Instances of Schneider’s generalized protocol of clock synchronization.

Damian Barsotti

August 28, 2014

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://afp.sourceforge.net/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 ... 24
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 abs_distrib_div ... 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**
\[
\begin{align*}
np &:: \text{nat} \quad \text{— Number of processes and} \\
\Delta &:: \text{Clocktime} \quad \text{— Fix value to discard processes} \\
\text{constants-ax:} &
\begin{align*}
0 &< \Delta \land np > 0
\end{align*}
\end{align*}
\]

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

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

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 the reading of the second process if the difference of the readings is grater than \( \Delta \), otherwise it returns the reading of the first one.

**definition**
\[
\begin{align*}
fiX &:: [(\text{process} \Rightarrow \text{Clocktime}), \text{process}, \text{process}] \Rightarrow \text{Clocktime where} \\
fiX f p l &= (\text{if } |f p - f l| <= \Delta \text{ then } (f l) \text{ else } (f p))
\end{align*}
\]

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**
\[
\begin{align*}
cfni &:: [\text{process}, (\text{process} \Rightarrow \text{Clocktime})] \Rightarrow \text{Clocktime where} \\
cfni p f &= (\sum_{l \in \{..<np\}}. fiX f p l) / (\text{real } np)
\end{align*}
\]

1.2 Translation Invariance property.

We first need to prove this auxiliary lemma.

**lemma** trans-inv’:
\[
\sum_{l \in \{..<np\}}. fX (\lambda y. f y + x) p l) = \\
(\sum_{l \in \{..<np\}}. fX f p l) + real np' * x
\]

apply (induct-tac np')
apply (auto simp add: cfni-def fX-def real-of-nat-Suc
distrib-right lessThan-Suc)
done

**1.3 Precision Enhancement property**

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

1.3.1 Auxiliary lemmas

**lemma finitC:**
\[ C \subseteq PR \Rightarrow \text{finite } C \]
proof -
assume \( C \subseteq PR \)
thus \( ?\text{thesis using finite-subset by auto} \)
qed

**lemma finitnpC:**
\[ \text{finite } (PR - C) \]
proof -
show \( ?\text{thesis using finite-Diff by auto} \)
qed

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

**lemma sum-abs-triangle-ineq:**
\[ \text{finite } S \Rightarrow \\
|\sum_{l \in S}. (f::'a \Rightarrow 'b::linordered-idom) l| \leq (\sum_{l \in S}. |f l|) \]
(is 
\[ \Rightarrow \ ?P S \] 
by (rule setsum-abs)

**lemma sum-le:**
\[
[\text{finite } S ; \forall r \in S. f r \leq b ] \\
\Rightarrow \\
(\sum_{l \in S}. f l) \leq real (\text{card } S) * b \\
(is [[ \text{finite } S ; \forall r \in S. f r \leq b ] \Rightarrow \ ?P S ])
\]
proof(induct S rule: finite-induct)
show \( ?P \{\} \) by simp
next
fix \( F \) \( x \)
assume \( \text{finit: finite } F \) and \( x \notin F \) and
\[
\text{HI1: } \forall r \in F. \ f r \leq b \implies \text{setsum } f F \leq \text{real } (\text{card } F) \ast b
\]
and \( \text{HI2: } \forall r \in \text{insert } x F. \ f r \leq b \)
from \( \text{HI1 } \) \( \text{HI2 } \) \( \text{finit } \) and \( x \notin F \)
have \( \text{setsum } f \ (\text{insert } x F) \leq b + \text{real } (\text{card } F) \ast b \)
by auto
also
have \( \ldots = \text{real } (\text{card } F) \ast b \)
by \( (\text{simp add: distrib-right real-of-nat-Suc}) \)
also
from \( \text{finit } x \notin F \) have \( \ldots = \text{real } (\text{card } F) \ast b \)
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 \( \text{finitC[where } C=C] \)
moreover
note \( \text{finitnpC[where } C=C] \)
moreover
have \( C \cap (\{..<np\} - C) = \{\} \) by auto
moreover
from \( hC \) have \( C \cup (\{..<np\} - C) = \{..<np\} \) by auto
ultimately
show \( ?\text{thesis} \)
using \( \text{setsum.union-disjoint[where } A=C \text{ and } B=\{..<np\} - C] \)
by auto
qed

lemma abs-sum-np-ineq:
assumes
\( hC: C \subseteq PR \)
shows
\[ |(\sum l \in \{..<np\}, \ (f::nat \Rightarrow \text{real}) l)| \leq
(\sum l \in C, \ |f l|) + (\sum l \in (\{..<np\} - C), \ |f l|) \]
(is \( ?\text{abs-sum} \leq ?\text{sumC + sumnpC} \))
proof –
from \( hC \) and \( \text{sum-np-eq[where } f=f] \)
have \( ?\text{abs-sum} = |(\sum l \in C, \ f l) + (\sum l \in (\{..<np\} - C), \ f l)| \)
(is \( ?\text{abs-sum} = |?\text{sumC'} + ?\text{sumnpC'}| \))
by simp
also
from \( \text{abs-triangle-ineq} \)
have ...\(\leq |?\text{sumC}| + |?\text{sumnpC}|\).
also
have ... \(\leq \text{sumC} + \text{sumnpC}\)
proof -
  from \(hC\) finitC sum-abs-triangle-ineq
have \(|?\text{sumC}| \leq \text{sumC}\) by blast
moreover
from finitnpC and
  sum-abs-triangle-ineq[where \(f=f\) and \(S=PR-C\)]
have \(|?\text{sumnpC}| \leq \text{sumnpC}\)
  by force
ultimately
show \(?\text{thesis}\) by arith
qed
finally
show \(?\text{thesis}\).
qed

