M. Kuhn
University of Cambridge

Coordinated Universal Time with Smoothed Leap Seconds (UTC-SLS)

Status of this Memo

Abstract

Coordinated Universal Time (UTC) is the international standard timescale used in many Internet protocols. UTC features occasional single-second adjustments, known as "leap seconds". These happen at the end of announced UTC days, in the form of either an extra second 23:59:60 or a missing second 23:59:59. Both events need special consideration in UTC-synchronized systems that represent time as a scalar value. This specification defines UTC-SLS, a minor variation of UTC that lacks leap seconds. Instead, UTC-SLS performs an equivalent "smooth" adjustment, during which the rate of the clock temporarily changes by 0.1% for 1000 seconds. UTC-SLS is a drop-in replacement for UTC. UTC-SLS can be generated from the same information as UTC. It can be used with any specification that refers to UTC but lacks provisions for leap seconds. UTC-SLS provides a robust and interoperable way for networked UTC-

synchronized clocks to handle leap seconds. By providing UTC-SLS instead of UTC to applications, operating systems can free most application and protocol designers from any need to even know about UTC leap seconds.

1 Introduction

Coordinated Universal Time (UTC) is an internationally agreed precise definition of time. Like its predecessor Greenwich Mean Time (GMT), UTC approximates the local mean solar time on Earth's prime meridian, which passes through Greenwich in London, England.

UTC is commonly available through radio time signals [ITU-768], global navigation satellite systems, and the Internet [RFC1305], with an accuracy ranging from a few milliseconds to a few nanoseconds. It is the basis of official time in many countries, where local time is legally defined to differ from UTC exactly by an integral number of half hours. UTC is commonly referred to in the specifications of computer applications, communication protocols [ISO8601, RFC3339] and application-program interfaces [C, POSIX].

UTC is defined and maintained by an internationally coordinated network of precision clocks ("atomic clocks"). The reference frequency they generate is far more stable than Earth's daily rotation, on which the traditional astronomical definitions of time are based. In the interest of backwards compatibility, UTC was defined such that an adjustment is made before UTC and a particular definition of astronomical time, known as UT1 [McC91], drift apart more than 0.9 seconds [ITU-460]. These adjustments have, in their current form, been applied since 1972 and are known as "leap seconds" [Nel01, IERS-C].

The definition of UTC [ITU-460] provides for two types of adjustment:

- The "insertion of a second" or "positive leap second" is the addition of a 86401-st second denoted "23:59:60" at the end of a UTC day.

- The "deletion of a second" or "negative leap second" is the skipping of the last, 86400-th second denoted "23:59:59" at the end of a UTC day.

Time is traditionally represented as a vector of integer numbers, denoting year, month, day, hour, minute, second, and some fraction of a second, for example written as YYYY-MM-DD hh:mm:ss.sss [ISO8601, RFC3339]. A normal UTC day is a progression of 86400 seconds, which are numbered 00:00:00 to 23:59:59.

When the rotation of Earth has fallen behind UTC, the insertion of a second into UTC (positive leap second) will be announced in [IERS-C]. On the announced date, the UTC day will be 86401 seconds long and a UTC clock will display at its end the sequence of seconds

```
..., 23:59:57, 23:59:58, 23:59:59, 23:59:60, 00:00:00, 00:00:01, ...
```

Likewise, should the rotation of Earth rush ahead of UTC, the deletion of a leap second will be announced, and on that date, the UTC day will be only 86399 seconds long and will end with the sequence of seconds

```
..., 23:59:57, 23:59:58, 00:00:00, 00:00:01, ...
```

Time distribution services, such as NTP [RFC1305], provide announcements of forthcoming leap seconds. These permit clocks synchronized to such a time service to display leap seconds exactly as defined in [ITU-460, IERS-C] and shown above.

Leap seconds were introduced because they provide the most convenient way of adjusting UTC for some applications, in particular those that distribute time via pulse-per-second signals. By inserting or deleting an entire second, such signals are not disturbed, and neither are any of the precision frequencies that are generated from them by frequency multiplication, including carrier frequencies of radio transmitters, television signal timings, etc.

In other applications, however, leap seconds are less convenient. Unlike humans, computers deal with time more easily as a single scalar value, rather than as a vector of integers.

