Story based content structuring in MATh

Michael Junk
University of Constance
78457 Konstanz, Germany
michael.junk@uni-konstanz.de

Sebastian Sahli
University of Constance
78457 Konstanz, Germany
sebastian.sahli@uni-konstanz.de

Abstract

Within the MATh project, we develop explanation strategies to help first year students adapt to unacquainted workflows in university mathematics. These strategies are designed and implemented as sets of rules which are closely tied to the common practice in first year courses. In order to make the rules easily accessible to beginners it is useful to align them with commonly known concepts. Notably, the common experience with creating, continuing and reusing stories in varying situations turns out to be a promising starting point. We present a story-based approach for generating and structuring mathematical content within the MATh language and demonstrate that it entails various mathematical notions like theorem, set, function, theory and model. Due to this intimate relation between stories as structuring constructs and associated mathematical objects, the approach differs from other systems.

1 Stories

According to Wulff [7], the human species is the story telling animal – in the form of stories, we store and retrieve experiences and stories are a central element in human communication. As participants of the communication process, we rarely question its underlying rules and mechanisms but it turns out that quite sophisticated models are necessary to offer explanations (see, for example, [8]). In the end, pinning down the concept of narrative seems to be quite difficult [9, 10]: as the term narrative is so widespread, its meaning is diffuse and a generally accepted definition is not readily available. Nevertheless, it is undeniable that important aspects of stories are to make life experience known, to transport meaning and to offer solution strategies to problems. This is also reflected in structural aspects of stories: a story typically begins by introducing the main characters and their basic situations. Within this model world, a certain problem drives the story forward, it leads to a development of the characters and ends in some sort of resolution of the problem.

Being fundamental units of communication, it is clear that stories are also essential in the mathematical discourse. Of course, due to the nature of the subject, the characters of mathematical stories are abstract objects which constitute abstract situations in which development (due to lack of time in the mathematical world) relies on deductive reasoning to disclose an increasing number of relations among the objects while the text proceeds. In the end, we can *use* a mathematical story due to the same reason we can use any other story: it is the ability to rediscover a story pattern in different situations which share the basic structure (fables and parables strongly depend on this principle).

Another peculiarity of mathematical communication is the concurrence of different language levels. Natural language is used on a meta level to talk about the design and structure of mathematical texts which may actually

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

be written in a purely formal language. Accordingly, stories appear on both levels but in this article, we focus on the role of stories in purely formal texts.

Our motivation to consider this aspect evolved out of the MATh project [1, 2, 3, 4] which summarizes different activities to help first year students adapt to characteristic workflows in university mathematics. In particular, we try to address questions which are independent of specific content and which are known to cause severe problems for beginners: How to get similarly acquainted to mathematical objects as to real objects? What is a good mental model of a variable? How to mentally organize changing goals and intermediate assumptions in proofs? How to deal with the patterns provided by definitions of notions, sets and functions? How to access components of theories in concrete cases?

Such questions are regularly addressed in various first year teaching classes at the University of Constance: A preparatory course for beginning math majors, an alternative study program during the first year with a focus on mathematical techniques and a lecture on mathematical modeling in the second semester. From the point of view of story structure, these courses just address writing the essential components: initializing stories is precisely the subject of the modeling lecture, where problem descriptions have to be translated from meta to formal level while moving a story forward relies on a good understanding of proof mechanisms which are taught in the other courses. In order to accompany and illustrate the overall process, we are currently designing and implementing an authoring tool for writing mathematical stories in the context of first year courses. The latest version of this tool is regularly used to support the modeling lecture. The experiences gained here drive the development of the tool and the MATh syntax. This way, the tool is closely tied to the common practice in beginner courses.

The formal MATh language employed in this tool acts as a model to illustrate and to exercise certain mechanical aspects of the mathematical workflow. By including the story concept into MATh, the important (and very frequent) context switching in mathematical texts is represented explicitly with syntactical constructs. In general, the MATh language has been developed to help students understand the role of formality in creating comprehensibility and objectivity while staying very close to the object-, proof-, and structuring concepts used in classical lectures. The language is only used exemplary and not continuously throughout the lectures as we do *not* aim at a complete formalization of math education.

In particular, we consider MATh texts as formal skeletons (and not substitutes!) of the less formal mathematical "stories" introduced in traditional lectures which address many other levels apart from the naked bones. As a condensed formal summary of the involved mathematical objects and their interplay, the skeleton is useful to formulate and explain general rules of "defining", "proving" and "structuring" mathematical content.

Our goal is reached when students have developed a clear idea of the formal underpinning of mathematics because this perspective is important to understand why mathematical ideas and strategies are developed and presented in the way they are. By particularly stressing the role of stories in this approach, we hope to obtain a representation of mathematical content and functioning in a form which is also systematically transferable to earlier phases of education. Since operating with stories is a very basic qualification, math education can start from this point and successively carve out the distinguishing features of *mathematical* stories as being very precise in their wording and causal structure with described patterns that are profoundly reusable within other stories. Along with the growing awareness of this careful interplay of stories, formal notions and mathematical symbols may be introduced gradually as references to already established components of the story network.

