

Instances of Schneider’s generalized protocol of clock synchronization.

Damian Barsotti

June 24, 2019

Abstract

Schneider [7] generalizes a number of protocols for Byzantine fault-tolerant clock synchronization and presents a uniform proof for their correctness. In Schneider’s schema, each processor maintains a local clock by periodically adjusting each value to one computed by a convergence function applied to the readings of all the clocks. Then, correctness of an algorithm, i.e. that the readings of two clocks at any time are within a fixed bound of each other, is based upon some conditions on the convergence function. To prove that a particular clock synchronization algorithm is correct it suffices to show that the convergence function used by the algorithm meets Schneider’s conditions.

Using the theorem prover Isabelle, we formalize the proofs that the convergence functions of two algorithms, namely, the Interactive Convergence Algorithm (ICA) of Lamport and Melliar-Smith [4] and the Fault-tolerant Midpoint algorithm of Lundelius-Lynch [5], meet Schneider’s conditions. Furthermore, we experiment on handling some parts of the proofs with fully automatic tools like ICS[3] and CVC-lite[2].

These theories are part of a joint work with Alwen Tiu and Leonor P. Nieto [1]. In this work the correctness of Schneider schema was also verified using Isabelle (available at <http://isa-afp.org/entries/GenClock.shtml>).

Contents

| | | |
|----------|---|----------|
| 1 | Interactive Convergence Algorithms (ICA) | 2 |
| 1.1 | Model of the system | 2 |
| 1.1.1 | Types in the formalization | 2 |
| 1.1.2 | Some constants | 3 |
| 1.1.3 | Convergence function | 3 |
| 1.2 | Translation Invariance property. | 3 |
| 1.3 | Precision Enhancement property | 4 |
| 1.3.1 | Auxiliary lemmas | 4 |
| 1.3.2 | Main theorem | 11 |

| | | |
|----------|--|-----------|
| 1.4 | Accuracy Preservation property | 12 |
| 1.4.1 | Main theorem | 14 |
| 2 | Fault-tolerant Midpoint algorithm | 16 |
| 2.1 | Model of the system | 16 |
| 2.1.1 | Types in the formalization | 16 |
| 2.1.2 | Some constants | 16 |
| 2.1.3 | Convergence function | 17 |
| 2.2 | Translation Invariance property | 17 |
| 2.2.1 | Auxiliary lemmas | 17 |
| 2.2.2 | Main theorem | 23 |
| 2.3 | Precision Enhancement property | 25 |
| 2.3.1 | Auxiliary lemmas | 25 |
| 2.3.2 | Main theorem | 32 |
| 2.4 | Accuracy Preservation property | 35 |
| A | CVC-lite and ICS proofs | 36 |
| A.1 | Lemma <code>abs_distrib_div</code> | 36 |
| A.2 | Bound for Precision Enhancement property | 37 |
| A.3 | Accuracy Preservation property | 37 |

1 Interactive Convergence Algorithms (ICA)

theory *ICAInstance* **imports** *Complex-Main* **begin**

This algorithm is presented in [4].

A proof of the three properties can be found in [8].

1.1 Model of the system

The main ideas for the formalization of the system were obtained from [8].

1.1.1 Types in the formalization

The election of the basics types was based on [8]. There, the process are natural numbers and the real time and the clock readings are reals.

type-synonym *process* = *nat*

type-synonym *time* = *real* — real time

type-synonym *Clocktime* = *real* — time of the clock readings (clock time)

1.1.2 Some constants

Here we define some parameters of the algorithm that we use: the number of process and the fix value that is used to discard the processes whose clocks differ more than this amount from the own one (see [8]). The defined constants must satisfy this axiom (if $np = 0$ we have a division by zero in the definition of the convergence function).

axiomatization

$np :: nat$ — Number of processes **and**
 $\Delta :: Clocktime$ — Fix value to discard processes **where**
constants-ax: $0 \leq \Delta \wedge np > 0$

We define also the set of process that the algorithm manage. This definition exist only for readability matters.

definition

$PR :: process\ set$ **where**
[simp]: $PR = \{..<np\}$

1.1.3 Convergence function

This functions is called “Egocentric Average” ([7])

In this algorithm each process has an array where it store the clocks readings from the others processes (including itself). We formalise that as a function from processes to clock time as [8].

First we define an auxiliary function. It takes a function of clock readings and two processes, and return de reading of the second process if the difference of the readings is grater than Δ , otherwise it returns the reading of the first one.

definition

$fiX :: [(process \Rightarrow Clocktime), process, process] \Rightarrow Clocktime$ **where**
 $fiX\ f\ p\ l = (if\ |f\ p - f\ l| \leq \Delta\ then\ (f\ l)\ else\ (f\ p))$

And finally the convergence function. This is defined with the builtin generalized summation over a set constructor of Isabelle. Also we had to use the overloaded *real* function to typecast de number np .

definition

$cfni :: [process, (process \Rightarrow Clocktime)] \Rightarrow Clocktime$ **where**
 $cfni\ p\ f = (\sum\ l \in \{..<np\}. fiX\ f\ p\ l) / (real\ np)$

1.2 Translation Invariance property.

We first need to prove this auxiliary lemma.

lemma *trans-inv'*:

```

( $\sum l \in \{..<np'\}$ .  $fiX (\lambda y. f y + x) p l$ ) =
  ( $\sum l \in \{..<np'\}$ .  $fiX f p l$ ) +  $real np' * x$ 
apply (induct-tac  $np'$ )
apply (auto simp add: cfni-def fiX-def of-nat-Suc
  distrib-right lessThan-Suc)
done

```

```

theorem trans-inv:
 $\forall p f x . cfni p (\lambda y. f y + x) = cfni p f + x$ 
apply (auto simp add: cfni-def trans-inv' distrib-right
  divide-inverse constants-ax)
done

```

1.3 Precision Enhancement property

An informal proof of this theorem can be found in [8]

1.3.1 Auxiliary lemmas

```

lemma finitC:
   $C \subseteq PR \implies finite C$ 
proof –
  assume  $C \subseteq PR$ 
  thus ?thesis using finite-subset by auto
qed

```

```

lemma finitnpC:
   $finite (PR - C)$ 
proof –
  show ?thesis using finite-Diff by auto
qed

```

The next lemmas are about arithmetic properties of the generalized summation over a set constructor.

```

lemma sum-abs-triangle-ineq:
 $finite S \implies$ 
   $|\sum l \in S. (f::'a \implies 'b::linordered-idom) l| \leq (\sum l \in S. |f l|)$ 
  (is ...  $\implies ?P S$ )
  (by (rule sum-abs))

```

```

lemma sum-le:
   $\llbracket finite S ; \forall r \in S. f r \leq b \rrbracket$ 
   $\implies$ 
   $(\sum l \in S. f l) \leq real (card S) * b$ 
  (is  $\llbracket finite S ; \forall r \in S. f r \leq b \rrbracket \implies ?P S$ )
proof(induct S rule: finite-induct)
  show  $?P \{\}$  by simp
next
  fix  $F x$ 

```

```

assume finit: finite F and xnotinF:  $x \notin F$  and
  HI1:  $\forall r \in F. f r \leq b \implies \text{sum } f F \leq \text{real } (\text{card } F) * b$ 
  and HI2:  $\forall r \in \text{insert } x F. f r \leq b$ 
from HI1 HI2 and finit and xnotinF
have  $\text{sum } f (\text{insert } x F) \leq b + \text{real } (\text{card } F) * b$ 
  by auto
also
have  $\dots = \text{real } (\text{Suc } (\text{card } F)) * b$ 
  by (simp add: distrib-right of-nat-Suc)
also
from finit xnotinF have  $\dots = \text{real } (\text{card } (\text{insert } x F)) * b$ 
  by simp
finally
show  $?P (\text{insert } x F)$  .
qed

```

```

lemma sum-np-eq:
assumes
  hC:  $C \subseteq PR$ 
shows
   $(\sum l \in \{..<np\}. f l) = (\sum l \in C. f l) + (\sum l \in (\{..<np\} - C). f l)$ 
proof -
  note finitC[where  $C=C$ ]
  moreover
  note finitnpC[where  $C=C$ ]
  moreover
  have  $C \cap (\{..<np\} - C) = \{\}$  by auto
  moreover
  from hC have  $C \cup (\{..<np\} - C) = \{..<np\}$  by auto
  ultimately
  show ?thesis
    using sum.union-disjoint[where  $A=C$  and  $B=\{..<np\} - C$ ]
    by auto
qed