A typical and prominent example of a scalar encoding of time is the "seconds since the epoch" timescale. It is defined in [POSIX] as a scalar encoding of a YYYY-MM-DD hh:mm:ss.sss value shown on a clock that approximates UTC. If UTC had no leap seconds, this value would represent the number of seconds that have elapsed since the start of the UTC day 1970-01-01.

If a clock is closely synchronized with UTC and receives leap-second announcements in advance, then [ITU-460] defines only what happens to an hh:mm:ss display near that leap second. However, if such a clock is required to output a scalar time, its implementor and users will face two problems. Firstly, typical scalar encodings of time have no unambiguous representation for points in time during a positive leap second. Secondly, both positive and negative leap seconds will cause a discontinuity in the scalar representation of time that may be disruptive for some applications.

[ITU-460] provides no guidance for handling scalar representations of time across leap seconds. [POSIX] leaves the exact behavior of its "seconds since the epoch" timescale implementation-defined. Most other protocol and API specifications remain equally silent on the issue.

This specification fills that gap by providing clock implementors with guidance on what a UTC-synchronized computer clock should output near a leap second. It specifies the exact behavior of a clock that displays a variant of UTC called UTC-SLS (UTC with Smoothed Leap Seconds).

2 Application of UTC-SLS

UTC-SLS is intended as a drop-in replacement for UTC in any application protocol interface or communication protocol that does not explicitely specify any particular behavior near a leap second.

UTC-SLS avoids the problems that arise when the UTC clock defined in [ITU-460] is converted into a scalar representation of time. It can be used by implementors of UTC-synchronized clocks with scalar output in order to achieve interoperable behavior with a low risk of disruption in the presence of leap seconds.

UTC-SLS should be used in computer applications and protocols independent of whether they represent time as a scalar value or in hh:mm:ss notation. Even though the hh:mm:ss representation is, in principle, capable of encoding inserted UTC leap seconds unambiguously, using the 23:59:60 notation from [ITU-460], in practice this is not feasible, because hh:mm:ss and scalar representations have to be converted into each other frequently.

If UTC-synchronized computer clocks provide their users routinely with an approximation of UTC-SLS instead of an approximation of UTC, then it can be hoped that far fewer specification authors and software developers will have to be aware of leap seconds. Leap seconds can remain a concern of the time-keeping specialists that implement the clock drivers in operating systems and the protocols used to synchronize these with external time signals.

UTC-SLS is not intended to be used as a drop-in replacement in specifications that are themselves concerned with the accurate synchronization of clocks and that have already an exactly specified behavior near UTC leap seconds (e.g., NTP [RFC1305], PTP [IEC1588]).

Some "real time" applications have very demanding clock-stability requirements, for which neither UTC nor UTC-SLS are suitable. Appendix B discusses application limits of UTC-SLS and alternatives.

3 Properties of UTC-SLS

On a UTC-SLS clock

- the time of day always progresses through 86400 different hh:mm:ss values, and always ends with ..., 23:59:58, 23:59:59, followed by 00:00:00, 00:00:01, ...;

- the time 23:59:60 never appears;

- the time never differs from UTC by more than one second;

- the time always equals UTC at full or half hours;

- the rate can deviate by exactly +/- 0.1% (+/- 1000 ppm).

4 Definition of UTC-SLS

A UTC-SLS clock shows the exact same time as a UTC clock, except during the last 1000 seconds of any UTC day that is extended or shortened through the insertion or deletion of one leap second at the end of that day.

4.1 Positive leap second

Whenever a UTC day is extended by an inserted second, during this positive leap second, values of the form 23:59:60.xxxx are displayed on a UTC clock, but no such timestamps appear on a UTC-SLS clock. Instead, during the last 1000 seconds of that UTC day, starting at 23:43:21, the UTC-SLS clock slows down to 0.999 times its normal rate, which means that the UTC-SLS clock progresses during that time only through 999 "slow seconds", each of which lasts 1000/999 seconds.