In order to demonstrate the use of stories, we continue as follows: First we present a simple mathematical model to give an impression how formal stories appear in MATh. Next, we briefly describe how basic mathematical concepts like variables, quantified expressions, sets and functions naturally arise in connection with stories by stripping of specific language details and just concentrating on the role of stories. In the following sections, we present more examples of usages of the story concept in the current MATh syntax. In the final section, we comment on the appearance of the story concept in other formalizations of mathematics.

2 Models as stories

Creating mathematical models from a problem statement amounts to the introduction of characters in the initial phase of a story. Working in the model then leads to a development towards the resolution of the problem. In order to illustrate the concept with a toy problem, we translate the following logical riddle to MATh. Here is the situation: *"Alice claims that Bob is a liar. Bob says: Claire lies. Claire asserts that Alice and Bob are liars. Who of the three is lying?"* In order to capture this initialization phase in a formal text (in MATh such texts are called (mental) *frames*), we require names like `A,B,C` to refer to the persons and by coining the notion `person` formally, we can refer to all names at once (note that the colon indicates the membership relation).

```
frame riddle
  given A, B, C : element;
  person := {A,B,C};
```

Next, we want to express that each person claims something about claims of other persons and we would call a person a liar, if he or she claims something which is false. Denoting the claim of a person `p` by `claim(p)`, the function `claim` maps elements of `person` to propositions which are treated as representations of truth values (elements of `Bool`) in MATh.

```
  given claim : person --> Bool;
```

The collection of liars can then be introduced as the set

```
  liar := {p:person with claim(p)=false};
```

Using these linguistic elements, the actual starting point can now be stated in a rather natural form which completes the definition of the initial state of the frame.

```
  claim(A) = (B:liar);
  claim(B) = (C:liar);
  claim(C) = (A:liar)/\(B:liar);
end
```

Subsequently, the story should be driven to a point where the precise relation between the elements `A,B,C` and the property `liar` is disclosed. The required deductive reasoning *within* the frame is started by extending the frame dynamically allowing additional definitions of names and proofs of statements (here, however, we skip proof steps for conciseness which is indicated by the trailing `.. claimed` expressions; of course, complete proofs must not contain such steps).

```
extend riddle
  truthTeller := person\liar;
  forall p:truthTeller holds claim(p) .. claimed;
  B:truthTeller .. claimed;
  C:liar .. claimed;
  A:liar .. claimed;
end;
```

While these internal conclusions show that Alice and Claire are lying it does not rule out that the whole story is flawed by hidden contradictions among the assumptions so that a consistent `claim`-function may not exist. In this case, however, no collection of elements *outside* of the model could satisfy the model conditions in place of `A,B,C` and `claim` provided the outside is free of contradictions. Such a consistency check shows that frames also have a natural outside perspective in which they appear as a predicate or a set to separate fitting from non-fitting element combinations. These interrelations between stories and other well known mathematical objects are developed more generally in the following section.

3 Starting with stories

Obviously, stories can only be formulated if a language is available and since stories come as lists of statements about some kind of objects it is clear that grammatic rules for the formation of statements and objects are required which in turn need specific signifiers to indicate different statement and object configurations. Working in this way, stories can be deconstructed into elementary constituents from which they can later be rebuilt. Here, however, we voluntarily keep the more elementary parts unspecified and put the story as a whole in the focus of the exposition. In order to introduce important notions and concepts, we mimic an axiomatic approach with the following stipulations.

(S1) Stories are named *linguistic items* created within a parent story. They consist of a sequence of *statements* interlaced with name declarations for linguistic items.

According to (S1), stories naturally form a tree structure: if A is the name of the root story and B is the (local) name of a story in A then the concatenation $A.B$ can be used as a unique global reference. Similarly, if B contains a local name N, the corresponding global name is $A.B.N$. Since each story S is based on a lists L of statements and declarations we will write $S = \mathbf{story}(L)$ for the purpose of this section.

(S2) Statements are linguistic items which express relations between linguistic items. They are *facts* if they are specifically related to prior facts of the surrounding story.

(S3) Linguistic items are *wellformed* if they are specifically related to prior facts and items of the surrounding story.

In practice, the derivation of new facts is regulated by some proof calculus and whether an expression is wellformed depends on grammatic and maybe semantic rules. Here we just use the consequences to classify stories.

(S4) A story S is *factual* if all its statements are facts and all its named items are wellformed (we write $\mathbf{factual}(S)$ for this). Otherwise S is called *fictional*.

(S5) A fictional story $S = \mathbf{story}(L)$ is *realistic* (we write $\mathbf{realistic}(S)$) if it is factual under its *hypothesis* which is a leading part $\mathbf{hyp}(S)$ of the list L whose statements are treated as facts and whose items are assumed to be wellformed. Otherwise S is called *phantastic*.

