LinguaFrancaClocks

Frédéric Boulanger

March 5, 2020

Contents

theory *LinguaFrancaClocks*

imports *Main*

begin

1 Basic definitions

Instants are represented as the natural numbers. A clock represents an event that may occur or not at any instant. We model a clock as a function from nat to bool, which is True at every instant when the clock ticks (the event occurs).

type-synonym *clock* = ‹*nat* ⇒ *bool*›

1.1 Periodic clocks

A clock is (k,p)-periodic if it ticks at instants separated by p instants, starting at instant k.

definition *kp-periodic* :: ‹$[nat,\ nat,\ clock] \Rightarrow bool$›
  where ‹$kp\text{-}periodic\ k\ p\ c \equiv$
    $(p > 0) \wedge (\forall n.\ c\ n = ((n \geq k) \wedge ((n - k)\ mod\ p = 0)))$›

A 1-periodic clock always ticks starting at its offset

lemma *one-periodic-ticks*:
  assumes ‹$kp\text{-}periodic\ k\ 1\ c$›
    and ‹$n \geq k$›
  shows ‹$c\ n$›
⟨*proof*⟩

A p-periodic clock is a (k,p)-periodic clock starting from a given offset.

definition ‹$p\text{-}periodic\ p\ c \equiv (\exists k.\ kp\text{-}periodic\ k\ p\ c)$›

lemma *p-periodic-intro*[*intro*]:
  ‹$kp\text{-}periodic\ k\ p\ c \Longrightarrow p\text{-}periodic\ p\ c$›
⟨*proof*⟩

No clock is 0-periodic.

lemma *no-0-periodic*:
  ‹$\neg p\text{-}periodic\ 0\ c$›
⟨*proof*⟩

A periodic clock is a p-periodic clock for a given period.

definition ‹$periodic\ c \equiv (\exists p.\ p\text{-}periodic\ p\ c)$›

lemma *periodic-intro1*[*intro*]:
  ‹$p\text{-}periodic\ p\ c \Longrightarrow periodic\ c$›
⟨*proof*⟩

lemma *periodic-intro2*[*intro*]:
  ‹$kp\text{-}periodic\ k\ p\ c \Longrightarrow periodic\ c$›
⟨*proof*⟩

1.2 Sporadic clocks

A clock is p-sporadic if it ticks at instants separated at least by p instants.

definition *p-sporadic* :: ‹$[nat,\ clock] \Rightarrow bool$›
  where ‹$p\text{-}sporadic\ p\ c \equiv \forall t.\ c\ t \longrightarrow (\forall t'.\ (t' > t \wedge c\ t') \longrightarrow t' > t + p)$›

Any clock is 0-sporadic

lemma *sporadic-0*: ‹$p\text{-}sporadic\ 0\ c$›
  ⟨*proof*⟩

We define sporadic clock as p-sporadic clocks for some non null interval p.

definition $\langle$*sporadic c* $\equiv$ ($\exists$ *p* > *0*. *p-sporadic p c*)$\rangle$

lemma *sporadic-intro*[*intro*]
  :$\langle\llbracket$*p-sporadic p c*;*p* > *0*$\rrbracket \Longrightarrow$ *sporadic c*$\rangle$
$\langle$*proof*$\rangle$

2 Properties of clocks

Some useful lemmas about modulo.

lemma *mod-sporadic*:
  assumes $\langle((n::nat)\ mod\ p = 0)\rangle$
    shows $\langle\forall n'.\ (n < n' \wedge n' < n{+}p) \longrightarrow \neg(n'\ mod\ p = 0)\rangle$
$\langle$*proof*$\rangle$

lemma *mod-sporadic′*:
  assumes $\langle((n::nat)\ mod\ p = 0)\rangle$
    shows $\langle\forall n'.\ (n < n' \wedge (n'\ mod\ p = 0)) \longrightarrow n' \geq n{+}p\rangle$
$\langle$*proof*$\rangle$

lemma *mod-offset-sporadic*:
  assumes $\langle(n::nat) \geq k\rangle$
      and $\langle(n - k)\ mod\ p = 0\rangle$
    shows $\langle\forall n'.\ (n < n' \wedge n' < n{+}p) \longrightarrow \neg((n'{-}k)\ mod\ p = 0)\rangle$
$\langle$*proof*$\rangle$

lemma *mod-offset-sporadic′*:
  assumes $\langle(n::nat) \geq k\rangle$
      and $\langle(n - k)\ mod\ p = 0\rangle$
    shows $\langle\forall n'.\ (n < n' \wedge ((n'{-}k)\ mod\ p = 0)) \longrightarrow n' \geq n{+}p\rangle$
$\langle$*proof*$\rangle$

