Cultural Calendars for Programming and Querying

François Bry[1] Jutta Haußer[2] Frank-André Rieß[1] Stephanie Spranger[1]

[1] Institute for Informatics, University of Munich, http://www.pms.ifi.lmu.de

[2] Japan-Zentrum, University of Munich, http://www.japan.lmu.de

contact: spranger@pms.ifi.lmu.de

Abstract

Calendar data such as dates are probably more than any other data domain a subject to interpretation. Calendars are human abstractions of the physical flow of time predominately depending on culture, especially religion and history. They allow for measuring time in different units like day, week, working day, and also teaching term. The vision of the Semantic Web is to enrich the current Web with well-defined meanings and to enable computers to meaningfully process such data. This paper contributes to the Semantic Web vision with a generic modeling language called CaTTS. Using CaTTS, one can easily define arbitrary calendars. The language provides with CaTTS-TDL, a tool to define calendars and CaTTS-FDL, a tool to define calendar data, in particular dates to give calendar data well-defined meanings. Furthermore, CaTTS provides a way to enable computers to meaningfully process the calendar data. Thus, CaTTS brings important cultural artefacts, the calendars, on the (Semantic) Web.

1 Importance of Calendars in Cultures

Calendar data such as dates are probably more than any other data domain a subject to interpretation, in almost any case depending on culture. E.g. the date "12/02/2005" is interpreted in France as 12^{th} February 2005 while it is interpreted as 2^{nd} December 2005 in the US. Many traditional Web sites and pages refer explicitly or implicitly to such calendar data. On the current Web, such data can hardly be interpreted by computers. The vision of the Semantic Web is to enrich the current Web with well-defined meaning and to enable computers to meaningfully process such data. This paper contributes to the Semantic Web vision with a generic modeling language called CaTTS [2, 3]. Using CaTTS, one can easily define arbitrary calendars including rather complicated ones. CaTTS provides with CaTTS-TDL, a tool to define calendars and CaTTS-FDL, a tool to define calendar data, in particular dates. Furthermore, CaTTS provides a way to enable computers to meaningfully process calendar data. A thesis underlying the work reported about in this article is that calendars are more conveniently expressed with dedicated language constructs and that calendar data and expressions are more efficiently processed with dedicated reasoning methods than with "axiomatic reasoning" of ontology languages like RDF and OWL.

Calendars are human abstractions of the physical flow of time. They allow for measuring time in different units like day, week, working day, and also teaching term predominately depending on culture, especially religion and history. E.g. the working days of a professional calendar used in some country are defined according to the days in the underlying cultural calendar used. Examples of cultural calendars are the Gregorian, the Julian, the Hebrew, the old and new Japanese calendars, and the old and new Chinese calendars.

Although people in Europe and in Japan both use the Gregorian calendar to measure time, to make appointments, or to schedule travels, the Gregorian calendar used in Japan and European countries present more (often rather subtle) differences than one might think at first: E.g. Christmas Day means December 25 in Greece, but January 7 in Russia, although, Greece and Russia are both dominated by the same (Christian Orthodox) religion. However, in Russia the Christmas Day