(S6) Names declared in $\mathbf{hyp}(S)$ which do not refer to specific linguistic items are called *parameters*. It is possible to replace parameters by well-defined linguistic items from the parent story.

From this classification three basic types of stories emerge which are all realized in MATh. The factual stories are called *paragraphs*. Since a paragraph has an empty hypothesis, it just encapsulates facts and proper definitions within its parent story. Paragraphs are used to structure the name space of a story and allow for aside considerations, similar to info boxes inside a newspaper article.

Next, realistic stories without parameters are called *conditions*. The hypothesis of a condition just restricts the objects of the parent story further and thus enables the investigation of special cases (for example the *abelian* condition in the *group* story is based on the commutativity hypothesis on the group operation). To state that a condition is satisfied, we use $\mathbf{satisfied}(S)$ as an abbreviation for $\mathbf{factual}(\mathbf{story}(\mathbf{hyp}(S)))$. In this case, the whole story turns out to be factual and can thus be viewed as a paragraph.

Finally, realistic stories with parameters are called *frames* (the parameters of the frame in section 2 are introduced with the `given` command). It is to be noted that the parameters are *constants* within the story while they appear as *variables* when the story is used metaphorically from outside. In other words, being a variable is not a property of an object alone but it is a property of an object *and* a perspective. In contrast, if mathematical variables are presented as being variable by themselves, the blurred relation to the well-accustomed concept of story parameters may explain some of the conceptual difficulties pupils encounter in connection with the introduction of variables in mathematics.

When we replace the (tuple of) parameters x of a story S by a tuple y of expressions from its parent story, the frame is transformed into a condition which we denote S_y. If $\mathbf{satisfied}(S_y)$ is a fact, the parameters y are compatible with the hypothesis of S and we call y an example of S (written as $\mathbf{isExample}(y, S)$).

Before we draw conclusions from this basic classification of stories, we want to finish the list by a final rule which stresses that stories are considered as dynamic entities.

(S7) Stories can be continued.

Taking into account that stories are quite intricate objects, it is not too surprising that they code a lot of well known mathematical concepts. For example the basic logical connectives of conjunction and implication can be recovered in the following form: if A and B are statements, the conjunction statement can be formulated as $\mathbf{factual}(\mathbf{story}([A, B]))$ while the implication relates to $\mathbf{realistic}(S)$ with the condition $S = \mathbf{story}([A, B])$ satisfying $\mathbf{hyp}(S) = [A]$.

As already indicated, every frame S gives rise to a *predicate* $\mathbf{isExample}(y, S)$ for linguistic items y in the parent story which could also be interpreted as *set* of all examples of S. If an example y of S is used to replace the parameters of S, every local name N in S turns into $(S_y).N$ which is well-defined in the parent story. In other words, every declared name in a frame gives rise to a *function* in the parent story whose domain is given by the examples of the frame. Similarly, every fact $S.F$ turns into a fact in the parent story once the parameters of S are replaced by an example y. Thus $S.F$ can be viewed as a *theorem* and $(S_y).F$ as its application to y. If S contains a large number of facts, it can be viewed as a *theory* (a collection of many theorems under the same premise).

Finally, universal quantification over some statement E depending on a parameter x is obtained in the form $\mathbf{realistic}(S)$ where x is declared as a parameter in $\mathbf{hyp}(S)$ and E is the only statement in the conclusion part of S.

While all our considerations so far apply to possible roles of stories relative to their parent story, similar relations hold between stories that are further apart in the tree structure. The crucial observation is that

a nested story like $S.T$ again appears with one of the basic story types. For example, if S and T are both paragraphs, then $S.T$ contains only facts and proper declarations when considered from the parent of S. Thus, $S.T$ behaves overall like a paragraph in this case. Similarly, a condition S combined with a paragraph appears like a single condition and any combination of a frame with some other story type remains frame-like. Applying these considerations recursively, the access to stories which are nested downwards is clarified. Similarly, sideward access is possible for example from $A.B.C$ to $A.D.E$. As the parameters of A are shared by both stories (provided A is a frame), we only have to provide parameters or check conditions of D and E if required to get access to the named components of $A.D.E$.

Apart from the obvious usage of stories to structure mathematical content there are some further applications which are detailed in the following sections.

4 Theories as stories

In section 2 we have seen that frames are natural candidates to formulate mathematical models as renarrations of non-mathematical problem statements. In fact, many mathematical theories are exactly such models albeit their origins may be disguised by abstraction. When working with theories, students are often unaware of the clear distinction between the internal perspective for theory extension and the external perspective for theory usage which requires instantiation of the parameters. Here, the formal syntax helps because it uncovers steps which are traditionally treated rather implicitly.

