Instances of Schneider’s generalized protocol of clock synchronization.

Damian Barsotti

February 23, 2021

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 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

\[ np :: \text{nat} \quad \text{Number of processes and} \]
\[ \Delta :: \text{Clocktime} \quad \text{Fix value to discard processes where} \]
\[ \text{constants-ax: } 0 <\Delta \land np > 0 \]

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

definition

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

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

definition

\[ fiX :: [(\text{process} \Rightarrow \text{Clocktime}), \text{process}, \text{process}] \Rightarrow \text{Clocktime} \quad \text{where} \]
\[ fiX f p l = (\text{if } |f p - f l| <\Delta \text{ then } (f l) \text{ else } (f p)) \]

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

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

1.2 Translation Invariance property.

We first need to prove this auxiliary lemma.

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

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:**
\[ C \subseteq PR \implies \text{finite } C \]
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:**
finite \( S \)
\[ |\sum_{l \in S.} (f::'a \Rightarrow 'b::linordered-idom) \ l| \leq (\sum_{l \in S.} |f \ l|) \]
(is \[ \text{... \implies } \exists P S \])
by (rule sum-abs)

**Lemma sum-le:**
\[ [\text{finite } S \ ; \ \forall \ r \in S. \ f \ r \leq b ] \]
\[ \Rightarrow \]
\[ (\sum_{l \in S.} f \ l) \leq \text{real (card } S) \ast b \]
(is \[ [\text{finite } S \ ; \ \forall \ r \in S. \ f \ r \leq b ] \Rightarrow \exists P S \])
proof(induct \( S \) rule: finite-induct)
show \[ \exists P \} \] by simp
next
fix \( F \ x \)

4
assume \( f \text{init}; \) finite \( F \) and \( x \notin F \) and
\[ H11: \forall r \in F. \ f \ r \leq b \implies \sum f \ P \leq \text{real} (\text{card} \ F) * b \]
and \( \text{HI2}: \forall r \in \text{insert} \ x \ F. \ f \ r \leq b \)

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

qed

lemma \( \text{sum-np-eq} \):
assumes \( hC: \ C \subseteq \text{PR} \)
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 \( \text{sum.union-disjoint[where A=C and B=\{..<np\} - C]} \)

by auto

qed

lemma \( \text{abs-sum-np-ineq} \):
assumes \( hC: \ C \subseteq \text{PR} \)
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 ?abs-sum \leq ?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 ?abs-sum \leq ?sumC' + ?sumnpC')

by simp
also
from \( \text{abs-triangle-ineq} \)
have ...<= |\(\sum C\)'| + |\(\sum np C\)'|.
also
have ... <= \(\sum C\) + \(\sum np C\)
proof=
  from \(hC\) finitC sum-abs-triangle-ineq
have |\(\sum C\)'| <= \(\sum C\) by blast
moreover
from finitnpC and
  sum-abs-triangle-ineq[where \(f=f\) and \(S=PR-C\)]
have |\(\sum np C\)'| <= \(\sum np C\)
  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 \(\text{fiX-ubound}\):
\(\text{fiX}\ f\ p\ l\ <\ f\ p\ +\ \Delta\)
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 \(f\ l\ <\ f\ p\ +\ \Delta\) by arith
finally
  show \(?thesis\) 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 constants-ax have \(f\ p\ <\ f\ p\ +\ \Delta\) by arith
finally
  show \(?thesis\) by arith
qed

lemma \(\text{fiX-lbound}\):
\(f\ p\ -\ \Delta\ <\ \text{fiX}\ f\ p\ l\)
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 \(f\ p\ -\ \Delta\ <\ f\ l\) by arith
finally
  show \(?thesis\) 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 \( f_p = \bar{f}X f_p l \) by (simp add: \bar{f}X-def)
finally
show \(?thesis\) by arith
qed

