

Polyfilling Accessible Chemistry Diagrams

Sorge, Volker

DOI:

[10.1007/978-3-319-41264-1_6](https://doi.org/10.1007/978-3-319-41264-1_6)

License:

None: All rights reserved

Document Version

Peer reviewed version

Citation for published version (Harvard):

Sorge, V 2016, Polyfilling Accessible Chemistry Diagrams. in K Miesenberger, C Bühler & P Penaz (eds), *Computers Helping People with Special Needs: 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part I*. vol. 9758, Lecture Notes in Computer Science, vol. 9758, Springer, pp. 43-50, 15th International Conference on Computers Helping People with Special Needs (ICCHP 2016), Linz, Austria, 13/07/16. https://doi.org/10.1007/978-3-319-41264-1_6

[Link to publication on Research at Birmingham portal](#)

Publisher Rights Statement:

Checked for eligibility: 01/11/2016

General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
- User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
- Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Polyfilling Accessible Chemistry Diagrams

Volker Sorge

School of Computer Science
University of Birmingham
`www.cs.bham.ac.uk/~vxs`

Progressive Accessibility Solutions Ltd.
Birmingham, UK
`progressiveaccess.com`

Abstract. We present a polyfill solution for replacing inaccessible images of molecules with fully web-accessible chemistry diagrams. Thereby marked up bitmap images on a client web site are extracted, server side recognised and semantically enriched to generate scalable vector graphics (SVG) with embedded chemical information. These graphics are then re-inserted into the original web page providing readers with accessibility features such as speech output and interactive exploration together with synchronised highlighting and magnification. Our solution works for most combinations of browsers and screen reading software on all major desktop platforms, and while it is currently only implemented for chemical diagrams, it is extensible to other STEM subject areas.

1 Introduction

Diagrams are an important means of conveying information in STEM subjects and they are ubiquitous in teaching material. Since they need to be understood in fine detail, their precise description is often very difficult and they thus present a major hurdle for inclusive education. Even in electronic teaching material, diagrams present an obstacle as they are generally given in raster-based image formats (e.g. gif, png, and jpeg), leaving them inaccessible for visually impaired learners: Screen readers can only pick up alternative texts that is usually not enough to convey a full description; and magnification tools struggle to deal with diagrams since magnification does not proportionally increase resolution leading to a loss of image quality.

To overcome this problem on the web, we present a new approach for making chemical diagrams accessible by automatically replacing inaccessible bitmap formats with fully web accessible scalable vector graphics (SVG) in web pages. The idea is implemented as a polyfill solution, that is, a JavaScript library that can be injected into any client web site, where marked up images are recognised and transformed server side into accessible SVG and re-inserted into the web page. In addition to recognising bitmap images, it can also exploit standard chemical markup notation to produce SVG diagrams.

The library draws on our work in fully automatically generating web accessible chemical diagrams, without the need for specialist tools for authoring or reading the accessible diagrams [8]. Instead it employs image analysis to recognise diagrams, semantic enrichment to derive detailed information on their

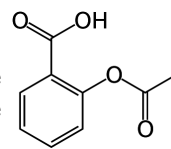
content, and regeneration into an annotated graphics format that makes them amenable to assistive technology. Web browsing software allows readers to interactively engage with diagrams by exploring them step-wise and on different layers, enabling aural rendering of diagrams and their individual components together with highlighting and magnification to assist readers with low vision or learning difficulties. And although originally developed as an assistive technology tool, the approach could also be exploited as a general teaching tool as well as extended to other STEM subject areas.

2 Producing Accessible Chemical Diagrams

Our approach is based on a procedure to recognise chemical diagrams from bitmap images and transform them into semantically enriched, fully web accessible SVG graphics. The procedure is fully automatic and combines three independent computational steps into a single software pipeline:

- (1) Image analysis recognises molecule diagrams.
- (2) Semantic enrichment computes detailed information on chemical molecules.
- (3) Reproduction of diagrams in navigatable Scalable Vector Graphics (SVG).

