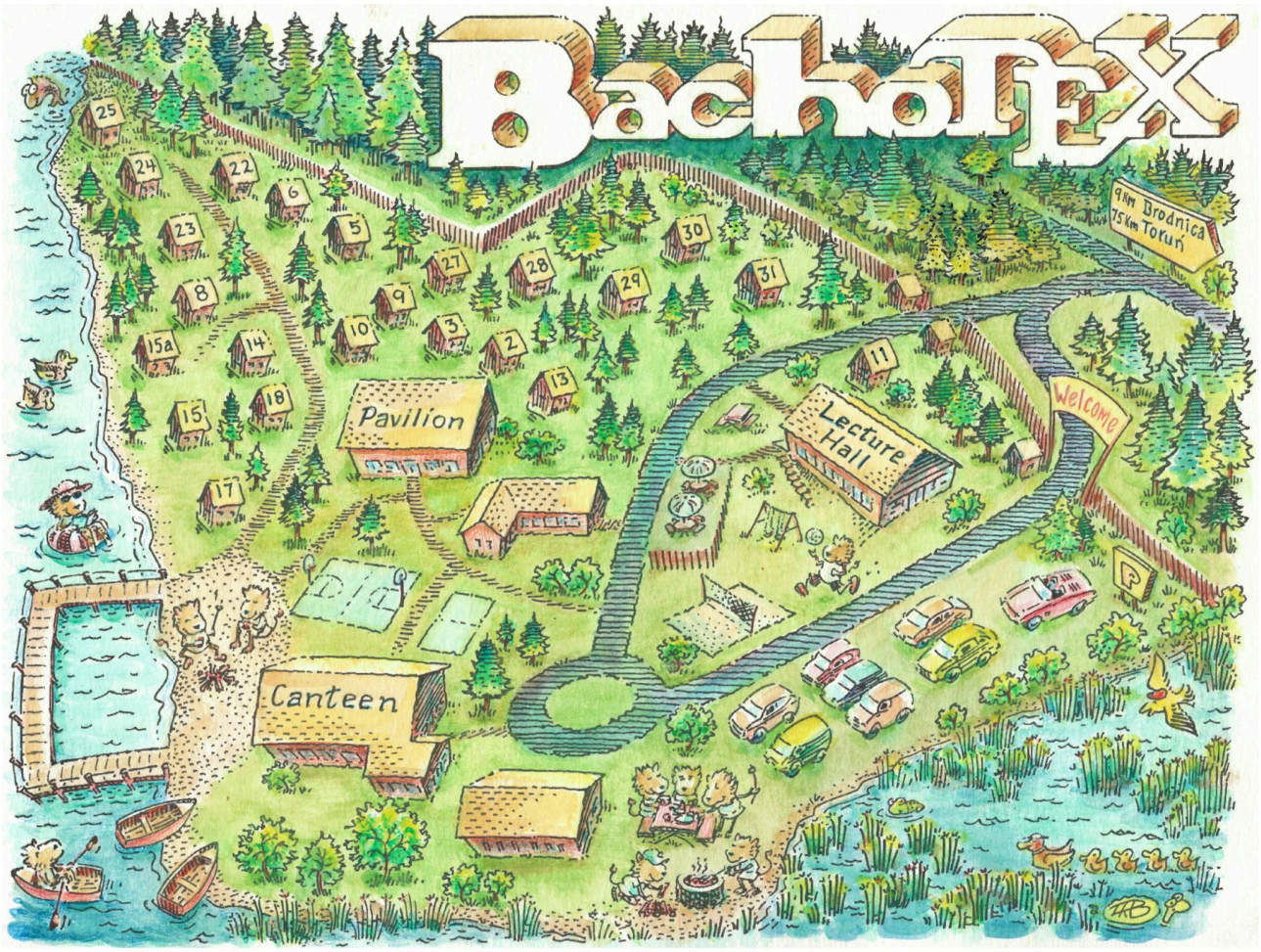
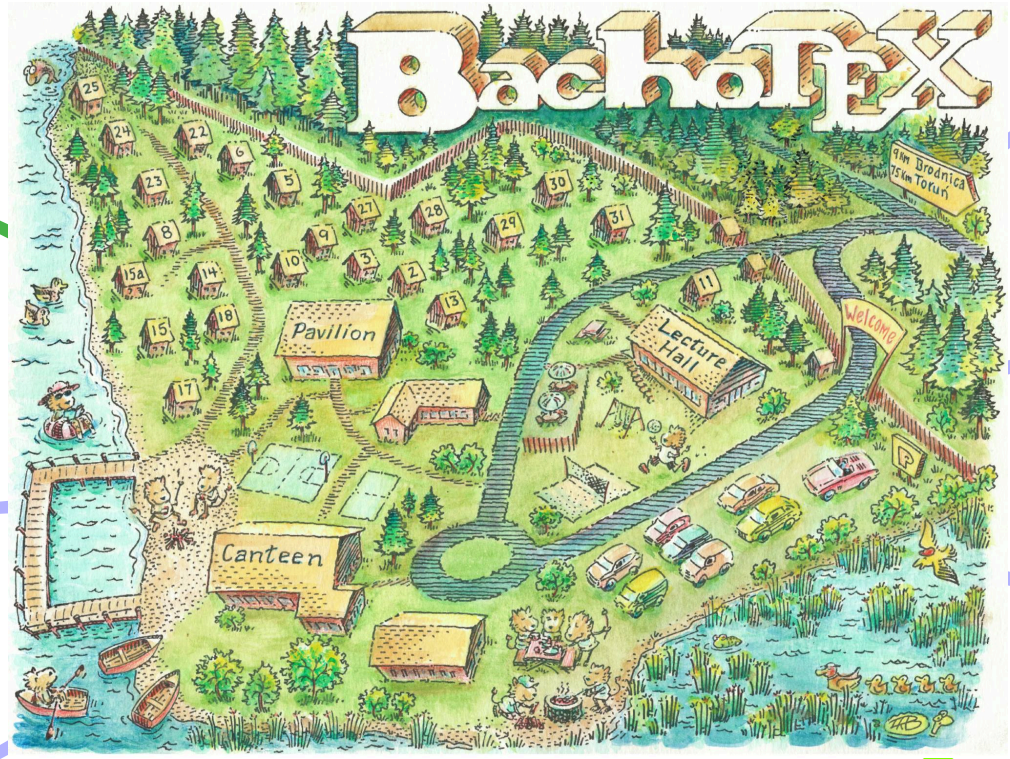


