On the Deterministic Qubit

Howon Lee

May 17 2020

Abstract

The deterministic qubit is the propositional variable. Its measurement is logical interpretation. A classical deterministic Fourier and Hadamard transform is presented with respect to a propositional data structure inspired from quantum states. These integral transforms match both the speed and the limitations of the quantum integral transforms. In addition, a way to handle entangling with similar speed and limitations is also given.

1 Introduction

It occurred to me that the essential portion of the quantum speedup of the Fourier-like transforms was the tensor product structure, not the amplitude semantics. But proper qubits are not necessary for this. This thought has strong humor value.

This viewpoint goes through for Hadamard transform and Fourier transform. This portion constitutes a simpler deterministic replication of the work of [1] [11], although without the use of either of their notation.

But they were not able to implement interesting quantum algorithms classically. At the root, this is because to do anything interesting entanglement is needed.

This may seem to require the amplitude semantics. This is not necessarily the case. A deterministic point of view on entanglement is given which has some caveats but is still fast and classical. This is the putative original contribution.

The underlying trick to the data structure is liberal usage of the Kronecker operators, Kronecker sum and Kronecker product. They are used *without* evaluation, upon a linear representation of propositional variables. They stay compressed in this way while we apply operators to them, and give us answers only upon interpretation.

2 Data Structure

Boolean functions can be represented by a vector that contains their outputs, given inputs in canonical order. This is of interest because Boolean functions, if you think of the output vectors as representing numbers, are more compressed representations of numbers than bit vectors are.

Consider a propositional variable A. We imagine that the literals can actually be in 4 states:

$$\begin{gathered}\text{Affirm} : A \\ \text{Negate} : \neg A \\ \text{Tautology} : \top \\ \text{Nullity(contradiction)} : \bot\end{gathered}$$

Construe these as having a mapping to 2-vectors.

$$\begin{gathered}\text{Affirm} : [01]^T \\ \text{Negate} : [10]^T \\ \text{Tautology} : [11]^T \\ \text{Nullity} : [00]^T\end{gathered}$$

It is also useful to think of a phase shifted tautology: $\top^\theta = [1, -1]^T$. This is the same logical value if constrained to be in $GF(2)$ but we don't think of it that way, in much the same way as the two trivial Hadamard superpositions get the same probability when measured, but are out of phase. Think of it as being on the unit circle in L^∞ norm.

Also note that contradiction is not on the Bloch sphere at all.

Given a Boolean function with corresponding propositional sentence of n variables, the function can represent 2^{2^n} numbers, as opposed to the bitvector of n bits, which can only represent 2^n numbers. However, in most cases the propositional sentences are much harder to work with, and the number of literals (as opposed to the number of variables) can explode. This work constitutes an exploration of the ways one can feasibly work with them for certain tasks.

You can factorize a Boolean function composed only of ANDs into a series of Kronecker products of ket-like vectors, applying the above mapping. I am not a physicist at all so don't expect too many actual bras or kets.

$$\begin{aligned}\neg A \wedge \neg B \wedge \neg C &= [10]^T \otimes [10]^T \otimes [10]^T \\ &= [10000000]^T\end{aligned}$$

Despite all the talk about bitvectors, there is nothing preventing these vectors from being complex-valued, negative-valued, etc. Of course that sometimes does not have meaning as a propositional sentence, but they still have meaning as states.

Note that the factored representations can be thought of as a compressed representation of the expanded representation. We wish to stay in that compressed representation all the time.

The Boolean ring also needs XOR. Construe XOR as Kronecker sum.

$$\neg A \wedge \neg B \oplus \neg C = [10]^T \otimes [10]^T \oplus [10]^T$$

Note that I overload $\oplus$ as both XOR and Kronecker sum.

To have a Kronecker sum for vectors, denote k_i as the full superposition vector $[111...111]^T \in \mathbb{C}^i$. Sometimes it is worth thinking of that vector as the Hadamard superposition of a bunch of 0 basis qubits. Consider k_c and k_d as sized to make the below operation make sense.

$$C \oplus D = C \otimes k_c + k_d \otimes D$$

The $+$ is not $\oplus$, it denotes an ordinary sum (an ordinary XOR construed on the whole bitvector, in this situation).