A (p+1)-periodic clock is p-sporadic.

lemma *periodic-suc-sporadic*:
  assumes $\langle$*p-periodic* (*p*+*1*) *c*$\rangle$
    shows $\langle$*p-sporadic p c*$\rangle$
$\langle$*proof*$\rangle$

3 Merging clocks

The result of merging two clocks ticks whenever any of the two clocks ticks.

definition *merge* :: $\langle$[*clock*, *clock*] $\Rightarrow$ *clock*$\rangle$ (**infix** $\langle\oplus\rangle$ *60*)
  where $\langle c1 \oplus c2 \equiv \lambda t.\ c1\ t \vee c2\ t\rangle$

lemma *merge-comm*: $\langle c \oplus c' = c' \oplus c\rangle$
$\langle$*proof*$\rangle$

Merging two sporadic clocks does not necessary yields a sporadic clock.

lemma *merge-no-sporadic*:
‹$\exists\, c\ c'.\ sporadic\ c \wedge sporadic\ c' \wedge \neg sporadic\ (c \oplus c')$›
⟨*proof*⟩

Get the number of ticks on a clock from the beginning up to instant n.

definition *ticks-up-to* :: ‹$[clock,\ nat] \Rightarrow nat$›
where ‹$ticks\text{-}up\text{-}to\ c\ n = card\ \{t.\ t \le n \wedge c\ t\}$›

There cannot be more than n event occurrences during n instants.

lemma ‹$ticks\text{-}up\text{-}to\ c\ n \le Suc\ n$›
⟨*proof*⟩

Counting event occurrences.

definition ‹$count\ b\ n \equiv if\ b\ then\ Suc\ n\ else\ n$›

The count of event occurrences cannot grow by more than one at each instant.

lemma *count-inc*: ‹$count\ b\ n \le Suc\ n$›
⟨*proof*⟩

Alternative definition of the number of event occurrences using fold.

definition *ticks-up-to-fold* :: ‹$[clock,\ nat] \Rightarrow nat$›
where ‹$ticks\text{-}up\text{-}to\text{-}fold\ c\ n = fold\ count\ (map\ c\ [0..<Suc\ n])\ 0$›

Alternative definition of the number of event occurrences as a function.

fun *ticks-up-to-fun* :: ‹$[clock,\ nat] \Rightarrow nat$›
where
‹$ticks\text{-}up\text{-}to\text{-}fun\ c\ 0 = count\ (c\ 0)\ 0$›
| ‹$ticks\text{-}up\text{-}to\text{-}fun\ c\ (Suc\ n) = count\ (c\ (Suc\ n))\ (ticks\text{-}up\text{-}to\text{-}fun\ c\ n)$›

Proof that the original definition and the function definition are equivalent. Use this to generate code.

lemma *ticks-up-to-is-fun*[*code*]: ‹$ticks\text{-}up\text{-}to\ c\ n = ticks\text{-}up\text{-}to\text{-}fun\ c\ n$›
⟨*proof*⟩

Number of event occurrences during an n instant window starting at t_0.

definition *tick-count* ::‹$[clock,\ nat,\ nat] \Rightarrow nat$›
where ‹$tick\text{-}count\ c\ t_0\ n \equiv card\ \{t.\ t_0 \le t \wedge t < t_0{+}n \wedge c\ t\}$›

The number of event occurrences is monotonous with regard to the window width.

lemma *tick-count-mono*:
assumes ‹$n' \ge n$›
shows ‹$tick\text{-}count\ c\ t_0\ n' \ge tick\text{-}count\ c\ t_0\ n$›
⟨*proof*⟩

The interval [t, t+n[contains n instants.

lemma *card-interval*:‹$card\ \{t.\ t_0 \leq t \wedge t < t_0{+}n\} = n$›
⟨*proof*⟩

There cannot be more than n occurrences of an event in an interval of n instants.

lemma *tick-count-bound*: ‹$tick\text{-}count\ c\ t_0\ n \leq n$›
⟨*proof*⟩

No event occurrence occur in 0 instant.

