The QED Manifesto after Two Decades — Version 2.0

Ittay Weiss*
School of Computing, Information and Mathematical Sciences, The University of the South Pacific, Suva, Fiji

* Corresponding author. Tel.: +679 323 32219; email: weittay@gmail.com
Manuscript submitted January 13, 2016; accepted March 20, 2016.
doi: 10.17706/jsw.11.8.803-815

Abstract: In 1994 the QED Manifesto described an ideal whereby mathematics is communicated via a computerized system in a fully formalized fashion complete with automatic proof checking and other derived tools facilitating profound improvements to the way mathematics is taught, the way new results are disseminated, and ultimately to how mathematics is practiced. Two decades later it is safe to say the dream is not yet a reality. Analyzing some of the difficulties met thus far in the realization of the original Manifesto we propose here Version 2.0, presenting a view of a different ideal and a description of a system realizing part of that ideal.

Key words: Mathematical knowledge management, QED Manifesto, automatic proof checking, MCML.

1. Introduction

The QED Manifesto [1], published in 1994, states:"QED is the very tentative title of a project to build a computer system that effectively represents all important mathematical knowledge and techniques. The QED system will conform to the highest standard of mathematical rigor, including the use of strict formality in the internal representation of knowledge and the use of mechanical methods to check proofs of the correctness of all entries in the system." Whether directly or indirectly inspired by the Manifesto, several systems are in existence that can be said to be work-in-progress towards realizing it. A survey of some of the more prominent projects in the context of the Manifesto is given in [2]. However, none of the systems today can be said to have realized the vision presented in the Manifesto. In current systems of automated proof checking, for instance Coq, HOL, Isabelle, and Mizar, the conversion of a typical proof to the formal system requires a considerable amount of work and typically results in a proof four times longer than what one started with. Quite pessimistically, [3] estimates the cost of constructing a fully formalized library of mathematics at 140 people years. Extrapolating from this current state of affairs the following discussion is a qualitative conjecture of the situation.

There appears to be a repulsive force between formality and cognitive comprehension, a sort of cognitive uncertainty principle, described as follows. For a given mathematical proof let $\Phi \geqslant 1$ measure how formal the proof is, with $\Phi = 1$ being completely formal and decreased formality corresponding to increased Φ. Similarly, let $\Gamma \geqslant 1$ measure how readable, from a cognitive point of view, the proof is, with $\Gamma = 1$ being extremely readable and decreased readability corresponding to increased Γ. Let ℓ denote the length of the proof. Experience suggests that there exists a neuro-psychological constant $h > 0$ such that the inequality $\Phi \cdot \Gamma \geq 1 + h \cdot \exp(\ell)$ holds for all proofs. The formula expresses the following heuristics. If a proof is very short, then both Φ and Γ can be close to 1 as there is little difference between how formal the proof is and

how cognitively understandable it is. For longer proofs the term exp (ℓ) becomes dominant and as soon as it passes a threshold determined by h, a fully formalized proof, i.e., having $\Phi = 1$, receives a high value of Γ, i.e., it is cognitively doomed. Clearly, the value of h varies from person to person and is dependent on personal experience and expertise. Nonetheless, judging by the uniformity of style in the communication of modern mathematics, h appears to be a cognitive constant across the population of mathematicians (it is perhaps a topic for neuropsychology to conduct experiments to estimate h). In the polarity ensuing from this uncertainty principle it can be said that the original Manifesto seeks to minimize Φ. Version 2.0, the subject matter of this article, is concerned with minimizing Γ. We note that the formal aspects of mathematics communication are dealt with extensively as a proper mathematical area of study (see [4] for an excellent overview). The cognitive aspects of mathematics communication are also addressed, albeit typically not by research mathematicians. This unfortunate disparity in practice represents a miscommunication which is perhaps yet another incarnation of the cognitive uncertainly principle.

It is crucial to state that QED Version 2.0 is neither meant to replace Version 1.0 nor to indicate that it is in any sense better. By analogy to programming languages, the difference between QED 1.0 and QED 2.0 is akin to the differences between low level programming paradigms and higher level ones, respectively. If formalizability and cognitive readability can be unified by some currently unknown clever mechanism (as [5] seems to suggest may be the case), or if the process of turning a 'higher level' proof into a formal 'lower level' one can be automated, then QED 1.0 and QED 2.0 will merge into a single product. Otherwise, each paradigm will have its own merits and its own applications, and products will coevolve utilizing various mixtures of the two versions.

The presentation of the QED 2.0 Manifesto given below follows in spirit the structure of the original Manifesto. In particular, we describe the ambition, contemplate and respond to objections to the proposed ideal, and discuss some technicalities. A significant difference from the original Manifesto is in the final section describing the Mathropolis system at its current stage of development, a system which is QED 2.0 in a strong sense. I developed Mathropolis and the ideas expressed below in tandem, one influencing the other, and I hope its inclusion here elucidates the abstract ideas presented. Much as was the case with the original Manifesto, the ideas presented herein are certainly already in circulation, if not in print then in the minds of practitioners. I thus happily relinquish all claims of originality.

