REPORT ON OpenDreamKit DELIVERABLE D4.9
In-place computation in active documents (context/computation)

MICHAEL KOHLHASE & TOM WIESING
FAU ERLANGEN-NÜRNBERG
HTTP://KWARC.INFO

Due on	02/28/2017 (Month 18)
Delivered on	02/27/2017
Lead	Jacobs University Bremen (JacobsUni)
Progress on and finalization of this deliverable has been tracked publicly at: `https://github.com/OpenDreamKit/OpenDreamKit/issues/97`	

DELIVERABLE DESCRIPTION, AS TAKEN FROM GITHUB ISSUE #97 ON 2017-02-27

- **WP4:** User Interfaces
- **Lead Institution:** Jacobs University Bremen
- **Due:** 2017-02-28 (month 18)
- **Nature:** Demonstrator
- **Task:** T4.6 (#74)
- **Proposal:** p. 48
- **Final report**

One of the most prominent features of a virtual research environment (VRE) is a unified user interface (UI). There are two complementary approaches that can serve as a basis for OpenDreamKit's mathematical VRE UI: computational notebooks and active structured documents. The former allows for mathematical text around the computation cells of a real-eval-print loop of a mathematical computational system and the latter makes semantically annotated documents active.

In D4.2 "Active/Structured Documents Requirements and existing Solutions" (#91) we reported on two systems in the OpenDreamKit project which follow respectively those two approaches: Jupyter – a notebook server for various computational systems – and MathHub.info – a platform for active mathematical documents. We identified commonalities and differences and developed a vision for integrating their functionalities.

As a first step into this direction we explore in this deliverable the requirements of integrating in-situ (i.e. in-document) computation – a forte and indeed the raison-d'être of notebooks – into conventional, narrative-structured mathematical documents. We present, analyze, and classify examples for in-situ computation and explore – in particular in a MathHub.info based prototype – how the active documents technology has to be extended to accommodate this functionality as a semantic service.

Contents

1. Introduction

In D4.2: "Active/Structured Documents Requirements and existing Solutions", we have proposed "Active Structured Documents" as a natural interface for interacting with mathematical knowledge for the working mathematician, and thus as a UI component for the OpenDreamKit Virtual Math Research Environment Toolkit. In a nutshell, active documents are documents which make aspects of the meaning of their contents explicit enough that it becomes machine-actionable in a document player that delivers services – in our case for computation – that can be triggered.

In [Koh+11] we present a system for active documents following the "active document paradigm" which defines Active Documents as semantically annotated documents associated with a content commons that holds the corresponding background ontologies. An *Active Document Player* embeds user-visible, interactive services like program execution, computation, visualization, navigation, information aggregation and information retrieval to make documents executable. We call this framework the Active Documents Paradigm (ADP; see Figure 1), since documents can also actively adapt to user preferences and environment rather than only executing services upon user request.

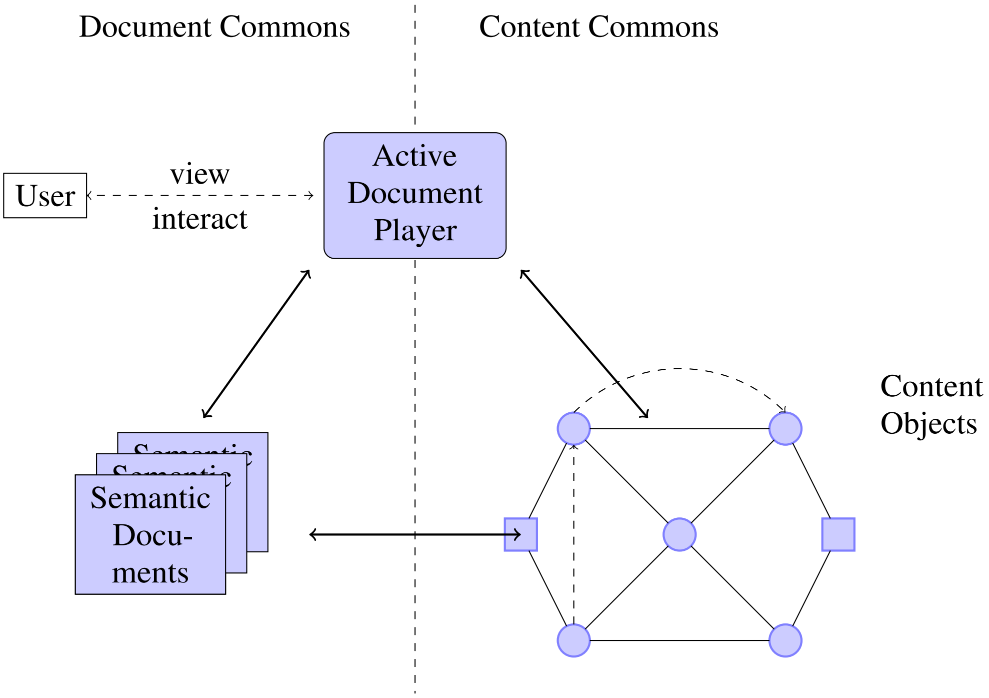


FIGURE 1. Active Documents

The ADP is implemented in the Active Documents Portal MathHub.info [MH] building on standard components as an instance of the Planetary system [Koh12].

In the long run, we propose to integrate active structured documents with the Jupyter notebooks, and as a first step into this direction we explore the requirements of integrating *in-situ* (i.e. in-document) *computation* – a forte and indeed the raison-d'être of notebooks – into conventional, narrative-structured mathematical documents; Section 2 presents, analyzes, and classifies examples for in-situ computation. We also explore how the active documents technology has to be extended to accomodate this functionality as a semantic service – see Section 3 for details. Section 4 details the implementation, and Section 5 concludes the report and gives directions of future research and development.

2. Examples of In-Situ Computation

In the following we will look at some examples to get a feeling for the applications.

2.1. Unit Conversions

Reading documents which contain units that one is not familiar with can be an annoying task, especially when uncommon units, such as solar masses or fortnights, occur in the document or when a document is written in a system of measurement different from the one the reader is used to. Of course, a google query or a similar service can be used to convert units, but this requires the reader to leave the document for another application, which in turn leads to a loss of focus and overall productivity. This can be averted by in-situ computation. Moreover, in contrast to existing unit conversion services, in-situ computation can also convert all expressions in a document – without noticeable effort for the user and without productivity loss.

