Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 687152, 8 pages
http://dx.doi.org/10.1155/2015/687152

Research Article

The Formalization of Discrete Fourier Transform in HOL

Zhiping Shi,[1] Yupeng Zhang,[1] Yong Guan,[1] Liming Li,[1] and Jie Zhang[2]

[1]*Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, Beijing 100048, China*
[2]*College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China*

Correspondence should be addressed to Zhiping Shi; shizp@cnu.edu.cn

Received 6 March 2015; Revised 7 June 2015; Accepted 25 June 2015

Academic Editor: Jean-François Monin

Traditionally, Discrete Fourier Transform (DFT) is performed with numerical or symbolic computation, which cannot guarantee 100% accurate analysis which may be necessary for safety-critical applications. Machine theorem proving is one of the formal methods that perform accurate analysis with completeness to some extent. This paper proposes the formalization of DFT in a higher-order logic theorem prover named HOL. We propose the formal definition of DFT and verify the fundamental properties of DFT. Two case studies are presented to illustrate usefulness and correctness of the formalized DFT, including formal verifications of Fast Fourier Transform (FFT) and cosine frequency shift.

1. Introduction

Fourier Transform (FT), through which signal is transformed from time domain to frequency one, is a fundamental and core method for signal processing. DFT is the discrete version of FT on both time domain and frequency domain which is adapted to the actual environment of computers and digital signal processing (DSP) chips. DFT has been widely used in many fields such as spectral analysis, data compressing, digital communication, signal processing, and the solution of partial differential equations [1, 2]. Therefore tremendous algorithms and packages have been implemented to perform DFT. The algorithms and packages are based on numerical or symbolic computation. The numerical computation intuitively produces approximate solutions because floating-point numbers are used to represent reals in computers. Computer algebra systems (CAS) such as MATLAB, Maple, and Mathematica offer a large collection of powerful algebraic packages and can efficiently perform symbolic computation precisely without round-off. But CAS still can output incorrect results because they have no mechanism to check correctness of results mathematically. Errors can arise in mismatching the side conditions of functions, the determination of equality of two expressions, or the definedness of expressions with respect to symbolic parameters [3, 4]. Moreover both numerical and symbolic computations are implemented in unverified huge computer programs, which are bug prone. Thus the analysis of DFT based on the above mentioned techniques cannot be relied upon for safety-critical applications of which even tiny errors can cause loss of human lives.

Formal methods have succeeded in precise analysis and verification of hardware and software systems in the past decades. Theorem proving, one of the formal methods, models the systems in logic and then reasons and verifies the model's properties using axioms, theorems, and inference rules in the logic. The proof is performed in a theorem prover [5, 6]. The precise analysis and verification of systems employing DFT need the formal implementation of DFT in a theorem prover. Formal methods have been successfully applied in the precise analyses and verifications of hardware and software systems in the past decades. As a formal method, theorem proving can be used to model the systems based on logic and then to reason and verify the model's properties using axioms, theorems, and inference rules in the logic. Generally, the proving tasks are performed in a theorem prover [5, 6]. For example, one needs to implement DFT formally in a theorem prover to perform the precise analyses and verifications based on DFT. In recent years, some

formalization and verification [7–10] have been done on Fast Fourier Transform (FFT), which is the most popular and efficient DFT method. All these works focused on the specific formal implementations of FFT at different levels in diverse algorithms. Bjesse [7] verified FFT hardware at the netlist level. Capretta [8] formalized FFT at the power list level in the Coq theorem prover, while Gamboa [9] performed similar work in ACL2. Akbarpour and Tahar [10] formally verified FFT at different abstraction levels including the real specification level, the floating and fixed-point description level, and the RT and netlist gate levels, and they also analyzed the round-off accumulation errors. Although FFT may be the most widely used implementation of DFT, it cannot take place of DFT completely. DFT has its own theoretical advantages and is much simpler than FFT in both theory and algorithm. Therefore, compared with FFT, DFT has been all along a dominant method in theoretical analysis, especially when theorem proving is used to analyze the functions and properties of a system model. The applications of DFT in theorem proving are very important in the early stages of engineering design. Generally, DFT can be adopted to perform real-time computation on local data in the process of sampling, while FFT computation can only be done when all the data is obtained. Besides, DFT has more advantages over FFT in the choice of sampling rate, the memory consumption of data, and so forth. To the best of our knowledge, no work has been done previously to formalize DFT in any theorem prover. In the present paper, DFT is for the first time formalized in the HOL theorem prover. To some extent, the present work is to develop a theoretical tool rather than to just provide an application. Systematic and detailed formalization of DFT is presented here to support flexible formal analyses of various systems.