To insert images in *Word,* position the cursor at the insertion point and either use Insert | Picture | From File or copy the image to the Windows clipboard and then Edit | Paste Special | Picture (with "Float over text" unchecked).

INTERNATIONAL ACADEMY PUBLISHING reserves the right to do the final formatting of your paper.

2. What Is the QED Project 2.0 and Why Is It?

2.1. From the Original Manifesto

"QED is the very tentative title of a project to build a computer system that effectively represents all important mathematical knowledge and techniques. The QED system will conform to the highest standards of mathematical rigor, including the use of strict formality in the internal representation of knowledge and the use of mechanical methods to check proofs of the correctness of all entries in the system." QED 2.0 is the very tentative title of a project to build a computer system that effectively represents all important mathematical knowledge and techniques. The QED 2.0 system will conform to the highest standard of clarity of exposition and accepted rigor, including paradigms for reusability, encapsulation, uniformization, and language independence in the communication of mathematics.

The QED 2.0 project will be a major scientific undertaking requiring the cooperation and efforts of hundreds of mathematicians, considerable ingenuity by many computer scientists, and a careful analysis of

the components and standard of the state-of-the-art of mathematics communication, as well as support from research agencies and the industry. In the interest of enlisting a wide community of collaborators and supporters, we now offer goals for the QED 2.0 project that we see as justifying the endeavor.

Goal 1: Independence from Convention. First and most simply, the way mathematics is typically encoded today, that is as a LaTeX file, necessitates a rigid choice of notation. Once an author had finished the type-setting of her latest book the LaTeX source code will only ever produce the same output, except for the formatting of the document (which is precisely why the author chose to use LaTeX). If the reader has a strong preference for a different notation (e.g., denoting functions by $f(x)$ as opposed to $x(t)$, naming sets by the letters S, T rather than A, B or X, Y , using additive vs. multiplicative notation in an abelian group, or using distinguishing symbols, e.g., $\oplus$ or $\boxdot$ rather than + or $\bullet$ for binary operations), then the entire source file will have to be adjusted in what would be an immensely time-consuming effort. The QED 2.0 system we envisage will decouple content from presentation. Using it, the author of a book will only concentrate on the content she wishes to convey. The QED 2.0 system will allow any user to specify (or choose from a repository) his favorite convention for the presentation of mathematics and will then dynamically converts the source file into a book in the preferred style of the user.

Goal 2: Independence of Content. The author of any mathematical content always keeps her intended reader in mind. She must streamline her writing to the assumed level of maturity and expertise of her imagined average reader. Any real reader whose actual level of expertise significantly deviates from that imagined by the author will experience the output as either unintelligible or tedious and boring. The QED 2.0 system, as part of the decoupling of content from presentation, will enable the author to spend less time thinking about the reader and more time on producing the content. QED 2.0 will then automatically produce a document tailored according to the particular requirements of each individual user, thereby diminishing the need for different books, addressing different audiences, but sharing essentially the same mathematical content.

Goal 3: Dissemination of New Results. Reading and writing articles is often difficult, and not only due to the need to grasp inherently complicated content. Even expert mathematicians, and certainly young ones, often find, while reading an article, that some of the terms and results used in it are foreign to them. They then consult earlier articles for those results and often find they need to repeat the process until they are in familiar territory. What was a single article to be consumed is now a pile of articles to be consulted. More often than not each article will employ different notation, only adding to the confusion. The process of writing an article exhibits the other side of the same coin. An article is to be concise, focusing on the new results. But to be concise readability is sometimes sacrificed. The QED 2.0 system will allow a reader of papers to access results more easily and for a writer of articles to code her newest result and its proof directly into the system. She will then choose her favorite convention of notation, and click the 'create article' button to automatically obtain a mathematically correct streamlining of all the results, up to a certain depth to avoid trivialities, leading to the proof of her new result. Better yet, we envisage a time when the 'create article' button will become obsolete. Instead, she will announce her new result on the QED 2.0 system, allowing any reader to choose his preferred mathematical convention and depth of expansion of auxiliary results.

Goal 4: Modularity and Reusability. With open source practices becoming ever more prevalent, it is an immediate realization that obtaining the LaTeX source file of any piece of mathematical work (book, article, final exam, a set of exercises, worked examples, solutions etc.) is of rather limited use when creating related content. Merging pieces of code from different sources into a single source is a daunting task due to terminology mismatches, style differences, and the order in which results are presented, to the extent that most authors prefer to just type it all up from scratch, thus rewriting the wheel time and again. The QED 2.0

system will have tools for a seamless reusability of code through encapsulation. It will enable incorporating snippets of code from different sources without any difficulty and will automatically detect a faulty linearization, i.e., when a certain result uses results presented later on, and will also suggest customizable faultless linearization.

