Extricating Meaning from Wikimedia Article Archives

Brian W. Curry, Andrew Trotman, and *Michael Albert*

Computer Science
University of Otago
Otago 9010 New Zealand

http://raiazome.com | (*andrew* | *malbert*)*@cs.otago.ac.nz*

Abstract *Wikimedia article archives (Wikipedia, Wiktionary, and so on) assemble open-access, authoritative corpora for semantic-informed datamining, machine learning, information retrieval, and natural language processing. In this paper, we show the* `MediaWiki` *wikitext grammar to be context-sensitive, thus precluding application of simple parsing techniques. We show there exists a worst-case bound on time complexity for all fully compliant parsers, and that this bound makes parsing intractable as well as constituting denial-of-service (DoS) and degradation-of-service (DegoS) attacks against all* `MediaWiki` *wikis. We show there exists a worse-case bound on storage complexity for fully compliant one-pass parsing, and that contrary to expectation such parsers are no more scalable than equivalent two-pass parsers. We claim these problems to be the product of deficiencies in the* `MediaWiki` *wikitext grammar and, as evidence, comparatively review 10 contemporary wikitext parsers for noncompliance with a partially compliant Parsing Expression Grammar (PEG).*

Keywords Document Standards, Information Retrieval, Web Documents, Wikipedia

1 Introduction

Wikimedia article archives assemble open-access, authoritative corpora for semantic-informed datamining, machine learning, information retrieval, and natural language processing. Unfortunately, these archives are described by an ad hoc document standard for which there exist no formal grammars for producing conformant parsers *or* conformant parsers beyond the de facto reference implementation, `Pywikipedia` [2].

In this paper, we show deficiencies in this standard to cause widespread non-compliance in third-party wikitext parsers, intractable storage and time complexity for all wikitext parsers (including `MediaWiki` *itself) and viable denial-of-service (DoS) and degradation-of-service (DegoS) attacks against the most recent official release of* `MediaWiki` *as of this writing,* `MediaWiki` *1.16.0.*

Proceedings of the 16th Australasian Document Computing Symposium, Melbourne, Australia, 10 December 2010.

As evidence of noncompliance, we comparatively review 10 contemporary wikitext parsers (including `MediaWiki` itself) against a partially compliant Parsing Expression Grammar (PEG) [6]**.** This review suggests there exists no fully compliant offline wikitext parser and only one fully compliant online wikitext parser *and it is not* `MediaWiki` *itself*: `Pywikipedia` [2]. Furthermore, the least compliantly parsed semantics are those we show induce worst-case intractability and insecurity in wikitext parsers.

As evidence of intractability, we present worst-case article archive input generally applicable to third party wikitext parsers. Analysis shows this input makes storage complexity prohibitive in disambiguation-compliant parsers and time complexity intractable in transclusion-compliant parsers.

As evidence of insecurity, we present worst-case article archive input specifically applicable to `MediaWiki` itself. Injecting this input into a local "clean-room" installation of the most recent official release of `MediaWiki` [1] shows this input makes all `MediaWiki` wikis susceptible to currently unresolved DoS and DegoS attacks, a compelling security flaw.

Finally, we present a partially compliant PEG. Constructed from exhaustive inspection of the `MediaWiki` codebase and real-world tests against local and remote `MediaWiki` wikis, this grammar matches and consumes most inter-article syntax (i.e., syntax conveying semantics *between* rather than *in* articles).

Due to its topical diversity, this paper's intended audience is threefold:

1. *Security consultants*, *system administrators* and *`MediaWiki`-invested policymakers*, given our exposure of prevailing vulnerabilities in the `MediaWiki` codebase (in Section 4).
2. *Information retrieval (IR)* and *natural language processing (NLP) specialists* as well as *`MediaWiki`-reliant dataminers*, given our findings of extensive noncompliance in official and third-party wikitext parsers (in Section 2) and appended publication of a partially compliant PEG (in the Appendix).
3. The *Wikimedia Foundation*, given our remedies of existing deficiencies in the Wikimedia article archive document standard (in Section 3).

	categorizations	disambiguations	occlusions	redirects	transclusions	wikilinks	interwikilinks
JWPL 0.453.4 [16]	*partial*	no	no	*partial*	*partial*	*partial*	*partial*
MediaWiki 1.16.0 [1]	**full**	no	**full**	**full**	**full**	**full**	**full**
mwlib (*7dbed545c6cd*) [11]	**full**	no	no	*partial*	**full**	*partial*	**full**
Parse::MediaWikiDump 1.0.6_01 [12]	*partial*	no	no	*partial*	no	no	no
Pywikipedia (*8616*) → *offline* [2]	*partial*	*partial*	**full**	*partial*	*partial*	*partial*	*partial*
Pywikipedia (*8616*) → *online* [2]	**full**	**full**	**full**	**full**	**full**	**full**	**full**
Yppy 0.0.8 [5]	**full**	*partial*	**full**	**full**	no	**full**	*partial*
Wiki2XML (*56074*) [9]	*partial*	no	*partial*	*partial*	*partial*	*partial*	*partial*
Wikipedia Miner (*92*) [10]	no	no	no	no	no	*partial*	no
Wikiprep 3.04 [7]	**full**	*partial*	*partial*	*partial*	*partial*	*partial*	*partial*
WWW:Wikipedia 1.97 [13]	no	no	no	no	no	*partial*	no

Table 1: Prevalence of fully compliant wikitext parsing

	fully parsed semantics	partially parsed semantics
Pywikipedia → *online*	**7**	*0*
MediaWiki	**6**	*0*
Yppy	**4**	*2*
mwlib	**3**	*2*
Pywikipedia → *offline*	**1**	*6*
Wikiprep	**1**	*6*
Wiki2XML	**0**	*6*
JWPL	**0**	*5*
Parse::MediaWikiDump	**0**	*2*
Wikipedia Miner	**0**	*1*
WWW:Wikipedia	**0**	*1*

Table 2: Parser compliance from Table 1

	fully compliant parsers	partially compliant parsers
disambiguations	**1**	*2*
occlusions	**3**	*2*
transclusions	**3**	*3*
interwikilinks	**3**	*4*
redirects	**3**	*5*
wikilinks	**3**	*6*
categorizations	**5**	*3*

Table 3: Semantic compliance from Table 1, ignoring offline Pywikipedia

2 Comparison

Comparative review of 10 contemporary wikitext parsers against our Appendix-presented grammar reveals common non-compliance. There exists no fully compliant offline parser and only one fully compliant online parser: Pywikipedia. Partially compliant parsers parsing most semantics include (in descending order of compliance): MediaWiki, Yppy and mwlib.