Alice in Bachotek

Frank Mittelbach
May 2016



Alice in Bachotek

Frank Mittelbach
May 2016

Introduction

The pagination problem

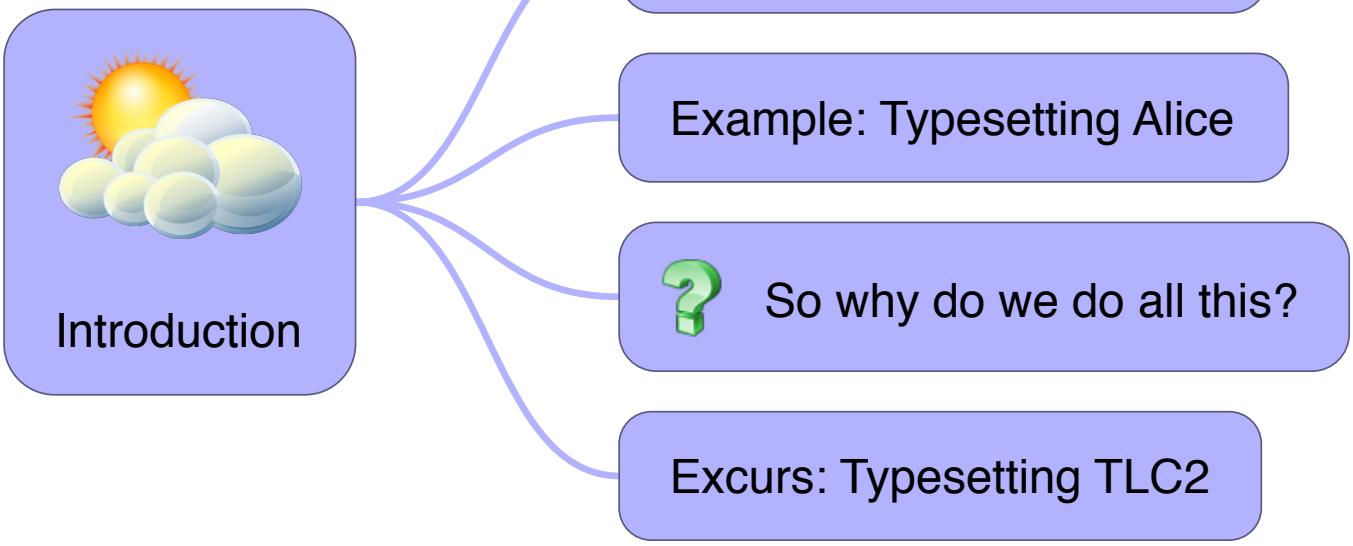
The pagination framework

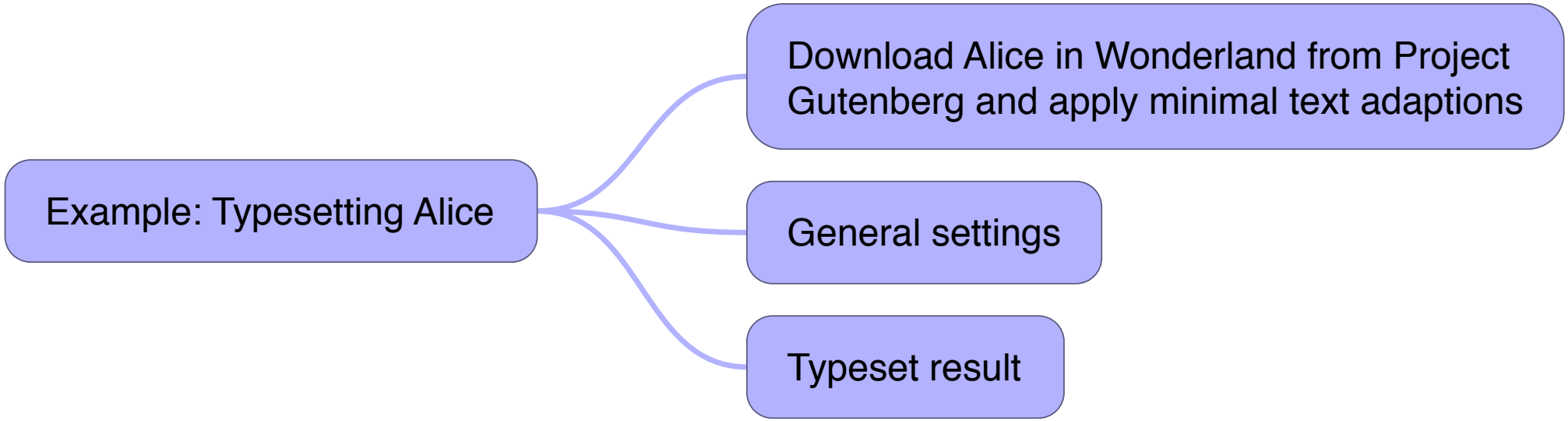
First results

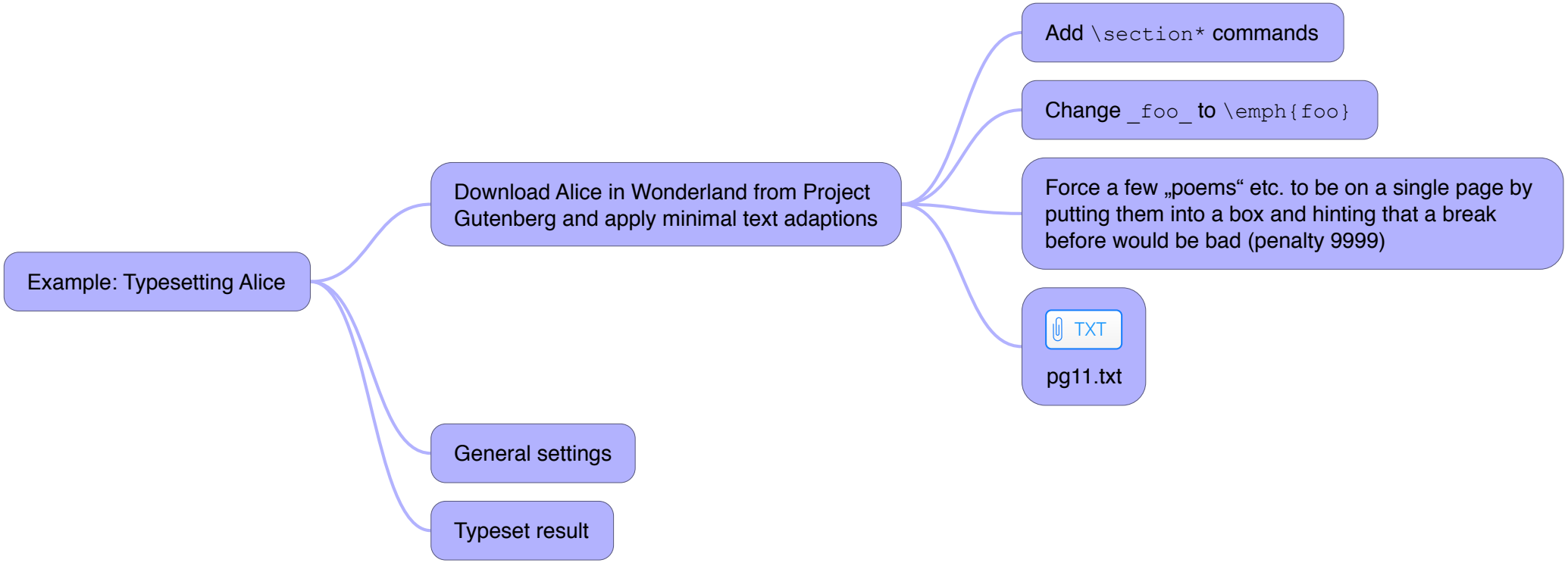
Conclusions

Roadwork

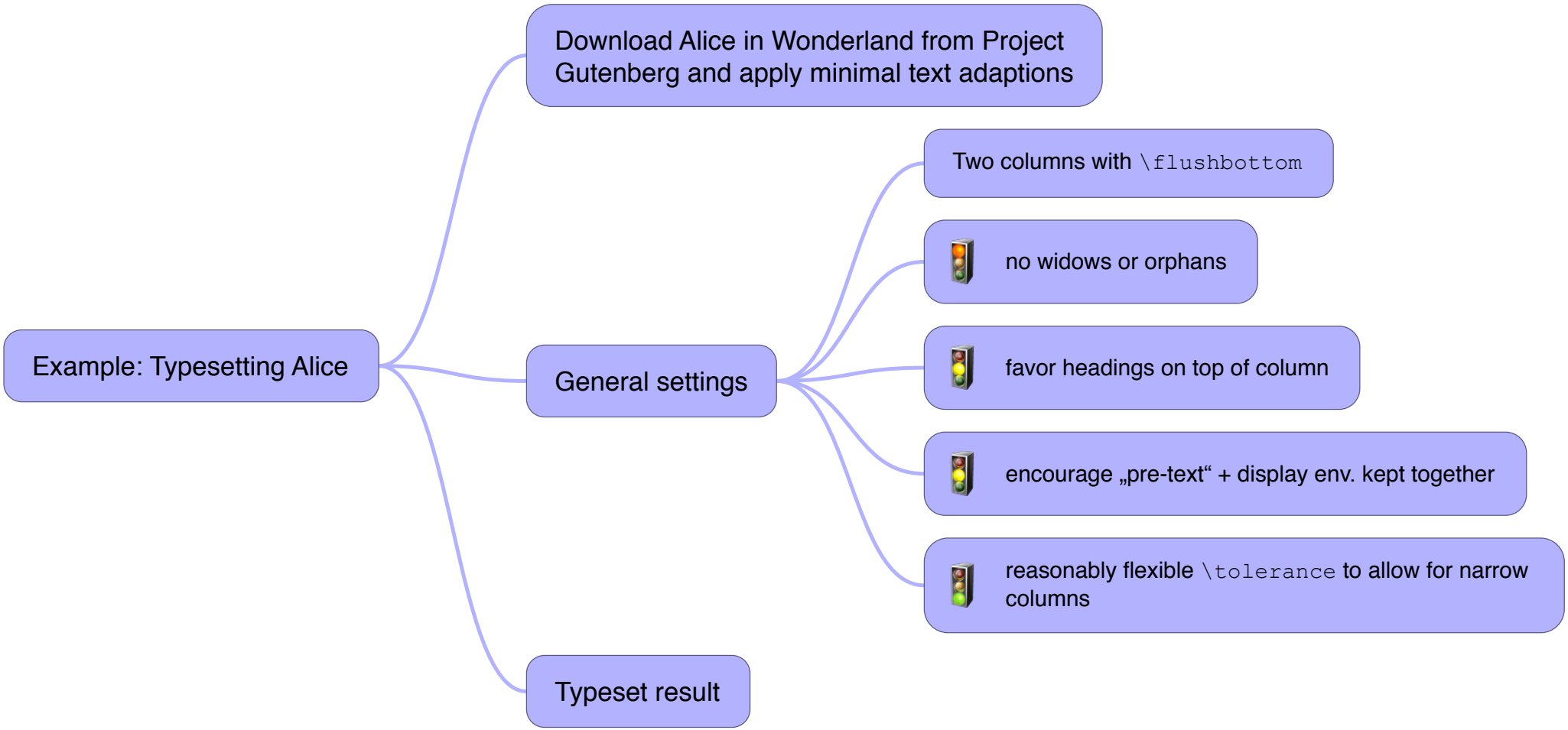
The tale of the tail of bugs


Introduction
What's this all about?
Example: Typesetting Alice
So why do we do all this?
Excurs: Typesetting TLC2

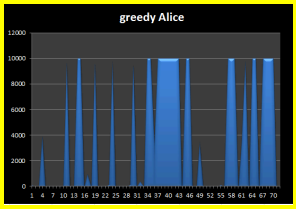

Example: Typesetting Alice
Download Alice in Wonderland from Project Gutenberg and apply minimal text adaptions
General settings
Typeset result


Example: Typesetting Alice
Download Alice in Wonderland from Project Gutenberg and apply minimal text adaptions
Add \section* commands
Change _foo_ to \emph{foo}
Force a few „poems“ etc. to be on a single page by putting them into a box and hinting that a break before would be bad (penalty 9999)
TXT
pg11.txt
General settings
Typeset result