is determined according to the Julian calendar while in Greece this day is determined according to the Gregorian calendar which is used in both countries in everyday life. Furthermore, Russia, Greece, and all other Christian Orthodox countries calculate Easter according to the Julian calendar although the Gregorian calendar is used for specifying other religious celebrations. Another particularity concerning (religious) holidays arises in Germany: people in Bavaria do not work on Epiphany (6^{th} January) while people in Schleswig-Holstein work on this day. The rational is that legal holidays are defined at the German federal level, not at the state level, i.e. Epiphany is a legal holiday in Bavaria but not in Schleswig-Holstein. Yet another curiosity concerns German universities and technical institutes: while at a technical institute a lecture would be announced by a time slot like "8:15 to 9:45", at a university a lecture with the same time slot would be announced "8:00 to 10:00 c.t.". The abbreviation c.t., Latin for "cum tempore", defines a time shift by a quarter of an hour. Beyond different interpretations of dates or holidays, the common-sense understanding of some calendar expressions vary: in Western countries like France or the US, 'Friday evening' denotes the eve of Friday but in some Islamic countries, the eve of Thursday. Furthermore, years are numbered differently in Japan and China than in European countries (and not in the same way in Japan, continental China, and Taiwan). E.g. December 24, 1926 is referred to in Japan as Taishō 15, the 12th month, the 24th day, while December 25, 1926 is Shōwa 1, the 12th month, the 25th day.[1] The rationale is that, following the old Chinese practice, years are numbered in the Japanese calendar after the era name. These era names are chosen by each emperor as a kind of governing slogan. The slogan of the present Japanese emperor, whose reign started on January 8th, 1989, Heisei has the meaning "creating peace". Often enough, however, the meanings of these era names being Chinese compounds are quite ambiguous and therefore often open – even if unsaid – new interpretations according to changing situations. Shōwa, the era which started 1926, can mean "brilliant Japan" as well as "radiant peace". Since in modern times it had become the habit not to change the governing slogan during one emperor's reign, this possibility to reread the meaning nevertheless permitted a new start even under the reign of still the same emperor. An emperor is so closely linked to his governing slogan, that instead of using his name, which would be actually a taboo, he is usually referred to as the emperor with the name of his era, i.e. Shōwa tennō, Heisei tennō etc.

These era names on one hand show a perception of time which is, as can be repeatedly found in Asia, circular. A parallel of this perception can also be found in the western view of the year. In the Asian, i.e. predominantly the Chinese, cultural sphere, however, even the year is separated in several circles, depending on the respective system of beliefs. In a country like Japan all these calendrical systems overlap, so it can be quite complicated and confusing to correctly refer to a day. Therefore, fixing a day for a certain occasion can be quite difficult and many calendars have to be consulted. To plan a wedding on a day, when - as it is done repeatedly during the year - the entrance of Buddha into Nirvana is commemorated, for example is supposed to be very unsuspicious. To avoid that, even until today most Japanese couples will consult a Buddhist priest.

This shows on the other hand that calendar systems manifest - as is the case with the era names as well - a great deal of inherent power. To become able to be in charge of time - even if it might be only your own - time has to be modeled. Watches and calendars as common devices can therefore also be seen as one step to personal independence as well as to equal communication. With the growing of a merchant class in pre-modern Japan such devices for common usage became more and more on demand, as there were a sine qua non for monetary acts. But since the calendar system that days was so complicated - for example the long and short months changed each year - and involved so much power, to print calendars was a strict state monopoly. To circumvent this law different modes of riddle like modeling calendars in prints, so called *e-goyomi,* were invented. The length of the months and often even the year was interwoven with the overall picture of the respective prints. Even though these calendars were strictly forbidden, due to their playful

[1] Even though for designating the months in Japanese now their mere numbers are mainly used,there exist other names that evoke the old calender. I.e. by mentioning the first month, e.g. Japanese automatically think of spring, because in the past the first month denoted the beginning of spring. This can still be seen in different greetings for the New Year, where "spring" is frequently used.

approach during the 18th and 19th century private editions were soon en vogue as well as coveted collection articles. Two examples will show how they work.

Figure 1: Japanese calendar illustrations [1], (a) p. 105 and (b) p.102.

At the first glance, Figure 1(a) illustrates only a good luck charm for New Year, showing the God of Good Fortune, Ebisu, together with a red snapper, *tai*[2]. However, the thread loosely laid around the fishing rod tells us in hardly legible letters the year, the 3rd year in the era Meiwa (1766), a dog-year, while the numbers for the long months are hidden in the folds of the garment of the God of Fortune and the numbers for the short months in the outline of the fish.