Table 1 summarizes this review. Rows signify reviewed parsers, columns reviewed semantics and row-column entries each parser's degree of compliance in parsing each semantic. Parser names are suffixed with the version or version control revision in parentheses we reviewed, preferring the latter for parsers whose most recent official release (as of this writing) was several months outdated or for which there was no official release (e.g., Pywikipedia).

We now discuss semantics, compliance issues associated with each and notable parsers compliantly addressing these issues.

2.1 Semantic compliance

Our review ignored all semantics other than those listed in the Table 1 header. Table 3 orders these semantics by ascending count of fully and partially compliantly parsed semantics in the first and second columns. Two of the three least compliantly parsed semantics produce worst-case bounds on wikitext parsing: *disambiguations* (producing prohibitive storage complexity for fully compliant one-pass parsing in Section 3.4) and *transclusions* (producing intractable time complexity for all fully compliant parsing in Sections 3.2 and 3.3). The remaining least compliantly parsed semantic, *occlusions*, is implicated in grammatical context-sensitivity (generating 75% of all context-sensitive productions in Section 3.1).

We first discuss *canonicalization,* a prerequisite for fully compliant parsing of 6 of our 7 reviewed semantics. Canonicalization reduces article titles in non-canonical to canonical form, enabling meaningful comparison between article titles regardless of form (e.g., a canonical article title ‘‘Talk:Hastur’’ refers to the same article as a non-canonical article title ‘‘TaLK__: hastur’’). There exist countably infinite non-canonical forms of each article title, so non-canonicalizing parsers return false negatives *and* positives by improperly matching non-canonical forms of the same article title as different articles. Canonicalizing parsers substitute, in article titles:

1. Embedded transclusions with their expansions.
2. Runs of space and underscore characters with a single space (e.g., ‘‘`_ __`’’ with ‘‘ ’’).
3. Namespace aliases with the corresponding namespace name (e.g., ‘‘`w:`’’, ‘‘`WP:`’’, and ‘‘`Project:`’’ with ‘‘`Wikipedia:`’’).
4. Named-, decimal-, and hexadecimal-style HTML entities with the corresponding character (e.g., ‘‘`&#230;`’’ and ‘‘`&aelig;`’’ with ‘‘æ’’).
5. Subpage prefix ‘‘`/`’’ with the current article title within namespaces enabling the subpage feature (e.g., ‘‘`/`’’ with ‘‘`Talk:Yuggoth`’’ for all wikilinks in that article).
6. Autoformats with one or more canonical wikilinks to actual articles. This includes the *month day/month name* date autoformat, reformatted to read the opposite (e.g., `[[15 March]]` with `[[March 15]]`), and *ISO 8601* date autoformat, reformatted to read as two wikilinks (e.g., `[[1890-08-20]]` with ‘‘`[[August 20]] [[1890]]`’’).

We now discuss specific semantics.

Categorizations classify one article under another, denoted by double square brackets and language-specific `Category` namespace name (e.g., a string `[[Category:Mythos]]` classifies the current article under that category). Categorization compliance implies language-specific matching *and* canonicalization. Parsers disregarding categorizations confuse categorizations for wikilinks, since the two share the same notation.

Disambiguations are articles *transcluding* (see below) at least one language-specific disambiguation template (e.g., a string `{{Disambig}}` classifies the current article as a disambiguation). Disambiguation compliance implies preparsing the ‘‘`MediaWiki:Disambiguationspage`’’ article or equivalent metadata for the set of all disambiguation template names, matching those names *and* canonicalization. Parsers disregarding disambiguations assume redirects and wikilinks to disambiguations to be semantically meaningful, but they are not (e.g., a non-ambiguous wikilink `[[Dagon]]` to that article is semantically meaningful while an ambiguous wikilink `[[Dagon (disambiguation)]]` to that disambiguation is not).

Occlusions hide content from end users, denoted by XML 1.1-conformant **(a)** opening tag consisting of the ‘`<`’ character, tag name, optional attributes and ‘`>`’ character, **(b)** tag-specific text, and **(c)** closing tag consisting of the ‘‘`</`’’ string, same tag name and ‘`>`’ character (e.g., ‘‘`<pre>[[Shoggoth]]</pre>`’’, which `MediaWiki` renders as the raw text `[[Shoggoth]]` rather than a wikilink to that article). Occlusion compliance implies matching this syntax *and* context-sensitively not matching any wikitext syntax in this syntax, as such tags *occlude* their content from conventional parsing. Parsers disregarding occlusions return false positives by improperly matching occluded wikitext.

Redirects symbolically link one article to another, denoted by `#REDIRECT` followed by a wikilink as the first wikitext for an article (e.g., a string ‘‘`#REDIRECT [[Azathoth]]`’’ redirects the current article to that). Redirect compliance implies matching *and* canonicalization. Parsers disregarding redirects confuse redirects for wikilinks, since the two share the same notation.

Transclusions dynamically expand one template's wikitext into the current article, denoted by double curly braces (e.g., a string `{{Arkham}}` expands that template's wikitext into the current article). Templates are articles under the `Template` namespace, so a transclusion `{{Template_name}}` actually transcludes an article named `Template:Template_name`. Transclusion compliance implies matching, canonicalization *and* fully recursive expansion. Parsers disregarding transclusions return false negatives by failing to expand transcluded wikitext. However, we show in Sections 3.2, 3.3 and 4 that the computational intractability of worst-case expansion makes such parsing inherently unsafe. Counter-intuitively, this implies that no transclusion compliance in a parser may be preferable to partial or full compliance.

Wikilinks link from one article to another on the same wiki, denoted by double square brackets (e.g., `[[Yog-Sothoth]]`). Wikilink compliance implies matching *and* canonicalization.

Interwikilinks link from one article to another on another wiki, denoted by double square brackets and a `MediaWiki`-recognized wiki name (e.g., `[[craftywiki:Horror]]`). Interwikilink compliance implies preparsing the ‘‘`List of Wikipedias`’’ and ‘‘`Meta:Interwiki map`’’ articles or equivalent metadata, matching *and* canonicalization – meriting distinction from mere wikilink compliance. Parsers disregarding interwikilinks confuse interwikilinks for wikilinks, since the two share the same notation. However, interwikilinks convey no meaningful semantics for most third-party parsers.