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

lemma \(\text{fiX-ubound}\):
\(\text{fiX f p l} \leq f p + \Delta\)
proof(cases \(|f p - f l| \leq \Delta\))
  assume \(asm\): \(|f p - f l| \leq \Delta\)
  hence \(\text{fiX f p l} = f l\) by (simp add: fiX-def)
also
from \(asm\) have \(f l \leq f p + \Delta\) by arith
finally
show \(?\text{thesis}\) by arith
next
  assume \(asm\): \(\neg|f p - f l| \leq \Delta\)
  hence \(\text{fiX f p l} = f p\) by (simp add: fiX-def)
also
from \(asm\) and \(\text{constants-ax}\) have \(f p \leq f p + \Delta\) by arith
finally
show \(?\text{thesis}\) by arith
qed

lemma \(\text{fiX-lbound}\):
\(f p - \Delta \leq \text{fiX f p l}\)
proof(cases \(|f p - f l| \leq \Delta\))
  assume \(asm\): \(|f p - f l| \leq \Delta\)
  hence \(\text{fiX f p l} = f l\) by (simp add: fiX-def)
also
from \(asm\) have \(f p - \Delta \leq f l\) by arith
finally
show \(?\text{thesis}\) by arith
next
  assume \(asm\): \(\neg|f p - f l| \leq \Delta\)
with constants-ax have \( f \ p - \Delta \leq f \ p \) by arith
also from asm have \( f \ p = fX \ p \ l \) by (simp add: fX-def)
finally show ?thesis by arith
qed

lemma abs-fiX-bound: \(|fX \ p \ l - f \ p| \leq \Delta\)
proof-
have \( f \ p - \Delta \leq f \ p \ l \) by arith
with fX-lbound fX-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 \( |fX \ p \ r - fX \ g \ q \ r| \leq 2 \ * \Delta + x + y \)
proof-
have \( |fX \ f \ p \ r - fX \ g \ q \ r| = \)
\( |fX \ f \ p \ r - f \ p + f \ p - fX \ g \ q \ r| \)
by auto
also have \( ... \leq |fX \ f \ p \ r - f \ p| + |f \ p - fX \ g \ q \ r| \)
by arith
also from abs-fiX-bound
have \( ... \leq \Delta + |f \ p - fX \ g \ q| \)
by simp
also
have \( ... = \Delta + |f \ p - g \ q + (g \ q - fX \ g \ q \ r)| \)
by simp
also from abs-triangle-ineq[where \( a = f \ p - g \ q \) and
\( b = g \ q - fX \ g \ q \ r \)]
have \( ... \leq \Delta + |f \ p - g \ q| + |g \ q - fX \ g \ q \ r| \)
by simp
also have \( ... = \Delta + |f \ p - g \ q| + |fX \ g \ q \ r - g \ q| \)
by arith
also from abs-fiX-bound
have \( ... \leq 2 \ * \Delta + |f \ p - g \ q| \)
by simp
also
have ... = 2 * \Delta + |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 2 * \Delta + |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: \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 + y
proof(cases |f p - f r| \leq \Delta)
case True
  note outer-IH = True
  show ?thesis
  proof(cases |g q - g r| \leq \Delta)
    case True
    show ?thesis
    proof
      from hpC and hby1 have 0 \leq y by force
      with hrC and hbx have |f r - g r| \leq 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 ... \leq |f r - f q| + |f q - g q|
      by arith
  qed
also
from hbx hby1 hpC hqC hrC have ... \leq x + y by force
finally
  show ?thesis.
qed
qed
next
case False
note outer-IH = False
show ?thesis
proof (cases \(| g \cdot q - g \cdot r| \leq \Delta\))
case True
show ?thesis
proof –
  from outer-IH and True
  have \(|\text{\(f\)} \cdot p \cdot r - \text{\(f\)} \cdot q \cdot r| = \(| f \cdot p - g \cdot r|\)
    by (auto simp add: \(f\)-def)
  also
  have "... = \(| f \cdot p - f \cdot r + f \cdot r - g \cdot r| \) by simp
also
from abs-triangle-ineq[where a = f \cdot p - f \cdot r and
    b = f \cdot r - g \cdot r] have ... \leq \(| f \cdot p - f \cdot r| + \(| f \cdot r - g \cdot r| \)
    by auto
also
from hbx hby1 hpC hqC hrC have ... \leq x + y by force
finally
  show ?thesis.
qed
next
case False
show ?thesis
proof –
  from outer-IH and False
  have \(|f\cdot p \cdot r - f\cdot q \cdot q| = \(| f \cdot p - g \cdot q|\)
    by (auto simp add: \(f\)-def)
  also
  have "... = \(| f \cdot p - f \cdot q + f \cdot q - g \cdot q| \) by simp
also
from abs-triangle-ineq[where a = f \cdot p - f \cdot q and
    b = f \cdot q - g \cdot q] have ... \leq \(| f \cdot p - f \cdot q| + \(| f \cdot q - g \cdot q| \)
    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
  \( y \leq \Delta \rightarrow |fiX f p r - fiX g q r| \leq x \)
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 hpC 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 \( \exists \)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 \( y \leq \Delta \))
  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
  case False
    with True show \( \exists \)thesis
      by simp
next
  case False
  with abs-dif-fiX-bound-C-aux1 and
have \(|f_X f p r - f_X g q r| < = x + y) by blast with False show \(\?thesis by simp
\)