lemma *tick-count-0*[*code*]: ‹$tick\text{-}count\ c\ t_0\ 0 = 0$›
⟨*proof*⟩

Event occurrences starting from instant 0 are event occurrences from the beginning.

lemma *tick-count-orig*[*code*]:
‹$tick\text{-}count\ c\ 0\ (Suc\ n) = ticks\text{-}up\text{-}to\ c\ n$›
⟨*proof*⟩

Counting event occurrences between two instants is simply subtracting occurrence counts from the beginning.

lemma *tick-count-diff*[*code*]:
‹$tick\text{-}count\ c\ (Suc\ t_0)\ n = (ticks\text{-}up\text{-}to\ c\ (t_0{+}n)) - (ticks\text{-}up\text{-}to\ c\ t_0)$›
⟨*proof*⟩

The merge of two clocks has less ticks than the union of the ticks of the two clocks.

lemma *tick-count-merge*:
‹$tick\text{-}count\ (c{\oplus}c')\ t_0\ n \leq tick\text{-}count\ c\ t_0\ n + tick\text{-}count\ c'\ t_0\ n$›
⟨*proof*⟩

4 Bounded clocks

An (n,m)-bounded clock does not tick more than m times in a n interval of width n.

definition *bounded* :: ‹$[nat,\ nat,\ clock] \Rightarrow bool$›
where ‹$bounded\ n\ m\ c \equiv \forall\, t.\ tick\text{-}count\ c\ t\ n \leq m$›

All clocks are (n,n)-bounded.

lemma *bounded-n*: ‹$bounded\ n\ n\ c$›
⟨*proof*⟩

A sporadic clock is bounded.

lemma *spor-bound*:
assumes ‹$\forall\, t{::}nat.\ c\ t \longrightarrow (\forall\, t'.\ (t < t' \wedge t' \leq t{+}n) \longrightarrow \neg(c\ t'))$›

shows ‹$\forall\, t$::*nat*. *card* $\{t'.\ t \le t' \wedge t' \le t{+}n \wedge c\ t'\} \le 1$›
⟨*proof*⟩

A sporadic clock is bounded.

lemma *spor-bound′*:
assumes ‹$\forall\, t$::*nat*. $c\ t \longrightarrow (\forall\, t'.\ (t < t' \wedge c\ t') \longrightarrow t' > t{+}n)$›
shows ‹$\forall\, t$::*nat*. *card* $\{t'.\ t \le t' \wedge t' \le t{+}n \wedge c\ t'\} \le 1$›
⟨*proof*⟩

An n-sporadic clock is (n+1, 1)-bounded.

lemma *spor-bounded*:
assumes ‹*p-sporadic n c*›
shows ‹*bounded* $(n{+}1)$ *1 c*›
⟨*proof*⟩

An n-sporadic clock is (n+2, 2)-bounded.

lemma *spor-bounded2*:
assumes ‹*p-sporadic n c*›
shows ‹*bounded* $(n{+}2)$ *2 c*›
⟨*proof*⟩

A bounded clock on an interval is also bounded on a narrower interval.

lemma *bounded-less*:
assumes ‹*bounded* n' *m c*›
and ‹$n' \ge n$›
shows ‹*bounded n m c*›
⟨*proof*⟩

The merge of two bounded clocks is bounded.

lemma *bounded-merge*:
assumes ‹*bounded n m c*›
and ‹*bounded* $n'\ m'\ c'$›
and ‹$n' \ge n$›
shows ‹*bounded* $n\ (m{+}m')\ (c{\oplus}c')$›
⟨*proof*⟩

The merge of two sporadic clocks is bounded.

lemma *sporadic-bounded1*:
assumes ‹*p-sporadic n c*›
and ‹*p-sporadic* $n'\ c'$›
and ‹$n' \ge n$›
shows ‹*bounded* $(n{+}1)$ *2* $(c{\oplus}c')$›
⟨*proof*⟩

4.1 Main theorem

The merge of two sporadic clocks is bounded on the min of the bounding intervals.

theorem *sporadic-bounded-min*:
  assumes ‹*p-sporadic n c*›
    and ‹*p-sporadic n′ c′*›
  shows ‹*bounded* ((*min n n′*)+*1*) *2* (*c*⊕*c′*)›
⟨*proof*⟩