2.2 Parser compliance

Table 2 orders parsers by descending count of fully compliant (first) and partially compliant (second) semantics parsed. Three of the four most compliant parsers are PYTHON-implemented. All four of the least compliant parsers are JAVA- and PERL-implemented. We failed to find a working non-interpreted implementation, though *non*-working non-interpreted implementations do exist (e.g., `FlexBisonParse` [14]). This suggests multi-paradigmal, dynamically typed, interpreted languages to be ideal mediums for wikitext parsing.

`Pywikipedia`'s full compliance is the collaborative result of its real-world use on `MediaWiki`-hosted wikis, the Wikimedia Toolserver and offline Wikimedia article archives. Our comparison is not entirely

fair, therefore: `Pywikipedia` queries remote `MediaWiki` APIs for language-specific metadata retrieval, transclusion expansion and article validation. Denying `Pywikipedia` network access reduces its compliance to beneath that of `mwlib` – a more judicious comparison, perhaps.

Offline parsers have no access to comparable APIs, necessitating they statically populate data structures with a priori site-specific metadata *as well as* independently implement template preprocessors, article validators, etc. To accommodate this, we split `Pywikipedia` into "`Pywikipedia` $\rightarrow$ *offline*" (i.e., when offline) and "`Pywikipedia` $\rightarrow$ *online*" (i.e., when online).

`MediaWiki`'s lack of full compliance is a result of its inability to distinguish disambiguation from non-disambiguation articles, as has been noted at English Wikipedia itself [4].

3 Parser analysis

We expose deficiencies in the wikitext grammar and, for each deficiency, recommend amendments to existing Wikimedia policy.

3.1 Grammatical context-sensitivity

We now show the wikitext grammar to be context-sensitive, principally due to the presence of occlusions.

Suppose the wikitext grammar to be context-free. Then the production "*wikilink* $\leftarrow$ *wikilink_begin wikilink_type wikilink_end*" context-freely matches wikilink `[[R'lyeh]]` in wikitext ''`<nowiki>[[R'lyeh]] </nowiki>`''. However, `MediaWiki` parses wikilink syntax in `nowiki` tags as raw text rather than a wikilink. Then this production must match context-sensitively, a contradiction.

The `nowiki` tag occludes its wikitext content from conventional parsing. There exist 8 occluding tags: `<!-...-!>`, `includeonly`, `nowiki`, `timeline`, `math`, `pre`, `source` and `syntaxhighlight`. The latter 4 accept optional attributes; the remainder do not. Additionally, the closing tag corresponding to an opening occluding tag matches non-greedily (and hence context-sensitively).

This and the number of occluding tags complicate occlusion matching, as evidenced by the ratio of the number of occlusion productions to total number of productions κ. Of the 91 total productions, 30 involve occlusions. Of the 12 total context-sensitive productions, 9 involve occlusions. Then $\kappa \simeq 1/3$ in the set of all productions and $\kappa = 3/4$ in the set of context-sensitive productions, indicating occlusions dominate both. However, occlusions convey no meaningful semantics apart from their disabling of meaningful semantics!

> As solution, we recommend Wikimedia distribute article archives encoding all *occluded* English punctuation as HTML entities (e.g., encoding ''`<nowiki>[[Cyaegha]]</nowiki>`'' as ''`<nowiki>&#91;&#91;Cyaegha&#93;&#93;</nowiki>`''). This encoding is bijective and thus losslessly decodable on article archive deserialization. Since non-occlusion productions do not decode HTML entities, non-occlusion productions cannot match in HTML entity-encoded occlusion wikitext. Then given such an archive such productions are context-free. The resulting wikitext grammar may omit occlusion productions without concomitant loss of semantic compliance, and our PEG reduces to 61 total productions of which only 3 remain context-sensitive. It can be shown that these are also convertable to context-free productions, but only by breaking backward compatibility in the wikitext grammar.

3.2 Worst-case time complexity

We now show fully compliant parsers to suffer intractable worst-case time complexity $\mathrm{O}\left(|\mathfrak{M}|c^{|\mathfrak{T}|}\right)$, $|\mathfrak{M}|$ the number of non-template articles, $|\mathfrak{T}|$ the number of templates and c the maximum number of template transclusions per non-template article.

Consider the article archive consisting of at least two articles, all residing in the `Main` and `Template` namespaces and no others. The article set is partitionable into set $\mathfrak{M}$ on the `Main` non-templates and poset $\mathfrak{T}$ on the `Template` templates. Suppose non-templates have arbitrary titles and templates have minimal length titles, such that lexicographic comparison defines a well-ordering on $\mathfrak{T}$ (e.g., the first template is entitled ''`a`'', the second ''`b`''). Suppose non-template wikitext maximally transcludes the first template (i.e., maximally many repetitions of ''`{{a}}`''), template wikitext maximally transcludes the next template in $\mathfrak{T}$ for all templates except the last (e.g., template ''`a`'' wikitext is maximally many repetitions of ''`{{b}}`''), and the last template in $\mathfrak{T}$ terminally expands to binomially distributed single digit '`0`' or '`1`'. Then all wikitext reduces to stochastic strings of '`0`' and '`1`' in the final expansion, and no transclusion expansion is losslessly memoizable. (Such expansions *are* memoizable if one does not mind the loss, as `MediaWiki`'s lossy memoization shows in Section 4.) Figure 1 depicts these assumptions with arrows signifying transclusion.

As wikitext length is bounded, the number of transclusions per article is bounded. Let c be this bound. Then each article comprises one node in the rooted tree of transclusions with fanout c where: **(a)** the first template in $\mathfrak{T}$ roots each such tree, **(b)** the last template in $\mathfrak{T}$ serves as the leaf nodes and **(c)** all other templates serve as internal nodes. All such trees are identical, so consider any tree r. By assumption r is complete of height $h = |\mathfrak{T}| - 1$. So r is size $s(c, |\mathfrak{T}|)$ given by

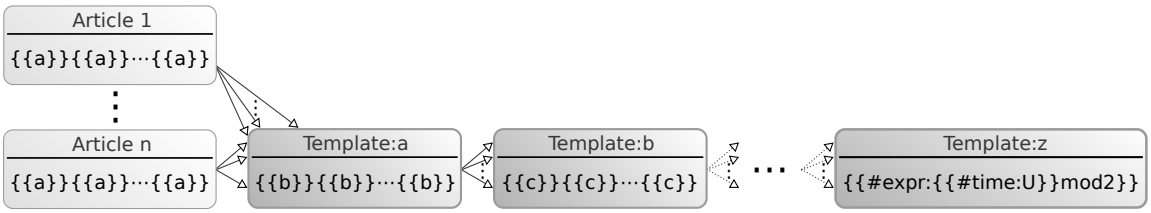

Figure 1: Article archive exhibiting worst-case time complexity

$$s(c, |\mathfrak{T}|) = \frac{c^{|\mathfrak{T}|} - 1}{c - 1}.$$

As each non-template recursively transcludes each such tree c times, the total number of transclusions $t(|\mathfrak{M}|, |\mathfrak{T}|)$ is given by

$$\begin{aligned} t(|\mathfrak{M}|, |\mathfrak{T}|) &= c|\mathfrak{M}|\, s(c, |\mathfrak{T}|) \\ &\in \mathrm{O}\left(\frac{c}{c-1}|\mathfrak{M}|\left(c^{|\mathfrak{T}|} - 1\right)\right) \\ &\in \mathrm{O}\left(|\mathfrak{M}| c^{|\mathfrak{T}|}\right). \end{aligned}$$

