Fuzzy Time Intervals
System Description of the FuTI–Library

Hans Jürgen Ohlbach

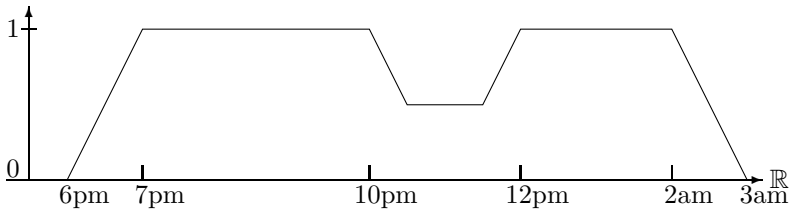
Institut für Informatik, Universität München
E-mail: ohlbach@lmu.de

Abstract. The FuTI–library is a collection of classes and methods for representing and manipulating fuzzy time intervals. Fuzzy time intervals are represented as polygons over integer coordinates. FuTI is an open source C++ library with many advances operations and highly optimised algorithms. Version 1.0 is now available from the URL http://www.pms.ifi.lmu.de/CTTN/FuTI.

1 Fuzzy Time Intervals

Fuzzy Intervals are usually defined through their membership functions. A membership function maps a base set to a real number between 0 and 1. This "fuzzy value" denotes a kind of degree of membership to a fuzzy set S. The base set for fuzzy time intervals is the time axis. In FuTI it is represented by the set $\mathbb{R}$ of real numbers. Real numbers allow us to model the continuous time flow which we perceive in our life. A fuzzy time interval in FuTI is now a fuzzy subset of the real numbers.

A typical fuzzy interval may look like:



Party Time

This set may represent a particular party time, where the first guests arrive at 6 pm. At 7 pm all guests are there. Half of them disappear between 10 and 12 pm (because they go to the pub next door to watch an important soccer game). Between 12 pm and 2 am all of them are back. At 2 am the first ones go home, and finally at 3 am all are gone. The fuzzy value indicates in this case the number of people at the party.

Fuzzy intervals in FuTI may be infinite, but the membership must be constant from certain time onwards.

2 Data Structures and Algorithms

There are four basic data types: time points, fuzzy values, fuzzy temporal intervals and y-functions.

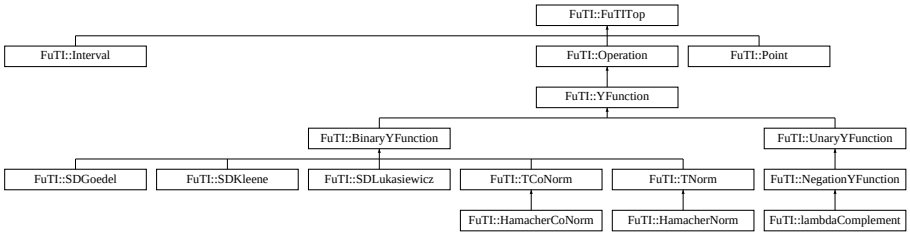
Time Points
The time points are points on the $\mathbb{R}$-axis. Arbitrary real numbers cannot be represented on computers. The choice is therefore between floating point numbers and integers as representation of time points. The range of floating point numbers is much higher than the range of integers. Unfortunately, algorithms operating on floating point numbers are prone to uncontrollable rounding errors. Therefore the FuTI–library *represents time with integer coordinates*. There is no assumption about the meaning of these integers. They may be years, seconds, picoseconds or even cycles of the Caesium 133 light. The system can use two types of integers, 64 bit long integers, and multiple precision integers from the GMP library (http://www.swox.com/gmp).

Fuzzy Values
Fuzzy values are usually real numbers between 0 and 1. A first choice would therefore be to use floating point numbers for the fuzzy values. Again, floating point numbers are prone to rounding errors. Moreover, computation with floating point numbers is more expensive than computation with integers. Therefore FuTI uses again integers instead of floating point numbers. This means of course that one cannot represent the fuzzy value 1 as the integer 1. We could then use just 0 and 1 and no other fuzzy value. Instead one better represents the fuzzy value 1 as a suitable unsigned integer of a certain bit size. Since fuzzy values are estimates only anyway, 16 bit unsigned integer (unsigned short int in C) are precise enough for fuzzy values.

Fuzzy Time Intervals
Fuzzy intervals are usually implemented by a representation of their membership functions. Arbitrary membership functions are almost impossible to represent precisely on a computer. A natural choice for realizing approximated fuzzy time intervals over integer time and integer fuzzy values is the representation with *envelope polygons* over integer coordinates. This has a number of advantages: the representation is compact and can nevertheless approximate the membership functions very well; simple structures, like crisp intervals, have a simple representation; we can use ideas and algorithms from Computational Geometry, there are very efficient algorithms for most of the problems, and it is clear where rounding errors can occur, and where not.

3 The Class Hierarchy

The data structures are organised in the following hierarchy:



3.1 Point

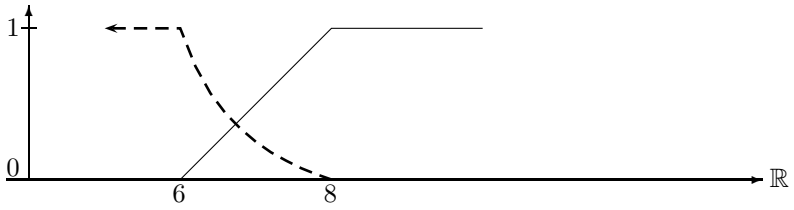
The class 'Point' represents 2D points with integer coordinates. These points form the vertices of the envelope polygons for fuzzy sets. Since two points determine a line segment, all the algorithms for line segments are also contained in 'Point'. The algorithms range from simple 'leftturn' tests up to integration over two multiplied linear functions, where the linear functions are determined by two lines.