We summarise how each of these steps proceeds in this section. For more details on the process see also [8]. As a running example we use the recognition and semantic enrichment of the chemical molecule for Aspirin, which is originally given as the bitmap image presented on the right.



2.1 Image Analysis

The Image analysis that recognises molecule diagrams is based on our previous work [5], which has shown itself superior in a number of international recognition competitions [4, 6]. It proceeds in two stages: *Image Segmentation* decomposes diagrams into a set of geometric primitives. *Diagram Recognition* uses chemical knowledge to assemble a basic representation of the diagram.

In the *Image Segmentation* step an image is vectorised in order to segment it into the main constituents making up the diagram, resulting in a set of distinct primitives like lines, circles, solid triangles, arcs, or character groups together with their geometric location in the original image. This process is robust not only with respect to the type of bitmap images (e.g., jpeg, png, tiff), but also with respect to differences in authoring styles or potential problems stemming from the image origin. For example, noise introduced by image capturing techniques such as scanning is removed in a pre-processing step. The result of the segmentation is then a textual representation of the geometric primitives. For our Aspirin example we get a set of 22 geometric primitives: 18 lines and 4 character groups. These are given partially in Fig. 1.

In the subsequent *Diagram Recognition* step the actual recognition of the molecule is performed by a rule engine, in which largely disjoint rules are repeatedly applied to the initial set of geometric primitives, rewriting it into a

```

60;4;336;279;188.992693;206.825861
chargroup;0;258;223;287;257
chargroup;0;195;114;224;148
chargroup;0;6;5;35;39
chargroup;OH;132;5;192;39
line;normal;82;62;85;56;4;4611686018427387904.0
line;normal;83;130;82;63;4;4611686018427387904.0
line;normal;83;276;145;240;4;-0.577093
line;normal;20;239;82;276;4;0.578389
line;normal;274;166;269;166;4;0.000000
line;normal;267;220;268;167;4;4611686018427387904.0
...
line;normal;81;62;34;36;4;0.575445
line;normal;86;56;127;32;4;-0.575723
line;normal;85;55;39;28;4;0.580944

```

Fig. 1. Abbreviated list of geometric primitives for Aspirin.

```

<molecule id="m1" xmlns="http://www.xml-cml.org/schema">
  <atomArray>
    <atom id="a1" elementType="C" x2="1.4301" y2="-0.6083" hydrogenCount="3"/>
    <atom id="a2" elementType="C" x2="-0.4599" y2="-1.6683" hydrogenCount="1"/>
    ....
  </atomArray>
  <bondArray>
    <bond id="b1" atomRefs2="a3 a1" order="S"/>
    <bond id="b2" atomRefs2="a5 a4" order="S"/>
    <bond id="b3" atomRefs2="a6 a5" order="S"/>
    <bond id="b4" atomRefs2="a7 a4" order="D"/>
    ....
  </bondArray>
</molecule>

```

Fig. 2. Abbreviated CML for Aspirin molecule.

graph representation of the given molecule diagram. Rules are defined in terms of preconditions and consequences. A rule is applicable if there exist geometric objects that satisfy its preconditions. Executing its consequence results in the removal of existing geometric objects and the addition of elements to the graph as well as possibly the addition of new geometric objects. In general, preconditions of different rules are mutually exclusive, and thus the order of rule application is irrelevant.

The graph structure resulting from the rewriting step serves as a basis from which efficient electronic representation formats can be generated. While these formats are largely equivalent, we concentrate on generating standard chemical file formats such as MOL and CML (Chemical Markup Language [2]), as well as InChI (International Chemical Identifier). For Aspirin the InChI is given as

InChI=1S/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h2-5H,1H3,(H,11,12) while the corresponding CML is given in parts in Fig. 2.