The HOL system is a well-developed theorem prover from the University of Cambridge. It is equipped with the formalized mathematic theories needed for DFT such as real analysis [11], complex field [12], and matrix theory [13, 14]. The rest of the paper is organized as follows. After the formal definition of DFT in Section 2, the fundamental properties of DFT are formalized and verified in Section 3. And then in Section 4, two case studies are presented to illustrate the usefulness of the proposed formalization of DFT, including formal analysis of FFT and the cosine frequency shift. Section 5 concludes the paper.

2. Formal Definition of DFT

For a real or complex sequence $x(n)$ with length of N in time domain, DFT transforms it to a new complex sequence $X(k)$ with the same length in frequency domain. The mathematical definition of DFT can be written as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j(2\pi/N)nk}, \quad 0 \le k \le N-1, \tag{1}$$

where $e^{-j(2\pi/N)nk}$, $0 \le n, k \le N-1$, is the frequency spectrum. For simplification, it is substituted by W_N^{nk} in the rest of the paper. The definition can be formalized in HOL as follows.

Definition 1. DFT_def

$$\forall f\, N\, k.\, DFT\ f\ N\ k = \mathrm{csum}(0, N)(\lambda n.f\ n * \exp(i * (-2 * pi/N * n * k))),$$

where $\mathrm{csum}(0, N) f$ accumulates the complex sequence f from number 0 item for N items. The function exp( ) is the base-e exponential function in which i stands for the imaginary unit and pi stands for π. The above mentioned symbols keep the same meanings in the rest of this paper.

When $X = \mathrm{DFT}(x)$, the inverse DFT (IDFT) can be expressed as follows:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j(2\pi/N)nk}, \quad 0 \le n \le N-1. \tag{2}$$

The IDFT can be formalized as follows.

Definition 2. IDFT_def

$$\forall g N\, n.IDFT\ g\ N\ n = \mathrm{csum}(0, N)(\lambda k.g\ k * \exp(i * (2 * pi/N * n * k)))/N,$$

where g stands for the DFT sequence of f with length of N.

3. Formal Verification of the Properties

In this section, the classical properties of DFT are formalized and verified based on Definition 1 in HOL. The formal verification of the properties can check whether or not the definition is correct and reasonable, and moreover the verified properties turn into theorems in HOL and they can be utilized directly to facilitate formal analysis of DFT based systems.

3.1. Some Useful Lemmas. To conveniently read and understand formalizations of properties of DFT, some useful lemmas verified in HOL are presented in this subsection. Consider

$$\forall r.f(r) = g(r) \Longrightarrow \sum_{r=m}^{m+n} f(r) = \sum_{r=m}^{m+n} g(r). \tag{3}$$

The formalization is as follows.

Lemma 3. *Equality of two complex functions' cumulative sum*

$$\forall (f{:}num\text{-} > complex)(g{:}num\text{-} > complex)(m{:}num)(n{:}num).(\forall (r{:}num).m \le r \wedge r < m+n \Rightarrow (f\ r = g\ r)) \Rightarrow (csum(m,n) f = csum(m,n) g),$$

where functions f and g are functions from the natural numbers to complex numbers for convenience to DFT.

Intuitively, W_N^{nk} is an N-periodic complex sequence. It holds that

$$W_N^{n(k+mN)} = W_N^{nk}. \tag{4}$$

The formalization is as follows.

Lemma 4. *Periodicity of W_N^{nk} is*

$$\forall k\, m\, n\, N.\, N <> 0 \wedge 0 \le k \wedge k < N \Rightarrow (\exp(i * (-2 * pi/N * n * (k + m * N))) = \exp(i * (-2 * pi/N * n * k))).$$

For a complex number z, IM z returns the imaginary component of z. One has

$$-(IMz) = IM(-z). \tag{5}$$

Lemma 5. *Inverse operation of imaginary parts of complex numbers*

$$\forall z : complex.\, -IMz = IM(-z).$$

Suppose g is a complex function; it holds that

$$g(0) = 0 \wedge g(x + y) = g(x) + g(y) \Longrightarrow g\left[\sum_{k=n}^{m-1} f(k)\right] = \sum_{k=n}^{m-1} g[f(k)]. \tag{6}$$

The formalization is as follows.

