= requiredTime

  availableTime (d,_,_) (d',_,_) = (/60) .
                             fromRational .
                             toRational $ (absTimeDiff d' d) :: Double

  requiredTime  (d,p,objs) (d',p',objs')
              | objs' == objs = return $ estimatedTravelTime p p'
              | otherwise = return . fst $ requiredTime'
                             where
                 requiredObjs = objs' \\ objs
                 objPlace = home
                 -- ^ here a query function
                 that returns the places that
                 contain the objects is required
                 requiredTime' =
                     foldl
                     (\(accTime,place) newPlace ->
                     (accTime + estimatedTravelTime place newPlace),newPlace) )
                     (0,p) [objPlace,p']

instance Planner (Date,Place,[Action]) (IO (CT.CTree Place)) where

   achievable _ s1@(d,p,r) s2@(d',p',r') =
                    achievable dbase (d,p,toEqu r) (d',p',toEqu r')
               where
               toEqu r = map unAction $ filter noPushs r
               unAction (Pull o) = o
               unAction (Maint o) = o
               noPushs (Push _) = True
               noPushs _ = False
```

Listing 11: ActivityBlocks Module

```haskell
module BackendModules.ActivityBlocks where

import Prelude
import Data.List
import Data.Monoid
import GHC.Generics
import Data.Aeson
import Foreign (unsafePerformIO)
import Control.Applicative ((<$>))
import Data.List
```

```
11 import Control.Monad (liftM2)
12 import BackendModules.TimeHandle
13 import BackendModules.AllenRelations
14 import BackendModules.Place
15 import BackendModules.Equipment
16 import BackendModules.Planner
17 import BackendModules.SpatialRelations
18 import BackendModules.Agent
19 import qualified BackendModules.Container as Container
20 import BackendModules.Geography (dbase, containingPlacesPerType, parentPlace,
       closePlaces,containingPlaces)
21 import qualified BackendModules.ContainmentTree as CT (CTree)
22 import BackendModules.Granularity (isFinerThan, coarsen)
23
24
25 -- ************************ Type Declarations ************************ --
26
27 type ID = String
28 type Descr = String
29 type Interval = (Date,Date)
30 type Due = Date
31 type Duration = Int
32
33 data Activity = PTS ID Interval Place [Action]
34               | PTL ID Interval (Place,Place) [Action]
35                deriving (Eq,Show,Read)
36
37 data Errand = GI ID Due Duration [Place] [Action] deriving (Eq,Show,Read)
38
39 data Potentials = Full Interval ([Place],[Place]) | Half Interval [Place] deriving (Eq,
       Show,Read,Generic)
40
41 data Block = Single Activity |
42              Flexible Errand |
43              Seq [Block] |
44              Alt [Block]  deriving (Eq,Show,Read)
45
46 class Blocks block where
47
48         startTime :: block -> Date
49         startTime = fst . temporalProjection
50
51         endTime :: block -> Date
52         endTime = snd . temporalProjection
53
54         startPlace :: block -> [Place]
55         endPlace :: block -> [Place]
56
57         getRequirements :: block -> [Action]
58
59         flatten :: block -> [block]
60
61         potential :: block -> [Potentials]
```

```
62
63          blockConcat :: block -> block
64
65          temporalProjection :: block -> Interval
66
67          placeProjection :: block -> [Place]
68
69          temporallyContained :: block -> block -> Bool
70          temporallyContained b1 b2 =
71                 let t1 = temporalProjection b1
72                     t2 = temporalProjection b2
73                     in case relation t2 t1 of
74                        Contains -> True
75                        StartedBy -> True
76                        FinishedBy -> True
77                        Equal -> True
78                            _ -> False
79
80          possibleBefore :: block -> block -> Bool
81          possibleAfter :: block -> block -> Bool
82          possibleAfter b1 b2 = possibleBefore b2 b1
83
84          possibleWithin :: block -> block -> Bool
85
86          parallel :: block -> block -> Bool
87          parallel b1 b2 | possibleBefore b1 b2 || possibleBefore b2 b1 = False
88                         | otherwise = True
89
90          chainAble :: block -> block -> Bool
91          chainAble a b = possibleBefore a b || possibleBefore b a
92
93          mutuallyExclusive :: block -> block -> Bool
94          mutuallyExclusive b1 b2 = not . or $ map (chainAble b1) (flatten b2)
95
96  -- ****************************** Instances ****************************** --
97
98  instance Monoid Block where
99
100         mempty = Seq []
101
102         (Seq []) `mappend` b = b
103         b `mappend` (Seq []) = b
104
105 -- ============================  Single - * ============================ --
106
107         mappend b1@(Single _) b2@(Single _)
108                 | possibleBefore b1 b2 = Seq [b1,b2]
109                 | possibleBefore b2 b1 = Seq [b2,b1]
110                 | otherwise = Alt [b1,b2]
111
112         mappend b1@(Single _) b2@(Flexible gi)
113                 | possibleBefore b1 b2 = Seq [b1,b2]
114                 | possibleBefore b2 b1 = Seq [b2,b1]
```

```
115                 | otherwise = Seq [shiftToFront b2 b1,b1]
116
117         mappend single@(Single b) b1@(Seq as)
118                 | possibleBefore single b1 = Seq $ single : as
119                 | possibleBefore b1 single = Seq $ as ++ [single]
120                 | possibleWithin single b1 = mconcat . (sortBy orderFunction) $ single
    : flatten b1 -- instead of merge function??
121                 | not $ mutuallyExclusive single b1 = blockConcat . Seq $ [mappend
    single (makeSeq parallelBlocks)] ++ (as \\ parallelBlocks)
122                 | otherwise = Alt [b1,single] -- mutually exclusive
123                  where
124                 parallelBlocks = [ a | a <- as, parallel single a]
125                 makeSeq bs = if length bs > 1 then Seq bs else head bs
126
127         mappend single@(Single b) b1@(Alt as)
128                 | possibleBefore single b1 = Seq $ [single,b1]
129                 | possibleBefore b1 single = Seq $ [b1,single]
130                 | mutuallyExclusive single b1 =  Alt $ single : as
131                 | otherwise = Alt merged
132                 where
133                 priors = (filter (possibleBefore single) as)
134                 --^^ all blocks possible prior to the new block
135                 afters = (filter (possibleAfter single) (as \\ priors))
136                 --^^ all after it
137                 parallels = (as \\ priors) \\ afters
138                 --^^ those mutually exclusive
139                 merged
140                    | null priors && null afters = parallels
141                    | null priors && (not . null) afters =
142                                    Seq [single,Alt afters] : parallels
143                    | (not . null) priors && null afters =
144                                    Seq [Alt priors,single] : parallels
145                    | otherwise = Seq [single,Alt afters]
146                                  : Seq [Alt priors,single]
147                                  : parallels
148
149 -- ============================  Flexible - * ============================== --
150         mappend b1@(Flexible _) b2@(Single _) = mappend b2 b1
151
152         mappend b1@(Seq bs) b2@(Flexible gi)
153                 | possibleBefore b1 b2 = Seq [b1,b2]
154                 | possibleBefore b2 b1 = Seq [b2,b1]
155                 | otherwise = shift b2 (reverse bs) []
156                 where
157                 shift fb (x:[]) rest = Seq $ (shiftToFront b2 x) : (x:rest)
158                 shift fb (x:y:xs) rest =
159                        if possibleAfter (shiftToFront b2 x) y
160                        then Seq $ reverse $ x:(shiftToFront b2 x):y:xs
161                        else shift (shiftToFront b2 x) (y:xs) (x:rest)
162
163         mappend b1@(Flexible bs) b2@(Seq gi)  = mappend b2 b1
164         mappend b1@(Alt _) b2@(Flexible _)
165                 | possibleBefore b1 b2 = Seq [b1,b2]
```