```

```

lemma abs-sum-np-ineq:
assumes
  hC:  $C \subseteq PR$ 
shows
   $|\sum l \in \{..<np\}. (f :: nat \Rightarrow real) l| \leq$ 
   $(\sum l \in C. |f l|) + (\sum l \in (\{..<np\} - C). |f l|)$ 
  (is ?abs-sum  $\leq$  ?sumC + ?sumnpC)
proof -
  from hC and sum-np-eq[where  $f=f$ ]
  have ?abs-sum =  $|\sum l \in C. f l| + |\sum l \in (\{..<np\} - C). f l|$ 
  (is ?abs-sum =  $|\text{?sumC}' + \text{?sumnpC}'|$ )
  by simp
also
from abs-triangle-ineq

```

```

have ... <= |?sumC'| + |?sumnpC'| .
also
have ... <= ?sumC + ?sumnpC
proof-
  from hC finitC sum-abs-triangle-ineq
  have |?sumC'| <= ?sumC by blast
  moreover
  from finitnpC and
    sum-abs-triangle-ineq[where f=f and S=PR-C]
  have |?sumnpC'| <= ?sumnpC
    by force
  ultimately
  show ?thesis by arith
qed
finally
show ?thesis .
qed

```

The next lemmas are about the existence of bounds that are necessary in order to prove the Precision Enhancement theorem.

```

lemma fiX-ubound:
  fiX f p l <= f p + Δ
proof(cases |f p - f l| ≤ Δ)
  assume asm: |f p - f l| ≤ Δ
  hence fiX f p l = f l by (simp add: fiX-def)
  also
  from asm have f l <= f p + Δ by arith
  finally
  show ?thesis by arith
next
  assume asm: ¬|f p - f l| ≤ Δ
  hence fiX f p l = f p by (simp add: fiX-def)
  also
  from asm and constants-ax have f p <= f p + Δ by arith
  finally
  show ?thesis by arith
qed

```

```

lemma fiX-lbound:
  f p - Δ <= fiX f p l
proof(cases |f p - f l| ≤ Δ)
  assume asm: |f p - f l| ≤ Δ
  hence fiX f p l = f l by (simp add: fiX-def)
  also
  from asm have f p - Δ <= f l by arith
  finally
  show ?thesis by arith
next
  assume asm: ¬|f p - f l| ≤ Δ

```

with *constants-ax* **have** $f p - \Delta \leq f p$ **by** *arith*
also
from *asm* **have** $f p = fiX f p l$ **by** (*simp add: fiX-def*)
finally
show *?thesis* **by** *arith*
qed

lemma *abs-fiX-bound*: $|fiX f p l - f p| \leq \Delta$
proof –

have $f p - \Delta \leq fiX f p l \wedge fiX f p l \leq f p + \Delta \longrightarrow ?thesis$
by *arith*
with *fiX-lbound* *fiX-ubound* **show** *?thesis* **by** *blast*
qed

lemma *abs-dif-fiX-bound*:

assumes

hbx: $\forall l \in C. |f l - g l| \leq x$ **and**
hby: $\forall l \in C. \forall m \in C. |f l - f m| \leq y$ **and**
hpC: $p \in C$ **and**
hqC: $q \in C$

shows

$|fiX f p r - fiX g q r| \leq 2 * \Delta + x + y$

proof –

have $|fiX f p r - fiX g q r| =$
 $|fiX f p r - f p + f p - fiX g q r|$
by *auto*

also

have $\dots \leq |fiX f p r - f p| + |f p - fiX g q r|$
by *arith*

also

from *abs-fiX-bound*

have $\dots \leq \Delta + |f p - fiX g q r|$
by *simp*

also

have $\dots = \Delta + |f p - g q + (g q - fiX g q r)|$
by *simp*

also

from *abs-triangle-ineq* [**where** $a = f p - g q$ **and**
 $b = g q - fiX g q r$]

have $\dots \leq \Delta + |f p - g q| + |g q - fiX g q r|$
by *simp*

also

have $\dots = \Delta + |f p - g q| + |fiX g q r - g q|$
by *arith*

also

from *abs-fiX-bound*

have $\dots \leq 2 * \Delta + |f p - g q|$
by *simp*

```

also
have ... = 2 * Δ + |f p - f q + (f q - g q) |
  by simp
also
from abs-triangle-ineq[where a = f p - f q and
  b = f q - g q]
have ... <= 2 * Δ + |f p - f q | + |f q - g q |
  by simp
finally
show ?thesis using hbx hby hpC hqC
  by force
qed

```

lemma *abs-dif-fiX-bound-C-aux1*:

assumes

```

hbx: ∀ l ∈ C. |f l - g l| <= x and
hby1: ∀ l ∈ C. ∀ m ∈ C. |f l - f m| <= y and
hby2: ∀ l ∈ C. ∀ m ∈ C. |g l - g m| <= y and
hpC: p ∈ C and
hqC: q ∈ C and
hrC: r ∈ C

```

shows

```

|fiX f p r - fiX g q r| <= x + y

```

proof(*cases* |f p - f r| ≤ Δ)

case *True*

note *outer-IH* = *True*

show ?thesis

proof(*cases* |g q - g r| ≤ Δ)

case *True*

show ?thesis

proof -

from *hpC* **and** *hby1* **have** 0 <= y **by** *force*

with *hrC* **and** *hbx* **have** |f r - g r| <= x + y **by** *auto*

with *outer-IH* **and** *True* **show** ?thesis

by (*auto simp add: fiX-def*)

qed

next

case *False*

show ?thesis

proof -

from *outer-IH* **and** *False*

have |fiX f p r - fiX g q r| = |f r - g q|

by (*auto simp add: fiX-def*)

also

have ... = |f r - f q + f q - g q| **by** *simp*

also

have ... <= |f r - f q| + |f q - g q|

by *arith*


```

    also
    from hbx hby1 hpC hqC hrC have ...  $\leq x + y$  by force
    finally
    show ?thesis .
  qed
next
case False
note outer-IH = False
show ?thesis
proof(cases  $|g\ q - g\ r| \leq \Delta$ )
  case True
  show ?thesis
  proof -
    from outer-IH and True
    have  $|fiX\ f\ p\ r - fiX\ g\ q\ r| = |f\ p - g\ r|$ 
      by (auto simp add: fiX-def)
    also
    have ... =  $|f\ p - f\ r + f\ r - g\ r|$  by simp
    also
    from abs-triangle-ineq[where  $a = f\ p - f\ r$  and
       $b = f\ r - g\ r$ ]
    have ...  $\leq |f\ p - f\ r| + |f\ r - g\ r|$ 
      by auto
    also
    from hbx hby1 hpC hrC have ...  $\leq x + y$  by force
    finally
    show ?thesis .
  qed
next
case False
show ?thesis
proof -
  from outer-IH and False
  have  $|fiX\ f\ p\ r - fiX\ g\ q\ r| = |f\ p - g\ q|$ 
    by (auto simp add: fiX-def)
  also
  have ... =  $|f\ p - f\ q + f\ q - g\ q|$  by simp
  also
  from abs-triangle-ineq[where  $a = f\ p - f\ q$  and
     $b = f\ q - g\ q$ ]
  have ...  $\leq |f\ p - f\ q| + |f\ q - g\ q|$ 
    by auto
  also
  from hbx hby1 hpC hqC have ...  $\leq x + y$  by force
  finally
  show ?thesis .
qed
qed

```

qed

lemma *abs-dif-fiX-bound-C-aux2*:

assumes

hbx: $\forall l \in C. |f l - g l| \leq x$ **and**
hby1: $\forall l \in C. \forall m \in C. |f l - f m| \leq y$ **and**
hby2: $\forall l \in C. \forall m \in C. |g l - g m| \leq y$ **and**
hpC: $p \in C$ **and**
hqC: $q \in C$ **and**
hrC: $r \in C$

shows

$y \leq \Delta \longrightarrow |fiX f p r - fiX g q r| \leq x$

proof

assume *hyd*: $y \leq \Delta$

show $|fiX f p r - fiX g q r| \leq x$

proof–

from *hpC* **and** *hrC* **and** *hby1* **and** *hyd* **have** $|f p - f r| \leq \Delta$
by force

moreover

from *hqC* **and** *hrC* **and** *hby2* **and** *hyd* **have** $|g q - g r| \leq \Delta$
by force

moreover

from *hrC* **and** *hbx* **have** $|f r - g r| \leq x$ **by auto**

ultimately

show *?thesis*

by (*auto simp add: fiX-def*)

qed

qed

lemma *abs-dif-fiX-bound-C*:

assumes

hbx: $\forall l \in C. |f l - g l| \leq x$ **and**
hby1: $\forall l \in C. \forall m \in C. |f l - f m| \leq y$ **and**
hby2: $\forall l \in C. \forall m \in C. |g l - g m| \leq y$ **and**
hpC: $p \in C$ **and**
hqC: $q \in C$ **and**
hrC: $r \in C$

shows

$|fiX f p r - fiX g q r| \leq$
 $x + (\text{if } (y \leq \Delta) \text{ then } 0 \text{ else } y)$

proof (*cases y <= Δ*)

case *True*

with *abs-dif-fiX-bound-C-aux2* **and**

hbx **and** *hby1* **and** *hby2* **and** *hpC* **and** *hqC* **and** *hrC*

have $|fiX f p r - fiX g q r| \leq x$ **by blast**

with *True* **show** *?thesis* **by simp**

next

case *False*

with *abs-dif-fiX-bound-C-aux1* **and**

hbx and hby1 and hby2 and hpC and hqC and hrC
have $|fiX f p r - fiX g q r| \leq x + y$ **by** *blast*
with *False show ?thesis by simp*
qed

