A Proof That Measured Data and Equations of Quantum Mechanics Can Be Linked Only by Guesswork

John M. Myers and F. Hadi Madjid

Abstract. The design and operation of a quantum-mechanical device as a laboratory instrument puts models written in equations of quantum mechanics in contact with instruments. In designing a quantum-mechanical device of high precision, such as a quantum computer, a scientist faces choices of models and of instruments, and the scientist must choose which model to link to which arrangement of instruments. This contact is recordable in files of a Classical Digital Process-control Computer (CPC) used both to calculate with the equations and to manage the instruments. By noticing that equations and instruments make contact in a CPC, we rewrite equations of quantum mechanics to explicitly include functions of CPC-commands to the instruments. This sets up a proof that a scientist's choice in linking mathematical models to instruments is unresolvable without guesswork to narrow the set of models from which one is to be chosen.

The proof presents the challenge of pursuing its implications. Scientists in any investigative endeavor inherit choices from the past and frame choices for the future, choices open to guesswork and visible in CPC files. To picture the framing of choices and relations among them, we adapt colored Petri nets. Constraining the events of the nets to produce output colors defined by definite functions of input colors excludes guesswork from the firing of net events, and by contrast highlights guesses entering a net fragment as colored tokens placed by a scientist or by instruments on input conditions. The availability of these net fragments makes choice and guesswork part and parcel of physics.

Net fragments as a means of expressing guess-demanding choices are applied to portray guesswork needed in testing and calibrating a quantum computer. The sample size required to test a quantum gate in a quantum computer is shown to grow as the inverse square of the error allowed in implementing the gate.

1. Introduction

This paper stems from earlier work [**1**] and a proof presented here showing that inquiry in quantum physics continually presents a scientist with choices of equations and of instruments, unresolvable by calculation and measurement. Something else is demanded of the scientist, which may as well be called a *guess*.[1] Challenged by the proof to look at its implications, we noticed that people in investigative

2000 *Mathematics Subject Classification.* Primary 81P68, 68Q05; Secondary 81P15, 68Q85.

[1]Other words are *hypothesis, Ansatz, assumption, axiom, postulate*, and sometimes *principle.*

endeavors inherit and frame choices open to guesswork, some of which show up clearly in the computers used in the endeavors.

Section 2 introduces the Classical Process-control Computer (CPC) with its special capacity to manipulate abstractions expressed as equations without contaminating them with its own physics.[2] A scientist can use a CPC not only to calculate with equations, but also to mediate the command of laboratory instruments via digital/analog (D/A) converters and to record experimental results returned from the instruments via analog/digital (A/D) converters. By noticing that both equations and instruments make contact in a CPC, we rewrite equations of quantum mechanics to explicitly include functions of CPC-commands to the instruments. This sets up the proof that the scientist's choice in linking equations to instruments is unresolvable without guesswork to narrow the set of models. A lattice of sets of models is defined, two widely used guesses that narrow the set of models are noted, and the concept of statistical distance between probability distributions is applied to quantum-mechanical models.

Section 3 provides language for displaying and analyzing guess-demanding choices visible in CPC files. To this end Turing machines are introduced and adapted to formalize the definition of a CPC. This allows fragments of colored Petri-nets, opened to exogenous influences, to portray the programming and running of programs in a network of CPC's operated by collaborating scientists. Many of these programs incorporate guesses. This general picture of process-control computation shows programs and other guesses as colors on tokens that a scientist enters on a Petri net that acts as a game board. Mechanisms for one scientist to judge programs (and hence guesses) made by another are sketched, leading to the first of many needs for concurrently operating CPC's.

Section 4 describes some examples in which guessing, quantum mechanics and CPC structures must interact in the building of a quantum computer as a laboratory instrument specified by equations of quantum mechanics. We show the need for guesses to link equations and instruments brings with it a need to test the quantum computing instruments and to calibrate them, guided by test results and guesses. Quantum mechanics imposes a peculiar structure on this testing, related to the statistical distance between models. For the measure of precision conventionally used in quantum computing, the sample size needed for testing a quantum gate is shown to increase as the inverse square of the tolerated imprecision. While many questions are left open to future work, the example demonstrates a frame for analysis and experiment broader than any quantum model alone, a frame that includes the testing of the mathematical models by results of the use of instruments, and so distinguishes what the model says from what the instruments do, allowing provision for guesswork as an ingredient in advancing both models and instruments.

2. Quantum-mechanical models and their links to instruments

Proving the necessity of guesses demands language to describe the linking of numbers in mathematical models to numbers pertaining to laboratory instruments,

[2]This capacity stems from regenerative amplifiers and clock-gated memory registers, two inventions used to make all computer hardware insensitive to manufacturing variations, so that, like the placing of a chess piece not quite in the center of a square, deviations in performance, within limits, do not matter. Its independence from its own physics distinguishes a classical computer from a quantum computer.

starting with mathematical language to describe a scientist's choosing one arrangement of laboratory instruments rather than another. We shall describe a situation in which a scientist chooses instruments by using a CPC keyboard to type strings of characters, much as Gödel, in mathematical logic, described equations as strings of characters. The scientist at the CPC keyboard writes and executes programs to command the operation of laboratory instruments and record their results. These programs make use of quantum-mechanical equations, which the scientist also writes into the CPC.

Quantum mechanics as a mathematical language expresses different measuring instruments by different operators, and thus has built into it a recognition that phenomena to be described cannot be independent of the instruments used to study them [**2**]. Still, this dependence is emphasized more some times than others. Some modeling merely assumes that instruments can be found, without saying how, to implement various combinations of state vectors and operators. Such models appear in theories of quantum computing to relate the multiplication of unitary operators to the solving of problems of interest. To see the need for another kind of model, suppose a scientist has computer-controlled instruments (such as lasers) with the potential to implement a quantum computer, and faces the question of what commands the CPC should transmit to the instruments and when it should transmit them in order implement one or another quantum gate. Determining the commands and their timing to implement a quantum gate expressed as a unitary matrix U_j takes a model that expresses the gates as unitary transformations in terms of commands that a process-control computer can transmit to the instruments. Curiously, models of this kind have not been much stressed in physics, and it is a merit of efforts to build quantum computers to make the importance of such models apparent.

2.1. Models and instruments make contact in a CPC. Part of a scientist's control of instruments can work through the use of a process-control computer that transmits commands to the (computer-controlled) instruments and records results produced by them. We confine our analysis to this part, excluding from consideration here (but by no means denigrating) hand work beyond the reach of a process-control computer. We shall portray cases in which a scientist chooses arrangements of instruments, chooses models, and puts the two in contact, linking models to instruments, during a CPC session starting after the instruments have been set up and put under control of a CPC and ending before the scientist has to tinker with the instruments in ways unreachable by the CPC. Within the CPC, laboratory instruments and mathematical models make contact when:

(1) a model resident in a CPC file is used to derive commands for the CPC to transmit to the instruments;
(2) instrumental results collected by a CPC are used to narrow down a set of models. (We shall later see feedback as an example of this.)

Such contact does not spring from nothing, but is brought about by design and depends on choices made by a scientist, including choices of what set of models to start with, what model to choose for use by a CPC in generating commands, and what experiments to run. To picture the design and operation of contact between models and instruments, imagine eavesdropping on CPC's used in various investigations. Commands sent to the instruments by the CPC and the results

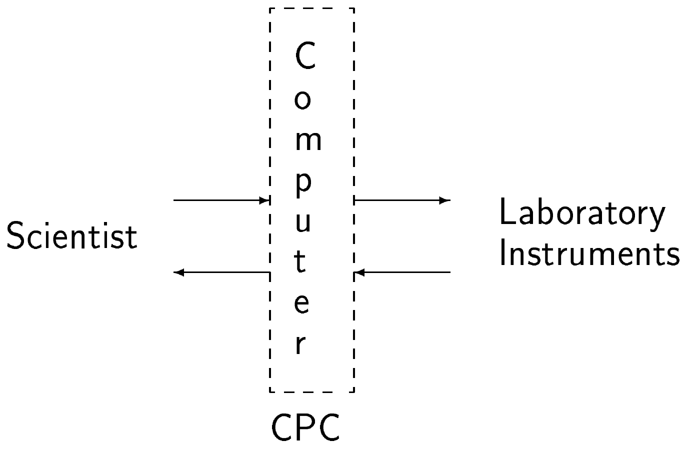

FIGURE 1. Computer mediating contact between scientist and instruments.

received from them, both numerical, are amenable to analysis, as is the scientist's writing of equations, programs, calls for program execution, etc.; we also eavesdrop on displays produced by the CPC for the scientist.