The following table shows the exact and simultaneous display of a UTC and UTC-SLS clock at various points in time near an inserted leap second:

UTC	UTC-SLS	Remarks
23:43:20.0000	23:43:20.0000	
23:43:21.0000	23:43:21.0000	UTC-SLS starts to diverge from UTC
23:43:22.0000	23:43:21.9990	UTC-SLS is now 1 ms behind UTC
23:43:23.0000	23:43:22.9980	UTC-SLS is now 2 ms behind UTC
23:43:24.0000	23:43:23.9970	UTC-SLS is now 3 ms behind UTC
... 995 seconds later ...		
23:59:59.0000	23:59:58.0020	UTC-SLS is now 998 ms behind UTC
23:59:60.0000	23:59:59.0010	UTC leap second starts

```
  00:00:00.0000   00:00:00.0000   UTC leap second ends, UTC-SLS = UTC
  00:00:01.0000   00:00:01.0000
```

Intermediate UTC-SLS values are defined through linear interpolation accordingly, for example:

```
        UTC           UTC-SLS

  23:43:21.1000   23:43:21.0999
  23:43:21.2000   23:43:21.1998
  ...
  23:59:60.9000   23:59:59.9001
```

4.2 Negative leap second

Whenever a UTC day is shortened by deleting a leap second, no values of the form 23:59:59.xxxx are displayed on a UTC clock, but all these timestamps nevertheless appear on a UTC-SLS clock. Instead, during the last 1000 seconds of that UTC day, starting at 23:43:19, the UTC-SLS clock accelerates to 1.001 times its normal rate, which means that the UTC-SLS clock progresses during that time through 1001 "fast seconds", each of which lasts 1000/1001 seconds.

The following table shows the exact and simultaneous display of a UTC and UTC-SLS clock at various points in time near a deleted leap second:

```
        UTC           UTC-SLS

  23:43:18.0000   23:43:18.0000
  23:43:19.0000   23:43:19.0000   UTC-SLS starts to diverge from UTC
  23:43:20.0000   23:43:20.0010   UTC-SLS is now 1 ms ahead of UTC
  23:43:21.0000   23:43:21.0020   UTC-SLS is now 2 ms ahead of UTC
  23:43:22.0000   23:43:22.0030   UTC-SLS is now 3 ms ahead of UTC
  ... 995 seconds later ...
  23:59:57.0000   23:59:57.9980   UTC-SLS is now 998 ms ahead of UTC
  23:59:58.0000   23:59:58.9990   UTC-SLS is now 999 ms ahead of UTC
  00:00:00.0000   00:00:00.0000   leap second skipped, UTC-SLS = UTC
  00:00:01.0000   00:00:01.0000
```

Intermediate values are again defined through linear interpolation, for example:

```
        UTC           UTC-SLS

  23:43:19.1000   23:43:19.1001
  23:43:19.2000   23:43:19.2002
  ...
```

23:59:58.9000 23:59:59.8999

5 Conversion between UTC and UTC-SLS

UTC and UTC-SLS clock displays (of the form hh:mm:ss.sss...) can unambiguously be converted into each other, as long as one knows whether there will be a leap second within 1000 seconds after the time to be converted, and if so, what its sign is.

For a given time of day written as hh:mm:ss (hh < 24 and mm < 60), let "since midnight" denote the value hh * 3600 s + mm * 60 s + ss * 1 s. Let U denote the "since midnight" value of a UTC clock display and let US denote the corresponding "since midnight" value of a UTC-SLS clock display. Let

```
L = +1 s   if the UTC day ends with an inserted leap second,
L = -1 s   if the UTC day ends with a deleted leap second, and
L =  0 s   otherwise.
```

Further, let

```
B = 86400 s + L - I
```

where I = 1000 s is the length of the smoothing interval.

If U < B, then U = US, otherwise

```
US = U - L * (U - B) / I
```

and

```
U  = B + (US - B) / (1 - L / I).
```

6 Security Considerations

Leap seconds are only one of several likely reasons due to which an application may experience disruptions in the operation of a UTC-synchronized clock. An important other one is the temporary loss of synchronization with UTC, for example due to interrupted communication channels or an operator error, and the need for subsequent resynchronization with UTC. Most security-sensitive systems cannot afford to rely on an assumption that their clock is always synchronized to UTC with less than +/- 1000 ms error, or that the rate of their clock does not deviate by up to +/- 0.1% when measured over intervals less than 1000 seconds long. It is, therefore, unlikely that the use of UTC-SLS is going to introduce any new security hazards into an application that would not exist without UTC-SLS and that are not due to unrealistic expectations in the

performance of any computer-implemented UTC-synchronized clock.