1.3.2 Main theorem

theorem *prec-enh:*

assumes

hC: C ⊆ PR and

hbx: ∀ l ∈ C. |f l - g l| ≤ x and

hby1: ∀ l ∈ C. ∀ m ∈ C. |f l - f m| ≤ y and

hby2: ∀ l ∈ C. ∀ m ∈ C. |g l - g m| ≤ y and

hpC: p ∈ C and

hqC: q ∈ C

shows $|cfni p f - cfni q g| \leq$

$(\text{real } (\text{card } C) * (x + (\text{if } (y \leq \Delta) \text{ then } 0 \text{ else } y))) +$

$\text{real } (\text{card } (\{..\lt np\} - C)) * (2 * \Delta + x + y)) / \text{real } np$

(is $|?dif-div-np| \leq ?B$ **)**

proof –

have $|(\sum l \in \{..\lt np\}. fiX f p l) -$
 $(\sum l \in \{..\lt np\}. fiX g q l)| =$

$|(\sum l \in \{..\lt np\}. fiX f p l - fiX g q l)|$

(is $|?dif| = |?dif'|$ **)**

by *(simp add: sum-subtractf)*

also

from *abs-sum-np-ineq hC*

have $\dots \leq$

$(\sum l \in C. |fiX f p l - fiX g q l|) +$

$(\sum l \in (\{..\lt np\} - C). |fiX f p l - fiX g q l|)$

(is $|?dif'| \leq ?boundC' + ?boundnpC'$ **)**

by *simp*

also

have $\dots \leq$

$\text{real } (\text{card } C) * (x + (\text{if } (y \leq \Delta) \text{ then } 0 \text{ else } y)) +$

$\text{real } (\text{card } (\{..\lt np\} - C)) * (2 * \Delta + x + y)$

(is $\dots \leq ?boundC + ?boundnpC$ **)**

proof –

have $?boundC' \leq ?boundC$

proof –

from *abs-dif-fiX-bound-C and*

hbx and hby1 and hby2 and hpC and hqC

have $\forall r \in C.$

$|fiX f p r - fiX g q r| \leq x +$

$(\text{if } (y \leq \Delta) \text{ then } 0 \text{ else } y)$

by *blast*

thus *?thesis using sum-le[where S=C] and finitC[OF hC]*

by *force*

qed

```

moreover
have ?boundnpC' <= ?boundnpC
proof -
  from abs-dif-fiX-bound and
    hbx and hby1 and hpC and hqC
  have  $\forall r \in \{..<np\} - C$ .  $|fiX\ f\ p\ r - fiX\ g\ q\ r| <= 2 * \Delta + x + y$ 
    by blast
  with finitnpC
  show ?thesis
    by (auto intro: sum-le)
qed
ultimately
show ?thesis by arith
qed
finally
have bound:  $|?dif| <= ?boundC + ?boundnpC$  .
thus ?thesis
proof-
  have ?dif-div-np = ?dif / real np
    by (simp add: cfni-def divide-inverse algebra-simps)
  hence  $|cfni\ p\ f - cfni\ q\ g| = |?dif| / real\ np$ 
    by force
  with bound show ?thesis
    by (auto simp add: cfni-def divide-inverse constants-ax)
qed
qed

```

1.4 Accuracy Preservation property

First, a simple lemma about an arithmetic propertie of the generalized summation over a set constructor.

lemma *sum-div-card*:

```

 $(\sum l \in \{..<n::nat\}. f\ l) + q * real\ n =$ 
 $(\sum l \in \{..<n\}. f\ l + q)$ 
(is ?Sl n = ?Sr n)

```

proof (*induct n*)

case 0 **thus** ?case **by** *simp*

next

case (*Suc n*)

thus ?case

by (*auto simp: of-nat-Suc distrib-left lessThan-Suc*)

qed

Next, some lemmas about bounds that are used in the proof of Accuracy Preservation

lemma *bound-aux-C*:

assumes

hby: $\forall l \in C. \forall m \in C. |f\ l - f\ m| <= x$ **and**

$hpC: p \in C$ and
 $hqC: q \in C$ and
 $hrC: r \in C$

shows
 $|fiX f p r - f q| \leq x$

proof (*cases* $|f p - f r| \leq \Delta$)
case *True*
then have $|fiX f p r - f q| = |f r - f q|$
by (*simp add: fiX-def*)
also
from *hby hqC hrC* **have** $\dots \leq x$ **by** *blast*
finally
show *?thesis* .

next
case *False*
then have $|fiX f p r - f q| = |f p - f q|$
by (*simp add: fiX-def*)
also
from *hby hpC hqC* **have** $\dots \leq x$ **by** *blast*
finally
show *?thesis* .

qed

lemma *bound-aux*:

assumes
 $hby: \forall l \in C. \forall m \in C. |f l - f m| \leq x$ and
 $hpC: p \in C$ and
 $hqC: q \in C$

shows
 $|fiX f p r - f q| \leq x + \Delta$

proof (*cases* $|f p - f r| \leq \Delta$)
case *True*
then have $|fiX f p r - f q| = |f r - f q|$
by (*simp add: fiX-def*)
also
have $\dots = |(f r - f p) + (f p - f q)|$
by *arith*
also
have $\dots \leq |f p - f r| + |f p - f q|$
by *arith*
also
from *True* **have** $\dots \leq \Delta + |f p - f q|$ **by** *arith*
also
from *hby hpC hqC* **have** $\dots \leq \Delta + x$ **by** *simp*
finally
show *?thesis* **by** *simp*

next
case *False*
then have $|fiX f p r - f q| = |f p - f q|$

by (*simp add: fiX-def*)
 also
 from *hby hpC hqC* have ... $\leq x$ by *blast*
 finally
 show *?thesis* using *constants-ax* by *arith*
 qed

1.4.1 Main theorem

lemma *accur-pres*:

assumes

hC: $C \subseteq PR$ and

hby: $\forall l \in C. \forall m \in C. |f l - f m| \leq x$ and

hpC: $p \in C$ and

hqC: $q \in C$

shows $|cfni p f - f q| \leq$

$$\frac{\text{real}(\text{card } C) * x + \text{real}(\text{card}(\{..\lt np\} - C)) * (x + \Delta)}{\text{real } np}$$

(*is ?abs1* \leq (*?bC* + *?bnpC*)/*real np*)

proof–

from *abs-sum-np-ineq* and *hC* have

$$\begin{aligned} &|\sum l \in \{..\lt np\}. fiX f p l - f q| \leq \\ &(\sum l \in C. |fiX f p l - f q|) + \\ &(\sum l \in (\{..\lt np\} - C). |fiX f p l - f q|) \end{aligned}$$

by *simp*

also

have

$$\dots \leq \frac{\text{real}(\text{card } C) * x + \text{real}(\text{card}(\{..\lt np\} - C)) * (x + \Delta)}{\text{real } np}$$

proof–

$$\text{have } (\sum l \in C. |fiX f p l - f q|) \leq \frac{\text{real}(\text{card } C) * x}{\text{real } np}$$

proof–

from *bound-aux-C* and

hby and *hpC* and *hqC*

have $\forall r \in C.$

$$|fiX f p r - f q| \leq x$$

by *blast*

thus *?thesis* using *sum-le*[*where S=C*] and *finitC*[*OF hC*]

by *force*

qed

moreover

$$\text{have } (\sum l \in (\{..\lt np\} - C). |fiX f p l - f q|) \leq \frac{\text{real}(\text{card}(\{..\lt np\} - C)) * (x + \Delta)}{\text{real } np}$$