lemma abs-fiX-bound: \(|\bar{f}X f_p l - f_p| \leq \Delta\)
proof–
have \( f_p - \Delta \leq f_p l \wedge \bar{f}X f_p l \leq f_p + \Delta \longrightarrow \?thesis\)
by arith
with \( \bar{f}X\)-bound \( \bar{f}X\)-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
\(|\bar{f}X f_p r - \bar{f}X g q r| \leq 2 \ast \Delta + x + y\)
proof–
have \(|\bar{f}X f_p r - \bar{f}X g q r| =
\ |\bar{f}X f_p r - f p + f p - \bar{f}X g q r|\)
by auto
also
have ... \( \leq |f p - \bar{f}X g q r| + |f p - f p| + |f p - \bar{f}X g q r|\)
by arith
also
from abs-fiX-bound
have ... \( \leq \Delta + |f p - \bar{f}X g q r|\)
by simp
also
have ... = \( \Delta + |f p - g q + (g q - \bar{f}X g q r)|\)
by simp
also
from abs-triangle-ineq[where \( a = f p - g q \) and \( b = g q - \bar{f}X g q r \)]
have ... \( \leq \Delta + |f p - g q| + |g q - \bar{f}X g q r|\)
by simp
also
have ... = \( \Delta + |f p - g q| + |\bar{f}X g q r - g q|\)
by arith
also
from abs-fiX-bound
have ... \( \leq 2 \ast \Delta + |f p - g q|\)
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 \( \theta \)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 \( \theta \)thesis
  proof(cases \(|g q - g r| \leq \Delta \))
    case True
    show \( \theta \)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 \( \theta \)thesis
    by (auto simp add: fiX-def)
  qed
next
  case False
  show \( \theta \)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 ... <= x + y by force
finally
show ?thesis .
qed
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 \(|fX f p r - fX g q r| = |f p - g r|
  by (auto simp add: fX-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 ... <= \(|f p - f r| + |f r - g r|
  by auto
also
from hbx hby1 hpC hqC have ... <= x + y by force
finally
show ?thesis .
qed
next
case False
show ?thesis
proof
- from outer-IH and False
have \(|fX f p r - fX g q r| = |f p - g q|
  by (auto simp add: fX-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 ... <= \(|f p - f q| + |f q - g q|
  by auto
also
from hbx hby1 hpC hqC have ... <= x + y by force
finally
show ?thesis .
qed
qed
\textbf{lemma} \textit{abs-dif-fiX-bound-C-aux2}:\newline
\textbf{assumes} \newline
\hspace{1em} \text{hbx:} \ \forall \ l \in C. \ |f \ l - g \ l| \leq x \ \text{and} \newline
\hspace{1em} \text{hby1:} \ \forall \ l \in C. \ \forall \ m \in C. \ |f \ l - f \ m| \leq y \ \text{and} \newline
\hspace{1em} \text{hby2:} \ \forall \ l \in C. \ \forall \ m \in C. \ |g \ l - g \ m| \leq y \ \text{and} \newline
\hspace{1em} \text{hpC:} \ p \in C \ \text{and} \newline
\hspace{1em} \text{hqC:} \ q \in C \ \text{and} \newline
\hspace{1em} \text{hrC:} \ r \in C \newline
\textbf{shows} \newline
\hspace{1em} y \leq \Delta \rightarrow |fiX f p r - fiX g q r| \leq x \newline
\textbf{proof}\newline
\hspace{1em} \text{assume} \ hyd: \ y \leq \Delta \newline
\hspace{1em} \text{show} \ |fiX f p r - fiX g q r| \leq x \newline
\hspace{1em} \textbf{proof}- \newline
\hspace{2em} \text{from} \ hpC \ \text{and} \ hrC \ \text{and} \ hby1 \ \text{and} \ hyd \ \text{have} \ |f \ p - f \ r| \leq \Delta \newline
\hspace{2em} \text{by} \ \text{force} \newline
\hspace{2em} \text{moreover} \newline
\hspace{3em} \text{from} \ hpC \ \text{and} \ hrC \ \text{and} \ hby2 \ \text{and} \ hyd \ \text{have} \ |g \ q - g \ r| \leq \Delta \newline
\hspace{3em} \text{by} \ \text{force} \newline
\hspace{3em} \text{moreover} \newline
\hspace{4em} \text{from} \ hrC \ \text{and} \ hbx \ \text{have} \ |f \ r - g \ r| \leq x \ \text{by} \ \text{auto} \newline
\hspace{4em} \text{ultimately} \newline
\hspace{5em} \text{show} \ ?\text{thesis} \newline
\hspace{5em} \text{by} \ (\text{auto simp add: fiX-def}) \newline
\textbf{qed}\newline
\textbf{lemma} \textit{abs-dif-fiX-bound-C}:\newline
\textbf{assumes} \newline
\hspace{1em} \text{hbx:} \ \forall \ l \in C. \ |f \ l - g \ l| \leq x \ \text{and} \newline
\hspace{1em} \text{hby1:} \ \forall \ l \in C. \ \forall \ m \in C. \ |f \ l - f \ m| \leq y \ \text{and} \newline
\hspace{1em} \text{hby2:} \ \forall \ l \in C. \ \forall \ m \in C. \ |g \ l - g \ m| \leq y \ \text{and} \newline
\hspace{1em} \text{hpC:} \ p \in C \ \text{and} \newline
\hspace{1em} \text{hqC:} \ q \in C \ \text{and} \newline
\hspace{1em} \text{hrC:} \ r \in C \newline
\textbf{shows} \newline
\hspace{1em} |fiX f p r - fiX g q r| \leq \newline
\hspace{2em} x + (\text{if} \ (y \leq \Delta) \ \text{then} \ 0 \ \text{else} \ y) \newline
\textbf{proof} (\text{cases} \ y \leq \Delta) \newline
\hspace{2em} \textbf{case} \ True \newline
\hspace{3em} \text{with} \ \textit{abs-dif-fiX-bound-C-aux2} \ \text{and} \newline
\hspace{3em} \text{hbx and hby1 and hby2 and hpC and hqC and hrC} \newline
\hspace{3em} \text{have} \ |fiX f p r - fiX g q r| \leq x \ \text{by} \ \text{blast} \newline
\hspace{3em} \text{with} \ True \ \text{show} \ ?\text{thesis} \ \text{by} \ \text{simp} \newline
\hspace{2em} \textbf{next} \newline
\hspace{3em} \textbf{case} \ False \newline
\hspace{4em} \text{with} \ \textit{abs-dif-fiX-bound-C-aux1} \ \text{and}
\[ \text{have } |f_X f p r - f_X g q r| \leq x + y \text{ by blast} \]

with False show \(\texttt{thesis by simp}\)

\textbf{1.3.2 Main theorem}

\textbf{theorem \texttt{prec-enh}}:

assumes
\[
\begin{align*}
  &hC: C \subseteq PR \text{ and } \\
  &hbx: \forall l \in C. \ |f l - g l| \leq x \text{ and } \\
  &hby1: \forall l \in C. \forall m \in C. \ |f l - f m| \leq y \text{ and } \\
  &hby2: \forall l \in C. \forall m \in C. \ |g l - g m| \leq y \text{ and } \\
  &hpC: p \in C \text{ and } \\
  &hqC: q \in C
\end{align*}
\]

shows \(|\texttt{cfni p f} - \texttt{cfni q g}| \leq \)

(\(\text{real } (\text{card } C) \ast (x + (\text{if } y \leq \Delta \text{ then } 0 \text{ else } y)) + \text{ real } (\text{card } (\{..<np\} - C)) \ast (2 \ast \Delta + x + y)) / \text{ real np}

(is | \texttt{?dif-div-np} | \leq \texttt{?B})

\textbf{proof–}

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)|

(is | \texttt{?dif}| \leq \texttt{?boundC'} + \texttt{?boundnpC'})

by simp

also

have \(\ldots \leq\)

\(|(\sum_{l \in C}. f_X f p l - f_X g q l|) + \text{ real } (\text{card } C) \ast (x + (\text{if } y \leq \Delta \text{ then } 0 \text{ else } y)) + \text{ real } (\text{card } (\{..<np\} - C)) \ast (2 \ast \Delta + x + y)

(is \(\ldots \leq \texttt{?boundC} + \texttt{?boundnpC}\))

\textbf{proof–}

have \(\texttt{?boundC'} \leq \texttt{?boundC}\)

