Improving Pattern Matching Performance in XSLT

John Lumley

jωL Research
Saxonica

Michael Kay

Saxonica

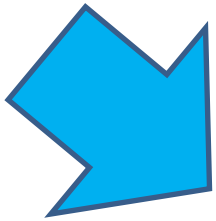
Synopsis

Some XSLT frameworks use lots of generic pattern templates

*[*predicate*]

with high pattern-matching costs

Investigation by Saxonica Ltd.



Improving performance for these:

- Investigating the pattern matching
- Common pattern preconditions
- Other 'oracle' possibilities
- Configuring such tuning

introductory apologies

- I have assumed you have some familiarity with XSLT

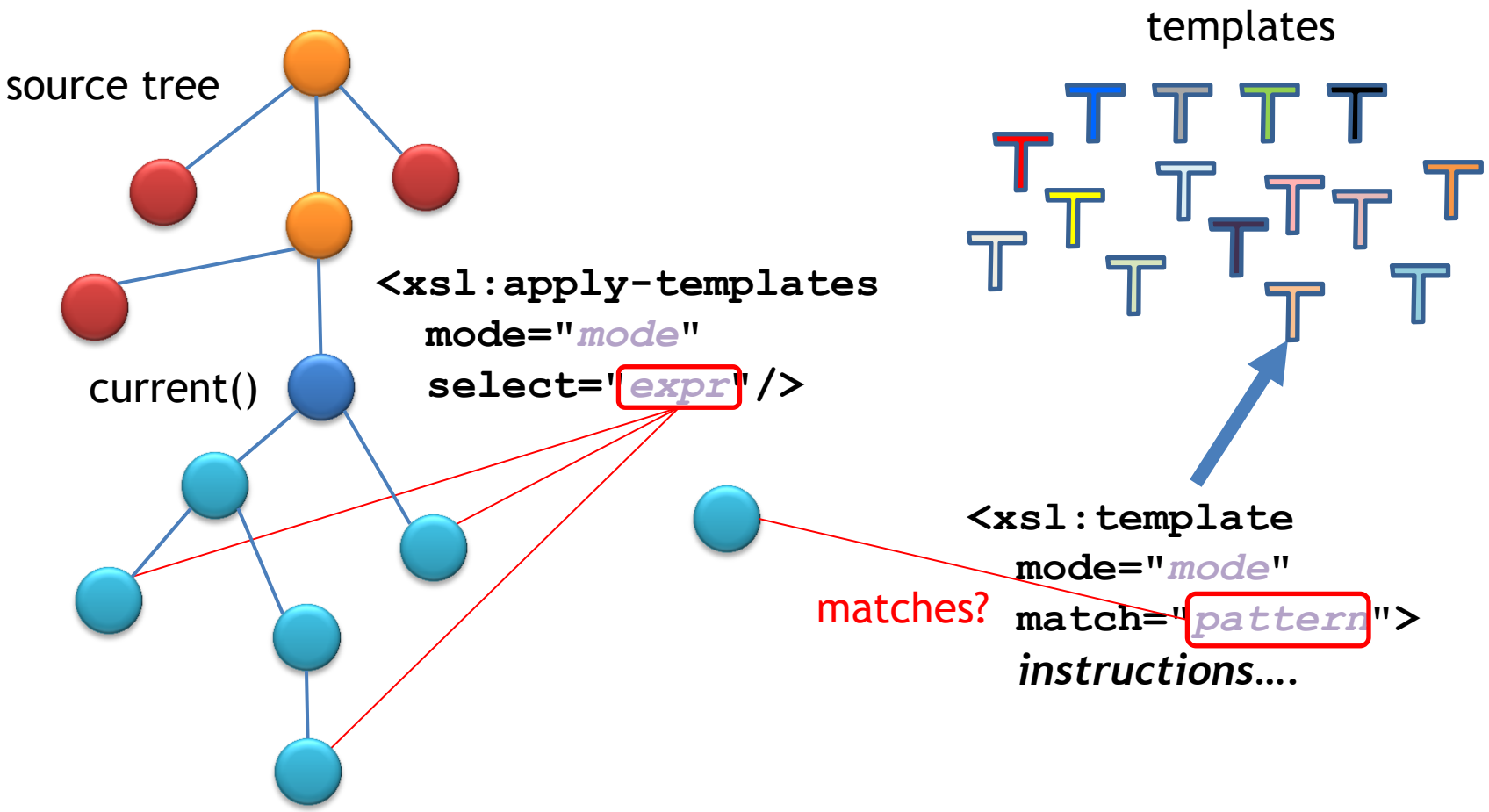
If not, then this talk might still amuse you with lots of graphs & pictures

- We discuss specific XSLT stylesheets (***DITA-OT***) operating on a particular XSLT engine (***Saxon***)

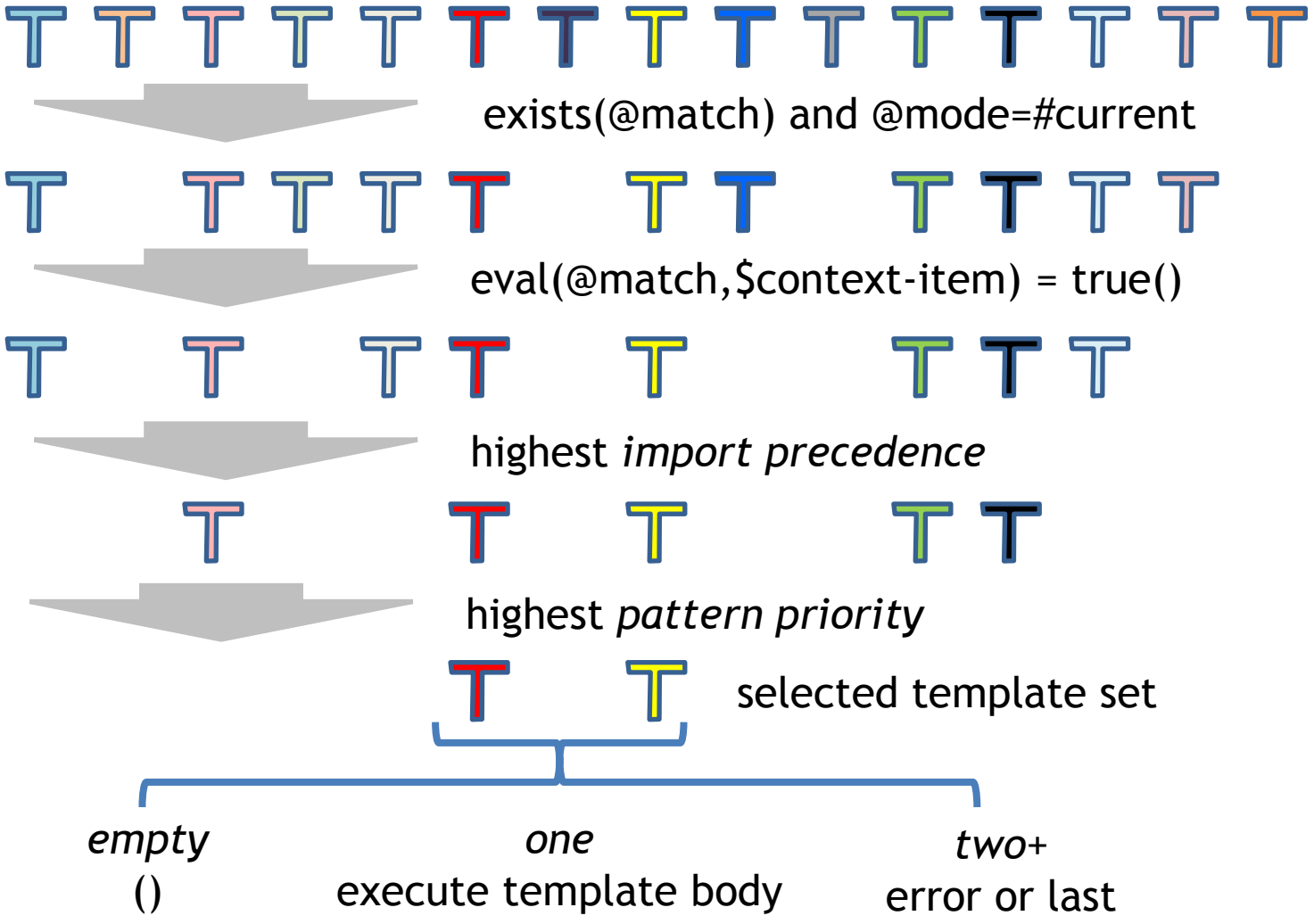
As the Americans caution:

your mileage may vary

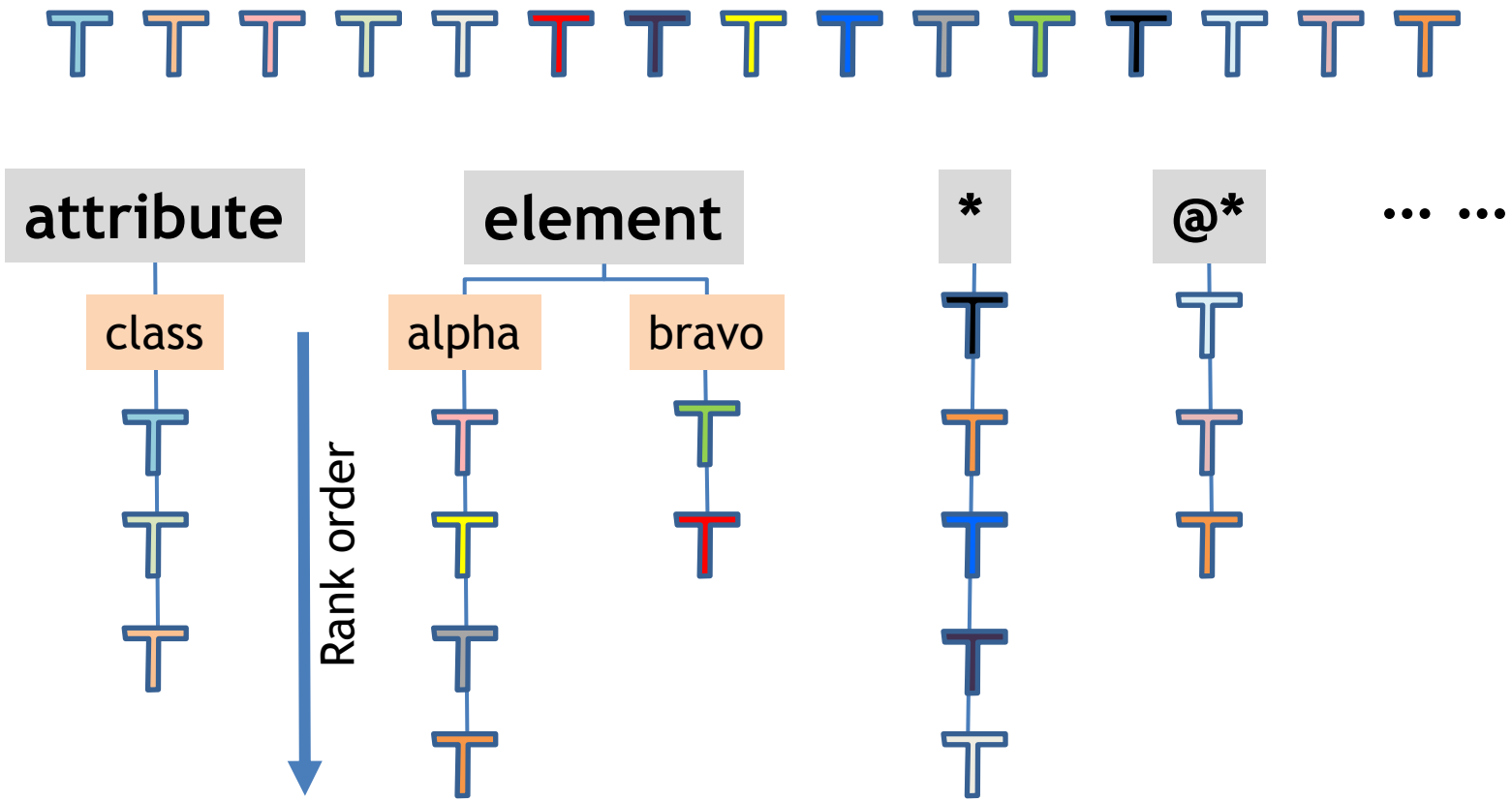
XSLT push operation



XSLT 'push' templates



What Saxon does



Differing vocabulary/framework architectures – *DocBook*

```
<d:itemizedlist>
   <d:listitem>
        <d:para>Suspending rule ambiguity
        checking.</d:para>
   </d:listitem>…

<xsl:template
    match="d:itemizedlist/d:listitem">

   … …
```