As example, let $|\mathfrak{M}| = 6$ and $|\mathfrak{T}| = 26$ such that template titles iterate the alphabet for the worst-case article archive of 32 articles. Clearly, each template title consumes 1 byte and each transclusion 5 bytes. Backward compatibility requires wikitext length be practicably bounded to 32KB [3]. Then $c = \lfloor {}^{32\mathrm{KB}}/_{5B} \rfloor = 6553$, and

$$\begin{aligned} t(|\mathfrak{M}|, |\mathfrak{T}|) &\in \mathrm{O}\left(6 \cdot 6553^{26}\right) \\ &\in \mathrm{O}\left(10^{100}\right). \end{aligned}$$

Parsing the worst-case article archive of only 32 articles expands on the order of a googol transclusions.

As solution, we recommend precaution in the `MediaWiki` codebase against pathological template abuse. Given the improbability of third-party reimplementations of fully compliant *and* precautionary transclusion preprocessors, we recommend Wikimedia distribute one additional article archive for each site

1. comprising all articles except those residing in the `Template` namespace, and
2. expanding transclusions in all article wikitext "in place" at archive creation time.

3.3 Worst-case time complexity (revisited)

We established a template-dependent worst-case time complexity for fully compliant wikitext parsing in the previous Section. Now we revisit the issue with a worst-case grammar establishing a grammar-dependent worst-case time complexity and showing the latter to aggravate constant costs associated with the former, producing an aggregate worst-case time complexity.

As the Appendix discusses, disambiguation-compliant grammars are archive-specific. They remain incomplete until parsing the ‘‘`MediaWiki:Disambiguationspage`’’ article, after which the DISAMBIGUATION production may be specified to match all archive-specific disambiguation template names and thereby complete the grammar. The size of this production and thus this grammar is a function of the number of such names.

As wikitext length is bounded, the number of disambiguation template names referenced in ‘‘`MediaWiki:Disambiguationspage`’’ wikitext is bounded. Let d be this bound. Then identifying disambiguation articles requires eligible transclusions (i.e., transclusions in wikitext residing in the `Main` namespace) be iteratively tested against d alternatives.

Recall that each non-template in the worst-case article archive of the previous Section consisted of maximally many such transclusions. Append a ‘‘`MediaWiki:Disambiguationspage`’’ article referencing maximally many disambiguation template names to this archive. Then the total cost of parsing transclusions $T(|\mathfrak{M}|, |\mathfrak{T}|)$ is given by

$$\begin{aligned} T(|\mathfrak{M}|, |\mathfrak{T}|) &= d \cdot t(|\mathfrak{M}|, |\mathfrak{T}|) \\ &\in \mathrm{O}\left(\frac{dc}{c-1}|\mathfrak{M}|\left(c^{|\mathfrak{T}|} - 1\right)\right) \\ &\in \mathrm{O}\left(t(|\mathfrak{M}|, |\mathfrak{T}|)\right). \end{aligned}$$

Parsing this article archive expands the same order of transclusions as the prior, but amplifies the constant cost associated with doing so. We now show these constants to be non-negligible.

As stated wikitext length is bounded to 32KB. Suppose disambiguation template names are 2 bytes in length. In ‘‘`MediaWiki:Disambiguationspage`’’ wikitext, the itemization of each such name requires a 12 byte prefix ‘‘`*[[Template:`’’ and 3 byte suffix ‘‘`]]\n`’’ for automated bot discovery. Then each such item consumes 17 bytes and $d = \lfloor {}^{32\mathrm{KB}}/_{17\mathrm{B}} \rfloor = 1927$, certainly within the range of 2 byte template names. Incorporating non-negligible constants, aggregate worst-case time complexity $T(|\mathfrak{M}|, |\mathfrak{T}|)$, $|\mathfrak{M}|$ the number of `Main` namespace articles and $|\mathfrak{T}|$ the number of `Template` namespace articles, is

$$T(|\mathfrak{M}|, |\mathfrak{T}|) = 1927 \cdot \mathrm{O}\left(|\mathfrak{M}| 6553^{|\mathfrak{T}|}\right).$$

As solution, we recommend Wikimedia eliminate article-specific circular dependencies in the wikitext grammar. To do so for disambiguations, we propose the `#DISAMBIG` pragma explicitly declaring an article to be a disambiguation page. This pragma maintains backward compatibility with `MediaWiki` syntax, third-party parsers and the article corpus itself by requiring this "magic word" prefix be suffixed with a disambiguation transclusion (e.g., by supplanting all instances of ''`{{Disambig}}`'' in English Wikipedia with ''`#DISAMBIG {{Disambig}}`''). This pragma also adds explicit invariance to the existing wikitext grammar: namely, that each disambiguation page be associate with one and only one disambiguation template. In the existing wikitext grammar, disambiguation pages may be associate with no such template (by explicitly categorizing themselves under `[[Category:Disambiguation]]` rather than transcluding such a template) or more than one (by transcluding more than one, in which case the resulting disambiguation is inconsistent). This has the beneficial by-product of eliminating additional context-sensitivity from the wikitext grammar, which when coupled with the recommendation of Section 3.1 reduces the number of context-sensitive productions to 2. For alternative solution, see English Wikipedia's ''`Wikipedia: Disambiguation pages aren't articles`''.

