Template @ DIKU, v. 1.1

Datalogisk institut, Copenhagen University (DIKU)
LaTeX

Simon Wiknther
zlp616@alumni.ku.dk

Juni 26, 2023.

Contents

1 Math extensions

The math exetensions were inspired as I was making my way through [Concrete Mathematics], and various courses at DIKU. It seems to me that these macros are useful in general.

1.1 Groups

Very often, mathematical expressions make use of grouping constructs such as $\lceil\rceil$, $\lfloor\rfloor$, $()$, etc. These constructs are relatively easy to use in LaTeX(with the `amsmath` package), despite the fact that one has to often distinguish between the left and right connectives, as with e.g. `\lfoor` and `\rfloor`. What makes these groups particularly impractical however, is that the height of the connectives is not automatically adjusted to the content they enclose. To this end, one may resort to using the commands `\left` and `\right`, as respective connective prefixes... Yuk! This lead to the specification of the following macros:

`\ceil{group}`	$\lceil \mathtt{group} \rceil$	
`\floor{group}`	$\lfloor \mathtt{group} \rfloor$	
`\set{group}`	$\{\mathtt{group}\}$	
`\seq{group}`	$[\mathtt{group}]$	(as in, *sequence*)
`\card{group}`	$\lvert\mathtt{group}\rvert$	(as in, *cardinality*)
`\chev{group}`	$\langle\mathtt{group}\rangle$	(as in, *chevrons*)
`\p{group}`	$(\mathtt{group})$	(as in, *parenstheses*)
`\st{group}`	$\mid \mathtt{group}$	(as in, *such that*)

1.2 Backus-Naur Form

`\nonterm{group}`	`<group>`
`\term{group}`	`‘group’`

1.3 Cormen

MERGE-SORT(A, p, r)

if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT(A, p, q)
    MERGE-SORT$(A, q + 1, r)$
    MERGE(A, p, q, r)

2 Box extensions

I've used ‘`tcolorbox`’ to create a bunch of different boxes that can be used throughout the program. Look in the ‘`boxes.tex`’ file, to see how they're defined and used.
TODO: MAYBE WRITE SOMETHING ABOUT THE SHOWCASEBOXCOMMAND

2.1 Sample Syntax Demonstrations

Here we will demonstrate how the syntax box works and is used.

Without example usage:

Command used:

- \newsyntax {<*title*>} {<*label*>} {<*syntax file*>}
  - **title**: `'The title of the syntax box'`
  - **label**: `'the unique identifier for cross-referencing'`
  - **syntax file**: `'the file path to the syntax definition file'`

Syntax §2.1: Syntax for creating function pointer in C without usage

```
<return type> (*<pointer variable name>)(<parameters>);
```

With example usage:

Command used:

- \newsyntax {<*title*>} {<*label*>} {<*syntax file*>} {<*output file*>} {<*output lang*>}
  - **title**: `'The title of the syntax box'`
  - **label**: `'the unique identifier for cross-referencing'`
  - **syntax file**: `'the file path to the syntax definition file'`
  - **output file**: `'the file path to the output example file'`
  - **output lang**: `'the programming language used in the output example.'`

Syntax §2.2: Syntax for creating function pointer in C with usage

```
<return type> (*<pointer variable name>)(<parameters>);
```

Example Usage:

```c
#include <stdio.h>
int sum(int a, int b){
    return a + b;
}
int main() {
    int (*functionPointer)(int, int); // Declare the function pointer
    functionPointer = &sum; // Point the function pointer to the sum function
    int result = functionPointer(5, 6); // Now you can use the function pointer to call sum
    printf("The sum is %d", result);
    return 0;
}
```

2.2 Sample Code Demonstrations

These boxes allow you to present various types of code snippets, whether they produce an output or not. We've showcased this flexibility with C and Java code examples. This framework isn't limited to just these languages - if '`lstlisting`' doesn't include your desired language, you have the freedom to create your own set. It's a versatile way to enhance code visibility and comprehension.