Although the CPC puts instruments in contact with equations involving quantum superposition, the CPC itself is a classical machine, free of quantum superposition, for it needs no quantum behavior within itself, neither to manipulate equations of quantum mechanics nor to manage laboratory instruments. For example, the writing of an expression $|0\rangle + |1\rangle$ for a superposition of quantum states makes use of written characters that themselves exhibit no superposition. And any command to instruments is likewise a character string, including a command to rotate a polarizing filter by 45 degrees to implement the superposed state $|0\rangle + |1\rangle$. Similarly, results of the use of instruments interpreted as demonstrating superposition arrive as bit strings, themselves devoid of superposition.

The CPC is situated between a scientist to its left and laboratory instruments to its right, as shown in figure 1. Working at the CPC, a scientist is limited in action at any moment to the resolution of the choice presented by the CPC at that moment, a choice defined by the files stored in its memory and the state of its processor, and exemplified by a menu displayed by the CPC. Our analysis of the CPC cannot reach beyond its buffers: neither to its left into the scientist, nor to its right where, invisible to eavesdropping, reside digital-to-analog (D/A) and analog-to-digital (A/D) converters and beyond them the laboratory instruments.

2.2. Quantum-mechanical models that recognize commands sent to instruments. For equations of quantum mechanics to model effects of a scientist's choices in arranging instruments, these choices must show up in the equations. To see how this can work, recall that quantum mechanics parses the functioning of instruments into state preparation, transformation, and measurement, three coordinated activities that generate *outcomes*, supposed visible in experimental results by some means unspecified. The three activities are described, respectively, by a state (as a unit vector representing a ray in a Hilbert space), a unitary operator, and a hermitian operator. The only way to make the scientist's choices in arranging instruments show up in quantum-mechanical equations is to make the state vector $|v\rangle$, the transformation operator U, or the measurement operator M, or some combination of them, depend on how these choices are resolved.

A simple and yet, so far as we know, original way to analyze a scientist's choice of arrangements of instruments is to suppose that during a CPC-mediated session the instruments are controlled by CPC-transmitted commands from a set B of possible commands, where $B \subset \mathcal{B}$ and $\mathcal{B}$ is the set of all finite binary strings. We formulate a core set of quantum mechanical models that express the probability of an outcome of instruments in response to a command $b \in B$ sent to the instruments by the CPC, as follows. Let $\mathcal{V}_B$, $\mathcal{U}_B$, and $\mathcal{M}_B$ be the sets of all functions $|v\rangle$, U, and M, respectively, with

$$\begin{aligned} |v\rangle : B &\to \mathcal{H}, \\ U : B &\to \{\text{unitary operators on } \mathcal{H}\}, \text{ and} \\ M : B &\to \{\text{hermitian operators on } \mathcal{H}\}. \end{aligned}$$

The core models exhibit discrete spectra for all $M \in \mathcal{M}$:

PROPERTY 1.

$$(\forall b \in B) M(b) = \sum_j m_j(b) M_j(b), \tag{2.1}$$

where $m_j : B \to \mathcal{R}$ (with $\mathcal{R}$ denoting the real numbers) is the j-th eigenvalue of M, and M_j is the projection onto the j-th eigenspace (so $M_j M_k = \delta_{j,k} M_j$).

Let $\Pr(j|b)$ denote the probability of obtaining the j-th outcome, given transmission by the CPC of a command b. Although not commonly seen in texts, this probability of an outcome given a command is the hinge pin for focusing on quantum mechanical modeling of uses of instruments. Quantum mechanics constrains the models to satisfy:

PROPERTY 2.

$$\Pr(j|b) = \langle v(b)|U^\dagger(b) M_j(b) U(b)|v(b)\rangle, \tag{2.2}$$

where the $\dagger$ denotes the hermitian adjoint.

(Within this modeling scheme, the Schrödinger equation relates a model at a later time to a model at an earlier time by a certain transformation operator U, dependent on the situation.)

Any choice from the sets $\mathcal{B}$, $\mathcal{V}_B$, $\mathcal{U}_B$, and $\mathcal{M}_B$ produces some quantum-mechanical model $(|v\rangle, U, M)_B$. Two models $(|v\rangle, U, M)_B$ and $(|v'\rangle, U', M')_B$ generate the same probabilities $\Pr(j|b)$ if they are unitarily equivalent, meaning there exists a $Q : B \to \{\text{unitary operators on } \mathcal{H}\}$ such that $(\forall b \in B)|v'(b)\rangle = Q(b)|v(b)\rangle$, $U'(b) = Q(b)U(b)Q^\dagger(b)$ and $M'(b) = Q(b)M(b)Q^\dagger(b)$. For this reason, any model $(|v\rangle, U, M)_B$ can be reduced to $(|v'\rangle, \mathbf{1}, M)_B$, where $|v'\rangle = U|v\rangle$ and $M' = M$ or, alternatively to $(|v\rangle, \mathbf{1}, M')_B$ where $M' = U^\dagger M U$.

More models are available in more general formulations. When we show that guesswork is necessary even to resolve choices among models of the core set, it follow that guesswork is necessary also to resolve the choices of among a larger set of models involving positive-operator-valued measures, superoperators, etc.

2.3. From results to quantum-mechanical outcomes. Before stating and proving the proposition that calculations and measurements cannot by themselves link models to outcomes obtained from instruments, we call to the reader's attention that outcomes themselves, in the sense of quantum mechanics, are produced by instruments only with the help of interpretive guesswork.

CLAIM 1. *To speak of actual instruments in the language of quantum mechanics one needs to associate results of the use of the instruments, recorded in a CPC, with* outcomes *in the sense of quantum mechanics or with averages of outcomes.*

Experimental results of the use of instruments become quantum-mechanical outcomes only by a scientist's act of interpreting the results as outcomes. The interpretation involves judgment and guesswork, not only to sidestep imperfections in the instruments, but as a matter of principle, even for the limiting case of instruments supposed free of imperfections. For example, light detectors used in experiments described by models of quantum optics generate experimental results; typically, each of L detectors reports to the CPC at each of a succession of K time intervals a detection result, consisting of 0 (for no detection) or 1 (for detection), so a record contains LK bits. Depending on judgments made about correlations from time interval to time interval and detector to detector, these LK bits may constitute LK quantum outcomes, or one quantum outcome, or some number in between. The number of outcomes in LK bits is determined neither by the experimental results (which in this case are just these bits) nor by general principles of quantum mechanics; yet the parsing of results into outcomes must occur, at least provisionally, before any comparison between equations and measured outcomes can be made. Henceforth, when we speak of outcomes, we presuppose that this piece of guesswork has been accomplished and a decision made to define the parsing of results into outcomes.

2.4. Calculation and measurement by themselves cannot link quantum models to recorded outcomes. Could it be that the general properties 1 and 2 suffice to determine a model (up to unitary equivalence) if only one collects enough measured results interpreted as outcomes? The answer is: "no; unless some special properties restrict the model more tightly than the form established by properties 1 and 2 alone, one can always find many unitarily inequivalent models $(|v\rangle, U, M)_B$, all of which produce probabilities that match perfectly the relative frequencies of outcomes."

To prove this we define some things to pose the issue more sharply. Let B denote the set of commands used to generate some set of outcomes interpreted from measured results.[3] For any $b \in B$, let $N(b)$ be the number of times that an outcome has been entered in the record for a run of the experiment for command b, and let $J(b)$ be the number of distinct outcomes for command b. For $j = 1, \ldots, J(b)$, let $\lambda_j(b)$ be the j-th distinct outcome obtained for command b, and let $n(j, b)$ be the number of times this j-th distinct outcome λ_j is recorded in response to command b. For all $j > J(b)$ let $\mu_j(b)$ be arbitrary real numbers, and for all $j \geq 1$ let $\phi(j, b)$ be arbitrary real numbers.