It is worth noting that the image segmentation is fully generic, that is, it is independent of the actual type of diagrams analysed. Only the diagram recogni-

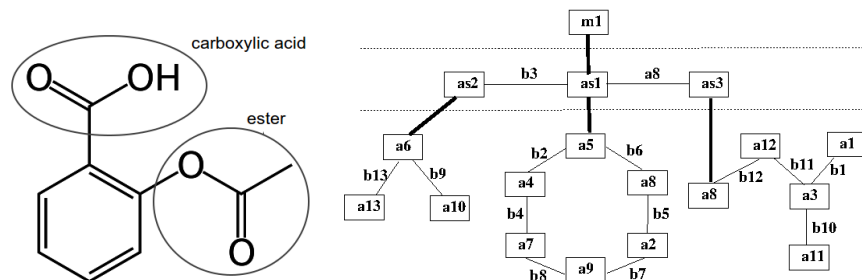


Fig. 3. Functional groups and abstraction graph for Aspirin molecule.

tion actually uses information on the domain (i.e., chemistry). Consequently, the approach is portable other STEM diagrams using similar geometric primitives simply by replacing the set of rules used for recognition and graph rewriting.

2.2 Semantic Enrichment

While the result of the image analysis is sufficient to reproduce the diagram and to distinguish molecules, the extracted information is still only sufficient for a flat representation, in the sense of describing single components of a diagram and their relationship to their direct neighbours. However, they lack sufficiently rich semantic that would be necessary to provide meaningful explanations of the diagram. Therefore the most challenging part is to enrich this representation with sufficient semantic meaning to allow the automatic generation of diagram descriptions that emulate human reading behaviour in its different facets, such as taking a casual glance at a drawing, getting an initial abstract overview of its components, before diving deeper into single components.

In our semantic enrichment we analyse the graph representing the molecule structure in order to identify chemically interesting compounds such as ring systems, aliphatic chains and functional groups exploiting cheminformatics algorithms implemented in the Chemistry Development Kit (CDK) [9]. This imposes a hierarchical structure on the molecule that we can later exploit for navigation. In addition identified substructures are being automatically named using on-line web services (e.g., [3, 10]) that generate common chemical names given a specification of a compound.

For the Aspirin molecule we identify three major components: a Benzene ring, two functional groups, Carboxylic Acid and Ester. These are depicted in Fig. 3 on the left. Once identified these components are combined and represented in the abstraction graph given in Fig. 3 on the right and names for them as well as for the overall molecule, i.e. Aspirin, are computed via ChemSpider [3]. The computed information is then represented as an XML structure to extend the basic CML representation to an enriched CML format that contains the administrative information for the abstraction graph as well as sufficient information on later generating speech strings for the diagram.

2.3 Annotated SVG Generation

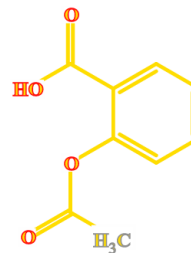
Already the non-enriched, generated CML representation of a molecule can serve as the basis to compute the corresponding diagram as SVG. Although there exist a number of solutions for this, these are exclusively geared towards rendering a diagram, discarding all chemical information in the process. That is, they will set all the geometric components, lines and characters, in a flat structure, losing information about bonds or atoms. As making a connection between the geometric component of the SVG and the bonds and atoms in the input CML file is important for the purpose of highlighting and magnification, we have implemented our own SVG renderer. It exploits SVG facilities to group elements together as well as to add attributes reflecting their chemical purpose and connecting them to their origins in CML.

3 Web Integration and Front End

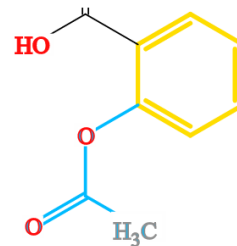
The SVG resulting from the recognition process can directly be included into the rendered page. However, we also want to exploit the abstraction graph structure to overlay the SVG with a navigation model to enable users to explore diagrams interactively. Unfortunately, this is not doable by embedding the navigation structure into the SVG alone, as SVG is a tree structure, while the abstraction graph is a hierarchical graph with shared elements. Consequently, we need this structure independently in the web page, and connect it to the SVG via JavaScript functionality.

