REWERSE©
reasoning on the web

A1-D8

An International Appointment Scheduling System – Requirement Analysis

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	IST506779/Munich/A1-D8/D/PU/a1
Responsible editors:	H. J. Ohlbach
Reviewers:	Jakob Henriksson
Contributing participants:	Munich
Contributing workpackages:	A1
Contractual date of deliverable:	30 August 2007
Actual submission date:	10 September 2007

Abstract
The REWERSE WG A1 is, among other things, concerned with the computational treatment of temporal notions (CTTN). In order to demonstrate that the developed methods and systems can solve nontrivial problems, it is planned to develop and implement a nontrivial application, an international appointment scheduling system. The scenario is that a number of people at different places in the world want to meet at some other place in the world. Every participant inserts his own constraints, and the system is supposed to find a solution which respects all constraints. The constraints should be as realistic as possible.

The ideas presented in this document go beyond the original requirement. The proposed methods should be powerful enough to solve many constraints of different types for many time objects which can also be of different types. In particular, fuzzy constraints for fuzzy intervals should be allowed, and constraints can be associated with different degrees of strictness.

The key idea for solving such heterogeneous constraints is to map the constraints to potential fields and to search a solution with minimal energy. Therefore the system will be called PoFiS (**Potential Field Scheduling**).

In this document only the basic ideas and a general system design are presented. The technical details of the method will be presented in the follow-up deliverable A1-D9. The main purpose of this document is to convice the reader that the approach is very promising in dealing with heterogeneous and complex constraints and to clarify what kind of work has to be done in order to realise the ideas.

Keyword List
appointment scheduling, potential field method

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

An International Appointment Scheduling System – Requirement Analysis

Hans Jürgen Ohlbach

Department of Computer Science, University of Munich
E-mail: `ohlbach@pms.ifi.lmu.de`

10 September 2007

Abstract
The REWERSE WG A1 is, among other things, concerned with the computational treatment of temporal notions (CTTN). In order to demonstrate that the developed methods and systems can solve nontrivial problems, it is planned to develop and implement a nontrivial application, an international appointment scheduling system. The scenario is that a number of people at different places in the world want to meet at some other place in the world. Every participant inserts his own constraints, and the system is supposed to find a solution which respects all constraints. The constraints should be as realistic as possible.

The ideas presented in this document go beyond the original requirement. The proposed methods should be powerful enough to solve many constraints of different types for many time objects which can also be of different types. In particular, fuzzy constraints for fuzzy intervals should be allowed, and constraints can be associated with different degrees of strictness.

The key idea for solving such heterogeneous constraints is to map the constraints to potential fields and to search a solution with minimal energy. Therefore the system will be called PoFiS (**Potential Field Scheduling**).

In this document only the basic ideas and a general system design are presented. The technical details of the method will be presented in the follow-up deliverable A1-D9. The main purpose of this document is to convice the reader that the approach is very promising in dealing with heterogeneous and complex constraints and to clarify what kind of work has to be done in order to realise the ideas.

Keyword List
appointment scheduling, potential field method

Contents

1 Introduction

The REWERSE WG A1 is, among other things, concerned with the computational treatment of temporal notions (CTTN). In order to demonstrate that the developed methods and systems can solve nontrivial problems, it is planned to develop and implement a nontrivial application, an international appointment scheduling system. The scenario is that a number of people at different places in the world want to meet at some other place in the world. Every participant inserts his own constraints, and the system is supposed to find a solution which respects all constraints. The constraints should be as realistic as possible.

This document contains a requirement analysis. A follow up deliverable will contain technical details of the planned system. A requirement analysis is the first activity in most industrial software development processes. It analyses the needs of the customer and describes the functionality of the desired product. The primary aspects are the needs of the customer. The methods, algorithms and tools which are to be used or developed to satisfy these needs are secondary. This makes it a typical top-down approach.

In a research environment, however, the approach is usually bottom-up. The bottom level consists of the techniques which have been or are to be developed in a research project. The top level may still be a concrete product, but since there is usually no concrete customer, one is free to design the product in a way that makes optimal use of the techniques which are in the main focus of the research project. Moreover, it is possible to design a product which does not only serve one concrete purpose, but which is useful for other quite different tasks. In our case it means that besides appointment scheduling, one can consider other scheduling applications, time tabling for schools and universities, for example.

This document is therefore a requirement analysis of the bottom-up type. We start with a description of the tools and methods and then work upwards to the design of the concrete scheduling system.

It is not the intention to compete with industrial job shop scheduling systems. They are optimised for processing large amounts of relatively uniform and simple constraints. In contrast to this our system will be designed for complex and heterogeneous constraints. Since the envisioned system will be interactive, the number of constraints is limited to what a few users still can oversee. Nevertheless, a time table for a school or university should still be within the capabilites of the system.

The document is organised as follows: the next section introduces the basic algorithmic idea for solving the scheduling problem. It is a potential field approach. There is a potential field associated with every relation between time objects. The potenital field causes a search algorithm to push time objects into locations with minimal energy The scheduling problem then amounts to finding a global configuration with minimal energy. This approach is extremely flexible because it allows one to combine all the different kinds of time points and time intervals and relations between them. Section 3 explains in more detail how the potential field approach deals with various kinds of phenomena. A short introduction to the CTTN system which will be used by the planner for working with temporal notions is given in Section 4. The theoretical work to be done in order to realise the potential field scheduling planner is sketched in Section 5 and a first system design is presented in Section 6. Finally, some hints to related work are given in Section 7.

2 The Basic Idea of the Potential Field Scheduling

The very idea of the potential field scheduling planner algorithm is to turn constraints into potential fields and to search a configuration with minimal energy. The purpose of this section is to give a first impression of the method and to show its flexibility in dealing with very heterogeneous constraints and time objects.

Let us start with the most simple example of all. The constraint is “the meeting should start before 10 am”. In mathematical terms this yields the relation $t < 10$ where t is the start point of the meeting. A corresponding potential field could be the vector in Fig. 1. The vertical dimension E indicates the energy of the potential field curve P

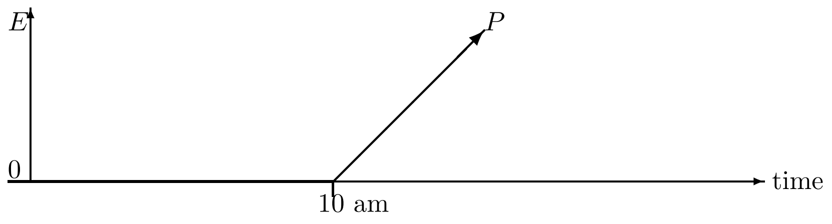

Figure 1: Potential Field for $t < 10$

Any time point t before 10 am would be a solution with energy $E = 0$. By adjusting the curve P one can generate other versions of the purely mathematical <-relation. For example, a potential field curve for “long before 10 am” could be as shown in Fig. 2 with a linear increase, or, as shown in Fig. 3 with an exponential increase.