TXT
pg11.txt


Example: Typesetting Alice
Download Alice in Wonderland from Project Gutenberg and apply minimal text adaptions
General settings
Two columns with \flushbottom
no widows or orphans
favor headings on top of column
encourage „pre-text“ + display env. kept together
reasonably flexible \tolerance to allow for narrow columns
Typeset result

Typeset result

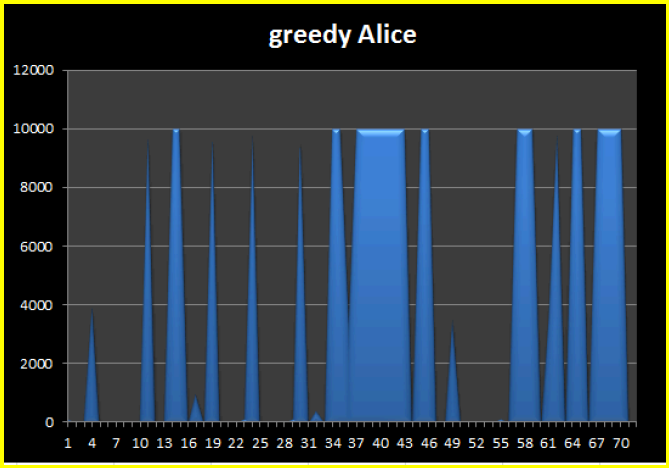


Medium desaster

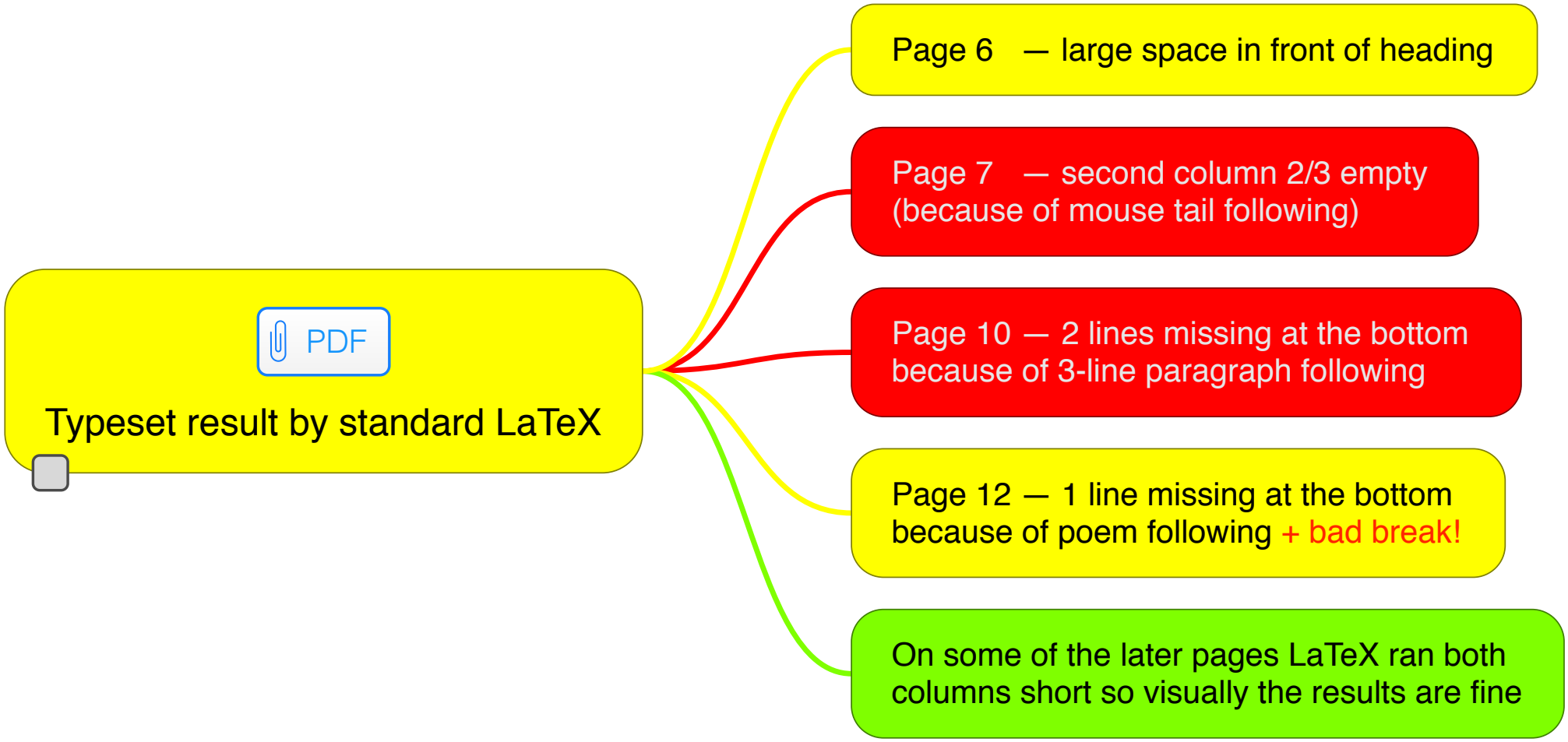
PDF

Typeset result by standard LaTeX

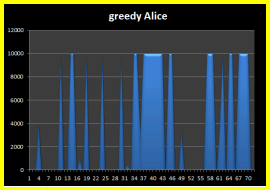
Consequences



Medium desaster


PDF
Typeset result by standard LaTeX
Page 6 — large space in front of heading
Page 7 — second column 2/3 empty
(because of mouse tail following)
Page 10 — 2 lines missing at the bottom
because of 3-line paragraph following
Page 12 — 1 line missing at the bottom
because of poem following + bad break!
On some of the later pages LaTeX ran both
columns short so visually the results are fine