proof –

from *bound-aux* and

hby and *hpC* and *hqC*

have $\forall r \in (\{..\lt np\} - C).$

$$|fiX f p r - f q| \leq x + \Delta$$

by *blast*
 thus *?thesis* using *sum-le*[**where** $S = \{..<np\} - C$]
 and *finitnpC*
 by *force*
 qed
 ultimately
 show *?thesis* by *arith*
 qed
 finally
 have *bound*: $|\sum l \in \{..<np\}. fiX f p l - f q|$
 $\leq real (card C) * x + real (card (\{..<np\} - C)) * (x + \Delta)$
 .
 thus
?thesis
 proof-
 from *constants-ax* have
 $res: inverse (real np) * real np = 1$
 by *auto*
 have
 $(cfni p f - f q) * real np =$
 $(\sum l \in \{..<np\}. fiX f p l) * real np / real np - f q * real np$
 by (*simp add: cfni-def algebra-simps*)
 also
 have ... =
 $(\sum l \in \{..<np\}. fiX f p l) - f q * real np$
 by *simp*
 also
 from *sum-div-card*[**where** $f = fiX f p$ **and** $n = np$ **and** $q = - f q$]
 have ... = $(\sum l \in \{..<np\}. fiX f p l - f q)$
 by *simp*
 finally
 have
 $(cfni p f - f q) * real np = (\sum l \in \{..<np\}. fiX f p l - f q)$
 .
 — cambia
 hence
 $(cfni p f - f q) * real np / real np =$
 $(\sum l \in \{..<np\}. fiX f p l - f q) / real np$
 by *auto*
 with *constants-ax* have
 $(cfni p f - f q) =$
 $(\sum l \in \{..<np\}. fiX f p l - f q) / real np$
 by *simp*
 hence $|cfni p f - f q| =$
 $|(\sum l \in \{..<np\}. fiX f p l - f q) / real np|$
 by *simp*
 also have
 $... = |(\sum l \in \{..<np\}. fiX f p l - f q)| / real np$
 by *auto*

```

finally have | cfni p f - f q | =
  |( $\sum_{l \in \{..<np\}}$ . fiX f p l - f q)| / real np
  .
with bound show ?thesis
  by (auto simp add: cfni-def divide-inverse constants-ax)
qed
qed
end

```

2 Fault-tolerant Midpoint algorithm

theory *LynchInstance* **imports** *Complex-Main* **begin**

This algorithm is presented in [5].

2.1 Model of the system

The main ideas for the formalization of the system were obtained from [8].

2.1.1 Types in the formalization

The election of the basics types was based on [8]. There, the process are natural numbers and the real time and the clock readings are reals.

type-synonym *process* = *nat*

type-synonym *time* = *real* — real time

type-synonym *Clocktime* = *real* — time of the clock readings (clock time)

2.1.2 Some constants

Here we define some parameters of the algorithm that we use: the number of process and the number of lowest and highest readed values that the algorithm discards. The defined constants must satisfy this axiom. If not, the algorithm cannot obtain the maximum and minimum value, because it will have discarded all the values.

axiomatization

np :: *nat* — Number of processes **and**

khl :: *nat* — Number of lowest and highest values **where**

constants-ax: $2 * khl < np$

We define also the set of process that the algorithm manage. This definition exist only for readability matters.

definition

PR :: *process set* **where**

[*simp*]: $PR = \{..<np\}$

2.1.3 Convergence function

This functions is called “Fault-tolerant Midpoint” ([7])

In this algorithm each process has an array where it store the clocks readings from the others processes (including itself). We formalise that as a function from processes to clock time as [8].

First we define two functions. They take a function of clock readings and a set of processes and they return a set of *khl* processes which has the greater (smaller) clock readings. They were defined with the Hilbert’s ε -operator (the indefinite description operator *SOME* in Isabelle) because in this way the formalization is not fixed to a particular election of the processes’s readings to discards and then the modelization is more general.

definition

$kmax :: (process \Rightarrow Clocktime) \Rightarrow process\ set \Rightarrow process\ set$ **where**
 $kmax\ f\ P = (SOME\ S. S \subseteq P \wedge card\ S = khl \wedge$
 $(\forall\ i \in S. \forall\ j \in (P - S). f\ j \leq f\ i))$

definition

$kmin :: (process \Rightarrow Clocktime) \Rightarrow process\ set \Rightarrow process\ set$ **where**
 $kmin\ f\ P = (SOME\ S. S \subseteq P \wedge card\ S = khl \wedge$
 $(\forall\ i \in S. \forall\ j \in (P - S). f\ i \leq f\ j))$

With the previus functions we define a new one *reduce*¹. This take a function of clock readings and a set of processes and return de set of readings of the not dicarded processes. In order to define this function we use the image operator (f') of Isabelle.

definition

$reduce :: (process \Rightarrow Clocktime) \Rightarrow process\ set \Rightarrow Clocktime\ set$ **where**
 $reduce\ f\ P = f' (P - (kmax\ f\ P \cup kmin\ f\ P))$

And finally the convergence function. This is defined with the builtin *Max* and *Min* functions of Isabelle.

definition

$cfnl :: process \Rightarrow (process \Rightarrow Clocktime) \Rightarrow Clocktime$ **where**
 $cfnl\ p\ f = (Max (reduce\ f\ PR) + Min (reduce\ f\ PR)) / 2$

2.2 Translation Invariance property.

2.2.1 Auxiliary lemmas

These lemmas proves the existence of the maximum and minimum of the image of a set, if the set is finite and not empty.

lemma *ex-Maxf*:

¹The name of this function was taken from [5].

```

fixes  $S$  and  $f :: 'a \Rightarrow ('b::linorder)$ 
  assumes  $fin: finite\ S$ 
  shows  $S \neq \{\}$   $\implies \exists m \in S. \forall s \in S. f\ s \leq f\ m$ 
using  $fin$ 
proof ( $induct$ )
  case empty thus ?case by simp
next
  case ( $insert\ x\ S$ )
  show ?case
  proof ( $cases$ )
    assume  $S = \{\}$  thus ?thesis by simp
  next
    assume  $nonempty: S \neq \{\}$ 
    then obtain  $m$  where  $m: m \in S \ \forall s \in S. f\ s \leq f\ m$ 
    using  $insert$  by blast
    show ?thesis
    proof ( $cases$ )
      assume  $f\ x \leq f\ m$  thus ?thesis using  $m$  by blast
    next
      assume  $\sim f\ x \leq f\ m$  thus ?thesis using  $m$ 
      by( $simp\ add:linorder-not-le\ order-less-le$ )
      ( $blast\ intro: order-trans$ )
    qed
  qed
qed

```

```

lemma  $ex-Minf$ :
fixes  $S$  and  $f :: 'a \Rightarrow ('b::linorder)$ 
  assumes  $fin: finite\ S$ 
  shows  $S \neq \{\}$   $\implies \exists m \in S. \forall s \in S. f\ m \leq f\ s$ 
using  $fin$ 
proof ( $induct$ )
  case empty thus ?case by simp
next
  case ( $insert\ x\ S$ )
  show ?case
  proof ( $cases$ )
    assume  $S = \{\}$  thus ?thesis by simp
  next
    assume  $nonempty: S \neq \{\}$ 
    then obtain  $m$  where  $m: m \in S \ \forall s \in S. f\ m \leq f\ s$ 
    using  $insert$  by blast
    show ?thesis
    proof ( $cases$ )
      assume  $f\ m \leq f\ x$  thus ?thesis using  $m$  by blast
    next
      assume  $\sim f\ m \leq f\ x$  thus ?thesis using  $m$ 
      by( $simp\ add:linorder-not-le\ order-less-le$ )
      ( $blast\ intro: order-trans$ )
    qed
  qed

```

qed
 qed
 qed

This trivial lemma is needed by the next two.

lemma *khl-bound*: $khl < np$
 using *constants-ax* by *arith*

The next two lemmas prove that *de* functions *kmin* and *kmax* return some values that satisfy their definition. This is not trivial because we need to prove the existence of these values, according to the rule of the Hilbert's operator. We will need this lemma many times because is the only thing that we know about these functions.

lemma *kmax-prop*:

fixes $f :: nat \Rightarrow Clocktime$

shows

$(kmax\ f\ PR) \subseteq PR \wedge card\ (kmax\ f\ PR) = khl \wedge$
 $(\forall i \in (kmax\ f\ PR). \forall j \in PR - (kmax\ f\ PR). f\ j \leq f\ i)$

proof–

have $khl \leq np \longrightarrow$

$(\exists S. S \subseteq PR \wedge card\ S = khl \wedge (\forall i \in S. \forall j \in PR - S. f\ j \leq f\ i))$