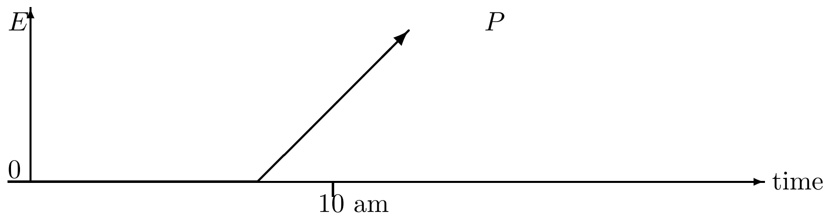

Figure 2: Potential Field for “long before 10 am”, linear increase.

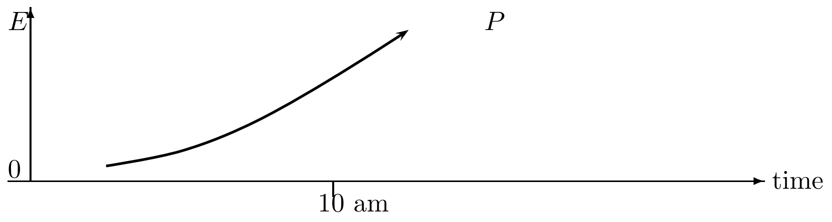

Figure 3: Potential Field for “long before 10 am”, exponential increase.

As another example, “shortly before 10 am” could be represented with the potential field as in Fig. 4. Any time point between 9 am and 10 am has energy $E = 0$ and is therefore a perfect solution.

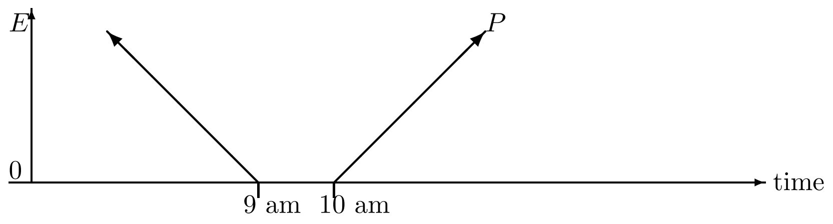

Figure 4: Potential Field for "shortly before 10 am"

These examples demonstrate

1. that variants of *before* relations can be dealt with;
2. since there are infinitely many variants, a scheduling system should have a graphical editor for the potential field curves. With this editor a user can build up a library of relations with the corresponding potential field curves.

2.1 Overlapping Potential Fields

The gradient of the potential field curves P in Fig. 1 and 2 after 10 am indicates the *strictness* of the constraint $t < 10$. If P is not vertical at 10 am, other constraints can push t after 10 am. The corresponding energy indicates how serious the constraint $t < 10$ is violated. This can happen if two or more constraints contradict each other.

Fig. 4 shows a very simple configuration where a perfect solution with energy $E = 0$ exists. In more complex configurations, potential fields can overlap. Consider the two *contradicting* constraints $t < 10$ and $t > 11$. A classical constraint solver would just report "inconsistent". With the potential field approach one can model that the constraints are not so strict and therefore violations are allowed to a certain degree. A "solution" with minimal energy yields then a value E which indicates how serious the violation is.

If there is more than one potential field, there are at least two different ways to combine them:

1. The resulting potential field is the *maximum* value of the participating potential fields.
2. The resulting potential field is the *sum* of the participating potential fields.

The two versions yield quite different results as Fig. 5 illustrates.
The two constraints in Fig. 5 a) have equal strength. Therefore the "maximum"-version yields a minimal energy exactly in the middle, whereas the "sum"-version yields a plateau. Any solution within the plateau violates the two constraints such that the "sum of the violations" is the same. The constraint $t > 11$ in Fig. 5 b) has twice the strength of the constraint $t < 10$. The "maximum"-version therefore yields a unique minimal energy solution where the $t > 11$ constraint is less violated than the $t < 10$ constraint. The "sum"-version, however, yields a minimal energy solution where the strong constraint is not violated at all at cost of the weaker constraint. Thus, the user has to make a strategic decision: the "maximum" version prefers solutions where violations are more equally distributed. The "sum" version, however, prefers solutions where stronger constraints dominate more.

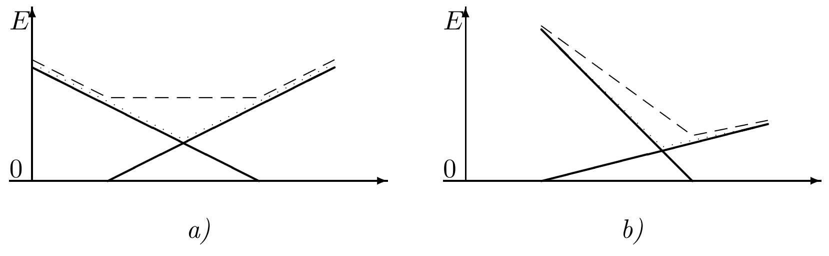

Figure 5: Maximum (dotted) and Sum (dashed) of two Potential Fields

2.2 Constraints between Fuzzy Intervals

In the previous section we have seen the potential field for time points. In this section we demonstrate that the very same basic idea also works for much more complex objects and relations.

Consider the constraint "the birthday party should end before late night". "Late night" determines a fuzzy interval as in Fig. 6 [8]. Since birthday parties usually do not have a fixed start and a fixed end, one can model them also with a fuzzy interval.

