Theorems in ConTEXt

Abstract
This article explains some of the recent advancements in ConTEXt enumeration mechanism that handles most of the requirements of theorem-like constructions.

Keywords
ConTEXt, theorems, enumerations

Introduction

In mathematics writing important results are usually presented in such a way that they stand out from the surrounding text and can be referenced later on. Theorems, lemmas, corollaries, and proofs are most common examples. In ConTEXt, enumerations can be used to define such theorem-like environments and this article explains how to do that.

The basics

A typical theorem looks like this.

Theorem 1 (Pythagoras Theorem) *The square on the hypotenuse is equal to the sum of the squares on the other two sides.*

It consists of a name (theorem), a number (in this case 1), an optional title (Pythagoras Theorem), and a body. We want to be able to configure the blank space before and after the theorem, the style of the name, number, title, and the body. Some attributes are shared amongst all theorem-like environments, others are specific to a particular environment. All common features can be set up using `\setupenumerations`. For example, in this document I have set

```
\setupenumerations
  [ before={\blank[big]},
    after={\blank[big]},
    location=serried,
    width=broad,
    distance=0.5em,
    headstyle=bold,
    titlestyle=bold,
    way=bytext,
    conversion=numbers]
```

The `before` and `after` keys are for setting up what goes before and after the enumerations. In this example, we use it to set up the blank spaces; later on I will show an example to use these keys to do fancy stuff. The `location` key sets up where the head and the title will be located. There are various options which are documented in the ConTEXt manuals. I use `serried` with `width` set to `broad` and `distance` equal to `0.5em`. This gives the typical style in which theorems are typeset. The keys `headstyle` and `titlestyle` set the style for head (theorem name and number) and title. There are also `headcolor` and `titlecolor` to set up colors. If you really want to fine tune the appearances of head and title, you can use the `headcommand` and `titlecommand` keys to use a custom command. The

`way` key configures the numbering; we can use `bytext`, `bychapter`, `bysection`, `bypage`, etc. The `conversion` key sets up the numbering; we can use `Characters`, `numbers`, `Roman`, or a customized conversion.

Once we have set up the style common for all enumerations, we can set up individual theorem-like environments.

```
\defineenumeration
  [theorem]
  [    text=Theorem,
      title=yes,
      style=italic,
       list=all,
   listtext={Theorem }]
```

This defines an environment (`\starttheorem ...\stoptheorem`) with the name "Theorem", an optional title and italic body style. It also sets a list `all` to store theorems, with the `listtext` as "Theorem " (notice the space). This enables us to place a list of theorems later on. This environment can be used as

```
\starttheorem[thm:pythagoras]{Pythagoras Theorem}
  The square on the hypotenuse is equal to
  the sum of the squares on the other two sides.
\stoptheorem
```

The `\starttheorem` command takes two optional arguments. The first optional argument is in square brackets and sets up the key for referencing: we can use `\in[thm:pythagoras]` to get the number of the theorem. The second argument is in curly brackets and sets the title of the theorem; the title is typeset according to `titlestyle` and `titlecolor` keys and is surrounded by `titleleft` and `titleright`, which by default are set to (and). If we want a theorem without a title, we can leave out the second optional argument. For example, to get this

Theorem 2 *The square on the hypotenuse is equal to the sum of the squares on the other two sides.*

we type

```
\starttheorem
  The square on the hypotenuse is equal to the sum of the squares
  on the other two sides.
\stoptheorem
```

We can also set up the proof environment using enumerations. A typical proof looks like

Proof I have discovered a truly marvelous proof of this, which this margin is too narrow to contain. □

Normally proofs do not have a number or a title, and have a symbol □ at the end to indicate the end of the proof. Proofs can be set up as

```
\defineenumeration
  [proof]
  [       text=Proof,
        number=no,
     headstyle=italic,
         title=no, %this is the default
   closesymbol={\mathematics{\square}},
         style=normal]
```

The number=no makes the proofs not numbered, the closesymbol key sets the close symbol (□) which is placed at the end of the environment and is pushed to the right edge of the line. The current algorithm for placing the close symbol is fairly simple and does not give the correct result in all cases. Sometimes (e.g., when the proof ends with an itemize environment) one has to manually tell ConTEXt where to place the close symbol by putting a \placeclosesymbol command at the appropriate place. This command is equivalent to the \qedhere command of the amsthm package in LaTEX. The closesymbolcommand key can be used to set a customized command to place the close symbol. The above proof was keyed in as

```
\startproof
  I have discovered a truly marvelous proof of this, which this margin
  is too narrow to contain.
\stopproof
```

Sharing numbers

Suppose we want to define a corollary environment, which shares the same number as theorem. So, if we key in

```
\startcorollary
  The sum of angles of a quadrilateral is $360^{\circ}$.
\stopcorollary
```

we will get

Corollary 3 The sum of angles of a quadrilateral is 360°.