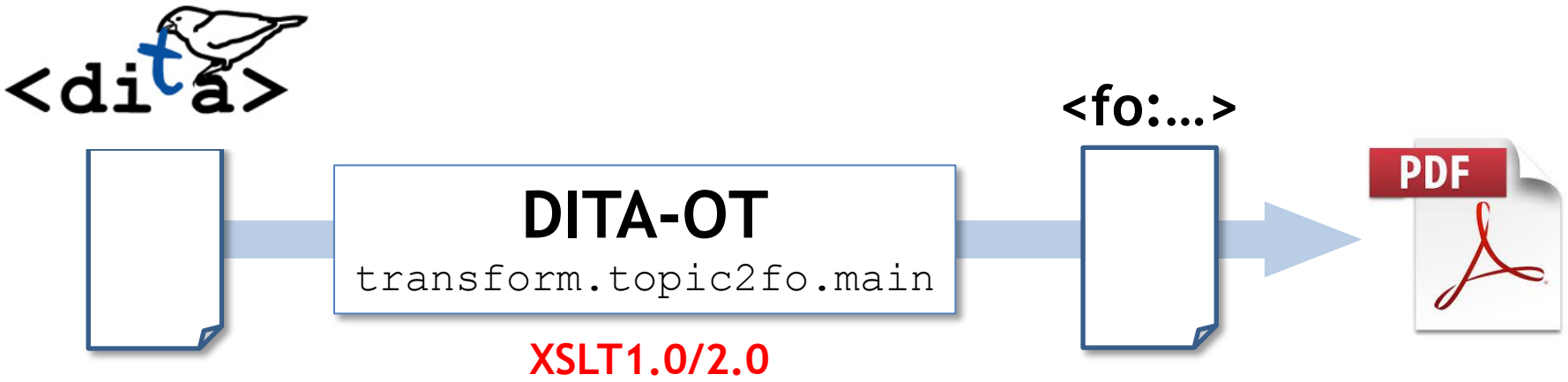
Differing vocabulary/framework architectures - *DITA*


```
<ul class="- topic/ul ">
    <li class="- topic/li ">
        Regeneration parts</li>…

<codeph class="+ topic/ph pr-d/codeph "…

<xsl:template match="
  *[contains(@class, ' topic/ul ')]/
  *[contains(@class, ' topic/li ')]">
  … …
```

A sample transformation

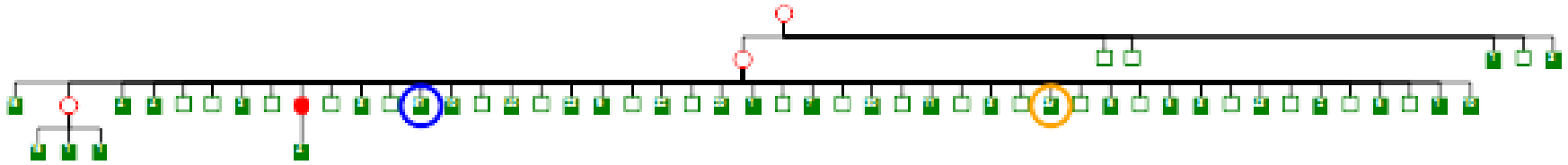


- 2.66 MB
- XML tree:
  - 13,066 elements
  - 46,831 attributes
  - 6,093 text

- 58 source files
- 70 modes
- Templates:
  - 418 pattern (258 #default)
  - 155 named

- 19,441 elements
- 91,048 attributes
- 6,140 text

80 pages
262 tables
4,8673 cells

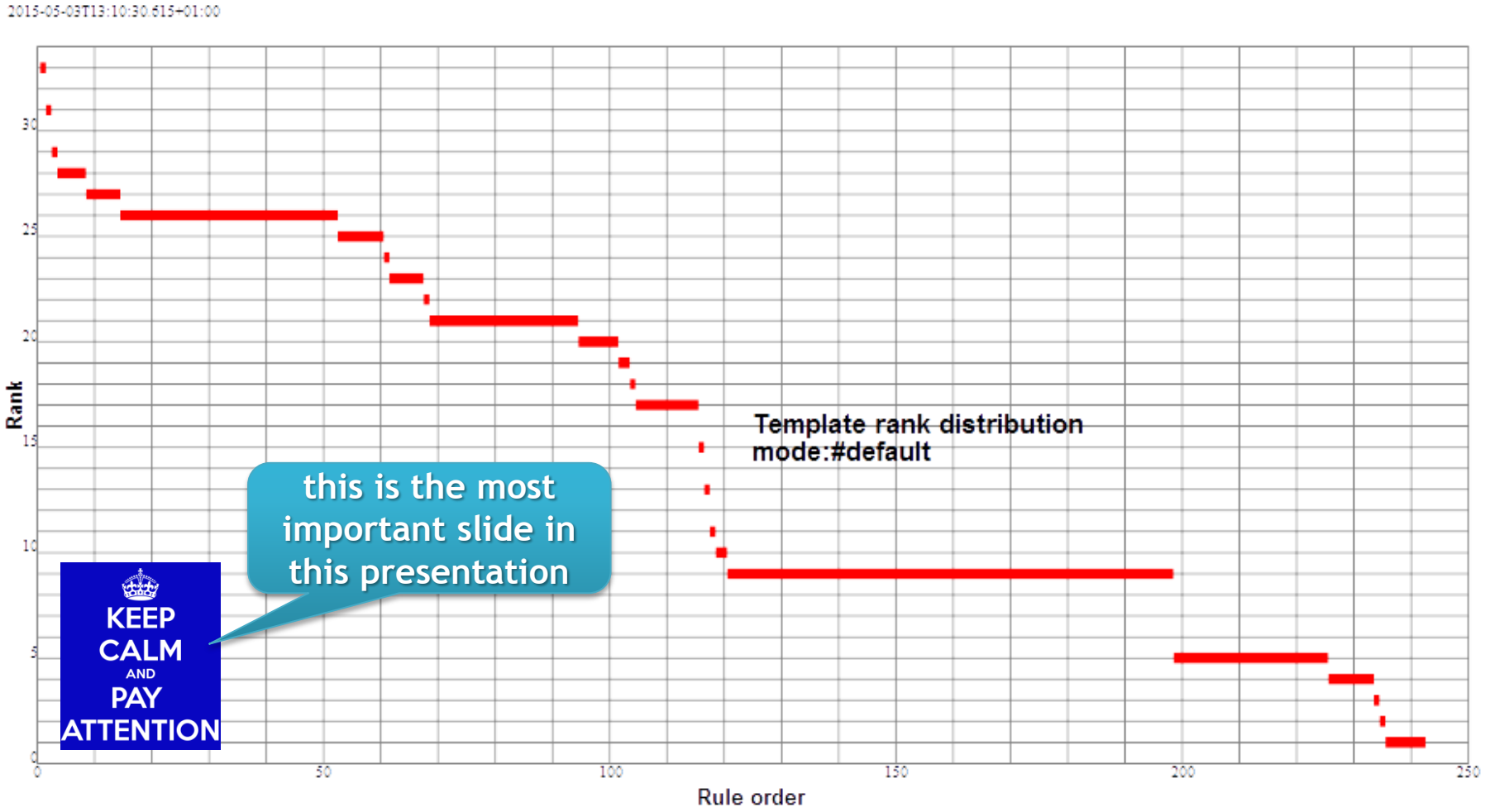


Significant Modes

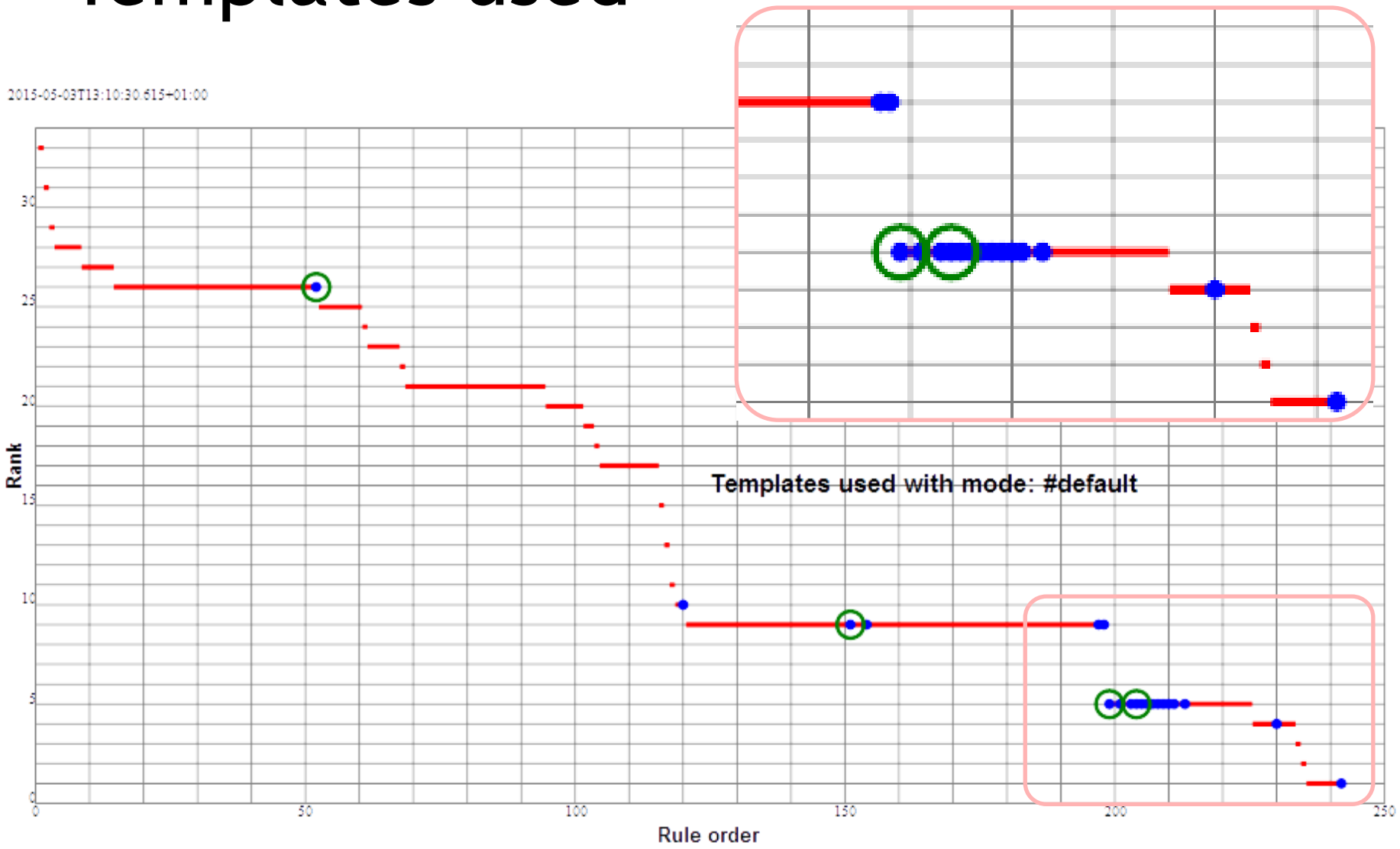
Mode	Purpose	invocations		time	
		#	%	/ ms	%
#default	General	13,095	17.2	4,330	97.8
toc	Table of Contents	22,088	29.1	51	1.1
bookmark	Bookmarks	37,752	49.7	33	0.8
	all templates	75,950			