2.2. Exploring Equations

A common example of in-situ computation is the exploration of mathematical models that are given as equations. In the simplest case, this can be equations like Einstein's mass-energy equivalence (1) – which we will use as a running example – and in other cases, this can be complex models like van Roosbroeck's models for drift and diffusion of electrons and holes in semiconductor devices [Far+16][1], which comprises partial differential equations, boundary conditions, and physical constants – a much more complex situation, but the possible interactions are essentially the same.

So let us come back to our running example: The equation for mass-energy equivalence is simple:

$$E = mc^2\,, \tag{1}$$

where E is the energy, m is the mass, and c is the speed of light. It appears in many documents, e.g. the Wikipedia article in Figure 2. In such a document – were it active – a scholar or interested layman might be interested to see what the energy equivalent of one gram of matter might be. Today a google query reveals a custom-made answer at [Ode], but really our scholar would like to just right click on the symbol m in 1, instantiate it to $1g$ and have the document *simplify the changed expression* (in-situ computation) to give the answer.

Conversely, she might want to know how what mass it would take to drive e.g. from Erlangen to Paris in a Tesla (which gets 6.25 km per kWh)[2]. Here she would like to just instantiate E with $776 \times 6.25 = 4850$ kWh and the document *solves the equation* $4850 = mc^2$ *for* m. Of course, the direct result $m = 4850\ \mathrm{kWh}/(299792458\ \frac{\mathrm{m}}{\mathrm{s}})^2 = 1.942704\ 10^{-7}$ kg is so minuscule that she wants to have it changed form she understand, e.g. the number of carbon atoms that weigh as much.

Of course, the computations themselves in our example are rather simple, and can be executed by any computer algebra system, and even complex examples like the van Roosbroeck models alluded to above would not tax modern systems exceedingly; indeed they are the kind of computations that are often carried out and documented in Jupyter notebooks.

The point here is that the envisioned in-situ computation service allows computation without changing to another system and avoids errors (data entry errors and data interpretation errors) induced by crossing system borders.

[1] We are currently studying this model, formalizing the inherent knowledge and augmenting (parts of) [Far+16] into an active document, see [Koh+] for first results. The methods reported on here will eventually be employed in this case study, which itself is beyond the scope of this deliverable report

[2] This is a natural and common question; see [Rte] which computes the mileage a car would get out of a 1/16 inch drop of water – the value it comes up with is 96.000 miles.

In physics, mass–energy equivalence states that anything having mass has an equivalent amount of energy and vice versa, with these fundamental quantities directly relating to one another by Albert Einstein's famous formula:

$$E = mc^2$$

This formula states that the equivalent energy (E) can be calculated as the mass (m) multiplied by the speed of light (c = about $3 \times 10^8 m/s$) squared.

FIGURE 2. From the Mass-Energy-Equivalence page at Wikipedia [Wik17]

2.3. Hypothetical Computations Playing with Constants

In the previous example, we only explored the equation by instantiating the variables (we are free to do so, since they are 'universal', i.e. the equation holds for all E and m). The 'constant' c is a different beast, it has a globally fixed value: $299\,792\,458$ m/s. In principle, we could instantiate such 'constants' as well, e.g. to answer questions like 'What would the word described in the paper look like if the speed of light were 88mph?' This kind of hypothetical computations are quite common e.g. in Physics, where some of the 'constants' – not the speed of light, but e.g. the proton decay (half-life of protons) – are 'constants' of unknown magnitude (with various theories projecting values between 10^{28} and ∞). Apart from the fact that playing with 'constants' essentially switches between 'alternative worlds', in-situ computation remains as useful as the instantiation of 'universal variables' from the last section.

A variant of the 'constants' case is the case of 'existential variables', which are introduced by declarations like "*for some*". In essence 'existential variables' behave like 'constants' in that they should not be substituted for and have a fixed value – albeit an unknown one. Indeed we can generalize equation (1) to "*There is a velocity c, such that $E = mc^2$.*" without changing much of the physical reality – only that it becomes more permissible to experiment with concrete values.

In fact, often 'constants' are not constant after all, for instance the value $c = 299\,792\,458$ m/s only holds for photons in vacuum, with other values for other media – including values near 88mph in exotic materials like Bose-Einstein condensates. Here, light speed becomes a function of the medium, and we have a "hidden parameter" to the equation, which comes into play in in-situ computation. We will not consider existential variables and functional dependencies in this report and leave the interfaces to future research.

2.4. Updating Values to Current or Historical Values

Additionally to the "computation with hypothetical values" discussed above, we often want to compute with "current or historical values", for time-dependent phenomena. Documents become much more useful if they can adapt values via in-situ computation. Examples uses would be e.g. papers on global warming that adapt to the present state of the art with newer models or data, another is the tongue-in-cheek query "Does it snow in Hell?" to Wolfram alpha [Wol], which interprets it to be a meterological query about the town of Hell, Norway, and the answer varies with the weather there – 0°C and rain at the time of writing this report.

There are many special-purpose documents that can already do that, though the mechanisms are largely special-purpose. Stock ticker widgets are inserted into news web pages, and weather apps supply the "current weather" in travel guides. But none of these allow computation with the values. An exception to this are (some) spreadsheets players that allow linking data cells to live feeds; with this spreadsheets – a well-understood form of special-purpose active documents – can do in-situ computation with live data feeds.

But there is also a case to be made for general documents with in-situ computation: Consider the case of the expression "*five gold doubloons*" used by Adam Smith in an intuition-building example in his seminal book "Wealth of Nations" from 1776 [Smi76]. The example cannot be understood by today's readers, unless they have an intuition about the comparative magnitude of this monetary unit – which was clear to contemporary readers. Indeed the example changes meaning depending on whether "*five gold doubloons*" buy a simple meal, a fine riding horse or a mansion. Here, an in-situ "computation" that explains their relative purchasing power would be extremely helpful for scholars and politicians (who love to quote Smith).