Goal 5: Learning and Teaching Management. With recent trends in online and adaptive learning the QED 2.0 system will have marked benefits for lecturers and for students. We expect that the QED 2.0 system will be integrated with course management tools to provide pinpointed and adaptive content. Lecturers will be able to design content for their course by selecting from the repository of knowledge and interactively build a course. Students will be provided with content that matches their preferences in both depth and style. Students will be encouraged, and some will be delighted, to contribute to the QED 2.0 project by enlarging and enhancing its repository and to form part of a community. Moreover, through the principle of separation of content from presentation, the content markup language itself will serve as a learning tool for students taking their first steps into the art of communicating mathematics.

Goal 6: Language Flexibility. The underlying logic and flow of a mathematical proof remain the same whether the proof is written in English, French, Chinese, Elvish, or Klingon. With the increase in long distance international collaborations and with advances in natural language processing the QED 2.0 system will have the capability to automatically translate between languages. The translation will employ the separation of content from form to guarantee that the original and the translation share the exact same mathematical content (in both logical validity and the flow of the proof). The translation of the rest of the text will be as accurate and useful as the latest natural language translation algorithms capabilities can produce.

Goal 7: Organization of Knowledge. The volume of mathematical knowledge is staggering. Not only is it the entirety of mathematics that is well beyond the scope of mastery for any single person, the same can be said of each of the major areas of mathematics (e.g., topology, algebra, analysis, logic, etc.). Moreover, an overwhelming quantity of new mathematics is accumulated each year. For users of mathematics as well as for novice and expert mathematicians, navigating the ocean of results is becoming an ability that appears to require superhuman capabilities. Both QED 1.0 and QED 2.0 aim to provide invaluable tools for this purpose. The system will have tools to navigate, search, and compare results. By analyzing the interdependencies in the coding of various results the system will be able to automatically locate similar results suspected of being related (or perhaps duplicates), thus identifying areas of mathematics that are more closely related than what appears to be. A related well-known problem in current mathematics is measuring the impact factor of an article. This is a nontrivial matter as indicated by ongoing discussion about the shortcomings of the various impact factor formulas commonly used (see [6] and the hundreds of articles citing it). With the organization of large portions of mathematics as envisaged by QED 2.0, the ability to measure not just the impact of an article but that of any given result or definition will become feasible. Such an atlas of the realms of mathematics by importance may serve to more efficiently direct research efforts.

Goal 8: Testing Ground for Other Domains of Knowledge. Obviously many of the goals set above have analogues in other areas of research and knowledge accumulation (e.g., physics, biology, medicine, etc.). Such areas of research will also greatly benefit from a system facilitating easier communication, adaptable output, and reusability of written content. QED Version 2.0 will serve as a testing ground for similar systems in other domains. Exploiting the highly structural nature of mathematics, it is expected to be easier to realize the ideas presented above than it would be to realize the analogues in other areas. The experience gained from constructing various component of QED Version 2.0 will be of great practical value when developing similar systems for other sciences, and, perhaps, beyond science.

3. Some Objections to the Idea of the QED 2.0 Project and Some Responses

Any worthwhile endeavor should be subjected to criticism. We test here the ability of the QED 2.0 philosophy to responding to criticism rising from our own suspicions of its weaknesses and doubts of its feasibility. Needless to say, this is not an exhaustive list of objections nor is the given responses the only possible ones.

Objection 1: Unfavorable Input-Work to Output Reward Ratio. The reason QED 1.0 has not materialized yet is due to the cumbersome nature of the coding process required when formalizing a proof; there is simply too much work for too little a reward. The QED 2.0 system will require a content markup language that could be just as cumbersome to use as the best of the currently existing languages for formalizing proofs and thus will fail to attract sufficiently many contributors to get off theground.

Reply to Objection 1: QED 2.0 is not a better version than QED 1.0 but rather a different vision of a common ideal. There is no a-priori reason to assume that the formalization process QED 2.0 will require is of the same level of complexity as that required by QED 1.0. In fact, we argue that some aspects of the QED 2.0 system can be accomplished by a coding process that is very much like LaTeX enhanced with variables and an automatic convention generation, resulting in code that is about as complicated as the corresponding LaTeX code would be.

Objection 2: Unfeasible Objective. Mathematics is now so large that the hope of incorporating all of mathematics into a system is utterly humanly impossible, especially since new mathematics is generated faster than it can be entered into any system.

Reply to Objection 2: Firstly, the QED 2.0 project will become useful with even a tiny portion of current mathematical knowledge coded into it. In fact, the system will facilitate a smoother creative process and easier production of mathematical content even without any prior content already present. For instance, a mathematician using QED 2.0 to communicate results in, e.g., group theory does not need to have set theory already present. The system will be able to expand and serialize results only if those are present in the repository, but the set theory component can be added later, and by somebody else. Secondly, the system will allow gradual and distributed modes of operation. Different mathematicians will introduce different coding of common mathematics knowledge as well as different conventions for notations. Much like computer language libraries, these modules will improve through the contributions of other users as the repository grows. As a result, gradually, the QED 2.0 system will become the medium for communicating results directly. In short, the immense rate of creation of new mathematics is a reason in favor of the QED 2.0 system, not against it.