Mode	# template patterns in mode			#templates matched
	element(*)	element(named)	attribute(named)	
#default	240	19	8	39
toc	2	4	0	3
bookmark	2	5	0	3

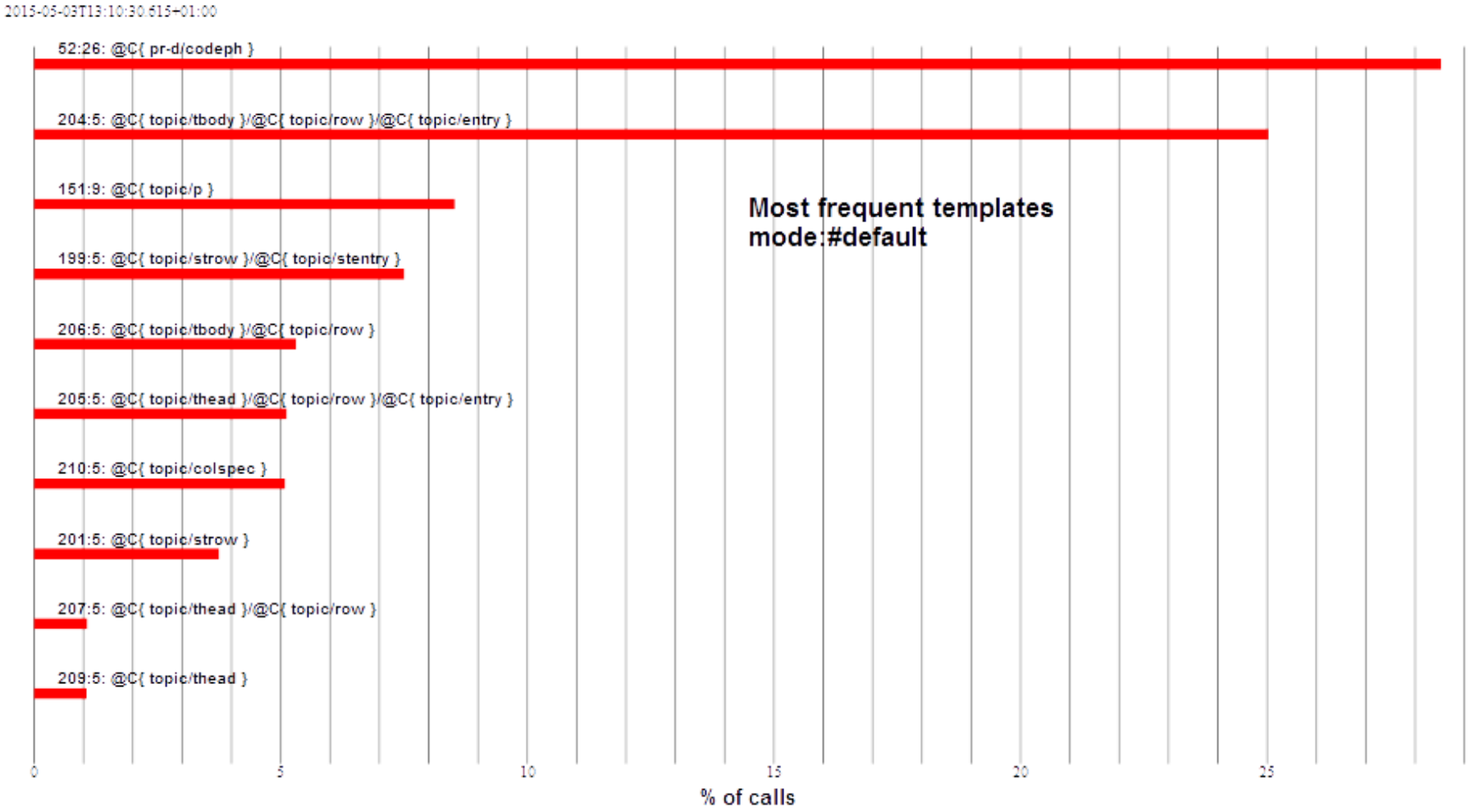
Template 'Rank'