APPENDIX A: Rationale

This section lists some of the options considered during the development of this specification, and indicates the reasons for the particular design chosen for UTC-SLS.

Functions that read a UTC-synchronized clock and return the current time as a scalar value could behave in a number of different ways near UTC leap seconds. If the scalar timescale allocates an interval of 86400 seconds for each UTC day and the goal is not to deviate in any way from UTC outside a leap second, possible behaviors during inserted UTC leap second include:

1a) The scalar value could jump backwards in time by one second at the start of an inserted leap second (so, for example, both 23:59:59.1 and 23:59:60.1 will have the same scalar representation, and the last second of the day repeats).

1b) The scalar value could jump backwards in time by one second at the end of an inserted leap second (so, for example, both 23:59:60.1 and 00:00:00.1 will have the same scalar representation, and the first second of the next day repeats).

1c) The function could return, in addition, a leap-second-in-progress bit, to disambiguate the scalar value.

1d) Where a separate second and subsecond value is returned, an unambiguous out-of-range subsecond component could be returned, until the inserted leap second is over (for example, a nanosecond count could overflow in the range 1000000000 to 1999999999).

1e) The function could delay its reading of the clock and not return until the inserted leap second is over.

1f) The clock could stop for 1 s right before reaching 00:00:00.

1g) The operating system could suspend all processes during an inserted leap second.

1h) The function could abort with an error code when called during an inserted leap second.

Deleted leap seconds leave less room for variety and a requirement that the returned value must not deviate in any way from UTC outside a leap second could only be achieved in one way:

1i) The scalar value jumps forward in time by one second when the deleted leap second 23:59:59 is reached, directly to 00:00:00.

Other options include:

2a) The scalar timescale could be defined by allocating 86401 seconds for each UTC day. This would provide an unambiguous scalar representation of an inserted leap second 23:59:60 at the end of each day.

2b) The scalar timescale could be defined by allocating the actual number of seconds to each day, this way representing a count of real seconds, rather than being a fixed encoding of a YYYY-MM-DD hh:mm:ss UTC clock reading. The conversion between such a scalar time and a vector representation of UTC would then dependent on the availability of a table of all leap seconds since the origin of the scale ("epoch").

2c) The clock could continue without any adjustment across a leap second, delaying the leap until the system is restarted.

2d) The scalar timescale could be defined by allocating 86400 seconds for each UTC day, but the rate at which the clock ticks through the seconds could be changed temporarily.

Options 1a) to 1i) and also 2a) all lead to discontinuities in the time scale. If an application measures the time interval between two events by subtracting two of the returned values, then with any of these options, the relative error made has no upper bound.

The problem with option 2b) is that the mapping between scalar time values and integer vectors representing the display of a UTC clock is no longer fixed. This mapping changes each time a new leap second is announced and is not predictable more than a few months into the future. So while option 2b) may be very attractive for applications with high accuracy requirements for time-interval measurements (e.g., navigating a spacecraft), it is not well suited for applications that rely on a stable future relationship with UTC (e.g., an office calendar).

Option 2c) avoids clock discontinuities while processes are running on the local system. However, it may hinder interoperability with systems that restart at different times.

While [C] permits all of the above approaches, [POSIX] explicitely forbids both 2a) and 2b) for its "seconds since the epoch" timescale.

This leaves option 2d), the solution chosen in this specification, as

the one that is easiest to manage and least likely to cause disruption for a large number of applications. The form of the required rate correction can be chosen in a number of ways, for example:

3a) The clock rate switches between three values: normal, slow, fast.

3b) The clock rate is ramped up linearly from its normal rate to a peak deviation, and then ramped back again to normal.

3c) The difference in offset of the timescale before and after the leap second is fed into the same control loop that keeps the offset and rate of the clock aligned with an external reference. The one-second step will be processed by the loop's low-pass filters, whose output gradually adjusts the rate and offset of the clock for a smooth transition.

Option 3a) was chosen for UTC-SLS, because it is the easiest of these alternatives to describe, understand, implement and verify. With it, the mapping between a fixed-frequency counter value and a scalar representation of UTC-SLS is a continuous function that is piece-wise defined through affine functions (polynomials of degree one).