3.4 Worst-case one-pass complexity

We now show fully compliant one-pass parsers to suffer prohibitive worst-case storage $\mathrm{O}(|\mathfrak{N}|)$, $|\mathfrak{N}|$ the number of articles, and scale no better than equivalent two-parse parsers in this case. *To exhibit these inefficiencies, this Section presents worst-case article archive input orthogonal to that of Section 3.3. While the two could be profitably composited into another aggregate worst case, that does not substantially revise the conclusion of this Section.*

Consider the article archive consisting of poset $\mathfrak{N} = (\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_{|\mathfrak{A}|}, \mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_{|\mathfrak{G}|})$, $\mathfrak{G}$ the non-empty set of grammar-generative articles comprising at least ''`MediaWiki:Disambiguationspage`'', ''`List of Wikipedias`'' and ''`Meta:Interwiki map`'' and $\mathfrak{A}$ the non-empty set of all remaining articles. Since the number of grammar-generative articles (3 under our PEG) is substantially smaller than the number of non-grammar-generative articles (3,447,220 for English Wikipedia as of this writing), $|\mathfrak{G}| \ll |\mathfrak{A}| \simeq |\mathfrak{N}|$.

Suppose all wikitext in $\mathfrak{A}$ consists of maximally many transclusions and/or wikilinks containing a colon. Then each such transclusion ambiguously signifies a possible disambiguation and each such wikilink a possible interwikilink or interlanguagelink. Since no parser may certify which is which until having parsed all wikitext in $\mathfrak{G}$, the resulting article archive exhibits *maximal semantic ambiguity*.

As example, the wikilink `[[Yog: Sothoth]]` ambiguously signifies either **(a)** a wikilink to that article, **(b)** an interwikilink to article ''`Sothoth`'' on the external wiki identified by interwiki prefix ''`yog`'' or **(c)** an interlanguagelink to the same article on the external Wikimedia wiki identified by language code ''`yog`'' (e.g., `http://yog.wikipedia.org/wiki/Sothoth`).

Suppose we implement a two-pass parser naïvely resolving these ambiguities as follows:

1. In the first pass, linearly search the article archive for all articles in $\mathfrak{G}$ ignoring all wikitext except that in $\mathfrak{G}$. These are the last articles in $|\mathfrak{N}|$, incurring storage cost $\mathrm{O}(|\mathfrak{G}|)$ and time cost $\mathrm{O}(|\mathfrak{N}|)$. Then generate the archive-specific grammar required for fully compliant parsing.
2. In the second pass, linearly parse all wikitext given this grammar, incurring time cost $\mathrm{O}(|\mathfrak{N}|)$.

Then the naïve two-pass parser suffers worst-case storage $\mathrm{O}(|\mathfrak{G}|)$ and time $2\,\mathrm{O}(|\mathfrak{N}|) = \mathrm{O}(|\mathfrak{N}|)$. Suppose we optimize this into a one-pass parser as follows:

1. Linearly parse wikitext until parsing all wikitext in $\mathfrak{G}$, caching all semantically ambiguous wikitext for subsequent reparsing. Since all wikitext in $\mathfrak{A}$ exhibits maximal semantic ambiguity, this incurs storage *and* time cost $\mathrm{O}(|\mathfrak{N}| - |\mathfrak{G}|)$.
2. Parse all wikitext in $\mathfrak{G}$ to generate the archive-specific grammar, incurring storage *and* time cost $\mathrm{O}(|\mathfrak{G}|)$.
3. Reparse all cached wikitext given this grammar, incurring time cost $\mathrm{O}(|\mathfrak{N}| - |\mathfrak{G}|)$.

Then the optimized one-pass parser suffers worst-case storage $\mathrm{O}(|\mathfrak{N}| - |\mathfrak{G}|) + \mathrm{O}(|\mathfrak{G}|) = \mathrm{O}(|\mathfrak{N}|)$, substantially worse than that of naïve two-pass parsing, and time $\mathrm{O}(|\mathfrak{N}| - |\mathfrak{G}|) + \mathrm{O}(|\mathfrak{G}|) + \mathrm{O}(|\mathfrak{N}| - |\mathfrak{G}|) \simeq \mathrm{O}(|\mathfrak{N}|)$, equivalent to that of naïve two-pass parsing.

Both parsers assume no prior indexing of compressed article archive input. We note in passing that pseudo-indexing is technically feasible: present-day Wikimedia article archives are bzip2-compressed files internally partitioned into blocks of default size 900KB, which while disallowing random access to exact byte offsets do allow random access to exact block offsets [8]. Indexing article title to 2-element tuple (b, y), b the offset to the compressed block in which that article begins and y the offset to that article's first uncompressed byte in that block, during one-pass parsing *could* reduce the real-world storage cost (by avoiding caching) at some additional time cost (by forcing re-decompressed seeking of on-disk blocks). Further research required.

Consider English Wikipedia, whose 12 GB archive `enwiki-20101011-pages-meta-current.xml.bz2` uncompresses to approximately $160 - 230$ GB. Then worst-case storage $\mathrm{O}(|\mathfrak{N}|)$ is prohibitive on high-volume archives and there exist compelling incentives not to implement one-pass parsers *without also implementing pseudo-indexing*.

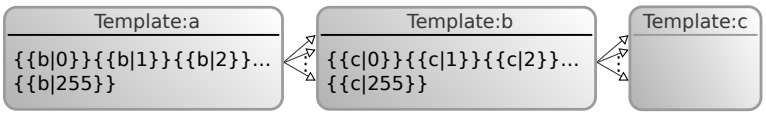

Figure 2: Article archive exhibiting our Denial-of-Service (DoS) exploit

As solution, see our solution to the problem of Section 3.3.

4 Worst-case exploitation

Denial-of-service (DoS) attacks render computing services unavailable to end users by exploiting hardware-, protocol- and application-level insecurities. Degradation-of-service (DegoS) attacks render such services non-performant by brute-force consumption of scarce computing resources (e.g., bandwidth, CPU load). As a practical demonstration, we now improve the worst-case article archive input of Section 3.2 into viable DoS and DegoS attacks on `MediaWiki` itself.

We tested these attacks against clean-room installation of the most recent official releases of `MediaWiki` (1.16.0), `MySQL` (5.1.50), `PHP` (5.3.3), the `Apache` HTTP Server (2.2.16) and `Linux` (2.6.36) as of this writing, where "clean-room" means:

1. No optional `MediaWiki` extensions and only those `PHP` extensions required by `MediaWiki`.
2. Default `MediaWiki`, `MySQL` and `PHP` settings.
3. Stock `Apache` HTTP Server and `Linux` kernel modules and settings.

We expect these attacks retroactively apply to *all* `MediaWiki` wikis regardless of version, configuration or system. However, we verified this neither locally or remotely against publicly accessible wikis. (In the latter case, doing so violates ISP and University acceptable use policies as well as constituting imprisonable offenses under domestic and international law).

4.1 DoS attack

While worst-case article archive input of Section 3.2 makes third-party parsing of article archives intractable, its memoization by the `MediaWiki` preprocessor makes this input inadequate for attacks on `MediaWiki` itself.