2.5. Computation with Document and Content Structure

In semantic documents we often have access to the dependency relation between document fragments – in the document itself and to other documents and the content commons. This can be used to compute new document structures. A striking example is the computation of guided tours for any concept in the document. A **guided tour** [Koh+11] is a document that is

- **self-contained** – it builds on the (estimated) knowledge of the reader
- **dependency-ordered** – it introduces new concepts only when all prerequisites are already introduced before or assumed to be known
- **goal-directed** – it introduces the goal concept
- **minimal** in some form – in the class of documents satisfying the three aspects above.

Other conditions (e.g. the existence of examples and practice/self-evaluation problems) may be added for special classes of guided tours.

Given an active document in the form described in Figure 1, we can compute guided tours from the document itself (e.g. by adding an "explain" option to the right-click menu of technical terms.)

2.6. Computation with Proofs

Scientific, legal, and policy documents often involve complex argumentations for or against certain statements. In the first categories, the argument can often take the form of – or at least approach – proofs. In the area of study of the OpenDreamKit project – Mathematics – arguments are usually quite literally proofs. These are notoriously difficult to convey, since proofs depend on the user's mathematical literacy [IK15], familiarity with the concepts/facts involved, and the proof techniques. "Proofs" as they are published in mathematic are an attempted best fit to the expected literacy and familiarity of an assumed average reader. In active documents we can (as in the guided tours above) adapt to the "real reader" either pre-emptively or by user interaction. This involves computation – which is best integrated as in-situ computation, e.g.

- calling automated theorem provers on a goal in a document;
- extending the level of explanation by doing that on a subgoal or deepening the level of explanation. E.g. from "obviously" to a full proof.

3. Information Architecture

For the OpenDreamKit project we have developed a general architecture for in-situ computation and tested it on the use cases 2.1 to 2.3 above, which involve computation with math objects – aka. "mathematical formulae"[3] in active documents.

Active Documents are represented in the OMDoc/MMT format [Koh06; MMT; Ian17] which combines document- and content strutures and thus allows to formalize both sides of active documents – the lower part in Figure 1; see D4.2: "Active/Structured Documents Requirements and existing Solutions" for an overview and comparison to other OpenDreamKit systems. The active document player is implemented in the MathHub system [MH] – see D4.3: "Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info" for an introduction in the context of the OpenDreamKit project – which relies on the MMT API [Rab13; MMT] for content/knowledge management services. In particular, the MMT API can communicate with many of the OpenDreamKit systems via the SCSCP remote procedure call standard [Ham+10]; see D3.3: "Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution" for an account of the exact state of affairs. We will use this facility for the actual computations.

To understand the realization of in-situ computation in active documents, we will first look at the information resources involved and only in the next section give go into concrete implementation details. The latter may change over the course of the OpenDreamKit project while the former will only be refined with more experience.

Active documents have **math objects** encoded in MathML [Aus+10]: In OMDoc/MMT-based active documents documents contain presentation MathML and are crosslinked – parallel markup – to Content MathML representations in the content commons; see Figure 1. In any case, the objects which contain identifiers – e.g. E, m, and c in (1), which are introduced (**declared**) in the text or taken for granted because they have definitions in the content commons that is assumed to be known by the reader; see [WGK11] for an experimental account of the linguistic side. These declarations and definitions induce a context that *gives meaning to the math objects* by explaining or binding these identifiers. For in-situ computation this context – we call it the **computation context** – must be identified and passed to the compute engine.

Our computation context is similar in spirit to the document and user contexts Rikko Verrijzer identifies in [Ver15] for MathDox educational documents [Mat]. His notions focus mainly on user modeling whereas ours are technically more generic, since our active documents have more elaborated document and content structures.

To get a feeling for the situation, consider Figure 3 which instantiates the abstract diagram from Figure 1 to the situation in our running example: Einstein's energy/mass equivalence. We have the two parts: the document commons with (a slightly rephrased fragment of) the document in Figure 2 on the left upper corner and the content comment on the other side of the dashed line.

The latter is encoded as an OMDoc/MMT theory graph – see [RK13] for technical details and [CICM1616] and D6.2: "Initial $\mathcal{DKS}$ base Design (including base survey and Requirements Workshop Report)" for an account of the applications in the OpenDreamKit project. All relevant concepts are grouped in named theories (the boxes with rounded corners), which introduce symbols and their properties e.g. the definition of the unit Joule in energy or the size of the speed of light in lightspeed. These theories are connected by theory morphisms – only inclusions $S \hookrightarrow T$ which make all declarations (symbols and properties) of S visible in T – and give an object-oriented, modular regime of formalizing mathematical knowledge.

[3] The other use cases are possible in this architecture. Use case 2.5 has been realized before – on a similar information architecture but different implemenation – and is currently being re-developed for the current version of the MMT API.