Therefore,

$$[10]^T \otimes [10]^T \oplus [10]^T = [01101010]^T$$

2.1 Interpretation of Data Structure

The propositional variable can then be seen as a sort of deterministic qubit. A is $|1\rangle$, $\top^{\theta}$ is $|-\rangle$, and so on.

Interpretation of the formula becomes, then, a sort of deterministic quantum measurement. Interpretation consists of indexing into the overall bitvector which the propositional formula is representing. This is the same as assigning a truth-value to the variables, as in ordinary logical interpretation. But there is the same phenomenon where interpretation consumes the formula, just as measurement consumes qubits.

A very material problem is finding the correct interpretation. Picking a random interpretation often degenerates into a probabilistic argument, so if one is doing interference a satisfying interpretation must be found. But the point of view on entangling given below seems to create polyspace XORSAT instances if the initial proposition is polyspace with regards to input, which happens in much of practical quantum computing. This seems to be the most material possible objection to this point of view.

The bitvector which the prepositional formula represents is the result column of the truth table of the prepositional formula.

The claim is that this data structure allows us to do nontrivial computations on the Boolean function, without actually expanding the output of the Boolean function into a large bit vector.

To interpret something nontrivial such as $[1, e^{2\pi i}]^T \otimes [1, 1]^T$, take the values in the corresponding indices according to the interpretation and then multiply for AND, add for XOR, etc. So if the two literals belonged to two different variables, and we were negating both, the result would be $e^{2\pi i} * 1 = e^{2\pi i}$. This is because interpretation is indexing into the overall vector which the proposition represents.

Often this is not precisely what was wanted, if we want to stay as a pure state. This is because the Bloch sphere in the L^∞ norm ends up being the Bloch cube. I believe the easiest way to fix this is to redo the definitions of sine and cosine, such that they conform with the unit circle in L^∞ being a square, and the Pythagorean relation ends up being such that,

$$\max(|\cos(x)|, |\sin(x)|) = 1$$

Then, Euler's equation $e^{ix} = \cos x + i \sin x$ has its usual meaning, just with a different definition of sine and cosine. You have to be careful with this, but most Fourier analysis can be carried through without material changes.

A prepositional product state (PPS) is a preposition which has only ANDs. In its linear form, then, it would have only Kronecker products, and so is unentangled. Entanglement consists of not being representable as a PPS, and therefore having to be a summation of different prepositional states. Entanglement is discussed more in section 4.1.

2.2 Properties of Kronecker operators

Here is the mixed-product property of the Kronecker product:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

With multiple Kronecker products, this turns into:

$$(\bigotimes_i A_i)(\bigotimes_i X_i) = \bigotimes_i (A_i X_i) \tag{1}$$

Consider also the case of Kronecker sums. If the left side of a compound left multiplication consists of a Kronecker product, we can also have a sort of mixed-product property.

We only care about cases where the left side is a Kronecker product because for the linear operators we are interested in, that will always be the case, and we deal with the entangling of the Fourier transform differently.

The right side will correspond to the linear propositional sentences and therefore could also have Kronecker sums.

$$\begin{aligned}(A \otimes B)(C \oplus D) &= (A \otimes B)(C \otimes k + k \otimes D)\\ &= (AC \otimes Bk_b) + (Ak_a \otimes BD)\end{aligned}$$

Consider for more iterated Kronecker sum.

$$\begin{gathered}(A \otimes B \otimes C)(D \oplus E \oplus F)\\ = (AD \otimes Bk_b \otimes Ck_c) + (Ak_a \otimes BE \otimes Ck_c) + (Ak_a \otimes Bk_b \otimes CF)\end{gathered}$$

In the general case,

$$(\bigotimes_i A_i)(\bigoplus_j B_j) = \sum_j (\bigotimes_i A_i M_i j) \tag{2}$$

where $M_i j$ is B_j if Kronecker delta $\delta_{ij} = 1$ and k_i otherwise.

And, of course, Kronecker products and sums may be mixed, in which case both equations 1 and 2 may be used. But we can left multiply by a linear operator that factorizes out under the Kronecker *product* only.