Figure 1(b) is one of the popular *bijin-e,* pictures of beautiful women, of the year 1824. Subtracted by the beauty the spectator hardly notices the delicate numbers headed by the character for "large" - this signifying that the following numbers are the long months - at the corpus of the shamisen (the musical instrument), the woman is holding in her hands.

2 Importance of Cultural Calendars for the Semantic Web

A programming language or a query language to be used in an inter-cultural context, e.g. on the Web, for retrieving data or offering world-wide services for processing temporal data should be aware of differences between cultural calendars: for example, scheduling the phone conference of three persons in France, Greece, and Japan, the language used to calculate a possible time slot has to consider the personal and professional time constraints of the persons. Furthermore, everyone's calendar data should be better expressed in the cultural calendar and time zone the person is used to. Calendar data casting (e.g. between Japanese and Gregorian year numbering) that must be performed for calculations should be invisible to the users.

Thus, so as to make it possible for every user to express calendar data in the calendar he/she is used to, the Calendar and Time Type System CaTTS [2, 3] has been developed. CaTTS consists of two languages, a *type definition language*, CaTTS-DL, and a *constraint language*, CaTTS-CL, of a (common) parser for both languages, and of a language processor for each language (cf. Figure 1). This article exemplifies CaTTS-DL's means to specify in a rather simple manner more or less complex cultural (and also professional) calendars. With CaTTS-DL, cultural differences of

[2]Interestingly enough, this is still another of the so popular rebuses in Japanese culture: tai (the name of the fish) is seen as an abbreviation for "omedetai" (to bring blessings upon s.o.).

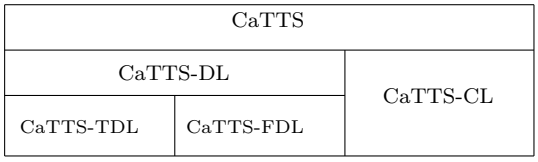

Figure 2: Sublanguages of CATTS.

```
calendar Gregorian =
 cal
   type second;
   type minute = aggregate 60 second @ second(1)...type week = aggregate 7 day @ day(1);
   type month = aggregate
     31 day named january,
     alternate month(i)
      | (i div 12) mod 4 == 0 &&
        ((i div 12) mod 400 != 100 || (i div 12) mod 400 == 0) -> 29 day
      | otherwise                                               -> 28 day
     end named february, ..., 31 day named december @ day(1);
   type year = aggregate 12 month @ month(1);
   group holiday = with select day(i) where true end;
 end
```

Figure 3: The standard Gregorian calendar in CaTTS-DL.

```
calendar_function French_Gregorian(C:Sig) =
 cal
   group holiday = with select C.day(i) where P
(* Tuesday in the carnival time: *)
    type mardi_gras for P =
      C.day(i) equals shift greg_easter(j) backward 47 C.day
(* Easter: *)
    type paques for P = C.day(i) equals greg_easter(j)
(* national day: *)
    type fete_nationale for P = relative i in C.july == 14
(* Christmas: *)
    type noel for P = relative i in C.december == 25
   end
 end
```

Figure 4: An excerpt of the French-Gregorian calendar in CaTTS-DL.

commonly used calendars can be conveniently modeled without redundantly re-specifying common aspects: calendars modeled in CaTTS-DL are composable in the sense that the language offers a means for modules. For example, the standard Gregorian calendar can be extended with a particular national calendar specifying national holidays. Furthermore, particularities like leap seconds[3], leap years, and time zones can be easily expressed in CaTTS-DL.

CaTTS-DL consists of two language fragments, CaTTS-TDL (for type definition language), to specify *calendric data types* specific to a particular calendar like "working day", "New Year's Eve", "Japanese year", or "spring", and CaTTS-FDL (for format definition language), to specify *date formats* for such data types like "Fri Jan 21 CEST 2005", "31/12/2004", "Shōwa 64-nen", or "Spring 2004".