`MediaWiki` memoizes *all transclusions of the same template and same template parameters in the same article* to the first expansion of the transclusion, even with transclusions expanding differently! As example, a `Template:Yog` with wikitext `{{CURRENTTIME}}` *should* expand differently in an article with wikitext ''`{{Yog}}{{Yog}}{{Yog}}`'' when the current time in seconds changes between the first and second or second and third expansion of that template. Of course, this is not what happens; `MediaWiki` forcefully sets the second and third expansion to the first regardless.

`MediaWiki` memoization negates the usefulness of our prior worst-case input. However by the above invariant, `MediaWiki` memoizes no transclusions of the same template *and different template parameters* in the same article. Then altering this input so the last template in $|\mathfrak{T}|$ is empty and all transclusions in all other templates in $|\mathfrak{T}|$ are uniquely parametrized within their templates prevents memoization.

`PHP` prematurely terminates scripts exceeding its *max_execution_time*, defaulting to $30s$. So there exists some least total number of transclusions t_1 for which `MediaWiki` exceeds this setting. Testing shows $t_1 \in \left(2^{15}, 2^{16}\right]$ for our local installation, so assume $t_1 = 2^{16}$ for convenience. But $256^2 = 2^{16}$, so 2 templates of 256 transclusions each induces `PHP` to prematurely terminate `MediaWiki`. Figure 2 depicts these assumptions with arrows signifying transclusion.

Submitting templates ''`b`'' and ''`c`'' to the target `MediaWiki` wiki does not trigger the attack; submitting template ''`a`'' does. Then the attack consists of first submitting the former two templates once each, then repetitively resubmitting the latter template. Each resubmission starves the target `MediaWiki` wiki with near 100% CPU load for exactly 30s, after which `PHP` interrupts the submission, `MySQL` rolls back the transaction and `MediaWiki` resumes receiving queries at normal CPU load. Then the attacker resubmits template ''`a`'' and the attack resumes.

Each resubmission enjoys a payload of only 2.2KB. So the attack is inherently asymmetric: a milliseconds worth of effort on the attacking machine generates 30s of high CPU load on the attacked machine. But 16 articles of 2 transclusions each also induces `MediaWiki` to exceed `PHP`'s *max_execution_time* setting, so the payload is reducible to 15B. The simplicity of this asymmetry lends itself to anonymous distribution via decentralized botnets [15], thus extending its scope to (largely) non-targetable resilient attack networks.

4.2 DegoS attack

However, a subversive alternative suggests itself: induce the target `MediaWiki` wiki to spin-wait *itself*.

There exists some greatest total number of transclusions t_0 for which `MediaWiki` does not exceed `PHP`'s *max_execution_time*. Then $t_0 = t_1 - 1$ when t_1 is known or the greatest number t_1 is known to be strictly greater than when t_1 is not known. Larger values induce longer downtime, so the former is preferable. $t_1 > 2^{15}$ in our case, so let $t_0 = 2^{14} = 128^2$.

We measured 2 templates of 128 transclusions each to consume $28 - 30$s of wall clock time per submission

of template ‘‘a’’, allowing MySQL to successfully complete submission transactions. Then submitting template ‘‘a’’ of 128 transactions of ‘‘b’’, template ‘‘b’’ of 128 transactions of ‘‘c’’ and template ‘‘c’’ empty as before initiates this attack. The attacker identifies high-edit articles, then injects one malicious transclusion of template ‘‘a’’ into each article – preferably adjacent to or embedded in existing transclusions in each article *and/or* accompanied by one or more seemingly benign edits to each article as a cloaking measure. Since each transclusion expands to nothing, search engines show no evidence of the attack. Since each transclusion consists of only 5B in high-edit articles consisting of up to 32KB, each article shows little evidence of the attack. Since each transclusion expansion in each article consumes the maximal amount of wall clock time without triggering MediaWiki, PHP, Apache, or kernel defenses, all subsequent edits on each article suffer the same spin-wait, and the DegoS attack is described.

As solution, we recommend Wikimedia implement safeguards against transclusion fanout in the MediaWiki codebase *and that all third-party parsers immediately follow suite*.

5 Conclusion

Parsing Wikimedia article archives has been shown to be non-trivial. Worst-case article archive input of maximal recursive transclusion renders parsing computationally intractable. Worst-case article archive input of maximal semantic ambiguity renders one-pass parsing storage prohibitive. Average-case input of context-sensitive occlusions, non-expanded transclusions, non-canonical wikilinks and ambiguous disambiguations and interwikilinks complicates parser compliance irrespective of worst case complexity.

Susceptibility of MediaWiki wikis to transclusion-enabled DoS and DegoS attacks suggests transclusion-ignoring parsers to be fundamentally more secure than transclusion-compliant parsers. For safety, third-party parsers attempting to recursively expand tranclusions must duplicate existing provisions against transclusion abuse in the MediaWiki codebase *as well as devise new provisions against these novel attacks*.

Comparative reviewal of 10 contemporary wikitext parsers reveals widespread syntactic and semantic non-compliance. As expected, the least compliantly parsed semantics match those responsible for aforementioned worst-case bounds. We recommend a parser-focused redress of Wikimedia article archive policies and of the MediaWiki wikitext grammar, as follows: **(a)** encode in-occlusion punctuation as HTML entities; **(b)** declare disambiguations via #DISAMBIG pragmas; **(c)** recursively expand transclusions in-place; **(d)** publicize safeguards against transclusion abuse.

References

[1] Various authors. MediaWiki 1.16.0. http://www.mediawiki.org, July 2010.

[2] Various authors. Pywikipedia (svn revision 8616). http://pywikipediabot.sourceforge.net, October 2010.

[3] Various authors. Wikipedia article size. http://en.wikipedia.org/wiki/Wikipedia:Article_size, September 2010.

[4] Various authors. Wikipedia disambiguation pages aren't articles. http://en.wikipedia.org/wiki/Wikipedia:Disambiguation_pages_aren't_articles, May 2010.

[5] Brian W. Curry. Yppy 0.0.8. http://bitbucket.org/leycec/yppy, October 2010.

[6] Bryan Ford. Parsing expression grammars: a recognition-based syntactic foundation. *ACM SIGPLAN Notices*, Volume 39, Number 1, pages 111–122, 2004.

[7] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, January 2007.

[8] Interiot. Random access [*on*] bzip2[-*compressed article archives*]. http://meta.wikimedia.org/wiki/User:Interiot/Random_access_bzip2, May 2007.

[9] Magnus Manske. Wiki2XML (svn revision 56074). http://toolserver.org/~magnus/wiki2xml/w2x.php, September 2009.