PROPOSITION 2.1. *Given any set of recorded outcomes associated with any set B of commands, the set of models satisfying properties 1 and 2 contains many*

[3]Practically speaking, B must be a finite set, but the proof holds also for B denumerably infinite.

unitarily inequivalent models $(|v\rangle, U, M)_B$*, each of which has a perfect fit with the set of outcomes, in the sense that*

$$(\forall b)(\forall 1 \le j \le J(b))\,\Pr(j|b) = n(j,b)/N(b). \tag{2.3}$$

Proof. It is instructive to start with the special case in which for some $b \in B$, there exist two or more distinct values of j for which $n(j) > 0$. For this case let the set $\{|j\rangle\}$ be an orthonormal basis of a separable Hilbert space. Define a subset S of models satisfying properties 1 and 2 as all models of the form $(|v\rangle, U, M)_B$, where

$$|v(b)\rangle \overset{\text{def}}{=} \sum_{j=1}^{J(b)} [n(j,b)/N(b)]^{1/2} \exp(i\phi(j,b)|j\rangle), \tag{2.4}$$

$$U(b) \overset{\text{def}}{=} \mathbf{1}, \tag{2.5}$$

$$M(b) \overset{\text{def}}{=} \sum_{j=1}^{J(b)} \lambda_j(b)|j\rangle\langle j| + \sum_{j=J(b)+1}^{\infty} \mu_j(b)|j\rangle\langle j|, \tag{2.6}$$

with μ_j and ϕ arbitrary real-valued functions. By invoking property 2, one checks that any such model has the claimed perfect fit; yet the set contains many unitarily inequivalent models, which predict conflicting statistics for some possible quantum measurement.[4] This proves the special case.

For the general case, modify the definitions above to

$$|v(b)\rangle \overset{\text{def}}{=} \sum_{j=1}^{J(b)} [n(j,b)/N(b)]^{1/2} |w_j\rangle, \tag{2.7}$$

$$U(b) \overset{\text{def}}{=} \mathbf{1}, \tag{2.8}$$

$$M(b) \overset{\text{def}}{=} \sum_{j=1}^{J(b)} \lambda_j(b)P_j + \sum_{j=J(b)+1}^{\infty} \mu_j(b)P_j, \tag{2.9}$$

where $P_jP_k = \delta_{j,k}P_j$, for all j the projection P_j has dimension greater than 1, and $|w_j\rangle$ ranges over all unit vectors of the eigenspace defined by $P_j|w_j\rangle = |w_j\rangle$. In particular, for any j, $\dim(P_j)$ can be as large as one pleases. Then even if there is only one outcome that is ever recorded, there are still many unitarily inequivalent models that perfectly fit the data. □

Proposition 2.1 implies that the density matrix, often supposed to be determined from measured data [**3**], is undetermined without assuming special properties shortly to be discussed; this follows by expressing the density matrix as $|v\rangle\langle v|$ and noticing that the phases of the off-diagonal elements are undetermined. We leave to the future the demonstration of additional ambiguity in the link between any set of recorded outcomes and models expressed in the mathematical language of quantum mechanics.

[4]This happens *e.g.* for primed and unprimed models if for any b, $n(1,b) \neq 0 \neq n(2,b)$ and $\phi(1,b) - \phi(2,b) \neq \phi'(1,b) - \phi'(2,b)$.

2.5. Statistically significant differences between models. In practice, a scientist has little interest in a model chosen so that its probabilities exactly fit measured relative frequencies. Rather, the scientist wants a simpler model with some appealing structure that comes reasonably close to fitting. Quantum mechanics encourages this predilection, because on account of statistical variation in the sample mean, functions that perfectly fit outcomes on hand at one time are not apt to fit perfectly outcomes acquired subsequently. We show here that accepting statistics no way takes away from the proof that measurements and equations by themselves cannot link models to instruments.

One needs a criterion for the statistical significance of a difference between two quantum-mechanical models (or between a model and measured relative frequencies). Here we limit our attention to models α and β which have a set B of commands in common and for which the spectra of M_α and M_β are the same. For a single command b, the question is whether the difference between the probability distributions $\Pr_\alpha(\cdot|b)$ and $\Pr_\beta(\cdot|b)$ is bigger than typical fluctuations expected in $N(B)$ trials. An answer is that two distributions are indistinguishable statistically in $N(b)$ trials unless

$$N(b)^{1/2} d(\Pr_\alpha(\cdot|b), \Pr_\beta(\cdot|b)) > 1, \tag{2.10}$$

where d is the statistical distance defined by Wooters in Eq. (10) of [**4**]. Furthermore, Wooters's Eq. (12) shows for two models α and β that differ only in the function $|v\rangle$,

$$d(\Pr_\alpha(\cdot|b), \Pr_\beta(\cdot|b)) \leq \cos^{-1} |\langle v_\alpha(b)|v_\beta(b)\rangle|. \tag{2.11}$$

To judge the significance of the difference between two models with respect to a set B of commands common to them, a scientist who chooses some weighting of different commands can define a weighted average of $d(\Pr_\alpha(\cdot|b), \Pr_\beta(\cdot|b))$ over all $b \in B$. The same holds if model β is replaced by relative frequencies of outcomes interpreted from measured results.

It is noteworthy that the set of models statistically indistinguishable from a given model can be much larger than would be the case if the "$\leq$" of (2.11) were an equality, as follows.

PROPOSITION 2.2. *For any set of outcomes, two models α and β of the form $(|v\rangle, \mathbf{1}, M)_B$ can perfectly fit the relative frequencies of the outcomes (Proposition 2.1) and yet be mutually orthogonal in the sense that $\langle v_\alpha|v_\beta\rangle = 0$*

PROOF. For any set of measured outcomes, there exists a perfectly fitting model α of the form in the proof of Proposition 2.1 for the general case, for which $(\forall j, b) dim(|w_j(b)\rangle) > 1)$, and a corresponding perfectly fitting model β such that $(\forall j, b)\langle w_{\alpha,j}(b)|w_{\beta,j}(b)\rangle = 0$. For these two models, $\langle v_(b)\alpha(b)|v_\beta(b)\rangle = \sum_j [n(j,b)/N(b)]\langle w_{\alpha,j}(b)|w_{\beta,j}(b)\rangle = 0$. □

Wooters extended the definition of statistical difference to unit vectors. While for any two unit vectors, there exist measurement operators that maximize the statistical distance between them, for any such operator there exist other vectors, mutually orthogonal, that have zero statistical distance relative to this operator. For this reason, among others, statistics still leaves the scientist needing something beyond calculation and measurement to determine a model, for the set of models closer than ϵ in weighted statistical distance to certain measured results certainly includes all the models that exactly fit the data and, without special restrictions dependent on guesses, this set includes models that are mutually orthogonal. Models close to given measured data are not necessarily close to each other in the predictions they make.

2.6. Lattices of models. Properties 1 and 2 set up a big set of models $(|v\rangle, U, M)_B$, $B \subseteq \mathcal{B}$, $|v\rangle \in \mathcal{V}, U \in \mathcal{U}, M \in \mathcal{M}$. Subsets of models of this set are a lattice under set intersection and union. Each command set B establishes a smaller lattice of sets of models, and these lattices will play a part in the testing and calibrating of quantum computers, discussed in section 4, where a scientist encountering problems with a model chooses a set of possible alternatives, and then tries to narrow it. Often this narrowing is seen as choosing values of parameters within a form of model in order to obtain a best fit, say with a criterion of minimizing statistical distance between frequencies of outcomes interpreted from measured results and probabilities calculated from the model. One is free to think of the estimating of parameters in the language of a lattice of models as the using of measured results to select a model from a set of models.

From Proposition 2.1 that showed that the whole set of models defined by properties 1 and 2 is too big to permit measured results to select a model, we have:

PROPOSITION 2.3. *For measured data to uniquely decide to within unitary equivalence which quantum-mechanical model of a set of models best fits experimental results interpreted as outcomes by a criterion of least statistical distance (or any other plausible criterion), the set of models must first be sufficiently narrowed, and this narrowing is underivable from the results and the basic properties 1 and 2 of quantum mechanics.*

Something beyond measured results and calculations from equations is required to narrow a set of models so that measured results can select a model that is "best" by some criterion. Such an act of choosing undefined by calculation and results of observation is what we have called a *guess.*

2.7. Hidden guesswork in conventional quantum mechanical models. The proof casts in a clear light maneuvers conventionally made to narrow down the set of models. Sometimes a community of physicists is in mutual agreement about guesses deemed appropriate, and this agreement obscures from notice the fact that a guess is invoked. As an example of a widely invoked guess, most modeling in quantum physics supposes that the scientist can vary b so as to vary $U(b)$ while holding $v(b)$ and $M(b)$ constant. Indeed, most models used in quantum physics are restricted to the subset of models having the special

PROPERTY 3. *The command b is the concatenation of separate commands for the three types of operations, so that*

$$b = b_v \parallel b_U \parallel b_M, \tag{2.12}$$

where here the $\parallel$ denotes concatenation of commands.