\[
\begin{align*}
\text{1.3.2 Main theorem} \\
\text{theorem prec-enh:} \\
\text{assumes} \\
hC & C \subseteq PR and \\
hbx & \forall l \in C. |f l - g l| < = x and \\
hby1 & \forall l \in C. \forall m \in C. |f l - f m| < = y and \\
hby2 & \forall l \in C. \forall m \in C. |g l - g m| < = y and \\
hpC & p \in C and \\
hqC & q \in C \\
\text{shows} \ |cfni p f - cfni q g| < = \\
(\text{real (card } C) \ast (x + (if (y < = \Delta) then 0 else y)) + \\
\text{real (card } ((..<np) - C)) \ast (2 \ast \Delta + x + y) / \text{real np} \\
\text{(is | ?dif-div-np | < = } \begin{array}{l}
\text{proof -} \\
\text{have } |(\sum l \in (..<np). f_X f p l ) - \\
(\sum l \in (..<np). f_X g q l)| = \\
|(\sum l \in (..<np). f_X f p l - f_X g q l)| \\
\text{(is | ?dif | = | ?dif |) } \\
\text{by (simp add: setsum-subtractf)} \\
\text{also} \\
\text{from abs-sum-np-ineq hC} \\
\text{have } ... < = \\
(\sum l \in C. |f_X f p l - f_X g q l| ) + \\
(\sum l \in (..<np) - C. |f_X f p l - f_X g q l| ) \\
\text{(is | ?dif | < = } \begin{array}{l}
\text{?boundC'} + \text{?boundnpC'} \\
\text{by simp} \\
\text{also} \\
\text{have } ... < = \\
\text{real (card } C) \ast (x + (if (y < = \Delta) then 0 else y))) + \\
\text{real (card } ((..<np) - C)) \ast (2 \ast \Delta + x + y) \\
\text{(is } ... < = \text{?boundC} + \text{?boundnpC } ) \\
\text{proof -} \\
\text{from abs-dif-fiX-bound-C and} \\
hbx and hby1 and hby2 and hpC and hqC \\
\text{have } \forall r \in C. \\
|f_X f p r - f_X g q r| < = x + \\
(\text{if (y < = } \Delta) \text{ then } 0 \text{ else y) } \\
\text{by blast} \\
\text{thus } \?thesis using sum-le[where S=C] and finitC[OF hC] \\
\text{by force} \\
\text{qed}
\end{array}
\end{align*}
\]
moreover
have \(?\text{boundnpC}' \leq \text{boundnpC}\)
proof -
  from \(\text{abs-dif-fiX-bound}\) and 
  \(\text{hbx}\) and \(\text{hby}\) and \(\text{hpC}\) and \(\text{hqC}\)
  have \(\forall r \in (\{..<\text{np}\} - \text{C}). |\text{fiX} f p r - \text{fiX} g q r| \leq 2 * \Delta + x + y\)
  by \(\text{blast}\)
  with \(\text{finitnpC}\)
  show \(?\text{thesis}\)
    by (auto intro: \(\text{sum-le}\))
  qed
ultimately
show \(?\text{thesis}\) by \(\text{arith}\)
qed
finally
have \(\text{bound}: |\text{dif}| \leq \text{boundC} + \text{boundnpC}\).
thus \(?\text{thesis}\)
proof -
  have \(?\text{dif-div-np} = \text{dif} / \text{real np}\)
    by (simp add: \(\text{cfni-def divide-inverse algebra-simps}\))
  hence \(|\text{cfni} \, f - \text{cfni} \, q | = |\text{dif}| / \text{real np}\)
    by \(\text{force}\)
  with \(\text{bound}\) show \(?\text{thesis}\)
    by (auto simp add: \(\text{cfni-def divide-inverse constants-ax}\))
  qed
qed

1.4 Accuracy Preservation property

First, a simple lemma about an arithmetic property of the generalized sum-
lation over a set constructor.

lemma \text{sum-div-card}:
(\(\sum \{\ldots <\text{n} : \text{nat} \}. \, f \, l\} + q * \text{real n} =
(\sum \{\ldots <\text{n} \}. \, f \, l + q \})
(is \(?\text{Sl n} = \text{Sr n}\))
proof (induct \(n\))
case 0 thus \(?\text{case}\) by \(\text{simp}\)
next
case \(\text{Suc } n\)
thus \(?\text{case}\)
  by (auto simp: \(\text{real-of-nat-Suc distrib-left lessThan-Suc}\))
qed

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

lemma \text{bound-aux-C}:
assumes
  \(\text{hby}: \forall \, l \in \text{C}. \forall \, m \in \text{C}. \, |\text{f} \, l - \text{f} \, m| \leq x\) and
hpC: \( p \in C \) and
hqC: \( q \in C \) and
hrC: \( r \in C \)
shows
\[ |f_X \, f \, p \, r - f \, q| \leq x \]

proof (cases \( |f \, p - f \, r| \leq \Delta \))

case True
then have \( |f_X \, f \, p \, r - f \, q| = |f \, r - f \, q| \)
by (simp add: fiX-def)
also
from hby hqC hrC have \( \ldots \leq x \) by blast
finally
show \( \text{thesis} \).

next
case False
then have \( |f_X \, f \, p \, r - f \, q| = |f \, p - f \, q| \)
by (simp add: fiX-def)
also
from hby hpC hqC have \( \ldots \leq x \) by blast
finally
show \( \text{thesis} \).
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
\[ |f_X \, f \, p \, r - f \, q| \leq x + \Delta \]

proof (cases \( |f \, p - f \, r| \leq \Delta \))

case True
then have \( |f_X \, f \, p \, r - f \, q| = |f \, r - f \, q| \)
by (simp add: fiX-def)
also
have \( \ldots = \ |(f \, r - f \, p) + (f \, p - f \, q)| \)
by arith
also
have \( \ldots \leq |f \, p - f \, r| + |f \, p - f \, q| \)
by arith
also
from True have \( \ldots \leq \Delta + |f \, p - f \, q| \) by arith
also
from hby hpC hqC have \( \ldots \leq \Delta + x \) by simp
finally
show \( \text{thesis by simp} \)
next
case False
then have \( |f_X \, f \, p \, r - f \, q| = |f \, p - f \, q| \)
by (simp add: fiX-def)
also
from hby hpC hpqC have ... <= x by blast
finally
show thesis using constants-ax by arith
qed