$(\text{is } khl \leq np \longrightarrow ?P\ khl)$

proof(*induct* (*khl*))

have $?P\ 0$ **by** *force*

thus $0 \leq np \longrightarrow ?P\ 0 ..$

next

fix n

assume *asm*: $n \leq np \longrightarrow ?P\ n$

show $Suc\ n \leq np \longrightarrow ?P\ (Suc\ n)$

proof

assume *asm2*: $Suc\ n \leq np$

with *asm* **have** $?P\ n$ **by** *simp*

then obtain S **where**

SinPR : $S \subseteq PR$ **and**

cardS: $card\ S = n$ **and**

HI: $(\forall i \in S. \forall j \in PR - S. f\ j \leq f\ i)$

by *blast*

let $?e = SOME\ i. i \in PR - S \wedge$

$(\forall j \in PR - S. f\ j \leq f\ i)$

let $?S = insert\ ?e\ S$

have $\exists i. i \in PR - S \wedge (\forall j \in PR - S. f\ j \leq f\ i)$

proof–

from *SinPR* **and** *finite-subset*

have *finite* ($PR - S$)

by *auto*

moreover

from *cardS* **and** *asm2* *SinPR*

have $S \subseteq PR$ **by** *auto*

hence $PR - S \neq \{\}$ **by** *auto*

ultimately
show *?thesis using ex-Maxf by blast*
qed
hence
ePRS: ?e ∈ PR-S and maxH: (∀j ∈ PR-S. fj ≤ f ?e)
by (*auto dest!: someI-ex*)
from *maxH and HI*
have (*∀i ∈ ?S. ∀j ∈ PR - ?S. fj ≤ fi*)
by *blast*
moreover
from *SinPR and finite-subset*
cardS and ePRS
have *card ?S = Suc n*
by (*auto dest: card-insert-disjoint*)
moreover
have *?S ⊆ PR using SinPR and ePRS by auto*
ultimately
show *?P (Suc n) by blast*
qed
qed
hence *?P khl using khl-bound by auto*
then obtain *S where*
S ⊆ PR ∧ card S = khl ∧ (∀i ∈ S. ∀j ∈ PR - S. fj ≤ fi) ..
thus *?thesis by (unfold kmax-def)*
(rule someI [where P=λS. S ⊆ PR ∧ card S = khl ∧ (∀i ∈ S. ∀j ∈ PR - S.
fj ≤ fi)])
qed

lemma *kmin-prop:*
fixes *f :: nat ⇒ Clocktime*
shows
(kmin f PR) ⊆ PR ∧ card (kmin f PR) = khl ∧
(∀i ∈ (kmin f PR). ∀j ∈ PR - (kmin f PR). fi ≤ fj)

proof–
have *khl ≤ np →*
(∃ S. S ⊆ PR ∧ card S = khl ∧ (∀i ∈ S. ∀j ∈ PR - S. fi ≤ fj))
(is khl ≤ np → ?P khl)
proof(*induct (khl)*)
have *?P 0 by force*
thus *0 ≤ np → ?P 0 ..*
next
fix *n*
assume *asm: n ≤ np → ?P n*
show *Suc n ≤ np → ?P (Suc n)*
proof
assume *asm2: Suc n ≤ np*
with *asm have ?P n by simp*
then obtain *S where*
SinPR : S ⊆ PR and

cardS: $\text{card } S = n$ **and**
HI: $(\forall i \in S. \forall j \in PR - S. f i \leq f j)$
by *blast*
let $?e = \text{SOME } i. i \in PR - S \wedge$
 $(\forall j \in PR - S. f i \leq f j)$
let $?S = \text{insert } ?e S$
have $\exists i. i \in PR - S \wedge (\forall j \in PR - S. f i \leq f j)$
proof–
from *SinPR* **and** *finite-subset*
have *finite* $(PR - S)$
by *auto*
moreover
from *cardS* **and** *asm2 SinPR*
have $S \subseteq PR$ **by** *auto*
hence $PR - S \neq \{\}$ **by** *auto*
ultimately
show *?thesis* **using** *ex-Minf* **by** *blast*
qed
hence
ePRS: $?e \in PR - S$ **and** *minH*: $(\forall j \in PR - S. f ?e \leq f j)$
by *(auto dest!: someI-ex)*
from *minH* **and** *HI*
have $(\forall i \in ?S. \forall j \in PR - ?S. f i \leq f j)$
by *blast*
moreover
from *SinPR* **and** *finite-subset* **and**
cardS **and** *ePRS*
have $\text{card } ?S = \text{Suc } n$
by *(auto dest: card-insert-disjoint)*
moreover
have $?S \subseteq PR$ **using** *SinPR* **and** *ePRS* **by** *auto*
ultimately
show *?P* $(\text{Suc } n)$ **by** *blast*
qed
qed
hence *?P khl* **using** *khl-bound* **by** *auto*
then obtain S **where**
 $S \leq PR \wedge \text{card } S = khl \wedge (\forall i \in S. \forall j \in PR - S. f i \leq f j) ..$
thus *?thesis* **by** *(unfold kmin-def)*
(rule someI [where P= $\lambda S. S \subseteq PR \wedge \text{card } S = khl \wedge (\forall i \in S. \forall j \in PR - S.$
 $f i \leq f j)$])
qed

The next two lemmas are trivial from the previous ones

lemma *finite-kmax*:

finite $(kmax f PR)$

proof–

have *finite PR* **by** *auto*

with *kmax-prop* **and** *finite-subset* **show** *?thesis*

by *blast*
qed

lemma *finite-kmin*:
finite (kmin f PR)

proof –
 have *finite PR* by *auto*
 with *kmin-prop* and *finite-subset* **show** *?thesis*
 by *blast*
qed

This lemma is necessary because the definition of the convergence function use the builtin Max and Min.

lemma *reduce-not-empty*:
reduce f PR \neq {}

proof –
 from *constants-ax* **have**
 $0 < (np - 2 * khl)$ by *arith*
also
 {
 from *kmax-prop kmin-prop*
 have $card (kmax f PR) = khl \wedge card (kmin f PR) = khl$
 by *blast*
 hence $card (kmax f PR \cup kmin f PR) \leq 2 * khl$
 using *card-Un-le[of kmax f PR kmin f PR]* by *simp*
 }
 hence
 $\dots \leq card PR - card (kmax f PR \cup kmin f PR)$
 by *simp*
also
 {
 from *kmax-prop* and *kmin-prop* **have**
 $(kmax f PR \cup kmin f PR) \subseteq PR$ by *blast*
 }
 hence
 $\dots = card (PR - (kmax f PR \cup kmin f PR))$
 apply (*intro card-Diff-subset[THEN sym]*)
 apply (*rule finite-subset*)
 by *auto*

finally
 have $0 < card (PR - (kmax f PR \cup kmin f PR))$.
 hence $(PR - (kmax f PR \cup kmin f PR)) \neq \{\}$
 by (*intro notI, simp only: card-0-eq, simp*)
 thus *?thesis*
 by (*auto simp add: reduce-def*)
qed

The next three are the main lemmas necessary for prove the Translation

Invariance property.

lemma *reduce-shift*:

fixes $f :: nat \Rightarrow Clocktime$

shows

$$f \text{ ' } (PR - (kmax f PR \cup kmin f PR)) = \\ f \text{ ' } (PR - (kmax (\lambda p. f p + c) PR \cup kmin (\lambda p. f p + c) PR))$$

apply (*unfold kmin-def kmax-def*)

by *simp*

lemma *max-shift*:

fixes $f :: nat \Rightarrow Clocktime$ **and** S

assumes *notEmpFin*: $S \neq \{\}$ *finite* S

shows

$$Max (f'S) + x = Max (\lambda p. f p + x) \text{ ' } S$$

proof –

from *notEmpFin* **have** $f'S \neq \{\}$ **and** $(\lambda p. f p + x) \text{ ' } S \neq \{\}$

by *auto*

with *notEmpFin* **have**

$$Max (f'S) \in f \text{ ' } S \quad Max ((\lambda p. f p + x) \text{ ' } S) \in (\lambda p. f p + x) \text{ ' } S \\ (\forall fs \in (f'S). fs \leq Max (f'S))$$

$$(\forall fs \in ((\lambda p. f p + x) \text{ ' } S). fs \leq Max ((\lambda p. f p + x) \text{ ' } S))$$

by *auto*

thus *?thesis* **by** *force*

qed

lemma *min-shift*:

fixes $f :: nat \Rightarrow Clocktime$ **and** S

assumes *notEmpFin*: $S \neq \{\}$ *finite* S

shows

$$Min (f'S) + x = Min (\lambda p. f p + x) \text{ ' } S$$

proof –

from *notEmpFin* **have** $f'S \neq \{\}$ **and** $(\lambda p. f p + x) \text{ ' } S \neq \{\}$

by *auto*

with *notEmpFin* **have**

$$Min (f'S) \in f \text{ ' } S \quad Min ((\lambda p. f p + x) \text{ ' } S) \in (\lambda p. f p + x) \text{ ' } S \\ (\forall fs \in (f'S). Min (f'S) \leq fs)$$

$$(\forall fs \in ((\lambda p. f p + x) \text{ ' } S). Min ((\lambda p. f p + x) \text{ ' } S) \leq fs)$$