```
166                 | possibleBefore b2 b1 = Seq [b2,b1]
167                 | otherwise = shiftToFront b1 b2
168
169         mappend b1@(Flexible _) b2@(Alt _) = mappend b2 b1
170
171         mappend b1@(Flexible _) b2@(Flexible _) = Seq [b1,b2]
172
173 -- ============================  Rest - * ============================ --
174         mappend b1@(Seq ss) b2@(Alt as)
175             | possibleBefore b1 b2 = Seq $ ss ++ [b2]
176             | possibleBefore b2 b1 = Seq $ b2 : ss
177             | otherwise =
178                  mconcat . (sortBy orderFunction) $ flatten b1 ++ flatten b2
179
180         mappend b2@(Alt bs) single@(Single b) = mappend single b2
181         mappend b1@(Seq as) single@(Single b)  = mappend single b1
182         mappend b1@(Alt as) b2@(Seq ss) = mappend b2 b1
183         mappend b1 b2 = mconcat . (sortBy orderFunction) $ flatten b1 ++ flatten b2
184
185 instance Blocks Errand where
186
187         temporalProjection (GI _ due dur _ _) = (fromUTC2Date startDate,due)
188                 where
189                      dueDate = fromDate2UTC due
190                      startDate = subtracting dur dueDate
191
192         placeProjection (GI _ due dur ps _) = ps
193
194         startPlace (GI _ due dur ps _) = ps
195
196         endPlace (GI _ due dur ps _) = ps
197
198         potential (GI _ due dur ps _) = [Full (fromUTC2Date startDate,due) (ps,ps)]
199                 where
200                      dueDate = fromDate2UTC due
201                      startDate = subtracting dur dueDate
202
203 instance Blocks Activity where
204
205         getRequirements (PTS _ _ _ r) = r
206         getRequirements (PTL _ _ _ r) = r
207
208         temporalProjection (PTS _ i _ _) = i
209         temporalProjection (PTL _ i _ _) = i
210
211         placeProjection (PTS _ _ p _) = [p]
212         placeProjection (PTL _ _ (p1,p2) _) = [p1,p2]
213
214         startPlace (PTS _ _ p _) = [p]
215         startPlace (PTL _ _ (p1,p2) _) = [p1]
216
217         endPlace (PTS _ _ p _) = [p]
218         endPlace (PTL _ _ (p1,p2) _) = [p2]
```

```
219
220         potential (PTS _ t p _) = [Full t ([p],[p])]
221         potential (PTL _ t (p1,p2) _) = [Full t ([p1],[p2])]
222
223 instance Blocks Potentials where
224
225         temporalProjection (Full i _) = i
226         temporalProjection (Half i _) = i
227
228         placeProjection p = nub . unsafePerformIO $ accessiblePlaces dbase p p
229
230 instance Blocks [Block] where
231
232         temporalProjection ls =
233               foldl boundInterval (head activityIntervals) activityIntervals
234                 where
235                 activityIntervals = map temporalProjection ls
236                 takeSmaller a b = if a <= b then a else b
237                 takeBigger a b = if a >= b then a else b
238                 boundInterval (a,b) (c,d) = (takeSmaller a c,takeBigger b d)
239
240         placeProjection ls = concat . (map placeProjection) $ ls
241
242         flatten bs = [bs]
243
244         getRequirements bs =
245               concat . map getR $
246               foldl propagation (mkPreCondS . head . reverse $ bs) (reverse bs)
247             where
248             getR (_,_,r) = r
249             propagation :: [(Date,Place,[Action])] -> Block -> [(Date,Place,[Action])]
250             propagation preCs b
251                  | null $ achievablePairs (makePostConds b) preCs = mkCondSet
252                    (fst $ temporalProjection b,startPlace b,getAllReqs preCs)
253                  | otherwise = mkPreCondS b
254             mkCondSet (t,ps,r) = [(t,p,r) | p <- ps]
255             getAllReqs preCs = map actionPropagation . filter noPulls . concat $
256                     [ r | (_,_,r) <- preCs]
257             makePostConds b = mkCondSet
258                     (snd $ temporalProjection b,endPlace b,getRequirements b)
259             actionPropagation (Push o) = Maint o
260             actionPropagation a = a
261             noPulls (Pull _) = False
262             noPulls _ = True
263             mkPreCondS :: Block -> [(Date,Place,[Action])]
264             mkPreCondS b = mkCondSet
265                   (fst $ temporalProjection b,startPlace b,getRequirements b)
266             achievablePairs ss1 ss2 =
267                   [(s1,s2) | s1 <- ss1 , s2 <- ss2, achievable dbase s1 s2]
268
269 instance Blocks Block where
270
271         getRequirements (Single a) = getRequirements a
```

```
272         getRequirements (Flexible a) = getRequirements a
273         getRequirements (Seq as) = getRequirements as
274         getRequirements (Alt as) = getRequirements as
275
276         temporalProjection (Single a) = temporalProjection a
277         temporalProjection (Flexible a) = temporalProjection a
278         temporalProjection (Seq as) = temporalProjection as
279         temporalProjection (Alt as) = temporalProjection as
280         temporalProjection (Flexible e) = temporalProjection e
281
282         placeProjection (Single a) = nub $ placeProjection a
283         placeProjection (Flexible a) = nub $ placeProjection a
284         placeProjection (Seq as) = nub . concat . (map placeProjection) . flatten $ as
285         placeProjection (Alt as) = nub . concat . (map placeProjection) . flatten $ as
286
287         startPlace (Single a) = startPlace a
288         startPlace (Flexible a) = startPlace a
289         startPlace (Seq as) = startPlace . head $ as
290         startPlace (Alt as) = concat $ map startPlace as
291
292         endPlace (Single a) = endPlace a
293         endPlace (Flexible a) = endPlace a
294         endPlace (Seq as) = endPlace . head . reverse $ as
295         endPlace (Alt as) = concat $ map endPlace as
296
297
298         possibleBefore a1 a2
299                 | (endTime a1) > (startTime a2) = False
300         --^^ temporally infeasible
301                 | (endTime a1) <= (startTime a2)
302                   && Container.contains (startPlace a2) (endPlace a1) = True
303         --^^from contained into containing granule
304                 | (endTime a1) <= (startTime a2)
305                   && Container.contains (endPlace a1) (startPlace a2) = True
306         --^^from building into room
307                 | otherwise =
308                   let a1Places = endPlace a1
309                       a2Places = startPlace a2
310                       a1Temp = endTime a1
311                       a2Temp = startTime a2
312                       placePairs = liftM2 (\x y -> (x,y)) a1Places a2Places
313                       minPlaceDist = minimum $
314                             map (\(x,y) -> euclideanDist x y) placePairs
315                       minDiPTLlacePair = head $
316                                 filter (\(x,y) ->
317                                        (euclideanDist x y) == minPlaceDist) placePairs
318                   in
319                   achievable dbase
320                         (a1Temp,fst minDiPTLlacePair :: Place)
321                              (a2Temp,snd minDiPTLlacePair :: Place)
322
323         possibleWithin a1 a2@(Seq xs)
324                 | possibleBefore a1 a2 || possibleBefore a2 a1 = False
```

```
325                 | possibleInside a1 (Seq $ sortBy orderFunction xs) = True
326                 | otherwise = False
327
328         possibleWithin a1 a2
329                 | possibleBefore a1 a2 || possibleBefore a2 a1 = False
330                 | possibleInside a1 a2 = True
331                 | otherwise = False
332
333         potential (Single (PTS _ _ _ _)) = []
334
335         potential (Single a@(PTL _ _ _ _)) = potential a
336
337         potential (Flexible a) = potential a
338
339         potential (Seq []) = []
340
341         potential (Seq (b:c:[])) =
342                  [Full (endTime b,startTime c) (endPlace b,startPlace c)]
343
344         potential (Seq (b:c:bs)) = (mkPotentials b c) : (potential $ Seq (c:bs))
345                 where
346                 mkPotentials b1 b2 =
347                      Full (endTime b1,startTime b2) (endPlace b1,startPlace b2)
348
349         potential b1@(Alt xs) = concat $ map (freeTimeFill altBound) xs
350                         where
351                         altBound = temporalProjection b1
352
353         blockConcat (Seq xs) = Seq $ concat $ map bCon xs
354                         where
355                         bCon (Seq xs) = xs
356                         bCon b = [b]
357         blockConcat (Alt xs) = Alt $ nub . concat $ map bCon xs
358                         where
359                         bCon (Alt xs) = xs
360                         bCon b = [b]
361         blockConcat x = id x
362
363
364         flatten (Alt bs) = concat $ map flatten bs
365         flatten (Seq bs) = concat $ map flatten bs
366         flatten b = [b]
367
368 instance TemporalRelations Interval where
369 -- Allens temporal relation algebra
370       relation ev1 ev2  | before = Before
371                         | after = After
372                         | during = During
373                         | equal = Equal
374                         | contains = Contains
375                         | meets = Meets
376                         | metBy = MetBy
377                         | overlaps = Overlaps
```