1.4.1 Main theorem

lemma accru-pres:
assumes
hC: C ⊆ PR and
hby: ∀ l∈C. ∀ m∈C. |f l − f m| <= x and
hpC: p∈C and
hqC: q∈C
shows | cfni p f − f q | <=
(real (card C) * x + real (card (..<np − C)) + (x + Δ))/real np
(is ?abs1 <= (?bC + ?bnpC)/real np)
proof−
from abs-sum-np-ineq and hC have
|∑ l∈{..<np}. fiX f p l − f q | <=
(∑ l∈C. | fiX f p l − f q |) +
(∑ l∈(..<np−C). | fiX f p l − f q |)
by simp
also
have ...
... <= real (card C) * x +
real (card (..<np − C)) * (x + Δ)
proof−
have (∑ l∈C. | fiX f p l − f q |) <=
real (card C) * x
proof−
from bound-aux-C and
hby and hpC and hqC
have ∀ r∈C.
|fiX f p r − f q | <= x
by blast
thus thesis using sum-le[where S=C] and finitC[OF hC]
by force
qed
moreover
have (∑ l∈(..<np−C). | fiX f p l − f q |) <=
real (card (..<np − C)) * (x + Δ)
proof−
from bound-aux and
hby and hpC and hqC
have ∀ r∈(..<np−C).
|fiX f p r − f q | <= x + Δ
by blast

thus \( ?\text{thesis using } \text{sum-le[where } S=\{..<np\} - C]\)

and \( \text{finite} \ np \ C \)

by force

qed

ultimately show \( ?\text{thesis by } \text{arith} \)

qed

finally have bound: \[ |\sum_{l \in \{..<np\}}. fX f p l - f q| \]

\[ \leq \text{real } (\text{card } C) \ast x + \text{real } (\text{card } \{..<np\} - C) \ast (x + \Delta) \]

thus \( ?\text{thesis} \)

proof —

from constants-ax have

res: inverse \( (\text{real } np) \ast \text{real } np = 1 \)

by auto

have \( (cfni \ p \ f - f q) \ast \text{real } np = \)

\( (\sum_{l \in \{..<np\}}. fX f p l) \ast \text{real } np / \text{real } np - f q \ast \text{real } np \)

by (simp add: cfni-def algebra-simps)

also have ...

\( (\sum_{l \in \{..<np\}}. fX f p l) - f q \ast \text{real } np \)

by simp

also from \( \text{sum-div-card[where } f = fX f p \ \text{and } n=\text{np} \ \text{and } q = -f q] \)

have ...

\( (\sum_{l \in \{..<np\}}. fX f p l - f q) \)

by simp

finally have \( (cfni \ p \ f - f q) \ast \text{real } np = (\sum_{l \in \{..<np\}}. fX f p l - f q) \)

— cambia

hence \( (cfni \ p \ f - f q) \ast \text{real } np / \text{real } np = \)

\( (\sum_{l \in \{..<np\}}. fX f p l - f q) / \text{real } np \)

by auto

with constants-ax have

\( (cfni \ p \ f - f q) = \)

\( (\sum_{l \in \{..<np\}}. fX f p l - f q) / \text{real } np \)

by simp

hence \( |cfni \ p \ f - f q| = \)

\( |(\sum_{l \in \{..<np\}}. fX f p l - f q) / \text{real } np| \)

by simp

also have ...

\( = |(\sum_{l \in \{..<np\}}. fX f p l - f q)| / \text{real } np \)

by auto
finally have | cfni p f − f q | =
| (∑ l∈{..<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
\[ \text{kmax} :: (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{process set} \Rightarrow \text{process set} \text{ where} \]
\[ \text{kmax} f P = (\text{SOME} S. S \subseteq P \land \text{card} S = \text{khl} \land
(\forall i \in S. \forall j \in (P - S). f j \leq f i)) \]

definition
\[ \text{kmin} :: (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{process set} \Rightarrow \text{process set} \text{ where} \]
\[ \text{kmin} f P = (\text{SOME} S. S \subseteq P \land \text{card} S = \text{khl} \land
(\forall i \in S. \forall j \in (P - S). f i \leq f j)) \]

With the previus functions we define a new one \( \text{reduce} \). 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 (\( \text{op ‘} \)) of Isabelle.

definition
\[ \text{reduce} :: (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{process set} \Rightarrow \text{Clocktime set} \text{ where} \]
\[ \text{reduce} f P = f \ ‘(P - (\text{kmax} f P \cup \text{kmin} f P)) \]

And finally the convergence function. This is defined with the builtin \( \text{Max} \) and \( \text{Min} \) functions of Isabelle.

definition
\[ \text{cfnl} :: \text{process} \Rightarrow (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{Clocktime} \text{ where} \]
\[ \text{cfnl} p f = (\text{Max} (\text{reduce} f PR) + \text{Min} (\text{reduce} f PR)) \div 2 \]

2.2 Translation Invariance property.

2.2.1 Auxiliary lemmas

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

lemma \text{ex-Maxf}:
\footnote{The name of this function was taken from [5].}
fixes $S$ and $f :: 'a \Rightarrow ('b::linorder)
assumes fin: finite $S$
shows $S \neq \{\} \Longrightarrow \exists m \in S. \forall s \in S. f m \leq f s$
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 $f x \leq f m$ 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 $S \neq \{\} \Longrightarrow \exists m \in S. \forall s \in S. f m \leq f s$
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
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 the 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 it is the only thing that we know about these functions.

**Lemma kmax-prop:**

**Fixes** \( f : nat \Rightarrow \text{Clocktime} \)

**Shows**

\[
(kmax f PR) \subseteq PR \land \text{card}(kmax f PR) = khl \land \\
(\forall i \in (kmax f PR), \forall j \in PR - (kmax f PR). f j \leq f i)
\]

**Proof—**

**Have** \( khl \leq np \rightarrow \\
(\exists S. S \subseteq PR \land \text{card} S = khl \land (\forall i \in S, \forall j \in PR - S. f j \leq f i))
\)

**Proof (induct (khl))**

**Have** \( \neg P 0 \) **by** `force`

**Thus** \( 0 \leq np \rightarrow \neg P 0 \)

**Next**

**Fix** \( n \)

**Assume** \( asm : n \leq np \rightarrow \neg P n \)

**Show** \( Suc n \leq np \rightarrow \neg P (Suc n) \)

**Proof**

**Assume** \( asm2 : Suc n \leq np \)

**With** `asm` **have** \( \neg P n \) **by** `simp`

**Then obtain** \( S \) **where**

**SinPR :** \( S \subseteq PR \land \text{card} S = n \) **and**

**cardS :** \( \text{card} S = n \) **and**

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

**By** `blast`

**Let** \( ?e = \text{SOME} i. i \in PR - S \land \\
(\forall j \in PR - S. f j \leq f i) \)

**Let** \( ?S = \text{insert} ?e S \)

**Have** \( \exists i. i \in PR - S \land (\forall j \in PR - S. f j \leq f i) \)

**Proof—**

**From** `SinPR` **and** `finite-subset`

**Have** \( \text{finite} (PR - S) \)

**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. f j ≤ f e)

by (auto dest!: someI-ex)

from maxH and HI

have (∀ i∈?S. ∀ j∈PR − ?S. f j ≤ f i)