by *auto*

thus *?thesis* **by** *force*

qed

2.2.2 Main theorem

theorem *trans-inv*:

fixes $f :: nat \Rightarrow Clocktime$

shows

$$cfnl p f + x = cfnl p (\lambda p. f p + x)$$

proof –

have $cfnl p (\lambda p. f p + x) =$

```

      (Max (reduce (λ p. f p + x) PR) + Min (reduce (λ p. f p + x) PR)) / 2
    by (unfold cfnl-def, simp)
  also
  have ... =
    (Max ((λ p. f p + x) ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))) +
      Min ((λ p. f p + x) ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR)))) / 2
    by (unfold reduce-def, simp)
  also
  have
  ... =
    (Max (f ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))) + x +
      Min (f ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))) + x) / 2
  proof-
  have finite (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))
    by auto
  moreover
  from reduce-not-empty have
    PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR) ≠ {}
    by (auto simp add: reduce-def)
  ultimately
  have
    Max ((λ p. f p + x) ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR)))
    =
    Max (f ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))) + x
    and
    Min ((λ p. f p + x) ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR)))
    =
    Min (f ‘
      (PR - (kmax (λ p. f p + x) PR ∪ kmin (λ p. f p + x) PR))) + x
    using max-shift and min-shift
    by auto
  thus ?thesis by auto
qed
also
from reduce-shift
have
... =
(Max (f ‘
  (PR - (kmax f PR ∪ kmin f PR))) + x +
  Min (f ‘
  (PR - (kmax f PR ∪ kmin f PR))) + x) / 2
by auto

```



```

also
have ... = ((Max (reduce f PR) + x) + (Min (reduce f PR) + x)) / 2
  by (auto simp add: reduce-def)
also
have ... = (Max (reduce f PR) + Min (reduce f PR)) / 2 + x
  by auto
finally
show ?thesis by (auto simp add: cfnl-def)
qed

```

2.3 Precision Enhancement property

An informal proof of this theorem can be found in [6]

2.3.1 Auxiliary lemmas

This first lemma is most important for prove the property. This is a consequence of the *card-Un-Int* lemma

```

lemma pigeonhole:
assumes
  finitA: finite A and
  Bss: B ⊆ A and Css: C ⊆ A and
  cardH: card A + k ≤ card B + card C
shows k ≤ card (B ∩ C)
proof -
  from Bss Css have B ∪ C ⊆ A by blast
  with finitA have card (B ∪ C) ≤ card A
    by (simp add: card-mono)
  with cardH have
    h: k ≤ card B + card C - card (B ∪ C)
    by arith
  from finitA Bss Css and finite-subset
  have finite B ∧ finite C by auto
  thus ?thesis
    using card-Un-Int and h by force
qed

```

This lemma is a trivial consequence of the previous one. With only this lemma we can prove the Precision Enhancement property with the bound $\pi(x, y) = x + y$. But this bound not satisfy the property

$$\pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S$$

that is used in [8] for prove the Schneider's schema.

```

lemma subsets-int:
assumes
  finitA: finite A and
  Bss: B ⊆ A and Css: C ⊆ A and

```

```

  cardH: card A < card B + card C
shows
  B ∩ C ≠ {}
proof–
  from finitA Bss Css cardH
  have 1 <= card (B ∩ C)
    by (auto intro!: pigeonhole)
  thus ?thesis by auto
qed

```

This lemma is true because $reduce\ f\ PR$ is the image of $PR - (kmax\ f\ PR \cup kmin\ f\ PR)$ by the function f .

```

lemma exist-reduce:
  ∀ c ∈ reduce f PR. ∃ i ∈ PR-(kmax f PR ∪ kmin f PR). f i = c
proof
fix c assume asm: c ∈ reduce f PR
thus ∃ i ∈ PR-(kmax f PR ∪ kmin f PR). f i = c
  by (auto simp add: reduce-def kmax-def kmin-def)
qed

```

The next three lemmas are consequence of the definition of $reduce$, $kmax$ and $kmin$

```

lemma finite-reduce:
  finite (reduce f PR)
proof(unfold reduce-def)
  show finite (f ‘ (PR - (kmax f PR ∪ kmin f PR)))
    by auto
qed

```

```

lemma kmax-ge:
  ∀ i ∈ (kmax f PR). ∀ r ∈ (reduce f PR). r <= f i
proof
fix i assume asm: i ∈ kmax f PR
show ∀ r ∈ reduce f PR. r ≤ f i
proof
fix r assume asm2: r ∈ reduce f PR
show r ≤ f i
proof–
  from asm2 and exist-reduce have
    ∃ j ∈ PR-(kmax f PR ∪ kmin f PR). f j = r by blast
  then obtain j
  where fjr:j ∈ PR-(kmax f PR ∪ kmin f PR) ∧ f j = r
    by blast
  hence j ∈ (PR - kmax f PR)
    by blast
  from this fjr asm
  show ?thesis using kmax-prop
    by auto
qed

```

qed
qed

lemma *kmin-le*:

$\forall i \in (kmin\ f\ PR). \forall r \in (reduce\ f\ PR). f\ i \leq r$

proof

fix *i* **assume** *asm*: $i \in kmin\ f\ PR$

show $\forall r \in reduce\ f\ PR. f\ i \leq r$

proof

fix *r* **assume** *asm2*: $r \in reduce\ f\ PR$

show $f\ i \leq r$

proof

from *asm2* **and** *exist-reduce* **have**

$\exists j \in PR - (kmax\ f\ PR \cup kmin\ f\ PR). f\ j = r$ **by** *blast*

then obtain *j*

where *fjr*: $j \in PR - (kmax\ f\ PR \cup kmin\ f\ PR) \wedge f\ j = r$

by *blast*

hence $j \in (PR - kmin\ f\ PR)$

by *blast*

from *this fjr asm*

show *?thesis* **using** *kmin-prop*

by *auto*

qed

qed

qed

The next lemma is used for prove the Precision Enhancement property. This has been proved in ICS. The proof is in the appendix A.1. This cannot be prove by a simple *arith* or *auto* tactic.

This lemma is true also with $0 \leq c$!!

lemma *abs-distrib-div*:

$0 < (c::real) \implies |a / c - b / c| = |a - b| / c$

proof

assume *ch*: $0 < c$

{

fix *d* :: *real*

assume *dh*: $0 \leq d$

have $a * d - b * d = (a - b) * d$

by (*simp add: algebra-simps*)

hence $|a * d - b * d| = |(a - b) * d|$

by *simp*

also with *dh* **have**

$\dots = |a - b| * d$

by (*simp add: abs-mult*)

finally

have $|a * d - b * d| = |a - b| * d$

.

```

}
with ch and divide-inverse show ?thesis
  by (auto simp add: divide-inverse)
qed

```

The next three lemmas are about the existence of bounds of the values Max ($reduce\ f\ PR$) and Min ($reduce\ f\ PR$). These are used in the proof of the main property.

lemma *uboundmax*:

assumes

hC: $C \subseteq PR$ **and**

hCk: $np \leq card\ C + khl$

shows

$\exists i \in C. Max\ (reduce\ f\ PR) \leq f\ i$

proof–

from *reduce-not-empty* **and** *finite-reduce*

have *maxrinr*: $Max\ (reduce\ f\ PR) \in reduce\ f\ PR$

by *simp*

with *exist-reduce*

have $\exists i \in PR - (kmax\ f\ PR \cup kmin\ f\ PR). f\ i = Max\ (reduce\ f\ PR)$

by *simp*

then obtain *pmax* **where**

pmax-in-reduc: $pmax \in PR - (kmax\ f\ PR \cup kmin\ f\ PR)$ **and**

fpmax-ismax: $f\ pmax = Max\ (reduce\ f\ PR)$..

hence $C \cap insert\ pmax\ (kmax\ f\ PR) \neq \{\}$

proof–

from *kmax-prop* **and** *pmax-in-reduc*

and *finite-kmax* **and** *hCk* **have**

$card\ PR < card\ C + card\ (insert\ pmax\ (kmax\ f\ PR))$

by *simp*

moreover

from *pmax-in-reduc* **and** *kmax-prop*

have $insert\ pmax\ (kmax\ f\ PR) \subseteq PR$ **by** *blast*

moreover

note *hC*

ultimately

show *?thesis*

using *subsets-int[of PR C insert pmax (kmax f PR)]*

by *simp*

qed

hence *res*: $\exists i \in C. i = pmax \vee i \in kmax\ f\ PR$ **by** *blast*

then obtain *i* **where**

iinC: $i \in C$ **and** *altern*: $i = pmax \vee i \in kmax\ f\ PR$..