```
                  | overlapped = Overlapped
                  | starts = Starts
                  | startedBy = StartedBy
                  | finishes = Finishes
                  | otherwise = FinishedBy
                    where
                      end = snd
                      start = fst
                      before = (end ev1) < (start ev2)
                      after = (start ev1) > (end ev2)
                      during = (start ev1) > (start ev2) && (end ev1) < (end ev2)
                      equal = (start ev1) == (start ev2) && (end ev1) == (end ev2)
                      contains = (start ev1) < (start ev2) && (end ev1) > (end ev2)
                      overlaps = (start ev1) < (start ev2) && (end ev1) < (end ev2)
                      overlapped = (start ev1) > (start ev2) && (end ev1) > (end ev2)
                      meets = (end ev1) == (start ev2)
                      metBy = (start ev1) == (end ev2)
                      starts = (start ev1) == (start ev2) && (end ev1) < (end ev2)
                      startedBy = (start ev1) == (start ev2) && (end ev1) > (end ev2)
                      finishes = (start ev1) > (start ev2) && (end ev1) == (end ev2)

instance Planner Potentials (IO (CT.CTree Place)) where

    accessiblePlaces kb (Full (sd,ed) (sp,ep)) _ = concat <$> sequence
            [accessiblePlaces kb (sd,sp') (ed,ep') | sp' <- sp , ep' <- ep]

    accessiblePlaces kb (Half (sd,ed) p') _ =
           (takeWhile accessible) . nub . concat <$> sequence
                 [ closePlaces place dbase |place <- (map getParentPlace p') ]
           where
           getParentPlace p = unsafePerformIO $
                              (\(Just x)->x) <$> parentPlace p kb
           accessible p = (timeGo p) <= availableTime (sd,head p') (ed,head p')
           timeGo p = minimum $ map (estimatedTravelTime p) p'

instance Container.Container Interval where

      contains (sd,ed) (sd',ed')
            | sd' `isFinerThan` sd =
               Container.contains sd ed' &&
               Container.contains sd sd' ||
               Container.contains ed ed' && Container.contains ed sd'
            | otherwise =
               (fromDate2UTC sd) <= (fromDate2UTC sd') &&
               (fromDate2UTC ed) >= (fromDate2UTC ed')

instance Planner Block (IO (CT.CTree Place)) where

       achievable _ a1 a2 =
            let a1Places = endPlace a1
                a2Places = startPlace a2
                a1Temp = endTime a1
                a2Temp = startTime a2
```

```
                placePairs = liftM2 (\x y -> (x,y)) a1Places a2Places
                minPlaceDist = minimum $
                               map (\(x,y) -> euclideanDist x y) placePairs
                minDiPTLlacePair = head $ filter
                      (\(x,y) -> (euclideanDist x y) == minPlaceDist) placePairs
            in
            achievable dbase
              (a1Temp,fst minDiPTLlacePair :: Place)
                (a2Temp,snd minDiPTLlacePair :: Place)

        availableTime b1 b2 = (/60) .
                              fromRational .
                              toRational $
                              (absTimeDiff (startTime b2) (endTime b1)) :: Double

        requiredTime  b1 b2 =
              return . minimum $ do
                                   endPls <- endPlace b1
                                   startPls <- startPlace b2
                                   let tt =
                                   estimatedTravelTime endPls startPls
                                   return tt

-- ************************* Utilities ************************* --

orderFunction x y | elem (x' `relation` y') [Before,Meets,Overlaps] = LT
                  | elem (x' `relation` y') [Equal,Contains,Starts] = EQ
                  | otherwise = GT
                where
                x' = temporalProjection x
                y' = temporalProjection y


freeTimeFill :: Interval -> Block -> [Potentials]
freeTimeFill boundary b1
        | relation boundary tb1 == Contains =
      [mkHalfFreeTime (startBound,
                       (startTime b1)) startP,
                       mkHalfFreeTime (endTime b1,endBound) endP]
        | relation tb1 boundary == Starts =
      [mkHalfFreeTime (endTime b1,endBound) endP]
        | relation tb1 boundary == Finishes =
      [mkHalfFreeTime (startBound,startTime b1) startP]
        | otherwise = []
                where
                tb1 = temporalProjection b1
                startBound = fst boundary
                endBound = snd boundary
                startP = startPlace b1
                endP = endPlace b1

mkFreeTime time place = (Full time place)
mkHalfFreeTime time place = (Half time place)
```

```
484
485
486 potentiallyContained :: Block -> Block -> Bool
487 potentiallyContained b1 b2 = let p = potential b2
488                                  pPlaces = concat . (map placeProjection) $  p
489                                  pTime = map temporalProjection  p
490                                  activityPs = placeProjection b1
491                                  activityT = temporalProjection b1
492                                  spatialCont = Container.contains pPlaces activityPs
493                                  tempCont = or $
494                                  (liftM2 Container.contains pTime [activityT])
495                              in  spatialCont && tempCont
496
497 -- Granularity has to be considered here!
498 possibleInside a1 (Single _) = False
499 possibleInside a1 (Flexible _) = False
500 possibleInside a1 (Seq (x:[])) = False --or $ map (possibleInside a1 []) ls
501 possibleInside a1 (Seq (x:y:xs))
502                 | possibleBefore x a1 && possibleBefore a1 y = True
503                 | otherwise = possibleInside a1 (Seq (y:xs))
504
505
506 possibleInside a1 al@(Alt xs) = potentiallyContained a1 al
507
508 shiftToFront f@(Flexible (GI id due duration p req)) block = (Flexible (GI id newDue
    duration p req))
509         where
510         blockTime = startTime block
511         requTime = round . unsafePerformIO $ (requiredTime f block)
512         newDue = fromUTC2Date
513                 . (subtracting requTime)
514                 . fromDate2UTC
515                 $ blockTime
```

Listing 12: Intention Module

```
1  module BackendModules.Intentions where
2
3  import Prelude
4  import Control.Applicative ((<$>))
5  import Data.List.Split (splitOn)
6  import Data.List
7  import Data.Tree
8  import Data.Monoid (mappend,mconcat)
9  import Control.Monad.State.Lazy (get, put,evalState)
10 import BackendModules.Container
11 import BackendModules.SemanticContainment
12 import BackendModules.ActivityBlocks
13 import qualified BackendModules.ContainmentTree as CT
14
15 -- ************************ Type Declarations ************************ --
16
```

```
17  type Identifier = String
18  type Group = [Intention]
19  type IntentionStructure = (Block,Group)
20
21  class IntentionStructures intention where
22
23          insertGroupInto :: intention -> Identifier -> intention -> intention
24          groupInto :: [intention] -> Identifier -> intention
25          addTo :: intention -> intention -> intention
26          returnIntentionById :: Identifier -> intention -> intention
27          returnIntentionByName :: String -> intention -> intention
28          returnIntentionByName n i = returnIntentionById (head $ name2ID n i) i
29          name2ID :: String -> intention -> [String]
30
31  -- ****************************** Instances ****************************** --
32
33  instance IntentionStructures (Block,[Intention]) where
34
35          insertGroupInto (b1,p1) id (b2,p2) =
36           (mappend b1 b2,nub . concat $
37              [insertIntention p (Intention id) p2 | p <- p1] )
38
39          groupInto structs groupName =
40            evalState (insertSeveral structs groupName)
41               ((Seq [],[Intention "Empty"]),(Seq [],[Intention groupName]))
42
43          addTo (b1,p1) (b2,p2) = (mappend b1 b2,mappend p1 p2)
44
45          returnIntentionById id (b,p) = (mconcat relevantBlocks,[Intention id])
46                where
47                pBranches = getParentBranches p id
48                atomIDs = getLeafs pBranches
49                bottomBlocks = BackendModules.ActivityBlocks.flatten b
50                relevantBlocks = [ bb | bb <- bottomBlocks,
51                                        ads <- atomIDs,
52                                        activityName' bb == ads]
53                activityName' (Single act) = case act of
54                           (PTS id _ _ _) -> id
55                                 (PTL id _ _ _) -> id
56                activityName' (Flexible (GI id _ _ _ _)) = id
57
58          name2ID n (_,is) = filter
59                 (\branch -> n == (last . splitBranch $ branch) ) branches
60             where
61                 branches = map name is
62                 splitBranch b = splitOn "." b
63
64  instance Blocks IntentionStructure where
65
66          startTime (b,_) = startTime b
67          endTime (b,_) = endTime b
68          startPlace (b,_) = startPlace b
69          endPlace (b,_) = endPlace b
```