Templates used



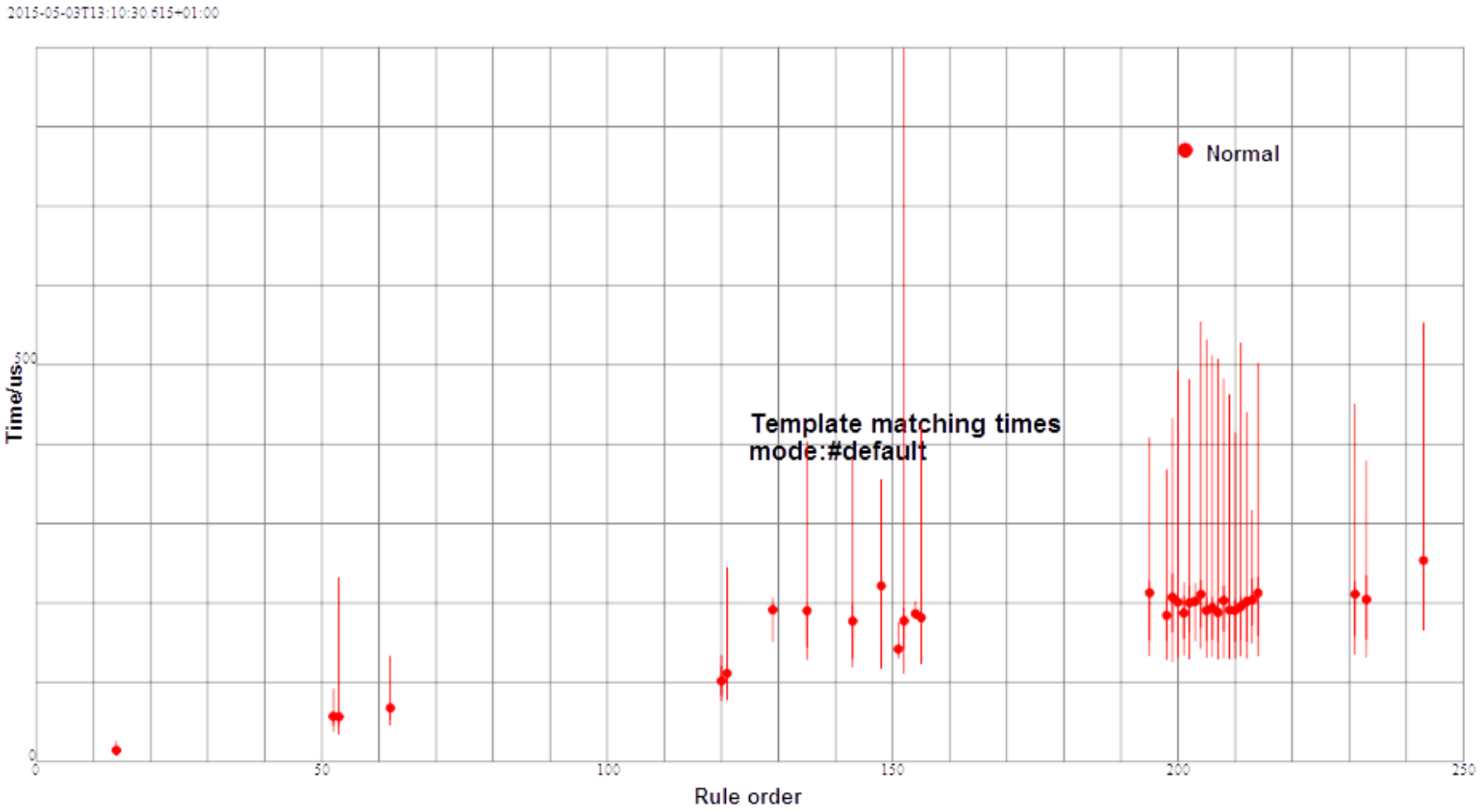
Most frequent templates



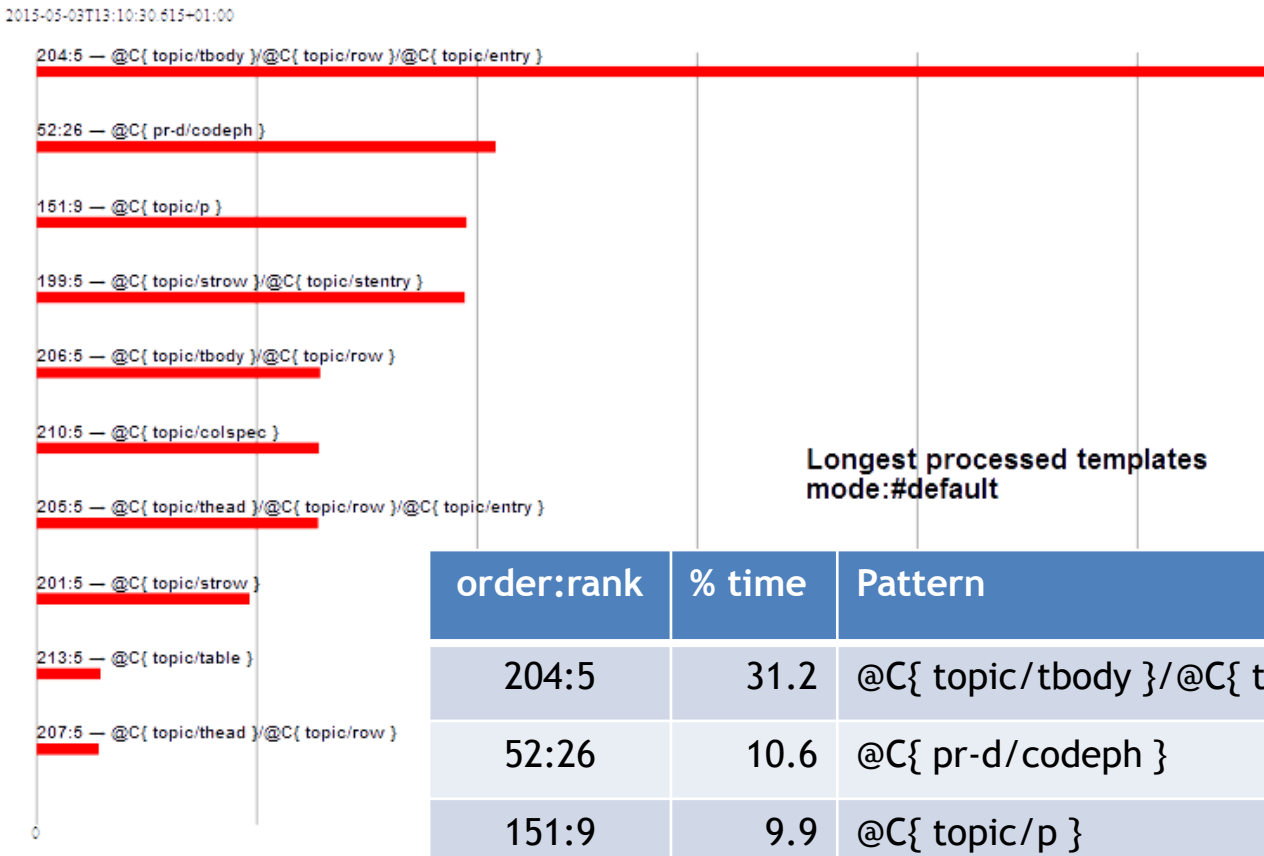
Frequent patterns, mode `#default`

Order	Rank	%calls	Pattern
52	26	28.5	*[contains(@class,' pr-d/codeph ')]
204	5	25.0	*[contains(@class,' topic/tbody ')]/ *[contains(@class,' topic/row ')]/ *[contains(@class,' topic/entry ')]
151	9	8.5	*[contains(@class,' topic/p ')]
199	5	7.5	*[contains(@class,' topic/strow ')]/ *[contains(@class,' topic/stentry ')]
206	5	5.3	*[contains(@class,' topic/tbody }/ *[contains(@class,' topic/row }
205	5	5.1	*[contains(@class,' topic/thead ')]/ *[contains(@class,' topic/row ')]/ *[contains(@class,' topic/entry ')]

Detailed time measurement

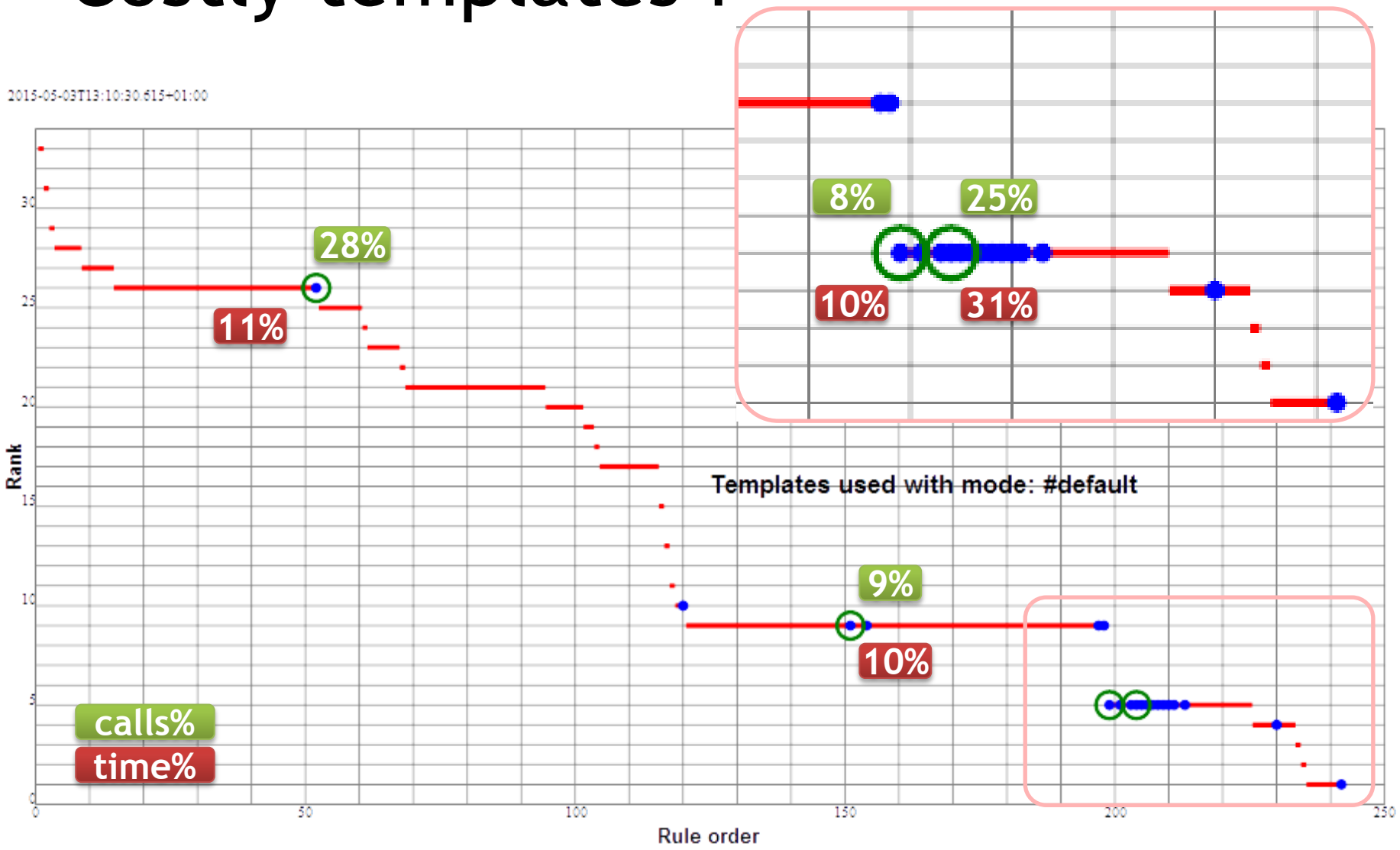


Most time-expensive patterns

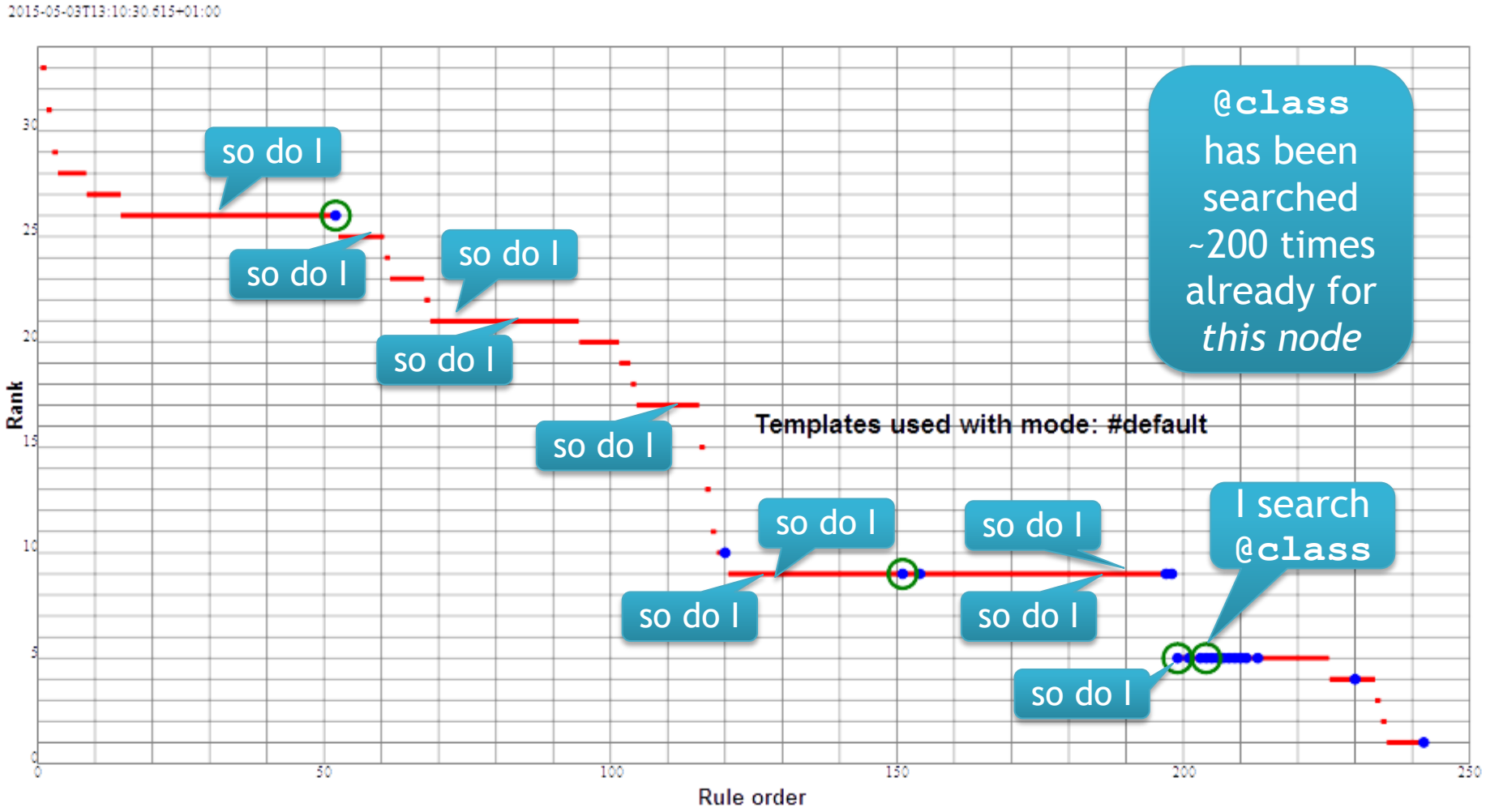


order:rank	% time	Pattern
204:5	31.2	@C{ topic/tbody }/@C{ topic/row }/@C{ topic/entry }
52:26	10.6	@C{ pr-d/codeph }
151:9	9.9	@C{ topic/p }
199:5	9.9	@C{ topic/strow }/@C{ topic/stentry }

Costly templates i



Costly templates ii



Can we improve?