As example, we consider the first steps in the theory of *general topology* which starts with the story of a topological space based on a set `X` together with a family of so called *open* subsets of `X`. In the corresponding frame, several set theoretical notions are used which are defined earlier on. The set operations `pow` (power set), `union` (union of an indexed set family) and `dom` (the domain of a function) are primitive statements of 𝕄ATh. For illustration purposes, we place the definition of a topological space inside a paragraph.

```
paragraph generalTopology
  frame space
    given X:set;
    given open c= pow(X);
    emptyset:open; X:open;
    forall F:mapsTo(open) holds union(F):open;
    forall F:mapsTo(open) with dom(F):finiteSet holds sec(F):open;
  end;
end;
```

Based on this hypothesis many subsequent notions can be defined and studied. To illustrate such an extension we present the definitions of closed sets (as complements of open sets in `X`) and general neighborhoods of points in `X` accompanied in each case with a simple theorem (the notation `{f(x) | x:M}` with a vertical bar refers to the set of values `f(x)` with `x` ranging over `M` as opposed to the set `{x:M with C}` which contains all elements of `M` which satisfy `C`).

```
extend generalTopology.space
  closed := {C c= X with (X\C):open};
  forall O:open holds (X\O):closed .. directly;
    conclude X\(X\O)=O .. see basicSetAlgebra;
    compress (X\O):closed;
  qed;

  nbhood(x:X) := {U | U c= X, O:open with x:O; O c= U};
  forall x:X, U,V:nbhood(x) holds sec(U,V):nbhood(x) .. claimed;
end;
```

In order to check syntax and proofs in practice, the 𝕄ATh interpreter would be manually invoked in order to process the current input file as a whole. In doing this, the justification `.. see basicSetAlgebra` indicates that the theorem from which the conclusion is drawn can be found in the paragraph `basicSetAlgebra` defined in the root story (such hints to the surrounding story of a theorem instead of an explicit reference by name or number are common practice in mathematics). In the second theorem, we shorten our presentation by employing the pseudo justification `.. claimed` which simply accepts the preceeding statement as being valid. Considering university mathematics from a formal point of view, this is by far the most frequently used justification.

In order to show how theories are used in other stories, we present a simple example which introduces the discrete topology on the natural numbers.

```
paragraph example1
  dis := (Nat, { {n} | n:Nat});
  dis:generalTopology.space .. claimed;
  forall n:Nat holds {n}:generalTopology.space.closed(dis) .. claimed;
end
```

In order to express that the singleton `{n}` is closed with respect to the discrete topology on `Nat`, we have to mention the story and component names together with the assignment of the story parameters. Since `space` has two parameters, we provide a pair `dis` which satisfies the hypothesis of the story when replacing the parameters in their order of appearance (the statement `y:S` is the syntactic form of **isExample**(y, S) in MATh). While referencing with the fully qualified name is reasonable in case of a single access, it becomes quite awkward if several references to the same story with the same parameter assignment are used. The following extension of the example takes care of this point.

```
extend example1
  use generalTopology.space with dis;
  Nat:closed .. claimed;
  forall n:Nat holds Nat:nbhood(n) .. claimed;
end
```

With the provided information in the `use` command, the expression `nbhood(n)` can be completed to `generalTopology.space.nbhood(discrete)(n)` automatically. However, if notions are simultaneously used for different parameter assignments the binding is dropped as in the following example which uses the discrete and the trivial topology.

```
paragraph example2
  dis := (Nat , { {n} | n:Nat});
  triv := (Nat, {emptyset, Nat});
  dis:generalTopology.space .. claimed;
  triv:generalTopology.space .. claimed;
  use generalTopology.space;
  forall n:Nat holds {n}:closed(dis) .. claimed;
  forall n:Nat holds not({n}:closed(triv)) .. claimed;
end
```

The use of such name completion schemes simulates the common practice in mathematics where notions are adopted from defining stories as long as the use is unambiguous (otherwise parameter references are added to the ambiguous notions to restore uniqueness). However, the parameter binding often has to be read off indirectly from comments on the meta level. A typical question type used in oral examinations shows that awareness for such naming conventions is counted among the basic qualifications. In the present context, one such questions could be, whether `{1}` is a closed set in the natural numbers. Since the parameters of the topological space are not specified, the correct answer should be: this cannot be answered if the open sets are not specified - it would be true for the discrete but false for the trivial topology, for example. Quite often, however, the answer is simply *yes* because students fill the missing parameter with the most usual choice in undergraduate mathematics which is the discrete topology. To avoid such mistakes, working with a formal system could raise the awareness for the need of parameter specifications when referring into the name space of frames.

In order to demonstrate the use of conditions, we consider an extension of the topological space by an additional separation axiom known under the name `T0`.

```
extend generalTopology.space
  condition T0
    forall x,y:X with not(x=y) holds
      exists O:open with (x:O; not(y:O));
  end;
end;
```

If conclusions are derived under this condition they can only be used if the condition is satisfied (similarly, access to named components of `T0` is limited by the condition). In MATh we use the convention that the name of a conditions also points to a `Bool` value which represents the truth value of the hypothesis. For the first example it can then be shown that `T0` holds.