Objection 3: Algorithmic feasibility. There is no evidence that an efficient content markup language can be devised that will be both effective in the internal organization of the repository and non-intimidating so as to attract enough collaborators.

Reply to Objection 3: There is no doubt that some aspects of the QED 2.0 system (such as automated notation according to coded convention guaranteeing consistent use of symbols throughout a document) is algorithmically feasible. Other QED 2.0 aspects are certainly related to other successful algorithms in natural language processing. We thus see sufficient encouraging algorithms around to indicate that the feasibility of the QED 2.0 project is not far-fetched.

Objection 4: Impossibility by Inertia. If QED 2.0 were feasible, it would have already been underway several decades ago.

Reply to Objection 4: Certainly the same could have been said about pre-TeX suggestions to improve the writing process of documents. Moreover, problems tend to be solved only after they are posed, and problems rarely get posed if they are small problems. The sheer immensity of mathematical knowledge is a rather recent phenomenon and with it the need to drastically rethink the modes of mathematics

communication is relatively new. Yes, QED 2.0 could have been under way decades ago but there was not enough incentive to justify changing the good old methods. In other words, it was not broken (enough), so nobody fixed it. But it can be done (or at the very least, there is no evidence it cannot be done).

Objection 5: Formality vs. Creativity. Good mathematicians will never agree to work with formal systems because they are syntactically so constricting as to be inconsistent with creativity.

Reply to Objection 5: If a formal system is formal enough to be able to justify using it yet informal enough so as not to constrict creativity, then the objection dissolves. The ideas presented herein stem from the suspicion that systems intending to formalize mathematics for the sake of automatic proof verification inherently require constrictive formalization processes. QED 2.0 calls for creating systems intending to formalize mathematics for the sake of easier communication. A key property of QED 2.0 is that with that aim in mind the formalization process is not only non-constrictive but in fact aids in the creative process. Good mathematicians will not only use QED 2.0 to communicate their results, they will actively participate in the coding of known results and the coding of useful conventions as part of their creative process.

Objection 6: Lack of Interest. Since the QED 2.0 system does not offer mechanical proof checking or assistance in finding proofs, mathematicians will see no reason in going through a formalization process for no apparent gain.

Reply to Objection 6: The key property of QED 2.0 is its immediate usefulness even without any content already in the system. It is a significant added value for any author to know that the content she just coded into QED 2.0 can produce much more flexible outputs than what essentially the same amount of coding using LaTeX can produce. It is also comforting to know that her code can easily be used again without the need to retype familiar things again and again. Therefore, the system is likely to draw the attention of sufficiently many contributors; both the philanthropist aiming at helping others save time and effort as well as the working mathematician seeking to save her own time and efforts. In a strong sense, the ability to more efficiently communicate mathematics is more important to most mathematicians than obtaining a machine validated stamp of approval of their results. Most mathematicians trust their own abilities to produce correct informal proofs, which they then wish to have other mathematicians appreciate. The aim of QED 2.0 is precisely that.

Objection 7: Finish QED 1.0 First! The QED 2.0 project represents an unreasonable diversion of resources to the creation of mere syntactical gadgets while QED 1.0 is still underway.

Reply to Objection 7: QED 2.0 is neither a successor nor a competitor of QED 1.0, and thus there is no immediate necessity to finish QED 1.0 before commencing with QED 2.0. Whether or not QED 1.0 can/should/will become a reality is independent of whether or not QED 2.0 can/should/will become a reality.

Objection 8: Doubt. The QED 2.0 system will be so large that it is inevitable that there will be mistakes in its structure, and since the QED 2.0 system will offer no automated verification means it will be unreliable.

Reply to Objection 8: Since QED 2.0 does not purport to automatically check any proof it contains it also does not suggest eliminating the need for expert mathematicians to verify claims and proofs. QED 2.0 will not automate or replace the standard process of refereeing as part of the publication of results. The QED 2.0 repository, much like the arXiv, will be a "contribute as you like - use at your own risk" system. The system will allow for an active discussion and improvement of results and proofs, whether anonymously or not, in order to enhancing the ability of any user in assessing the quality of any piece of mathematics she investigates. QED 2.0 aims to free authors from the need to worry about syntax, but the semantics, i.e., the mathematical correctness of proofs and the validity of stated theorems, remains the responsibility of the users.

4. Units

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). **This applies to papers in data storage.** For example, write "15 Gb/cm^2 (100 Gb/in^2)." An exception is when English units are used as identifiers in trade, such as "3½ in disk drive." Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength H is A/m. However, if you wish to use units of T, either refer to magnetic flux density B or magnetic field strength symbolized as $\mu_0 H$. Use the center dot to separate compound units, e.g., "A·m^2."