Consider $(\bigotimes_i X_i)(\bigodot_i Y_i)$, where $\odot \in \{\otimes, \oplus\}$.

$$(\bigotimes_i X_i)(\bigodot_i Y_i) = (X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(Y_1 \odot (\bigodot_{i \in 2\ldots} Y_i)) \tag{3}$$

Case if $\odot = \otimes$:

$$(X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(Y_1 \otimes (\bigodot_{i \in 2\ldots} Y_i)) = (X_1 Y_1) \otimes ((\bigotimes_{i \in 2\ldots} X_i)(\bigodot_{i \in 2\ldots} Y_i))$$

Case if $\odot = \oplus$:

$$\begin{gathered}(X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(Y_1 \oplus (\bigodot_{i \in 2\ldots} Y_i))\\ = (X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(Y_1 \otimes k_a + k_b \otimes (\bigodot_{i \in 2\ldots} Y_i))\end{gathered}$$

where a and b are just chosen to make the product work.

$$= (X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(Y_1 \otimes k_a) + (X_1 \otimes (\bigotimes_{i \in 2\ldots} X_i))(k_b \otimes (\bigodot_{i \in 2\ldots} Y_i))$$

$$= (X_1 Y_1) \otimes ((\bigotimes_{i \in 2\ldots} X_i) k_a) + X_1 k_b \otimes ((\bigotimes_{i \in 2\ldots} X_i)((\bigodot_{i \in 2\ldots} Y_i)))$$

In the implementation, you cannot recurse naively because you need to deal with the associative structure of the right term. The left term associates because Kronecker product associates with itself, but it doesn't associate with Kronecker sums (just like ordinary products and sums), so the right term may have structure which is trivial but annoying to deal with.

So, iff we stay in the propositional form, at each step we add n new terms if $\odot = \oplus$ and 1 new term if $\odot = \otimes$. Therefore, left multiplication by a linear operator factorable into Kronecker product can only square the size of the proposition.

I highly suspect that you could do better than $O(p^2)$ where the p is the number of terms in the proposition, but this suffices for now.

3 Propositional Integral Transforms

3.1 Propositional Hadamard Transform

The linear operator of interest here is the matrix which corresponds to the Walsh-Hadamard (Hadamard) transform.

The Hadamard transform without normalization in matrix form is defined as such:

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$H_n = H_1 \otimes H_{n-1}$$

I omit the normalization in calculations but you can add it whereever you'd like or omit if you are thinking that you are working in L^∞. The deterministic semantics of interpretation means that it usually doesn't matter.

Applying the transform on a vector consists of multiplying the vector by a matrix.

$$r = Hv$$

But note that, in the case where the Boolean function corresponding to the vector is a propositional product state, both the matrix and the vector are composed of Kronecker products of small matrices and vectors.

$$r = (H_1 \otimes H_2 \otimes H_3 ...)(v_1 \otimes v_2 \otimes v_3 ...)$$

Then use equation 1 to factor things out:

$$r = (H_1 v_1) \otimes (H_2 v_2) \otimes (H_3 v_3) ...$$

Note that this is a propositional state still. Therefore, keeping it as a PPS, the only computation that needs to be done is the factored matrix-vector multiplications.

The size of the propositional state is $O(\log N)$ with respect to N, the size of the bit vector which the state represents. The computation for one interpretation is linear with respect to the size of the propositional state. But full expansion would take $O(N \log N)$ time, where there are N interpretations that take $\log N$ time each, so the speedup is only for when one interpretation is asked for.

Consider $v_1 = [01]^T, v_2 = [10]^T, v_3 = [01]^T$.

$$Hv = H_1 v_1 \otimes H_1 v_2 \otimes H_1 v_3$$

$$Hv = ([1, -1]^T \otimes [1, 1]^T \otimes [1, -1]^T)$$

And we are in superposition. Of course the operator is still involutive and unitary (if scaled) and so on.

