Implementing Solar Astronomical Calendars

Nachum Dershowitz
Edward M. Reingold

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 West Springfield Avenue
Urbana, Illinois 61801-2987, USA

July 19, 2001

In this note we describe a unified implementation of calendars whose year is based on the astronomical solar cycle—that is, on the precise solar longitude at a specified time. For example, the astronomical Persian calendar begins its new year on the day when the vernal equinox (approximately March 21) occurs before apparent noon (the middle point of the day, not clock time) and is postponed to the next day if the equinox is after apparent noon. Other calendars of this type include the French Revolutionary calendar and the future form of the Bahá'í calendar. Our approach also offers a slight simplification to the implementation of the Chinese lunisolar calendar.

Our descriptions here build on, and assume familiarity with, the collection of Common Lisp functions given in Appendix B of our book, *Calendrical Calculations*, Cambridge University Press, 1997. For each calendar considered there, a date is converted to and from a *fixed date* (or R.D.), which is a day numbering starting with January 1, 1 (Gregorian) as R.D. 1.

We begin by giving, in the next section, some general support functions; we follow that with a section describing a simple astronomical solar calendar that allows us to describe such implementations in general. Then we show how the necessary computations follow easily for the French Revolutionary, the astronomical Persian, and the future Bahá'í calendars.

Some Support Functions

First, we need to convert back and forth between local (clock) time and apparent (sundial) time expressed in *moments*, that is, days and fraction of day since R.D. 0 (and not in Julian day numbers as used for the astronomical calculations in *Calendrical Calculations*):

```
(defun local-from-apparent-moment (moment)
  ;; TYPE moment -> moment
  ;; Local time from sundial time.
  (- moment (equation-of-time (jd-from-moment moment))))

(defun apparent-from-local-moment (moment)
  ;; TYPE moment -> moment
  ;; Sundial time at local time.
  (+ moment (equation-of-time (jd-from-moment moment))))
```

The functions `equation-of-time` and `jd-from-moment` are defined in *Calendrical Calculations* and have the obvious effect.

We also need a function to tell us directly the solar longitude at a given moment (instead of at a given Julian day number, as calculated by the function `solar-longitude` given in *Calendrical Calculations*):

```
(defun solar-longitude-at-moment (moment)
  ;; TYPE moment -> angle
  ;; Longitude of sun at moment.
  (solar-longitude (jd-from-moment moment)))
```

Finally, we will find it convenient to use a Common Lisp macro to search for the smallest integer after some `initial` value for which a given `condition` holds:

```
(defmacro next (index initial condition)
  ;; TYPE (* integer (integer->boolean)) -> integer
  ;; First integer greater or equal to initial such that
  ;; condition holds.
  `(do ((,index ,initial (1+ ,index)))
       (,condition ,index)))
```

This macro is a simplified version of the macro `sum` in *Calendrical Calculations*.

Astronomical Functions

The key to implementing any astronomical solar calendar is to determine the day of the most recent new year on or before a given fixed date. We assume that the new year begins on the day when the solar longitude was at least `l` at a *critical moment* which is defined for every fixed date `d` by a function `critical-moment`. The longitude `l` need not actually be that of the new year; the following macro simply finds the fixed date at the critical moment of which the longitude reaches `l`:

```
(defmacro solar-new-year-on-or-before (date l d critical-moment)
  ;; TYPE fixed-date angle * (fixed-date -> moment) -> fixed-date
  ;; Last fixed date on or before date when solar longitude
  ;; was at least l at critical-moment (UT) of that date.
  (let* ((theta (gensym))
         (start (gensym)))
    `(let* ((,d ,date)
            (,theta ;; Longitude at critical moment on given date.
             (solar-longitude-at-moment ,critical-moment))
            (,start ;; Date before prior occurrence of l
             (- ,date 5
                ;; Approximate number of days since longitude was l.
                (floor (* mean-tropical-year 1/360
                          (degrees (- ,theta ,l)))))))
       (next ,d ,start (<= ,l
                           (solar-longitude-at-moment ,critical-moment)
                           (+ 2 ,l))))))
```

The idea of this macro is as follows. First we determine the solar longitude `theta` at the critical moment of the given fixed `date`. We use this value to tell us approximately how far back we must go before `date` to reach the desired longitude `l`; for safety (since the apparent speed of the sun varies) we go back an additional five days. (The function `degrees` normalizes an angle to the range 0..360.) Then, we search forward with `next`, day by day, from the underestimate `start`, until we find the date on which the solar longitude has reached `l` by the critical moment. The variable `d` is just the formal parameter of the function `critical-moment`.