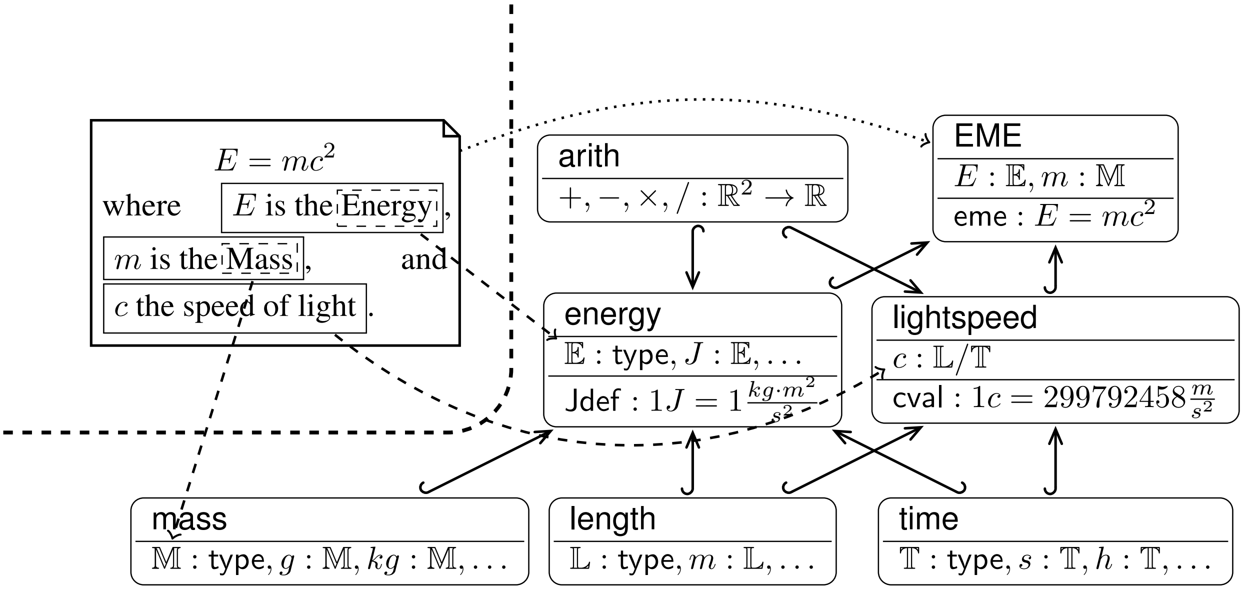

$E = mc^2$
where
E is the Energy,
m is the Mass,
and
c the speed of light.
arith
$+, -, \times, / : \mathbb{R}^2 \to \mathbb{R}$
EME
$E : \mathbb{E}, m : \mathbb{M}$
eme : $E = mc^2$
energy
$\mathbb{E}$: type, $J : \mathbb{E}, \ldots$
Jdef : $1J = 1\frac{kg \cdot m^2}{s^2}$
lightspeed
$c : \mathbb{L}/\mathbb{T}$
cval : $1c = 299792458\frac{m}{s^2}$
mass
$\mathbb{M}$: type, $g : \mathbb{M}, kg : \mathbb{M}, \ldots$
length
$\mathbb{L}$: type, $m : \mathbb{L}, \ldots$
time
$\mathbb{T}$: type, $s : \mathbb{T}, h : \mathbb{T}, \ldots$

FIGURE 3. $E = mc^2$ as an Active Document

Documents are marked up in terms of its document, statement, and phrase structure in the ADP. In particular, we mark up

(1) the sectioning structure – omitted in our running example,
(2) statements – the assertion for $E = mc^2$ coincides with the whole document D in Figure 3, and
(3) the phrase structure – here declarations are shown as boxes and technical terms as dashed boxes.

Finally, the marked up structures in the document commons are cross linked to the content commons to create **parallel markup**[4] at all levels. We see three dashed arrows: two connect the technical terms "Energy" and "Mass" in the dashed boxes to the corresponding concepts in the content tree and one that connects the whole declaration "c is the speed of light" to the corresponding declaration in the theory ligthspeed. The dotted arrow on the top of Figure 3 represents still another parallel alignment relation, it is the "home theory" relation, which makes all concepts from a theory – the **home theory**; here EME – in a document fragment. All parallel markup relations must be refinements of this relation to be well-justified in OMDoc/MMT.

The crucial observation is that together, the home theory relation, its refinements, and the document markup give a notion of context for the computation. The (required) declarations for the local identifiers E and m in D, and the (optional) declaration for the identifier c inherited form the theory lightspeed give meaning to $E = mc^2$ and also determine what can be computed. Instantiating the *variables* (locally declared identifiers) m and E give rise to the computations in Section 2.2, whereas the "replacing" the *constant* (an identifier inherited from a theory) c with a different value the hypothetical calculations from section 2.3. We can even predict the grade of hypotheticality by the inheritance distance in the content theory graph.

Formally, the **computation context** of a formula comes in the form of **declarations**, i.e. triples of the form $c : \tau = \delta$, where c is the name of the declared identifier, τ optionally) its type, and (again optionally) δ its definiens. In our running example, the context of $E = mc^2$ consists of three declarations: $E : \mathbb{E}$, $m : \mathbb{M}$, and $c : \mathbb{L}/\mathbb{T} = 299792458$ (leaving out units for

[4] The idea of "parallel markup" has been pioneered by the MathML format [Aus+10], which uses it to connect equivalent sub-formulae in presentation and content Markup, and [Ian17] generalizes it to all levels in the OMDoc/MMT setting.

the moment). It can be inferred from the information in Figure 3; but let us make the active document fragment more explicit.

Listing 1 shows the HTML5 representation for the active document fragment in Figure 3. It shows the (presentation) MathML representation of $E = mc^2$, followed by the text part in which the declarations are marked up with span elements of class o_declaration for the variable declarations and o_symbol for the symbol declaration. These carry the OMDoc/MMT attributes o:**for** and o:scope. The former points to the identifier being declared, and the latter points at the element in which this declaration is active. In this case, since the scope is the other <p> element the declarations govern the identifiers in the displayed equation $E = mc^2$.

Listing 1. Native Markup for an Active Document Fragment

```
<p id=”p” xmlns:o=”http://omdoc.org/ns”>
 <math display=”block” xref=”http://mathhub.info/ODK/ActiveComputationDemo?EME?EME”>
  <mi>E</mi>
  <mo>=</mo>
  <mrow>
   <mi>m</mi>
   <mo> &#8290;</mo>
   <msup><mi>c</mi><mn>2</mn></msup>
  </mrow>
</math>
 where
 <span class=”o_declaration” o:for=”#E” o:scope=”#p”>
  <math><mi id=”E”>E</mi></math> is the
  <span class=”term” xref=”http://mathhub.info/ODK/ActiveComputationDemo?Energy?Energy”>energy</span>
 </span>,
 <span class=”o_declaration” o:for=”#m” o:scope=”#p”>
  <math><mi id=”m”>m</mi></math> is the
  <span class=”term” xref=”http://mathhub.info/ODK/ActiveComputationDemo?Mass?mass”>mass</span>
 </span>, and
 <span class=”o_symbol” o:for=”#c” o:scope=”#p”>
  <math><mi id=”c” xref=”http://mathhub.info/ODK/ActiveComputationDemo?Lightspeed?c”>c</mi></math> the
  <span class=”term” xref=”http://mathhub.info/ODK/ActiveComputationDemo?Lightspeed?cDef”>speed of light</spa
 </span>.
</p>
```

Listing 2. $E = mc^2$ in Content MathML

```
<constant name=”eme”>
 <type>
  <math>
   <apply><eq/>
    <ci>E</ci>
    <apply><times/>
     <ci>m</ci>
     <apply><power/>
      <csymbol cd=”Lightspeed”>c</csymbol>
      <cn>2</cn>
     </apply>
    </apply>
   </apply>
  </math>
 </type>
</constant>
```

Now let us have a look at how this enables computation: The displayed equation $E = mc^2$ linked into the content commons via its xref attribute, which points to the (constant) declaration in Listing 2. The constant element combines the system name eme with the statement of the equation in the **type** element[5]. As the equation is represented as the "operator tree" in Content MathML, it is fully disambiguated functionally and can therefore directly be computed with in a computational engine (e.g. the MMT system itself or a computer algebra system like GAP or SageMath) after instantiation of the variables with concrete values.