CaTTS-DL's language processor is a *static type checker*. Static type checking (i.e. at compile time to check whether or not expressions or definitions in a program obey the typing rules of the language without any evaluation taking place) is a well-established formal method to ensure program and system behavior and to enforce high-level modularity properties [5]. CaTTS's type checker can be used for static type checking of programs or specifications in *any* language (e.g. XQuery, XSLT, XML Schema), using dates and temporal events enriched with type annotations after some calendar specified in CaTTS-DL. In particular, it is used for the static type checking of temporal constraint programs in CaTTS-CL, the constraint language of CaTTS. Oversimpled, CaTTS static type checker works as follows. Consider a calendar C specified in CaTTS-DL and a program P with type annotations referring to types and/or formats specified in the calendar C. CaTTS' static type checker verifies and/or extends the type annotations in P generating a type checked version P' of P.

CaTTS' static type checker ensures context-aware processing of calendric data types (e.g. whether Christmas Day must be interpreted as December 25 or January 7) when such data is used for programming or querying. Furthermore, CaTTS' type checker prevents from semantically

[3]Leap seconds are intercalated seconds added to keep physical time in phase with earth rotation.

```
import Julian;
calendar_function Greek_Gregorian(C:Sig) =
 cal
   macro shifted_epact(greg_y) = (14 + 11*(greg_y mod 19)) mod 30;
   macro julian_year(greg_y) = if greg_y > 0 then greg_y else greg_y - 1;
   macro julian_april_19th_ndx(jul_y) = k where day(k) during Julian.year(jul_y) &&
       relative index day(k) in Julian.april == 19;
   macro paschal_moon(greg_y) = day(k) where jul_y == julian_year(greg_y) &&
       k == julian_april_19th_ndx(jul_y) - shifted_epact(greg_y);
   group holiday = with select C.day(i) where P
(* Monday in the carnival time: *)
    type kathari_deftera for P =
       day(i) equals shift pascha(j) backward 47 C.day
(* Easter: *)
    type pascha for P = day(i) within year(y) && sunday(j) after paschal_moon(y)
                        && day(i) equals min(sunday(j))
(* Greek Independence Day: *)
    type imera_anexartisias for P = relative i in C.march == 25
(* Christmas: *)
    type christougenna for P = relative i in C.december == 25
   end
 end
```

Figure 5: An excerpt of the Greek-Gregorian calendar in CaTTS-DL.

imprecise specifications (e.g. to shift a day forward by a month). Of course, one could (probably easily) extend the language CaTTS if such form of imprecision is wished.

3 Modeling Cultural Calendars in CaTTS

This Section exemplifies the modeling of cultural calendars in France, Greece, and Japan according to a commonly used standard Gregorian calendar using CaTTS-DL.

A Standard Gregorian Calendar. The modeling of a standard Gregorian calendar independent of any cultural particularities in CaTTS-DL (cf. Figure 3) is considered in the following. A CaTTS calendar specification begins with the keyword `calendar` and ends with the keyword `end`. The calendar specification given in Figure 3 binds a calendar to the identifier `Gregorian` (i.e. the name of the calendar that can be referred to). This CaTTS-DL calendar specification consists of a set of calendar data type definitions (each identified by the keyword `type` followed by some user-defined name) and a group definition (identified by the keyword `group` followed by some user-defined name). Groups can be used to collect a set of type definitions having some intended relationship. The first type defined is `second`. It has no further properties. The CaTTS system interprets such a type definition simply as an indexed set isomorphic to the integers. If the programmer wants to relate his/her type definition for seconds to a real flow of time, he/she can use CaTTS' pre-defined type `reference. reference` is the calendar unit "second" of Unix (UTC seconds with midnight at the onset of Thursday, January 1 of year 1970 (Gregorian) as fixed point indexed by 1). The type definition of the type `minute` is derived from that of the type `second.` The CaTTS language processor interprets this type definition as an *aggregation subtype* of the type `second` such that each of its elements comprises 60 seconds[4] (denoted `aggregate 60 second`) and that the minute that has number 1, i.e. `minute(1)` comprises all seconds between `second(1)` in `second(60)` (`minute(2)` those between `second(61)` and `second(120)`, etc. (denoted `@ second(1)`). Such a type definition allows the CaTTS language processor to compute the seconds contained in a specific minute, i.e. to *cast* elements from one type to those of another. Any further type definition follows the same pattern. The definitions are straightforward following the rules of the Gregorian calendar [4]. Since Gregorian months have different lengths, a CaTTS type `month` is defined with a repeating pattern of the twelve months. The months February, which is one day longer in each Gregorian leap year is defined by an additional pattern which specifies the