```
70          temporalProjection (b,_) = temporalProjection b
71          placeProjection (b,_) = placeProjection b
72          potential (b,_) = potential b
73          possibleBefore (b,_) (b',_) = possibleBefore b b'
74          possibleWithin (b,_) (b',_) = possibleWithin b b'
75
76  -- ************************ Utilities ************************* --
77
78  getParentBranches :: [Intention] -> [Char] -> [Intention]
79  getParentBranches intBranches id =
80        [iB | iB <- intBranches , contains (Intention id) iB ]
81
82  getLeafs :: [Intention] -> [ID]
83  getLeafs parentBranches = map (last . (splitOn ".") . name) parentBranches
84
85  insertSeveral [] _ = do
86                  (_,struct) <- get
87                  return struct
88
89  insertSeveral (x:xs) s = do
90          (currentThing,structure) <- get
91          put (x,insertGroupInto x s structure)
92          insertSeveral xs s
```

Listing 13: SemanticContainment Module

```
1   module BackendModules.SemanticContainment where
2
3   import Prelude
4   import Data.List.Split
5   import Data.List
6   import Data.Tree
7   import Control.Applicative ((<$>))
8   import qualified BackendModules.ContainmentTree as CT
9   import BackendModules.Container
10
11  -- ************************ Type Declarations ************************* --
12
13  newtype Intention = Intention { name :: String } deriving (Eq,Show,Read)
14
15
16
17
18  class (Container intention, Eq intention) => Groups intention where
19
20          type Hierarchy :: *
21          insertIntention :: intention -> intention -> Hierarchy -> Hierarchy
22          getSuperIntention :: intention -> Maybe intention
23          getSubIntentions :: intention -> Hierarchy -> [intention]
24          level :: intention -> Int
25
26  -- ***************************** Instances ***************************** --
```

```
27
28 instance Container Intention where
29
30         contains i1 i2 = isPrefixOf (name i1) (name i2)
31
32 instance Groups Intention where
33
34         type Hierarchy = [Intention]
35         insertIntention i1 i2 p = (Intention $ (name i2) ++ "." ++ (name i1)) : (p \\ [
       i1])
36         getSuperIntention i1 | (length splitted) > 1 = Just $ Intention . (intercalate
       ".") $ splitted \\ [i1Name]
37                    | otherwise = Nothing
38                                  where
39           splitted = splitOn "." (name i1)
40              i1Name = last . splitOn "." $ name i1
41
42         getSubIntentions i1 p = children
43                                  where
44                                  containers = filter (\x -> contains i1 x || contains x i1) p
45           immediateSubNodes = filter (\x -> (level x - level i1) == 1) containers
46                                  children = immediateSubNodes{--do
47                                          branch <- immediateSubNodes
48                                          let branchPart = takeWhile (/= name i1) $ reverse $
       splitOn "." (name branch)
49                                          return . Intention . intercalate "." . reverse $
       branchPart --}
50         level = length . (splitOn ".") . name
```

Listing 14: PIS Module

```
1  module BackendModules.Pis where
2
3  import Prelude
4  import Data.List
5  import Data.Monoid
6  import GHC.Generics
7  import Data.Aeson
8  import BackendModules.TimeHandle
9  import BackendModules.Agent
10 import BackendModules.Place
11 import BackendModules.Intentions
12 import BackendModules.ActivityBlocks
13
14 -- ************************* Type Declarations ************************* --
15
16 data PiType = Pic | Doc
17              deriving (Eq,Show,Generic,Read)
18
19 data PI = PI { uri :: String,
20                creationTime :: Date,
21                piType :: PiType }
```

```
22                deriving (Eq,Show)
23
24  class PIS pis where
25
26     getPIitemByActivity :: Identifier -> IntentionStructure -> pis -> pis
27     getPIitemByPlace :: Place -> Agent -> pis -> pis
28
29  -- ***************************** Instances ***************************** --
30
31  instance PIS [PI] where
32
33    getPIitemByActivity i block pis =
34          filter (\x -> (creationTime x) `within` activityBound) pis
35               where
36             intention
37                 | mempty /= (fst $ returnIntentionById i block)  =
38                                                 returnIntentionById i block
39                 | otherwise = returnIntentionByName i block activityBound =
40                   (startTime . fst $ intention, endTime . fst $ intention)
41             within c (a,b) = c <= b && c >= a
```

Listing 15: ContainmentTree Module

```
1   module BackendModules.ContainmentTree where
2
3   import Data.List
4   import Data.Tree
5   import Control.Applicative
6   import Prelude
7
8   type CTree a = Tree a
9
10  depth  :: CTree a -> Int
11  depth (Node _ []) = 1
12  depth (Node _ succs) = 1 + maximum (map depth succs)
13
14  printTree :: (a -> String) -> Tree a -> IO ()
15  printTree f = putStrLn . drawTree . (f <$>)
16
17  returnElem :: Eq a => (a -> a -> Bool) -> a -> Tree a -> Maybe (Tree a)
18  returnElem f el tree@(Node place subTrees)
19          | el == place = Just tree
20          | otherwise =
21            let container = find (\x -> contains f x (Node el [])) subTrees
22                in case container of
23                   Nothing -> Nothing
24                   (Just c) -> if (node2Elem c) /= el
25                               then returnElem f el c
26                               else Just c
27
28  getChildren :: Eq a => (a -> a -> Bool) -> Tree a -> a -> Forest a
29  getChildren f tree place = let element = returnElem f place tree in
```

```
                                                case element of
                                                     Just (Node _ el) -> el
                                                     Nothing -> []

treeInsert :: (Eq a) => (a -> a -> Bool) -> CTree a -> CTree a -> CTree a
treeInsert f tree@(Node p subTrees) placeEntry@(Node pe pl)
                  | contains f tree placeEntry =
                      let container = find (\x -> contains f x placeEntry) subTrees
                          restTrees c = filter (\x -> x /= c) subTrees
                          containedTrees =
                                 filter (\x -> contains f placeEntry x) subTrees
                                   in case container of
                                      Nothing -> if containedTrees == []
                                                 then Node p (placeEntry:subTrees)
                                                 else
                                                 Node p
                                                 ((Node pe (pl ++ containedTrees)):
                                                 ((\\) subTrees containedTrees))
                                      (Just c) -> Node p
                                         ((treeInsert f c placeEntry):
                                         (restTrees c))
                   | contains f placeEntry tree = treeInsert f placeEntry tree
                   | otherwise = treeInsert f tree placeEntry

getParent :: (Eq a) => (a -> a -> Bool) -> CTree a -> a -> Maybe (CTree a)
getParent f tree@(Node p subTrees) place = travers tree place []
            where
            travers tree@(Node p subTrees) place pNodes
                | p == place = if null pNodes
                               then Nothing
                               else Just $
                               head . (filter (\(Node p _) -> p /= place)) $ pNodes
                | contains f tree (Node place []) =
                let container =
                    find (\x -> contains f x (Node place []))  subTrees
                    in case container of
                       Nothing -> Just tree
                       (Just c@(Node cp _)) -> travers c place (c:tree:pNodes)

preorderWith :: (t -> Bool) -> Tree t -> [t]
preorderWith f (Node p []) | f p = [p]
                           | otherwise = []

preorderWith f (Node p subTrees@(x:xs))
                | f p = p : (concat (map (preorderWith f) subTrees) )
                | otherwise = (concat (map (preorderWith f) subTrees) )

contains :: (a -> a -> Bool) -> CTree a -> CTree a -> Bool
contains f (Node p1 _) (Node p2 _) = p1 `f` p2

node2Elem (Node p _) = p
```

Listing 16: Usecases Module

```
module BackendModules.Usecases where

import BackendModules.ActivityBlocks
import BackendModules.Intentions
import BackendModules.SemanticContainment
import BackendModules.Place
import BackendModules.Planner
import BackendModules.Geography
import BackendModules.TimeHandle
import BackendModules.Equipment
import BackendModules.Pis
import BackendModules.Agent
import Data.Monoid
import Data.List (sortBy)
import System.IO.Unsafe
import Control.Applicative
import Prelude