3.1. Information Architecture for Unit Conversion

For the automatic conversion of units, we assume that the document contains formulas in MathML with cross references between its presentational part and its content MathML. The content MathML needs to be annotated with the semantics of the expression, which includes the information about the present units.

[5] This is a consequence of using the Curry/Howard isomorphism at work; we have elided the details of the type here.

4. Implementation

The implementation of in-situ computation is organized along the information model outlined in the last section. It is realized as a set of Javascript modules in the JOBAD framework [JOBAD], a Javascript framework for instrumenting (active) documents with user interactions; see [GLR09] for details.

We will first discuss unit conversion (see Section 2.1) as a special case of in-situ computation and then generalize to the case of a non-trivial context and generalize to arbitrary computations in Section 4.2.

4.1. Unit Conversion

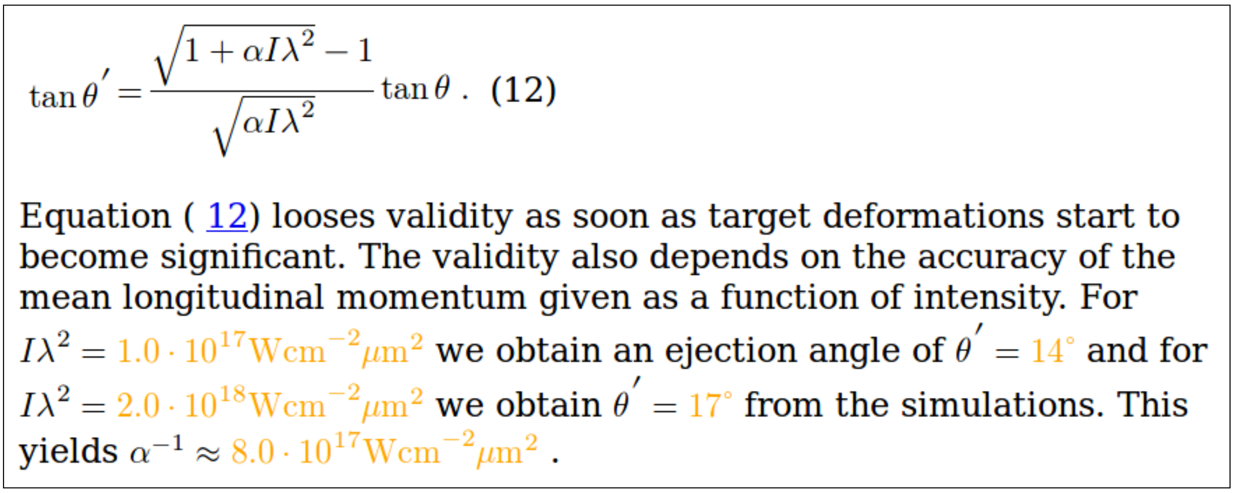
Unit conversion is a special case of in-situ computation, because

- the computations are essentially limited to unit conversion and
- context is trivial, since the computational objects – the quantity expressions – consist only of a number and a unit expression. The units are all defined constants and local identifiers do not exist.

Ulrich Rabenstein of the KWARC group has implemented a semantics extraction procedure that finds quantity expression in HTML5 documents, analyzes their content structure and represents them in content MathML and stored as (standoff) RDF annotations. These can be used to feed in-situ computations.

The user interface for unit conversion can be implemented directly in JOBAD by delegating the conversion (the content MathML representation of the quantity expressions has sufficient information) to a unit converter – we use the units package from Astropy [Ast+13], an extendible python library for astronomy – whose result can be converted to Presentation MathML for inclusion in the document. We show the user interaction here.

Before starting to convert something, the user can highlight all quantity expressions in a given document. This results in a document, as shown in Figure 4. We further use this snippet as an example.