5 Proof steps as stories

While stories naturally appear when *defining* structures, they also appear on a smaller scale when *proving* single statements about or within structures. Consider for example the proof from the topological space example above:

```
forall O:open holds (X\O):closed .. directly;
  conclude X\(X\O)=O .. see basicSetAlgebra;
  compress (X\O):closed;
qed
```

In this *direct proof* (initiated by the `.. directly` justification), the indented proof statements can be seen as the extension of a (nameless) frame built from the `forall`-statement, where `O:open` is the parameter and no additional assumptions hold. The proof succeeds if the conclusion `(X\O):closed` is true when reaching the corresponding `qed` command.

The same applies for example, for proofs *by contradiction*, which have the following form: `assume not(A); [statements] hence A.` The command `assume` generates a condition with `not(A)` as hypothesis while the provided statements are considered as an extension. The proof succeeds if a contradiction exists before the closing command. In a similiar way, other proof steps of Gentzen's calculus of natural deduction can be described in terms of a story and one or more subgoals which need to be satisfied after the story has been suitably extended by the user.

6 Background stories

While story definitions give rise to canonically associated objects, also object definitions may be accompanied by corresponding stories. As an example we consider the definition of the limit of a sequence. Consisting of four nested quantifications, however, the meaning of the classical convergence condition is difficult to grasp for beginners and quite some time has to be devoted to the explanation of the concept by giving geometric meaning to each quantification. In other words, the definition of the limit is a *longer story*. This structural aspect can be visualized and accentuated with the syntactic form of the definition in MATh. In the approach presented here, the starting point is the geometric idea of the limit while the quadruple quantification is derived subsequently as an equivalent condition.

We start with a sequence `a` and first detour into the story of a `simple` decreasing and non-negative sequence. Here, the infimum is the natural limit which can later be accessed with the function `simple.lim` in the surrounding story.

```
lim for convergent.lim from frame
  given a:sequence;
extended by
  frame simple
    given b:sequence;
    b:decreasing;
    forall n:Nat holds b(n) geq 0;
  extended by
    lim := inf(img(b));
  end;
```

Looking back at the sequence `a` we now compute the radius of the circle around a given number `A` which contains all sequence members starting from some index `n`. This radius happens to shrink as `n` increases, i.e. it is a simple sequence. In particular, if the radius shrinks to 0, the sequence is closing in on `A` so that `A` becomes a limit *candidate*.

```
radius(A:R)(n:Nat) := sup({abs(a(k)-A) | k:Nat with k geq n});
forall A:R holds radius(A):simple .. claimed;
candidate := {A:R with simple.lim(radius(A)) = 0};
```

We see here, that the MATh text reproduces only the skeleton of the accompanying more figurative description in natural language by providing precise definitions of all involved mathematical objects and facts. In particular, it should not be considered as a substitute of the more prosaic version which is much better in transmitting the basic idea. It is rather a summary in full precision that also captures the original story structure. Note that this structure is not completely linear in this case due to the side story related to simple sequences.

Continuing in the definition, we would now show uniqueness of candidates. After that, it all depends on the existence of candidates to come to *the* limit of a sequence. This is done in the conditional story `convergent` and

precisely from this sub-story, the value of the function `lim` is taken.

```
    !A:candidate .. claimed;
    condition convergent
      exists A:candidate;
    extended by
      lim := the A:candidate;
    end;
  end;
```

While this definition captures the limit construction in conceivable geometric terms, it is not well suited for proving convergence of concrete sequences. Therefore, the derivation of the well known ϵ-characterization would be the next step. This proof naturally splits into parts associated to the subconcepts `simple` and `convergent` so that extending these stories is very natural. Finally, the full characterization is stated and proved as an extension of the main story `lim`.

Of course, from a mathematical point of view, the same steps can be carried out without structuring them in stories and sub-stories. But there are practical and didactical advantages in using stories. For example, the notions `simple`, `radius` and `candidate` are used to illustrate the underlying idea but experience shows that consequent proofs are shortened with the characterization of convergence. In such a case, mentioning the geometric ideas stays important but the short-lived notions should not clutter the name space of the surrounding story. Similarly, the uniqueness of the limit is a crucial step towards the definition but will not be referenced afterwards. Encapsulation therefore relieves the name and fact management of the bigger story while keeping the more specific results and constructions available.

Secondly, the structuring seems to resemble the natural way of thinking about problems. Whenever a construction requires more thought, we fix the ingredients as given for the whole process which alleviates the definition of auxiliary objects and facts, because the ingredients can be used without the need of passing them as arguments. Once the construction is finished, the dependence on the *general assumptions* (a term that is frequently encountered before lengthy constructions) turns all auxiliary objects into functions of the ingredients.