In order to share the numbering with theorems, we need to define corollary as

```
\defineenumeration
  [corollary]
  [    text=Corollary,
     number=theorem,
       list=all,
   listtext={Corollary }]
```

The number=theorem key tells ConTEXt to use the same number for theorems and corollaries. Notice that the number key is smart: number=no means that the enumeration will not be numbered, number=theorem means that the enumeration will have the same number as theorems.

Framed and shaded theorems

In LaTEX, ntheorem package provides a means to get shaded and framed theorems. In ConTEXt, the enumerations provide enough hooks to make this possible. Suppose we want all axioms to be framed with a random frame. First let us define the fancy text background

```
\definetextbackground
    [axiomframe]
    [           mp=background:random,
          location=paragraph,
     rulethickness=1pt,
             width=broad,
        leftoffset=1em,
       rightoffset=1em,
            before={\testpage[3]\blank[3*big]},
             after={\blank[3*big]}
    ]
```

where background:random is defined as

```
\startuseMPgraphic{background:random}
   path p;
   for i = 1 upto nofmultipars :
    p = (multipars[i]
     topenlarged 10pt
     bottomenlarged 10pt) randomized 4pt ;
   fill p withcolor lightgray ;
   draw p withcolor \MPvar{linecolor}
    withpen pencircle scaled \MPvar{linewidth};
   endfor;
\stopuseMPgraphic
```

Now, we want hook this frame to the definition of an axiom. We can use

```
\defineenumeration
   [axiom]
   [    text=Axiom,
      before={\startaxiomframe},
       after={\stopaxiomframe},
       title=yes,
     stopper=,
    location=top,
        list=all,
    listtext={Axiom }]
```

after which

```
\startaxiom {Playfair's axiom}
   Exactly one line can be drawn through any point not on a given line
   parallel to the given line.
\stopaxiom
```

gives

Axiom 1 (Playfair's axiom)

Exactly one line can be drawn through any point not on a given line parallel to the given line.

We can use all the fancy features of framed texts to get fancy backgrounds for enumerations.

Sharing styles

Suppose we spent a lot of effort in defining a fancy style for axioms. Now, suppose we want to define two new enumerations, postulate and proposition, with the same style. We want postulate and axioms to share numbering, but we want propositions to be numbered on their own. We could copy the entire style of axiom for defining postulates and propositions, but ConTEXt provides an easier method. We can define postulates as

```
\defineenumeration
  [postulate]
  [axiom]
  [    text=Postulate,
```

```
      list=all,
  listtext={Postulate }]
```

and then use `\startpostulate ...\stoppostulate` to get

Postulate 2 (Parallel Postulate)

If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two right angles, then the two lines inevitably must intersect each other on that side if extended far enough.

Notice that postulates have the same style as axioms, and they also share the numbering with axioms. Now, if we do not want to share the number, but still want to share the style we can say

```
\defineenumeration
  [proposition]
  [axiom]
  [    text=Proposition,
     number=proposition,
       list=all,
   listtext={Proposition }]
```

and then use `\startproposition ...\stopproposition` to get

Proposition 1

To construct an equilateral triangle on a given finite straight line.

Notice that the proposition has got a number of its own. We could have also used the `number` key to share the number with an already defined enumeration.

List of things

Often we want to display a list of theorems. For example, here is a list of all theorems used in this article.

We get this by keying in

```
\placelist[enumeration:all][width=6em,criterium=all]
```

Observe that we had used `list=all` in all the enumerations we defined. We also set the `listtext` to appropriate strings. The list of theorems in stored in `enumeration:all` list, and can be recalled by `\placelist`. The `criterium=all` is needed so that we can place the theorems from all sections, not just from the current section.

Conclusion

Enumerations take care of most of the features needed for typesetting theorems. There are a few things that need improvement. It would be nice to have an automated placement of close symbol that works correctly with formulas and itemizations. The `ntheorem` package in LaTeX shows how it can be done using a two pass mechanism. I personally am not too happy with the way lists work; it would be nice if the list text was equal to text by default (rather than the name of the enumeration which starts with a lowercase letter) and list width was calculated automatically.

Aditya Mahajan
adityam@umich.edu