[10] David Milne. An open-source toolkit for mining Wikipedia. In *New Zealand Computer Science Research Student Conference (NZCSRSC) 2009 Proceedings*, Auckland, New Zealand, April 2009.

[11] PediaPress. mwlib (git revision 7dbed545c6cd). http://code.pediapress.com/wiki/wiki/mwlib, October 2010.

[12] Tyler Riddle. Parse::MediaWikiDump 1.0.6_01. http://search.cpan.org/dist/Parse-MediaWikiDump, June 2010.

[13] Ed Summers and Brian Cassidy. WWW::Wikipedia 1.97. http://search.cpan.org/dist/WWW-Wikipedia/, June 2010.

[14] Timwi and Magnus Manske. FlexBisonParse (svn revision 71620). http://svn.wikimedia.org/viewvc/mediawiki/trunk/parsers/flexbisonparse, August 2010.

[15] Ryan Vogt, John Aycock and Michael J. Jacobson, Jr. Army of botnets. In *Network and Distributed System Security Symposium*. Internet Society, 2007.

[16] Torsten Zesch, Christof Müller and Iryna Gurevych. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation*, Marrakech, Morocco, May 2008.

Appendix

This Appendix presents a partially compliant wikitext grammar for programmatically generating article archive parsers. We showed this grammar to be context-sensitive in Section 3.1, so context-free metalanguages such as Backus-Naur Form (BNF) do not apply. Instead, we apply the recently developed Parsing Expression Grammar (PEG) metalanguage to context-sensitively describe this grammar [6].

Page constraints, readability concerns and initial application to the production of wikilink graphs make this PEG only partially compliant. It parses most *inter*-article semantics (relating two or more articles) but no *intra*-article semantics (relating the structure within an article). This includes all *categorization*, *disambiguation*, *redirect*, *wikilink*, and *interwikilink* semantics as well as some *transclusion* semantics, and none else.

Productions in **boldface** are language-specific. They remain unset until after parsing revelant metadata relevant from the archive preamble. When the article archive preamble fails to provide such metadata (e.g., for the ***wikilink_day_month_month*** production), a parser either requires a priori knowledge of the language under inspection or must delete all such productions and productions requiring these productions (e.g., the *wikilink_day_month* production). By default, language-specific productions in the grammar below assume English Wikipedia.

Productions in SMALL CAPS are archive-specific. They remain unset until after parsing relevant articles from the archive body. Such articles are *grammar-generative*, in that their wikitext assists the parser to generate itself. Prior to parsing the set of all grammar-generative articles, the grammar is incompletely generated. In this incomplete state, wikitext potentially matching one or more unset productions cannot be reliably consumed and must either be discarded or cached for subsequent reparsing. Parsers performing the former are necessarily two-pass; parsers performing the latter are one-pass. In either case, compliant parsers must iteratively bootstrap themselves in a language-specific manner to eventual completion.

Productions split by "←" are directly context-free. Productions split by "↩" are directly context-sensitive. Conveniently, most productions are the former.

We invite the interested reader to review `Yppy`[5], open-source graph theoretic software implementing this formalism as Python-compatible regular expressions.

wikitext ↩ *redirect*? (*occlusion* | *transclusion* | *wikilink* | .)

redirect ← *redirect_prefix redirect_begin wikilink wikilink_end*

redirect_prefix ← *whitespace* redirect_magic_word whitespace** `":"`? *whitespace**

redirect_magic_word ← `"#" [Rr] [Ee] [Dd] [Ii] [Rr] [Ee] [Cc] [Tt]`

redirect_begin ↩ *wikilink_begin* | . *redirect_begin*

occlusion ← *includeonly_tag* | *nowiki_tag* | *timeline_tag* | *comment_tag* | *math_tag* | *pre_tag* | *source_tag* | *syntaxhighlight_tag*

includeonly_tag ← *includeonly_tag_open includeonly_tag_body*

includeonly_tag_open ← *tag_begin* `"includeonly"` *tag_end*

includeonly_tag_body ↩ *tag_begin* `"/includeonly"` *tag_end* | . *includeonly_tag_body*

nowiki_tag ← *nowiki_tag_open nowiki_tag_body*

nowiki_tag_open ← *tag_begin* `"nowiki"` *tag_end*

nowiki_tag_body ↩ *tag_begin* `"/nowiki"` *tag_end* | . *nowiki_tag_body*

timeline_tag ← *timeline_tag_open timeline_tag_body*

timeline_tag_open ← *tag_begin* `"timeline"` *tag_end*

timeline_tag_body ↩ *tag_begin* `"/timeline"` *tag_end* | . *timeline_tag_body*

comment_tag ← *comment_tag_open comment_tag_body*

comment_tag_open ← *html_entity_less_than* `"!-"`

comment_tag_body ↩ `"-"` *html_entity_greater_than* | . *comment_tag_body*

math_tag ← *math_tag_open math_tag_body*

math_tag_open ← *tag_begin* `"math"` *tag_end_attributes*

math_tag_body ↩ *tag_begin* `"/math"` *tag_end* | . *math_tag_body*

pre_tag ← *pre_tag_open pre_tag_body*

pre_tag_open ← *tag_begin* `"pre"` *tag_end_attributes*

pre_tag_body ↩ *tag_begin* `"/pre"` *tag_end* | . *pre_tag_body*

source_tag ← *source_tag_open source_tag_body*

source_tag_open ← *tag_begin* `"source"` *tag_end_attributes*

source_tag_body ↩ *tag_begin* `"/source"` *tag_end* | . *source_tag_body*

syntaxhighlight_tag ← *syntaxhighlight_tag_open syntaxhighlight_tag_body*

syntaxhighlight_tag_open ← *tag_begin* `"syntaxhighlight"` *tag_end_attributes*

syntaxhighlight_tag_body ↩ *tag_begin* `"/syntaxhighlight"` *tag_end* | . *syntaxhighlight_tag_body*

tag_begin ← *html_entity_less_than*

tag_end ← *whitespace* html_entity_greater_than*

tag_end_attributes ← *tag_end_attribute* whitespace* html_entity_greater_than*

tag_end_attribute ← *whitespace*+ `[a-zA-Z] ([a-zA-Z0-9] | "-" | "_" | ".")* '='" (!'"' .)* '"'`

transclusion ← *transclusion_begin transclusion_body transclusion_end*

transclusion_begin ← `"{{"` *wikilink_whitespace* transclusion_begin_subst*?