5 Logical time

Logical time is a natural number that is attached to instants. Logical time can stay constant for an arbitrary number of instants, but it cannot decrease. When logical time stays constant for an infinite number of instants, we have a Zeno condition.

typedef *time* = ‹{*t*::*nat*⇒*nat*. *mono t*}›
  ⟨*proof*⟩

setup-lifting *type-definition-time*

A chronometric clock is a clock associated with a time line.

type-synonym *chronoclock* = ‹*clock*×*time*›

@term$c \ \nabla \ t$ tells whether chronometric clock c ticks at instant t.

definition *ticks* :: ‹[*chronoclock*, *nat*] ⇒ *bool*› (**infix** ‹∇› *60*)
  where ‹$c \ \nabla \ t \equiv (fst \ c) \ t$›

@termc_t is the logical time on clock c at instant t.

lift-definition *time-at* :: ‹[*chronoclock*, *nat*] ⇒ *nat*› (‹-_› [*60*, *60*])
  is ‹$\lambda c \ t. \ (snd \ c) \ t$› ⟨*proof*⟩

lemmas *chronoclocks-simp*[*simp*] = *ticks-def time-at-def*

As consequence of the definition of the *time* type, (∇) is monotonous for any clock.

lemma *mono-chronotime*:
  ‹*mono* (*time-at c*)› ⟨*proof*⟩

An event occurs at a given time if the clock ticks at some instant at that time.

definition *occurs* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
  where ‹$occurs \ n \ c \equiv \exists k. \ (c \ \nabla \ k \wedge c_k = n)$›

An event occurs once at a given time if the clock ticks at exactly one instant at that time.

definition *occurs-once* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
  where ‹$occurs\text{-}once \ n \ c \equiv \exists! k. \ (c \ \nabla \ k \wedge c_k = n)$›

lemma *occurs-once-occurs*:

‹$occurs\text{-}once\ n\ c \Longrightarrow occurs\ n\ c$›
⟨*proof*⟩

A clock is strict at a given time if it ticks at most once at that time.

definition *strict-at* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
where ‹$strict\text{-}at\ n\ c \equiv (occurs\ n\ c \longrightarrow occurs\text{-}once\ n\ c)$›

definition *strict-clock* :: ‹*chronoclock* ⇒ *bool*›
where ‹$strict\text{-}clock\ c \equiv (\forall n.\ strict\text{-}at\ n\ c)$›

5.1 Chrono-periodic and chrono-sporadic clocks

The introduction of logical time allows us to define periodicity and sporadicity on logical time instead of instant index.

definition *kp-chronoperiodic* :: ‹[*nat*, *nat*, *chronoclock*] ⇒ *bool*›
where ‹$kp\text{-}chronoperiodic\ k\ p\ c \equiv (p > 0) \wedge (\forall n.\ occurs\ n\ c = ((n \geq k) \wedge ((n - k)\ mod\ p = 0)))$›

definition *p-chronoperiodic* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
where ‹$p\text{-}chronoperiodic\ p\ c \equiv \exists k.\ kp\text{-}chronoperiodic\ k\ p\ c$›

definition *chronoperiodic* :: ‹[*chronoclock*] ⇒ *bool*›
where ‹$chronoperiodic\ c \equiv \exists p.\ p\text{-}chronoperiodic\ p\ c$›

A clock is strictly chronoperiodic if it ticks only once at the logical times when it ticks.

definition *chronoperiodic-strict* :: ‹[*chronoclock*] ⇒ *bool*›
where ‹$chronoperiodic\text{-}strict\ c \equiv chronoperiodic\ c \wedge strict\text{-}clock\ c$›

definition *p-chronoperiodic-strict* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
where ‹$p\text{-}chronoperiodic\text{-}strict\ p\ c \equiv p\text{-}chronoperiodic\ p\ c \wedge strict\text{-}clock\ c$›

lemma ‹$chronoperiodic\text{-}strict\ c \Longrightarrow chronoperiodic\ c$›
⟨*proof*⟩

definition *p-chronosporadic* :: ‹[*nat*, *chronoclock*] ⇒ *bool*›
where ‹$p\text{-}chronosporadic\ p\ c \equiv$
$\forall t.\ occurs\ t\ c \longrightarrow (\forall t'.\ (t' > t \wedge occurs\ t'\ c) \longrightarrow t' > t + p)$›