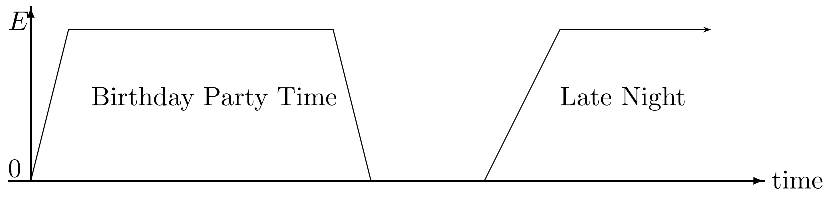

Figure 6: "Birthday Party Time" and "Late Night"

Now there is the problem to model a relation *before* for fuzzy intervals. In the literature there are different ways to define versions of fuzzy binary relations [13, 16, 20]. For two concrete intervals I and J all of the proposals for fuzzy relations yield a fuzzy value between 0 and 1. In case of the fuzzy 'before'-relation, a result of 1 means that I is definitely before J, and a result of 0 means that I is definitely not before J. Any value between 0 and 1 indicates a degree of beforeness. Fig. 7 shows for a particular version of a before relation the resulting fuzzy value (dashed line) for the case that J is kept fixed and I is moved along the time axis [16] (The dotted line indicates the position of I where the fuzzy before relation has fallen to 0.)

Fig. 8 shows an example for a fuzzy *meets*-relation. The dotted line shows the position of the interval I where $meets(I, J)$ is maximal.

Finally, Fig. 9 shows an example for a fuzzy *starts*-relation. The dotted line shows the position of the interval I where $starts(I, J)$ is maximal.

These examples show that, given two intervals with a fixed shape, but variable position, one can compute a fuzzy distribution from the fuzzy relation between them. The fuzzy distribution is also an ordinary fuzzy interval. Let us call it the *relation interval.* Its shape depends on the shape of the two fuzzy intervals, and its position depends on the position of one of the

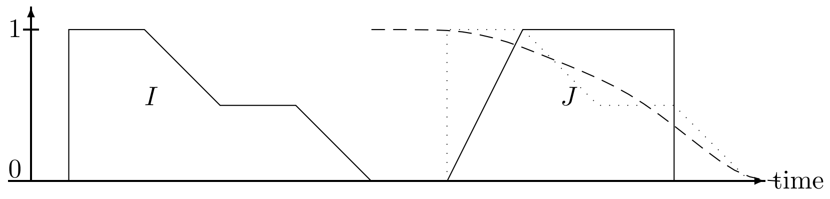

Figure 7: Fuzzy 'before' Relation

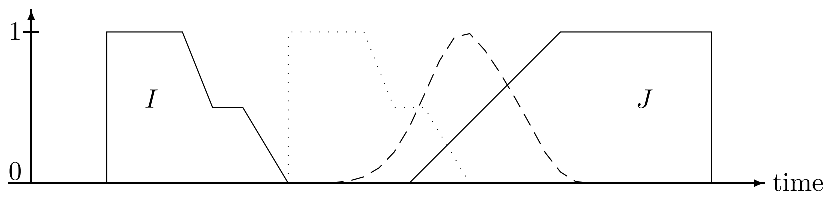

Figure 8: Fuzzy 'meets' Relation

two intervals. Very important is that *the relation interval can be computed once before the scheduling algorithm starts.*

Can we turn the relation interval R into a potential field? The answer is *yes*, in two steps. The first step is just to invert R's membership function: $P(x) = 1 - R(x)$. The solid line in Fig. 10 shows the inverted fuzzy 'before'-relation of Fig. 7 (dotted line).

The thus generated potential field can now be used to move the interval I into a position with minimal energy, in this case, the "before region".

In a second step one can modify the so generated potential field to indicate how strict the constraint is. The area where the relation interval is 0 is in the inverted relation interval just the constant 1. This cannot be used to define a degree of constraint violation. Therefore one should turn the constant area into a rising potential field, as indicated in Fig. 11. This can be done automatically or with a potential field editor. The gradient indicates the strictness of the relation.

2.3 The Scheduling Algorithm

The key part of the scheduling algorithm is a fixed point iteration. In each iteration step a single time object (time point, crisp or fuzzy interval) is chosen, its potential field is computed in the context of the locations of the other time objects and the object is moved to a location with minimal energy.

The shapes of the potential fields can all be precomputed, in general as polygons, such that the computation of the potential field in a particular iteration step is a simple combination of polygons. This in turn can be computed with a sweep line algorithm. The complexity of the sweep line algorithm is in the order of the vertices and intersection points of the polygons. In the same sweep one can locate the position with minimal energy.

For grid floating time objects (Sect. 3.3) it may occur that the location with minimal energy is not an admissible location. For example, if the interval is supposed to be a day, it cannot start at, say, 8 pm. In this case a further search step is necessary. Starting from the position t

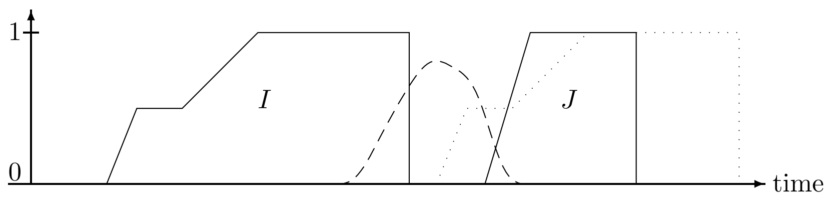

Figure 9: Fuzzy ‘starts’ Relation

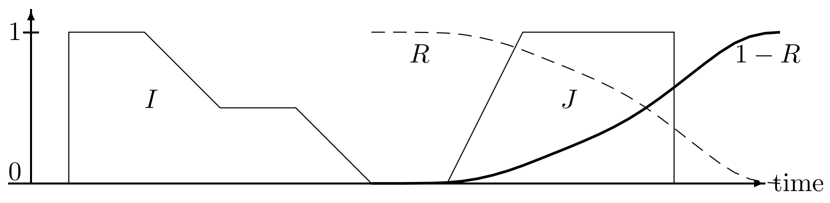

Figure 10: Inverted Fuzzy ‘before’ Relation

with minimal energy, the interval is moved to an admissible position to the left of t and to an admissible position to the right of t. The one with smaller energy is taken.

Since the iteration steps require concrete locations of $n-1$ time objects, we must determine an initial configuration. The easiest way is to place all time objects at some default location. It is, however, more efficient to try to satisfy simple constraints right at the beginning. For example, a constraint like $t < 10$ could easily be satisfied by placing t initially at some time point before 10.

Without further precaution the algorithm may oscillate between different configurations with minimal energy. Therefore it is necessary to monitor the progress and break such kind of loops.