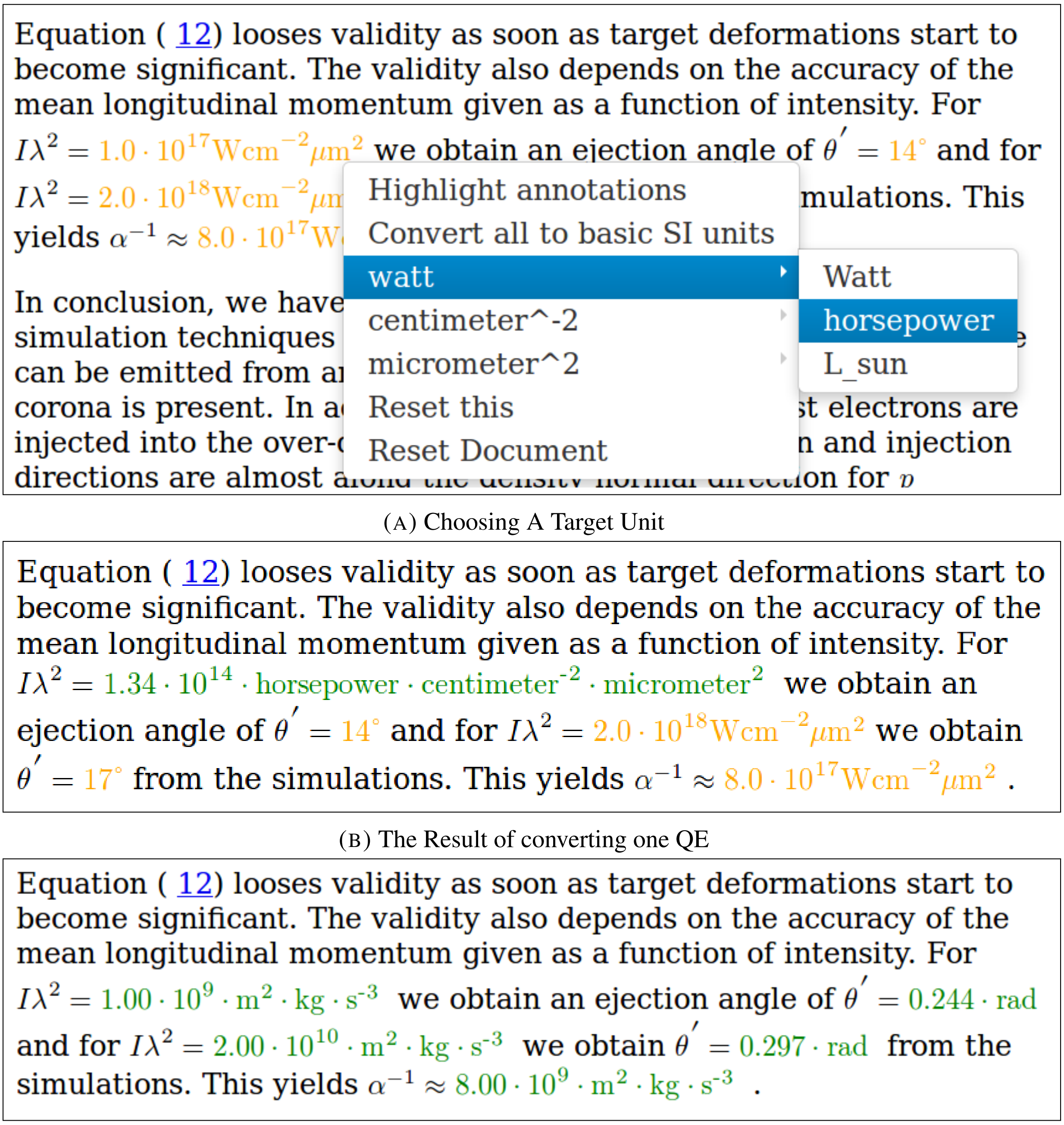
$$\tan\theta' = \frac{\sqrt{1+\alpha I\lambda^2}-1}{\sqrt{\alpha I\lambda^2}}\tan\theta\,. \quad (12)$$

Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain an ejection angle of $\theta' = 14^\circ$ and for $I\lambda^2 = 2.0\cdot 10^{18}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain $\theta' = 17^\circ$ from the simulations. This yields $\alpha^{-1} \approx 8.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$.

FIGURE 4. Highlighting Quantity Expressions in [Ruh+98]

In the first case, the user wants to convert a unit in just one expression to an equivalent one, say watt to horsepower. For that, she can right-click on this particular expression and choose a target unit (e.g. horsepower) from the list of units that are equivalent to Watt. Figure 5a demonstrates this and Figure 5b displays the result of the computation.

The current example only allows local conversions, but of course the user also wants to convert units document-wide – ideally from one system of measurement to another. Figure 5c shows the result of a prototypical implementation, which converts all units to irreducible SI base units. This



Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain an ejection angle of $\theta' = 14^\circ$ and for $I\lambda^2 = 2.0\cdot 10^{18}\mathrm{Wcm}^{-2}\mu\mathrm{m}$ mulations. This yields $\alpha^{-1} \approx 8.0\cdot 10^{17}\mathrm{W}$

In conclusion, we have simulation techniques e can be emitted from a corona is present. In a st electrons are injected into the over- n and injection directions are almost on for ν

(A) Choosing A Target Unit

Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.34\cdot 10^{14}\cdot \mathrm{horsepower}\cdot \mathrm{centimeter}^{-2}\cdot \mathrm{micrometer}^2$ we obtain an ejection angle of $\theta' = 14^\circ$ and for $I\lambda^2 = 2.0\cdot 10^{18}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain $\theta' = 17^\circ$ from the simulations. This yields $\alpha^{-1} \approx 8.0\cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$.

(B) The Result of converting one QE

Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.00\cdot 10^{9}\cdot \mathrm{m}^2\cdot \mathrm{kg}\cdot \mathrm{s}^{-3}$ we obtain an ejection angle of $\theta' = 0.244\cdot \mathrm{rad}$ and for $I\lambda^2 = 2.00\cdot 10^{10}\cdot \mathrm{m}^2\cdot \mathrm{kg}\cdot \mathrm{s}^{-3}$ we obtain $\theta' = 0.297\cdot \mathrm{rad}$ from the simulations. This yields $\alpha^{-1} \approx 8.00\cdot 10^{9}\cdot \mathrm{m}^2\cdot \mathrm{kg}\cdot \mathrm{s}^{-3}$.

(C) Converting a Document To SI

FIGURE 5. In-Situ Unit Conversion

could, for instance, be extended to automatically convert all quantity expressions in a document from imperial to metric units and vice versa.