Lemma 6. *Compound operation and accumulating operation*

$$\forall f\, g\, n\, m.(\forall x\, y.(g\, 0 = 0) \wedge (g(x + y) = g\, x + g\, y)) \Rightarrow (g(csum(n, m)\, f) = csum(n, m)(\lambda k.g(f\, k))).$$

For the conjugate operator $*$ and accumulating function, there is a similar property as follows:

$$\left[\sum_{k=n}^{m-1} f(k)\right]^* = \sum_{k=n}^{m-1} [f(k)]^*, \tag{7}$$

where $*$ stands for the conjugate operator of complex numbers. The formalization is as follows.

Lemma 7. *Conjugate operation and accumulating operation*

$$\forall f\, N.\mathrm{conj}(csum(0, N)(\lambda n.f\, n)) = csum(0, N)(\lambda n.\mathrm{conj}(f\, n)),$$

where the function conj *returns the conjugate of a complex number.*

Accumulation of a series of complex numbers can be performed on the real and imaginary parts, respectively. Consider

$$\sum z = \sum RE(z) + \sum IM(z), \tag{8}$$

where the functions RE and IM return the real and imaginary components of a complex number, respectively. The formalization is as follows.

Lemma 8. *Accumulating operation of complex numbers*

$$\forall N : num(z\, w : num \longrightarrow real).csum(0, N)(\lambda n : num.(z\, n, w\, n)) = (sum(0, N)(\lambda n.(z\, n)), sum(0, N)(\lambda n.(w\, n))).$$

3.2. Implicit Periodicity. DFT is derived from the Discrete Fourier Series. Considering W_N^{nk} is an N-periodic sequence, suppose $x(n)$ is a series with N-periodic extension. Then for $X = \mathrm{DFT}(x)$, X is periodic. One has

$$X(k + mN) = X(k), \quad 0 \le k \le N - 1,\ m \in Z. \tag{9}$$

The formal description is as follows.

Theorem 9. *DFT_PERIODIC*

$$\forall k\, m\, N : num.((N) : real <> (0) \wedge (0 \le k \wedge k < N)) \Rightarrow (DFT(f : num- > real)(N : num)(k + m * N) = DFT(f : num \longrightarrow real)(N : num)k).$$

3.3. Linearity. For two N-length sequences $x_1(n)$ and $x_2(n)$, $X_1(k)$ and $X_2(k)$ are their DFT sequences. The linearity is as follows:

$$\mathrm{DFT}(ax_1(n) + bx_2(n)) = aX_1(k) + bX_2(k). \tag{10}$$

We verified this property as the following theorem.

Theorem 10. *DFT_LINEAR*

$$\forall f1\, f2\, f3\, a\, b\, k\, N.(\forall n.0 \le n \wedge n < N \Rightarrow (f3\, n = a * f1\, n + b * f2\, n)) \Rightarrow (DFT\, f3\, N\, k = a * DFT\, f1\, N\, k + b * DFT\, f2\, N\, k).$$

The linearity of DFT shows that it can be applied in discrete linear system. The proof is based on Definition 1 and Lemma 3.

3.4. Symmetry. The time domain and frequency domain of DFT are conjugated dually and written as

$$X^*(-k)_N = \mathrm{DFT}(x^*(n)). \tag{11}$$

The formal description is as follows.

Theorem 11. *DFT_SYMM*

$$\forall k\, N : num.((N) : real <> (0) \wedge (0 \le k \wedge k < N)) \Rightarrow (\mathrm{conj}(csum(0, N)(\lambda(n : num).(f\, n, 0) * \exp(i * (-(2) * pi/N * n * (-k))))) = (csum(0, N)(\lambda(n : num).\mathrm{conj}((f\, n), 0) * \exp(i * (-(2) * pi/N * n * k))))).$$

During verifying the theorem, we need to introduce a subgoal:

$$!n : \mathrm{num.conj}(\exp(i * (k * 2 * pi * inv(N) * n))) = \exp(i * (k * -2 * pi * inv(N) * n)) \tag{12}$$

in order to simplify the goal. Then apply some general operation rules in real script and Lemma 3 to make this theorem proved.

3.5. Conjugated Symmetry. The sequence after DFT is of conjugated symmetry. Consider

$$X^*(k) = X[((-k))_N]. \tag{13}$$

The formal description is as follows.

Theorem 12. *DFT_CONJ_SYMM*

$\forall k\, N{:}num\,(f{:}num \longrightarrow real).\mathrm{conj}(csum(0,N)(\lambda(n : num).\,(f\,n,0) * \exp(i * (-(2) * pi/N * n * k)))) = csum(0,N)\,(\lambda(n : num).\,(f\,n,0) * \exp(i * (-(2) * pi/\,N * n * (-k))))$.