[4]In CaTTS-DL, it is possible to define a type `minute` that considers leap seconds, as well.

```
calendar_function Japanese_Gregorian(C:Sig) =
 cal
  group holiday = with select C.day(i) where P
(* spring begin: *)
    type setsubun for P = relative i in C.february >= 3 &&
                          relative i in C.february <= 4
(* emperor's birthday: *)
    type showa for P = relative i in
            select C.april(i) where C.april(i) after C."1926" &&
                                    C.april(i) before C."1989"   == 29
    type heisei for P = relative i in
            select C.december(i) where C.december(i) after C."1988" == 23
  end
  group festival = with select C.day(i) where P
(* star festival: *)
    type tanabata for P = relative i in C.july == 7
    type shichigosan for P = relative i in C.november == 15
  end
 end
```

Figure 6: The Japanese-Gregorian calendar in CaTTS-DL.

leap year rule for the Gregorian calendar using the CaTTS language construct `alternate...end`. The group `holiday` just selects all days. It is further specified for each of the following cultural calendars.

Note that particularities like time zones can be easily expressed in a CaTTS-DL calendar specification. Calendar specifications of other cultural calendars in CaTTS-DL, in particular the Islamic and Hebrew calendars are given in [2, 3].

Culture-specific Calendars based on the Gregorian. A specification of the Gregorian calendar including some French holidays in CaTTS-DL is given in Figure 4[5]. In order not to re-specify the previously specified Gregorian calendar, we define a CaTTS calendar function with name `French_Gregorian` and apply this function to the standard Gregorian calendar (denoted `calendar_function French_Gregorian(C:Sig)`[6]). The resulting CaTTS-DL calendar specification contains *any* definitions of the standard Gregorian calendar (cf. Figure 3) and *re-defines* the definition of the group `holiday:` a `group` is delimited by the keywords `with` and `end`. The keyword `with` is followed by a `select...where` construct (i.e. a CaTTS' constructor to defined *inclusion subtypes*) used to select specific elements form a type (here form type `day`). The name `P` is bound in each of the type definitions within the group to the "conditions" that must be fulfilled by the elements of the types defined. The French national holiday and the Christmas Day are defined using CaTTS' relative construct (denoted `relative i in M` where `i` is the index of an element of a type defined in CaTTS-DL and `M` is a type defined in CaTTS-DL. In the case for the French national holiday, the relative construct is used to select from every July that day element which is relatively to July the 14^{th} (denoted `relative i in july == 14`). Easter is defined by a user-defined CaTTS' function (denoted `greg_easter`; not given here for space reasons). Any selected day must by a day in this function (denoted `day(i) equals greg_easter(j)`). The most important day of Carnival in France, the Tuesday which is always 47 days before Easter is defined by a backward shift of easter by 47 days. Any selected day must then equal that special Tuesday (denoted `day(i) equals shift greg_easter(j) backward 47 day`).

In the same way as we have previously defined a Gregorian calendar specific to France, we define one specific to Greece including some important Greek holidays in CaTTS-DL (cf. Figure 5). The same CaTTS language constructs are used with the difference that Easter is computed by another function defining Orthodox Easter[7].

[5]CaTTS' prototype only supports ASCII for identifiers as any other programming or modeling language.