- Rule preconditions — partitioning large rule sets by common (boolean) conditions
- Using *oracle guarantees*, shortcuts not applicable to all stylesheets:
  - Exploiting template mutual exclusivity
  - Pre-processing significant data
  - Pattern rewrites
- Configuring stylesheet execution

Common preconditions

- **`chapter/title[`*`condition1`*`], chapter/title[`*`condition2`*`], chapter/para, chapter/section ...`**
- **`exists(parent::chapter)` → `chapter/title[`*`condition1`*`], chapter/title[`*`condition2`*`], chapter/para, chapter/section ...`**
- **pre: `exists(parent::chapter)` → `title[`*`condition1`*`],title[`*`condition2`*`], para, section ...`**

Preconditions for DITA-OT

`exists(@class)` ✘ they *all* have one

`contains(@class, string`$_i$`)` ✘ very little commonality

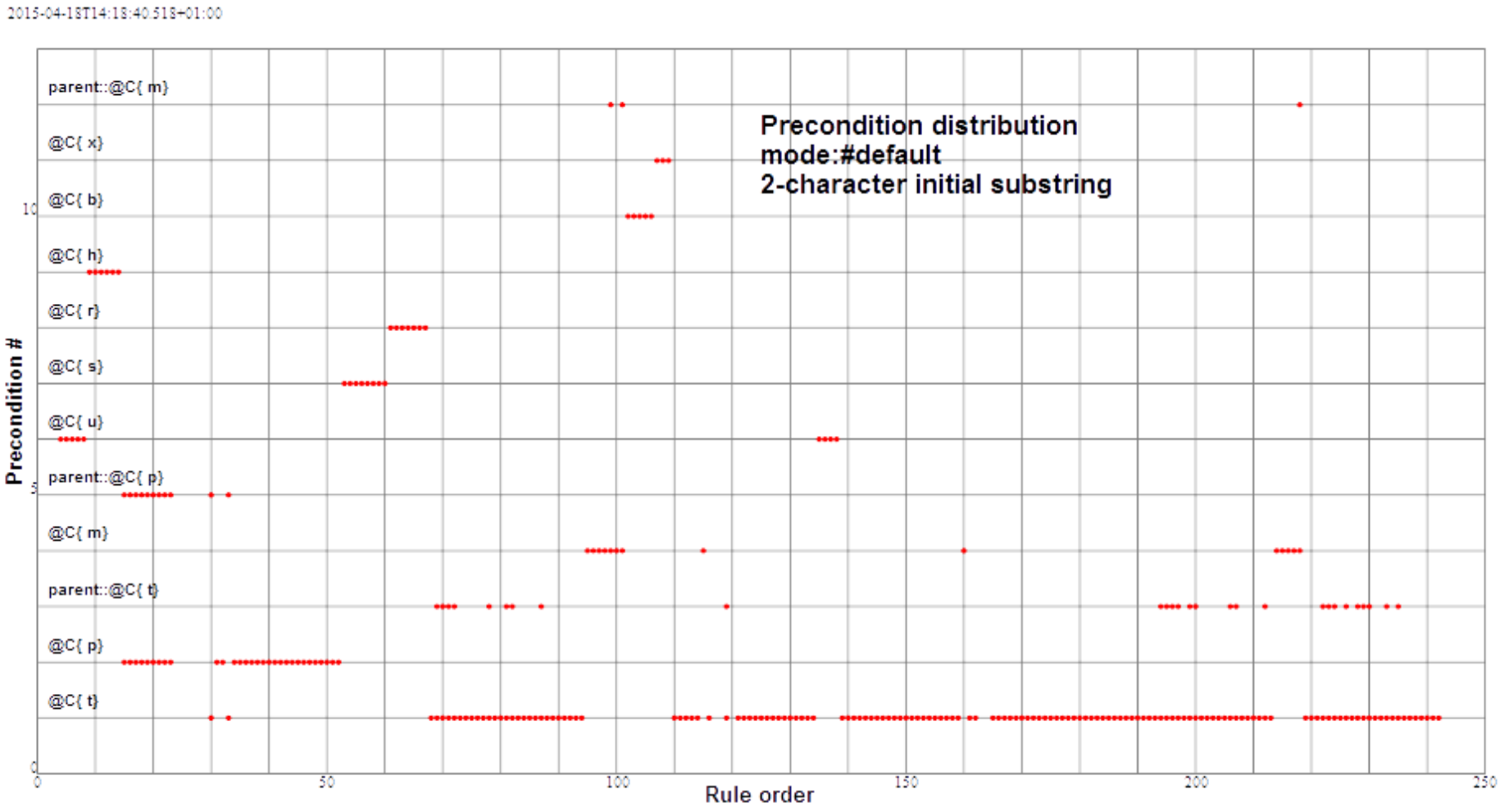
GOAL: p preconditions each shared by ~m patterns
'minimum work': $p \approx m \approx \sqrt{N}$

precondition-for `(contains(@class, string`$_i$`))` →
`contains(@class, any-substring-of(string`$_i$`))`

Initial Substring size	1	2	3-5	6	7	8
# preconditions	1	12	14	16	46	75
Largest set	250	146	121	121	121	17

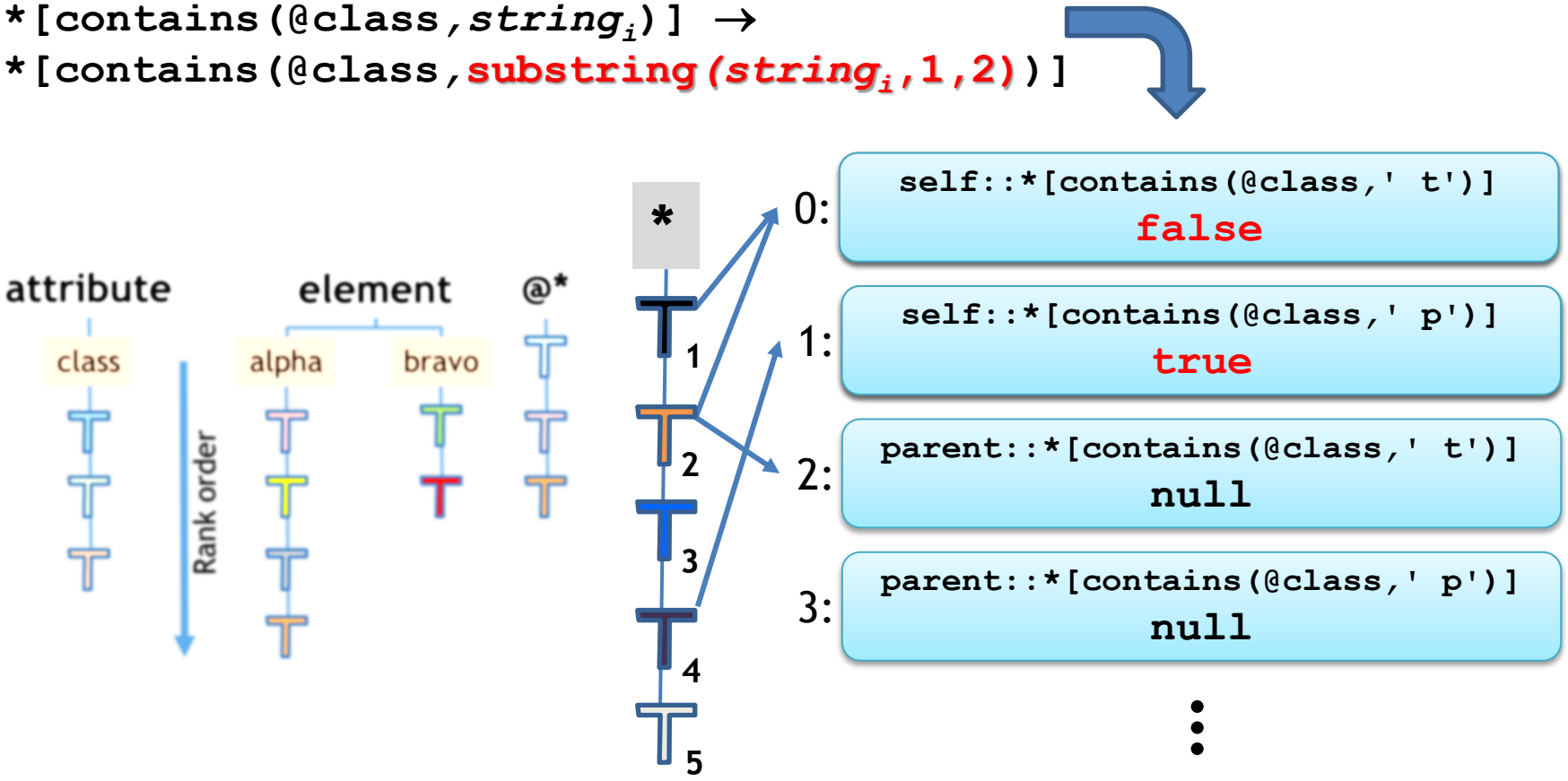
`contains(@class,'abcdef')` **&&**
pre:`contains(@class,'abc')` ✘→ `contains(@class,'def')`

Substring precondition distribution

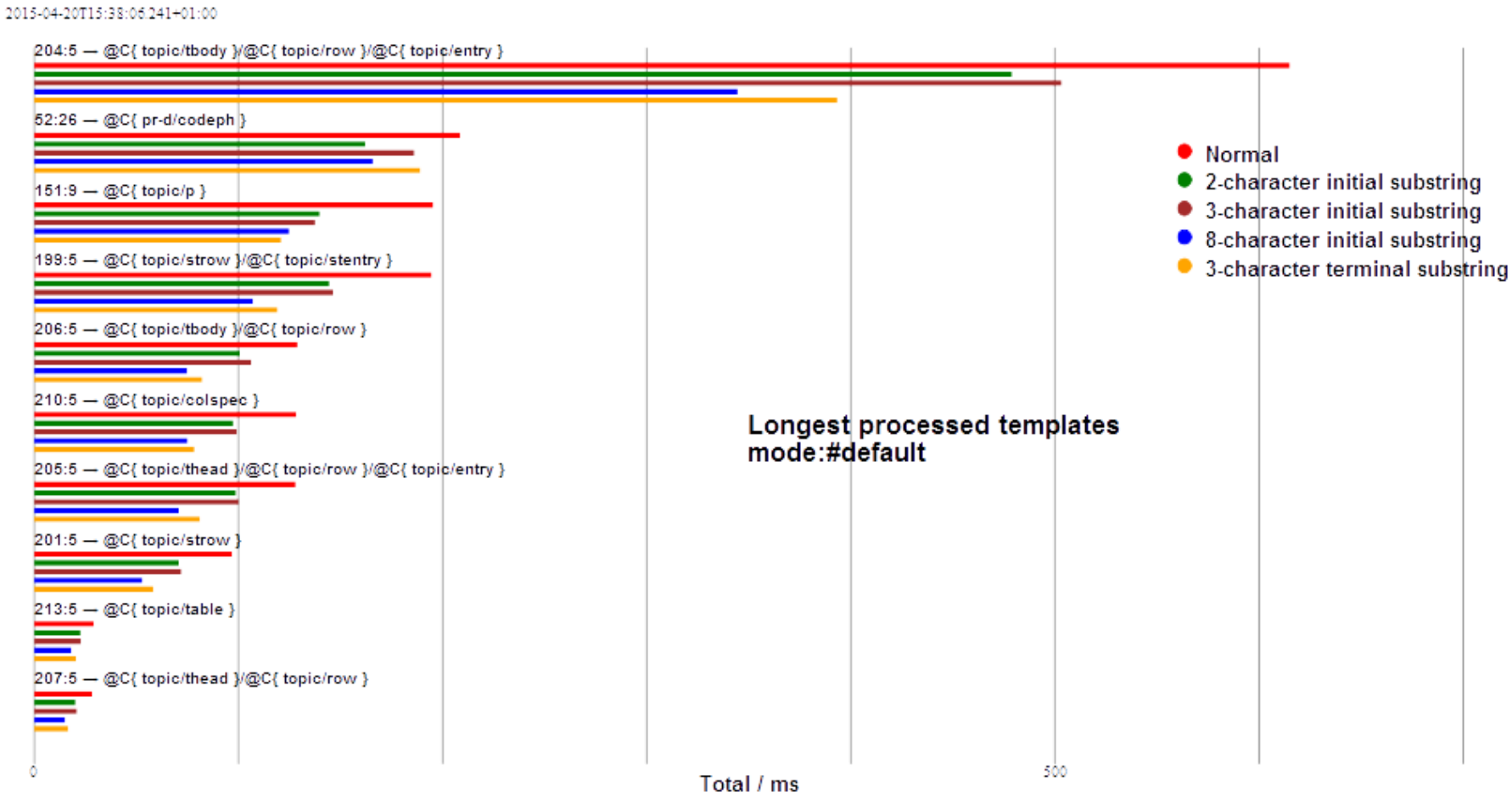


Implementing preconditions

*[contains(@class,$string_i$)] →
*[contains(@class,substring($string_i$,1,2))]



