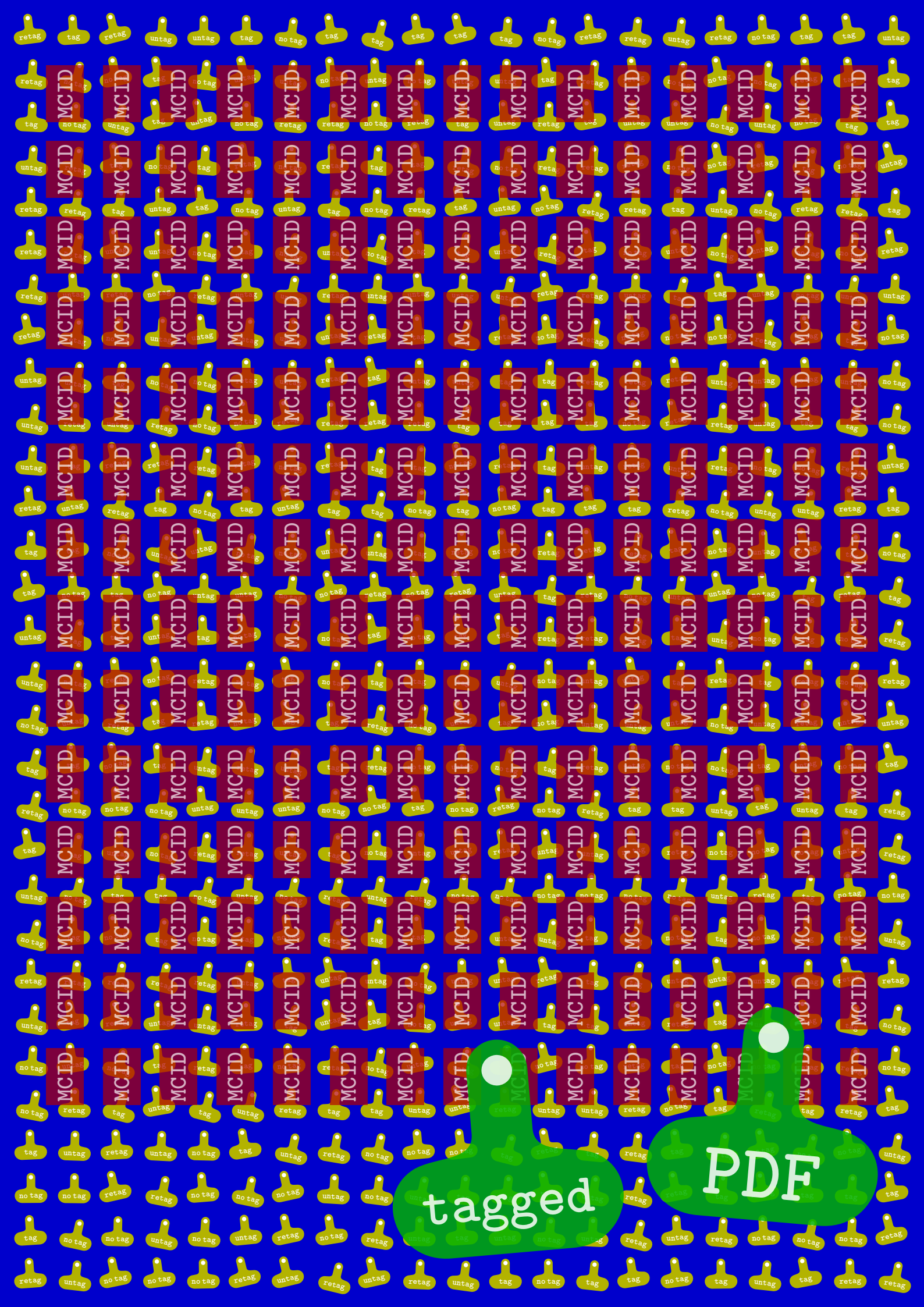
MCID
tag
untag
retag
no tag
tagged
PDF

1 Why do we tag

Around 2010 tagged pdf showed up in ConTEXt. Apart from demonstrating that it could be done it served little purpose because only full Acrobat could show a structure tree and in the more than a decade afterwards no other viewer did something with it. However for some users it was a necessity.