7 Interfaces as stories

The notation `{5,42,11,5}` is well established to denote a particular finite set whose meaning depends on the given data `t=(5,42,11,5)`. More specifically, if the tuple `t` is considered as a function on the numbers $1,\dots,4$, then `{5,42,11,5}` is just the image set of `t`. In MATh this relation would be written as

```
{5,42,11,5} = img(5,42,11,5);
```

Since the construction *removes* information related to the frequency and ordering of the entries, the set expression contains more information than the signified set. This may lead to problems in formal proofs compared to informal ones. For example, the statement that `42` is an element of the set is *evident* on an informal level because we can *point* to `42` as second entry. If we want to copy this strategy in a formal proof, pointing turns into the statement `42=t(2)` about the tuple which, however, is not accessible to us. In order to enable such access to the inner structure of basic MATh expressions, predefined stories are available which describe the components and their interplay in a general way. For example, `expression.enumerationSet` comprises the names `indexing` and `object` where `object = img(indexing)` holds and the syntax rules ensure that indexing refers to an explicit tuple. The above mentioned argument can now be carried out in the form

```
42:{5,42,11,5} .. see expression.enumerationSet with 42=indexing(2);
```

where the explanation contains enough information to detect a corresponding theorem in the mentioned story. Employing the usual extension of stories, additional results can be added. For example, if basic results on the cardinality of finite sets are already available, a result on the upper estimate can easily be attached:

```
extend expression.enumerationSet
  card(object) <= length(indexing) .. claimed;
end
```

Subsequently, this enables justifications like

```
card({5,42,11,5}) <= 4 .. see expression.enumerationSet;
```

In a similar way, extensions of the stories to basic relations and logical connectives allow formulations of alternative proof techniques which can be evoked in the justification part of a corresponding proof step.

Another important story which acts like an interface is the root of the story tree. It provides initially available objects, stories and facts as background for all further stories and therefore appears like a summary of the story behind the language itself. Before listing essential components of this summary we want to give some comments on the developments which have led to the current approach. From the very beginning, one of the important driving forces was the demand for a simple model description language to be used by first year students in the lecture on mathematical modeling. Since our underlying concept of a model is quite broad (it matches the concept of a *theory* in [5] as a structure comprising of a finite number of abstract mathematical objects whose interplay emerges from a list of axioms) also set theory in the form of ZF, for example, should just appear as one among many possible models. This approach, however rules out that the language sticks to one particular theory from the beginning. On the other hand, as has been pointed out in section 3, telling stories about objects naturally leads to the appearance of functions and predicates (which can also be interpreted as naive sets) and it does not take long that such derived linguistic items become objects of other stories. These considerations eventually led to the following design principles (showing that the availability of basic set operations and the ability to talk about objects essentially coincide):

- The objects of MATh stories are called (naive) elements.
- To allow stories about properties of elements, (naive) sets of elements are considered as elements. Sets must be subsets of other sets.
- To allow stories about actions on elements, (naive) functions from sets into sets are considered as elements.
- Tuples are considered as functions on initial segments of the natural numbers. To this end, the natural numbers are considered as a set.
- To allow stories about truth, `true` and `false` are elements.
- Stories about the set of all properties of elements from a given set should be possible. This introduces power sets of sets as elements.
- Stories about the collection of all elements from several sets should be possible. This introduces the union of a set of sets as elements.
- Talking about the set of functions between two sets should be possible. This introduces corresponding function sets as elements.

Extending these rules by some inductive properties of the natural numbers, all explicit set theoretic constructions (like the extension to other number ranges) can be carried out. However, it would still not be possible to tell stories which apply to general collections of elements like the theories of metric or topological spaces because story parameter must always be limited by some already given set (to avoid Russell-type paradoxes). In order to allow such general stories, a collection of naive elements is provided in the root story which is large enough to contain the "usual" sets encountered in a normal math curriculum but small enough to still be usable as a naive set. It is denoted `element` and the axiomatic rules ensure that it is closed under the basic set operations (similar to a Grothendieck universe). Since `Nat0` is contained in `element`, all "usual sets" constructed from the natural numbers are captured. Altogether, the setup of the root story ensures the following relations for `element` which refer to the derived notions `set` and `function`:

```
set := {x:element with x c= element};
Nat0:set; Bool:set;
forall A:set holds (pow(A) c= set; pow(A):set);
forall A,B:set holds (A --> B):set;
function := union(A --> B for A,B:set);
forall A:set holds (A --> element) c= function;
forall S:function with img(S) c= set holds union(S):set;
```

While this approach definitely suffices to run the modeling lecture and all basic courses, it will reach its limit once collections bigger than sets are considered as objects of other stories. For example, `set` is a naive set (as subset of `element`) but it is not contained in `set` itself so that the discrete topology on it could not be considered as an example of `generalTopology.space` in section 4.