\textbf{proof –}

from \texttt{abs-dif-fiX-bound-C} and

\textbf{proof –}

from \texttt{abs-dif-fiX-bound-C} and

\(\texttt{hbx and hby1 and hby2 and hpC and hqC}\)

have \(\forall r \in C.\ |

\(\texttt{f}_X f p r - \texttt{f}_X g q r| \leq x + \)

(if \(y \leq \Delta\) then 0 else y)

by blast

thus \(\texttt{thesis using sum-le}[\texttt{where S=C}] \text{ and } \texttt{finitC[OF hC]}\)

by force

\textbf{qed}
moreover
have ?boundnpC' <= ?boundnpC
proof -
  from abs-dif-fiX-bound and
  hbx and hby1 and hpC and hpC
  have \( \forall r \in (\ldots < np) - C \). \[|fiX f p r - fiX g q r| <= 2 \cdot \Delta + x + y\]
  by blast
  with finitnpC
  show \( \neg \neg \neg \) by (auto intro; sum-le)
  qed
ultimately
  show \( \neg \neg \neg \) by arith
  qed
finally
  have bound: \(|\text{dif}| <= ?\Delta \) + ?boundnpC .
  thus \( \neg \neg \neg \) 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| = |\text{dif}| / real np
  by force
  with bound show \( \neg \neg \neg \) 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 sum-
mation over a set constructor.

lemma sum-div-card:
\[
(\sum_{l \in \{\ldots < n \atop n: nat\}}. f l) + q * \text{real } n =
\sum_{l \in \{\ldots < n\}}. f l + q)
\]
(is \( ?\sum n = ?\sum n\))

proof (induct n)
case 0 thus \( \neg \neg \neg \) case by simp
next
case (Suc n)
thus \( \neg \neg \neg \) 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
\( |f_X \ f \ p \ r - f \ q| \leq x \)
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: \( f_X \)-def)
also
from hby hqC hrC have ... \( \leq x \) by blast
finally
show ?thesis .
next
case False
then have \( |f_X \ f \ p \ r - f \ q| = |f \ p - f \ q| \)
  by (simp add: \( f_X \)-def)
also
from hby hpC hqC have ... \( \leq x \) 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
\( |f_X \ f \ p \ r - f \ q| \leq x + \Delta \)
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: \( f_X \)-def)
also
have ... \( = |f \ r - f \ p| + (f \ p - f \ q| \)
  by arith
also
have ... \( \leq |f \ p - f \ r| + |f \ p - f \ q| \)
  by arith
also
from True have ... \( \leq \Delta + |f \ p - f \ q| \) by arith
also
from hby hpC hqC have ... \( \leq \Delta + x \) by simp
finally
show ?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 ... \leq x by blast
finally
show thesis using constants-ax by arith
qed

1.4.1 Main theorem

lemma accru-pres:
assumes
  hC: C \subseteq PR and
  hby: \forall l \in C. \forall m \in C. \vert f l - f m \vert \leq x and
  hpC: p \in C and
  hpqC: q \in C
shows \( \vert cf\{p f q \} \vert \leq \frac{(\text{real } (\text{card } C) \times x + \text{real } (\text{card } (\{..<np\} - C))) \times (x + \Delta))}{\text{real np}} \)
(is \( \text{abs} I \leq \frac{\text{abs}(C \pm \text{npC})}{\text{real np}} \))
proof-
from abs-sum-np-ineq and hC have
  \( \sum_{l \in \{..<np\}}. \text{fi}X f\ p\ l - f\ q \) \leq
  \( (\sum_{l \in C}. \vert \text{fi}X f\ p\ l - f\ q \vert) + (\sum_{l \in (\{..<np\} - C)}. \vert \text{fi}X f\ p\ l - f\ q \vert) \)
by simp
also
have
  ... \leq \text{real } (\text{card } C) \times x + \text{real } (\text{card } (\{..<np\} - C))) \times (x + \Delta))
proof-
from bound-aux-C and
hby and hpC and hpqC
have \( \forall r \in C. \vert \text{fi}X f\ p\ r - f\ q \vert \leq x \)
by blast
thus thesis using sum-le[where S=C] and finitC[OF hC]
by force
qed
moreover
have \( \sum_{l \in (\{..<np\} - C)}. \vert \text{fi}X f\ p\ l - f\ q \vert \leq \text{real } (\text{card } (\{..<np\} - C))) \times (x + \Delta)) \)
proof-
from bound-aux and
hby and hpC and hpqC
have \( \forall r \in (\{..<np\} - C). \vert \text{fi}X f\ p\ r - f\ q \vert \leq x + \Delta \)
by blast
thus \texttt{thesis} using \texttt{sum-le[where S=\{..<\texttt{np}\} - C]}
and \texttt{finitnpC}
by force
qed
ultimately
show \texttt{thesis} by arith
qed
finally
have bound: \(|\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q| \\
\leq \texttt{real (card C)} * x + \texttt{real (card \{}\{..<np\} - C\)} \} * (x + \Delta) \\
.
thus \texttt{thesis}
proof--
from constants-ax have
res: inverse (real np) * real np = 1
by auto
have \( (\texttt{cfni p f - f q}) * \texttt{real np} = \\
(\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l) * \texttt{real np} / \texttt{real np} - f\ q * \texttt{real np} \\
by (simp add: cfni-def algebra-simps)
also
have ... = \\
(\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l) - f\ q * \texttt{real np} \\
by simp
also
from sum-div-card[where f=\texttt{fiX}\ f\ p and n=\texttt{np} and q=- f\ q]
have ... = (\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q) \\
by simp
finally
have \( (\texttt{cfni p f - f q}) * \texttt{real np} = (\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q) \\
.
--- cambia
hence \( (\texttt{cfni p f - f q}) * \texttt{real np} / \texttt{real np} = \\
(\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q) / \texttt{real np} \\
by auto
with constants-ax have
\( (\texttt{cfni p f - f q}) = \\
(\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q) / \texttt{real np} \\
by simp
hence \(| \texttt{cfni p f - f q} | = \\
| (\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q) / \texttt{real np} | \\
by simp
also have
... = \(| (\sum_{l\in\{..<np\}}. \texttt{fiX}\ f\ p\ l - f\ q)| / \texttt{real np} \\
by auto
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 \( \varepsilon \)-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**

\[
\begin{align*}
\text{kmax} & : (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{process set} \Rightarrow \text{process set} \\
\text{kmax} f P &= (\text{SOME} \ S. \ S \subseteq P \land \text{card} \ S = khl \land \\
& \quad (\forall \ i \in S. \ \forall \ j \in (P-S). \ f j < = f \ i)
\end{align*}
\]

**definition**

\[
\begin{align*}
\text{kmin} & : (\text{process} \Rightarrow \text{Clocktime}) \Rightarrow \text{process set} \Rightarrow \text{process set} \\
\text{kmin} f P &= (\text{SOME} \ S. \ S \subseteq P \land \text{card} \ S = khl \land \\
& \quad (\forall \ i \in S. \ \forall \ j \in (P-S). \ f i < = f \ j)
\end{align*}
\]

With the previous 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 (\( (\cdot) \)) of Isabelle.