2.2.1 C and Java Code Example (No Output)

Command used:

- \newlisting {<*title*>} {<*label*>} {<*code language*>} {<*code file*>}
  - **title**: '`The title of the code listing`'
  - **label**: '`the unique identifier for cross-referencing`'
  - **code language**: '`the programming language of the code`'
  - **code file**: '`the file path to the source code file`'

Code §2.1: A '`C`' code example *without* output

```c
#include <stdio.h>
void fun(int a)
{
    printf("Value of a is %d\n", a);
}

int main()
{
    void (*fun_ptr)(int) = &fun;
    (*fun_ptr)(10);
    return 0;
}
```

Code §2.2: A 'Java' code example *without* output

```java
@FunctionalInterface
interface Printer {
    void print(String msg);
}

public class Main {
    public static void main(String[] args) {
        Printer printer = (String msg) -> {
            System.out.println(msg);
        };

        printer.print("Hello, world!");
    }
}
```

2.2.2 C Code Example (Includes Output)

Command used:

- \newlisting {<*title*>} {<*label*>} {<*code language*>} {<*code file*>} {<*example file*>}
  - **title**: `'The title of the code listing'`
  - **label**: `'the unique identifier for cross-referencing'`
  - **code language**: `'the programming language of the code'`
  - **code file**: `'the file path to the source code file'`
  - **example file**: `'the file path to an example output or use of the code.'`

Code §2.3: A 'C' code example *with* output

```c
#include <stdio.h>
void fun(int a)
{
    printf("Value of a is %d\n", a);
}

int main()
{
    void (*fun_ptr)(int) = &fun;
    (*fun_ptr)(10);
    return 0;
}
```

Example Output:

```
Value of a is 10
```

2.3 Sample Lemma Demonstrations

Command used:

- \mlenma {<*title*>} {<*label*>} {<*content*>}
  - **title**: `'The title of the lemma'`
  - **label**: `'The unique identifier for cross-referencing'`
  - **content**: `'The content of the lemma'`

Lemma §2.1 Arithmetic Sum Lemma

The sum S of an arithmetic sequence with first term a, common difference d, and n terms is given by: $S = \frac{n}{2}[2a + (n-1)d]$.

Proof: Adding the sequence $a, a+d, \ldots, a+(n-1)d$ to its reverse yields $2S = n[2a + (n-1)d]$. Thus, $S = \frac{n}{2}[2a + (n-1)d]$. QED.

2.4 Sample Definition Demonstrations

Command used:

- \newdfn {<*title*>} {<*label*>} {<*content*>}
  - **title**: `'The title of the definition'`
  - **label**: `'The unique identifier for cross-referencing'`
  - **content**: `'The detailed explanation or description'`

Definition §2.1 VSpace

A vector space (V, F) over a field F (usually $\mathbb{R}$ or $\mathbb{C}$) is a set V along with two operations, vector addition and scalar multiplication, satisfying the following axioms:

1. $\forall \mathbf{u}, \mathbf{v} \in V$, $\mathbf{u} + \mathbf{v} \in V$.
2. $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$, $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$.
3. $\exists \mathbf{0} \in V$ such that $\forall \mathbf{u} \in V$, $\mathbf{u} + \mathbf{0} = \mathbf{u}$.
4. $\forall \mathbf{u} \in V$, $\exists \mathbf{v} \in V$ such that $\mathbf{u} + \mathbf{v} = \mathbf{0}$.
5. $\forall \mathbf{u} \in V$ and $\forall a, b \in F$, $a \cdot (b \cdot \mathbf{u}) = (a \cdot b) \cdot \mathbf{u}$.
6. $\forall \mathbf{u} \in V$ and $\forall a \in F$, $a \cdot \mathbf{u} \in V$.

2.5 Sample Table Demonstrations

2.6 Making References to the Boxes

Soon to come... When I am not too lazy...

3 References

References

[Concrete Mathematics] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. 1994, 2nd ed. Addison-Wesley Longman Publishing Co., Inc. Boston, USA. ISBN 0201558025.