3.2 Propositional Fourier Transform

The quantum Fourier transform [3], written in bra-ket notation goes like this:

$$\text{QFT}(|x_1 x_2 \ldots x_n\rangle) = \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i\,[0.x_n]}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\,[0.x_{n-1}x_n]}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i\,[0.x_1 x_2 \ldots x_n]}|1\rangle\right)$$

where

$$[0.x_1 \dots x_m] = \sum_{k=1}^{m} x_k 2^{-k}.$$

There are two parts to the quantum Fourier transform: the Hadamard transform and the controlled phase gates. The Hadamard transform is detailed above. Here, noting that we don't have a no-cloning theorem for propositional states, we follow Griffiths and Niu[6] in making a measurement and using the results of the measurement to disentangle the controlled phase gate into a single-qubit phase gate. This they call the semiclassical QFT. Unlike in the actual semiclassical QFT, if the transform is wanted before the last step of the computation, one can just copy the state.

Time is $O((\log N)^2)$, as fast as the non-semiclassical QFT. This is with respect to N, the cardinality of the bit vector that the PPS is representing. Again there is the caveat that this holds only when one interpretation is asked for, also similar to quantum Fourier transform.

This is what is required to apply the Fourier transform on a PPS. So given a PPS as a list of Numpy arrays, in Python this ends up being something like:

```
import numpy as np
from typing import List, Tuple

def complex_had(inp):
    return inp.dot(sci_lin.hadamard(2)).astype(np.complex128)

def omega_m(m):
    return np.exp(2 * np.pi * 1j / (2 ** m))

def apply_fourier(vecs: List[np.array], interp: Tuple[int]):
    res = 1.0 + 0.0j
    for vec_idx, vec in enumerate(vecs):
        # bakes in assumption that it's a product state.
        # to unbake it, need to rewrite to handle Kron sum also
        res *= complex_had(vec)[interp[vec_idx]]
        if interp[vec_idx] == 1:
            for offset, other_vec_idx in enumerate(range(vec_idx + 1, len(vecs))):
                vecs[other_vec_idx][1] *= omega_m(offset + 2)
    return res
```

The function is given the interpretation explicitly as a tuple equal in size to the list of Numpy arrays because, again, we can get away with it, with no no-cloning theorem. However, it is also straightforward to actually do the semiclassical Fourier transform by measuring one qubit at as time, as in [6].

This ends up taking the Fourier transform in the Bloch sphere for L^2 norm. Extension to the Bloch cube and number-theoretic Fourier transform is also straightforward.

4 Entangling and Oracles

4.1 Entangling

Superposition in and of itself is not special. Proper quantum algorithms also have entangling.

This is a deterministic Bell state.

$$|\Psi^+\rangle = \neg A \oplus \neg B$$

This is the XOR of $\neg A$ and $\neg B$, which if we are expanding the formula ends up being Kronecker sum. This is because, expanded out into vector form, the vector is $[0110]^T$, which corresponds to an ordinary quantum Bell state.

So deterministic 'nonlocality' ends up being the one-time pad [9], which of course is an entirely classical phenomenon, if a very special one.

Consider the CNOT gate on $(H_1\neg A) \otimes \neg B$, where the CNOT operator is the usual one.

$$(H_1\neg A) \otimes \neg B = [1010]^T$$

$$\mathbf{CNOT}((H_1\neg A) \otimes \neg B) = [1001]^T = \neg A \oplus B$$

So, you can construe the CNOT gate as an operator on the propositional formula, entangling it into a Bell state, just as with the ordinary quantum states. On the level of the individual interpretations, it applies as a matrix operator, but it is possible to also examine it only with respect to the overall Boolean formula, where it involutes the Hadamard operator on A, changes the $\otimes$ to an $\oplus$ (then back, if applied again), and negates B.

Taken that way, the overall CNOT operation is involutive in the same way that adding a variable XORed to the formula is involutive.

If the entangled state can still be represented as one propositional formula, with using Kronecker sums, equation 2 means that an application of a linear operator factorable by tensor product on it gets only a polynomial number of propositional formulas. So this is the way of dealing with entanglement without an exponential expansion of sums.

If we expanded the Kronecker sums into sums of Kronecker products, we would get into the exponential expansion of terms which we are trying to dealing with by using entangled states. So keep the Kronecker sums as they are, and just do interpretations at the end, just like how we just measure at the end in an ordinary quantum computation.

