FV Time

Ana Borges, Quim Casals, Juan Conejero,
Mireia González, Eduardo Hermo and Joost J. Joosten

Summary

The FV Time Library is a Coq-verified implementation of the UTC standard and some utilities to make it more usable for both critical and regular programming. While most time softwares don't include leap seconds, FV Time does.

The FV Time Manager is a standalone executable available for Linux and Windows, allowing the user to interact with the Time Library from the command line.

Leap Seconds

The Earth's rotation period varies, slowing down unpredictably, but the time standard defines days as 86400 SI seconds (measured with atomic clocks). This standard would gradually differ from solar time. This may be irrelevant for some purposes, but when civil time is somehow involved this can present a problem: without adjustments, noon in atomic time could even happen during the night (in the sense of period with no solar light). This problem is the motivation for defining leap seconds.

The UTC standard states that days always have 24 hours, every hour has 60 minutes, but minutes can have between 59 and 61 seconds. When a minute of 61 seconds occurs, the extra second is called a leap second. When UTC differs in 0.9 seconds of UT1, a leap second is inserted. The leap second is inserted at second 23:59:60 of a chosen UTC date. The last day of December and June are preferred, and then the last day of March or September, and the last day of any other month as last preference. All leap seconds as of 2022 have been scheduled for either June 30 or December 31. Up to April 2022 there hasn't been any negative leap second.

Features

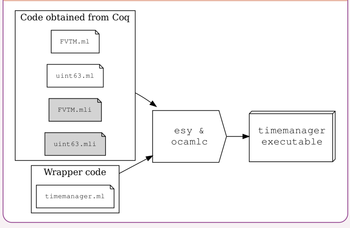
The main goal of the Time Library is to provide verified functions translating between UTC times (with leap seconds) and timestamps.

The `calendar.v` file describes the specifications of our library, using MathComp's `nat`. Since the specifications are optimized to be intuitive and not efficient, we provide a second file, `Hinnant.v`, with alternative implementations. The implementation of the datestamp algorithm and its inverse are inspired by the ones described by Howard Hinnant, hence the name of the file.

To achieve even better performance all functions were refined to Coq's integers primitive types, proving that over the range of primitives the same lemmas were valid.

Adding and subtracting durations or time intervals doesn't respect basic arithmetical properties, due to the irregular periods that Gregorian calendar and UTC define. This problem affects all the components to the left of seconds. To solve this problem we implemented two different types of operations in time arithmetic. The first ones, shift functions, are the conventional operation over components. The second ones form a new standard of operations called add-formal. They behave consistently with basic arithmetical properties.

Compilation



Comparison to DateTime

Comparison of FV Time Library to Microsoft's Datetime

Microsoft constructors and projections	FV update to UTC	FV new algorithm
DateTime (string)	mkDate	-
DateTime (Int32)	from_utc_timestamp	-
DateTime	-	date_of_time
Day	day	-
DayOfWeek	weekday_of_date	-
DayOfYear	-	-
Hour	hour	-
Minute	minute	-
Month	month	-
Now	-	-
Second	second	-
TimeOfDay	-	clock_of_time
Today	-	-
Year	year	-
-	-	max_second
ToTimeReal	utc_timestamp	-
ToString	-	-
Add	add_formal	-
AddDays	shift_utc_days	add_formal_days
AddHours	shift_utc_hours	add_formal_hours
AddMinutes	shift_utc_minutes	add_formal_minutes
AddSeconds	shift_utc_seconds	add_formal_seconds
AddYears	shift_utc_years	add_formal_years
Compare	le_time, lt_time	-
DaysInMonth	days_of_month	-
IsLeapYear	is_leap_year	-
Substract (DateTime)	-	time_difference
Substract (TimeSpan)	-	substract_formal

Hinnant's algorithms

The Gregorian calendar has a 400 year cyclic structure, the algorithms base their calculations on this structure, ruling out the actual year from the core computations. This helps with the addition of leap days, preceding March 1st.

We call each cycle of 400 years an era. The first era starting the 1st of March of year 0.

However, following the Time Unix tradition, the seconds and days count begin at 1970-01-01 and end 9999-12-31.

References

Hinnant's Algorithms:
`https://howardhinnant.github.io/date_algorithms.html`

Proyecto RTC-2017-6740-7 financiado por MCIN/AEI/10.13039/501100011033 y por FEDER.