thus *?thesis*

proof(*cases i=pmax*)

case *True*

with *iinC fpmax-ismax* **show** *?thesis* **by** *force*

next

```

    case False
    with altern maxrnr fpmax-ismax kmax-ge
    have f pmax <= f i by simp
    with iinC fpmax-ismax show ?thesis by auto
  qed
qed

lemma lboundmin:
  assumes
    hC:  $C \subseteq PR$  and
    hCk:  $np \leq \text{card } C + khl$ 
  shows
     $\exists i \in C. f i \leq \text{Min } (\text{reduce } f PR)$ 
  proof -
    from reduce-not-empty and finite-reduce
    have minrnr:  $\text{Min } (\text{reduce } f PR) \in \text{reduce } f PR$ 
      by simp
    with exist-reduce
    have  $\exists i \in PR - (kmax f PR \cup kmin f PR). f i = \text{Min } (\text{reduce } f PR)$ 
      by simp
    then obtain pmin where
      pmin-in-reduc:  $pmin \in PR - (kmax f PR \cup kmin f PR)$  and
      fpmin-ismin:  $f pmin = \text{Min } (\text{reduce } f PR)$  ..
    hence  $C \cap \text{insert } pmin (kmin f PR) \neq \{\}$ 
    proof -
      from kmin-prop and pmin-in-reduc
      and finite-kmin and hCk have
         $\text{card } PR < \text{card } C + \text{card } (\text{insert } pmin (kmin f PR))$ 
      by simp
      moreover
      from pmin-in-reduc and kmin-prop
      have  $\text{insert } pmin (kmin f PR) \subseteq PR$  by blast
      moreover
      note hC
      ultimately
      show ?thesis
        using subsets-int[of PR C insert pmin (kmin f PR)]
        by simp
    qed
    hence res:  $\exists i \in C. i = pmin \vee i \in kmin f PR$  by blast
    then obtain i where
      iinC:  $i \in C$  and altern:  $i = pmin \vee i \in kmin f PR$  ..
    thus ?thesis
    proof (cases i = pmin)
      case True
      with iinC fpmin-ismin show ?thesis by force
    next
      case False
      with altern minrnr fpmin-ismin kmin-le

```

```

    have  $f\ i \leq f\ pmin$  by simp
    with inC f pmin-ismin show ?thesis by auto
  qed
qed

lemma same-bound:
assumes
   $hC: C \subseteq PR$  and
   $hCk: np \leq card\ C + khl$  and
   $hnk: 3 * khl < np$ 
shows
   $\exists i \in C. Min\ (reduce\ f\ PR) \leq f\ i \wedge g\ i \leq Max\ (reduce\ g\ PR)$ 
proof-
  have b1:  $khl + 1 \leq card\ (C \cap (PR - kmin\ f\ PR))$ 
  proof(rule pigeonhole)
    show finite PR by simp
  next
    show  $C \subseteq PR$  by fact
  next
    show  $PR - kmin\ f\ PR \subseteq PR$  by blast
  next
    show  $card\ PR + (khl + 1) \leq card\ C + card\ (PR - kmin\ f\ PR)$ 
  proof-
    from hnk and hCk have
       $np + khl < np + card\ C - khl$  by arith
    also
      from kmin-prop
      have  $\dots = np + card\ C - card\ (kmin\ f\ PR)$ 
      by auto
    also
      have  $\dots = card\ C + (card\ PR - card\ (kmin\ f\ PR))$ 
    proof-
      from kmin-prop have
         $card\ (kmin\ f\ PR) \leq card\ PR$ 
        by (intro card-mono, auto)
      thus ?thesis by (simp)
    qed
  also
    from kmin-prop
    have  $\dots = card\ C + card\ (PR - kmin\ f\ PR)$ 
  proof-
    from kmin-prop and finite-kmin have
       $card\ PR - card\ (kmin\ f\ PR) = card\ (PR - kmin\ f\ PR)$ 
      by (intro card-Diff-subset[THEN sym])(auto)
    thus ?thesis by auto
  qed
  finally
  show ?thesis
    by (simp)

```

```

qed
qed

have  $C \cap (PR - kmin\ f\ PR) \cap (PR - kmax\ g\ PR) \neq \{\}$ 
proof(intro subsets-int)
  show finite PR by simp
next
  show  $C \cap (PR - kmin\ f\ PR) \subseteq PR$ 
  by blast
next
  show  $PR - kmax\ g\ PR \subseteq PR$ 
  by blast
next
  show  $card\ PR <$ 
     $card\ (C \cap (PR - kmin\ f\ PR)) + card\ (PR - kmax\ g\ PR)$ 
  proof-
    from kmax-prop and finite-kmax
    have  $card\ (PR - kmax\ g\ PR) = card\ PR - card\ (kmax\ g\ PR)$ 
    by (intro card-Diff-subset, auto)
    with kmax-prop have
       $card\ (PR - kmax\ g\ PR) = card\ PR - khl$  by simp
    with b1
    show ?thesis by arith
  qed
qed

hence
   $\exists i. i \in C \wedge i \in (PR - kmin\ f\ PR) \wedge i \in (PR - kmax\ g\ PR)$ 
  by blast
then obtain i where
  in-C: i ∈ C and
  not-in-kmin: i ∈ (PR - kmin f PR) and
  not-in-kmax: i ∈ (PR - kmax g PR) by blast
have  $Min\ (reduce\ f\ PR) \leq f\ i$ 
proof(cases i ∈ kmax f PR)
  case True
    from reduce-not-empty and finite-reduce have
       $Min\ (reduce\ f\ PR) \in reduce\ f\ PR$  by auto
    with True show ?thesis
    using kmax-ge by blast
  next
    case False
    with not-in-kmin
    have  $i \in PR - (kmax\ f\ PR \cup kmin\ f\ PR)$ 
    by blast
    with reduce-def have f i ∈ reduce f PR
    by auto
    with reduce-not-empty and finite-reduce
    show ?thesis by auto

```

```

qed
moreover
have  $g\ i \leq \text{Max}(\text{reduce } g\ PR)$ 
proof(cases  $i \in \text{kmin } g\ PR$ )
  case True
    from reduce-not-empty and finite-reduce have
       $\text{Max}(\text{reduce } g\ PR) \in \text{reduce } g\ PR$  by auto
    with True show ?thesis
    using kmin-le by blast
  next
    case False
    with not-in-kmax
    have  $i \in PR - (\text{kmax } g\ PR \cup \text{kmin } g\ PR)$ 
      by blast
    with reduce-def have  $g\ i \in \text{reduce } g\ PR$ 
      by auto
    with reduce-not-empty and finite-reduce
    show ?thesis by auto
qed
moreover
note in-C
ultimately
show ?thesis by blast
qed

```

2.3.2 Main theorem

The most part of this theorem can be proved with CVC-lite using the three previous lemmas (appendix A.2).

theorem *prec-enh*:

assumes

```

hC:  $C \subseteq PR$  and
hCF:  $np - nF \leq \text{card } C$  and
hFn:  $3 * nF < np$  and
hFk:  $nF = \text{khl}$  and
hbx:  $\forall l \in C. |f\ l - g\ l| \leq x$  and
hby1:  $\forall l \in C. \forall m \in C. |f\ l - f\ m| \leq y$  and
hby2:  $\forall l \in C. \forall m \in C. |g\ l - g\ m| \leq y$  and
hpC:  $p \in C$  and
hqC:  $q \in C$ 

```

shows $|\text{cfnl } p\ f - \text{cfnl } q\ g| \leq y / 2 + x$

proof –

```

from hCF and hFk
have hCk:  $np \leq \text{card } C + \text{khl}$  by arith
from hFn and hFk
have hnk:  $3 * \text{khl} < np$  by arith
let ?maxf =  $\text{Max}(\text{reduce } f\ PR)$ 
  and ?minf =  $\text{Min}(\text{reduce } f\ PR)$ 
  and ?maxg =  $\text{Max}(\text{reduce } g\ PR)$ 