-- ************************************************************************
-- *                                                                      *
-- *                              Scenario A                             *
-- *                                                                      *
-- ************************************************************************

-- ======================= Pure Block Structure ============================== --

flight = Single
         (PTL "Vienna-Brussel"
         (fromString2Date "2013-05-14-18-20",fromString2Date "2013-05-14-20-05")
          (vienna,brussel) [])

bookReturn = Flexible
             (GI "Book return"
             (fromString2Date "2013-05-16-19-00") 10 [tuwien] [])

buySouvenir = Flexible
              (GI "Souvenir"
              (fromString2Date "2013-05-18-00-00") 10 leuvenShops [])

leuvenShops = unsafePerformIO $ containingPlaces dbase leuven

keynote = Single
          (PTS "keynote"
          (fromString2Date "2013-05-15-09-00",fromString2Date "2013-05-15-10-00")
          agileRoomA [])

session1 = Single
           (PTS "session1"
           (fromString2Date "2013-05-15-10-30",fromString2Date "2013-05-15-12-00")
            agileRoomA [])

```

```
52  session2 = Single
53             (PTS "session2"
54             (fromString2Date "2013-05-15-10-30",fromString2Date "2013-05-15-12-00")
55              agileRoomB [])
56
57  session3 = Single
58             (PTS "session3"
59             (fromString2Date "2013-05-15-10-30",fromString2Date "2013-05-15-12-00")
60              agileRoomC [])
61
62  mS = mconcat [session1,session2,session3]
63
64  session4 = Single
65             (PTS "session4"
66             (fromString2Date "2013-05-15-13-00",fromString2Date "2013-05-15-15-00")
67              agileRoomA [])
68
69  session5 = Single
70             (PTS "session5"
71             (fromString2Date "2013-05-15-13-00",fromString2Date "2013-05-15-15-00")
72              agileRoomB [])
73
74  session6 = Single
75             (PTS "session6"
76             (fromString2Date "2013-05-15-13-00",fromString2Date "2013-05-15-15-00")
77              agileRoomC [])
78
79  as = mconcat [session4,session5,session6]
80
81  session7 = Single
82             (PTS "session7"
83             (fromString2Date "2013-05-15-15-30",fromString2Date "2013-05-15-17-30")
84              agileRoomA [])
85
86  session8 = Single
87             (PTS "session8"
88             (fromString2Date "2013-05-15-15-30",fromString2Date "2013-05-15-17-30")
89              agileRoomB [])
90
91  session9 = Single
92             (PTS "session9"
93             (fromString2Date "2013-05-15-15-30",fromString2Date "2013-05-15-17-30")
94              agileRoomC [])
95
96  keynote2 = Single
97             (PTS "keynote2"
98             (fromString2Date "2013-05-16-09-00",fromString2Date "2013-05-16-10-00")
99              agileRoomA [])
100
101 session10 = Single
102              (PTS "session10"
103              (fromString2Date "2013-05-16-10-30",fromString2Date "2013-05-16-12-00")
104               agileRoomA [])
```

```
105
106 session11 = Single
107             (PTS "session11"
108             (fromString2Date "2013-05-16-10-30",fromString2Date "2013-05-16-12-00")
109              agileRoomB [])
110
111 session12 = Single
112             (PTS "session12"
113             (fromString2Date "2013-05-16-10-30",fromString2Date "2013-05-16-12-00")
114              agileRoomC [])
115
116 agileMeeting = Single
117                (PTS "agileMeeting"
118                (fromString2Date "2013-05-16-13-00",fromString2Date "2013-05-16-14-30")
119                 agileRoomA [])
120
121 posterSession = Single
122                 (PTS "posterSession"
123                 (fromString2Date "2013-05-16-14-30",fromString2Date "2013-05-16-15-30")
124                  conferenceSite [])
125
126 session13 = Single
127             (PTS "session13"
128             (fromString2Date "2013-05-16-15-30",fromString2Date "2013-05-16-17-30")
129              agileRoomA [])
130
131 session14 = Single
132             (PTS "session14"
133             (fromString2Date "2013-05-16-15-30",fromString2Date "2013-05-16-17-30")
134              agileRoomB [])
135
136 session15 = Single
137             (PTS "session15"
138             (fromString2Date "2013-05-16-15-30",fromString2Date "2013-05-16-17-30")
139              agileRoomC [])
140
141 busride = Single
142           (PTL "busride"
143           (fromString2Date "2013-05-16-17-45",fromString2Date "2013-05-16-19-00")
144           (conferenceSite,galaDinnerSite) [])
145
146 dinner = Single
147          (PTS "dinner"
148          (fromString2Date "2013-05-16-19-00",fromString2Date "2013-05-16-23-00")
149           galaDinnerSite [])
150
151 keynote3 = Single
152            (PTS "keynote3"
153            (fromString2Date "2013-05-17-09-00",fromString2Date "2013-05-17-10-30")
154             agileRoomA [])
155
156 bestPaper = Single
157             (PTS "bePTLaper"
```

```
158             (fromString2Date "2013-05-17-11-00",fromString2Date "2013-05-17-12-30")
159              agileRoomA [])
160
161 closingSession = Single
162                  (PTS "closingSession"
163                  (fromString2Date "2013-05-17-12-30",fromString2Date "2013-05-17-13-30"
    )
164                   agileRoomA [])
165
166 flightBack = Single
167              (PTL "Brussel-Vienna"
168              (fromString2Date "2013-05-17-20-00",fromString2Date "2013-05-17-21-45")
169              (brussel,vienna) [])
170
171 day1 = mconcat
172      [keynote,session1,
173       session2,session3,
174       session4,session5,
175       session6,session7,
176       session8,session9]
177
178 day2 = mconcat
179       [session10,session11,
180        session12,agileMeeting,
181        posterSession,session13,
182        session14,session15,
183        keynote2,busride,dinner]
184
185 day3 = mconcat [keynote3,bePTLaper,closingSession]
186
187 agile = mconcat [day1,day2,day3]
188
189 -- ******************** Conceptual Block Structure ******************** --
190
191
192 agileS = groupInto [day1S,day2S,day3S] "Agile"
193
194 keynoteS = (keynote,[Intention "keynote"])
195
196 session1S = (session1,[Intention "session1"])
197 session2S = (session2,[Intention "session2"])
198 session3S = (session3,[Intention "session3"])
199
200 session4S = (session4,[Intention "session4"])
201 session5S = (session5,[Intention "session5"])
202 session6S = (session6,[Intention "session6"])
203
204 session7S = (session7,[Intention "session7"])
205 session8S = (session8,[Intention "session8"])
206 session9S = (session9,[Intention "session9"])
207
208 morningSession = groupInto [session1S,session2S,session3S] "MorningSession"
209 afternoonSession = groupInto [session4S,session5S,session6S] "AfternoonSession"
```

```
210 eveningSession = groupInto [session7S,session8S,session9S] "EveningSession"
211 day1S = groupInto [keynoteS,morningSession,afternoonSession,eveningSession] "Day1"
212
213 keynote2S = (keynote2,[Intention "keynote2"])
214 session10S = (session10,[Intention "session10"])
215 session11S = (session11,[Intention "session11"])
216 session12S = (session12,[Intention "session12"])
217
218 agileMeetingS = (agileMeeting,[Intention "agileMeeting"])
219 posterSessionS = (posterSession,[Intention "posterSession"])
220
221 session13S = (session13,[Intention "session13"])
222 session14S = (session14,[Intention "session14"])
223 session15S = (session15,[Intention "session15"])
224
225 busrideS = (busride,[Intention "busride"])
226 dinnerS = (dinner,[Intention "dinner"])
227
228 morningSessionD2 = groupInto
229                    [session10S,session11S,session12S] "MorningSessionD2"
230
231 afternoonSessionD2 = groupInto
232                    [session13S,session14S,session15S] "AfternoonSessionD2"
233
234 day2S = groupInto
235       [keynote2S,morningSessionD2,
236        afternoonSessionD2,busrideS,dinnerS] "Day2"
237
238 keynote3S = (keynote3,[Intention "keynote3"])
239 bestPaperS = (bestPaper,[Intention "bePTLaper"])
240 closingSessionS = (closingSession,[Intention "closingSession"])
241
242 day3S = groupInto [keynote3S,bestPaperS,closingSessionS] "Day3"
243
244 flightThereS = (flight,[Intention "Vienna-Brussel"])
245
246 flightBackS = (flightBack,[Intention "Brussel-Vienna"])
247
248 flightS = groupInto [flightThereS,flightBackS] "Flight"
249
250 bookReturnS = (bookReturn,[Intention "Book return"])
251 buySouvenirS = (buySouvenir,[Intention "Souvenir"])
252
253 agileTrip = groupInto [flightS,agileS] "AgileTrip"
254
255 allS = groupInto [agileTrip,bookReturnS,buySouvenirS] "All"
256
257 -- ************************************************************************
258 -- *                                                                      *
259 -- *                              Scenario B                              *
260 -- *                                                                      *
261 -- ************************************************************************
262
```