Typeset result



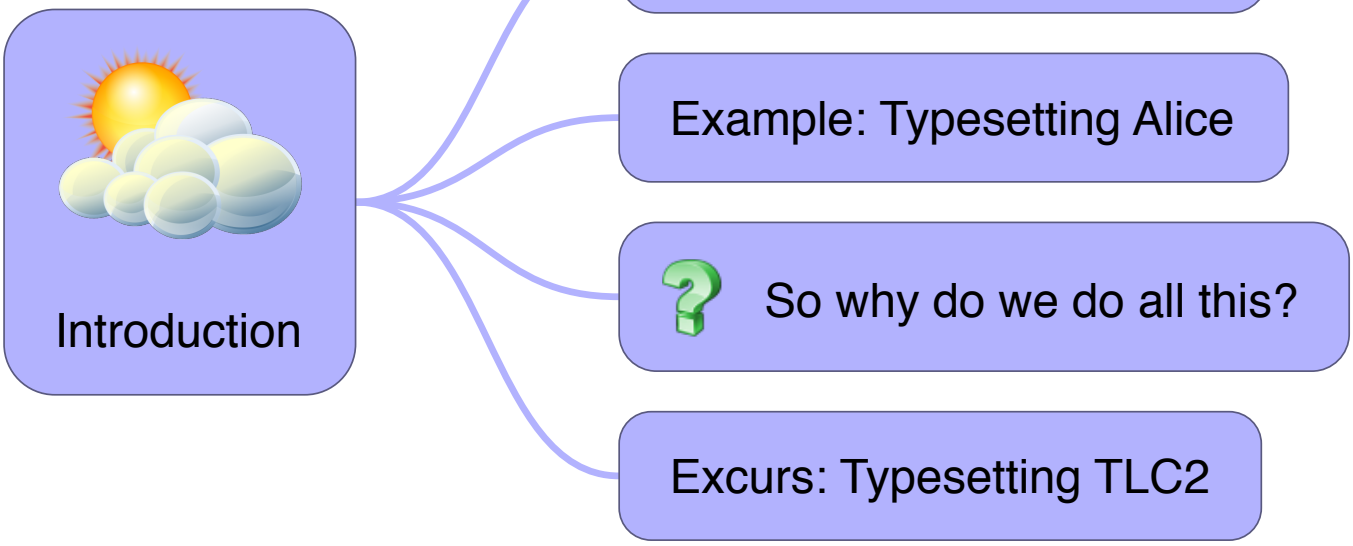
Medium desaster

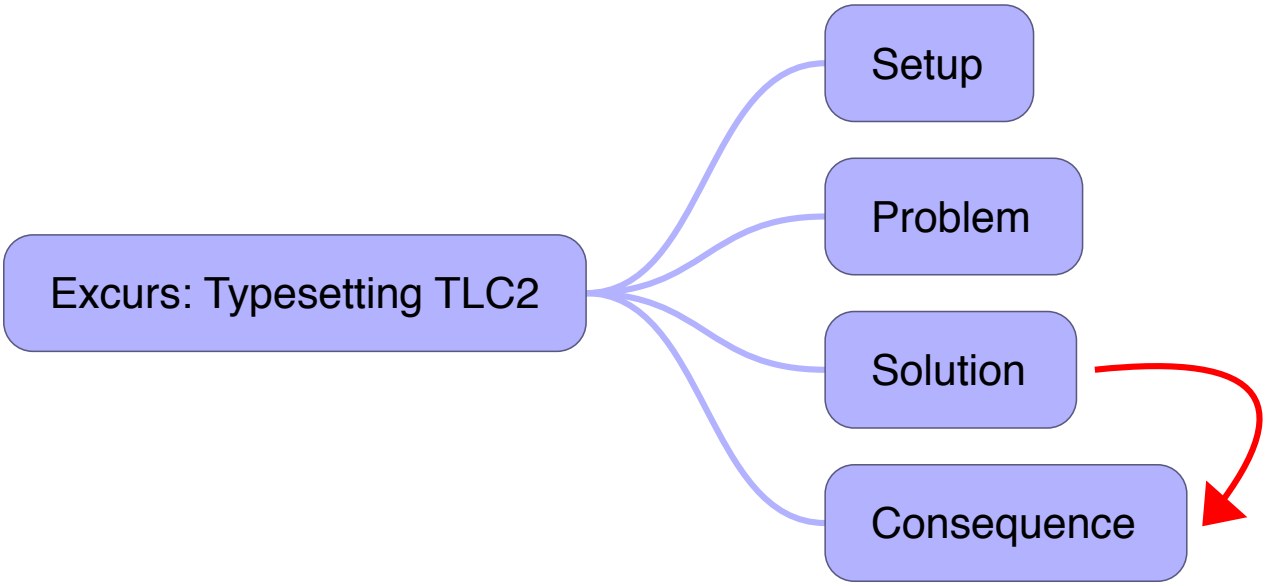
PDF

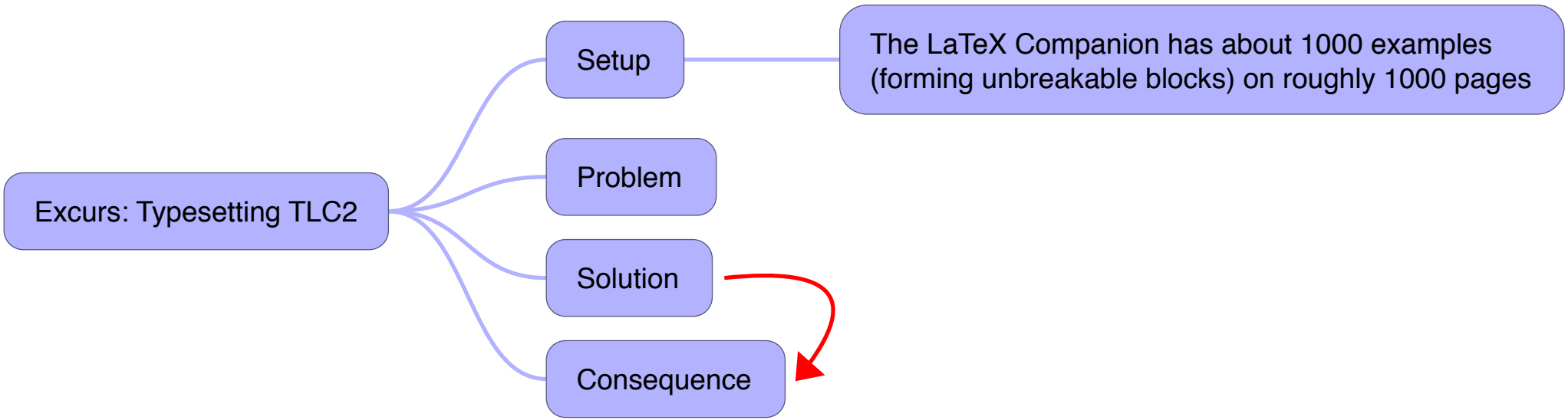