3.6. Frequency Shift. The phase shift of the sequence x in the time domain corresponds to the circular shift of the sequence X in the frequency domain. The frequency shift contains the left shift and the right shift. They can be used in formal verification of cosine frequency shift. One has

$$\mathrm{DFT}\left[x(n)\,e^{-j(2\pi/N)nl}\right] = X\left[((k+l))_N\right], \tag{14}$$

$$\mathrm{DFT}\left[x(n)\,e^{j(2\pi/N)nl}\right] = X\left[((k-l))_N\right]. \tag{15}$$

The left shift and the right shift are formalized into Theorems 13 and 14, respectively.

Theorem 13. *DFT_FREQUENCY_LSHIFT*

$\forall k\, N\, l\, f.$
$N <> 0 \wedge 0 \le k \wedge k < N \Rightarrow (csum(0,N)(\lambda n.n * \exp(i * (-2 * pi/N * n * l)) * \exp(i * (-2 * pi/N * n * k))) = csum(0,N)(\lambda n.f\,n * \exp(i * (-2 * pi/N * n * (k+l)))))$.

Theorem 14. *DFT_FREQUENCY_RSHIFT*

$\forall k\, N\, l\, f.$
$N <> 0 \wedge 0 \le k \wedge k < N \Rightarrow (csum(0,N)(\lambda n.f\,n * \exp(i * (2 * pi/N * n * l)) * \exp(i * (-2 * pi/N * n * k))) = csum(0,N)(\lambda n.f\,n * \exp(I * (-2 * pi/N * n * (k-l)))))$.

3.7. Time Shift. Because of the duality property between the time domain and the frequency one, the circular shift of x in the time domain corresponds to the phase shift of X in the frequency domain. Consider

$$\mathrm{DFT}\left[x(n+l)_N\right] = X(k)\,W_N^{-kl}. \tag{16}$$

The formal description is as follows.

Theorem 15. *DFT_TIME_SHIFT*

$\forall f\, N\, n\, l\, k.$
$N <> 0 \wedge 0 \le k \wedge k < N(csum(0,N)(\lambda n.f(n+l) * \exp(i * (-2 * pi/N * n * k))) = \exp(i * (2 * pi/N * l * k)) * DFT\ f\ N\ k)$.

3.8. Convolution. The convolution can be obtained as the product's IDFT of two sequences' DFT. The property facilitates a remarkable simplification. For two N-length sequences $x_1(n)$ and $x_2(n)$, their circular convolution $y(n)$ is defined as follows:

$$y(n) = \sum_{m=0}^{N-1} x_1(m)\,x_2\left[((n-m))_N\right], \quad 0 \le n \le N-1. \tag{17}$$

The formal definition is as follows.

Definition 16. DFT_CONVOLUTION

$\forall f\, g\, N\, n.$
DFT_CONVOLUTION $f\ g\ N\ n = sum(0,N)(\lambda m.\ f\,m * g(n-m))$.

Let Y be DFT of y; the circular convolution theorem holds as follows:

$$Y(k) = \mathrm{DFT}\left[y(n)\right] = X_1(k)\,X_2(k), \quad 0 \le k \le N-1. \tag{18}$$

The formal theorem is as follows.

Theorem 17. *DFT_CONVOLUTION_TIME*

$\forall f\, g\, N\, n\, k.$
$N <> 0 \wedge 0 \le k \wedge k < N \Rightarrow (csum(0,N)(\lambda n.\ sum(0,N)(\lambda m.f\,m * g(n-m)) * \exp(i * (-2 * pi/N * n * k))) = DFT\ f\ N\ k * DFT\ g\ N\ k)$.

According to the theorem, the convolution can be calculated from the IDFT of Y. Benefiting from Fast FT algorithms, the computing complexity can be reduced greatly.

4. Applications

To illustrate the correctness and usefulness of our work, we formally verify FFT and cosine frequency shift as examples in this section. DFT always has been implemented by employing FFT, which is one of the most important algorithms. The formal verification of FFT is the basis to verify applications which employed FFT. Frequency shift is a significant property which should be taken into account at DFT applications. The two examples are chosen for their representativeness.

4.1. FFT. Without question, DFT is extremely useful in many fields but it often takes too long time to compute DFT directly by the definition, especially for long date sequences. Therefore, many FFT algorithms had been developed to perform DFT quickly. Directly computing DFT of an N-length sequence takes $O(N^2)$ arithmetical operations; by comparison, FFT algorithms take $O(N * \log N)$ arithmetical operations to produce the same result. According to Formula (1) of DFT, the rotating operator W_N^{nk}, which is a periodic complex exponential sequence, enables the complexity reduction.