**definition**

\[
\begin{align*}
\text{reduce} f P &= f \ (P \ - \ (\text{kmax} f P \cup \text{kmin} f P))
\end{align*}
\]

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

**definition**

\[
\begin{align*}
\text{cfnl} p f &= (\text{Max} \ (\text{reduce} f \text{PR}) + \text{Min} \ (\text{reduce} f \text{PR})) / 2
\end{align*}
\]

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** \( \text{ex-Maxf} \):

\(^{1}\text{The name of this function was taken from [5].}\)
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 $m \in S \forall s \in S. f m \leq f s$
    using insert by blast
  show ?thesis
  proof (cases)
  assume $f x \leq f m$ thus ?thesis using $m$ by blast
  next
  assume $\neg f x \leq f m$ 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 $m \in S \forall s \in S. f m \leq f s$
    using insert by blast
  show ?thesis
  proof (cases)
  assume $f m \leq f x$ thus ?thesis using $m$ by blast
  next
  assume $\neg f m \leq f x$ thus ?thesis using $m$
    by (simp add: linorder-not-le order-less-le)
    (blast intro: order-trans)
This trivial lemma is needed by the next two.

**Lemma** \textit{khl-bound}: \[ khl < np \]

**Using** \textit{constants-ax} **by** \textit{arith}

The next two lemmas prove that \textsc{de} functions \textit{kmin} and \textit{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** \textit{kmax-prop}:

**Fixes** \[ f :: \textit{nat} \Rightarrow \textit{Clocktime} \]

**Shows**

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

**Proof**

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

  **Proof** (**induct** \( khl \))

  - **Have** \( \textit{PR} \ 0 \ \textit{by force} \)

  - **Thus** \( 0 <= np \rightarrow ?P \ 0 \ .. \)

**Next**

**Fix** \( n \)

**Assume** \( asm: n <= np \rightarrow ?P \ n \)

**Show** \( \textit{Suc} n <= np \rightarrow ?P \ (\textit{Suc} n) \)

**Proof**

- **Assume** \( asm2: \textit{Suc} n <= np \)

  **With** \( asm \) **have** \( ?P \ n \ \textit{by simp} \)

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

  - \( \textit{SinPR} : S \subseteq \textit{PR} \) **and**

  - \( \text{card} S \) **and**

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

  **By** \textit{blast}

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

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

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

  **Proof**

  - **From** \( \textit{SinPR} \) **and** \textit{finite-subset}

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

    **By** \textit{auto}

  **Moreover**

  - **From** \( \text{card} S \) **and** \( asm2 \ \textit{SinPR} \)

  - **Have** \( S \subseteq \textit{PR} \) **by** \textit{auto}

  **Hence** \( \textit{PR} - S \neq \emptyset \) **by** \textit{auto}
ultimately
show \textit{thesis using} \textit{ex-Maxf by blast}
qed

hence
\textit{ePRS: \(e \in PR-S\) and \textit{maxH: (\(\forall j \in PR-S. f j \leq f e\))}}
by (\textit{auto dest!: someI-ex})

from \textit{maxH and HI}
have \((\forall i \in ?S. \forall j \in PR-S. f j \leq f i)\)
by blast

moreover
from \textit{SinPR and finite-subset}
card\(S\) and \textit{ePRS}
have \((\forall i \in \textit{?S}. \forall j \in PR-\textit{?S}. f j \leq f i)\)
by blast

moreover
have \(\textit{?S} \subseteq PR\) using \textit{SinPR and ePRS} by auto
ultimately
show \(\textit{?P} (\textit{Suc n})\) by blast
qed

hence \(\textit{?P khl}\) using \textit{khl-bound by auto}

then obtain \(S\) where
\(S \subseteq PR \land \text{card } S = \text{khl} \land (\forall i \in S. \forall j \in PR-S. f j \leq f i)\) ..

thus \(\textit{thesis by (unfold kmax-def)}\)
(rule someI [where \(P=\lambda S. S \subseteq PR \land \text{card } S = \text{khl} \land (\forall i \in S. \forall j \in PR-S. f j \leq f i)\)])

qed

lemma \textit{kmin-prop}:
fixes \(f :: \text{nat} \Rightarrow \text{Clocktime}\)
shows\\n\((\text{kmin } f \ PR) \subseteq PR \land \text{card } (\text{kmin } f \ PR) = \text{khl} \land (\forall i \in (\text{kmin } f \ PR). \forall j \in PR-(\text{kmin } f \ PR). f i \leq f j)\)
proof-

have \(\text{khl} \leq \text{np} \rightarrow\)
(\(\exists S. S \subseteq PR \land \text{card } S = \text{khl} \land (\forall i \in S. \forall j \in PR-S. f i \leq f j)\))
(is \(\text{khl} \leq \text{np} \rightarrow \text{?P khl}\))
proof\(\text{ (induct } (\text{khl})\))

have \(\text{?P 0 by force}\)
thus \(0 \leq \text{np} \rightarrow \text{?P 0} \).