4.2 Oracles

It looks like true black-box oracles are not possible, we need propositional formulas for the functions which we are treating as oracles to shuffle them around. So Deutsch's remains a creature of proper quantum algorithms only. I will still call them oracles, but be mindful of this point.

Importantly, this is why the proven quantum lower bounds don't apply.

But modular exponentiation, among all sorts of other fun functions, is not a black box, so I suspect this is all still of interest.

Consider a half adder. It can be constructed with two propositional formulae, for the x_1, sum, and x_2, carry:

- $x_1 = A \oplus B$

- $x_2 = A \wedge B$

Obviously, with the AND, the system is not unitary. We have a 4-variable formula to represent inputs and outputs, and we want a unitary transformation corresponding to the half adder. Call this formula:

$$z = y_1 \wedge y_2 \wedge y_3 \wedge y_4$$

With the consideration that $y_i \in \{\bot \top A \neg A\}$ before the oracle is applied.

You could actually construct the entire oracle as a matrix, interpret the formula for all interpretations, and apply the matrix (but note that there's an exponential slowdown with respect to the quantum version there), but note that we can just work entirely in the propositional formula space, without exponential slowdown.

$$U_f z = y_1 \wedge y_2 \wedge y_3 \oplus (x_1) \wedge y_4 \oplus (x_2)$$

$$U_f z = y_1 \wedge y_2 \wedge y_3 \oplus (y_1 \oplus y_2) \wedge y_4 \oplus (y_1 \wedge y_2)$$

So if you are stricken with exponential size of the propositional formula, this won't help you. But many important quantum applications don't have this problem, or only sometimes have this problem. It seems, most importantly, that Shor's doesn't have this problem, because folks write down small circuits for modular exponentiation[10] all the time.

There is not a strict requirement of unitarity with operations on propositional sentences, unlike with qubit registers. Therefore, there is also a simpler way to do the oracle, which is to just apply the function to the inputs. This has the problem of changing the register size, but the above also effectively changes the register size.

$$z = (y_1 \oplus y_2) \wedge (y_1 \wedge y_2)$$

Also do not forget that there is no no-cloning theorem for propositional states.

To get the phase oracle, initialize the initial variables to $|-\rangle$, or $\top^\theta$, or the phase shifted tautology, to get superposition right off the bat and then apply the oracle.

4.3 Fractal Structure

The prepositional state data structure is actually leftovers from a particularly spirited but failed attempt to attack number partitioning. The columns of a table for the dynamic programming solution to number partition [8] can be construed as large sets, which are bitvectors but could be represented as propositions as in the way I described: the progression of columns in that algorithm is the OR of the previous set and a shifted version of the set.

The difficulties of having to do interpretation do not matter for number partitioning (or subset sum) because one index of the dynamic program table gets queried anyhow. I was trying to get the shift done via Fourier transform in $GF(2)$ (so Hadamard transform) construed as propositional formulae. But the OR is the troublemaker.

This was thought of because I wanted the ideal dynamic program table to be square (to a constant factor) at criticality of number partition [8]. When you poke at the critical phase transition of anything, as the SFI folks know, a fractal pops out with pretty good regularity, and I figured I could model it as a linear iterated function system[5]. So note that the propositional states are linear iterated function systems.

4.4 Grover's

Note that the speedup of Grover's search [7] is derived from quantum mechanics depending on the L^2 norm instead of probability, which is based upon the ordinary L^1 norm [2]. Not only the fact of the speedup, but the amount, where functional inversion is done in $O(N^{\frac{1}{2}})$.

We must investigate the question as to whether the data structure presented here could be construed as being based upon the L^∞ norm, being deterministic and the variable states having pretty intuitive meanings in that norm. This is not a certain thing, even though I spoke of it with certainty in several points here.

This has strong comedy value, because the BBBV result [2] shows that one cannot do better than Grover's speedup if one is to ignore the structure of NP-complete problems - but this one would just make Grover's speedup better. Probably you could use interference in a lot of classical algorithms but finding a classical Grover's is most exciting for this reason.