More technically, we employ AJAX to import the SVG together with the semantically enriched chemical information as an SVG+XML media type into the web page. Some injected JavaScript code then enables interactive exploration of diagrams. The main idea is that a user can enter a diagram and interactively browse through its components on different levels and in different granularity. The components are presented to the reader by making descriptions available for aural rendering by a screen reader through pushing text strings into a ARIA live region. Parts of the molecule structure can be focused, both by highlighting and optionally via magnification. These functions are implemented via CSS changes and moving the SVG viewport, respectively. All interactions and voicing operations are thus implemented browser, screen reader, and platform independent, exploiting HTML5, CSS and WAI-ARIA standards.

Considering again our example molecule Aspirin, its CML structure allows us to generate an SVG diagram that can be imported into any modern web browser. On the right is an image of the SVG diagram already highlighted on the top molecule level, that corresponds to the entire Aspirin molecule. When browsing the molecule highlighting is adjusted via CSS and focused parts are magnified using the SVG viewport.



Browsing the molecule diagram allows us to move vertically between the different levels of the abstraction graph as well as horizontally between nodes on a single level of the graph. For example, one can move from the top level Aspirin molecule, down to the major component level and around this level. The image on the right depicts a move on the major component level, from the Benzene ring to the functional group Carboxylic Acid. This step is voiced as “Benzene ring with Carboxylic Acid at substitution 1”.



4 Polyfilling Diagrams on the Web

Polyfills are a technology that aims to mask the discrepancy in provision of functionality and APIs from different web browsers and platforms by providing uniform APIs, allowing developers to implement a homogeneous user experience in a platform independent way [7]. While often intended to be of a temporary nature, to fill the gap between new web standards and their lack of browser implementation, and in particular ensuring backwards compatibility for old browsers, many have become permanent solutions over time. A prominent example is MathJax [1], that was originally designed as a stop gap for displaying mathematics on the web until the MathML standard would be implemented in all browsers. However, since only very few browsers implement MathML (and even then only incomplete), MathJax has become the de facto rendering solution for Mathematics. Moreover, it can deal not only with MathML but with the more prevalent formats of \LaTeX and ASCIIMath, replacing those expressions in web pages by DOM components that produce the correct visual rendering.

We have chosen a similar approach to tackling the problem of inaccessible diagrams on the Web. The basic idea is to provide a polyfill solution in form of a JavaScript library called *DIAGcess* that can replace inaccessible chemical diagrams or notation by fully accessible SVG diagrams upon page load. *DIAGcess* can be included via a `script` tag into a website and bitmap images of chemical molecules can be marked for transformation. The actual transformation is performed server side and its result, the accessible SVG diagram, replaces the original bitmap image in the page, using the described AJAX functionality.

We thereby support three different types of transformations, depending on the markup of an image element. Figure 4 displays the HTML element for including Aspirin as a bitmap in a web page. The element contains three dedicated data attributes `data-chemistry-access`, `-cas`, and `-inchi`. If the former attribute is set to true, the image will be transformed and replaced. How exactly the transformation works depends on which of the two latter attributes are given:

- If only `data-chemistry-access` is given, *DIAGcess* will run the entire recognition process starting in step (1) of the procedure presented in Sec. 2.
- If `data-chemistry-inchi` is provided with a legal InChI code *DIAGcess* will directly generate the accessible SVG diagram, effectively starting at step (2) of the process, omitting image analysis.

```



```

Fig. 4. HTML element for including Aspirin image in page.

```

<div class="ChemAccess-element" tabindex="0" role="application"
  aria-label="Navigatable molecule" has-cml="true" has-svg="true">
  <div class="cml" aria-hidden="true">
    <molecule id="m1" xmlns="http://www.xml-cml.org/schema">
      ...
    </molecule>
  </div>
  <div class="svg" aria-hidden="true">
    <svg xmlns="http://www.w3.org/2000/svg" width="200" height="175" ...>
      ...
    </svg>
  </div>
</div>

```

Fig. 5. HTML element for including accessible SVG diagram of Aspirin in page.

- If only **data-chemistry-cas** is given, the attribute contains the unique registry number of the molecule provided by the American Chemical Society. *DIAGcess* will then call ChemSpider [3] to determine the corresponding InChI code and proceed as in the previous case.