next

fix \(n\)

assume \(asm: n \leq \text{np} \rightarrow \text{?P } n\)
show \(\text{Suc } n \leq \text{np} \rightarrow \text{?P } (\text{Suc } n)\)
proof

assume \(asm2: \text{Suc } n \leq \text{np}\)
with \(asm\) have \(\text{?P } n\) by simp
then obtain \(S\) where
\(\text{SinPR : } S \subseteq PR\) and
cardS: card $S = n$ and
$HI$: $(\forall i \in S. \forall j \in PR - S. f_i \leq f_j)$
by blast
let $?e = SOME i. i \in PR - S \land
(\forall j \in PR - S. f_i \leq f_j)$
let $?S = 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 $\exists i. i \in PR - S \land (\forall j \in PR - S. f_i \leq f_j)$
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
qed
hence $?P khl$ using khl-bound by auto
then obtain $S$ where
$S \subseteq PR \land \text{card } S = khl \land (\forall i \in S. \forall j \in PR - S. f_i \leq f_j)$ ..
thus $\exists i. i \in S. \forall j \in PR - S. f_i \leq f_j])$
(rule someI [where $P = \lambda S. S \subseteq PR \land \text{card } S = 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 $\exists i. i \in S. \forall j \in PR - S. f_i \leq f_j)$
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
      hence card (kmax f PR ∪ kmin f PR) <= 2 * khl
        using card-Un-le[of kmax f PR kmin f PR] by simp
  }
  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 ∪ kmin f PR)) ≠ {}
    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**

\[
\begin{align*}
 f' \ (PR - (k_{\max} f \ PR \cup k_{\min} f \ PR)) &= \\
 f' \ (PR - (k_{\max} (\lambda p. f p + c) \ PR \cup k_{\min} (\lambda p. f p + c) \ PR))
\end{align*}
\]

**Apply** (unfold \( k_{\min}-\text{def} \) \( k_{\max}-\text{def} \))

**By** simp

**Lemma** max-shift:

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

**Assumes** \( \text{notEmpFin}: S \neq \{\} \) **finite** \( S \)

**Shows**

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

**Proof**

- from \( \text{notEmpFin} \) have \( f'S \neq \{\} \) and \( (\lambda p. f p + x) ' S \neq \{\} \)
  - by auto
- with \( \text{notEmpFin} \) 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 \( \text{thesis by force} \)

**Qed**

**Lemma** min-shift:

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

**Assumes** \( \text{notEmpFin}: S \neq \{\} \) **finite** \( S \)

**Shows**

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

**Proof**

- from \( \text{notEmpFin} \) have \( f'S \neq \{\} \) and \( (\lambda p. f p + x) ' S \neq \{\} \)
  - by auto
- with \( \text{notEmpFin} \) 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 \( \text{thesis by force} \)

**Qed**

2.2.2 Main theorem

**Theorem** trans-inv:

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

**Shows**

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

**Proof**

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

23
(Max (reduce (\lambda\ p. f p + x) \textit{PR}) + \textit{Min} (reduce (\lambda\ p. f p + x) \textit{PR})) / 2
by (unfold \textit{cfnl-def}, simp)
also
have ...
(Max ((\lambda\ p. f p + x) ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) +
\textit{Min} ((\lambda\ p. f p + x) ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) / 2
by (unfold reduce-def, simp)
also
have ...
(Max (f ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) + x +
\textit{Min} (f ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) + x) / 2
proof--
have finite (PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))
by auto
moreover
from reduce-not-empty have
PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}) \neq {}
by (auto simp add: reduce-def)
ultimately
have
Max ((\lambda\ p. f p + x) ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR})))
\textit{=}
Max (f ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) + x
and
Min ((\lambda\ p. f p + x) ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR})))
\textit{=}
Min (f ')
(PR - (\textit{kmax} (\lambda\ p. f p + x) \textit{PR} \cup \textit{kmin} (\lambda\ p. f p + x) \textit{PR}))) + x
using max-shift and min-shift
by auto
thus \textit{thesis} by auto
qed
also
from reduce-shift have ...
(Max (f ')
(PR - (\textit{kmax} f \textit{PR} \cup \textit{kmin} f \textit{PR}))) + x +
\textit{Min} (f ')
(PR - (\textit{kmax} f \textit{PR} \cup \textit{kmin} f \textit{PR}))) + x) / 2
by auto

24
also have ... = \((\text{Max} (\text{reduce } f \ PR) + x) + (\text{Min} (\text{reduce } f \ PR) + x)\) / 2 by (auto simp add: reduce-def)
also have ... = \((\text{Max} (\text{reduce } f \ PR) + \text{Min} (\text{reduce } f \ PR)) / 2 + x\) 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: \(B \subseteq A\) and Css: \(C \subseteq A\) and
cardH: card A + k <= card B + card C
shows k <= card \((B \cap C)\)
proof-
from Bss Css have B \(\cup\) C \(\subseteq\) A by blast
with finitA have card \((B \cup C)\) <= card A
by (simp add: card-mono)
with cardH have
h: k <= card B + card C - card \((B \cup C)\)
by arith
from finitA Bss Css and finite-subset
have finite B \(\land\) 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: \(B \subseteq A\) and Css: \(C \subseteq A\) 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:r ∈ PR−(kmax f PR ∪ kmin f PR) ∧ f j = r
  by blast
hence j ∈ (PR − kmax f PR)
  by blast
from this fj:r asm
show ?thesis using kmax-prop
  by auto
qed
lemma kmin-le:
\[ \forall i \in (\text{kmin } f \text{ PR}), \forall r \in (\text{reduce } f \text{ PR}). f i \leq r \]
proof
fix i assume asm: i \in kmin f PR
show \( \forall r \in \text{reduce } f \text{ PR}. f i \leq r \)
proof
fix r assume asm2: r \in \text{reduce } f \text{ PR}
show f i \leq r
proof
from asm2 and exist-reduce have \( \exists j \in \text{PR}-(\text{kmax } f \text{ PR} \cup \text{kmin } f \text{ PR}). f j = r \) by blast
then obtain j
where fjr: j \in \text{PR}-(\text{kmax } f \text{ PR} \cup \text{kmin } f \text{ PR}) \land f j = r
by blast
hence j \in (\text{PR} - \text{kmin } f \text{ 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:
\[ 0 < c \implies |a \div b / c| = |a - b| / c \]
proof
assume ch: 0 < c
{
fix d :: real
assume dh: 0 <= 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| * d
by (simp add: abs-mult)
finally
have |a * d - b * d| = |a - b| * d
.
The next three lemmas are about the existence of bounds of the values \( \text{Max} (\text{reduce } f \text{ PR}) \) and \( \text{Min} (\text{reduce } f \text{ PR}) \). These are used in the proof of the main property.