5. More or Less Related Efforts

5.1. Figures and Tables

From the original Manifesto: "In some sense project QED is already underway, via a very diverse collection of projects. Unfortunately, progress seems greatly slowed by duplication of effort and by incompatibilities. If the many people already involved in work related to QED had begun cooperation twenty-five years ago in pursuing the construction of a single system (or a federation of subsystems) incorporating the work of hundreds of scientists, a substantial part of the system including at least all of undergraduate and much of first year graduate mathematics and computer science, could already have been incorporated into the QED system by now. We offer as evidence the non-trivial fragments of that body of theorems that has been successfully completed by existing proof-checking systems."

Much of this remains true for QED Version 2.0. Before we embark on a short review of some modern projects viewed through the lens of the ideas presented above, we wish to point at the fact that now, two decades further past the original Manifesto, progress is still slowed down by even more projects addressing various issues in the formalization and communication of mathematics and while significant portions of undergraduate mathematics have been formalized, one must admit that this formalization brought little change (if any at all) to the way mathematics is communicated and taught. Perhaps the formalization of mathematics is not of great interest to the working mathematician, the student of mathematics, or the user of mathematics, while other issues (those of communication) are.

We can point at several major areas of recent development which share some aspects with QED 2.0 philosophies. Firstly, automated reasoning, an active field of research which already produced several systems for automatic theorem proving and for automatic proof checking, is an effort clearly in the spirit of QED Version 1.0, but also with some aspects supportive of QED 2.0 ideas (to emphasize again that QED 2.0 is not meant as a successor to QED 1.0, consider the strongly QED 1.0 Flyspeck Project, and see [7], a project aimed at proving Kepler's conjecture on sphere packing). Second, the emergence and development of web-based content markup languages, like MathML, OpenMath, and OMDoc, also share various aspects with QED 2.0 ideas. Third, content management (obviously a topic of immense importance for any system aiming at effectively encoding large volumes of mathematical knowledge) is addressed in the field of Mathematical Knowledge Management, and in this context we mention:

- Publishing mathematics lecture notes as linked data (see [8]);
- Collaborative content creation (see [9]);
- Languages for coding mathematics (e.g., [10]); and
- Aspects of data retrieval (see [11] and [12]).

Finally, we mention sTeX (see [13] and [14]), a package of TeX macros providing a semantic enrichment to TeX production. Another recent aspect is influenced by cloud technologies and scale development, see,

e.g., [15] and [16]. We note that all of these projects exhibit various mixtures of QED 1.0 and QED 2.0 ideologies.

With the advancement of various formalizing systems for automatic proof checkers, as already noted above, there is (still) a clear divide between the formal and the readable. This issue, which in a sense is the raison d'êtres of the QED 2.0 philosophy, is starting to receive attention early on in the specification of emerging systems in development. We mention here the MoSMath project (see [17]) and FMathL - Formal Mathematical Language (see [18]) - a project carried out at the University of Vienna. FMathL is a modeling and documentation language for mathematics designed to achieve the following goals (quoted from [19]):

• Separate completely modeling aspects (declarative) and algorithmic aspects (imperative).

• The model is (almost) publishing quality documentation; the latter can be automatically generated from the model.

• The formal model is specified close to how it would be communicated informally when describing it in a lecture or paper, except that it does not suppress any relevant detail.

• Enables users to express arbitrary mathematics.

• Easy exchange of problems in high level format.

• Very transparent use, fully modular.

• User extensible.

• Mathematician-friendly: write as in a textbook.

• Any mathematical problem can be specified (though not necessarily successfully solved by a solver to which it is passed - some problems are too difficult or even undecidable).

• Prepares the way for an easy-to-use formalized mathematics interface covering the contents of undergraduate mathematics courses and beyond.

• Later extensions only affect the interfaces, not the problem structure, and can be contributed by third parties.

The fact that the projects mentioned above, and other projects, involve aspects that are both QED 1.0 and QED 2.0 is not surprising and illustrates two phenomena. The first is that it is certainly possible, and desirable, to mix the two paradigms according to need and ambition. The second phenomenon is the recent rise in projects that address issues of usability for the working mathematician on a level that is at least as important as the ability of the system to automatically check the validity of the coded content - perhaps a first wave towards applicability in the spirit of QED 2.0 at the cost of losing (some) aspects of QED 1.0 capabilities.

6. Mathropolis-Some Technicalities by Means of an Example

An important early technical step will be to 'get off the ground', linguistically speaking, which we will do by rooting the QED 2.0 system in a 'root content markup language', whose description requires only a few pages of typical lingo-mathematical text. Anticipating the development of various variants descending from the root language the main task of the root language is to set a common ground, identifying and articulating adequate and complete key components of mathematics communication. The terms 'adequate' and 'complete' are borrowed from their use in logic. A logical system is adequate if it only allows the derivation of true statements from axioms while it is complete if the system is strong enough to derive all true statements. The root markup language will be linguistically adequate in the sense that it does not allow one to code a syntactically faulty proof and it will be complete in the sense that it allows the coding of any syntactically correct proof.