transclusion_begin_subst ← `[Ss] [Uu] [Bb] [Ss] [Tt] ":"`

transclusion_end ← *wikilink_whitespace** `"}}"`

transclusion_body ↩ *disambiguation* | *transclusion* | . *transclusion_body*

DISAMBIGUATION ← Unset prior to parsing the ‘‘`MediaWiki:Disambiguationspage`’’ article.

wikilink ← *wikilink_begin wikilink_type wikilink_end*

wikilink_begin ← *wikilink_open wikilink_whitespace**

wikilink_end ← *wikilink_whitespace* wikilink_close*

wikilink_open ← `"[["`

wikilink_close ← `"]]"`

wikilink_whitespace ← `" " | "_"`

wikilink_type ↩ *wikilink_category* | *wikilink_day_month* | *wikilink_iso_8601* | *wikilink_normal*

wikilink_category ← *wikilink_namespace_category wikilink_qualifier_end wikilink_body*

wikilink_day_month ← *wikilink_day_month_day wikilink_whitespace* wikilink_day_month_month*

wikilink_day_month_day ← `[0-9] [0-9]`?

wikilink_day_month_month ← `[Jj] "anuary" | [Ff] "ebruary" | [Mm] "arch" | [Aa] "pril" | [Mm] "ay" | [Jj] "une" | [Jj] "uly" | [Aa] "ugust" | [Ss] "eptember" | [Oo] "ctober" | [Nn] "ovember" | [Dd] "ecember"`

wikilink_iso_8601 ← *wikilink_iso_8601_year* `"-"` *wikilink_iso_8601_month* `"-"` *wikilink_iso_8601_day*

wikilink_iso_8601_year ← `[0-9] [0-9] [0-9] [0-9]`

wikilink_iso_8601_month ← `[0-9] [0-9]`

wikilink_iso_8601_day ← `[0-9] [0-9]`

wikilink_normal ← (*wikilink_subpage_prefix* | *wikilink_qualifier*)? *wikilink_body*

wikilink_subpage_prefix ← `"/"` *wikilink_whitespace**

wikilink_qualifier ← *wikilink_qualifier_prefix*? *wikilink_qualifier_body wikilink_qualifier_end*

wikilink_qualifier_prefix ← `":"` *wikilink_whitespace**

wikilink_qualifier_body ← *wikilink_namespace* | *wikilink_interlanguage* | *wikilink_interwiki*

wikilink_qualifier_end ← *wikilink_whitespace** `":"` *wikilink_whitespace**

wikilink_namespace ← *wikilink_namespace_main* | *wikilink_namespace_talk* |
wikilink_namespace_user | *wikilink_namespace_user_talk* |
wikilink_namespace_wikipedia | *wikilink_namespace_wikipedia_talk* |
wikilink_namespace_file | *wikilink_namespace_file_talk* |
wikilink_namespace_mediawiki | *wikilink_namespace_mediawiki_talk* |
wikilink_namespace_template | *wikilink_namespace_template_talk* |
wikilink_namespace_help | *wikilink_namespace_help_talk* |
wikilink_namespace_category | *wikilink_namespace_category_talk* |
wikilink_namespace_special | *wikilink_namespace_media*

wikilink_namespace_main ← `[Mm] [Aa] [Ii] [Nn]`

wikilink_namespace_talk ← `[Tt] [Aa] [Ll] [Kk]`

wikilink_namespace_user ← `[Uu] [Ss] [Ee] [Rr]`

wikipedia_namespace_user_talk ← *wikilink_namespace_user wikilink_namespace_talk_end*

wikilink_namespace_wikipedia ← `[Ww] [Ii] [Kk] [Ii] [Pp] [Ee] [Dd] [Ii] [Aa]` |
`[Pp] [Rr] [Oo] [Jj] [Ee] [Cc] [Tt]` |
`[Ww] [Pp]` | `[Ww]`

wikilink_namespace_wikipedia_talk ← (`[Ww] [Ii] [Kk] [Ii] [Pp] [Ee] [Dd] [Ii] [Aa]` |
`[Pp] [Rr] [Oo] [Jj] [Ee] [Cc] [Tt]`) *wikilink_namespace_talk_end* |
`[Ww] [Tt]`

wikilink_namespace_file ← `[Ff] [Ii] [Ll] [Ee]` | `[Ii] [Mm] [Aa] [Gg] [Ee]`

wikilink_namespace_file_talk ← *wikilink_namespace_file wikilink_namespace_talk_end*

wikilink_namespace_mediawiki ← `[Mm] [Ee] [Dd] [Ii] [Aa] [Ww] [Ii] [Kk] [Ii]`

wikilink_namespace_mediawiki_talk ← *wikilink_namespace_mediawiki wikilink_namespace_talk_end*

wikilink_namespace_template ← `[Tt] [Ee] [Mm] [Pp] [Ll] [Aa] [Tt] [Ee]`

wikilink_namespace_template_talk ← *wikilink_namespace_template wikilink_namespace_talk_end*

wikilink_namespace_help ← `[Hh] [Ee] [Ll] [Pp]`

wikilink_namespace_help_talk ← *wikilink_namespace_help wikilink_namespace_talk_end*

wikilink_namespace_category ← `[Cc] [Aa] [Tt] [Ee] [Gg] [Oo] [Rr] [Yy]`

wikilink_namespace_category_talk ← *wikilink_namespace_category wikilink_namespace_talk_end*

wikilink_namespace_special ← `[Ss] [Pp] [Ee] [Cc] [Ii] [Aa] [Ll]`

wikilink_namespace_media ← `[Mm] [Ee] [Dd] [Ii] [Aa]`

wikilink_namespace_talk_end ← *wikilink_whitespace+ wikilink_namespace_talk*

WIKILINK_INTERLANGUAGE ← Unset prior to parsing the ‘‘`List of Wikipedias`’’ article.

WIKILINK_INTERWIKI ← Unset prior to parsing the ‘‘`Meta:Interwiki map`’’ article.

wikilink_body ← (!*wikilink_invalid_char* .)+ *wikilink_anchor*? *wikilink_label*?

wikilink_anchor ← *wikilink_whitespace** `"#"` (!*wikilink_invalid_char* .)*

wikilink_label ← *wikilink_whitespace** `"|"` (!*wikilink_label_invalid_char* .)*

wikilink_invalid_char ← `"#" | "<" | ">" | "[" | "]" | "|" | "{" | "}" | "\t" | "\n"`

wikilink_label_invalid_char ← *wikilink_close* | `"\t" | "\n"`

html_entity_less_than ← `"<"`

html_entity_greater_than ← `">"`

whitespace ← `" " | "\t" | "\n"`