**Lemma uboundmax:**

**Assumes**
- \( hC: C \subseteq \text{PR} \) and
- \( hCk: np \leq \text{card } C + \text{kh} \)

**Shows**
\[ \exists \ i \in C. \ \text{Max} (\text{reduce } f \text{ PR}) \leq f i \]

**Proof**

- From \( \text{reduce-not-empty} \) and \( \text{finite-reduce} \)
- Have \( \text{maxrinr}: \ \text{Max} (\text{reduce } f \text{ PR}) \in \text{reduce } f \text{ PR} \)
- By simp
- With \( \text{exist-reduce} \)
- Have \( \exists \ i \in \text{PR} \setminus (\text{kmax } f \text{ PR} \cup \text{kmin } f \text{ PR}). f i = \text{Max} (\text{reduce } f \text{ PR}) \)
- By simp
- Then obtain \( \text{pmax} \) where
  - \( \text{pmax-in-reduc}: \text{pmax} \in \text{PR} \setminus (\text{kmax } f \text{ PR} \cup \text{kmin } f \text{ PR}) \) and
  - \( \text{fpmax-ismax}: f \text{pmax} = \text{Max} (\text{reduce } f \text{ PR}) \)
- Hence \( C \cap \text{insert } \text{pmax} (\text{kmax } f \text{ PR}) \neq \{\} \)

**Proof**

- From \( \text{kmax-prop} \) and \( \text{pmax-in-reduc} \)
- And \( \text{finite-kmax} \) and \( hCk \)
- Have \( \text{card } \text{PR} < \text{card } C + \text{card } (\text{insert } \text{pmax} (\text{kmax } f \text{ PR})) \)
- By simp
- Moreover
- From \( \text{pmax-in-reduc} \) and \( \text{kmax-prop} \)
- Have \( \text{insert } \text{pmax} (\text{kmax } f \text{ PR}) \subseteq \text{PR} \) by blast
- Moreover
- Note \( hC \)
- Ultimately
- Show \( ?\text{thesis} \)
  - Using \( \text{subsets-int}[\text{of PR } C \text{ insert } \text{pmax} (\text{kmax } f \text{ PR})] \)
  - By simp

**Qed**

Hence \( \exists \ i \in C. \ i=\text{pmax} \lor i \in \text{kmax } f \text{ PR} \) by blast

Then obtain \( i \) where
- \( \text{iinC}: i \in C \) and altern: \( i=\text{pmax} \lor i \in \text{kmax } f \text{ PR} \)
- Thus \( ?\text{thesis} \)

**Proof** (cases \( i=\text{pmax} \))

- Case \( \text{True} \)
  - With \( \text{iinC} \) \( \text{fpmax-ismax} \) show \( ?\text{thesis} \) by force

Next
case False with altern maxrinr fpmax-ismax kmax-ge
have \( f \) \( pmax \leq f \) \( i \) by simp
with \( i \in C \) fpmax-ismax show \(?thesis\) by auto
qed
qed

lemma \( \text{lboundmin:} \)
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 minrinr: \( \text{Min} (\text{reduce} f PR) \in \text{reduce} f PR \)
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
\( \text{pmin-in-reduc}: pmin \in PR^-(kmax f PR \cup kmin f PR) \)
and
\( \text{fpmin-ismin}: f pmin = \text{Min} (\text{reduce} f PR) \).
\( \) hence \( C \cap \text{insert} pmin (kmin f PR) \neq {} \)
proof−
from kmin-prop and pmin-in-reduc
\( \) and finite-kmin and hCk have
\( \) card \( PR \leq \text{card} C + \text{card} (\text{insert} pmin (kmin f PR)) \)
by simp
moreover
from pmin-in-reduc and kmin-prop
have 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 \( C: \exists i \in C. i = pmin \lor i \in kmin f PR \) by blast
then obtain \( i \) where
\( \) in\( C: i \in C \) and altern: \( i = pmin \lor i \in kmin f PR \).
\( \) thus \(?thesis\)
proof(cases \( i = pmin \))
\( \) case True
\( \) with \( i \in C \) fpmin-ismin show \(?thesis\) by force
next
\( \) case False
\( \) with altern minrinr fpmin-ismin kmin-le
have \( f \ i \leq f \ p_{\min} \) \textbf{by simp}

with \( \iin C \ f p_{\min} \ - \ i \min \) \textbf{show} \ ?thesis \ \textbf{by auto}

qed

qed

\textbf{lemma} same-bound:

\textbf{assumes}

\begin{itemize}
  \item \( hC: \ C \subseteq PR \) \textbf{and}
  \item \( hCk: \ np \leq \ \text{card} \ C + khl \) \textbf{and}
  \item \( hnk: \ 3 \ast khl < np \)
\end{itemize}