3 Basic Considerations about Time Objects

3.1 Symbolic or Concrete Time Objects

A time point can be either concrete, i.e., a real number as the coordinate on the time line, or just a symbolic variable name whose interpretation is a time point on the time line. In some temporal logics time points can even be implicitly introduced, for example by a $\Diamond$-operator. In the same way we can distinguish the representation of time intervals either as concrete convex subsets of the time line, or symbolically represented by a name.

In a symbolic calculus one can, for example, derive for the time points t_1, t_2 and t_3 and the relations $t_1 < t_2$ and $t_2 < t_3$, via transitivity, $t_1 < t_3$. For symbolic time intervals there is Allen's calculus which allows one, for example, to derive from I_1 before I_2 and I_2 meets I_3 the fact I_1 before I_3 [1].

The strategic decision for a scheduling system is now: do we work with concrete time objects or with symbolic names? In the latter case we would need a symbolic calculus, for example, some variant of Allen's constraint calculus for intervals. Scheduling problems need in fact to

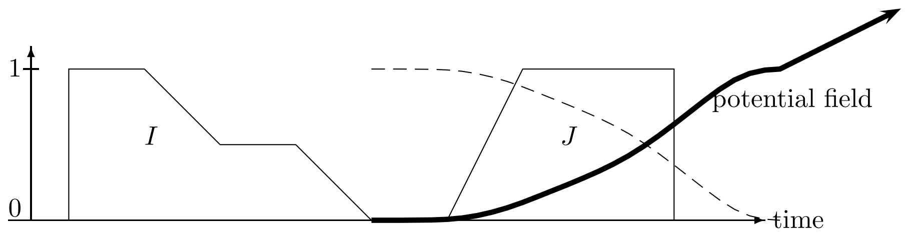

Figure 11: Final Potential Field for Fuzzy Before-relation

deal with time points and intervals whose position is *not* determined. Therefore a symbolic approach might seem appropriate. There are, however, at least three arguments against a purely symbolic approach:

1. Besides *floating* time objects, i.e. those which are not located at the time line there are usually also *fixed* time objects. An example is "the meeting must take place before Friday, 5pm". Here we have a floating time point, the time of the meeting, and a fixed time point, Friday 5pm. Therefore the calculus must be able to deal with fixed time objects as well.
2. The system should also be able to deal with time intervals with an internal structure, in particular with fuzzy intervals. An example could be the representation of "late night" as in Fig. 6. For the description of a fuzzy interval it is essential to have concrete coordinates, even if the fuzzy interval is not precisely located at the time axis (if, for example, it is not clear which night is meant).
3. The solution of a scheduling problem should be a concrete distribution of the time objects over the time line. That means, from a logical point of view, we are faced with a model finding problem, not a general inference problem. Thus, purely symbolic approaches, whose models are kept implicit, are not appropriate in this case.

A pragmatic compromise between the two versions is: All time objects are located at concrete places at the time line. Floating time objects have an additional *offset* parameter which can be changed by the scheduling algorithm. A floating time object is placed at the beginning at a default position at the time line. It is the task of the scheduling algorithm to determine the right offset value. This simplifies the treatment of fuzzy intervals considerably because one can represent the shape of the membership function with concrete coordinates and move the interval around by changing the offset value.

3.2 Crisp Intervals with Fuzzy Length

It turns out that working with concrete time objects simplifies the treatment of another kind of fuzzy intervals, those with uncertain length. Consider the situation: "Meeting A starts at 10 am and lasts for about 2 hours. Meeting B should be after meeting A". Meeting A determines a crisp time interval with uncertain length. The uncertain length can be represented by a probability distribution over the size of the intervals (see Fig. 12).

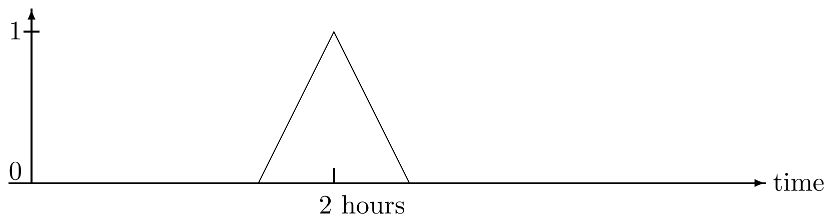

Figure 12: “About two hours” with uncertain length

The situation in the above example is depicted in Fig. 13

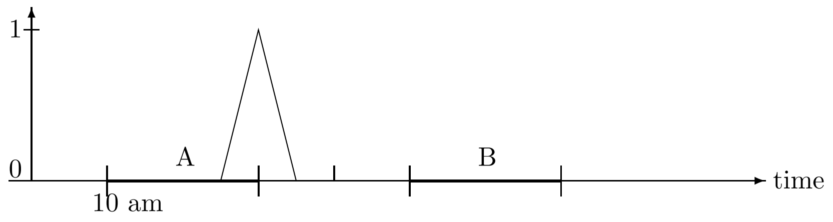

Figure 13: Meeting B after meeting A, meeting A with fuzzy length.

If such an interval with uncertain length is placed at a concrete position at the time line, it is possible to turn the probability distribution into a fuzzy value of an ordinary fuzzy interval. We must distinguish three cases:

1. The interval has a precise starting point, but uncertain length (example: the meeting starts exactly at 10 am and lasts for *about* 2 hours.)

2. The interval has a precise end point, but uncertain length (example: the meeting must end at 12 pm, and lasts for *about* 2 hours).

3. Both the starting point and the end point are uncertain (example: the meeting starts *around* 10 am and lasts for *about* 2 hours.)

Here we consider only the first case. The other cases are similar. Consider an interval as in Fig. 12. For a particular time point t there is a certain probability to belong to the interval. This probability is given by the probability that the length of the interval is longer than $(t - t_0)$ where t_0 is the starting point of the interval. This probability in turn is given by the formula $1 - \int_0^{t-t_0} \varphi(l)dl$ where $\varphi(l)$ is the probability distribution over the length of the interval. The resulting fuzzy interval for “about 2 hours” looks like in Fig. 14.

The consequence is that intervals with uncertain length can be treated with the same mechanisms as fuzzy intervals.