Substring preconditions



Consulting the *oracle*

- Reassurances as practical truths, not applicable to all stylesheets:
  - Mutual exclusivity of templates:
    - Suspending rule ambiguity checks
    - Reordering templates & imports
  - Pre-tokenizing significant data

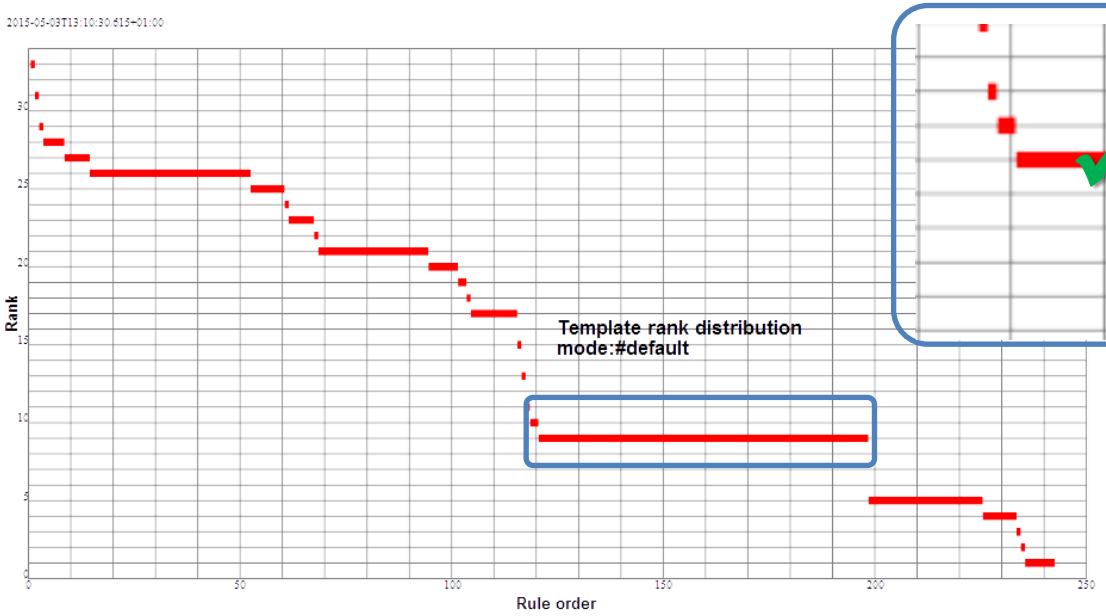
Mutual exclusivity: 'Un-disambiguating' rules

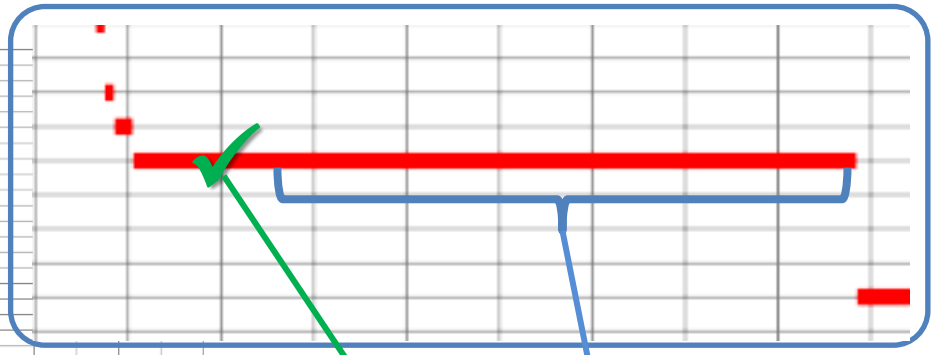
selected template set

empty
()