Given an epoch for a solar astronomical calendar (that is, the fixed date of the first day of the first year of that calendar), we want to be able to determine the first fixed date `y` years after the epoch with solar longitude at least `l` at the critical moment of that date. We assume that the integral longitude preceding the critical moment of the epoch day is the longitude that determines the new year. We do this as follows:

```
(defmacro solar-event (epoch y l d critical-moment)
  ;; TYPE fixed-date integer angle * (fixed-date -> moment)
  ;; -> fixed-date
  ;; First fixed date y solar years after epoch with solar
  ;; longitude at least l at critical-moment of that date.
  ;; It is assumed that the integral longitude preceding the
  ;; critical-moment of the epoch day is the longitude that
  ;; determines the new year.
  (let* ((epochal-longitude (gensym)))
    `(let* ((,d ,epoch)
            (,epochal-longitude
             (floor (solar-longitude-at-moment
                     ,critical-moment))))
       (solar-new-year-on-or-before
        (floor (+ ,epoch 10
                  (* mean-tropical-year ,y)
                  (* mean-tropical-year 1/360
                     (degrees (- ,l ,epochal-longitude)))))
        ,l ,d ,critical-moment))))
```

We find the epochal longitude (the integral solar longitude during the day of the epoch) and compare it to the desired longitude `l`. This tells us approximately what fraction of a year past the new year is needed for the sun to return to longitude `l`. Adding `y` mean tropical years and that fractional year to the epoch, plus ten days as a safety factor, gives a fixed date following the desired solar event. The macro `solar-new-year-on-or-before`, given that date, then finds the preceding fixed date when the solar longitude already reached `l` at its `critical-moment`.

A Simple Example: The Urbana Astrological Calendar

In this section we describe a simple, hypothetical solar astronomical calendar that we call the "Urbana Astrological Calendar." We begin with a constructor for dates on this calendar; each date is a triple:

```
(defun urbana-astrological-date (month day year)
  ;; TYPE (urbana-astrological-month urbana-astrological-day
  ;; TYPE  urbana-astrological-year) -> urbana-astrological-date
  (list month day year))
```

The new year on this calendar begins on the day after the winter solstice in Urbana, Illinois (we use the function `urbana-winter` from *Calendrical Calculations*). The epoch is this event in Gregorian year 2000, R.D. 730,476 = December 22, 2000 (Gregorian), which is the start of year 0 on this calendar:

```
(defconstant urbana-astrological-epoch
  ;; TYPE fixed-date
  ;; Winter solstice in the year 2000.
  (ceiling (urbana-winter 2000)))
```

The calendar has twelve months, each a true solar month of 30° solar longitude, corresponding to a zodiacal constellation. Each month comprises between 29 and 32 days, depending on the precise time for the sun to move through the required 30°. The critical moment is midnight standard time at the start of the day in question. The following are used to convert local midnight in Urbana (360 minutes behind Greenwich) to U.T.:

```
(defconstant urbana-time-zone
  ;; TYPE minute
```

```
  ;; The difference (in minutes) of Central time zone
  ;; from Universal Time.
  -360)

(defun midnight-in-urbana (date)
  ;; TYPE fixed-date -> moment
  ;; Moment in U.T. at midnight in Urbana.
  (universal-from-local date urbana-time-zone))
```

The function `universal-from-local` converts local time to U.T. time.