definition ‹$p\text{-}chronosporadic\text{-}strict\ p\ c \equiv p\text{-}chronosporadic\ p\ c \wedge strict\text{-}clock\ c$›

definition ‹$chronosporadic\ c \equiv (\exists p > 0.\ p\text{-}chronosporadic\ p\ c)$›

definition ‹$chronosporadic\text{-}strict\ c \equiv chronosporadic\ c \wedge strict\text{-}clock\ c$›

lemma *chrono-periodic-suc-sporadic*:
assumes ‹$p\text{-}chronoperiodic\ (p+1)\ c$›
shows ‹$p\text{-}chronosporadic\ p\ c$›

⟨*proof*⟩

lemma *chrono-periodic-suc-sporadic-strict*:
assumes ‹*p-chronoperiodic-strict* $(p+1)$ *c*›
shows ‹*p-chronosporadic-strict p c*›
⟨*proof*⟩

Number of ticks up to a given logical time. This counts distinct ticks that happen at the same logical time.

definition *chrono-dense-up-to* ::‹[*chronoclock*, *nat*] ⇒ *nat*›
where ‹$\textit{chrono-dense-up-to}\ c\ n = \textit{card}\ \{t.\ c_t \leq n \land c\ \nabla\ t\}$›

A clock is Zeno if it ticks an infinite number of times in a finite amount of time.

definition *zeno-clock* :: ‹*chronoclock* ⇒ *bool*›
where ‹$\textit{zeno-clock}\ c \equiv (\exists\omega.\ \textit{infinite}\ \{t.\ c_t \leq \omega \land c\ \nabla\ t\})$›

Number of occurrences of an event up to a given logical time. This does not count separately ticks that occur at the same logical time.

definition *chrono-up-to* ::‹[*chronoclock*, *nat*] ⇒ *nat*›
where ‹$\textit{chrono-up-to}\ c\ n = \textit{card}\ \{t.\ t \leq n \land \textit{occurs}\ t\ c\}$›

For any time n, a non Zeno clock has less occurrences than ticks up to n. This is also true for Zeno clock, but we count ticks and occurrences using *card*, and in Isabelle/HOL, the cardinal of an infinite set is 0, so the inequality breaks when there are infinitely many ticks before a given time.

lemma *not-zeno-sparse*:
assumes ‹¬*zeno-clock c*›
shows ‹$\textit{chrono-up-to}\ c\ n \leq \textit{chrono-dense-up-to}\ c\ n$›
⟨*proof*⟩

6 Tests

abbreviation ‹$c1{::}\textit{clock} \equiv (\lambda t.\ t \geq 1 \land (t-1)\ \textit{mod}\ 2 = 0)$›
abbreviation ‹$c2{::}\textit{clock} \equiv (\lambda t.\ t \geq 2 \land (t-2)\ \textit{mod}\ 3 = 0)$›

value ‹*c1 0*›
value ‹*c1 1*›
value ‹*c1 2*›
value ‹*c1 3*›

value ‹*c2 0*›
value ‹*c2 1*›
value ‹*c2 2*›
value ‹*c2 3*›
value ‹*c2 4*›
value ‹*c2 5*›

lemma ‹*kp-periodic 1 2 c1*›
⟨*proof*⟩

lemma ‹*kp-periodic 2 3 c2*›
⟨*proof*⟩

abbreviation ‹$c3 \equiv c1 \oplus c2$›

value ‹*map c1* [*0*,*1*,*2*,*3*,*4*,*5*,*6*,*7*,*8*,*9*,*10*]›
value ‹*map c2* [*0*,*1*,*2*,*3*,*4*,*5*,*6*,*7*,*8*,*9*,*10*]›
value ‹*map c3* [*0*,*1*,*2*,*3*,*4*,*5*,*6*,*7*,*8*,*9*,*10*]›

lemma *interv-2*:‹$\{t{::}nat.\ t_0 \leq t \wedge t < t_0 + 2 \wedge 1 \leq t \wedge (t - 1)\ mod\ 2 = 0\} = \{t.\ (t = t_0 \vee t = t_0 + 1) \wedge 1 \leq t \wedge (t - 1)\ mod\ 2 = 0\}$›
⟨*proof*⟩

lemma ‹*bounded 2 1 c1*›
⟨*proof*⟩

end