```
263 -- ====================== Pure Block Structure ============================== --
264
265 viennaShops = unsafePerformIO $ containingPlaces dbase tuwien
266
267 morningLecture = Single
268                  (PTS "Lecture 1"
269                  (fromString2Date "2013-05-15-10-00",fromString2Date "2013-05-15-12-00")
270                   seminarRoom [])
271
272 middleLecture = Single
273                  (PTS "Lecture 2"
274                  (fromString2Date "2013-05-15-13-30",fromString2Date "2013-05-15-14-30")
275                   seminarRoom [])
276
277 eveningLecture = Single
278                   (PTS "Lecture 3"
279                   (fromString2Date "2013-05-15-18-00",fromString2Date "2013-05-15-20-00"
    )
280                    seminarRoom [Maint laptop])
281
282 meetFriend = Flexible
283              (GI "ReturnDVD" (fromString2Date "2013-05-15-12-30")
284              10 [tuwien] [Push dvd])
285
286 buyGroceries = Flexible
287                (GI "Buy Groceries" (fromString2Date "2013-05-15-20-00")
288                20 viennaShops [])
289
290 scenarioB = mconcat [morningLecture,
291                      middleLecture,
292                      eveningLecture,meetFriend]
293
294 -- ******************** Conceptual Block Structure ******************** --
295
296 morningLectureS = (morningLecture,[Intention "Lecture 1"])
297
298 middleLectureS = (middleLecture,[Intention "Lecture 2"])
299
300 eveningLectureS = (eveningLecture,[Intention "Lecture 3"])
301
302 meetFriendS = (meetFriend,[Intention "ReturnDVD"])
303
304 buyGroceriesS = (buyGroceries,[Intention "Buy Groceries"])
305
306 lectures = groupInto [morningLectureS,middleLectureS,eveningLectureS] "Lectures"
307 private = groupInto [meetFriendS,buyGroceriesS] "Private"
308
309 scenarioBDay = groupInto [lectures,private] "ScenarioBDay"
310
311 -- ****************************************************************************
312 -- *                                                                          *
313 -- *                              Functionalities                                 *
314 -- *                                                                          *
```