Of course, one could rewrite the theory for larger sets like `pow(element)` but apart from being an almost literal copy it would not prevent the problem for even larger sets like `pow(pow(element))`.

A possible way out of this dilemma uses the observation, that `element` is a perfect universe to talk about `Nat0` and `Bool` and all its linguistic consequences as it contains these sets and is adequately closed under set operations. However, if we start talking about collections which are as big or bigger than `element`, we enter a new linguistic universe which behaves like another instance of the root story with the only difference that it also contains `element` as a set in the narrow sense. As a consequence, all facts and declarations of the root story can be adoped to the bigger universe.

To distinguish the different variants of the declarations, we add a tilde symbol as prefix so that `~element` is the universe to talk about `element` and all its linguistic consequences as it contains this set and is adequately closed under set operations. Similarly, `~~element` would be the universe to talk about `~element` and so on. In this way, the discrete topology on all sets can be formulated in the following way:

```
paragraph example3
  dis := (set, { {S} | S:set});
  dis:~generalTopology.space .. claimed;
  forall S:set holds {S}:~generalTopology.space.closed(dis) .. claimed;
end
```

Altogether, the introduction of a restricted selection of elements allows to formulate general stories and in combination with the idea of meta universes, these stories are applicable to arbitrary naive sets provided they are considered at a suitable level in the hierarchy of universes.

8 Stories in other formalizations

In the previous sections we have shown that the fundamental story concept is a frequently recurring pattern in the mathematical workflow and its integration in the MATh system has been demonstrated. In the remaining part of the paper, we want to investigate how the story aspect is treated in other approaches which deal with the formal representation of mathematical knowledge, ranging from automatic and interactive theorem provers (like Isabelle[12], IMPS[15], Coq[13] and Mizar[11]) to general knowledge representation frameworks like MMT[14].

What almost all the different systems have in common, is that they allow grouping knowledge into (mostly named and possibly nested) modules. In [6], such *module systems*[1] and their possible properties are described in a very general way together with actual realizations in concrete systems. Therefore [6] constitutes a perfect basis for comparing MATh to other languages in terms of their realization of story-based concepts. In the following, we strongly refer to the terms and concepts mentioned in this paper. However, comparing MATh to all other languages in terms of all aspects mentioned in [6] would go beyond the scope of this paper. Thus, we only do a comparison in terms of some key features of common module systems.

The modules have different names in different languages and slightly differ in their characteristics. Their basic ideas, however, remain the same across different systems. For now, we want to refer to modules as *theories*, as they are called for example in IMPS and MMT.

Theories can be seen as the story equivalent in module systems. A theory usually consists of a list of *declarations*. From a story-based point of view, a declaration can be seen as the introduction of a new named linguistic item. In a declaration, names that have been declared earlier can be used so that a declaration can also express relations between linguistic items. Both properties conform to the structure of a story as described in section 3.

From a module system's point of view, a MATh-story also consists of a list of (not necessarily named) declarations. This is justified by mathematical practice, where one usually doesn't want to name *every* statement or proof.

In MATh, the continuation property of stories is covered by the extension mechanism. It allows to re-enter an already started story and continue it by adding conclusions using the declarations made in the earlier part of the story. The concept that comes closest to this continuation mechanism in other module system are *imports*. An import allows to add all (or some) declarations of one theory to another one. Importing a story `S` into another one and using the imported declarations to create new declarations can be interpreted as continuing the original story.

The other way round, the extension of a story `S` in MATh can be considered as the definition of a new story which immediatly imports `S`. In contrast to other systems, this newly created story is *unnamed*. This reflects common mathematical practice. When proving a conclusion in an already defined concept, it is common to

[1] In [6], a module system is defined as "*formal language that provides constructs to express high-level design patterns such as namespaces, imports, parametricity, encapsulation, etc.*"

	MATh	Isabelle	IMPS	MMT
story	story	locale	theory	theory
continuation	extension	import	import	structure
external use	example	interpretation	interpretation	view

Figure 1: Some important story concepts and their closest counterparts in different systems

refer to this extended concept by the same name. As a drawback, this approach enforces a lot of implicit import handling. When refering to a story `S`, we in fact refer to an implicitly constructed story which imports all extentsion of `S`, which are currently in scope. This again means, that an extension of `S` imports all other extensions of `S` currently available. Since all these extensions are unnamed, this procedure raises a lot of potential name clashes which are resolved based on qualified names using the file names of the respective extension as qualifiers. This approach strongly emphazises the *dynamic aspect* of creating stories, whereas in other systems stories are rather static objects (but allow a more finely grained access to different versions of the same story).

Maybe the most important aspect of mathematical stories is their external reusability. One of the most natural mathematical actions is to apply some abstract situation to a concrete one (that's what abstract definitions are made for), e.g. by applying a theorem to some arguments or constructing a concrete topological space and exploit its theorems.