According to these models, one can vary any one of the three while holding the other two fixed. This specializes (2.1) to the more restrictive form:

$$\Pr(j|b) = \langle v(b_v)|U^\dagger(b_U)M_j(b_M)U(b_U)|v(b_v)\rangle. \tag{2.13}$$

An additional constraining guess characterizes models widely used in the analysis of quantum computers, a guess prompted by the desire to generate a unitary transformation as a product of other unitary transformations that serve as "elementary quantum gates." For example, one may want to generate the unitary transformation $U(b_{U,1})U(b_{U,2})$. To generate it one causes the CPC to transmit some b_U. For quantum computing to have an advantage over classical computing, the determination of this b_U in terms of $b_{U,1}$ and $b_{U,2}$ must be of polynomial complexity [**5**]. It is usually assumed that b_U is the simplest possible function of $b_{U,1}$ and $b_{U,2}$, as follows.

Let $B_U \subset B$ be a set of instrument-controlling commands, thought of as strings that can be concatenated. Suppose the function U has the form $U(b_1 \parallel b_2) = U(b_2)U(b_1)$ for all $b_1 \parallel b_2 \in B_U$ (note reversal of order). Then we say the function U respects concatenation.

PROPERTY 4. *Quantum computation employs a subset of models in which U respects concatenation.*

REMARK 2.1. We present properties 1 through 4 not as properties of laboratory instruments, but as properties that a scientist can choose to demand of models. Whether the instruments act that way is another question. There are reasons, relaxation and other forms of decoherence among them, to expect limits to the precision with which instruments can behave in accord with properties 3 and 4. All four properties are used often enough to be conventions, in the sense that a convention is a guess endorsed by a community.

3. Petri nets to show choices open to guesswork

In orchestrating contact between mathematical models and laboratory instruments, scientists set up chains of cause and effect, expressed in computer programs with their "if-then" structure, not as static propositions but as designs for action. Such designs are implemented in experiments; an example is a feedback loop that adjusts the orientation of a filter according to a rule that tells what adjustment to make in immediate response to a result recorded by a light detector. On a more relaxed time scale, physicists make other connections by analyzing outcomes of one generation of experiment, using the equations of a model, to set up design instruments for a next generation. As remarked above, contact between equations and instruments depends on choices made by scientists, including choices of what set of models to start with, what model to choose for use by a CPC in generating commands, and what experiments to run. If these choices could be resolved by some combination of calculation and measurement, one could argue that they are irrelevant to physics. But the propositions of the preceding section show this is not the case, so the design and operation of contact between equations and instruments, with its ineradicable dependence on guesswork, cries out for attention as part and parcel of physics.

Although widespread in practice, the design of contact between equations and instruments is in its infancy as a topic for theoretical attention. A beginning can be seen in Benioff's analysis of sequences of measurements (described quantum mechanically) in which subsequent measurements are functions of outcomes of preceding measurements [**6**]. Called decision procedures, these involve classical feedback control equations to control instruments described quantum mechanically, in some cases with proved advantages [**7**]. These efforts dealt with measurements occurring at a single location. Designs that put equations and instruments in contact over a network of cooperating investigators are wide open for future attention.

Logic in experiments, in feedback loops at many time scales, is logic in action. This is the logic of models that relate instrument commands to quantum vectors and operators. Here we adapt Petri nets to provide mathematical language by which to express and analyze designs for contact between equations and instruments, designs that include sequencing of effects, decision rules, and interactions among sequences of effects that scientists implement in their instruments. The nets will highlight choices resolvable only by resort to guesswork; they serve as a language with which one can express formally how guessing works in physics, case by case, within CPC-mediated investigations.

3.1. Requirements of CPC's. In order to adapt Petri nets to showing guess-demanding choices visible in CPC's, we start by clarifying how a CPC differs from a Turing machine, on the way to adapting the Turing machine to process control and to use in a network of collaborating scientists. This lays the groundwork for introducing Petri nets.

3.1.1. *Timing in the execution of commands.* The first thing that makes process-control computing special is timing. In the context of quantum-mechanical models, each unitary transformation maps states possible in one situation to states possible in another situation; for quantum computing this means mapping states possible at an earlier time to states possible at a later time. Thus a unitary transformation is implemented not all at once, but over a time duration. In practice, that duration depends on how the instruments implement the transformation. A written command b_U acts as a musical score. Like sight reading at a piano, executing a program containing the command b_U requires converting the character string b_U—the score—into precisely timed actions—the music. The piano keys, in this analogy, include the output buffers that control the amplitude, phase, frequency, and polarization of lasers of an ion-trap quantum computer or of radio-frequency transmitters for a nuclear-magnetic-resonance (NMR) quantum computer.

For this reason executing a command b_U requires parsing it into pieces (signals) and implementing each signal at a time, the specification of which is contained in the string b_U. Either the CPC that executes a program in which b_U is written parses the command into signals and transmits each signal at its appointed time, or the instruments receiving the command b_U, unparsed, contain programmable counters operating in conjunction with a clock that do this timed parsing. Such programmable counters themselves constitute a special-purpose CPC. So either the scientist's CPC must execute commands by issuing an appropriately timed sequence of signals, or some other CPC attached to the instruments must do this. Either way, the capacity to execute programmed motion in step with a clock is a requirement for a CPC, distinct from and in addition to requirements to act as a Turing machine.

3.1.2. *Firewalls in a network of computers.* Just as axioms set up branches of mathematics, guesses set up rules for the conduct of experiments and the interpretation of their results, rules often embedded in CPC's. Collaborating scientists accept guesses from each other, at least provisionally, use these in experimenting and modeling; they evaluate some of them, sometimes refining or replacing them. This poses a problem for CPC-mediated inquiry, where guesses engender computer programs, for a scientist's guess can reprogram a CPC, often for better but sometimes, by malice or accident, for worse. Scientists in a collaboration need to test each other's programs and to limit the influence of any program, making the scope of influence of a CPC program a matter for negotiation among the collaborators.

An easy but narrow case is that of a computer running Gödel's test for validity of a claimed derivation [**8**]. To think about such testing, one models the computer by a Turing machine designed to start from a tape on which the claimed derivation is written and to halt leaving a "yes" or "no" on the tape, according to whether the claim is or is not valid. Such a Turing machine can be emulated by a universal Turing machine executing a testing program to check a passive (non-executed) file containing the claimed derivation.

Not just derivations, but also programs need to be tested with respect to what they do when they are executed. But what is to keep an executing program under test from infecting the program that tests it? Hardware walls of some kind are needed. By limiting our analysis to exclude remote login and insisting on computers that distinguish physically one interface from another, we can see a basic structure for testing programs and for limiting the reach of guesses of any one scientist in a network of CPC's, based on operating two or more CPC's concurrently with controlled interfaces between them, so the testing program and the program under test execute on separate CPC's, with an interface controlled by the testing CPC. By virtue of concurrent operation of CPC's with controlled interfaces, guesses made by collaborators can set up programs that frame choices open to guessing by any one scientist, and that test the performance of the scientist's programs within that frame of choice, allowing freedom to a scientist to program one part of the investigation while insulating other parts. Hardware walls that limit the reach of one person's guesses at any moment are one many motivations for stressing a network of concurrently operation CPC's.

3.2. Turing machines and Petri nets. Here we provide language for displaying and analyzing guess-demanding choices visible in files of CPC's used by collaborating scientists who on occasion reprogram those choices. As a model of a CPC, we assume that each CPC of a network is a Turing Machine adapted for Process-control (TMP), to be defined. Making sense of networks of TMP's handling equations and controlling instruments calls for a descriptive capacity that allows for various viewpoints at various levels of detail. We introduce a specialized use of fragments of colored Petri-nets, opened to exogenous influences, to portray the programming and running of programs in a network of TMP's operated by collaborating scientists.[5]

Different viewpoints and levels of detail are accommodated by morphisms in the category of nets. Isomorphisms between Petri nets trade net detail for color detail [**9**]. These will be combined with coarsening maps that suppress detail, for example by mapping colored tokens to black tokens. We will show how the programming of a universal TMP (UTMP) portrayable as a single Petri net can produce any number of patterns of use of instruments and equations, portrayable by a host of different Petri nets. This general picture of process-control computation will show programs and other guesses as colors on tokens that a scientist enters on a game board defined by a fragment of a Petri net, and equations of quantum mechanics

[5]Our use of Petri nets is impressionistic and a more technical presentation will doubtless be rewarded by exposing issues here overlooked.

written as guesses by a scientist will be seen as colors on tokens that take part in directing and interpreting the use of laboratory instruments.