3.3 Free Floating or Grid Floating Time Objects

A statement like “I’ll arrive sometime this afternoon” specifies a time point which can be at any time within the given bounds, this afternoon. It can be represented by an ordinary variable

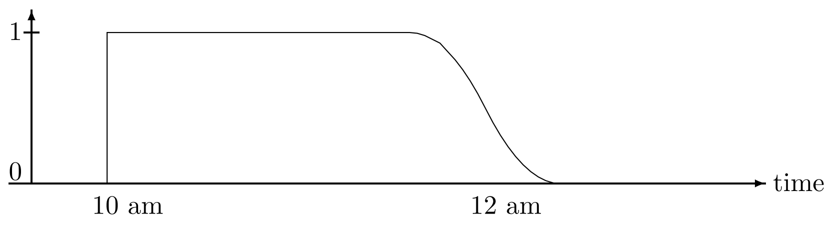

Figure 14: "About two hours" as concrete interval

t which is bound by the inequations $a \leq t \leq b$.

A constraint like 'the meeting must start before noon', on the other hand, refers to a time point, *noon*, which cannot float freely. It is bound to the grid of all 'noon values'. A scheduling algorithm which tries to find an optimal position for 'noon' can only jump between the allowed values for 'noon'. Let us call this phenomenon *grid floating*. The grid can be determined in an even more indirect way than 'noon'. Consider the statements "The two meetings should be in the same month." and "The joint press conference should be at the 31st." This determines a floating month interval, but it must be a month with a 31st day, i.e. January, March, May, July, August, October or December. The grid for the 'grid floating month interval' is determined by the condition that it has a 31st day.

In a generalised setting the grid can be determined by any kind of condition (e.g. "month without heavy rainfall"). Scheduling algorithms must therefore have access to an interface which allows them to check external conditions for grid floating time objects.

The above example, 't before noon', is actually untypical for grid floating time objects. One could of course treat 'noon' as an ordinary time object for which a location with minimal energy must be determined. Since 'noon' determines a sequence of time points within a sequence of days, one could turn a constraint 't before noon' into a potential field which pushes t before noon (see Fig. 15). This way 'noon' would be eliminated from the iteration.

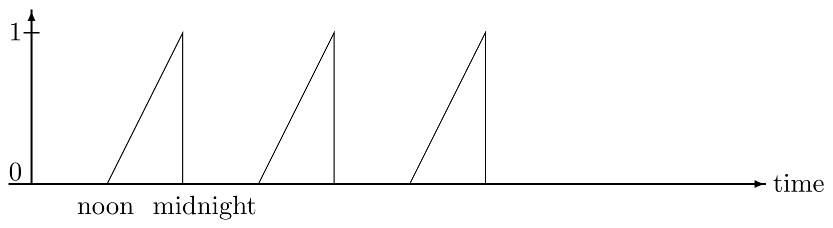

Figure 15: 'before noon' as potential field.

4 The CTTN-System

The CTTN-system (Computational Treatment of Temporal Notions [15]) will be a key component of the PoFiS-system. Therefore we give a very brief overview over its components. There are three main components:

1. The FuTI-module (Fuzzy Temporal Intervals) is designed for representing and manipulating time points, crisp and fuzzy intervals. It contains a large number of operations on

these intervals. For the purposes of the PoFiS-system it has to be extended to handle also crisp intervals with uncertain length and also to perform the transformation to normal fuzzy intervals.

2. The PartLib-module (Partitioning Library) can represent and manipulate periodic temporal notions like days, years, seasons, holiday sequences, Easter time, sunrises, tides etc. There are a number of different XML-formats for specifying periodic temporal notions in different ways, but they can be accessed via a common interface. A further component of PartLib is a CalendarSystem class. It provides various different calendar systems, simple ones like the Gregorian system, but also complex ones, for example for a traveller crossing different time zones and local calendar systems.

3. The Geotemporal Specification Language (GeTS) is a functional language with many built-ins specifically tailored for temporal notions. GeTS can, for example, be used to define fuzzy point-interval and interval-interval relations.

   A fuzzy point-interval relation which maps a fuzzy interval I to the fuzzy interval of all points before I could be formulated in GeTS as follows:

   ```
   PIR::Before(Interval I) =
        if isEmpty(I) then []
        else complement(extend(I,positive))
   ```

   The function `extend(I,positive)` extracts the front part of I and extends it to $+\infty$. The `complement` function complements the extended interval, i.e. it maps the fuzzy membership function $I(t)$ to $1 - I(t)$.

   The next function computes a fuzzy interval-interval relation by averaging a point-interval relation (the parameter B) for the second interval over the first interval. It also treats some extreme cases.

```
IIR::Before(Interval I, Interval J, Interval->Interval B) =
     case isEmpty(I) or isEmpty(J): 0.0,
          isInfinite(I,right) or isInfinite(J,left): 0.0,
          (point(I,right,support) <= point(J,left,support)): 1.0,
          isInfinite(I,left):
              Let K = intersection(I,J) in
                  if(isEmpty(K)) then 1.0 else integrateAsymmetric(K,B(J))
     else integrateAsymmetric(I,B(J))
```

   The function could, for example, be called with `IIR:Before(a,b,PIR::Before)` where the `B`-parameter is the previously defined point-interval relation.

So far, there is only a socket based interface to the CTTN-system. The development of a web based CTTN client is planned. A suitable CTTN-client with sophisticated editing features considerably simplifies the maintenance of CTTN-libraries with definitions of temporal notions.

A further extension of the CTTN-system is a link to the EFGT-net [19]. The EFGT-net (Events in thematic Fields, Geographical and Temporal context) is developed for the representation of *named entities* in a three dimensional context, the thematic field, the geographic and temporal location. Together with the CTTN-system it can be used to represent notions like "before the death of former chancellor Konrad Adenauer".

Since the PoFiS planner needs direct access to the data structures and algorithms of the CTTN-system, it will be an integral part of it.

5 Theoretical Work to be Done

Before a concrete system design can be worked out, a number of theoretical problems need to be solved.

5.1 Fuzzy Intervals

All the details about concrete fuzzy intervals have been worked out and implemented in the FuTI-module of the CTTN-system. The representation and treatment of crisp intervals with uncertain length, however, has still to be worked out. Most of it is straight forward, even the transformation into fuzzy intervals by integrating over the length distribution, yet it stil has to be done.

5.2 Periodic Temporal Notions

There are quite a number of approaches for representing periodic temporal notions in the literature, for example [11, 3, 7, 17, 4, 10, 14]. One of them is implemented in the PartLib-module of the CTTN-system [18]. A notion like 'noon' is realised in these approaches either as a sequence of time points, the 'noons', or as a sequence of intervals, from one noon to the next noon. For representing a relation like "before noon", however, this is not enough. The common understanding of "before noon" is relative to a day. "Before noon" is true from midnight until noon, and false from noon until the next midnight. Thus, a representation of "before noon" needs both the sequence of time points that correspond to "noon" and the sequence of days that contain the noons.