In the discussion above it was mentioned that any particular project may mingle both QED 1.0 and QED 2.0 philosophies. Referring back to the cognitive uncertainty principle and the Γ, Φ scales, projects may be

classified by how much they attempt to minimize Γ and how much they aim to minimize Φ. A useful computer system will, of course, seek to minimize both as best as the cognitive uncertainty principle allows. If formality and cognitive readability cannot coexist, then Γ and Φ cannot both attain their minimum in any given formal system, necessitating a compromise. It is then unclear what the optimal ratio of Γ to Φ would be for the purposes of the general mathematics community and it is likely that various mixtures will be developed.

We now describe a system which is purely Γ, that is it aims to minimize Γ while completely ignoring Φ; it has neither tools for automatic proof checking nor any means to protect the user from logical fallacies (in this sense TeX and LaTeX are also purely Γ). The main objective of this project is to test the limits of what can be achieved by completely neglecting formality and semantics, and to develop, by experience, new techniques and technologies, in anticipation of a future merge with QED 1.0 paradigms. The Mathropolis system consists of a mathematical content markup language, called MCML, a compiler that reads MCML code and outputs LaTex-ready snippets, and a growing repository of mathematical content written in MCML. The markup language MCML itself has three layers; one to code commands, one to code notation conventions, and one to code content. To illustrate the capabilities of Mathropolis we consider a very simple yet typical example of LaTeX production of mathematical content. Consider the claim

Let A, B, C be sets. If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

And its proof

Let $a \in A$. Since $A \subseteq B$ it follows that $a \in B$.

Since $B \subseteq C$ it follows that $a \in C$. We thus conclude that if $a \in A$, then $a \in C$ and therefor $A \subseteq C$.

The LaTeX codes that produces this output are, respectively, Let A, B, C be sets. If $A\subseteq B$ and $B\subseteq C$, then $A\subseteq C$.

And

Let $a\in A$. Since $A\subseteq B$ it follows that $a\in B$. Since $B\subseteq C$ it follows that $a\in C$. We thus conclude that if $a\in A$, then $a\in C$ and therefor $A\subseteq C$.

The following are the corresponding MCML content coding for the claim

=> Statement[Set Inclusion Transitivity]
=> Ingredients: A,B,C ==> Set
=> Auxiliary:
a1 = $A \subseteq B$,
a2 = $B \subseteq C$,
c = $A \subseteq C$
==> If $a1$ and $a2$, then c <==
And for its proof
=> Proof of [Set Inclusion Transitivity]
==> Let $x \in A$. Since $a1$ it follows that
$x \in B$. Since $a2$ it follows that $x \in C$.
We thus conclude that if $x \in A$, then $x \in C$ and therefor c. <==

The resemblance with LaTeX is apparent, and deliberate, but it is only a surface resemblance which under the hood is shattered in two important aspects. Firstly, none of the above is a LaTeX command. Instead these are MCML commands coded using the command coding capabilities of MCML. The purely object-oriented command coding unit (which is omitted here) consists of defining the mathematics type Set (which is used in describing the "Ingredients" in the code above) and in particular specifies that \subseteq is a method of Set. The "Ingredients" line above thus instantiates three objects of type Set and thus the subsequent calls to, e.g., $A \subseteq B$ are valid MCML calls. Similarly, $x \in A$ is a call to an MCML

coded method \in (attached to the object A, so contravariant method calls are allowed) which instantiates an object of type Element (a mathematical type which can never be instantiated directly, only via calls done by instances of Set). It is important to emphasize that, for instance, the mathematics type Set is not a data type and does not implement a set-like data structure at all. It only exists in order to define; what are valid syntactical commands one can perform with a set. In particular, it is impossible to add elements to any of the Set instances, or to query for membership, or to perform any other mathematical operation on them.

Secondly, the example above illustrates the separation of content from presentation. Mathropolis can convert the MCML code above into a LaTeX snippet once it is provided with a notation convention unit, coded in MCML as well. The convention unit specifies whether sets are named by letters from the beginning, middle, or end of the alphabet, or whether sets are to be subscripted, superscripted, or primed. It also specifies the way elements of sets should be named. Thus, when executed with a particular convention, Mathropolis will produce the exact same LaTeX snippets given above, while when run with a different choice of convention unit the resulting output LaTeX code will produce the statement

Let S1, S2, S3 be sets. If S1 ⊆ S2 and S2 ⊆ S3, then S1 ⊆ S3.

With the proof

Let x ∈ S1. Since S1 ⊆ S2 it follows that x ∈ S2. Since S2 ⊆ S3 it follows that x ∈ S3. We thus conclude that if x ∈ S1, then x ∈ S3 and therefor S1 ⊆ S3.

The convention coding capabilities allow for the notation to depend on the number of instances of, e.g., sets or elements in a particular scope, as well as on the nature of the variables or auxiliary expressions. Moreover, during compilation of the MCML code into LaTeX, the produced labels of each component are guaranteed to be unique within the relevant scope (or to produce an error if the convention is inconsistent and unresolvable). This allows the user to concentrate fully on the mathematical content and be assured that after compilation with a particular convention the end result will be notational consistent.