3.2.1. *Writing* vs. *executing a program.* Computers rest on the writing of motionless characters on a page to describe something moving, a puzzle solved in music by writing notes on staves, to be read in step with a swinging pendulum that chops time into moments, so that written notes that portray a still picture for each moment direct the motion of the playing of a musical instruments [**10**]. The logical machinery of a computer moves in response to triggering signals, "tick" and "tock", synchronized to distinct phases of the swinging of a pendulum. Computer designers employ truth tables, each of which specifies the response of a clocked circuit at a tock to a stimulus present as an input at a preceding tick. A row of a truth-table can be drawn as a transition in a Finite State Machine (FSM). By coupling an FSM to a memory of unlimited capacity, one arrives at the theoretical concept of a Turing machine, various special cases of which perform various special tasks [**11, 12, 13**]. And here is the crux of programming: because a state machine is describable by still writing—a table—a Turing machine can be designed to be universal. By coding into its memory the table that describes any given special Turing machine, one causes the universal Turing machine to emulate the given special Turing machine. So, apart from speed and memory requirements, the single universal Turing machine can be put to doing any of the things that any of the special Turing machines can do, making it potentially convenient, once adapted to process control, to designing and implementing contact between equations and instruments. (But demands for quick response require in some cases devices streamlined to a special task better modeled by a special Turing machine than by a universal one.) The next tasks are to adapt the Turing machines, special and universal, to process control, and after that to express them formally by use of colored Petri net fragments.

3.2.2. *Turing machine for process control (TMP).* To adapt a Turing machine as a model of a process-control computer, we leave the coupling of the FSM to the memory unchanged but add input and output buffers to the FSM. As for the FSM, at whatever level of detail of description one chooses, the control structure of a program (with its "if-then" statements) can be viewed as an FSM consisting of (classical) states drawn as circles, connected by directed arcs, with each arc labeled by an input I that selects it and by an output O [**12**]; a fragment of such a picture is shown in figure 2(a). An FSM serves as a game board on which a single token can be placed to mark the "current state." Heading toward the hooking together of FSM's to make a Petri net, we suppose that each arc in the FSM is punctuated by a tick event and a tock event, drawn as small boxes, enlarging the FSM into a special case of a condition-event Petri net fragment, as shown in figure 2(b). Once colors are introduced, states shown as dashed circles pointing into an event of the FSM from outside will become the means to express the entrance of guesses. These states are assumed to receive tokens put into them by scientists and instruments undescribed by events of the net. Similarly, dashed states pointed to by arcs from an event are assumed to have tokens taken from them by agents undescribed by events of the net. Figure 2(c) streamlines the picture to the form we shall use, in which more or less vertical arcs are understood to point downward, the dashed states are left undrawn, as are all states with one input and one output event. To emphasize the input and output arcs with their extra tokens, we often call this an FSM fragment to distinguish it from the FSM form of figure 2(a).

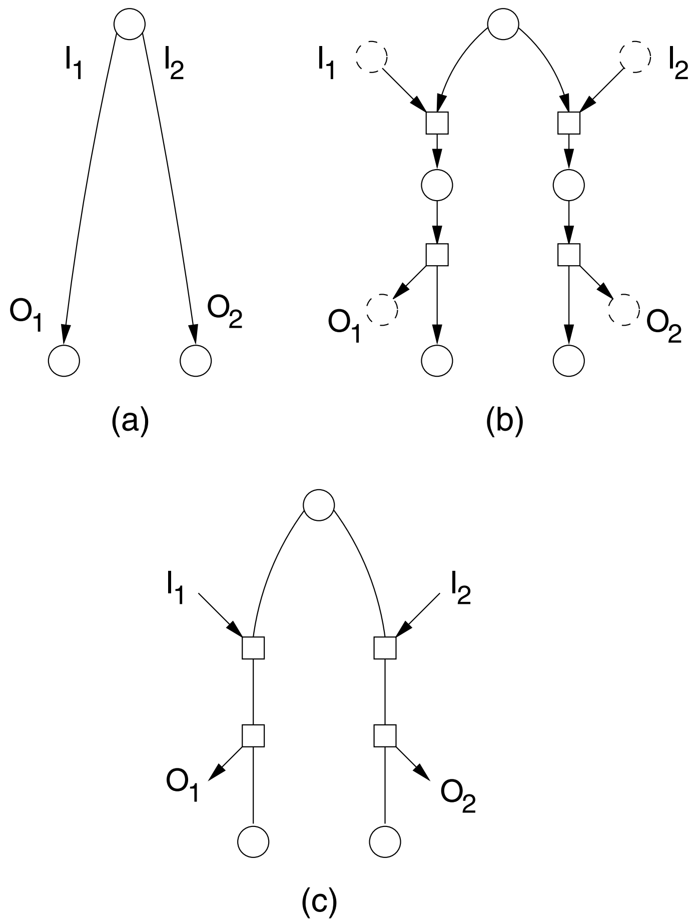

FIGURE 2. Fragment of FSM.

To define a Turing machine for Process-control (TMP), we adapt the FSM of a Turing machine to have for each of its states a cartesian product of states of a set of clocked internal registers and, in addition, input buffers and output buffers, which allow input/output transactions with a scientist, with laboratory instruments, and with other TMP's.

3.2.3. *Colored tokens.* By replacing the black tokens of an FSM fragment by a colored tokens and adjoining to each event a function that defines colors on output tokens in terms of colors on input tokens, any FSM fragment can be mapped one-to-one to the drastic form of figure 3, in which color changes substitute for most of the moves of black tokens on a bigger net. A "fork in the road" for black tokens, turns into a choice between red and green, so to speak, so the descriptive burden is taken up by the functions f_{tick} and f_{tock}; f_{tick} defines the color of a token placed on an internal state depending on a list of colors, one for each input, while f_{tock} defines a list of output colors depending on the color of the token on the internal state. The vertical arc is to be read as directed downward, and the big circles at the top and bottom of a path signify that the path is wrapped around a cylinder,

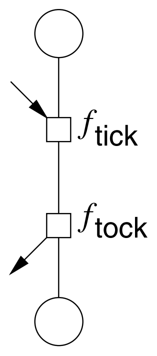

FIGURE 3. FSM with detail pushed to coloring.

so the top is a continuation of the bottom, *i.e.* a loop. An FSM fragment in which the token carries a color will be called a colored FSM.

3.2.4. *Other mappings.* Less drastic mappings are also possible. Any two states of a single FSM can be merged without breaking any arcs by augmenting the color rules in the events that feed them and the events fed by them. If a set of states connected to one another by events is mapped into a single state, the single state then connects to an event that loops back to it; this results in a place-transition Petri net, but not a condition-event net. We restrict the mappings dealt with here to ones that avoid pasting tick and tock events together, thereby avoiding self loops. Two events of an FSM that link the same pair of states can be merged by distinguishing external inputs and outputs by color instead of by place.

The mappings discussed so far are net isomorphisms: they map markings of one net bijectively to markings of the other and preserve the one-step reachability of one marking from another (by the firing of an event). Inverses of these bijections take more richly to less richly colored nets. Going in this direction depends on each state of a colored FSM having a set of possible colors associated with it [**9**]; then any colored transition corresponds one-to-one to a set of transitions obtained by partitioning sets of colors of input states, as illustrated in figure 4 for a two-in, two-out transition with color sets A, B, C, and D, each partitioned into "+" and "−" subsets. For this to make sense, it must be that an event which has tokens in all its inputs cannot fire unless the colors of the tokens comprise an element of the domain of its color function; we assume this firing rule.

One gets a coarser description by use of a surjective map that is not an isomorphism by dropping the color distinction and dropping the color functions from the transitions; this coarsening, however, preserves a one-to-one correspondence between the number of firings in one net and the number in another. All these maps are continuous in the net topology [**14**], and, as emphasized by Petri [**15**], nets form a category in which the morphisms are continuous maps, an idea that extends to nets with colored tokens [**9**].