I don't know if the speedup would really get us to $O(N^{\frac{1}{\infty}}) = O(1)$ (really the oracle would dominate), but it would be worth a try. Inversion of propositions can be interpreted as finding satisfying assignments, so this is of great interest.

I admit it probably will not work, but it will be pretty good entertainment. Probably the catch is that there is a requirement for a polyspace representation of the function on the Boolean ring, so it could just end up being XORSAT. But note that the Sierpinski linear operator to transform truth table results columns to Zhegalkin form [12] is factorable with respect to Kronecker product.

In a related note, Eldar and Shor put out a retracted preprint for theoretical preliminaries for the preparation of a possible quantum attack on learning with errors[4], one of the more important putative post-quantum public cryptography methods. They had to retract it because some of the geometric arguments did not go through. It is of great interest if those geometric arguments go through and if an attack could be created in the L^∞ norm.

4.5 Implementations

This document should have come in a Github repo with implementations of the entangling phenomenon and the integral transforms, but not Simon's. There are a few implementation notes not present in this work which are present right besides the implementations. Here is a link to the Github repo.

`https://github.com/howonlee/deterministic-qubit`

4.6 Acknowledgements

Thanks to various folks who I won't name. Thanks to the YOSPOS kids for the well-justified mockery, and the CSPAM kids for the also well-justified mockery a year ago.

I had started on a portion of Shor's algorithm before I realized the consequences of what that would entail. Thanks in particular to P—-face of YOSPOS, who yelled at me enough that I thought a bit harder about figuring out a way to publish without Shor's but with entanglement and interference, so folks have some time if they're convinced.

References

[1] Dorit Aharonov, Zeph Landau, and Johann Makowsky. *The quantum FFT can be classically simulated **. 2007. URL: https://arxiv.org/pdf/quant-ph/0611156.pdf (visited on 05/17/2020).

[2] Charles H. Bennett et al. "Strengths and Weaknesses of Quantum Computing". In: *SIAM Journal on Computing* 26 (Oct. 1997), pp. 1510–1523. (Visited on 05/11/2020).

[3] Don Coppersmith. “An approximate Fourier transform useful in quantum factoring”. In: *arXiv:quant-ph/0201067* (Jan. 2002). URL: https://arxiv.org/abs/quant-ph/0201067 (visited on 05/18/2020).

[4] Lior Eldar and Peter W. Shor. “An Efficient Quantum Algorithm for a Variant of the Closest Lattice-Vector Problem”. In: *arXiv preprint* (Nov. 2016). URL: https://arxiv.org/abs/1611.06999 (visited on 05/17/2020).

[5] Kenneth J Falconer. *Fractal geometry*. Wiley, 1990.

[6] Robert B. Griffiths and Chi-Sheng Niu. “Semiclassical Fourier Transform for Quantum Computation”. In: *Physical Review Letters* 76 (Apr. 1996), pp. 3228–3231. URL: https://arxiv.org/abs/quant-ph/9511007 (visited on 05/18/2020).

[7] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In: *STOC* 1996 (Nov. 1996). URL: https://arxiv.org/abs/quant-ph/9605043.

[8] Stephan Mertens. *The Easiest Hard Problem: Number Partitioning*. URL: https://arxiv.org/pdf/cond-mat/0310317.pdf (visited on 05/18/2020).

[9] Claude E. Shannon. “Communication Theory of Secrecy Systems*”. In: *Bell System Technical Journal* 28 (Oct. 1949), pp. 656–715. (Visited on 04/26/2019).

[10] Rodney Van Meter and Kohei M. Itoh. “Fast quantum modular exponentiation”. In: *Physical Review A* 71 (May 2005). URL: https://arxiv.org/pdf/quant-ph/0408006.pdf (visited on 05/18/2020).

[11] Nadav Yoran and Anthony J. Short. “Efficient classical simulation of the approximate quantum Fourier transform”. In: *Physical Review A* 76 (Oct. 2007). URL: https://arxiv.org/pdf/quant-ph/0611241.pdf (visited on 05/17/2020).

[12] Ivan Ivanovich Zhegalkin. “On the technique of calculating propositions in symbolic logic (Sur le calcul des propositions dans la logique symbolique)”. In: *Matematicheskii Sbornik* 34 (1927).