In 2024 we picked up on tagging because due to regulations (especially in higher education) demands for tagged pdf in the perspective of accessibility popped up. We will not go into details here but just mention that we want to make sure that users can meet these demands.

As of now (2024) we have little expectations when it comes to tagging. The ongoing discussions about how to tag, how to interpret the specification, what to validate, and what to expect from applications are likely to go on for a while, so the best we can do is keep an eye on it and adapt when needed. If we have opinions, these will be exposed in other documents (and articles).

This manual is work in progress!

Hans Hagen
Mikael Sundqvist

2 Tagging text

As mentioned in the introduction, we need to satisfy validators that are imposed on those working in education (often via web interfaces with little information on what actually gets checked, it's business after all). It is not that hard to fool them and make documents compliant, so that is what we can do anyway. It is also possible to let these tools do some auto tagging but our experiments showed that this is a disaster. So, we end up with a mix of relatively rich tagging that we feel good with. When we're a decade down the road we expect that with a little help from large language models a decent verbose tagging is better than a crappy suboptimal one.

One reason for tagging is that it could permit extraction but there are better solutions to that: if there is something shown in a table or graphic, why not add the dataset. We currently add MathML and bibTEX blobs but more can become possible in the future (this also depends on user demand).

Another application is reflow but when that is needed, why not go html or distribute different output. When accessibility is the target one has to wait till more is clear how that is actually supposed to work. Often the recommendations are to use Arial, little color, simple sectioning etc, so that gives little reason to use pdf at all.

All that said, we assume that pdf level 2 is used, if only because it looks like validators aim for that. Also, if you find pre level 2 documents produced elsewhere, often tagging is so bad or weird that one can as well ignore it.

Tagging in a document is enabled with:

```
\setupbackend[format=pdf/ua-2]
\setuptagging[state=start]
```

The first command ensures that the right data ends up in the pdf file, and the second one enables tagging. As long as you're working on a document you can comment these commands which saves you some runtime and give way smaller files.[1]

[1] With `\enabledirectives[backend.usetags=crap]` you can map to old built-in ua-1 tag names but that will fail with version 2 validation because it has a simplified (or restricted) model.

3 Tagging math

Tagging math at level 2 is still experimental but works as follows. Instead of tagging the atoms and structures, as we do in level 1, we generate a MathML attachment and put a so called actual text on the math structure node. This text can be spoken by reading machinery. The MathML is not that rich but we can enable more detail when needed. However, given the way (presentational) MathML evolved we are somewhat pessimistic. Instead of adding a few more elements that would help to provide structure, some features are dropped. Also, support in browsers comes and goes, either native or depending on JavaScript.

Because there is much freedom in how mathematical symbols and constructs are used, you might need to help math tagging bit. The process is driven by group sets that refer to domains. An example of a domain is chemistry. For now we just mention that this features is there and as time flies by we can expect more granular usage.

```
\definemathgroupset
  [mydomain]
  [every] % a list of dictionaries

\setmathgroupset
  [mydomain]
```

For now you can ignore these commands because we default to `every`.

Todo: list all possible dictionaries.

You can control the tagger by specifying what symbols and characters actually mean, for instance:

```
\registermathfunction[f]
\registermathfunction[g]

% \registermathsymbol[default][en][u][the vector]
% \registermathsymbol[default][en][v][the vector]
% \registermathsymbol[default][en][A][the matrix]

\registermathsymbol[default][en][lowercasebold]            [the vector] % [of]
\registermathsymbol[default][en][uppercasesansserifnormal][the matrix]
```

From the language tag being used here you can deduce that this can be done per language.

You can trace math translations with:

```
\setupnote[mathnote][location=page]
\enabletrackers[math.textblobs]
```

which is what we used when developing these features. In a few hundred page math book one easily gets thousands of notes.

In `examples-mathmeanings` you can find a lot of examples. In due time we expect to offer more translations. The English and Swedish are for now the benchmark.[2] Likely other languages will be served by Tomáš Hala as result of courses on typesetting. Feel free to contact all those involved in this.

[2] As a proof of concept, at BachoTEX 2024, the Ukrain translations were provided by Team Odessa, but they need some tuning.

4 Tracing

Todo