3.2.5. *Disciplined coarsening of time.* Some other kinds of continuous coarsening maps bundle up multiple event firings into a single firing; as when one describes *e.g.* "running a program" as a single event. This brings us to the first of several areas open to future work, for, more than other computing, process control benefits from well defined timing, and in particular from machine and software design that

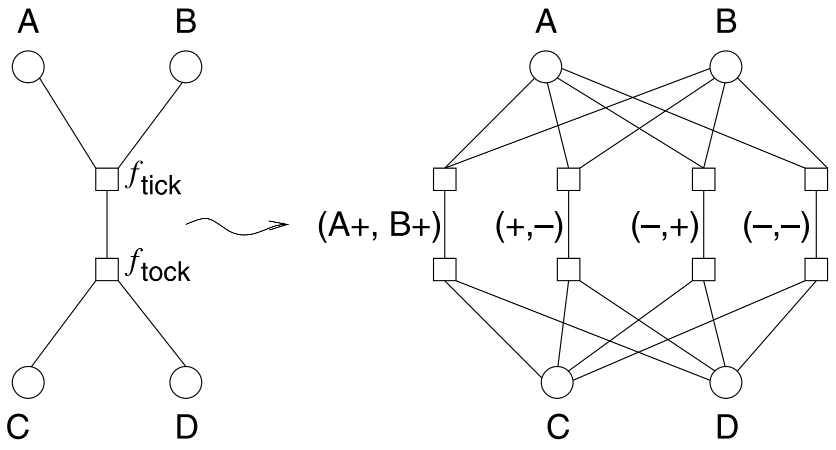

FIGURE 4. From color detail to net detail.

allows systematic, well controlled mappings that take a certain number of firings in an FSM to a single firing, so that one can think at a coarser level while still maintaining discipline in timing.

A striking example of the need to design programs that run in the same time for all inputs from some set I occurs in quantum computing. For example, suppose that $\mathbf{U}$ is the universal unitary operator defined by Deutsch to operate on basis states of the form $|s;\mathbf{n};\mathbf{m}\rangle$ where s is the location of the scanned square, $\mathbf{n}$ is the state of the FSM-processor ($\mathbf{n}=0$ is the starting state and $\mathbf{n}=1$ the halt state) and $\mathbf{m}$ is the tape [**16**]. For this to work in a computation that takes advantage of quantum superposition, one needs $\exists r[(\forall x \in I)\mathbf{U}^r|0;0;x,0\rangle = |0;1;x,f(x)\rangle]$; however, this is by no means implied for a program π_f for which (as is usual in borrowed classical programs) one can assure only $(\forall x \in I)(\exists r(x))[\mathbf{U}^{r(x)}|0;0;x,0\rangle = |0;1;x,f(x)\rangle]$ [**17**]. An interesting topic for future study is the complexity of converting various classes of programs with variable running time to programs running in a time independent of the input for some set of inputs.

3.2.6. *Cartoon of UTMP.* Ignoring the laboratory instruments for the moment, by connecting input- and output-signals from a suitable FSM to a scientist and coupling the FSM to an unlimited memory, one gets a Universal Turing Machine (UTM) that provides for continual communication with a scientist, as shown in figure 5(a), in which boxes connected by a horizontal line are read as a single event. We cartoon the UTM in the condensed form of figure 5(b). By adding input- and output-signals from the FSM to laboratory instruments and to other FSM's, one gets a Universal Turing Machine adapted for Process control (UTMP), as shown in figure 5(c); again almost all of the burden of description is in the color functions, here called T_1 and T_2 (for Turing) that define a finite state machine that operates a UTMP. We assume that at some level of description, the ticks and tocks of the UTMP slice time into moments not only for the UTMP but also for the scientist at a keyboard and the instruments on the laboratory bench; we assume that input tokens from the scientist and from the instruments arrive at the UTMP synchronized with the UTMP pendulum. If the scientist enters nothing at a given clock tick, then the token taken by the UTMP from the input buffer for the scientist

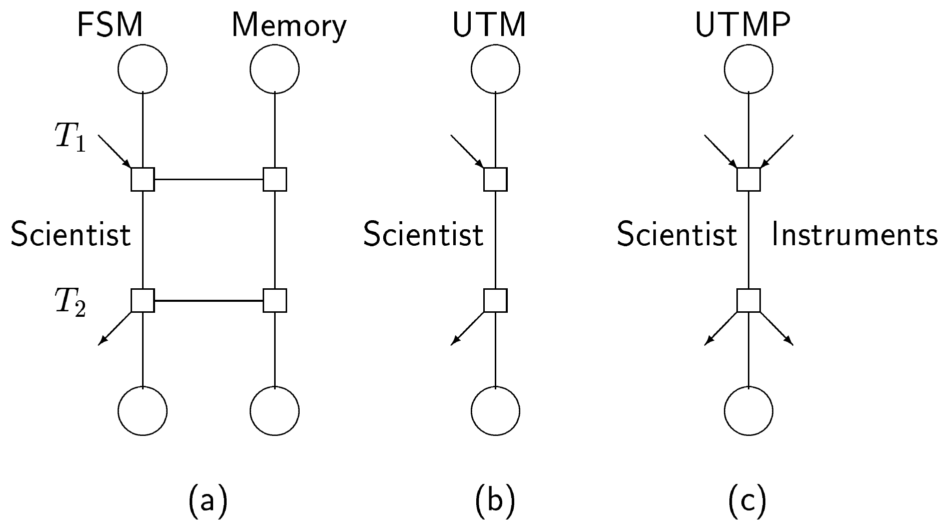

FIGURE 5. From FSM to UTMP.

carries the color "empty," and if the instruments enter nothing, the token from the input buffer for the instruments carries the color "empty"; similarly the UTMP marks output tokens with the color "empty" if it writes nothing else on them.

3.2.7. *A scientist controls a UTMP.* To see the structure imposed on physics by the UTMP, one must think as if the UTMP were delivered to a scientist in a bare condition: no installed software,[6] the FSM in a starting state, and the memory all blank. We assume that the function T_1 operating on empty input tokens, the starting state of the FSM, and a blank memory produces empty output tokens and makes no change in the FSM state or the memory or the memory location scanned. Finally, we invoke the universality of a UTM to assume that the functions T_1 and T_2 are fixed (by a manufacturer, so to speak) independent of whatever laboratory instruments need to be considered and independent of all action by the scientist. These assumptions imply

PROPOSITION 3.1. *Whatever a UTMP does besides staying in its starting state and taking in and putting out empty tokens is in response to input tokens.*

We invoke this proposition to view the scientist as precluded from defending questionable management of equations or instruments by saying "the computer did it." If a CPC does something, it executes a program; we view the scientist as responsible for any program entered (as a colored token) into the UTMP and for running the program on any particular occasion.[7]

3.2.8. *Reprogramming always an option.* We assume the UTMP is isomorphic to the net shown in figure 6, so that the scientist has a recurring choice of letting the UTMP run as programmed or of interrupting it to reprogram it. By programming a UTMP, a scientist can simulate an arbitrary special Turing machine. At will, the scientist can interrupt a program in execution to change to a program that

[6]The scientist can borrow software and install it, but is responsible for it.

[7]This rules out taking for granted the operating system, instrument-managing programs, a simulator, and whatever other programs come pre-installed in a commercially available CPC.

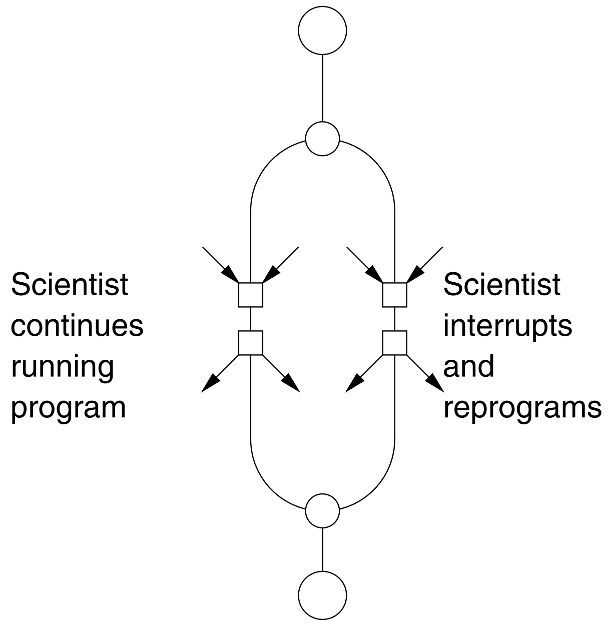
Scientist continues running program
Scientist interrupts and reprograms

FIGURE 6. Alternative modes controlled by scientist.

simulates a different special Turing machine, corresponding to a different FSM and a different net. One can glimpse this in figure 4, where it is apparent that if the colors are limited to the sets A^+ and B^+, then six of the eight events are precluded from firing, and the net is in effect reduced to the fragment defined by the selected colors. In this way the part of the net that actually fires, corresponding to the event "Use existing program" of figure 6, is variable in how it acts and in the net by which one portrays it in more detail, according to the scientist's actions in providing and running programs.