As mentioned above, MCML coding does not protect one from making logical fallacies, even very blatant ones. For instance, if instead of the mathematically correct MCML code for the proof above one would write

=> Proof of [Set Inclusion Transitivity]

==> Let $x \in C$. Since $a1$ it follows that $x \in B$. Since $a2$ it follows that $x \in A$.
We thus conclude that if $x \in A$, then $x \in C$ and therefor c. <==

Then MCML will compile without warnings or errors and produce a LaTeX snippet of mathematical nonsense. However, MCML does protect one from syntactical nonsense to a significant degree. For instance, MCML will compile without warnings only if all variables were defined before used and if all commands are indeed methods of the corresponding mathematics types. So, for instance, since general sets cannot be added, a command such as $A+B$ somewhere in the MCML code above will cause a *"Warning: '+' is not a method of Set [encountered on line ...]"* message. The reason that such calls produce a warning and not an error is in order to allow for a smooth transferability from LaTeX code to MCML code. Any part of MCML code which is neither a defined variable nor a method is treated either as text or as a hardcoded LaTeX command, but produces a warning.

The process of converting a written mathematical proof, for instance already as a LaTeX document, into the Mathropolis system requires one to separate the content from the form by declaring the variables and auxiliary expressions that play a role in the proof. The amount of work this requires is moderate. A proof tends to use the same variables multiple times and the same auxiliary expressions repeatedly. Heuristically, the number of variables and auxiliary expressions is logarithmic in the length of a proof, and thus the preparatory part preceding the proof would typically be rather short. The coding of the proof itself may actually become shorter than the original due to the use of short variable names for auxiliary expressions which may be quite lengthy. All in all, the conversion rate from LaTeX to MCML code is, roughly, length

preserving.

The compilation process of MCML code to LaTeX code consists of creating a tree representation of the expressions and the method calls and then, in conjunction with the convention unit, dynamically producing unique labels as required at each scope. One other feature of Mathropolis is that it can automatically expand the tree to include definitions of concepts that are used, statements of theorem that are called, as well as their proofs. So, for instance, Mathropolis can display not only the claim and proofs above but also the definitions of set inclusion and set membership automatically. With an ever-growing repository of mathematical content coded in Mathropolis one can produce customized streamlining of results and their proofs, delving to any desired depth of expansion.

7. Concluding Thoughts

New computing technologies have transformed common practices in numerous areas of human activity. Typically, there is an initial tendency to adhere to one's familiar and well-tested ways but if a new technological development proves to truly provide new useful tools and to free practitioners from tedious tasks not directly related with their art, then it catches like fire. The community does not always know what tools it needs but when given the right tools the community will adopt them. In the mathematics community, an example of this phenomenon is TeX/LaTeX which required a short adjustment period and subsequently became the main means of communication.

In contrast to the revolution created by document markup languages, tools of automated reasoning in the spirit of QED 1.0 do not catch like fire. It seems that proof assistants are of limited use and that interactive theorem proving environments are more of a didactical tool than a research tool (see [20] for a more details discussion). This situation can be attributed to the fact that such tools are not (yet) friendly enough. However, with two decades past QED 1.0, one must at least entertain a deeper reason. Perhaps the average mathematician does not need the assistance of computers for finding proofs or validating proofs. She can do that on her own quite well. What she would appreciate though is a better way to communicate her results on a computer.

TeX has nothing to do with mathematics. It is a typesetting system. Similarly, QED 2.0 has nothing to do with formal mathematics, logic, or semantics. It is a typesetting ideology. Deeply inspired by the success of TeX we draw upon it to recognize the next revolution: the elimination of typewriters.

The Typewriter — A Parable. When my grandmother wrote her first book she did so on a typewriter. She had made sure that words rhymed as she wanted them to and that verses correlated to her heart's desire in order to please the ears of the reader. She had plenty of material written on various pieces of paper and then she had to produce a manuscript. Typing page after page on her trusty typewriter her thoughts had turned from fantasy in her mind to fantasy written in stone. The word 'stone' here should be taken quite seriously. If her fingers made a mistake typing the wrong letter here and there, the result was a usually unsuccessful attempt to erase the mistake using various techniques that sooner rather than later turned to wishful thinking and eventually to a crumpled up sheet of paper in the wastebasket and a fresh one placed into the typewriter in order to re-type the entire page (if not the entire chapter, or the entire book).

When my grandmother wrote her second book she did such on a word processor. She had fewer pieces of paper lying around and did most of her creative writing directly into the word processor. The fantasy in her mind had made its way to her trusty floppy disks to be accurately reproduced time and again on the computer screen. She was delighted at the ease with which she corrected typos and even amazed with the kindness of the computer to fix them for her, and she was pleased with her ability to allow entire sentences to drift across the screen to find their perfect location. When the typing was done, her words conveyed her

thoughts as she wanted them to and it was time to tend to the formatting of the text. Once the creative writing was finished, the spacebar, backspace, and arrow keys were the only keys on the keyboard to be used. Making sure that the manuscript was visually pleasant became a daunting task. An extra space here to align one line with another caused a mis-alignment somewhere else. A decision to change the font in the caption of one chapter necessitated a painstaking correction in font size to all other chapter captions. Automation heaven was gone without a trace and it felt again like working on a typewriter, albeit a very sophisticated one.