```


and $?ming = Min$ (reduce g PR)
from *abs-distrib-div*
have $|cfnl\ p\ f - cfnl\ q\ g| =$
 $|?maxf + ?minf + - ?maxg + - ?ming| / 2$
by (*unfold cfnl-def*) *simp*
moreover
have $|?maxf + ?minf + - ?maxg + - ?ming| \leq y + 2 * x$
— The rest of the property can be proved by CVC-lite (see appendix A.2)
proof (*cases* $0 \leq ?maxf + ?minf + - ?maxg + - ?ming$)
case *True*
hence
 $|?maxf + ?minf + - ?maxg + - ?ming| =$
 $?maxf + ?minf + - ?maxg + - ?ming$ **by** *arith*
moreover
from *uboundmax hC hCk*
obtain mxf
where $mxf \in C$ **and**
 $mxf: ?maxf \leq f\ mxf$ **by** *blast*
moreover
from *lboundmin hC hCk*
obtain mng
where $mng \in C$ **and**
 $mng: g\ mng \leq ?ming$ **by** *blast*
moreover
from *same-bound hC hCk hnk*
obtain $m xn$
where $m xn \in C$ **and**
 $m xn f: ?minf \leq f\ m xn$ **and**
 $m xn g: g\ m xn \leq ?maxg$ **by** *blast*
ultimately
have
 $|?maxf + ?minf + - ?maxg + - ?ming| \leq$
 $(f\ mxf + - g\ mng) + (f\ m xn + - g\ m xn)$ **by** *arith*
also
from $m xn \in C$ *hbx abs-le-D1*
have
 $\dots \leq (f\ mxf + - g\ mng) + x$
by *auto*
also
have
 $\dots = (f\ mxf + - f\ mng) + (f\ mng + - g\ mng) + x$
by *arith*
also
have $\dots \leq y + (f\ mng + - g\ mng) + x$
proof—
from $mxf \in C$ $mng \in C$ *hby1 abs-le-D1*
have $f\ mxf + - f\ mng \leq y$
by *auto*
thus $?thesis$

```

    by auto
qed
also
from mnginC hbx abs-le-D1
have ... <= y + 2 * x
    by auto
finally
show ?thesis .
next
case False
hence
|?maxf + ?minf + - ?maxg + - ?ming| =
  ?maxg + ?ming + - ?maxf + - ?minf by arith
moreover
from uboundmax hC hCk
obtain mxg
  where mxginC: mxg ∈ C and
    mxg: ?maxg <= g mxg by blast
moreover
from lboundmin hC hCk
obtain mnf
  where mnfinC: mnf ∈ C and
    mnf: f mnf <= ?minf by blast
moreover
from same-bound hC hCk hnk
obtain mxn
  where mxninC: mxn ∈ C and
    mxnf: ?ming ≤ g mxn and
    mxng: f mxn ≤ ?maxf by blast
ultimately
have
  |?maxf + ?minf + - ?maxg + - ?ming| <=
    (g mxg + - f mnf) + (g mxn + - f mxn) by arith
also
from mxninC hbx
have ... <= (g mxg + - f mnf) + x
  by (auto dest!: abs-le-D2)
also
have
  ... = (g mxg + - g mnf) + (g mnf + - f mnf) + x
  by arith
also
have ... <= y + (g mnf + - f mnf) + x
proof-
from mxginC mnfinC hby2 abs-le-D1
have g mxg + - g mnf <= y
  by auto
thus ?thesis
  by auto

```

```

qed
also
from mnfinC hbx
have ... <=  $y + 2 * x$ 
  by (auto dest!: abs-le-D2)
finally
show ?thesis .
qed
ultimately
show ?thesis
  by simp
qed

```

2.4 Accuracy Preservation property

No new lemmas are needed for prove this property. The bound has been found using the lemmas *uboundmax* and *lboundmin*

This theorem can be proved with ICS and CVC-lite assuming those lemmas (see appendix A.3).

theorem *accur-pres:*

assumes

hC: C ⊆ PR and

hCF: np - nF <= card C and

hFk: nF = khl and

hby: ∀ l ∈ C. ∀ m ∈ C. |f l - f m| <= y and

hqC: q ∈ C

shows $|cfnl\ p\ f - f\ q| <= y$

proof –

from *hCF and hFk*

have *npleCk: np <= card C + khl* **by** *arith*

show *?thesis*

proof(*cases f q <= cfnl p f*)

case *True*

from *npleCk hC and uboundmax*

have $\exists i \in C. Max\ (reduce\ f\ PR) <= f\ i$

by *auto*

then obtain *pi* **where**

hpiC: pi ∈ C and

fpiGeMax: Max (reduce f PR) <= f pi **by** *blast*

from *reduce-not-empty*

have $Min\ (reduce\ f\ PR) <= Max\ (reduce\ f\ PR)$

by (*auto simp add: reduce-def*)

with *fpiGeMax* **have**

cfnlLefpi: cfnl p f <= f pi

by (*auto simp add: cfnl-def*)

with *True* **have**

$|cfnl\ p\ f - f\ q| <= |f\ pi - f\ q|$

by *arith*

```

with hpiC and hqC and hby show ?thesis
  by force
next
  case False
  from npleCk hC and lboundmin
  have  $\exists i \in C. f i \leq \text{Min} (\text{reduce } f PR)$ 
    by auto
  then obtain qi where
    hqiC: qi  $\in C$  and
    fqiLeMax: f qi  $\leq \text{Min} (\text{reduce } f PR)$  by blast
  from reduce-not-empty
  have  $\text{Min} (\text{reduce } f PR) \leq \text{Max} (\text{reduce } f PR)$ 
    by (auto simp add: reduce-def)
  with fqiLeMax
  have f qi  $\leq \text{cfnl } p f$ 
    by (auto simp add: cfnl-def)
  with False have
     $|\text{cfnl } p f - f q| \leq |f qi - f q|$ 
    by arith
  with hqiC and hqC and hby show ?thesis
    by force
qed
qed

end

```

A CVC-lite and ICS proofs

A.1 Lemma abs_distrib_div

In the proof of the Fault-Tolerant Mid Point Algorithm we need to prove this simple lemma:

lemma *abs-distrib-div*:

$$0 < (c::real) \implies |a / c - b / c| = |a - b| / c$$

It is not possible to prove this lemma in Isabelle using *arith* nor *auto* tactics. Even if we added lemmas to the default simpset of HOL.

In the translation from Isabelle to ICS we need to change the division by a multiplication because this tools do not accept formulas with this arithmetic operator. Moreover, to translate the absolute value we define e constant for each application of that function. In ICS it is proved automatically.

File `abs_distrib_mult.ics`:

It was not possible to find the proof in CVC-lite because the formula is not linear. Two proofs where attempted. In the first one we use lambda abstraction to define the absolute value. The second one is the same translation that we do in ICS.

File `abs_distrib_mult.cvc`:

File `abs_distrib_mult2.cvc`:

A.2 Bound for Precision Enhancement property

In order to prove Precision Enhancement for Lynch's algorithm we need to prove that:

$$\mathbf{have} \ |Max (reduce f PR) + Min (reduce f PR) + \\ - Max (reduce g PR) + - Min (reduce g PR)| \leq y + 2 * x$$

This is the result of the whole theorem where we multiply by two both sides of the inequality.

In order to do the proof we need to translate also the lemmas *uboundmax*, *lboundmin*, *same_bound* (lemmas about the existence of some bounds), the axiom *constants_ax* and the assumptions of the theorem.

We make five different translations. In each one we were increasing the amount of eliminated quantifiers.

File `bound_prec_enh4.cvc`:

Note that we leave quantifiers in some assumptions.

In the next file we also try to do the proof with all quantifiers, but CVC cannot find it.

File `bound_prec_enh.cvc`:

We also try to do the proof removing all quantifiers and the proof was successful.

File `bound_prec_enh7.cvc`:

From this last file we make the translation also for ICS adding a constant for each application of the absolute value. In this case ICS do not find the proof.

File `bound_prec_enh.ics`:

A.3 Accuracy Preservation property

The proof of this property was successful in both tools. Even in CVC-lite the proof was found without the need of removing the quantifiers.

File `accur_pres.cvc`:

File `accur_pres.ics`:

References

- [1] D. Barsotti, L. P. Nieto, and A. Tiu. Verification of clock synchronization algorithms: Experiments on a combination of deductive tools. In *Proceedings of AVOCS 2005*, volume 145 of *ENTCS*, pages 63–68. Elsevier

Science B. V., 2005. Available by WWW from <http://www.cs.famaf.unc.edu.ar/~damian/publications/clock.pdf>.

- [2] CVC Lite home page. <http://verify.stanford.edu/CVCL/>, 2006.
- [3] ICS: Integrated Canonizer and Solver home page. <http://ics.csl.sri.com/>, 2006.
- [4] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *J. ACM*, 32(1):52–78, 1985.
- [5] J. Lundelius and N. Lynch. A new fault-tolerant algorithm for clock synchronization. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 75–88, New York, NY, USA, 1984. ACM Press.
- [6] P. S. Miner. Verification of fault-tolerant clock synchronization systems. NASA Technical Paper 3349, NASA Langley Research Center, November 1993.
- [7] F. B. Schneider. Understanding protocols for Byzantine clock synchronization. Technical Report TR 87–859, Cornell University, Dept. of Computer Science, Upson Hall, Ithaca, NY 14853, 1987.
- [8] N. Shankar. Mechanical verification of a generalized protocol for byzantine fault tolerant clock synchronization. In J. Vytupil, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 571 of *Lecture Notes in Computer Science*, pages 217–236, Nijmegen, The Netherlands, jan 1992. Springer-Verlag.