There are a lot of other temporal notions, which show a similar phenomenon. "Before Christmas", for example, is meant relative to a year. It is true from the beginning of a year until Christmas, and false from Christmas until New Years Eve. Even relations like "before the Second World War" may refer to another sequence, the sequence of wars. The general understanding is that "before the Second World War" determines only the interval between the two world wars, and not the infinite interval before the second world war.

That means, in order to represent relations like "before X" where X is a periodic temporal notion we need a formalism where the periodic temporal notion X can be treated in the context of another periodic temporal notion Y (noon within days, Christmas within years, Wednesdays within weeks etc.).

The phenomenon is particularly important for scheduling problems. Many constraints look in this case like, for example, $t > 8am$ ("the meeting must start after 8 am"). It means that the relation is true between 8 am and midnight, and false from midnight until 8 am. "8 am" is in this case a periodic temporal notion in the context of a day partitioning of the time axis.

5.3 Relation Mappings

It will be very inconvenient if the PoFiS-clients require the user to edit all potential fields directly. Even a graphical editor for potential field curves does not help much. A text based specification, for example "end(meeting_A) before noon" would be much more user friendly. Nevertheless, since the system needs potential fields there must be a mapping from names like 'before' to a corresponding potential field. In Section 2.2 we have illustrated already that potential fields can be generated more or less automatically from the relations between time objects. Only the strictness of a relation is a parameter which modifies a potential field independently of the underlying relation.

Therefore the first step is to develop a mapping from relation names like 'before' to the corresponding relations. This problem has at least two dimensions. The first dimension is that there are usually different versions of the relations ('shortly before', 'before', 'long before' etc.) The second dimension is that the relations depend on the type of the objects. As we have already explained, it makes a difference for the interpretation of a relation like 'A before X' if X is a simple time point or a periodic temporal notion (like 'noon').

Even more complex are parametric binary relations. As an example, consider the constraint "meeting A and meeting B should be at the same day". It contains two different pieces of information. The first is that both meetings should be *within a day.* The second information is that both meetings should be at *the same day.* The first constraint can be mapped to a potential field which pushes A and B independently of each other into a day interval. The second constraint could be treated as follows: during the iteration step of the PoFiS-planner where, say interval B is to be moved to a location with minimal energy, a potential field for B is created which pushes B into the day interval which contains A (see Fig. 16)

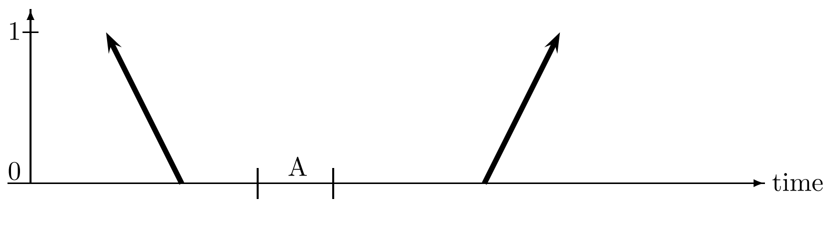

Figure 16: Potential Field for 'same day'

Thus, we need to develop a formalism for constraints like "A and B should be in the same X". It maps "should be in the same" to a data structure which tells the PoFiS-planner what to do with these constraints.

In simple cases one can represent the mapping from relation names to relations by just associating a, possibly fuzzy, interval with the relation name. If the time objects are fuzzy intervals, however, the relation depends on the structure of the interval's membership functions. In this case one must associate a relation name with a construction function that takes two time objects and generates a relation interval. This can be done, for example, by using a function like IIR::Before (Section 4) to iterate over the first interval in oder to generate a relation interval like in Fig. 7.

5.4 Potential Field Mappings

So far we have the steps:

relation name $\mapsto$ construction function $\mapsto$ relation interval $\mapsto$ potential field

where the last step, from a relation interval to a potential field requires only to add a strictness value. It may, however, turn out that more complex relations must be mapped to a potential field in a different way. Therefore the system should implement a general notion of a *potential field mapping* where the above sketched sequence is only one possibility.

The investigations around potential field mappings do not require complex theoretical work, but merely strategic decisions which concern the usability of the system.

5.5 Grid Floating Time Objects

In the iteration step of the PoFiS-planner a time object is first moved to a position with minimal energy and then, if it is grid floating, to an admissible position nearby. In order to do this an interface is necessary which gives the PoFiS-planner access to functions like "move the object X to the next grid position before/after time point t". This interface needs access to data structures which contain the information about the admissible grid positions. The grid itself can be represented with the mechanisms of the CTTN's PartLib-module. Thus, as far as can be seen now, the interface can be quite generic and directly accesses the PartLib functions.

What is not so straight forward is the user interface for this kind of information. The user might, for example, just want to write "lectures should start at even hours". So far, the notion of "even hours" can be formulated in the PartLib system by means of a corresponding XML-definition. Since this is quite cumbersome, one should develop a natural language interface for this kind of constraints.

5.6 The PoFiS-Planner

The PoFiS-planner algorithm is a heuristic search algorithm. A theoretical investigation of its behaviour, under which conditions it terminates, its complexity, whether it can find optimal solutions etc. is very interesting, but also very difficult. Moreover, the behaviour of such algorithms tend to change considerably if the heuristics are changed. A theoretical investigation with heuristic A may be quite useless if the actual implementation uses heuristic B. Therefore it is much easier and makes more sense if the theoretical investigation is done only after an implementation is available. An implemented system allows one to experiment and to gain a better feeling about its behaviour. This is usually quite helpful for theoretical investigations.

6 System Design

6.1 System Structure

The overall system structure of the appointment scheduler is depicted in Fig. 17.