To determine a fixed date from an Urbana Astrological date we use the year of the astrological date to determine the number of elapsed solar years; we use the month of the astrological date to determine the solar longitude at the start of that month. With these two values, the macro `solar-event` tells us the fixed date of occurrence of entry into that constellation in Urbana, to which we add the day of the astrological date minus one:

```
(defun fixed-from-urbana-astrological (u-date)
  ;; TYPE urbana-astrological-date -> fixed-date
  ;; Fixed date of Urbana Astrological date.
  (let* ((month (standard-month u-date))
         (day (standard-day u-date))
         (year (standard-year u-date)))
    (+ day -1
       (solar-event urbana-astrological-epoch year
                    (degrees (+ 270 (* (1- month) 30)))
                    x (midnight-in-urbana x)))))
```

To invert the process and convert a fixed date to an Urbana Astrological date, we find the previous new year (day after the solstice in Urbana) and use that to find the year number by dividing the number of elapsed days since the epoch by the mean length of a year, compute the month number from the solar longitude on that fixed date (counting one month for each 30° past the longitude of 270° of the solstice), and compute the day of the month by subtracting the given fixed date from the start of the month:

```
(defun urbana-astrological-from-fixed (date)
  ;; TYPE fixed-date -> urbana-astrological-date
  ;; Urbana Astrological (month day year) corresponding to
  ;; fixed date.
  (let* ((new-year (solar-new-year-on-or-before
                    date 270
                    x (midnight-in-urbana x)))
         (year (round (/ (- new-year urbana-astrological-epoch)
                         mean-tropical-year)))
         (month (1+ (mod (quotient (- (solar-longitude-at-moment
                                        (midnight-in-urbana date))
                                       270)
                                    30)
                         12)))
         (day (- date -1
                 (fixed-from-urbana-astrological
                  (urbana-astrological-date month 1 year)))))
    (urbana-astrological-date month day year)))
```

For example, this function tells us that November 12, 1945 (Gregorian) = R.D. 710,347 is day 22 of month 11 of year −56 on the Urbana Astrological calendar.

The French Revolutionary Calendar

The (original) French Revolutionary calendar had its new year begin on the day of the autumnal equinox (solar longitude = 180°) in Paris using apparent (sundial) time. Thus,

```
(defun midnight-in-paris (date)
  ;; TYPE fixed-date -> moment
  ;; Moment in U.T. when it's next apparent midnight in Paris.
  (local-from-apparent-moment
   (universal-from-local (1+ date) french-time-zone)))

(defun french-new-year (f-year)
  ;; TYPE french-year -> fixed-date
  ;; Fixed date of French Revolutionary new year f-year.
  (solar-event french-epoch (1- f-year) 180
             x (midnight-in-paris x)))
```

gives the fixed date of the first day of year `f-year` of the Revolution.

The Astronomical Persian Solar Calendar

The year of the astronomical Persian calendar begins on the day when the vernal equinox occurs in Teheran before noon (sundial or apparent time—the middle point of daylight) and is postponed to the next day if the equinox is after noon. Since Teheran is 3.5 hours ahead of U.T., we define

```
(defconstant teheran-time-zone
  ;; TYPE minute
  ;; The difference (in minutes) of the Teheran time zone
  ;; from Universal Time.
  210)

(defun noon-in-teheran (date)
  ;; TYPE fixed-date -> moment
  ;; Moment in U.T. when it's apparent noon in Teheran.
  (local-from-apparent-moment
   (universal-from-local (+ 1/2 date) teheran-time-zone)))
```

The fixed date of the new year is found by

```
(defun astronomical-persian-new-year (p-year)
  ;; TYPE persian-year -> fixed-date
  ;; Fixed date of astronomical Persian new year p-year.
  (solar-event persian-epoch
             (if (< p-year 0)
                 p-year  ;; There's is no year 0.
               (1- p-year))
             0 x (noon-in-teheran x)))
```

Using the above function, we can compare the dates of the Persian new year on the astronomical calendar to those of arithmetical calendar implemented in *Calendrical Calculations*. For 1000–1800 A.P., we find they disagree on the 28 years shown in Table 1. Outside of this range disagreement is far more common, occurring almost every fourth year.