3.2 Interval

This is the most important class in the FuTI–library. It contains the representation of fuzzy intervals as polygons. There are two types of operations on these polygons. The first type consists of some dozens of 'hardwired' operations which transform the fuzzy sets in a certain way. Computing three different types of hulls (monotone, convex and crisp hull) is an example. Multiplying a crisp or fuzzy interval with a linear or Gaussian distribution is another example. Further operations are, for example, normalised integrations over the membership functions, from past to future and the other way round.

The second type are parameterised operations on fuzzy intervals where the parameters themselves are operations on membership functions (called Y-functions in FuTI).

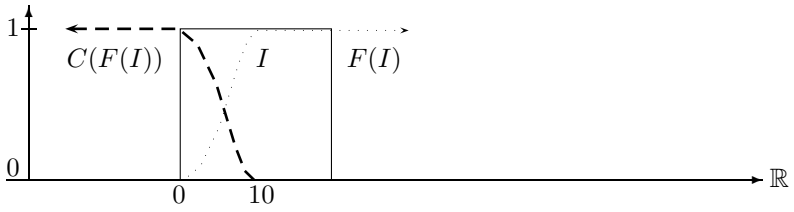
An example for a *unary* transformation of a fuzzy interval is the complement operation, which is defined by a negation function on the membership function. The so-called λ-complement $n_\lambda(y) \stackrel{\text{def}}{=} \frac{1-y}{1+\lambda y}$ function can be used for this purpose.



λ-Complement for $\lambda = 2$

The function n_λ is then a parameter to a 'unary-transformation' operation.

More complex combinations of these transformation functions can compute, for example, a fuzzified point–interval 'before' relation:



Fuzzy Point–Interval Relation

$F(I)$ is a fuzzified and extended version of the interval I where a Gaussian distribution is multiplied with I. C is a complement operation. The result, $C(F(I))$ gives for every time point t the fuzzy value for 't is before I'.

Besides unary transformations, there is also a function 'binary–transformation' on intervals, which is parameterised by a function that takes two fuzzy values as input and computes a new fuzzy value. Set operations like union or intersection, which are determined by so-called T-Norms and T-Conorms are examples for binary transformations.

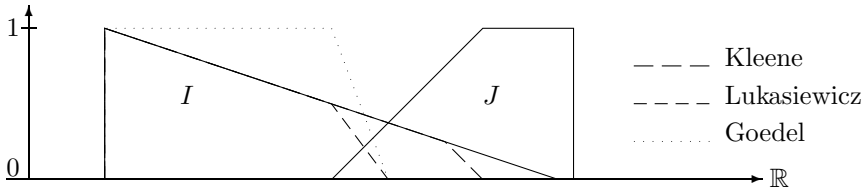
A particular binary transformation is the (normalised) integration over multiplied membership functions. A definition of a fuzzy interval–interval 'before' relation as the weighted average over the point–interval 'before' relation $B(J)$ could be $\mathit{before}(I,J) \stackrel{\text{def}}{=} \int I(t) \cdot B(J)(t)\ dt/|I|$ which is computed by a suitable binary transformation function.

Some of the transformations are non-linear, i.e. they turn straight lines into curves. The algorithms in FuTI approximate the curves automatically with sufficiently dense polygons.

3.3 Operation

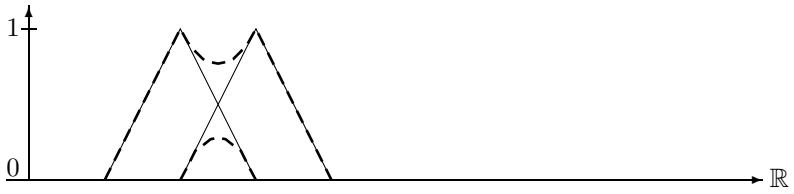
This class is the top class for all unary and binary Y-functions and other operations on intervals. The subclasses of 'Operation' implement a standard repertoire of Y-functions. New Y-functions can easily be added by adding further subclasses of the class 'YFunction'.

The classes 'SDGoedel', 'SDKleene' and 'SDLukasiewicz' implement binary Y-functions which realise three different types of fuzzy set difference operations.



Set Difference $I \setminus J$

The classes 'TCoNorm' and 'TNorm', with their subclasses 'HamacherCoNorm' and 'HamacherNorm' are used for realising fuzzy union and intersection operations. The picture below illustrates the operations.



Hamacher Intersection and Union

The class 'UnaryYFunction' has, so far, only subclasses for standard and *lambda*-complement.

4 Summary

The FuTI-library is a component of the CTTN-system (Computational Treatment of Temporal Notions) [1], a program for evaluating temporal expressions like 'three weeks after Easter'. CTTN is currently under development. CTTN contains in particular the specification language GeTS for specifying and working with temporal notions [2]. Many of the language primitives in GeTS are the operations of the FuTI–library. Other language primitives in GeTS use the PartLib–library for representing periodical temporal notions [3]. GeTS is in particular suitable for specifying fuzzy relations between fuzzy time intervals. Therefore FuTI is only one piece in a bigger mosaic. Some of the design decisions in FuTI are motivated by the needs of the GeTS language. Nevertheless the API for FuTI is general enough to be useful also for other applications.

Acknowledgements

This research has been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

References

1. 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.
2. Hans Jürgen Ohlbach. GeTS – a specification language for geo-temporal notions. Research Report PMS-FB-2005-29, Inst. für Informatik, LFE PMS, University of Munich, June 2005. URL: http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2005-29.
3. 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.