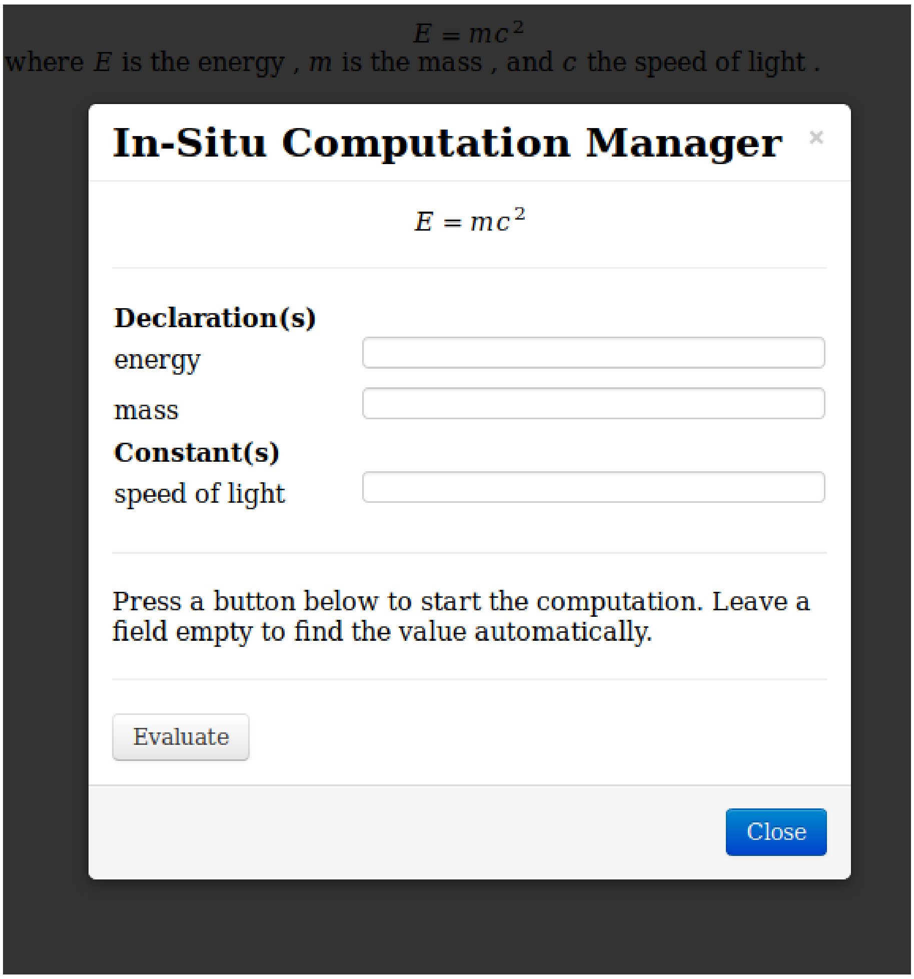
4.2. A General Framework for In-Situ Computation

In addition to the example above, we have also implemented a prototype of the general in-situ computation manager detailed above. A right click on a formula F triggers the JOBAD menu, which has an "Active Computation" field.

- First, the **context extractor**, a function that for all the ci elements in the content formula C associated with F, and tries to find the associated variable declarations by going up the parent chain of F and the symbol declarations from the home theory. Note that using the content MathML representation C of F gets us around disambiguation problems:

even if the presentation of F is ambiguous (e.g. by using variable or constant names multiple times), C is not.

- The variable context is displayed to the user prompting instantiation in a popup form: the **in-situ computation manager** (see Figure 6, which allows to give values for the components of the equation, pick different actions (simplification, equation solving, . . .) and ways of providing the results (in-place, footnote, . . .). As the current system is only a prototype, one can currently only select the Evalutation Action.
- In a second step, the user-supplied values are parsed into content MathML, inserted into C, yielding the content MathML expression C', which is then shipped to the computational engine. Currently we only support the MMT system as a computational engine, but this is not a restriction, since MMT can delegate computations to engines like GAP, Sage, PARI, . . . via the SCSCP protocol [D3.317].
- Finally, the result R of computing C' – a content MathML expression – is inserted back into the original computation context. This context can then be presented in presentation MathML and inserted into the document according to the method the user selected [6].



$E = mc^2$

where E is the energy , m is the mass , and c the speed of light .

In-Situ Computation Manager

$E = mc^2$

Declaration(s)

energy

mass

Constant(s)

speed of light

Press a button below to start the computation. Leave a field empty to find the value automatically.

FIGURE 6. In-Situ Computation Manager

4.3. Code Availability, Licensing and Demos

For both of the examples in this section, an implementation is available under Open Source license terms. For reasons of lacking MathML support in other browsers, we have only tested these demos in Firefox.

[6] Currently, the system presents the user with the computed context directly.

A demo of the unit conversion is available at `http://ash.eecs.jacobs-university.de/`. A user can first select a document and then repeat the procedure detailed above in Section 4.1. The source code of the demo is available in the repository at `https://gl.kwarc.info/urabenstein/Semanticextraction/tree/master/server`. The server can be executed locally following the steps in the corressponding README file.

A protoype of the General Framework for In-Situ Computation is also availble. It can be found at `http://ash.eecs.jacobs-university.de/prototype/` and shows the basics of the process explained in Section 4.2. it consists of a frontend component, the source code of which can be found at `https://gl.mathhub.info/ODK/ActiveComputationDemo`, as well as a backend component inside the MMT system. The source code to the backend component can be found at `https://github.com/UniFormal/MMT/tree/master/src/mmt-odk/src/info/kwarc/mmt/odk/activecomp`.

5. Conclusion and Future Work

We have presented a general framework for in-situ computation in active documents. This is a contribution towards using mathematical documents – the traditional form mathematicians interact with mathematical knowledge and computations – as a user interface for a mathematical virtual research environments. This is also a step towards integrating the two main UI frameworks under investigation in the OpenDreamKit project: Jupyter notebooks and active documents – see D4.2: "Active/Structured Documents Requirements and existing Solutions" – at a conceptual level. The system is prototypical at the moment, but can already be embedded into active documents via a Javascript framework and is ready for use in the OpenDreamKit project. The user interface and SCSCP connections are quite fresh and need substantial testing and optimizations.

In the current state of the system , we have concentrated on in-situ computation with MathML formulae, which covers the first three use cases from Section 2. The main problem with extending this method to the other ones is flexiformalizing the data and knowledge at the document and structure levels and implementing the narration composition engines.

Acknowledgements

This report has profited from discussions with Dennis Müller and Ulrich Rabenstein from the KWARC group in Erlangen. Many of the underlying concepts have evolved in discussions of the first author with Dr. Mihnea Iancu.

References

[Aus+10] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0*. W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: `http://www.w3.org/TR/MathML3`.

[CICM1616] Paul-Olivier Dehaye et al. "Interoperability in the OpenDreamKit Project: The Math-in-the-Middle Approach". In: *Intelligent Computer Mathematics 2016*. Conferences on Intelligent Computer Mathematics. (Bialystok, Poland, July 25–29, 2016). Ed. by Michael Kohlhase et al. LNCS 9791. Springer, 2016. URL: `https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP6/CICM2016/published.pdf`.