\textbf{shows}

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

\textbf{proof}–

\begin{itemize}
  \item \( \text{have} \ b1: \ khl + 1 \leq \ \text{card} \ (C \cap (PR - k\min f PR)) \)
  \item \textbf{proof}(\text{rule pigeonhole})
    \begin{itemize}
      \item \textbf{show} \ finite \ PR \ \textbf{by simp}
    \end{itemize}
  \item \textbf{next}
    \begin{itemize}
      \item \textbf{show} \ C \subseteq PR \ \textbf{by fact}
    \end{itemize}
  \item \textbf{next}
    \begin{itemize}
      \item \textbf{show} \ PR - k\min f PR \subseteq PR \ \textbf{by blast}
    \end{itemize}
  \item \textbf{next}
    \begin{itemize}
      \item \textbf{show} \ \text{card} \ PR + (khl + 1) \leq \text{card} \ C + \text{card} \ (PR - k\min f PR) \)
    \end{itemize}
  \item \textbf{proof}–
    \begin{itemize}
      \item \textbf{from} \ hnk \ and \ hCk \textbf{ have}
        \begin{itemize}
          \item \text{np} + khl < np + \text{card} \ C - khl \ \textbf{by arith}
        \end{itemize}
      \item \text{also}
        \begin{itemize}
          \item \textbf{from} \ kmin-prop
            \begin{itemize}
              \item \textbf{have} \ \ldots = np + \text{card} \ C - \text{card} \ (k\min f PR)
              \textbf{by} \ \text{auto}
            \end{itemize}
          \item \text{also}
            \begin{itemize}
              \item \textbf{have} \ \ldots = \text{card} \ C + (\text{card} \ PR - \text{card} \ (k\min f PR))
              \textbf{proof}–
                \begin{itemize}
                  \item \textbf{from} \ kmin-prop \textbf{ have}
                    \begin{itemize}
                      \item \text{card} \ (k\min f PR) \leq \text{card} \ PR
                      \textbf{by} \ \text{(intro card-mono, auto)}
                    \end{itemize}
                  \item \textbf{thus} \ ?thesis \ \textbf{by} \ \text{(simp)}
                \end{itemize}
              \end{itemize}
            \end{itemize}
          \item \textbf{also}
            \begin{itemize}
              \item \textbf{from} \ kmin-prop
                \begin{itemize}
                  \item \textbf{have} \ \ldots = \text{card} \ C + \text{card} \ (PR - k\min f PR)
                  \textbf{proof}–
                    \begin{itemize}
                      \item \textbf{from} \ kmin-prop \textbf{ and} \ finite-kmin \textbf{ have}
                        \begin{itemize}
                          \item \text{card} \ PR - \text{card} \ (k\min f PR) = \text{card} \ (PR - k\min f PR)
                          \textbf{by} \ \text{(intro card-Diff-subset[THEN sym])(auto)}
                        \end{itemize}
                      \item \textbf{thus} \ ?thesis \ \textbf{by} \ \text{auto}
                    \end{itemize}
                \end{itemize}
              \item \text{qed}
            \end{itemize}
          \item \textbf{also}
            \begin{itemize}
              \item \textbf{from} \ kmin-prop
                \begin{itemize}
                  \item \textbf{have} \ \ldots = \text{card} \ C + \text{card} \ (PR - k\min f PR)
                  \textbf{proof}–
                    \begin{itemize}
                      \item \textbf{from} \ kmin-prop \textbf{ and} \ finite-kmin \textbf{ have}
                        \begin{itemize}
                          \item \text{card} \ PR - \text{card} \ (k\min f PR) = \text{card} \ (PR - k\min f PR)
                          \textbf{by} \ \text{(intro card-Diff-subset[THEN sym])(auto)}
                        \end{itemize}
                      \item \textbf{thus} \ ?thesis \ \textbf{by} \ \text{auto}
                    \end{itemize}
                \end{itemize}
              \item \text{qed}
            \end{itemize}
          \end{itemize}
        \end{itemize}
      \item \text{finally}
        \begin{itemize}
          \item \textbf{show} \ ?thesis
            \textbf{by} \ \text{(simp)}
        \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
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 \( (C \cap (PR - kmin f PR)) + (PR - kmax g PR) \)
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 h1
    show \?thesis by arith
qed
qed

hence
\( \exists i. i \in C \land i \in (PR - kmin f PR) \land i \in (PR - kmax g PR) \)
by blast
then obtain i where
  in-C: \( i \in C \) and
  not-in-kmin: \( i \in (PR - kmin f PR) \) and
  not-in-kmax: \( i \in (PR - kmax g PR) \) by blast
have \( Min (reduce f PR) <= f i \)
proof(cases \( i \in kmax f PR \))
  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 \( f i \in reduce f PR \)
    by auto
  with reduce-not-empty and finite-reduce
    show \?thesis by auto

31
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 \( \text{thesis} \)
    using kmin-le by blast
next
  case False
  with not-in-kmax
  have \( i \in \text{PR} - (\text{kmax } g\ PR \cup \text{kmin } g\ PR) \)
    by blast
  with reduce-def have \( g\ i \in \text{reduce } g\ PR \)
    by auto
  with reduce-not-empty and finite-reduce
  show \( \text{thesis} \) by auto
qed
moreover
note in-C
ultimately
show \( \text{thesis} \) 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: \( C \subseteq \text{PR} \) and
  hCF: \( np - nF \leq \text{card } C \) and
  hFn: \( 3 \ast nF < np \) 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 hFk and hFk
  have hnk: \( 3 \ast \text{khl} < np \) by arith
let
  \( \text{maxf} = \text{Max}\ (\text{reduce } f\ PR) \)
  and \( \text{minf} = \text{Min}\ (\text{reduce } f\ PR) \)
  and \( \text{maxy} = \text{Max}\ (\text{reduce } g\ PR) \)
and \( \text{?ming} = \text{Min} (\text{reduce g PR}) \)

from abs-distrib-div

have \(|\text{cfnl p f - cfnl q q}| = \)
\(|\text{?maxf} + \text{?minf} + - \text{?maxg} + - \text{?ming}| / 2 \)
by (unfold \text{cfnl-def}) simp

moreover have \(|\text{?maxf} + \text{?minf} + - \text{?maxg} + - \text{?ming}| <= y + 2 * x \)
— The rest of the property can be proved by CVC-lite (see appendix A.2)

proof ( cases 0 <= \text{?maxf} + \text{?minf} + - \text{?maxg} + - \text{?ming} )

case True

hence \(|\text{?maxf} + \text{?minf} + - \text{?maxg} + - \text{?ming}| = \)
\text{?maxf} + \text{?minf} + - \text{?maxg} + - \text{?ming} by arith

moreover from uboundmax hC hCk

obtain \text{mxf}

where \text{mxfinC: mxf} \in C and
\text{maxf: ?maxf <= f mxf} by blast

moreover from lboundmin hC hCk

obtain \text{mng}

where \text{mnginC: mng} \in C and
\text{ming: g mng <= ?ming} by blast

moreover from same-bound hC hCk hnk

obtain \text{mxn}

where \text{mxninC: mxn} \in C and
\text{mxnf: ?minf <= f mxn and}
\text{mxng: g mxn <= ?maxg} by blast

ultimately have

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

also from \text{mxninC} hbx abs-le-D1

have ...
<= (f mxf + - g mng) + x
by auto

also have ...
= (f mxf + - f mng) + (f mng + - g mng) + x
by arith

also have ...
<= y + (f mng + - g mng) + x

proof–

from \text{mxfinC mnginC hby1 abs-le-D1}

have f mxf + - f mng <= y
by auto

thus \text{?thesis}
by auto

qed
also

from \textit{mnginC hbx abs-le-D1}

have \( \ldots \leq y + 2 \ast x \)

by auto

finally

show \textit{thesis}.