For the periodic complex-power sequence W_N^{nk}, it holds conjugated symmetry and reducibility as follows:

$$W_N^{-nk} = \left(W_N^{nk}\right)^*. \tag{19}$$

The formalization is as follows.

Lemma 18. *WN_CONJ_SYMM*

$\forall k\, m\, n\, N.N <> 0 \wedge 0 \le k \wedge k < N \Rightarrow (\exp(i * (2 * pi/N * n * k)) = \mathrm{conj}(\exp(i * (-2 * pi/N * n * k))))$,

$$W_{Nm}^{nkm} = W_N^{nk}. \tag{20}$$

The formalization is as follows.

```
FFT:
∀ (k N m:num)(f:num->real) (f1:num->real) (f2:num->real).
    (N <> 0) ∧ (N = 2 ** m) ∧ (0 <= k ∧ k < N DIV 2) ∧ ((f1(r) = f(2*r)) ∧ (f2(r) = f(2*r + 1))) ∧
    (DFT f1 ((N DIV 2):num) k = csum (0,N DIV 2) (λr. f1 r * exp (i * (-2 * pi/(N/2) * r * k)))) ∧
    (DFT f2 ((N DIV 2):num) k = csum (0,N DIV 2) (λr. f2 r * exp (i * (-2 * pi/(N/2) * r * k)))) ∧
    (DFT f N k = csum (0,N) (λn. f n * exp (i * (-2 * pi/N * n * k)))) ∧
    (DFT f N (k + (N DIV 2)) = csum (0,N) (λn. f n * exp (i * (-2 * pi/N * n * (k + (N/2))))))
     ⇒
        ((DFT f N k = DFT f1 ((N DIV 2):num) k + exp (i * (-2 * pi/N * k)) *
        DFT f2 ((N DIV 2):num) k) ∧
        (DFT f N (k + (N DIV 2)) = DFT f1 ((N DIV 2):num) k + exp (i * (-2 * pi/N * (N/2))) *
         exp (i * (-2 * pi/N * k)) * DFT f2 ((N DIV 2):num) k))
        ⇒
            ((DFT f N k = DFT f1 ((N DIV 2):num) k + exp (i * (-2 * pi/N * k)) *
            DFT f2 ((N DIV 2):num) k) ∧
            (DFT f N (k + (N DIV 2)) = DFT f1 ((N DIV 2):num) k - exp (i * (-2 * pi/N * k)) *
             DFT f2 ((N DIV 2):num) k))
```

Algorithm 1: The formal definition of FFT.

Lemma 19. *WN_REDUCE*

$\forall k\, m\, n\, N. N <> 0 \wedge m <> 0 \wedge 0 \leq k \wedge k < N \Rightarrow (\exp(i * (-2 * pi/(N * m) * n * k * m)) = \exp(i * (-2 * pi/N * n * k)))$.

Because of the conjugated symmetry of the rotating operator, some multiplying terms can be merged and the amount of the multiplications reduces by a half roughly; because of the periodicity and the reducibility of the rotating operator, a DFT of a long length sequence can be factorized into multiple DFTs of short sequences, where the shorter the sequence, the lower the computing complexity. Based on the ideas above mentioned, many FFT algorithms had been developed. In this section, we formally verify the well-known radix-2 Cooley-Tukey algorithm.

From the reducibility of W_N^{nk}, taking $N/2$-length DFT and $m = 2$, it holds that

$$W_N^{2nk} = W_{N/2}^{nk}. \tag{21}$$

The formalization is as follows.

Lemma 20. *WN_REDUCE_HALF*

$\forall r\, N\, k\, m.$
$N <> 0 \wedge (N = 2 * * m) \Rightarrow (\exp(i * (-2 * pi/N * (2 * r) * k)) = \exp(i * (-2 * pi/(N/2) * r * k)))$.