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. f i ≤ f j) ..

thus thesis by (unfold kmax-def)

(rule someI [where P=λS. S ⊆ PR ∧ card S = khl ∧ (∀ i∈S. ∀ j∈PR − S. f j ≤ f i)])

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). f i ≤ f j)

proof –

have khl <= np →

(∃ S. S ⊆ PR ∧ card S = khl ∧ (∀ i∈S. ∀ j∈PR − S. f i ≤ f j))

(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: card $S = n$ and
HI: $(\forall i \in S. \forall j \in PR - S. f_i \leq f_j)$
by blast
let $\ ?e = \text{SOME } i. i \in PR - S \land
(\forall j \in PR - S. f_i \leq f_j)$
let $\ ?S = \text{insert } \ ?e \ S$
have $\exists i. i \in PR - S \land (\forall j \in PR - S. f_i \leq f_j)$

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 card $\ ?S = Suc \ n$
by (auto dest: card-insert-disjoint)
moreover
have $\ ?S \subseteq PR$ using SinPR and ePRS by auto
ultimately
show $?P (Suc \ n)$ by blast
qed

hence $?P \ khl$ using khl-bound by auto
then obtain $S$ where
$S \subseteq PR \wedge \text{card } S = \text{khl} \land (\forall i \in S. \forall j \in PR - S. f_i \leq f_j)$ ..
thus $?thesis$ by (unfold \kmin-def)
(rule someI \[where \ P=\lambda S. S \subseteq PR \wedge \text{card } S = \text{khl} \land (\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 ≠ {}
proof -
from constants-ax have
  0 < (np − 2 * khl) by arith
also
{ from kmax-prop kmin-prop
  have card (kmax f PR) = khl ∧ card (kmin f PR) = khl
    by blast
  moreover
from finite-kmax and finite-kmin card-Un-Int[THEN sym]
  have card (kmax f PR ∪ kmin f PR) +
    card (kmax f PR ∩ kmin f PR) =
    card (kmax f PR) + card (kmin f PR)
    by auto
  ultimately
  have card (kmax f PR ∪ kmin f PR) ≤ 2 * khl
    by auto
}
hence
... ≤ card PR − card (kmax f PR ∪ kmin f PR)
by simp
also
{ from kmax-prop and kmin-prop have
  (kmax f PR ∪ kmin f PR) ⊆ PR by blast
}
hence
... = card (PR − (kmax f PR ∪ kmin f PR))
apply (intro card-Diff-subset[THEN sym])
apply (rule finite-subset)
by auto

finally
have 0 < card (PR − (kmax f PR ∪ 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 \(f :: \text{nat} \Rightarrow \text{Clocktime}\)
shows \(f' (PR-(kmax f PR \cup kmin f PR)) = f' (PR-(kmax (\lambda p. f p + c) PR \cup kmin (\lambda p. f p + c) PR))\)
apply (unfold kmin-def kmax-def)
by simp

lemma max-shift:
fixes \(f :: \text{nat} \Rightarrow \text{Clocktime}\) and \(S\)
assumes notEmpFin: \(S \neq \{\}\) finite \(S\)
shows \(\text{Max} (f'S) + x = \text{Max} ( (\lambda p. f p + x) ' S)\)
proof−
from notEmpFin have \(f'S \neq \{\}\) and \((\lambda p. f p + x) ' S \neq \{\}\)
by auto
with notEmpFin have \(\text{Max} (f'S) \in f'S \text{ Max} ((\lambda p. f p + x)'S) \in (\lambda p. f p + x) ' S\)
\((\forall fs \in (f'S). fs \leq \text{Max} (f'S))\)
\((\forall fs \in ((\lambda p. f p + x)'S). fs \leq \text{Max} ((\lambda p. f p + x)'S))\)
by auto
thus \(?thesis by force
qed

lemma min-shift:
fixes \(f :: \text{nat} \Rightarrow \text{Clocktime}\) and \(S\)
assumes notEmpFin: \(S \neq \{\}\) finite \(S\)
shows \(\text{Min} (f'S) + x = \text{Min} ( (\lambda p. f p + x) ' S)\)
proof−
from notEmpFin have \(f'S \neq \{\}\) and \((\lambda p. f p + x) ' S \neq \{\}\)
by auto
with notEmpFin have \(\text{Min} (f'S) \in f'S \text{ Min} ((\lambda p. f p + x)'S) \in (\lambda p. f p + x) ' S\)
\((\forall fs \in (f'S). \text{Min} (f'S) \leq fs)\)
\((\forall fs \in ((\lambda p. f p + x)'S). \text{Min} ((\lambda p. f p + x)'S) \leq fs)\)
by auto
thus \(?thesis by force
qed
2.2.2 Main theorem

**theorem** \textit{trans-inv}:  

**fixes** \( f :: \text{nat} \Rightarrow \text{Clocktime} \)  

**shows**  

\( \text{cfnl } p \ f + x = \text{cfnl } (\lambda \ p \ . \ f p + x) \)  

**proof**  

**have** \( \text{cfnl } p \ (\lambda \ p \ . \ f p + x) = \)  

\( (\text{Max } (\text{reduce } (\lambda \ p \ . \ f p + x) \ PR) + \text{Min } (\text{reduce } (\lambda \ p \ . \ f p + x) \ PR)) / 2 \)  

by \((\text{unfold cfnl-def, simp})\)  

also  

**have** \( \ldots = \)  

\( (\text{Max } ((\lambda \ p \ . \ f p + x) ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)))) + x + \)  

\( \text{Min } ((\lambda \ p \ . \ f p + x) ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)))) / 2 \)  

by \((\text{unfold reduce-def, simp})\)  

also  

**have** \( \ldots = \)  

\( (\text{Max } f ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) + x + \)  