one
execute template body

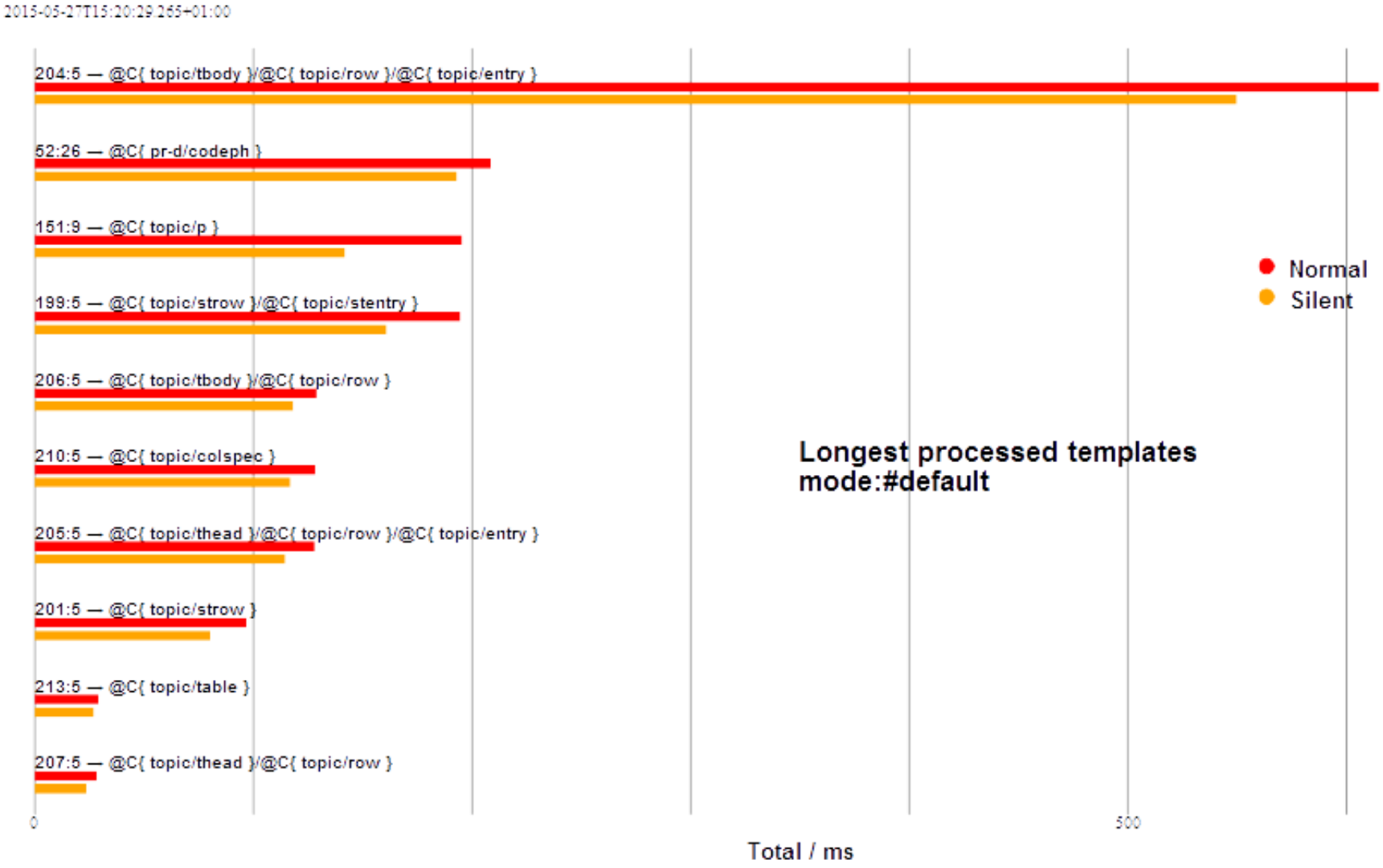
two+
error or ***last***



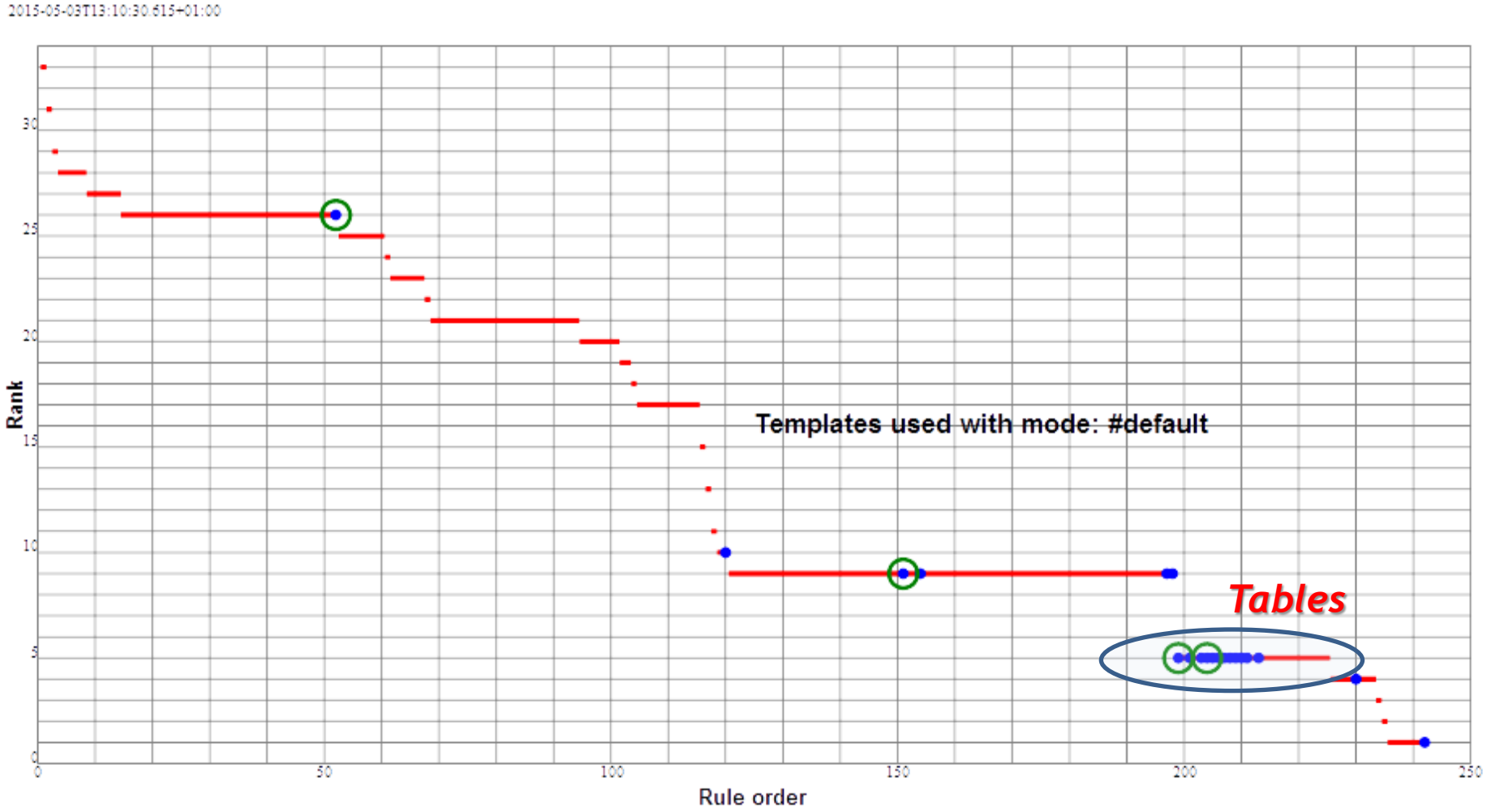


Match this…

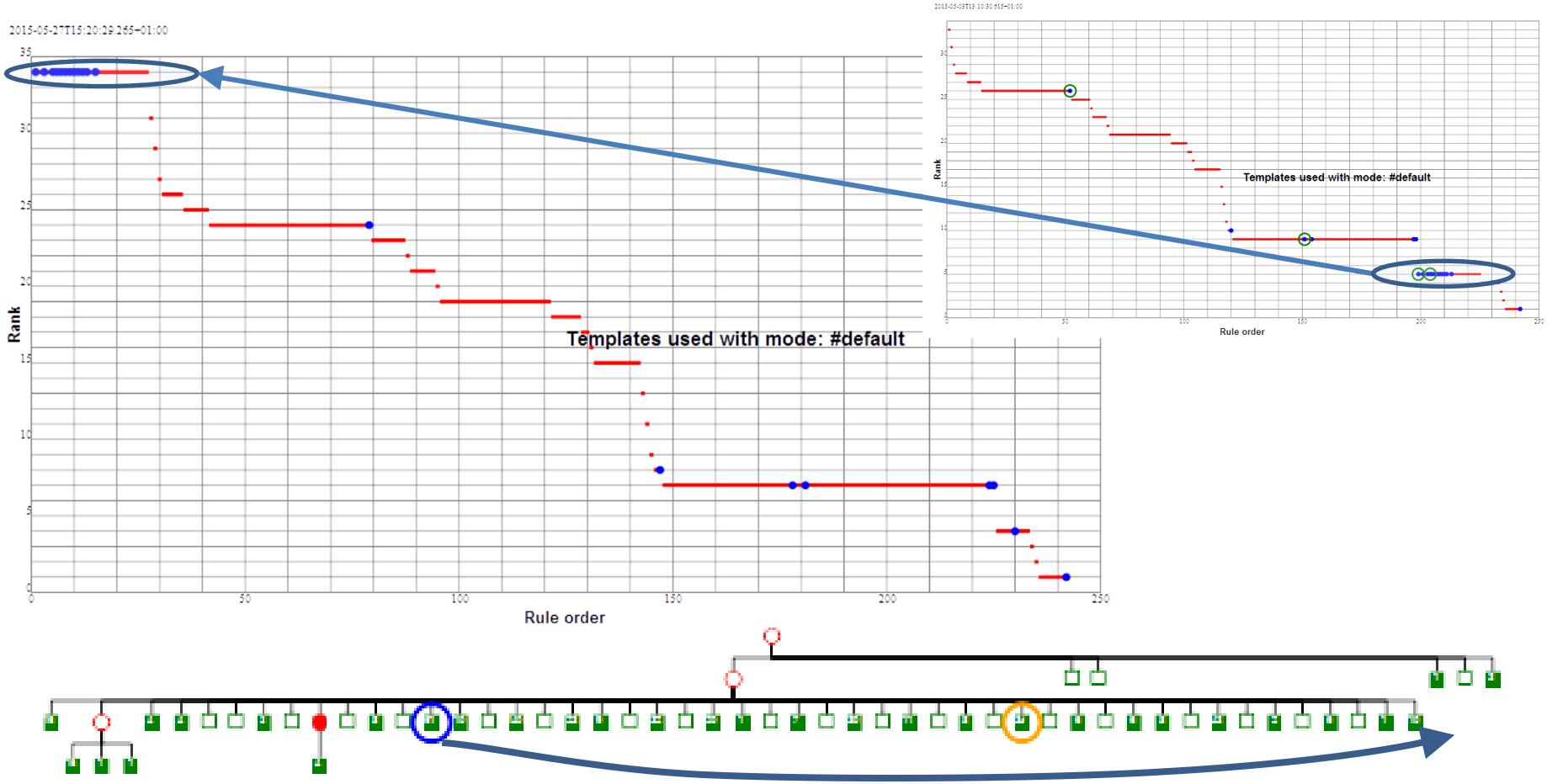
… no need to check these

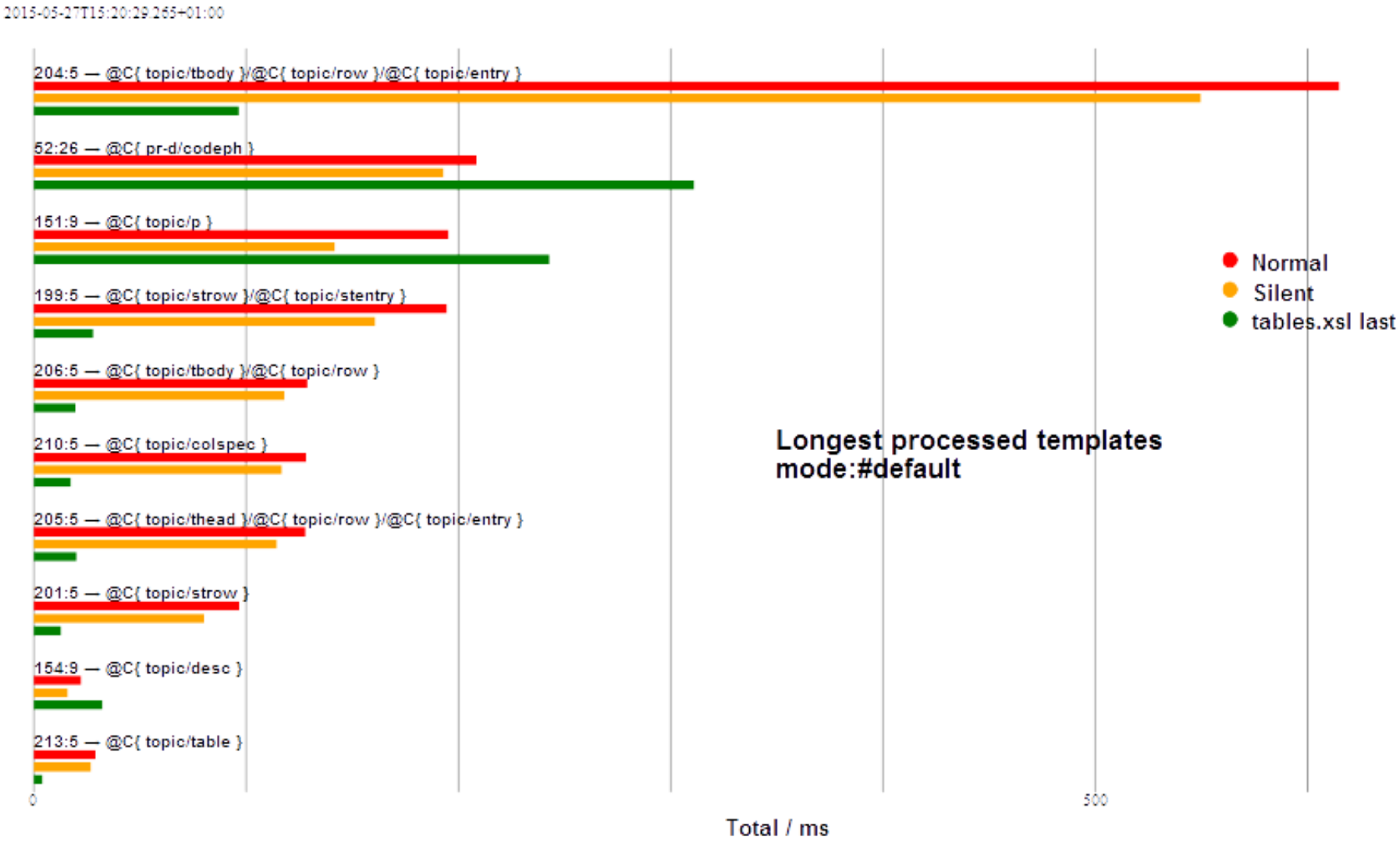

2015-05-27T15:20:29.265+01:00
204:5 — @C{ topic/tbody }/@C{ topic/row }/@C{ topic/entry }
52:26 — @C{ pr-d/codeph }
151:9 — @C{ topic/p }
199:5 — @C{ topic/strow }/@C{ topic/stentry }
206:5 — @C{ topic/tbody }/@C{ topic/row }
210:5 — @C{ topic/colspec }
205:5 — @C{ topic/thead }/@C{ topic/row }/@C{ topic/entry }
201:5 — @C{ topic/strow }
213:5 — @C{ topic/table }
207:5 — @C{ topic/thead }/@C{ topic/row }
Normal
Silent
Longest processed templates
mode:#default
0
500
Total / ms

'Mutually exclusive': promoting stylesheets



'Mutually exclusive': promoting stylesheets




2015-05-27T15:20:29.265+01:00
204:5 — @C{ topic/tbody }/@C{ topic/row }/@C{ topic/entry }
52:26 — @C{ pr-d/codeph }
151:9 — @C{ topic/p }
199:5 — @C{ topic/strow }/@C{ topic/stentry }
206:5 — @C{ topic/tbody }/@C{ topic/row }
210:5 — @C{ topic/colspec }
205:5 — @C{ topic/thead }/@C{ topic/row }/@C{ topic/entry }
201:5 — @C{ topic/strow }
154:9 — @C{ topic/desc }
213:5 — @C{ topic/table }
Normal
Silent
tables.xsl last
Longest processed templates
mode:#default
0
500
Total / ms

Pre-tokenizing `@class` data

```
R1: *[contains(@class,' topic/entry ')]
R2: *[contains(@class,' topic/row ')]
R3: *[contains(@class,' topic/row ')]/
    *[contains(@class,' topic/entry ')]

R1: *[tokenize(@class,'\s+')='topic/entry']
R2: *[tokenize(@class,'\s+')='topic/row']
R3: *[tokenize(@class,'\s+')='topic/row']/
    *[tokenize(@class,'\s+')='topic/entry
```