3.2.9. *Plug and play.* To see how UTMP's can be connected (as well as the detail of how the FSM of a TM or TMP is connected to the memory), we introduce a signal that is phased just opposite to an FSM: the signal takes an input at a tock event and issues an output at a tick event. Then FSM A can send a signal (which can convey a message as a token color) to B (which can be either another FSM or a memory), as shown in figure 7, provided the signal path is short enough compared to the clock rate of the machines. This use of a signal synchronizes A with B. For two-way communication, one adds a signal going the other way. If communication over a distance long compared to the clock period is called for, then a chain of communication over intermediating UTMP's, is necessary, with the result that more firings of an event of A are required before a consequence of one firing can propagate to B and return as a property of a color on a token at a later firing of the A-event. The use of colored tokens sets up an area for future investigation of replacing the awkward definition of *synchronic distance* [**18**] with a measure of synchronization that counts firings in circuits of color effects, without having to add artificial elements to a net.

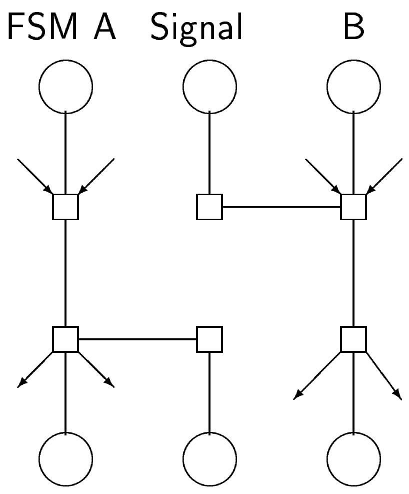

FIGURE 7. Signal from A to B.

3.3. Net fragments formalized. Portraying logic operating in CPC's calls for fragments of Petri nets, not complete nets, to allow for guesses as token colors definable neither by results of experiments nor by calculations. From among the standard definitions of a Petri net, the one we use is (S, E, F), where S is a set of states, E is a set of events, and $F \subseteq S \times E \cup E \times S$ is the flow relation. In order to make room for guesses from a scientist and results of instruments inexpressible in the logic defined by a net but essential to setting it up, the nets used are all *net fragments*, which we define as follows. A net fragment is a structure (S, S_I, S_O, E, F) where S is a set of states of CPC's, and S_I is a set of states of input signals (*e.g.* from A/D converters to a CPC input buffer), disjoint from S, allowing for input to the CPC from a scientist and laboratory instruments. S_O is a set of states of output signals disjoint from both S and S_I, allowing for output from the CPC; the flow relation is expanded so $F \subseteq [(S \cup S_I) \times E] \cup [E \times (S \cup S_O)]$. States of S_I are assumed to have tokens placed in them by some means beyond the net, and states of S_O are assumed to have tokens removed from them by means beyond the net. Our pictures show stubs of arcs from states of S_I to events and from events to states of S_O while omitting the circles for these states. Associated with a net fragment is a "reduced net" obtained by omitting the states of S_I and S_O (and dropping the arc stubs); using this reduced net, one can explore issues of liveness and safety [**19**]. The events of E express computer logic and nothing else. As an example of a guess used in designing contact between equations and instruments, a mathematical model entered by a scientist as a colored token in an S_I state can assert whatever rules the scientist chooses to relate tokens received from instruments in S_I states to commands sent to them as colored tokens in S_O states. In this way the net fragment expresses the difference between such a model, with its guesswork, as a color on a token and how the instruments actually behave by producing colored tokens on their own.

4. Net-based portrait of guesswork needed to test and calibrate a quantum computer

In section 2 choices of equations to link to instruments were shown inescapably open to guesswork, bidding to make guesswork part and parcel of physics. The availability of net fragments described in section 3 brings within physics the study of contacts between equations and instruments by making available to analysis relations of sequence, concurrency and choice expressed in these contacts and in the guess-dependent actions that set the contacts up. Here we turn from nets themselves to attention to an example problem in which a net illustrates an important structure needed to link equations to instruments. Besides the net explicitly shown in figure 8, the availability of nets provides a framework in which to view the main topic of this section, the problem of resolving a choice of commands by which a CPC manages a quantum computer. That framework can be used in the future to ask other questions, to do with: how do the necessities of quantum-mechanical models, classical process control, and guesswork interact; how are FSM's as program structures affected by use of models that are quantum mechanical; how does the need for CPC's to mediate between quantum-mechanical equations and instruments change our understanding of quantum mechanics?

Turning to the case at hand, some telling illustrations of guesswork needed to link models to instruments arise in quantum computing. To build a quantum computer, say to solve problems of factorizing [**20**] and searching [**21**], a scientist must choose quantum-mechanical equations and laboratory instruments to work in harmony. Quantum computational models call for quantum gates that are unitary transformations, each a tensor product of an operator on a 1-bit or 2-bit subspace of the Hilbert space $\mathcal{H}$ and identity operators for the other factors of the tensor product. Note that each permutation of a non-identity factor with an identity factor is a distinct gate, calling for a distinct command to the instruments that implement it. For this reason, the number of quantum gates for an n-bit quantum computer grows faster than n. Call this number $G(n)$ and let the set of gates be $U_1, \ldots, U_G$. The most commonly used models of quantum computers can be put in the form [**22**]:

- Prepare a starting state independent of the input (*e.g.* the integer to be factorized).
- Transform the state by a product of quantum gates that depends on the input.
- Make a measurement independent of the input.

For an example, suppose the scientist assumes properties of models 1 through 4 and looks for the model that gives the least mean-square deviation between relative frequencies of outcomes and probabilities calculated by (2.13). To factorize an integer I, a classical computer program is converted to a product of $K(I)$ quantum gates, a number that rises faster than linearly with $\log I$. To obtain the effect of multiplying the gate transformations, the scientist must first have solved the model to determine the command $b_{U,j}$ for each gate U_j occurring in the product. As in the portrait in section 3 of putting tokens into a net fragment, the scientist programs a CPC to transmit a command b_v to prepare an initial state $|v\rangle$, commands $b_{U,j}$ for the gates needed, and a command b_M for a measurement. This endeavor is known to exhibit the following four features:

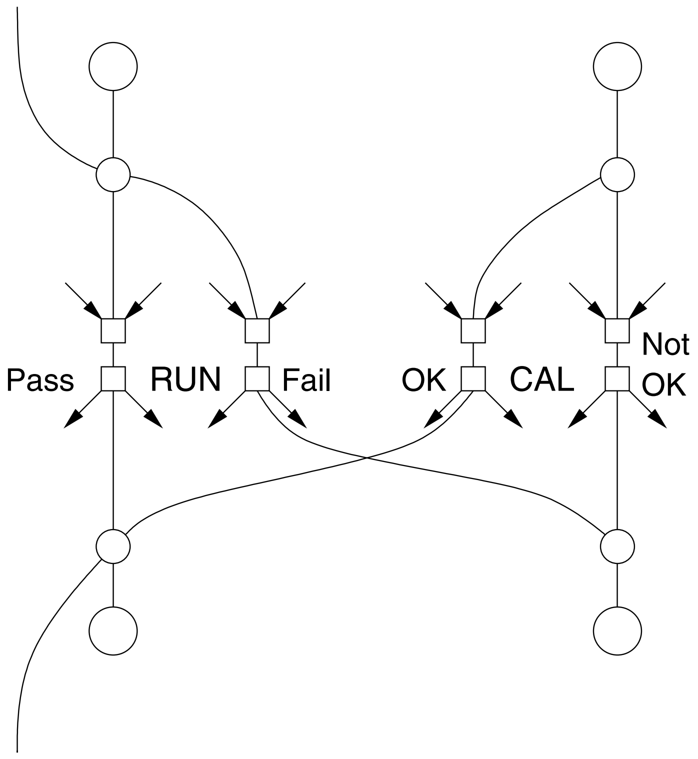

FIGURE 8. Alternating between running and testing a QC.

(1) The instruments are valuable as a quantum computer insofar as their results substitute for a more costly classical calculation defined by the model.
(2) An inexpensive classical computation (*e.g.* with the CPC) tests whether outcomes interpreted from results correctly solve the problem.
(3) Quantum indeterminacy imposes a positive probability that a result fails to provide a correct answer, so multiple tries with the instruments are the rule, and a wrong answer does not by itself imply a fault in the instruments.
(4) The tolerable imprecision of instruments implementing the chosen model of a quantum gate diminishes as the inverse of the number of gates $K(I)$ in the sequence [**23**].