\( \text{Min } f ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) ) / 2 \)  

**proof**  

**have** \( \text{finite } (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR) \)  

by \text{auto}  

moreover  

from \( \text{reduce-not-empty} \) have  

\( \text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR) \neq \{ \} \)  

by \( \text{auto simp add: reduce-def} \)  

ultimately  

**have**  

\( \text{Max } ((\lambda \ p \ . \ f p + x) ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) ) = \)  

\( \text{Max } f ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) + x \)  

and  

\( \text{Min } ((\lambda \ p \ . \ f p + x) ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) ) = \)  

\( \text{Min } f ^ \prime \ (\text{PR} - (kmax (\lambda \ p \ . \ f p + x) \ PR) \cup kmin (\lambda \ p \ . \ f p + x) \ PR)) + x \)  

using \text{max-shift and min-shift}  

by \text{auto}  

thus \( \text{thesis by auto} \)  

qed  

also  

from \( \text{reduce-shift} \)
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 conse-
cuence 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 ∪ 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 fj: j ∈ PR − (kmax f PR ∪ kmin f PR) ∧ f j = r
        by blast
hence \( j \in (PR - \text{kmax } f \ PR) \)
by blast
from this fjr asm
show ?thesis using kmax-prop
by auto
qed
qed
qed

lemma \text{kmin-le}:
\( \forall i \in (\text{kmin } f \ PR), \forall r \in (\text{reduce } f \ PR), f i \leq r \)
proof
fix \( i \) assume asm: \( i \in \text{kmin } f \ PR \)
show \( \forall r \in (\text{reduce } f \ PR), f i \leq r \)
proof
fix \( r \) assume asm2: \( r \in (\text{reduce } f \ PR) \)
show \( f i \leq r \)
proof
from asm2 and exist-reduce have
\( \exists j \in (PR - (\text{kmax } f \ PR \cup \text{kmin } f \ PR)), f j = r \) by blast
then obtain \( j \)
where fjr: \( j \in (PR - (\text{kmax } f \ PR \cup \text{kmin } f \ PR)) \wedge f j = r \)
by blast
hence \( j \in (PR - \text{kmin } f \ PR) \)
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 \( \theta \leq c \) !!

lemma \text{abs-distrib-div}:
\( \theta < (c :: \text{real}) \implies |a / c - b / c| = |a - b| / c \)
proof
assume ch: \( \theta < c \)
{ 
  fix \( d :: \text{real} \)
  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
}
\[ |a - b| \cdot d = |a - b| \cdot d \]

finally
have \[ |a \cdot d - b \cdot d| = |a - b| \cdot d \]

\}
with \textbf{ch} and \textbf{divide-inverse} show \( \text{thesis} \)
by (auto simp add: divide-inverse)
qed

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

\textbf{lemma uboundmax:}
assumes \( hC: C \subseteq PR \text{ and} \)
\( hCk: np \leq \text{card C} + khl \)
shows \( \exists i \in C. \text{Max} (\text{reduce f PR}) \leq f i \)
proof−
from \( \text{reduce-not-empty and finite-reduce} \)
have \( \text{maxrinv: Max (reduce f PR) } \in \text{reduce f PR} \)
by simp
with \( \text{exist-reduce} \)
have \( \exists i \in \text{PR} -(\text{kmax f PR } \cup \text{kmin f PR}). f i = \text{Max (reduce f PR)} \)
by simp
then obtain \( pmax \) where
\( \text{pmax-in-reduc: pmax } \in \text{PR}-(\text{kmax f PR } \cup \text{kmin f PR} ) \text{ and} \)
\( \text{fpmax-ismax: f pmax = Max (reduce f PR)} \) ..
hence \( C \cap \text{insert pmax (kmax f PR) } \neq \{\} \)
proof−
from \( \text{kmax-prop and pmax-in-reduc} \)
and \( \text{finite-kmax and hCk} \) have
\( \text{card PR } < \text{card C} + \text{card (insert pmax (kmax f PR))} \)
by simp
moreover
from \( \text{pmax-in-reduc and kmax-prop} \)
have \( \text{insert pmax (kmax f PR) } \subseteq \text{PR} \text{ by blast} \)
moreover
note \( hC \)
ultimately
show \( \text{thesis} \)
using subsets-int[of \( \text{PR C insert pmax (kmax f PR) }\)]
by simp
qed
hence \( \text{res: } \exists i \in C. i=pmax \lor i \in \text{kmax f PR } \text{ by blast} \)
then obtain \( i \) where
\[ i \in C \text{ and altern: } i = \text{pmax} \lor i \in \text{kmax f PR} \]

thus \( ?\text{thesis} \)

proof (cases \( i = \text{pmax} \))

\begin{itemize}
  \item case \( \text{True} \)
    with \( \text{inC f pmax-ismax show \( ?\text{thesis by force} \)) \)
  \item case \( \text{False} \)
    with \( \text{altern maxrinr f pmax-ismax kmax-ge} \)
    have \( f \text{ pmax} \leq f i \) by simp
    with \( \text{inC f pmax-ismax show \( ?\text{thesis by auto} \)) \)
\end{itemize}

qed

qed

lemma \( \text{lboundmin} \):

assumes
\( \text{hC: } C \subseteq \text{PR} \text{ and} \)
\( \text{hCk: } \text{np} \leq \text{card } C + \text{khl} \)

shows
\( \exists i \in C. f i \leq \text{Min (reduce f PR)} \)

proof –

\( \text{from reduce-not-empty and finite-reduce} \)
\( \text{have minrinr: Min (reduce f PR) } \in \text{reduce f PR} \)
\( \text{by simp} \)

\( \text{with exist-reduce} \)
\( \text{have } \exists i \in \text{PR} \setminus (\text{kmax f PR } \cup \text{kmin f PR}) \text{. } f i = \text{Min (reduce f PR)} \)
\( \text{by simp} \)

then obtain \( \text{pmin where} \)

\( \text{pmin-in-reduc: } \text{pmin } \in \text{PR} \setminus (\text{kmax f PR } \cup \text{kmin f PR}) \text{ and} \)
\( \text{fpmin-ismin: } f \text{ pmin } = \text{Min (reduce f PR)} \).