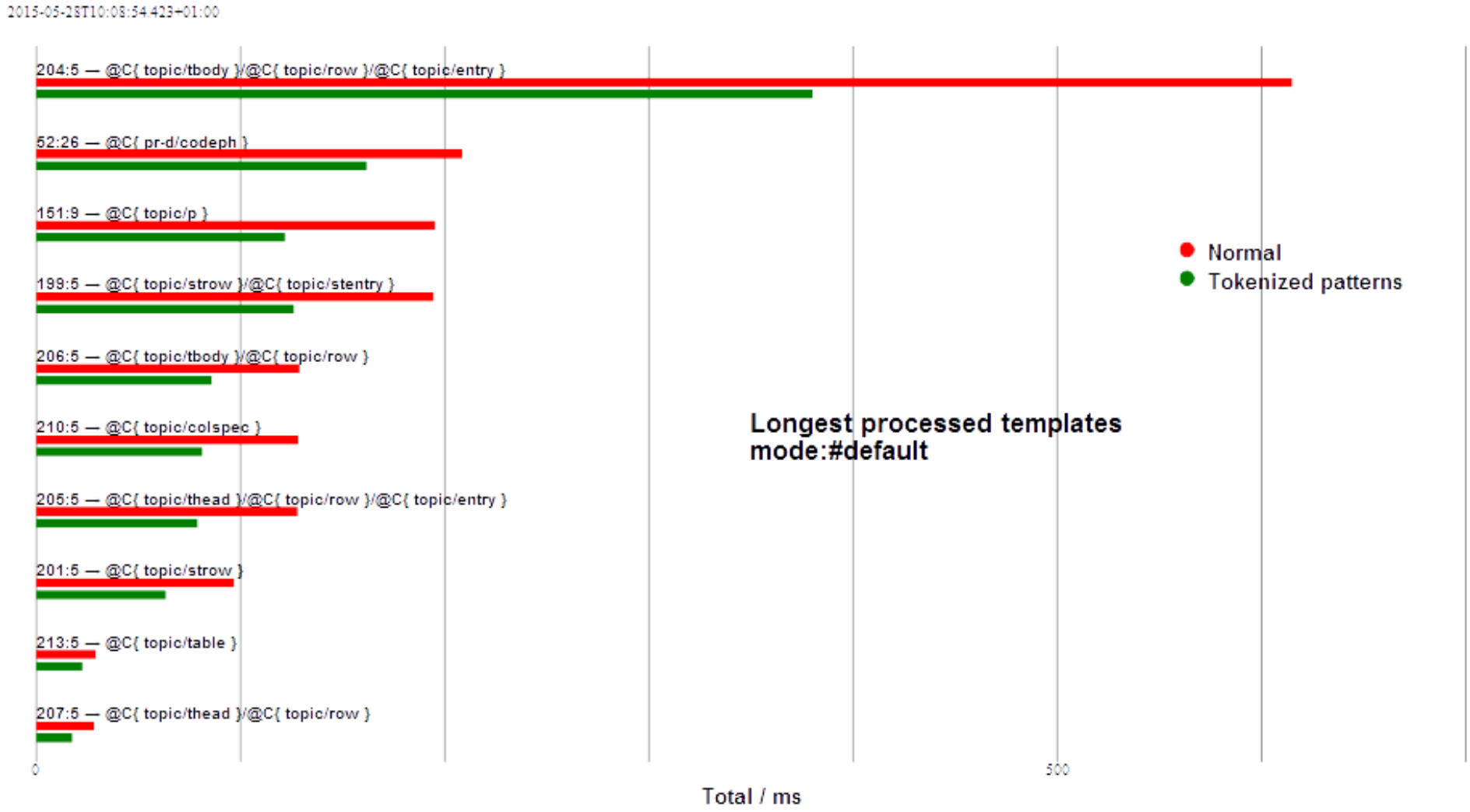
```
$tokens.self.class := tokenize(self::*/@class,'\s+')
$tokens.parent.class := tokenize(parent::*/@class,'\s+')

$precondition_M := $tokens.self.class = 'topic/entry'
$precondition_N := $tokens.self.class = 'topic/row'
$precondition_P := $tokens.parent.class = 'topic/row'

R1: $precondition_M && *
R2: $precondition_N && *
R3: $precondition_P && $precondition_M && *
```

XPath 3.1

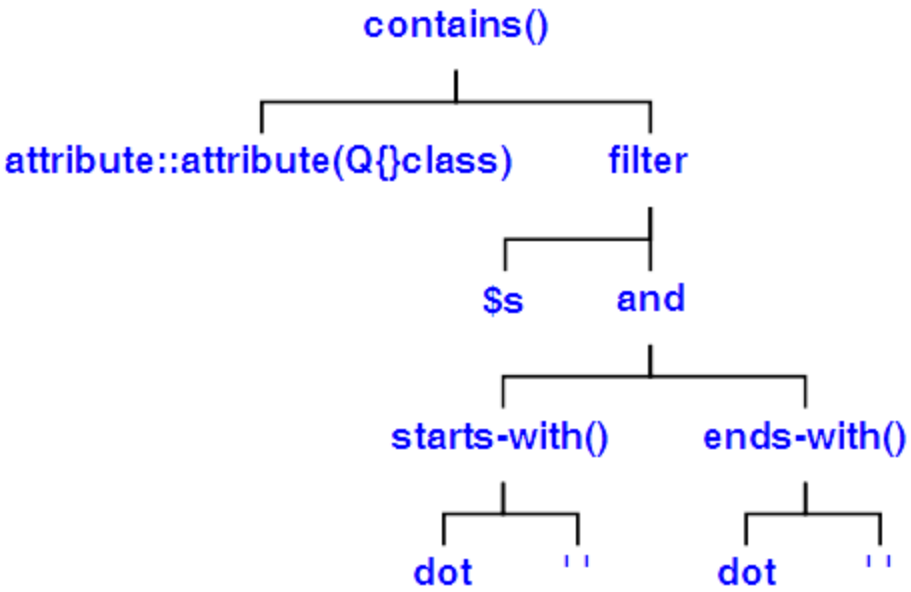
```
*[contains-token(@class,'topic/entry')]
```

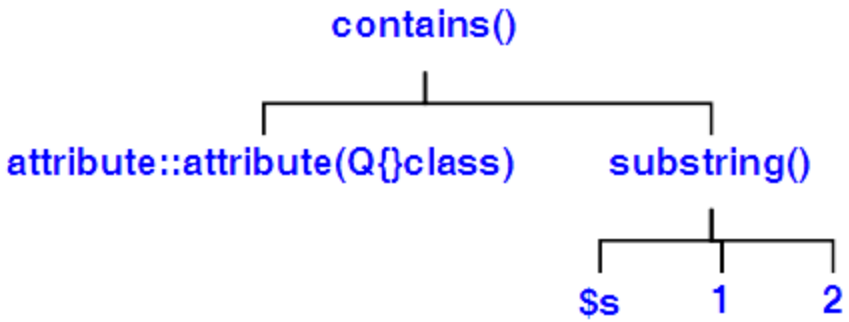

2015-05-28T10:08:54.423+01:00
204:5 — @C{ topic/tbody }/@C{ topic/row }/@C{ topic/entry }
52:26 — @C{ pr-d/codeph }
151:9 — @C{ topic/p }
199:5 — @C{ topic/strow }/@C{ topic/stentry }
206:5 — @C{ topic/tbody }/@C{ topic/row }
210:5 — @C{ topic/colspec }
205:5 — @C{ topic/thead }/@C{ topic/row }/@C{ topic/entry }
201:5 — @C{ topic/strow }
213:5 — @C{ topic/table }
207:5 — @C{ topic/thead }/@C{ topic/row }
Normal
Tokenized patterns
Longest processed templates
mode:#default
0
500
Total / ms

Configuring the tuning

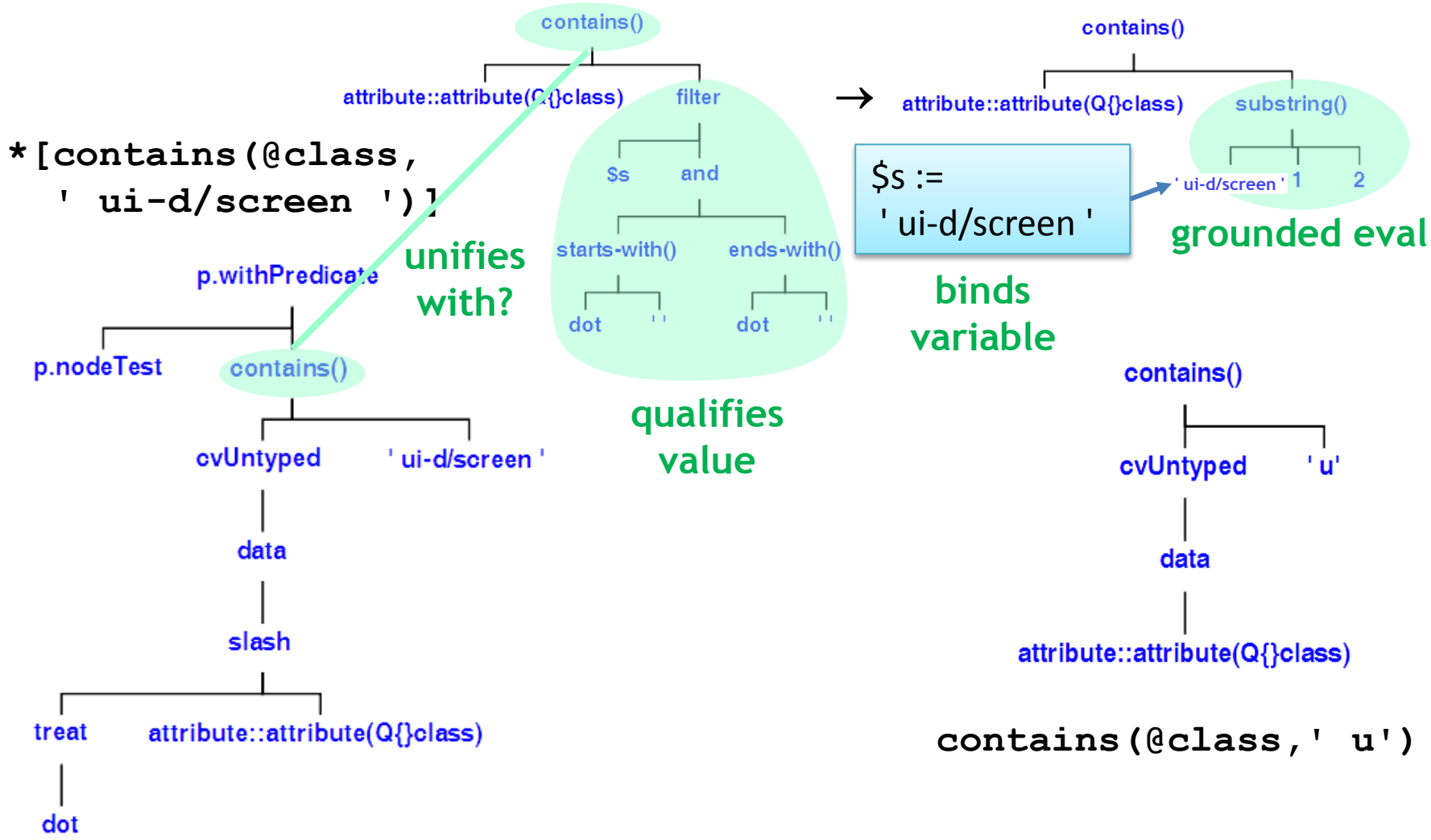
Define preconditions via patterns (*cf.* Snelson):

```
contains(@class,                    →   contains(@class,
   $s[starts-with(.,' ')                    substring($s,1,2))
    and ends-with(.,' ')]
```





Unifying for preconditions



Conclusions

- Large sets of `*[predicate]` XSLT patterns can be *very* expensive
  (DITA is paying *a lot* for `@class` extensibility)
- Preconditions are practical: *but which ones?*
- Other oracle measures can help
  - *'This document is mostly tables'*
- 'Tuning' can be configured via patterns
  - Watch SAXONICA XSLT AND XQUERY PROCESSING