With option 3b), that mapping would become a continuously differentiable function that is defined piece-wise through polynomials of degree two. This increase in conceptual and computational complexity would have the benefit of limiting the rate at which the rate of the clock changes. There may be some specialist applications that could benefit from such a limit, in particular those where a control loop is used to steer the motion of a large mass (e.g., a real or simulated "fly wheel") in relation to a UTC-synchronized clock. Besides the question whether UTC-based time scales are appropriate at all for such applications, given that each would have its own particular engineering constraints, it seems more appropriate to use a special-purpose timescale for each, rather than attempt to try and accommodate them all in a general-purpose solution like UTC-SLS.

Option 3c) may seem easiest to implement with an already existing control loop. However, there are a large number of design choices and parameters with such control loops, which designers may want to optimize based on the properties of their particular clock hardware and communication channel. Therefore, standardizing any particular one control loop for the purpose of smoothing leap seconds would either place a severe restriction on the designer of a control loop that is primarily needed to track the reference clock, or it would lead to the implementation of separate control loops, defeating the only advantage of option 3c).

Having decided to simply switch from the clock's normal rate to a single faster or slower smoothing rate near a leap second, two more choices need to be made.

The time interval during which the clock runs at the smoothing rate could be placed

4a) entirely after the leap second;

4b) entirely before the leap second;

4c) centered around the leap second.

Option 4c) would have the advantage that the maximum deviation between UTC and UTC-SLS is limited to only half a second. However, this maximum deviation would be reached at midnight, which is a time commonly used to schedule events and deadlines. Both options 4a) and 4b) lead to a maximum deviation between UTC and UTC-SLS of one second, but with them, UTC and UTC-SLS are identical at midnight. For this reason, option 4c) was not chosen.

Many time-signal providers transmit a leap-second announcement during some time before the leap second, and stop doing so as soon as the leap second is over. With option 4a), a UTC time-signal receiver that is switched on directly after a leap second would not be able to learn about the leap second that has just happened, and would, therefore, not be able to apply the correction necessary to convert UTC into UTC-SLS. With option 4b), the time during which UTC and UTC-SLS differ and the time during which leap-second announcements are usually transmitted overlaps, ensuring that a newly activated UTC receiver can very quickly synchronize with UTC-SLS.

The final parameter to be chosen is the length I of the smoothing interval, which also defines the factor 1 - L/I by which the clock rate changes. The chosen value I = 1000 s (or 16 minutes and 40 seconds) fulfills the following requirements:

5a) The resulting maximal rate error of L/I should be well below any human perception limit for changes in length, duration, rate, rhythm, or pitch, to make it unlikely that anyone will directly perceive the rate change with unaided senses.

5b) The length of I should be below half an hour (I < 1800 s), such that UTC and UTC-SLS are identical at every half and full hour in every time zone that differs from UTC by an integral number of full or half hours. This way, many exact deadlines remain unaffected by the difference between UTC and UTC-SLS.

5c) The length I should be less than 59 minutes, which is the advance warning time given by some existing time services for an upcoming leap second (e.g., the German DCF77 transmitter).

5d) Choosing a power of 10 times one second for the value of I ensures that the conversion between UTC and UTC-SLS is easy to understand and perform when both times are displayed as decimal numbers.

The choice of I = 1000 s is the largest value that fulfills all of the above criteria.

APPENDIX B: Application Limits

Where applications use a UTC-SLS clock to measure the time interval between two events, and do not correct for the variable rate, the maximum possible relative error due to leap seconds is 0.1%. This maximum error can only be reached for intervals of 1000 s or shorter, and decreases for longer intervals.

The vast majority of computer applications have far less exacting requirements for time-interval measurements and can, therefore, use UTC-SLS without any concern for leap seconds.

Some multimedia standards specify stricter clock-stability requirements, which cannot be met within the constraints listed in Appendix A. For example, [MPEG] defines a 27 MHz system clock frequency with a maximum permitted frequency error of 0.003% (30 ppm, parts per million) and a maximum permitted rate change with time of 75 millihertz per second or 0.002777 ppm/s.

Whether UTC-SLS can still be used with such specifications depends on the requirements of a particular application. Strict clock-rate tolerances, like those given in [MPEG], can be critical in tightly synchronized television-studio networks. On the other hand, requirements may be far more relaxed if the same audio-visual data is streaming over the Internet. There, the receiver must buffer data anyway for several seconds to compensate network jitter, and a leap second smoothed over 1000 seconds becomes negligible compared to the normal variability in network delay.