To realize this kind of external applications of stories, there are different mechanisms available in different systems. One very universal approach is the *view* (as called in MMT) or *interpretation* (as called in Isabelle or IMPS[16]). A view from a theory `S` into a theory `T` maps the declarations of `S` to expressions over `T` while preserving some structural properties. This view then is a toplevel object itself and can be used to access the mapped declarations of `S`. Speaking in terms of stories, a view is similar to rewriting a story with new characters instead of the old ones and storing the result as a new story.

In MATh, views don't exist as independent concepts, but are implicitly available. Consider a story `T`, in which we declare some list of objects `y` which satisfies `y:S`. Then `y` induces a view from `S` to `T` which in the declaration of `S` maps (recursively) all parameters of `S` to the entries of `y`. This view cannot be accessed as a whole, but if `N` is a name declared in `S` the declaration by the view can be accessed by `S.N(y)`. This reflects mathematical practice, where views are usually not called by a name, but occur implicitly all the time. In conclusion, we see that several story-based language concepts of MATh overlap with established and well understood approaches for structuring mathematical content (see also Figure 1). Despite this overlap, there are also notable differences. According to [17], most systems support either stratified or integrated grouping constructs. In that respect, our approach appears to deviate since object and module level are strongly intertwined in MATh where, generically, each object is related to a story and each story gives rise to an associated object.

At the same time, the story-based approach is very useful to explain recurring patterns in mathematical practice to beginners. Since stories are closely connected to the concepts of variables, sets, functions and theories, and since working in story networks highlights the important (and very frequent) context switching in mathematical texts, the concept may serve as a fundamental building block in math education which could be used (with varying degree of formality) at all levels of math education.

Finally, the dynamic structure of stories in MATh enables a sequential and incremental generation of theory networks which can be organized similar to standard text books: The main explanatory route is supplemented by sideline considerations which are developed as they are needed to a degree which just suffices to continue the main argument. Such mutually depending developments of theories are important for beginners as they illustrate the use of auxiliary structures and results while working on a more specific goal.

In our opinion, the versatility of the story concept is very interesting and deserves further investigations.

References

[1] MATh-Homepage, `http://www.mmath.de/en`.

[2] Junk, M., Hölle, S., Sahli, S.: *Formalized Mathematical Content in Lecture Notes on Modelling and Analysis*. In: Rabe, F., Farmer, W. M., Passmore, G. O. and Youssef, A. (eds.) Intelligent Computer Mathematics. Lecture Notes in Computer Science, vol. 11006, pp. 125-130. Springer (2018)

[3] Junk, M., Hölle, S., Sahli, S.: *A Meta Language for Mathematical Reasoning.* In: Osman Hasan, O., Kaliszyk, C., Naumowicz, A. (eds.) Proceedings of the Workshop Formal Mathematics for Mathematicians (FMM), Hagenberg, Austria (2018)

[4] Junk, M., Sahli, S.: *How Natural is Formal Mathematics?* NFM 2020 – Workshop on Natural Formal Mathematics (2020).
https://cicm-conference.org/2020/NFM/paper_2_Junk_Sahli.pdf

[5] A. Tarski. *Introduction to Logic and to the Methodology of Deductive Sciences.* Oxford University Press, 1941.

[6] Rabe F., Kohlhase, M.: *A Scalable Module System.* Information and Computation, **230**, 1-54 (2013).

[7] Wulff, H. J.: *Life is made up of stories.* Televizion, **26**, 4-7 (2013).

[8] Basten, F.: *Going Narrative: But Where Will It Take Us?* Narrative Works: Issues, Ivestigations, & Interventions, 2(1), 48-66 (2012).

[9] Schiff, B.: *The Function of Narrative: Toward a Narrative Psychology of Meaning.* Narrative Works: Issues, Ivestigations, & Interventions, 2(1), 33-47 (2012).

[10] Hyvärinen, M.: *Revisiting the narrative turns.* Life Writing, 7(1), 69-82 (2010).

[11] Mizar-Homepage, http://www.mizar.org/.

[12] Isabelle-Homepage, https://isabelle.in.tum.de/.

[13] Coq-Homepage, https://coq.inria.fr/ .

[14] MMT-Homepage, https://uniformal.github.io/.

[15] Farmer, W.M., Guttman, J.D., Thayer, F.J.: *IMPS: An interactive mathematical proof system.* J Autom Reasoning 11, 213–248 (1993). https://doi.org/10.1007/BF00881906.

[16] Farmer, W.M., Guttman, J.D., Thayer, F.J.: *Little theories.* In: Kapur D. (eds) Automated Deduction—CADE-11. CADE 1992. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 607. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-55602-8_192.

[17] Müller D., Rabe F., Kohlhase M.: *Theories as Types.* In: Galmiche D., Schulz S., Sebastiani R. (eds) Automated Reasoning. IJCAR 2018. Lecture Notes in Computer Science, vol 10900.