Because the number of gates required in the product rises with the size of the integer to be factorized, feature 4 implies that passing the test for smaller integers is no guarantee against failure of the instruments to factorize larger integers, unless the model or the instruments or both are refined. This requires, in turn, that a CPC intended for use on progressively larger integers be organized to switch between a mode of using the quantum computer and a mode of inquiring into its performance, *e.g.* so as to determine commands that make it behave more precisely in accord with the desired quantum gates. This calls for a program for the CPC that expands the events “Use existing program” of figure 6 to that of figure 8.

4.1. Navigating the lattice of models to get better commands. As an example of what goes on within the coarsely portrayed event "Calibrate," suppose a scientist who uses a model α of the form $(|v\rangle, U, M)_B$ finds it works for small integers, but fails for bigger ones, which require more precise gates, which in turn requires calibrating (*i.e.* adjusting) the commands used to generate gates. This means giving up model α and choosing some alternative model β. A scientist does not choose a model all at once, but starts with some set of models and then narrows down on a smaller set, sometimes to a single model, a process open to guesswork at various stages. At one stage, the scientist may need to relax a constraint on models, leading to a bigger set of models from which to choose; at another stage the scientist may guess a new constraint, narrowing the set of models under consideration. By such a back and forth procedure, the scientist gives up U_α and arrives at a new function U_β (and hence a new model) with the hope that solving this function for a command $b_{U,j,\beta}$ for gates U_j, $j = 1, 2\dots$, that will succeed for factorizing larger integers than did the commands obtained from U_α. (This makes a need for models adapted to homing in on results, with some metric on B, so that a small change in the command b_U results in a small change in *e.g.* $U(b_U)$; while properties 3 and 4 are a start, going beyond them is left to the future.)

To get a better model, the scientist guesses a set of models and hopes to find within it a model that better fits measured results interpreted as outcomes. If no model of the set adequately fits these outcomes, the scientist can first broaden the set of models and next try to guess a property that will narrow the set, not to the original model, but to one that fits better. The recognition of guesswork assures us that so long as progressively more ambitious goals of precision keep being introduced, there is no end to the need for adjusting both models and the laboratory instruments.

4.2. Sample sizes needed to choose between models of gates. As discussed in section 2.5, the number of trials needed to statistically distinguish one model from another is bounded from below by the inverse square of a weighted statistical distance between the two models. Small numbers of experimental results can sometimes decide between distant models, but never between models that are close. In particular, distinguishing experimentally between two models for quantum gates can demand large samples:

PROPOSITION 4.1. *Models α and β that differ only in U, with spectral norm $\| U_\alpha(b_U) - U_\beta(b_U) \| = \epsilon > 0$, are statistically indistinguishable for a command b unless*

$$N(b) \geq \epsilon^{-2}. \tag{4.1}$$

PROOF. The models α and β under the stated condition are unitarily equivalent to a pair of models that differ only in $|v\rangle$ with $\cos|\langle v_\alpha|v_\beta\rangle| \leq \epsilon$. The proposition then follows from (2.10) and (2.11). □

We argue elsewhere that this is a serious and heretofore unappreciated challenge to bringing instruments into working order as quantum computers, made visible by attention to the need for guesswork in linking of laboratory instruments to equations of quantum mechanics [**24**].

5. Concluding remarks

Gödel proved that no one true structure could be generated by sitting in a room with blinds drawn, writing down axioms. Quantum mechanics tells us that with the blinds up and the world of physical measurement available, the situation remains much the same. Just as the openings for new axioms are uncloseable in mathematical logic, so in physics guesswork is part of the foundation.

The net formalism can be put to use both to address improving the contacts between equations and instruments, fostering advances in theory and in instrumentation, and, at a more abstract level, to pose problems pertaining to universal Turing machines adapted to process control. By formalizing commands to instruments, the techniques presented here extend the reach of set-based mathematics into the area of contact between equations and instruments, and open to study within physics of some of what physicist do in the course of doing physics. This extends a parallel beachhead established already in mathematics by Gödel's study of what a mathematician does to prove a theorem and Turing's analysis of a mathematician who makes a note by which to resume an interrupted computation.

6. Acknowledgment

We acknowledge Amr Fahmy for showing us our debt to Gödel's proof of incompleteness in mathematical logic. We are indebted to Steffen Glaser, Raimund Marx, and Wolfgang Bermel for introducing us to the subtleties of laboratory work aimed at nuclear-magnetic-resonance quantum computers [**25**]. To David Mumford we owe our introduction to quantum computing from the standpoint of pure mathematics. We are greatly indebted to Anatol W. Holt and to C. A. Petri for conversations years ago, in which each pointed in his own way to the still mysterious expressive potential of nets.

References

[1] F. H. Madjid and John M. Myers, *Formal distinction between quantum states and outcomes of their measurement*, Meas. Sci. Technol. **8** (1997), 465–472.

[2] P. A. M. Dirac, *The principles of quantum mechanics*, 4-th ed., Oxford University Press, Oxford, 1958.

[3] E. G. Beltrametti and G. Cassinelli, *The Logic of Quantum Mechanics*, Addison-Wesley, Reading, MA, 1981.

[4] W. K. Wooters, *Statistical distance and Hilbert space*, Phys. Rev. D **23** (1981), 357–362.

[5] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, MA, 1994.

[6] P. A. Benioff, *Decision procedures in quantum mechanics*, J. Math. Phys. **13** (1972), 908–915.

[7] F. H. Madjid and J. M. Myers, *Linkages between the calculable and the incalculable in quantum theory*, Annals of Physics **221** (1993), 258–305.

[8] K. Gödel, *Über formal unentscheidbare Sätze der* Principia mathematica *und verwandter Systems I*, Monatshefte für Mathematik und Physik **38** (1931), 173–198.

[9] K. Jensen, *Coloured Petri nets: Basic concepts, analysis methods and practical use*, Monographs in Theoretical Computer Science, an EATCS Series, Springer-Verlag, Berlin, Vol. 1, 2nd ed., 1996, Vol. 2, 1995.

[10] A. W. Crosby, *The measure of reality: Quantification and western society, 1250-1600*, Cambridge University Press, Cambridge, 1997.

[11] A. M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Math. Soc. **42** (1937), 230–265.

[12] R. P. Feynman, *Feynman lectures on computation*, Addison-Wesley, Reading, MA, 1996.

[13] G. S. Boolos and R. C. Jeffrey, *Computability and logic*, 3rd ed., Cambridge University Press, Cambridge, 1989.

[14] H. J. Genrich, K. Lautenbach, and P. S. Thiagarajan, *Elements of general net theory*, in W. Brauer, ed., *Net theory and applications*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, **254** (1987), 338–358.

[15] C. A. Petri, *General net theory*, in B. Shaw, ed., *Computing System Design*, Proceedings of the Joint IBM University of Newcastle upon Tyne Seminar, University of Newcastle upon Tyne, 7–10 September, 1976.

[16] D. Deutsch, *Quantum theory, the Church–Turing principle and the universal quantum computer*, Proc. R. Soc. Lond. A **400** (1985), 97–117.

[17] J. M. Myers, *Can a universal quantum computer be fully quantum?*, Phys. Rev. Letters **78** (1997), 1823–1824.

[18] U. Goltz, *Synchronic distance*, in W. Brauer, W. Reisig, and G. Rozenburg, eds., *Petri nets: central models and their properties*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, **254** (1987), 338–358.

[19] G. Berthelot, *Transformations and decompositions of nets*, in W. Brauer, W. Reisig, and G. Rozenburg, eds., *Petri nets: central models and their properties*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, **84** (1980), 359–376.

[20] P. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, Proc. of the 35th annual symposium on foundations of computer science, IEEE Computer Society, Los Alamitos, CA, 1994, pp. 124–134.

[21] L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th annual ACM symposium on the theory of computing, ACM Press, New York, NY, 1996, pp. 212–219.

[22] D. Deutsch, *Quantum computational networks*, Proc. R. Soc. Lond. A **425** (1989), 73–90.

[23] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM Journal on Computing **26** (1997), 1411–1473.

[24] J. M. Myers and F. H. Madjid, "Contact between laboratory instruments and equations of quantum mechanics," to be published in E. Donker and A. R. Pirich, eds, *Quantum Computing*, Proc. SPIE, **4047**, 2000.

[25] R. Marx, A. F. Fahmy, J. M. Myers, W. Bermel, and S. J. Glaser, *Approaching five-bit NMR quantum computing*, Phys. Rev. A., **62** (2000), 012310.

(John M. Myers) Gordon McKay Laboratory, Harvard University, Cambridge, MA 02138, USA

E-mail address, John M. Myers: `myers@deas.harvard.edu`

(F. Hadi Madjid) 82 Powers Road, Concord, MA 01742, USA