Since it should allow different people at different places to work on the same scheduling problem, it has a client-server architecture. The clients are just Web applications in ordinary Web browsers, with some embedded Java applets. The server consists of a CTTN-server together with the scheduling planner. The CTTN-server deals with the representation and the algorithms for crisp and fuzzy intervals, for periodic temporal notions and for general user defined time functions in the GeTS language. The corresponding definitions are kept in CTTN

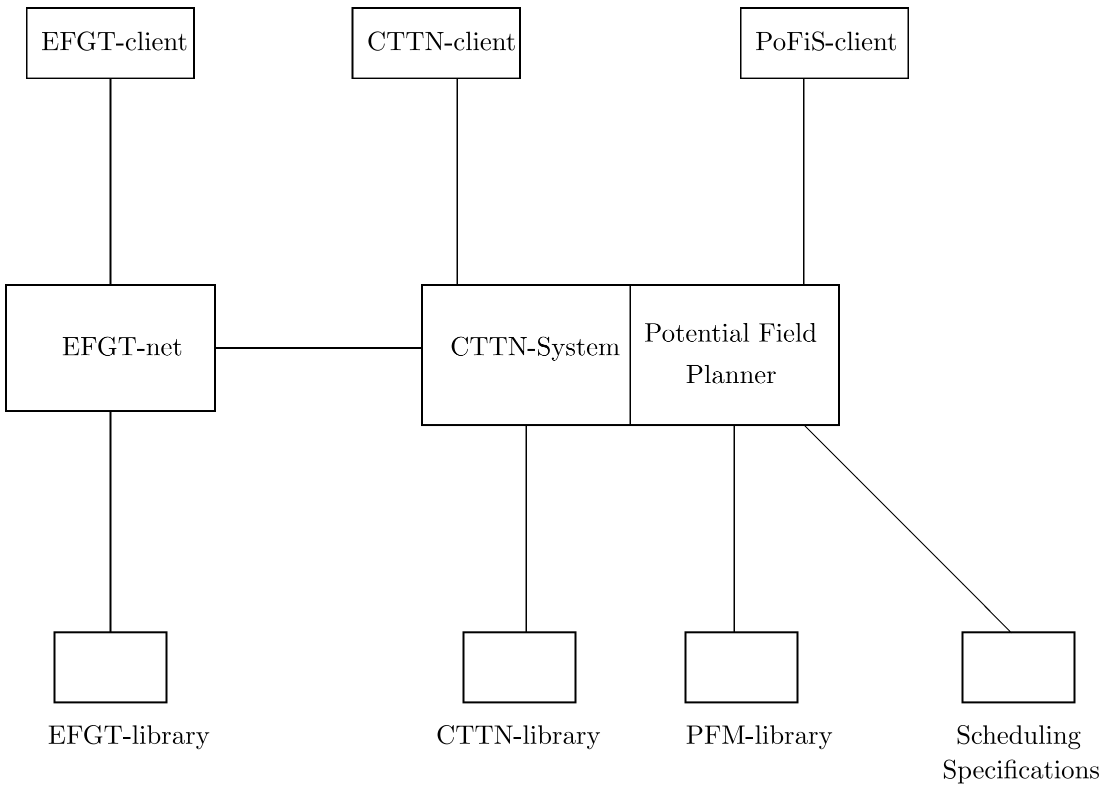

Figure 17: The Architecture of the PoFiS-Planner

specification files, which can be at any place in the internet. They are edited with the corresponding CTTN clients (also Web applications in Web browsers). The CTTN-server is linked to the EFGT-net which is used to represent named entities.

The PoFiS planner maintains “scheduling problems” and runs the scheduling planner. A scheduling problem consists of time objects and relations between them. The transition from the constraints between the time objects to the potential fields is done by “potential field mappings”, which are stored in PFM-libraries.

6.2 PFM-Clients and Workflows

The PoFiS-clients are web browser applications which serve four purposes:

1. to define a context of temporal notions;
2. to edit potential field mappings and to fill the PFM-libraries;
3. to edit the time objects and constraints
4. to monitor and supervise the scheduling planner.

The four steps are also the four main steps in the scheduling workflow:

Step 1: Context of temporal notions.
Very simple scheduling problems with constraints consisting of, for example, Allen's interval relations, can be solved without reference to any real world temporal notions. A constraint like "the meeting should start before noon", however, needs access to at least a realistic definition of 'noon'.

Therefore the first step in formulating a scheduling problem is usually the definition of a context of temporal notions. This consists of:

- the definition of one or more calendar systems;
- the definition of additional periodic temporal notions (noon, weekend, school holidays etc.);
- the definition of concrete events ("soccer world cup in 2006", "the day when Johnny Cramer died" etc.);
- the definition of user-specific temporal notions in the GeTS language.

The temporal notions context consists either of predefined concepts in CTTN-libraries or the EFGT-net, or it must be defined using the CTTN-client or the EFGT-net interface. This would be a step which comes before a new scheduling problem is initiated. The job of the PoFiS-client in this step is only to load corresponding definitions into the temporal context as part of a scheduling problem.

Step 2: Potential Field Mappings
The potential field mapping should be as generic as possible. Therefore they can be edited and stored in PFM-libraries independently of the current scheduling problem. There may also be very specific PFMs which make sense only for a particular scheduling problem. Therefore they can also become part of a currently active scheduling problem.

Step 3: Time Objects and Constraints
In this step all the time points, intervals and relations between the time objects are defined. The relations are turned into potential fields. Automatically generated potential fields may still be edited, for example, in order to adjust the strictness of a relation.

Step 4: Monitoring the scheduling planner
In more complex scenarios it is very unlikely that it is possible to run the planner and to take the result as it is. For example, if conflicts occur it may be necessary to adjust the strictness of the constraints or to change the constraint completely. Therefore the progress of the algorithm should be visualised and it should be possible to intervene at any time.

6.3 A Typical Session

A typical session with the Scheduling Planner could be as follows:

1. A user authenticates himself. New users need to be registered.
2. The user browses the CTTN concepts to check whether the necessary temporal notions are in the system (calendar systems, holidays etc.). Missing concepts can be added using the CTTN and EFGT-clients.

3. The user browses the potential field mappings to check whether suitable PFMs are available. Missing ones are added using the PFM-editor.
4. The user creates a new scheduling job or opens an existing one (which could have been edited previously by somebody else).
5. The user defines time objects and the constraints.
6. The user starts the scheduling planner. He monitors the progress. The planner can be stopped at any time and the constraints can be edited again.
7. The result of the scheduling planner is exported in a suitable XML-format.

7 Related Work

Timetabling problems have been investigated since decades and triggered quite significantly the research about constraint reasoning. Although there exist a number of good timetabling software products for schools, universities and in particular for industrial scheduling problems, they are not as widely used as one would expect it. Reasons may be, for example:

- there may be important time objects and constraints which can not be formulated in the system;
- the user may prefer small changes to previous solutions (last years time table, for example), instead of a more optimal, but completely reorganised solution, the system finds;
- the interface may be too cumbersome not giving the user enough control over the search.
- etc.