[6]This notation `C:Sig` just allows for applying the calendar function to *any* CaTTS-DL calendar specification `C` (in the present case `Gregorian`) of type `Sig`. Calendar typing is introduced in [2].

[7]The algorithm for computing Orthodox Easter given in Figure 5 in CaTTS-DL is a standard algorithm taken from [4]. Note that a function for computing Easter in the French calendar (cf. Figure 4) can be defined similarly in CaTTS-DL

And finally, in the same way, we specify a Japanese calendar depending on the standard Gregorian calendar in CaTTS-DL (cf. Figure 6). The Japanese Gregorian calendar specifies in addition to a re-definition of the group `holiday` (of the standard Gregorian calendar in Figure 3) a further group named `festival.` The later defines some important Japanese festivals, in particular Tanabata (the star festival) and Shichigosan.[8] For the type definition in these two groups CaTTS relative construct is used again. The particularities are considered in the following: since spring begin is celebrated on February 3 and 4, two relative expressions are conjuncted (denoted `relative i in february >= 3 && relative i in february <= 4`). Another public holiday is the birthday of the emperor (tennō no tanjōbi). However, this notion naturally changes with the emperors. From 1926 until 1989, when the Shōwa tennō passed away, the 29th April was celebrated as the emperor's birthday[9], but from then on changed to December 23rd, the birthday of the new Heisei tennō. These birthdays are defined by finite types selecting those 29^{th} Aprils between 1926 an 1989 (denoted `relative i in select C.april(i) where C.april(i) after C."1926"` && `C.april(i) before C."1989" == 29`) for the era named Shōwa, and selecting those 23^{rd} Decembers which are after 1988 for the era named Heisei (denoted `relative i in select C.december(i) where C.december(i) after C."1988" == 23`). Note that `select december(i) where december(i) after "1988"` is a *local* CaTTS-DL type definition without binding an identifier to the type.

As in the different Western calendars, there are also holidays with differing dates in the Japanese calendar, e.g. the seijin no hi (the day of Coming of Age). All young people who turn twenty years old in that year are celebrated always on the second Monday in January.

Culture-specific Date Formats. Apart from calendars themselves, formats used to render dates of the types of such calendars also depend on culture, especially on common-sense use. E.g. the order of units in a date format is frequently determined by the natural language they are used in, e.g.

(US)	"November fifth, two thousand three"	→	"11/5/2003"
(F)	"Le cinq novembre deux mille trois"	→	"5/11/2003"
(D)	"Fünfter November Zweitausenddrei"	→	"5.11.2003"
(J)	"Nisensannen jūichigatsu itsuka"	→	"2003.11.5"

Roughly speaking, formats could be (and are in CaTTS-FDL) constructed from numbers and delimiters, both of which are of arbitrary representation and order. CaTTS-FDL supports definition of any such format in an elegant and intuitive manner.

In CaTTS-FDL, formats are collected in *catalogs.* Figure 7 exemplifies the definition of a catalog called `SampleCatalog`.[10] This catalog contains three different formats, namely `US`, `French`, `JapDot` and `JapRomaji`[11].

Every format is introduced by the keyword `format` immediately followed by an identifier and a type annotation, then by a format descriptor (consisting of a series of string constants and/or identifiers, the latter optionally[12] followed by <*representation*>) and finally by a list of constraints[13] relating the format to the intended value. Inside these constraints, the identifier of the format serves as a constraint variable representing the value of the format.

The formats given in figure 7 merely differ in delimiters (`/` vs. `.`) and order (m-d-y vs. d-m-y vs. y-m-d), except for `JapRomaji` which has an additional constraint for `r`. Given some `day(x)` we can render it according to `JapRomaji` by resolving the appropriate constraints, i.e. find the year

[8] "Shichi Go San" means "Seven Five Three". On Shichigosan girls of age three and seven and boys of age three and five are celebrated and it is prayed for their good health and growth.