Starting in either of the latter two steps has the obvious advantage that they will always yield fully correct diagrams as no recognition errors can be introduced. In all cases, the original image element in the web site will be replaced by a container element presented in Fig. 5 that wraps together both the semantically enriched CML structure (i.e., an augmented version of the one presented in Fig. 2) and the annotated SVG, tagged the appropriate ARIA elements.

5 Conclusions

The presented work flow transforms inaccessible molecule images into accessible interactive SVG diagrams without the need for manual intervention. It neither relies on authors to produce accessible content nor requires users to install and learn additional software tools. We have integrated this process into the JavaScript library *DIAGcess* that acts as a polyfill solution, allowing automatic transformation of images in web pages and easy inclusion into web content.

As the image analysis phase can introduce errors, we believe that the ability to generate accessible SVG from unique chemical identifiers is of great advantage in the context of *DIAGcess*, where we assume that web page authors have knowledge on the molecules included in a page and can thus provide the correct identifiers. The advantage of using a polyfill solution rather than immediately

integrating the accessible SVG is, that it allows to exploit updates and improvements made in the library. Moreover, certain browsers do not support the inclusion of XML document structures, necessary for navigation, directly from source. In addition, the code for navigation does not need to be provided locally.

We have carried out user testing for the diagram navigation via a demonstrator page with a collection of accessible molecules. The feedback we have received was primarily positive, particularly commending the simplicity of the navigation model, but results are not yet statistically significant. However, we hope that making *DIAGcess* available will lead to more feedback, as it allows users to easily work with material they are interested in.

DIAGcess has been implemented as a platform independent tool, and can run in all major modern browsers that support SVG. Similarly the browser front-end that supports navigation, highlighting and aural rendering, works with the majority of modern screen readers that support ARIA live regions, on all major desktop platforms. The library has to be installed and included locally for now, however, we aim to eventually distribute it via content distribution network.

Although the server back-end has currently been realised for chemical diagrams, only, the approach is not restricted to chemistry but extensible to diagrams in other STEM subjects. In fact, both the image analysis and the semantic enrichment procedure are separated into a generic and a domain specific part, where the latter can be parameterised by providing rule sets for syntactic recognition and semantic interpretation, respectively. Consequently future work will consist of extending our ideas to diagrams in other STEM subjects, like mathematics, physics and biology.

References

1. MathJax Consortium. MathJax version 2.5, 2014. <http://www.mathjax.org>.
2. P. Murray-Rust and H. Rzepa. Chemical markup, xml and the world-wide web. the cmlDOM. *J. Chem. Inf. Comput. Sci.*, 41(5):1113–1123, 2001.
3. Royal Society of Chemistry. Chempidder: Search and share chemistry. <http://chemspider.com>.
4. N. Sadawi, A.P. Sexton, and V. Sorge. Performance of MolRec at TREC 2011: Overview and analysis of results. In *Proc. of TREC-20*. NIST, 2011.
5. N. Sadawi, A.P. Sexton, and V. Sorge. Chemical structure recognition: a rule-based approach. In *Doc. Recognition & Retrieval XIX*, volume 8297. SPIE, 2012.
6. N. Sadawi, A.P. Sexton, and V. Sorge. Molrec at CLEF 2012: Overview and analysis of results. In *CLEF Evaluation Labs*, 2012. <http://clef2012.org>.
7. Remy Sharp. What is a polyfill?, October 2010. <https://remysharp.com/2010/10/08/what-is-a-polyfill>.
8. V. Sorge, M. Lee, and S. Wilkinson. End-to-end solution for accessible chemical diagrams. In *Proceedings of the 12th Web for All Conference*, page 6. ACM, 2015.
9. C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E. Willighagen. The chemistry development kit. *J. Chem. Inf. Comput. Sci.*, 43(2):493–500, 2003.
10. CADD Group Chemoinformatics Tools and User Services. Chemical identifier resolver. <http://cactus.nci.nih.gov/chemical/structure>.