For computing DFT of N-length sequence $x(n)$, $X(k) = \mathrm{DFT}(x(n))$, the sequence $x(n)$ is divided into two $N/2$ length subsequences in terms of the parity of n, as follows:

$$\begin{aligned} x_1(r) &= x(2r), \quad r = 0, 1, \ldots, \frac{N}{2} - 1, \\ x_2(r) &= x(2r+1), \quad r = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{22}$$

Suppose that $X_1(k) = \mathrm{DFT}(x_1(r))$, $X_2(k) = \mathrm{DFT}(x_2(r))$, $0 \leq k \leq N/2 - 1$; FFT can be described as follows:

$$\begin{aligned} X(k) &= X_1(k) + W_N^k X_2(k), \quad 0 \leq k \leq \frac{N}{2} - 1, \\ X\left(k + \frac{N}{2}\right) &= X_1(k) - W_N^k X_2(k), \quad 0 \leq k \leq \frac{N}{2} - 1. \end{aligned} \tag{23}$$

FFT is consisting of the inversion processing and the butterfly shaped computation. And according to the above expressions, the FFT of $x(n)$ is the concatenation of Formula (23). The formal description of FFT is shown in Algorithm 1.

The FFT can be formally verified as an implementation of DFT. Firstly, use DEF_def and WN_REDUCE_HALF to rewrite the goal. We also need to introduce a subgoal "$\exp(i * (-2 * pi/N * (N/2))) = -1$" to simplify the current goal. Then we can use some tactics in hol4 and some appropriate operation rules in real script and complex script to extend and transform the goal as well as WN_PERIODIC, WN_CONJ_SYMM, WN_REDUCE, and other properties mentioned above. The procedure of validation is shown in Algorithm 2.

4.2. Cosine Frequency Shift. Frequency shift is an important property of DFT, and it can derive many other similar properties like cosine frequency shift. This subsection presents formal verification of cosine frequency shift. A typical cosine frequency shift can be formulated as follows:

$$\begin{aligned} &\mathrm{DFT}\left[x(n)\cos\left(\frac{2\pi m}{N}n\right)\right] \\ &\quad = \frac{1}{2}\left[X(k-m) + X(k+m)\right]. \end{aligned} \tag{24}$$

To verify the formula above using the frequency shift property of DFT, $\cos(n)$ should be transformed to exponential formula with Euler's formula as follows.

```
REWRITE_TAC [DFT_def, WN_REDUCE_HALF] THEN REPEAT STRIP_TAC
THENL
    //the initial goal was decomposed into 2 subgoals
    [ASM_REWRITE_TAC []
    //the 1st subgoal proved!,
    POP_ASSUM MP_TAC THEN
    Q.SUBGOAL_THEN "exp (i * (-2 * pi/N * (N/2))) = -1" ASSUME_TAC
    //the 1st auxiliary subgoal was added!
    THENL
        [REWRITE_TAC [EXP_IMAGINARY] THEN
        Q.SUBGOAL_THEN "-(2) * (pi:real)/N * (N/2):real = -pi" ASSUME_TAC
        THENL
            [REWRITE_TAC[real_div] THEN ONCE_REWRITE_ TAC
            [REAL_MUL_ASSOC] THEN REWRITE_TAC [GSYM REAL_NEG_LMUL] THEN
            ONCE_REWRITE_TAC [REAL_MUL_SYM] THEN
            REWRITE_TAC [GSYM REAL_MUL_ASSOC] THEN
            Q.SUBGOAL_THEN "inv (N:real) * N:real = 1:real" ASSUME_TAC
            //the 2nd auxiliary subgoal was added!
            THENL
                [REWRITE_TAC [GSYM REAL_1] THEN
                MATCH_MP_TAC REAL_MUL_LINV THEN REWRITE_TAC [REAL_0] THEN
                METIS_TAC [],
                ASM_REWRITE_TAC [] THEN
                REWRITE_TAC [GSYM real_1, REAL_MUL_RID] THEN REWRITE_TAC
            [REAL_MUL_ASSOC] THEN Q.SUBGOAL_THEN "inv (2:real) * 2:real = 1:real"
            ASSUME_TAC
                //the 3rd auxiliary subgoal was added!
                THENL
                    [Q.SUBGOAL_THEN "inv 2:real <> 0:real" ASSUME_TAC
                    //the 4th auxiliary subgoal was added!
                    THENL
                        [REWRITE_TAC [REAL_INV_1OVER]
                        THEN SRW_TAC[] [],
                        REWRITE_TAC [GSYM REAL_1]
                        THEN MATCH_MP_TAC REAL_MUL_LINV
                        THEN REWRITE_TAC [REAL_0]
                        THEN SRW_TAC[] []],
                    ASM_REWRITE_TAC []
                    THEN SRW_TAC [] [GSYM real_1, REAL_MUL_LID]]],
                ASM_REWRITE_TAC []
                THEN REWRITE_TAC[COS_NEG, SIN_NEG] THEN
                REWRITE_TAC [COS_PI, SIN_PI] THEN
                REWRITE_TAC [COMPLEX_1] THEN
                REWRITE_TAC [complex_of_real, complex_neg] THEN REWRITE_TAC [RE, IM,
            FST, SND]
            ]
            //all the auxiliary subgoals was proved!
        ONCE_ASM_REWRITE_TAC [] THEN
        ONCE_REWRITE_TAC [COMPLEX_MUL_COMM] THEN
        REWRITE_TAC [GSYM COMPLEX_NEG_LMUL] THEN
        REWRITE_TAC [COMPLEX_MUL_LID] THEN
        REWRITE_TAC [GSYM COMPLEX_NEG_RMUL] THEN
        METIS_TAC [GSYM complex_sub]]
        //the second subgoal proved!
    ]
```