next

\begin{itemize}
  \item \textbf{case} False
  \item hence
  \begin{align*}
    |\ ?maxf + \ ?minf + - \ ?maxg + - \ ?ming| &= \\
    \ ?maxg + \ ?ming + - \ ?maxf + - \ ?minf \text{ by arith}
  \end{align*}
\end{itemize}

moreover

from \textit{uboundmax hC hCk}

obtain \textit{mxg}

\begin{itemize}
  \item where \textit{mzxnginC: mxg} \in C \text{ and}
    \item \( \text{mxg: } \ ?maxg \leq g \ mxg \text{ by blast} \)
\end{itemize}

moreover

from \textit{lboundmin hC hCk}

obtain \textit{mnf}

\begin{itemize}
  \item where \textit{mnfinC: mnf} \in C \text{ and}
    \item \( \text{mnf: } f \ mnf \leq \ ?minf \text{ by blast} \)
\end{itemize}

moreover

from \textit{same-bound hC hCk hnk}

obtain \textit{mxn}

\begin{itemize}
  \item where \textit{mzxninC: mxn} \in C \text{ and}
    \item \( \text{mxn: } \ ?ming \leq g \ mxn \text{ and} \)
    \item \( \text{mxng: } f \ mxn \leq \ ?maxf \text{ by blast} \)
\end{itemize}

ultimately

have

\begin{align*}
  |\ ?maxf + \ ?minf + - \ ?maxg + - \ ?ming| &= \\
  (g \ mxg + - f \ mnf) + (g \ mxn + - f \ mn) \text{ by arith}
\end{align*}

also

from \textit{mzxninC hbx}

have \( \ldots \leq (g \ mxg + - f \ mnf) + x \)

by \( \text{(auto dest!: abs-le-D2)} \)

also

have

\( \ldots = (g \ mxg + - g \ mnf ) + ( g \ mnf + - f \ mnf) + x \)

by arith

also

have \( \ldots \leq y + ( g \ mnf + - f \ mnf) + x \)

proof–

\begin{itemize}
  \item from \textit{mzxginC mnfinC hby2 abs-le-D1}
  \item have \( g \ mxg + - g \ mnf \leq y \)
    \item by auto
  \item thus \textit{thesis}
    \item by auto
\end{itemize}
qed
also
from \textit{mnfinC hbx}
have \ldots \leq y + 2 * x
by (auto dest!: abs-le-D2)
finally
show \textit{?thesis}.
qed
ultimately
show \textit{?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 \underline{uboundmax} and \underline{lboundmin}

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

\textbf{theorem} \texttt{accur-pres}:
\textbf{assumes}
\hspace{1cm} \texttt{hC: } C \subseteq PR and
\hspace{1cm} \texttt{hCF: } np - nF \leq \text{card } C and
\hspace{1cm} \texttt{hFk: } nF = khl and
\hspace{1cm} \texttt{hby: } \forall \ l \in C. \forall \ m \in C. \ |f l - f m| \leq y and
\hspace{1cm} \texttt{hqC: } q \in C
\textbf{shows} \ | \texttt{cfnl p f - f q} \ | \leq y
\textbf{proof-}
\hspace{1cm} from \texttt{hCF and hFk}
\hspace{1cm} have \texttt{npleCk: } np \leq \text{card } C + khl by arith
\hspace{1cm} show \textit{?thesis}
\hspace{1cm} 
\hspace{1cm} proof(cases \texttt{f q \leq cfnl p f})
\hspace{1cm} \hspace{1cm} case True
\hspace{1cm} \hspace{1cm} from \texttt{npleCk hC and uboundmax}
\hspace{1cm} \hspace{1cm} have \exists \ i \in C. \ Max (reduce f PR) \leq f i
\hspace{1cm} \hspace{1cm} by auto
\hspace{1cm} \hspace{1cm} then obtain \texttt{pi} where
\hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{hpiC: } pi \in C and
\hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{fpiGeMax: } Max (reduce f PR) \leq f pi by blast
\hspace{1cm} \hspace{1cm} from \texttt{reduce-not-empty}
\hspace{1cm} \hspace{1cm} have \texttt{Min (reduce f PR) \leq Max (reduce f PR)}
\hspace{1cm} \hspace{1cm} by (auto simp add: reduce-def)
\hspace{1cm} \hspace{1cm} with \texttt{fpiGeMax} have
\hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{cfnlLefpi: } cfnl p f \leq f pi
\hspace{1cm} \hspace{1cm} \hspace{1cm} by (auto simp add: cfnl-def)
\hspace{1cm} \hspace{1cm} with \texttt{True} have
\hspace{1cm} \hspace{1cm} \hspace{1cm} \texttt{| cfnl p f - f q | \leq | f pi - f q |}
\hspace{1cm} \hspace{1cm} 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 cfnl p f
by (auto simp add: cfnl-def)
with False have
| 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:

\begin{align*}
\text{lemma abs-distrib-div:} \\
\theta \leq (c:real) & \implies |a / c - b / c| = |a - b| / c
\end{align*}

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.
A.2 Bound for Precision Enhancement property

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

\[
\text{have } |\max (\text{reduce } f \text{ PR}) + \min (\text{reduce } f \text{ PR}) + \max (\text{reduce } g \text{ PR}) + \min (\text{reduce } g \text{ PR})| \leq y + 2 \cdot x
\]

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 \texttt{uboundmax}, \texttt{lboundmin}, \texttt{same_bound} (lemmas about the existence of some bounds), the axiom \texttt{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 \texttt{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 \texttt{bound_prec_enh.cvc}:

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

File \texttt{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 \texttt{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 \texttt{accur_pres.cvc}:

File \texttt{accur_pres.ics}:

References