In particular, it is usually quite difficult or even impossible to represent fuzzy information in timetabling systems. Therefore *fuzzy temporal reasoning* has become an active research area. One branch of fuzzy temporal reasoning is concerned with modelling vague temporal information about crisp events. Dubois and Prade, for example, introduced a possibilistic framework for fuzzy dates and fuzzy temporal constraints [9]. Various improvements resulted in fuzzy temporal constraint networks [12, 2, 6]. Rough sets have also been used for representing time spans and events [5]. In this approach lower and upper bounds are used for defining temporal relations between two events.

Closer to the approach in this paper are the works where temporal relations are defined as fuzzy relations [20, 13, 16]. To the best of my knowledge, it has not yet been tried to use fuzzy relations as the basis for a scheduling planner. In particular the approach with potential fields, as proposed in this paper, is therefore really new. Potential fields have, however, been used for other problems, for example, robot navigation [21].

8 Summary

This paper has introduced a new approach for scheduling planners with fuzzy time intervals and relations. The relations are mapped to a potential field and the planner searches a configuration with minimal energy. The approach is very flexible because it can handle a number of different phenomena:

- the strictness of a constraint can be fine tuned and solutions which violate the constraints to a certain degree can be found;
- different types of time objects can be handled:
  - fixed and floating time points
  - fixed and floating crisp time intervals
  - fixed and floating fuzzy intervals
  - free floating and grid floating time objects
  - periodic temporal notions and relations involving them
- different constraint types can be handled. The only requirement is that there is a mapping from a constraint type to a potential field.

The proposed approach for the search for an optimal solution is a heuristic fixed point iteration. Concrete experiments have to fine tune the heuristics. A very top level system design of an international appointment scheduling system has also been presented. More theoretical and implementational details will be given in Deliverable A1-D9.

Acknowledgements

The idea to use potential fields for the fuzzy scheduling planner was triggered by a remark of Stefano Bertolo (project officer of REWERSE) during the 2007 review meeting.

References

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Senén Barro, Roque Marín, José Mira, and Alfonso R. Patón. A model and a language for the fuzzy representation and handling of time. *Fuzzy Sets Syst.*, 61(2):153–175, 1994.

[3] Claudio Bettini, Sushil Jajodia, and Sean X. Wang. *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer Verlag, 2000.

[4] Claudio Bettini, Sergio Mascetti, and X. Sean Wang. Mapping calendar expressions into periodical granularities. In C. Combi and G. Ligozat, editors, *Proc. of the 11th International Symposium on Temporal Representation and Reasoning*, pages 87–95, Los Alamitos, California, 2004. IEEE.

[5] Thomas Bittner. Approximate qualitative temporal reasoning. *Annals of Mathematics and Artificial Intellignece*, 36(1-2):39–80, 2004.

[6] Alfonso Bosch, Manuel Torres, and Roque Marín. Reasoning with disjunctive fuzzy temporal constraint networks. In *"Proceedings of the 9th International Symposium on Temporal Representation and Reasoning (TIME 2002)"*, pages 36–43, 2002.

[7] Diana R. Cukierman and James P. Delgrande. Expressing time intervals and repetition within a formalization of calendars. *Computational Intelligence*, 14(4):563–597, 1998.

[8] Didier Dubois and Henri Prade, editors. *Fundamentals of Fuzzy Sets.* Kluwer Academic Publisher, 2000.

[9] Didier Dubois and Henry Prade. Processing fuzzy temporal knowledge. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):729–744, 1989.

[10] Curtis E. Dyreson, Wikkima S. Evans, Hing Lin, and Richard T. Snodgrass. Efficiently supporting temporal granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587, 2000.

[11] B. Leban, D. Mcdonald, and D.Foster. A representation for collections of temporal intervals. In *Proc. of the American National Conference on Artificial Intelligence (AAAI)*, pages 367–371. Morgan Kaufmann, Los Altos, CA, 1986.

[12] Roque Marín, Senén Barro, Alfonso Bosch, and José Mira. Modeling the representation of time from a fuzzy perspectiv. *Cybernetics and Systems: An International Journal*, 25(2):217–231, 1994.

[13] Gábor Nagypál and Boris Motik. A fuzzy model for representing uncertain, subjective and vague temporal knowledge in ontologies. In *Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics, (ODBASE)*, volume 2888 of *LNCS*. Springer-Verlag, 2003.

[14] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proc. of the first International Conference on Information and Knowledge Management*, volume 752 of *Lecture Notes in Computer Science*, pages 161–169. Springer Verlag, 1993.

[15] Hans Jüergen Ohlbach. Computational treatement of temporal notions – the CTTN system. In François Fages, editor, *Proceedings of PPSWR 2005*, Lecture Notes in Computer Science, pages 137–150, 2005. see also URL: http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2005-30.

[16] Hans Jürgen Ohlbach. Relations between fuzzy time intervals. In *Proceedings of 11th International Symposium on Temporal Representation and Reasoning, Tatihoui, Normandie, France (1st–3rd July 2004)*, pages 44–51. IEEE Computer Society, 2004. See also http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2004-33.

[17] Hans Jürgen Ohlbach. Modelling periodic temporal notions by labelled partitionings – the PartLib library. In S. Artemov, H. Barringer, A. d'Avila Garces, L. C. Lamb, and J. Woods, editors, *Essays in Honour of Dov Gabbay*, volume 2, pages 453–498. College Publications, King's College, London, 2005. ISBN 1-904987-12-5. See also http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2005-28.

[18] Hans Jürgen Ohlbach. Modelling periodic temporal notions by labelled partitionings of the real numbers – the PartLib library. Research Report PMS-FB-2005-28, Inst. für Informatik, LFE PMS, University of Munich, June 2005. URL: http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2005-28.

[19] Klaus U. Schulz and Felix Weigel. Systematics and architecture for a resource representing knowledge abo ut named entities. In Jan Maluszynski Francois Bry, Nicola Henze, editor, *Principles and Practice of Semantic Web Reasoning*, pages 189–208, Berlin, 2003. Springer-Verlag.

[20] Etienne E. Kerre Steven Schockaert, Martine De Cock. Imprecise temporal interval relations. In *Fuzzy Logic and Applications, 6th International Workshop (WILF 2005)*, number 3849 in LNAI, pages 108–113. Springer Verlag, 2006.

[21] Kuo-Yang Tu and Jacky Baltes. Fuzzy potential energy for a map approach to robot navigation. *Robotics and Autonomous Systems*, 54(7):574–589, 2006.