Algorithm 2: The validation of FFT.

```
val EULER_COS = store_thm("EULER_COS",
"∀n:real. (cos n,0) = (1)/(2) ∗ (exp (i ∗ n) + exp (i ∗ (-n)))",
GEN_TAC THEN REWRITE_TAC [EXP_IMAGINARY] THEN
REWRITE_TAC [complex_add, RE, IM, FST, SND] THEN
REWRITE_TAC [COS_NEG, SIN_NEG] THEN
SRW_TAC [][REAL_DOUBLE] THEN
REWRITE_TAC [complex_scalar_lmul, RE, IM, FST, SND]
THEN SRW_TAC [] [REAL_MUL_ASSOC])
```

Algorithm 3: The formalization of EULER_COS.

```
val DFT_MOD_COS = store_thm("DFT_MOD_COS",
" ∀ (k N m:num)(f:num->real).
((N):real <> (0) ∧ (0 <= k ∧ k < N)) ∧
(DFT f n (k - m) = csum(0,N) (\(n:num). (f n) ∗ exp(i ∗ ((-2) ∗ pi/N ∗ n ∗ (k - m))))) ∧
(DFT f n (k + m) = csum(0,N) (\(n:num). (f n) ∗ exp(i ∗ ((-2) ∗ pi/N ∗ n ∗ (k + m)))))
⇒
    ((csum(0,N) (\(n:num). (f n ∗ (cos((2) ∗ pi/N ∗ n ∗ (m)),0) ∗ exp (i ∗ (-2 ∗ pi/N ∗ n ∗ k))))) = (1)/(2) ∗
(DFT f n (k-m)) + (1)/(2) ∗ (DFT f n (k + m)))"
REPEAT STRIP_TAC THEN ASM_REWRITE_TAC [] THEN REWRITE_TAC[EULER_COS] THEN
REWRITE_TAC [COMPLEX_ADD_SCALAR_LMUL, COMPLEX_ADD_RDISTRIB, CSUM_ADD] THEN
Q.SUBGOAL_THEN "∀ (a b:real) c:complex d:complex. a ∗ (b ∗ c) ∗ d = b ∗ (a ∗ c ∗ d)" ASSUME_TAC
//the 1st auxiliary subgoal was added!
THENL
    [REPEAT STRIP_TAC THEN REWRITE_TAC[COMPLEX_LMUL_SCALAR_LMUL] THEN
    ONCE_REWRITE_TAC [COMPLEX_SCALAR_MUL_COMM] THEN
    GEN_REWRITE_TACLAND_CONV [GSYM COMPLEX_SCALAR_MUL_COMM] THEN
    GEN_REWRITE_TAC LAND_CONV [COMPLEX_SCALAR_LMUL] THEN
ONCE_REWRITE_TAC [COMPLEX_SCALAR_MUL_COMM] THEN
    SRW_TAC [] [GSYM COMPLEX_SCALAR_RMUL]
    //the 1st auxiliary subgoal proved!,
    ASM_REWRITE_TAC [] THEN REWRITE_TAC [CSUM_RMUL] THEN
    REWRITE_TAC [GSYM COMPLEX_ADD_SCALAR_LMUL] THEN
    AP_TERM_TAC THEN REWRITE_TAC [real_div, REAL_NEG_LMUL] THEN
    REWRITE_TAC [GSYM real_div] THEN
    BINOP_TAC
    //The initial goal was decomposed into 2 subgoals
    THENL
        [MATCH_MP_TAC DFT_FREQUENCY_RSHIFT THEN
        METIS_TAC []
        //The 1st subgoal proved,
        MATCH_MP_TAC DFT_FREQUENCY_LSHIFT THEN
        METIS_TAC []
        The 2nd subgoal proved
        ]
    ]
```

Algorithm 4: The formalization of cosine frequency shift.