hence \( C \cap \text{insert pmin (kmin f PR) } \neq \{\} \)

proof –

\( \text{from kmin-prop and pmin-in-reduc} \)
\( \text{and finite-kmin and hCk } \text{have} \)
\( \text{card PR } < \text{card } C + \text{card (insert pmin (kmin f PR))} \)
\( \text{by simp} \)

moreover
\( \text{from pmin-in-reduc and kmin-prop} \)
\( \text{have insert pmin (kmin f PR) } \subseteq \text{PR by blast} \)

moreover
\( \text{note hC} \)

ultimately
\( \text{show } ?\text{thesis} \)

using subsets-int[of PR C insert pmin (kmin f PR)]

by simp

qed

hence \( \exists i \in C. i = \text{pmin } \lor i \in \text{kmin f PR} \) by blast

then obtain \( i \) where

\( \text{inC: } i \in C \text{ and altern: } i = \text{pmin } \lor i \in \text{kmin f PR} \).

thus \( ?\text{thesis} \)

29
proof\(\text{(cases } i=p\text{min)}\)
  case True
    with \(i\in C\) \(\text{fpmin-ismin}\) show \(\text{thesis by force}\)
next
  case False
    with \(\text{altern minrinr fpmin-ismin kmin-le}\)
    have \(f\ i\ <=\ \text{fmin}\ by\ \text{simp}\)
    with \(i\in C\) \(\text{fpmin-ismin}\) show \(\text{thesis by auto}\)
qed
qed

lemma \text{same-bound}:\nassumes
  \(hC: C \subseteq PR\) and
  \(hCk: np <= \text{card } C + khl\) and
  \(hnk: 3 \times khl < np\)
shows
  \(\exists\ i\in C.\ \text{Min (reduce } f\ PR) <= f\ i \land g\ i <= \text{Max (reduce } g\ PR)\)
proof\(\)
  have \(b1: khl + 1 <= \text{card } (C \cap (PR - \text{kmin } f \ PR))\)
proof\(\text{(rule pigeonhole)}\)
    show \(\text{finite } PR\) by simp
next
    show \(C \subseteq PR\) by fact
next
    show \(PR - \text{kmin } f \ PR \subseteq PR\) by blast
next
    show \(\text{card } PR + (khl + 1) \leq \text{card } C + \text{card } (PR - \text{kmin } f PR)\)
proof\(\)
    from \(hnk\) and \(hCk\) have
      \(np + khl < np + \text{card } C - khl\) by arith
    also
    from \(\text{kmin-prop}\)
    have \(.. = np + \text{card } C - \text{card } (\text{kmin } f \ PR)\)
      by auto
    also
    have \(.. = \text{card } C + (\text{card } PR - \text{card } (\text{kmin } f \ PR))\)
proof\(\)
    from \(\text{kmin-prop}\) have
      \(\text{card } (\text{kmin } f \ PR) <= \text{card } PR\)
    by \(\text{intro card-mono, auto}\)
    thus \(\text{thesis by (simp)}\)
qed
also
from \(\text{kmin-prop}\)
have \(.. = \text{card } C + (\text{card } PR - \text{card } (\text{kmin } f \ PR))\)
proof\(\)
from \(\text{kmin-prop}\) and \(\text{finite-kmin}\) have
  \(\text{card } PR - \text{card } (\text{kmin } f \ PR) = \text{card } (PR - \text{kmin } f \ PR)\)

by (intro card-Diff-subset[THEN sym])(auto)
thus thesis by auto
qed
finally
show thesis
by (simp)
qed
qed

have C ∩ (PR − kmin f PR) ∩ (PR − kmax g PR) ≠ {}
proof (intro subsets-int)
show finite PR by simp
next
show C ∩ (PR − kmin f PR) ⊆ PR
by blast
next
show PR − kmax g PR ⊆ PR
by blast
next
show card PR <
  card (C ∩ (PR − kmin f PR)) + card (PR − kmax g PR)
proof−
from kmax-prop and finite-kmax
have (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 − khd by simp
with b1
show thesis by arith
qed
qed

hence
  ∃ i. i ∈ C ∧ i ∈ (PR − kmin f PR) ∧ i ∈ (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) ≤ f i
proof (cases i ∈ kmax f PR)
  case True
  from reduce-not-empty and finite-reduce have
    Min (reduce f PR) ∈ reduce f PR by auto
  with True show thesis
    using kmax-ge by blast
  next
  case False
  with not-in-kmin
have \( i \in PR - (\text{Kmax } f PR \cup \text{Kmin } f PR) \)

by blast

with reduce-def have \( f i \in \text{reduce } f PR \)

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 \( i \in \text{Kmin } g PR \))

\begin{itemize}
  \item 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
  \item 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
  \end{itemize}

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).

\textbf{theorem prec-enh:}

\textbf{assumes}

\( hC: C \subseteq PR \) and
\( hCF: \text{np } - \text{nF } \leq \text{card } C \) and
\( hFn: 3 * \text{nF } < \text{np } \) and
\( hFk: \text{nF } = \text{khl } \) and
\( hhx: \forall \ l \in C. |f l - g l| \leq x \) 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
\( hqC: q \in C \)

\textbf{shows} \( |cfern p f - cfern q g| \leq y / 2 + x \)

\textbf{proof}

from hCF and hFk
have \( hCk: np \leq \text{card } C + khl \) by arith

from \( hFn \) and \( hFk \)

have \( hnk: 3 \times khl < np \) by arith

let

\begin{align*}
?\maxf &= \text{Max} (\text{reduce } f \ PR) \\
?\minf &= \text{Min} (\text{reduce } f \ PR) \\
?\maxg &= \text{Max} (\text{reduce } g \ PR) \\
?\ming &= \text{Min} (\text{reduce } g \ PR)
\end{align*}

from \( \text{abs-distrib-div} \)

have

\[ |cfnlp f - cfnl q g| = \]
\[ |?\maxf + ?\minf + - ?\maxg + - ?\ming| / 2 \]

by (unfold \( cfnl\)-def) simp

moreover

have \[ |?\maxf + ?\minf + - ?\maxg + - ?\ming| <= y + 2 * x \]