Examples for other applications with strict clock-stability requirements include spacecraft navigation systems, where the motion of bodies is measured and predicted far more accurately than with 0.1% error, or the control of distributed scientific instruments that operate in a global reference frame, such as telescopes and seismographs. Such time-critical applications are usually implemented on special real-time hardware and software that reduce

the many scheduling and timing hazards of general-purpose platforms, such as operating systems with preemptive multitasking. Some real-time programming environments provide several clocks with different stability properties. For example, in [POSIX], the clock_gettime() function distinguishes between a "REALTIME" and a "MONOTONIC" clock. The former is meant to approximate UTC, and can do so in the form of UTC-SLS. The "MONOTONIC" clock, on the other hand, does not approximate any external clock, but aims to be as rate stable as possible. It may just count the seconds since the last system restart. It is, therefore, a more suitable choice for sensitive time-interval measurements.

A number of standard time scales exist that are, in contrast to UTC and UTC-SLS, not synchronized with Earth's rotation. They are entirely defined by precision clocks and feature no leap seconds. UTC falls behind these timescales by one more second with each positive leap second. Some examples are:

- International Atomic Time (TAI) is a leap-second free timescale defined by the same clock network that determines UTC. It was approximately identical to Universal Time on 1958-01-01 and was exactly 10 seconds ahead of UTC by 1972-01-01.

- GPS Time is a timescale used within the U.S. Department of Defense Global Positioning System. It has its origin at 1980-01-06 00:00 UTC, when TAI was 19 s ahead of UTC. Therefore, GPS Time remains 19 s behind TAI. (Many GPS receivers can display both GPS Time and UTC.)

- Terrestrial Time is an International Astronomical Union timescale used in astronomical tables. It is 32.184 s ahead of TAI.

It can be expected that some future operating systems will maintain an additional clock that is synchronized with one of these timescales. Such clocks are well suited for highly accurate long-term time-interval measurements. However, they lack the close relationship to legal time that UTC-synchronized clocks offer. And because a clock synchronized to TAI or GPS Time may still need substantial readjustment after a temporary loss of synchronization, they may not guarantee the same short-term rate stability as a clock like "MONOTONIC" that is not externally synchronized.

Normative References

[ITU-460] "Standard-frequency and time-signal emissions", ITU-R Recommendation TF.460-6, International Telecommunication Union, Geneva, 2002.

[IERS-C] Gambis, D., "Bulletin C", International Earth Rotation and Reference Systems Service (IERS), Paris, France. http://hpiers.obspm.fr/iers/bul/bulc/

Informal References

[Nel01] Nelson, R., et al., "The leap second: its history and possible future", Metrologia, Vol. 38, 2001, pp. 509-529.

[McC91] McCarthy, D.: "Astronomical Time", Proceedings of the IEEE, Vol. 79, No. 7, July 1991, pp. 915-920.

[ITU-768] "Standard frequencies and time signals", ITU-R Recommendation TF.768-5, International Telecommunication Union, Geneva, 2002.

[ISO8601] "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601, International Organization for Standardization, Geneva, 2004.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC1305] D. Mills, "Network Time Protocol (Version 3) : Specification, Implementation and Analysis", RFC 1305, March 1992.

[IEC1588] "Precision clock synchronization protocol for networked measurement and control systems", IEC 61588, International Electrotechnical Commission, Geneva, 2004.

[POSIX] The Open Group, "Single UNIX Specification", Version 3, Base Specifications Issue 6, IEEE Std 1003.1, 2004 edition. http://www.unix.org/

[C] "Programming languages -- C", ISO/IEC 9899, International Organization for Standardization, Geneva, 1999.

[MPEG] "Information technology -- Generic coding of moving pictures and associated audio information -- Part 1: Systems", ISO/IEC 13818-1, 1997.

Author's Address

Markus G. Kuhn
University of Cambridge
Computer Laboratory

15 JJ Thomson Avenue
Cambridge CB3 0FD
England

Email: Markus.Kuhn@cl.cam.ac.uk
Phone: +44 1223 334676
WWW: http://www.cl.cam.ac.uk/~mgk25/

Full Copyright Statement

Intellectual Property