Persian Year	Astronomical New Year R.D. Gregorian	Arithmetical New Year R.D. Gregorian
1016	597,616 = March 20, 1637	597,617 = March 21, 1637
1049	609,669 = March 20, 1670	609,670 = March 21, 1670
1082	621,722 = March 21, 1703	621,723 = March 22, 1703
1111	632,314 = March 20, 1732	632,315 = March 21, 1732
1115	633,775 = March 20, 1736	633,776 = March 21, 1736
1144	644,367 = March 20, 1765	644,368 = March 21, 1765
1177	656,420 = March 20, 1798	656,421 = March 21, 1798
1210	668,473 = March 21, 1831	668,474 = March 22, 1831
1243	680,526 = March 20, 1864	680,527 = March 21, 1864
1404	739,331 = March 21, 2025	739,330 = March 20, 2025
1437	751,384 = March 21, 2058	751,383 = March 20, 2058
1470	763,437 = March 21, 2091	763,436 = March 20, 2091
1532	786,082 = March 21, 2153	786,081 = March 20, 2153
1565	798,135 = March 21, 2186	798,134 = March 20, 2186
1569	799,596 = March 21, 2190	799,595 = March 20, 2190
1598	810,188 = March 22, 2219	810,187 = March 21, 2219
1631	822,241 = March 21, 2252	822,240 = March 20, 2252
1660	832,833 = March 21, 2281	832,832 = March 20, 2281
1664	834,294 = March 21, 2285	834,293 = March 20, 2285
1693	844,886 = March 22, 2314	844,885 = March 21, 2314
1697	846,347 = March 22, 2318	846,346 = March 21, 2318
1726	856,939 = March 22, 2347	856,938 = March 21, 2347
1730	858,400 = March 22, 2351	858,399 = March 21, 2351
1759	868,992 = March 21, 2380	868,991 = March 20, 2380
1763	870,453 = March 21, 2384	870,452 = March 20, 2384
1788	879,584 = March 21, 2409	879,583 = March 20, 2409
1792	881,045 = March 21, 2413	881,044 = March 20, 2413
1796	882,506 = March 21, 2417	882,505 = March 20, 2417

Table 1: Years in the range 1000–1800 A.P. on which the astronomical Persian calendar differs from the arithmetical Persian calendar implemented in *Calendrical Calculations*.

The Future Bahá'í Calendar

The Bahá'í year was intended by the official rules to begin at sunset following the vernal equinox. Each day begins at sunset the prior evening. The computation of the time of sunset depends on the latitude and longitude of the location on earth. Using sunset in Haifa, Israel (the exact location has yet to be determined), as the critical moment, we get:

```
(defun sunset-in-haifa (date)
  ;; TYPE fixed-date -> moment
  ;; Moment in U.T. at prior sunset in Haifa.
  (let* ((latitude 32.82)
         (longitude 35.98)
         (offset (* 4 longitude)))
    (universal-from-local
     (+ date -1 (sunset (1- date) latitude longitude))
     offset)))
```

where `sunset`, as given in *Calendrical Calculations*, returns the local mean time of sunset.

The epoch for the calendar is the fixed date of this event in 1844, R.D. 673,221 = March 20, 1844 (Gregorian), which we specify using the conversion functions for the Gregorian calendar given in *Calendrical Calculations*:

```
(defconstant future-bahai-epoch
  ;; TYPE fixed-date
  ;; Fixed date of start of astronomical Bahai calendar.
  (solar-new-year-on-or-before
   (fixed-from-gregorian (gregorian-date april 1 1844))
   0 x (sunset-in-haifa x)))
```

It is now a simple matter to compute:

```
(defun future-bahai-new-year (b-year)
  ;; TYPE bahai-year -> fixed-date
  ;; Fixed date of future Bahai calendar new year b-year.
  (solar-event future-bahai-epoch (1- b-year) 0
               x (sunset-in-haifa x)))
```