```
315 -- ****************************************************************************
316
317 findGaps :: Block -> SpatialGranules -> [Potentials]
318 findGaps b g = filter (\p -> isGap p) . potential $ b
319       where
320    isGap (Full _ (sps,eps)) = (>1) . length . nub $ (sps ++ eps)
321         isGap _ = False
322
323
324 containingPlace :: Block -> Place
325 containingPlace b = commonParent . nub $ placeProjection b
326
327 commonParent :: [Place] -> Place
328 commonParent (p:ps) =  foldl
329                        (\granule nplace ->
330                              commonUpperGranule granule nplace ) p ps
```

References

Abdalla, A. (2012). Latyourlife: A geo-temporal task planning application. In *Advances in Location-Based Services*, Lecture Notes in Geoinformation and Cartography, pages 305–325. Springer Berlin Heidelberg. 17, 35, 38, 51

Abdalla, A. and Frank, A. U. (2011). Personal geographic information management. In *Proceedings of the Workshop on Cognitive Engineering for Mobile GIS*, Belfast, USA,. CEUR Workshop Proceedings. viii, 5, 17

Abdalla, A. and Frank, A. U. (2012). Combining trip and task planning: How to get from a to passport. In *Geographic Information Science*, pages 1–14. Springer. 79

Abdalla, A. and Frank, A. U. (2014). Designing spatio-temporal pim tools for prospective memory support. In *Principle and Application Progress in Location-Based Services*, pages 227–242. Springer. x, 38, 42, 52, 53, 111

Abdalla, A., Weiser, P., and Frank, A. U. (2013). Design principles for spatio-temporally enabled pim tools: A qualitative analysis of trip planning. In *Geographic Information Science at the Heart of Europe*, pages 323–336. Springer. viii, 22, 27, 28, 29

Alexander, C. (1979). *The timeless way of building*, volume 1. Oxford University Press. 12, 35

Barreau, D. and Nardi, B. A. (1995). Finding and reminding: file organization from the desktop. *ACM SIGCHI Bulletin*, 27(3):39–43. 7, 9, 10, 53

Bell, C. G., Gemmell, J., and Haag, J. (2009). *Total recall: How the e-memory revolution will change everything*. Dutton New York, NY. 10

Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G., and Ducheneaut, N. (2004). What a to-do: studies of task management towards the design of a personal task list manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 735–742. ACM. 32

Bettini, C., Dyreson, C. E., Evans, W. S., Snodgrass, R. T., and Wang, X. S. (1998). A glossary of time granularity concepts. In *Temporal databases: research and practice*, pages 406–413. Springer. 61

Bettini, C., Jajodia, S., and Wang, S. (2000). *Time granularities in databases, data mining, and temporal reasoning*. Springer. x, 47, 48, 61, 62, 65, 96

Bittner, T. and Smith, B. (2001). A taxonomy of granular partitions. In *Spatial Information Theory*, pages 28–43. Springer. 59

Boardman, R. and Sasse, M. A. (2004). Stuff goes into the computer and doesn't come out: a cross-tool study of personal information management. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 583–590. ACM. 11

Bruce, H., Jones, W., and Dumais, S. (2004). Keeping and re-finding information on the web: What do people do and what do they need? *Proceedings of the American Society for Information Science and Technology*, 41(1):129–137. 7

Buckland, M. and Lancaster, L. (2004). Combining place, time, and topic. *D-Lib Magazine*, 10(5):1082–9873. 55

Butler, D. (2006). Virtual globes: The web-wide world. *Nature*, 439(7078):776–778. 2

Câmara, L., Guerreiro, T., Gonçalves, D., and Jorge, J. A. (2008). Realfind: managing personal items in the physical world. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, pages 3321–3326. ACM. 37

Capra, R. G. and Pérez-Quiñones, M. A. (2003). Re-finding found things: An exploratory study of how users re-find information. *arXiv preprint cs/0310011*. 7

Capra III, R. G. (2006). *An investigation of finding and refinding information on the web*. PhD thesis, Virginia Polytechnic Institute and State University. 7

Capra III, R. G., Manuel, A., et al. (2005). Using web search engines to find and refind information. *Computer*, (10):36–42. 7

Catarci, T., Dix, A., Katifori, A., Lepouras, G., and Poggi, A. (2007). Task-centred information management. In *Digital Libraries: Research and Development*, pages 197–206. Springer. 11, 30

Catarci, T., Habegger, B., Poggi, A., Dix, A., Ioannidis, Y., Katifori, A., and Lepouras, G. (2006). Intelligent user task oriented systems. In *Proceedings of the Second SIGIR Workshop on Personal Information Management (PIM)*. 11, 115

Chase, W. G. and Simon, H. A. (1973). Perception in chess. *Cognitive psychology*, 4(1):55–81. 8

Chrisman, N. R. (1982). *Methods of spatial analysis based on error in categorical maps*. PhD thesis, University of Bristol. 64

Coleman, D. J., Georgiadou, Y., Labonte, J., et al. (2009). Volunteered geographic information: The nature and motivation of produsers. *International Journal of Spatial Data Infrastructures Research*, 4(1):332–358. 2

Comer, D. (1979). Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137. 58

Coppock, J. and Rhind, D. (1991). *The history of GIS, geographic information system*, volume 39. 1

Couclelis, H. and Gale, N. (1986). Space and spaces. *Geografiska Annaler. Series B. Human Geography*, pages 1–12. 64, 69

Crabtree, A., Hemmings, T., Rodden, T., and Mariani, J. (2003). Informing the development of calendar systems for domestic use. In *ECSCW 2003*, pages 119–138. Springer. 31

De Lisi, R. (1987). A cognitive-developmental model of planning. *Blueprints for thinking: The role of planning in cognitive development. Friedman, Sarah L., Ellin Kofsky Scholnick, and Rodney R. Cocking, eds.*, pages 79–109. 24

Dix, A., Katifori, A., Poggi, A., Catarci, T., Ioannidis, Y., Lepouras, G., and Mora, M. (2007). From information to interaction: in pursuit of task-centred information management. In *Proceedings of DELOS Conference*. 11

Downs, R. M., Stea, D., et al. (1977). *Maps in minds: Reflections on cognitive mapping*. Harper & Row New York. 15

Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., and Robbins, D. C. (2003). Stuff i've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 72–79. ACM. 10

Egenhofer, M. and Franzosa, R. (1991). Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174. 53

Espeter, M. and Raubal, M. (2009). Location-based decision support for user groups. *Journal of Location Based Services*, 3(3):165–187. 49

Fertig, S., Freeman, E., and Gelernter, D. (1996). "finding and reminding" reconsidered. *ACM SIGCHI Bulletin*, 28(1):66–69. 9

Frank, A. U. (1996). Hierarchical spatial reasoning. Technical report, Internal Report: Dept. of Geoinformation, Technical University Vienna. 64

Frank, A. U. (1997). Spatial ontology: A geographical information point of view. In *Spatial and temporal reasoning*, pages 135–153. Springer. 58

Frank, A. U. and Kuhn, W. (1999). A specification language for interoperable gis. In *Interoperating Geographic Information Systems*, pages 123–132. Springer. 94

Frank, A. U., Volta, G. S., and Mcgranaghan, M. (1997). Formalization of families of categorical coverages. *International Journal of Geographical Information Science*, 11(3):215–231. 64, 65

Freeman, E. and Gelernter, D. (1996). Lifestreams: A storage model for personal data. *ACM SIGMOD Record*, 25(1):80–86. 9

Freundschuh, S. M. and Egenhofer, M. J. (1997). Human conceptions of spaces: implications for gis. *Transactions in GIS*, 2(4):361–375. 64, 69

Gemmell, J., Bell, G., and Lueder, R. (2006). Mylifebits: a personal database for everything. *Communications of the ACM*, 49(1):88–95. 9

Gemmell, J., Bell, G., Lueder, R., Drucker, S., and Wong, C. (2002). Mylifebits: fulfilling the memex vision. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238. ACM. 9

Gibson, J. J. (1979). The ecological approach to visual perception. 13

Gill, A. (1976). *Applied Algebra for the Computer Scientist*. Prentice-Hall, Englewood Cliffs, New Jersey. 6, 60

Golledge, R. and Gärling, T. (2001). Spatial behavior in transportation modeling and planning. 15, 16

Golledge, R. G. (1997). *Spatial behavior: A geographic perspective*. Guilford Press. 15

Gollwitzer, P. M. (1993). Goal achievement: The role of intentions. *European review of social psychology*, 4(1):141–185. 22, 23, 81

Goodchild, M. (2009). Neogeography and the nature of geographic expertise. *Journal of Location Based Services*, 3(2):82–96. 2

Goodchild, M. F. (2014). Twenty years of progress: Giscience in 2010. *Journal of Spatial Information Science*, (1):3–20. 37

Gould, P. and White, R. (2012). *Mental maps*. Routledge. 16

Graf, P. and Uttl, B. (2001). Prospective memory: A new focus for research. *Consciousness and Cognition*, 10(4):437–450. 10, 18, 26

Gregory, D., Johnston, R., Pratt, G., Watts, M., and Whatmore, S. (2009). *The dictionary of human geography*. Wiley-Blackwell. 17

Hagerstrand, T. (1970). What about people in regional science? *Papers of the Regional Science Association*, 24:7–21. 17

Hayes-Roth, B. and Hayes-Roth, F. (1979). A cognitive model of planning*. *Cognitive science*, 3(4):275–310. 25, 29

Hobbs, J. R. (1985). Granularity. In *In Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Citeseer. 58, 59

Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., Smyth, G., Kapur, N., and Wood, K. (2006). Sensecam: A retrospective memory aid. *UbiComp 2006: Ubiquitous Computing*, pages 177–193. 9

Hornsby, K. S. and Cole, S. (2007). Modeling moving geospatial objects from an event-based perspective. *Transactions in GIS*, 11(4):555–573. 48, 73

Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., and Kolas, D. (2013). A geo-ontology design pattern for semantic trajectories. In *Spatial Information Theory*, pages 438–456. Springer. 42

Hutchinson, H., Bederson, B. B., Plaisant, C., and Druin, A. (2003). Family calendar survey. 31

Jakle, J. A., Brunn, S. D., Roseman, C. C., and Press, D. (1976). *Human spatial behavior: A social geography*, volume 298. Duxbury Press North Scituate. p93: activity spaces are an importat manifestation of our every day lifes, and, in addition,represent an importatn process through which we gain information about and attach meaning to our environment". 15, 16

Janowicz, K. (2010). The role of space and time for knowledge organization on the semantic web. *Semantic Web*, 1(1):25–32. 11, 18, 35