[D3.317] Alexander Konovalov et al. *Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution*. Deliverable D3.3. OpenDreamKit, 2017. URL: `https://github.com/OpenDreamKit/OpenDreamKit/raw/master/WP3/D3.3/report-final.pdf`.

[Far+16] Patricio Farrell et al. *Numerical methods for drift-diffusion models*. Tech. rep. To appear in: Handbook of Optoelectronic Device Modeling and Simulation: Lasers, Modulators, Photodetectors, Solar Cells, and Numerical Models — Volume Two. Ed. J. Piprek, Taylor & Francis, 2017. Berlin: WIAS, 2016.

[GLR09] Jana Gičeva, Christoph Lange, and Florian Rabe. "Integrating Web Services into Active Mathematical Documents". In: *MKM/Calculemus Proceedings*. Ed. by Jacques Carette et al. LNAI 5625. Springer Verlag, July 2009, pp. 279–293. ISBN: 978-3-642-02613-3. URL: `https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf`.

[Ham+10] Kevin Hammond et al. "Easy Composition of Symbolic Computation Software: A New Lingua Franca for Symbolic Computation". In: *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. ACM Press, 2010, pp. 339–346.

[Ian17] Mihnea Iancu. "Towards Flexiformal Mathematics". PhD thesis. Bremen, Germany: Jacobs University, 2017.

[IK15] Mihnea Iancu and Michael Kohlhase. "Math Literate Knowledge Management via Induced Material". In: *Intelligent Computer Mathematics 2015*. Conferences on Intelligent Computer Mathematics. (Washington DC, USA, July 13–17, 2015). Ed. by Manfred Kerber et al. LNCS 9150. Springer, 2015, pp. 187–202. ISBN: 978-3-319-20615-8. URL: `http://kwarc.info/kohlhase/papers/cicm15-induced.pdf`.

[JOBAD] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. URL: `https://github.com/KWARC/JOBAD` (visited on 02/18/2012).

[Koh+] Michael Kohlhase et al. *A Case study for Active Documents and Formalization in Math Models: The van Roosbroeck Model*. URL: `https://mathhub.info/MitM/models` (visited on 02/05/2017).

[Koh+11] Michael Kohlhase et al. "The Planetary System: Web 3.0 & Active Documents for STEM". In: *Procedia Computer Science* 4 (2011): *Special issue: Proceedings of the International Conference on Computational Science (ICCS)*. Ed. by Mitsuhisa Sato et al. Finalist at the Executable Paper Grand Challenge, pp. 598–607. DOI: `10.1016/j.procs.2011.04.063`. URL: `http://kwarc.info/kohlhase/papers/epc11.pdf`.

[Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: `http://omdoc.org/pubs/omdoc1.2.pdf`.

[Koh12] Michael Kohlhase. “The Planetary Project: Towards eMath3.0”. In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM). (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 448–452. ISBN: 978-3-642-31373-8. arXiv: `1206.5048 [cs.DL]`.

[Mat] *MathDox – Interactive Mathematics*. URL: `http://www.mathdox.org` (visited on 02/04/2017).

[MH] *MathHub.info: Active Mathematics*. URL: `http://mathhub.info` (visited on 01/28/2014).

[MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. project web site. URL: `https://uniformal.github.io/` (visited on 08/30/2016).

[Ode] Dr. Sten Odenwald. *Special & General Relativity Questions and Answers, How do you actually use Einstein's famous equation E = mc-squared?* URL: `https://einstein.stanford.edu/content/relativity/q388.html` (visited on 02/04/2017).

[Rab13] Florian Rabe. “The MMT API: A Generic MKM System”. In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics. (Bath, UK, July 8–12, 2013). Ed. by Jacques Carette et al. Lecture Notes in Computer Science 7961. Springer, 2013, pp. 339–343. ISBN: 978-3-642-39319-8. DOI: `10.1007/978-3-642-39320-4`.

[RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: `http://kwarc.info/frabe/Research/mmt.pdf`.

[Rte] *Random Thoughts – $E = mc^2$*. URL: `http://2000clicks.com/r/EEqualsMCSquared.htm` (visited on 02/04/2017).

[Ruh+98] H. Ruhl et al. “Collimated electron jets by intense laser beam-plasma surface interaction under oblique incidence”. In: (1998). DOI: `10.1103/PhysRevLett.82.743`. arXiv: `physics/9807021`.

[Smi76] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. W. Strahan and T. Cadell, 1776.

[Ver15] Rikko Verrijzer. “Context in interactive mathematical documents : personalizing mathematics”. PhD thesis. 2015, p. 233. ISBN: 978-90-386-3975-8. URL: `https://pure.tue.nl/ws/files/9193921/20151203_Verrijzer.pdf`.

[WGK11] Magdalena Wolska, Mihai Grigore, and Michael Kohlhase. “Using discourse context to interpret object-denoting mathematical expressions”. In: *Towards Digital Mathematics Library, DML workshop*. Ed. by Petr Sojka. Masaryk University, Brno, 2011, pp. 85–101. URL: `https://svn.kwarc.info/repos/lamapun/doc/DML11/paper.pdf`.

[Wik17] Wikipedia. *Mass–energy equivalence — Wikipedia, The Free Encyclopedia*. 2017. URL: `\url{https://en.wikipedia.org/w/index.php?title=Mass%E2%80%93energy_equivalence&oldid=763358012}` (visited on 02/04/2017).

[Wol] *Wolfram—Alpha*. URL: `http://www.wolframalpha.com` (visited on 01/05/2013).

[Ast+13] Astropy Collaboration et al. “Astropy: A community Python package for astronomy”. In: *Astronomy & Astrophysics* Volume 558, A33 (Oct. 2013). http://www.astropy.org, A33. DOI: `10.1051/0004-6361/201322068`. arXiv: `1307.6212 [astro-ph.IM]`.

Disclaimer: this report, together with its annexes and the reports for the earlier deliverables, is self contained for auditing and reviewing purposes. Hyperlinks to external resources are meant as a convenience for casual readers wishing to follow our progress; such links have been checked for correctness at the time of submission of the deliverable, but there is no guarantee implied that they will remain valid.