— 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 \( \text{uboundmax } hC \ hCk \)

obtain \( mxf \)

where \( \text{mxfinC: mxf} \in C \) and

\( \text{maxf: } ?\maxf \leq f \text{ mxf} \) by blast

moreover

from \( \text{lboundmin } hC \ hCk \)

obtain \( mng \)

where \( \text{mnginC: mng} \in C \) and

\( \text{ming: } g \text{ mng} \leq ?\ming \) by blast

moreover

from \( \text{same-bound } hC \ hCk \ hnk \)

obtain \( mxn \)

where \( \text{mxninC: mxn} \in C \) and

\( \text{mxnf: } ?\minf \leq f \text{ mxn} \) and

\( \text{mxng: } g \text{ mxn} \leq ?\maxg \) by blast

ultimately

have

\[ | ?\maxf + ?\minf + - ?\maxg + - ?\ming| <= \]
\[ (f \text{ mxf} + - g \text{ mng}) + (f \text{ mxn} + - g \text{ mxn}) \] by arith

also

from \( \text{mxninC hbx abs-le-D1} \)

have

\[ ... <= (f \text{ mxf} + - g \text{ mng}) + x \]

by auto

also

have

\[ ... = (f \text{ mxf} + - f \text{ mng}) + (f \text{ mng} + - g \text{ mng}) + x \]

by arith

also
have ... \leq y + (f \text{min} + g \text{min}) + x

proof-
from \text{mxfinC} \text{minfinC} \text{hby1 abs-le-D1}
have \ f \text{max} + -f \text{min} \leq y
  by auto
thus \ ?\text{thesis}
  by auto
qed
also
from \text{minfinC} \text{hbx abs-le-D1}
have ... \leq y + 2 * x
  by auto
finally
show \?\text{thesis}.
next
case False
hence
| ?\text{max} + ?\text{min} + - ?\text{maxg} + - ?\text{ming} | =
  ?\text{maxg} + ?\text{ming} + - ?\text{maxf} + - ?\text{minf} by arith
moreover
from \text{uboundmax hC hCk}
obtain mxg
  where \text{mxginC}: mxg \in C and
    maxg: ?\text{maxg} \leq g \text{mxg} by blast
moreover
from \text{lboundmin hC hCk}
obtain mnf
  where \text{mnfinC}: mnf \in C and
    minf: f \text{mnf} \leq ?\text{minf} by blast
moreover
from \text{same-bound hC hCk hnk}
obtain mxn
  where \text{mxninC}: mxn \in C and
    mxnf: ?\text{ming} \leq g \text{mxn} and
    mxng: f \text{mxn} \leq ?\text{maxf} by blast
ultimately
have
| ?\text{maxf} + ?\text{minf} + - ?\text{maxg} + - ?\text{ming} | <=
  (g \text{mxg} + - f \text{mnf}) + (g \text{mxn} + - f \text{mxn}) by arith
also
from \text{mxninC} \text{hbx}
have ... \leq (g \text{mxg} + - f \text{mnf}) + x
  by \text{auto dest!: abs-le-D2}
also
have
... = (g \text{mxg} + - g \text{mnf}) + (g \text{mnf} + - f \text{mnf}) + x
  by arith
also
have ... \leq y + (g \text{mnf} + - f \text{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  | cfml p f - f q | <= y

proof–
from  hCF  and  hFk
have  npleCk:  np <= card C + khl  by  arith
show  ?thesis
proof(cases f q <= cfml p f)
case  True
from  npleCk  hC  and  uboundmax
have  ∃ i∈C. Max (reduce f PR) <= f i
  by  auto
then obtain  pi  where
  hpiC:  pi ∈ C and
  fnGeMax:  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 \texttt{fpIGeMax} have
\begin{itemize}
\item \texttt{cfnlLePi}: \texttt{cfnl p f \leq f pi}
\item by (\texttt{auto simp add: cfnl-def})
\end{itemize}
with \texttt{True} have
\begin{itemize}
\item \texttt{| cfnl p f - f q | \leq | f pi - f q |}
\item by \texttt{arith}
\end{itemize}
with \texttt{hqC} and \texttt{hqC} and \texttt{hqby} show \texttt{?thesis}
by \texttt{force}
next
case \texttt{False}
from \texttt{npleCk hC and lboundmin}
have \exists \ i \in C. f i \leq Min (reduce f PR)
by \texttt{auto}
then obtain qi where
\begin{itemize}
\item \texttt{hqC}: qi \in C and
\item \texttt{fqiLeMax}: f qi \leq Min (reduce f PR) by \texttt{blast}
\end{itemize}
from \texttt{reduce-not-empty}
have \texttt{Min (reduce f PR) \leq Max (reduce f PR)}
by (\texttt{auto simp add: reduce-def})
with \texttt{fqiLeMax}
have \texttt{f qi \leq cfnl p f}
by (\texttt{auto simp add: cfnl-def})
with \texttt{False} have
\begin{itemize}
\item \texttt{| cfnl p f - f q | \leq | f qi - f q |}
\item by \texttt{arith}
\end{itemize}
with \texttt{hqC} and \texttt{hqC} and \texttt{hqby} show \texttt{?thesis}
by \texttt{force}
qed
qed
end

\section{CVC-lite and ICS proofs}

\subsection{Lemma abs\_distrib\_div}

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

\textbf{lemma abs\_distrib\_div:}
\[ \theta < (c::real) \implies |a / c - b / c| = |a - b| / c \]

It is not possible to prove this lemma in Isabelle using \texttt{arith} nor \texttt{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 \(c\) 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 were 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:

\[
\text{have } |\text{Max (reduce } f \text{ PR) + Min (reduce } f \text{ PR) +} \\
- \text{Max (reduce } g \text{ PR) + - Min (reduce } g \text{ PR)| } \leq y + 2 \times x
\]

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 where 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 find without the need of removing the quantifiers.

File accur_pres.cvc:
File accur_pres.ics:
References