Jones, S. L. P. (2003). *Haskell 98 language and libraries: the revised report*. Cambridge University Press. 94

Jones, W. (2004). Finders, keepers? the present and future perfect in support of personal information management. *First monday*, 9(3). 7, 35

Jones, W. and Teevan, J. (2007). *Personal Information Management*. University of Washington Press. 3, 5, 6, 7, 53

Jordan, T., Raubal, M., Gartrell, B., and Egenhofer, M. (1998). An affordance-based model of place in gis. In *8th Int. Symposium on Spatial Data Handling, SDH*, volume 98, pages 98–109. 12, 13

Kalnikaite, V., Sellen, A., Whittaker, S., and Kirk, D. (2010). Now let me see where i was: understanding how lifelogs mediate memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2045–2054. ACM. 10

Katifori, A., Vassilakis, C., Daradimos, I., Lepouras, G., Ioannidis, Y., Dix, A., Poggi, A., and Catarci, T. (2008). Personal ontology creation and visualization for a personal interaction management system. In *Proceedings of PIM Workshop, CHI*, page 15. 11, 30

Kolars, J. F., Nystuen, J. D., and Bell, D. (1975). *Physical geography: environment and man*. McGraw-Hill New York. 64, 69

Kreitler, S. and Kreitler, H. (1987a). Conceptions and processes of planning: The developmental perspective. *Blueprints for thinking: The role of planning in cognitive development*, pages 205–272. 24, 25

Kreitler, S. and Kreitler, H. (1987b). Plans and planning: Their motivational and cognitive antecedents. *Blueprints for thinking: The role of planning in cognitive development*, pages 110–178. 22, 24

Kuhn, W. (2009). A functional ontology of observation and measurement. In *GeoSpatial Semantics*, pages 26–43. Springer. 94

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, 26(12):2267–2276. 58

Lakoff, G. and Johnson, M. (2008). *Metaphors we live by*. University of Chicago press. 64

Lansdale, M. W. (1988). The psychology of personal information management. *Applied ergonomics*, 19(1):55–66. 8

Lepouras, G., Dix, A., Katifori, A., Catarci, T., Habegger, B., Poggi, A., and Ioannidis, Y. (2006). Ontopim: From personal information management to task information management. *Personal Information Management: Now That We are Talking, What Are We Learning?*, page 78. 11

Longley, P., Goodchild, M., Maguire, D., and Rhind, D. (2001). *Geographic Information Systems and Science*. Wiley. 17

Lynch, K. (1960). *The image of the city*, volume 11. MIT press. 12

Malone, T. W. (1983). How do people organize their desks?: Implications for the design of office information systems. *ACM Transactions on Information Systems (TOIS)*, 1(1):99–112. 7, 8, 9, 10, 21

Mark, D., Chrisman, N., Frank, A., McHaffie, P., and Pickles, J. (1997). The gis history project. 1

Mayer, R. (1990). Problem solving. *The Blackwell Dictionary of Cogntive Psychology, Oxford, M.W. Eysneck (Ed.), UK:Blackwell.* 25

McDaniel, M. A. and Einstein, G. O. (2000). Strategic and automatic processes in prospective memory retrieval: A multiprocess framework. *Applied cognitive psychology*, 14(7):S127–S144. viii, 26, 27

McDaniel, M. A. and Einstein, G. O. (2011). The neuropsychology of prospective memory in normal aging: A componential approach. *Neuropsychologia*, 49(8):2147–2155. 26

Miller, G. A., Galanter, E., and Pribram, K. H. (1986). *Plans and the structure of behavior.* Adams Bannister Cox. 25

Miller, H. J. (1991). Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information System*, 5(3):287–301. 17

Montello, D. R. (1993). Scale and multiple psychologies of space. In *Spatial information theory a theoretical basis for gis*, pages 312–321. Springer. 44, 64, 67, 69

Norman, D. A. (1993). Things that make us smart. 31

Parsons, T. (1964). *The Social System.(1951).* 15

Piaget, J. (1960). *Psychology of Intelligence*. Totowa, NJ: Littelfield, Adams & Co. 23

Prelipcean, A. C., Schmid, F., and Shirabe, T. (2015). A space time alarm. In *Progress in Location-Based Services 2014*, pages 187–198. Springer. 42, 51

Putz, S. (1994). Interactive information services using world-wide web hypertext. In *First International Conference on the World-Wide Web, May 25-27, 1994 in Geneva, Switzerland*. Elsevier Science BV. 1

Raubal, M. (2001). Human wayfinding in unfamiliar buildings: a simulation with a cognizing agent. *Cognitive Processing*, 2(3):363–388. 94

Raubal, M. and Kuhn, W. (2004). Ontology-based task simulation. *Spatial Cognition and Computation*, 4(1):15–37. 94

Raubal, M., Miller, H., and Bridwell, S. (2004). User-centred time geography for location-based services. *Geografiska Annaler: Series B, Human Geography*, 86(4):245–265. 17, 35, 42, 72

Raubal, M., Winter, S., Teβmann, S., and Gaisbauer, C. (2007). Time geography for¡ i¿ ad-hoc¡/i¿ shared-ride trip planning in mobile geosensor networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(5):366–381. 17, 35

Reitsma, F. and Bittner, T. (2003). Scale in object and process ontologies. In *Spatial Information Theory. Foundations of Geographic Information Science*, pages 13–27. Springer. 37, 58

Richter, D., Vasardani, M., Stirlng, L., Richter, K.-F., and Winter, S. (2013). Zooming in–zooming out hierarchies in place descriptions. In *Progress in Location-Based Services*, pages 339–355. Springer. x, 63, 64, 69

Roedinger, H. L. (1996). *Prospective memory: Theory and applications*. L. Erlbaum. 18

Russell, S. and Norvig, P. (2010). *Artificial intelligence: a modern approach*. Prentice hall. 25

Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2):115–135. 58

Sacerdoti, E. (1975a). The nonlinear nature of plans. Technical report, DTIC Document. 25

Sacerdoti, E. (1975b). A structure for plans and behavior. Technical report, DTIC Document. 25

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260. 58

Schank, R. C. and Abelson, R. P. (1975). *Scripts, plans, and knowledge*. Yale University. 22

Schatzki, T. R. (1991). Spatial ontology and explanation. *Annals of the Association of American Geographers*, 81(4):650–670. 12, 13, 58

Schatzki, T. R. (2010). *Site of the Social: A Philosophical Account of the Constitution of Social Life and Change*. Penn State Press. 12

Schmidt, M. and Weiser, P. (2012). Web mapping services: Development and trends. In *Online Maps with APIs and WebServices*, pages 13–21. Springer. 2

Scholnick, E. K. and Friedman, S. L. (1987). The planning construct in the psychological literature. *Blueprints for thinking: The role of planning in cognitive development*, pages 3–38. 23, 25

Searle, J. R. (1995). *The construction of social reality*. Simon and Schuster. 61

Sellen, A., Louie, G., Harris, J., and Wilkins, A. (1997). What brings intentions to mind? an in situ study of prospective memory. *Memory*. 10

Sellen, A. and Whittaker, S. (2010). Beyond total capture: a constructive critique of lifelogging. *Communications of the ACM*, 53(5):70–77. 9, 10, 51, 110

Sellen, A. J., Fogg, A., Aitken, M., Hodges, S., Rother, C., and Wood, K. (2007). Do life-logging technologies support memory for the past?: an experimental study using sensecam. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 81–90. ACM. 9, 10

Smith, B. (1995). On drawing lines on a map. In *Spatial Information Theory A Theoretical Basis for GIS*, pages 475–484. Springer. 59

Stevens, A. and Coupe, P. (1978). Distortions in judged spatial relations. *Cognitive psychology*, 10(4):422–437. 64

Timpf, S., Volta, G., Pollock, D., and Egenhofer, M. (1992). A conceptual model of wayfinding using multiple levels of abstraction. *Theories and methods of spatio-temporal reasoning in geographic space*, pages 348–367. 36, 58

Tomitsch, M., Grechenig, T., and Wascher, P. (2006). Personal and private calendar interfaces support private patterns: diaries, relations, emotional expressions. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, pages 401–404. ACM. 31

Tuan, Y.-F. (1974). *Topophilia*. Englewood Cliffs, NJ: Prentice-Hall. 35

Tuan, Y.-F. (1975). Place: an experiential perspective. *Geographical Review*, pages 151–165. 12, 13, 35

Tulving, E. and Thomson, D. M. (1973). Encoding specificity and retrieval processes in episodic memory. *Psychological review*, 80(5):352. 8

Turner, A. (2006). *Introduction to neogeography*. " O'Reilly Media, Inc.". 2

Wang, D. and Cheng, T. (2001). A spatio-temporal data model for activity-based transport demand modelling. *International Journal of Geographical Information Science*, 15(6):561–585. 88, 89

Wen, J. (2003). Post-valued recall web pages: User disorientation hits the big time. *IT & Society*, 1(3):184–194. 7

Whyte, W. H. (2012). *City: Rediscovering the center*. University of Pennsylvania Press. 35

Winograd, E. (1988). Some observations on prospective remembering. *Practical Aspects of Memory: Current Research and Issues (Vol. 1)*, 1:348–353. 10

Winter, S., Kuhn, W., and Krüger, A. (2009). Guest editorial: Does place have a place in geographic information science? 12

Zerubavel, E. (1985). *Hidden Rhythms: schedules and calendars in social life*. Univ of California Press. 14, 15

Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., and Terveen, L. (2004). Discovering personal gazetteers: an interactive clustering approach. pages 266–273. 44

Amin Abdalla
P.O. Box 1220, Vienna,Austria
Wulzendorfstr. 77/E/155
Tel.: +43660/6550333
abdalla.a84@gmail.com

Education:

Phd in GIScience
Vienna University of Technology, Austria
09/2010 - 2015

MSc. Geographic Information Science
University of Edinburgh, Scotland
09/2009 – 09/ 2010

BSc. Regional Planning and Development
Vienna University of Technology, Austria
10/2006-08/2009

Arabic Language Studies
Fajr Language Institute, Cairo/Egypt & African International University, Khartoum/Sudan
09/2005 – 06/2006

Highschool Graduation
Hegelgasse 14, Vienna/Austria
09/2000 – 10/2004

Work Experience:

Software Engineer
CNS Solutions & Support GmbH
04/2015

University Assistant
Vienna University of Technology, Austria
09/2010 – 12/2014

Tutor GIS course (Part Time)
Vienna University of Technology, Austria
02/2009 – 07/2009

Full Time Internship
Western Union International Bank (Central), Vienna/Austria
07/2009 – 09/2009

Front Agent (Part Time)
Western Union International Bank (Branch), Vienna/Austria
07/2006 – 07/2009

Volunteer (Part Time)
Humanic Relief Austria
10/2007 – 07/2009

Interviewer (Part Time)
Triconsult Market Research, Vienna/Austria
03/2001 – 06/2004