Lemma 21. *EULER_COS:*

$$\cos(n) = \frac{1}{2}\left(e^{in} + e^{-in}\right). \tag{25}$$

Its formalization is shown in Algorithm 3.

Based on the definition of DFT and the lemma EULER_COS, Formula (25) can be formally described and verified as in Algorithm 4.

In the description of cosine frequency shift as shown in Algorithm 4, $\mathrm{DFT}[x(n)\cos((2\pi m/N)n)]$ is directly expanded by the definition of DFT to facilitate the proving process.

5. Conclusion

DFT is widely used in DSP and linear time invariant (LTI) systems which could be in safety-critical fields where formal methods are expected to be employed to assure accurate and

complete verification. This paper presented the formalization of DFT in HOL, including formal definition of DFT, formal verification of properties of DFT, and two case studies demonstrating usefulness and correctness of our work. Our work can be equipped as a library of HOL to facilitate formal verification of systems employing DFT.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors thank Professor Shengzhen Jin for his helpful suggestions. This work was supported by the International Cooperation Program on Science and Technology (2010DFB10930 and 2011DFG13000), the National Natural Science Foundation of China (61170304, 61104035, 61373034, 61303014, and 61472468), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality, and the Scientific Research Base Development Program of the Beijing Municipal Commission of Education (TJSHG201310028014).

References

[1] S. W. Smith, *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, Newnes, 2002.

[2] E. O. Brigham, *The Fast Fourier Transform and Its Applications*, Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[3] A. Adams, M. Dunstan, H. Gottliebsen, T. Kelsey, U. Martin, and S. Owre, "Computer algebra meets automated theorem proving: integrating maple and PVS," in *Theorem Proving in Higher Order Logics*, R. J. Boulton and P. B. Jackson, Eds., vol. 2152 of *Lecture Notes in Computer Science*, pp. 27–42, Springer, Berlin, Germany, 2001.

[4] K. Homann and J. Calmet, "Combining theorem proving and symbolic mathematical computing," in *Integrating Symbolic Mathematical Computation and Artificial Intelligence: Second International Conference, AISMC-2 Cambridge, United Kingdom, August 3-5, 1994 Selected Papers*, J. Calmet and J. A. Campbell, Eds., vol. 958 of *Lecture Notes in Computer Science*, pp. 18–29, Springer, Berlin, Germany, 1995.

[5] M. J. C. Gordon, "Mechanizing programming logics in higher-order logic," in *Current Trends in Hardware Verification and Automated Theorem Proving*, pp. 387–439, Springer, 1989.

[6] L. Liu, O. Hasan, and S. Tahar, "Formal reasoning about finite-state discrete-time Markov chains in HOL," *Journal of Computer Science and Technology*, vol. 28, no. 2, pp. 217–231, 2013.

[7] P. Bjesse, "Automatic verification of combinational and pipelined FFT circuits," in *Computer Aided Verification*, vol. 1633 of *Lecture Notes in Computer Science*, pp. 380–393, Springer, Berlin, Germany, 1999.

[8] V. Capretta, "Certifying the fast fourier transform with Coq," in *Theorem Proving in Higher Order Logics*, vol. 2152 of *Lecture Notes in Computer Science*, pp. 154–168, Springer, 2001.

[9] R. A. Gamboa, "The Correctness of the Fast Fourier Transform: A Structural Proof in ACL2," *Formal Methods in System Design*, Special Issue on UNITY, 2002.

[10] B. Akbarpour and S. Tahar, "A methodology for the formal verification of FFT algorithms in HOL," in *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD '04)*, pp. 37–51, Austin, Tex, USA, November 2004.

[11] J. Harrison, "Constructing the real numbers in HOL," *Formal Methods in System Design*, vol. 5, no. 1-2, pp. 35–59, 1994.

[12] Z. Shi, L. Li, Y. Guan, X. Song, M. Wu, and J. Zhang, "Formalization of the complex number theory in HOL4," *Applied Mathematics & Information Sciences*, vol. 7, no. 1, pp. 279–286, 2013.

[13] Z. Shi, Y. Zhang, Z. Liu et al., "Formalization of matrix theory in HOL4," *Advances in Mechanical Engineering*, vol. 6, Article ID 195276, 16 pages, 2014.

[14] Z. Shi, Z. Liu, Y. Guan, S. Ye, J. Zhang, and H. Wei, "Formalization of function matrix theory in HOL," *Journal of Applied Mathematics*, vol. 2014, Article ID 201214, 10 pages, 2014.