When I wrote my first book I did such on a document processor (i.e., LaTeX). I did not need to worry about the layout nor too much about spelling and could thus concentrate on the mathematical content. One day, with several chapters already written I realized it would have been better to use the symbol V for a vector space rather than R. The automation capabilities of the search-and replace dialogue-box offered little comfort and I found myself manually going over tens of thousands of LaTeX code lines changing my original unfortunate choice of notation. I felt constricted by the linear progression of the writing and the constant need to contemplate the reader's whims and needs regarding depth of exposition, number of illustrating examples, hints to unsolved exercises, suggestions for further reading, etc. Once more, it felt like working on a typewriter.

The paper-like rigidity of a LeTeX document is something to be replaced by a dynamic process of mathematical content creation. If TeX represents the phase shift from word processing to document processing, then QED 2.0 represents the transition from document processing to content processing. Just as document markup languages freed the author from the need to format the document, so will the content markup languages of the QED Version 2.0 paradigm free the author from the need to format the mathematical content.

Acknowledgment

The author is grateful to Rahel Berman for clarifying conversations on the subject matter of this article.

References

[1] Boyer, R. *et al*. (1994). The QED manifesto. *Automated Deduction–CADE*, *12*, 238–251.

[2] Wiedijk, F. (2007). The QED manifesto revisited. *Studies in Logic, Grammar and Rhetoric*, *10(23)*, 121–133.

[3] Wiedijk, F. (2002). Estimating the cost of a standard library for a mathematical proof checker. Retrieved, from: http:// www. cs. ru. nl/freek/notes/mathstdlib2.pdf

[4] Hales, T. C.(2008). Formal proof. *Notices of the AMS*, *55(11)*, 1370–1380.

[5] Asperti, A., Geuvers, H., & Natarajan, R. (2009). Social processes, program verification and all that. *Mathematical Structures in Computer Science*, *19(5)*, 877– 89.

[6] Amin, M., & Mabe, M. (2000). Impact factors: Use and abuse. *Perspectives in publishing*, *1(2)*, 1–6.

[7] Kaliszyk, C., & Urban, J. (2014). Learning-assisted automated reasoning with flyspeck. *Journal of Automated Reasoning*, 1–41.

[8] David, C., Kohlhase, M., Lange, C., Rabe, F., Zhiltsov, N., & Zholudev, V. (2010). Publishing math lecture notes as linked data. *The Semantic Web: Research and Applications*, 370–375.

[9] Kohlhase, M., & Anghelache, R. (2003). Towards collaborative content management and version control for structured mathematical knowledge. *Mathematical Knowledge Management*, 147–161.

[10] Steven, K., Jeremy, A., & Harvey, F. (2009). A language for mathematical knowledge management. *Studies in Logic, Grammar and Rhetoric*, *18(31)*, 51–66.

[11] Iancu, M., Kohlhase, M., & Prodescu, C. (2014). Representing, archiving, and searching the space of

mathematical knowledge. *Mathematical Software–ICMS*, 26–30.

[12] Groza, T., Handschuh, S., Moller, K., & Decker, S. (2007). Salt- ¨ semantically annotated\ mbox {\ LaTeX} for scientific publications. *In The Semantic Web: Research and Applications*, 518–532.

[13] Kohlhase, A., Kohlhase, M., & Lange, C. (2010). Stex+: A system for flexible formalization of linked data. *Proceedings of the 6th International Conference on Semantic Systems*.

[14] M. Kohlhase. (2008). Using latex as a semantic markup format. *Mathematics in Computer Science*, *2(2)*, 279– 304.

[15] Obua, S., Fleuriot, J., Scott, P., & Aspinall, D. (2014). Proofpeer: Collaborative theorem proving. *arXiv preprint* arXiv.

[16] Ring, M., & Luth, C. (2014). Collaborative interactive theorem proving with clide. *Interactive* Theorem *Proving*, 467–482.

[17] Schodl, P., Neumaier, A., Kofler, K., Domes, F., & Schichl, H. (2012). Towards a self-reflective, context-aware semantic representation of mathematical specifications. *Algebraic Modeling Systems*, 11–32.

[18] Neumaier, A. (2010). Fmathl–formal mathematical language. *Proceedings of the 85th Meeting of the GOR Working Group Real World Optimization*.

[19] Mat. Retrieved August 21, 2015, from: http://www.mat.univie.ac.at/~neum/ FMathL.html

[20] Asperti, A., & Coen, C. S. (2010). Some considerations on the usability of interactive provers. *Intelligent Computer Mathematics*, 147–156.

Ittay Weiss received his B.Sc. and the M.Sc. degrees in mathematics from the Hebrew University, Israel, in 2001 and 2003, and his Ph.D. in mathematics from Utrecht University, The Netherlands, in 2007. His main mathematics research interests are in topology and metric space theory, while he is also interested in research on the interface between computers and mathematicians.