[9] Note that each old emperor's birthday continues to be a public holiday under a different name. Shōwa tennō's birthday in today's calendar is e.g. referred to as the Green Day (midori no hi).

[10] The calendar type signature given in this specification allows for rendering and reading dates referring to *any* CaTTS-DL calendar of type `Sig` (i.e. `Gregorian` as well as any result of one the given calendar functions).

[11] We're implementing a format using rōmaji for sake of printer compatability. Using kanji would be equally possible due to Unicode.

[12] Giving no representation leads to interpreting that portion of the format a plain unsigned number.

[13] There's no difference – apart from precedence – between `&&` and a comma. As a convention, we use `&&` for conjugating constraints belonging to the same value (year, month or day number).

```
catalog SampleCatalog =
 cat
  format US:day = m "/" d "/" y where
     US within year(y),
     US within M:month && m == relative index M       in year,
                          d == relative index US in month;
  format French:day = d "/" m "/" y where
     French within year(y), ...
  format JapDot:day = y "." m "." d where ...
  format JapRomaji:day = y "nen" m "gatsu" d DayReading where
     JapRomaji within year(y),
     JapRomaji within M:month && m == relative index M          in year,
                              d == relative index JapRomaji in month,
               DayReading reads if d <= 10 || d mod 10 == 4 then "ka" else "nichi";
 end
```

Figure 7: A sample catalog of various formats in CaTTS-FDL.

number `y` the day lies within, then find the month `M` the day lies within and equate `M`'s relative index in a year to `m`, equate the day number `d` to the relative index of `day(x)` in a month, and finally choose `r` to read "(tsuitachi)", "ka" or "nichi", depending on the value of `d`.

4 Conclusion

Considering the calendars of France, Greece, and Japan this article has presented a (novel) tool, CaTTS, for modeling cultural calendars.

CaTTS is type formalism for calendars and temporal data. CaTTS provides with dedicated language constructs to conveniently model calendars and dedicated reasoning methods to efficiently process calendar data, especially static type checking. The basic principle of CaTTS' static type checker is predicate subtyping [2] according to (novel) time granularity inclusion and aggregation introduced for and with CaTTS. Predicate subtyping is used to define calendric data types representing time granularities [3], conversions between those data types, and a means to specify date formats for the elements of such data types. CaTTS-CL, CaTTS' constraint language, is statically typed after CaTTS-DL type definitions. CaTTS-CL is a language to declaratively express a wide rang range of temporal and calendric constraint problems over finite time intervals. CaTTS-DL is already prototypically implemented. A prototype implementation of a constraint solver is in development.

A future goal is to integrate CaTTS (i.e. the definition language CaTTS-DL with its static type checker and CaTTS-CL with its constraint solver) into a (Semantic) Web query language like Xcerpt [6], thus contributing to the so-called "internationalization" of the Web, i.e. making important cultural artefacts present on the Web.

References

[1] Shibuyakuritsu Shoto bijutsukan (ed.). *Mojie to emoji no keifu, Tōkiō*. 1996.

[2] François Bry, Frank-André Rieß, and Stephanie Spranger. CaTTS: Calendar Types and Constraints for Web Applications. In *Proc. 14th Int. World Wide Web Conference, Japan*, 2005.

[3] François Bry and Stephanie Spranger. Towards a Multi-calendar Temporal Type System for (Semantic) Web Query Languages. In *Proc. 2nd Int. Workshop Principles and Practice in Semantic Web Reasoning*, LNCS 3208. Springer-Verlag, 2004.

[4] Nachum Dershowitz and Edward Reingold. *Calendrical Calculations: The Millennium Edition*. Cambridge University Press, 2001.

[5] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[6] Sebastian Schaffert and François Bry. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Proc. Extreme Markup Languages, Canada*, 2004.

Acknowledgment: This research has been funded in part by the PhD Program Logics in Computer Science (GKLI) and the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Program project REWERSE number 506779 (cf. http://rewerse.net).