Typeset result by standard LaTeX

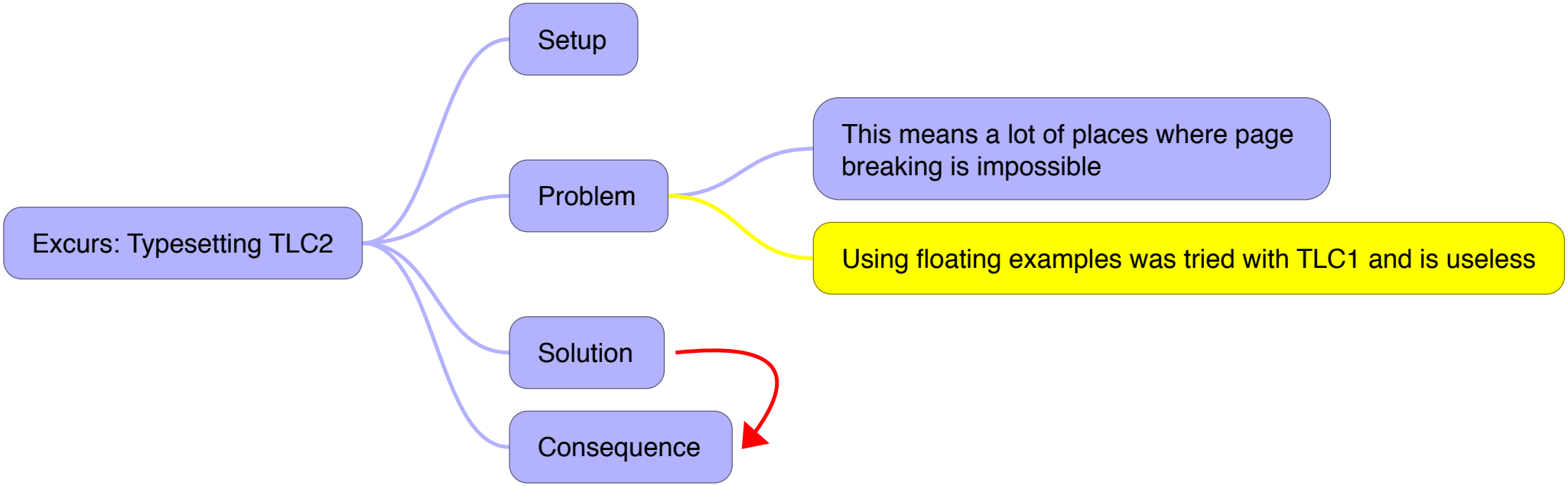
Consequences

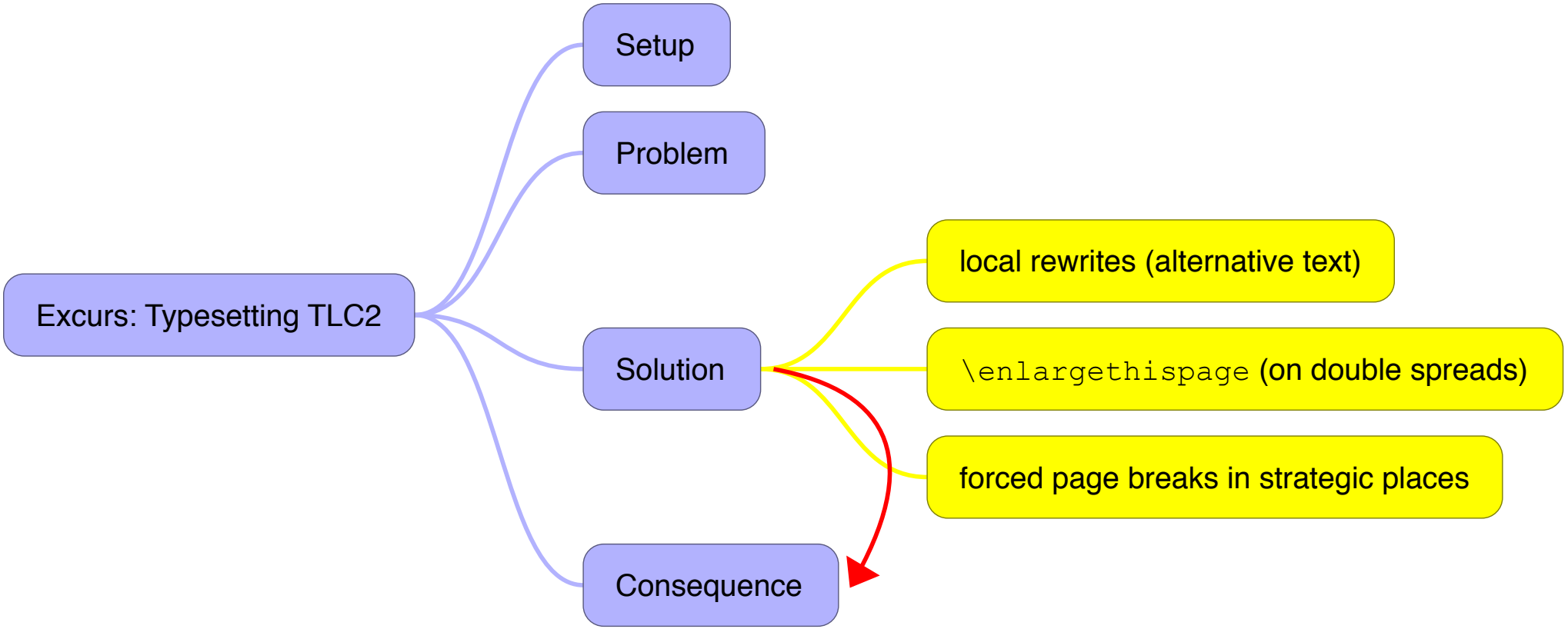
- A lot of manual labor to fix it
- Iterative process, thus time-consuming
- Source get cluttered with formatting instructions — not suitable for other formattings

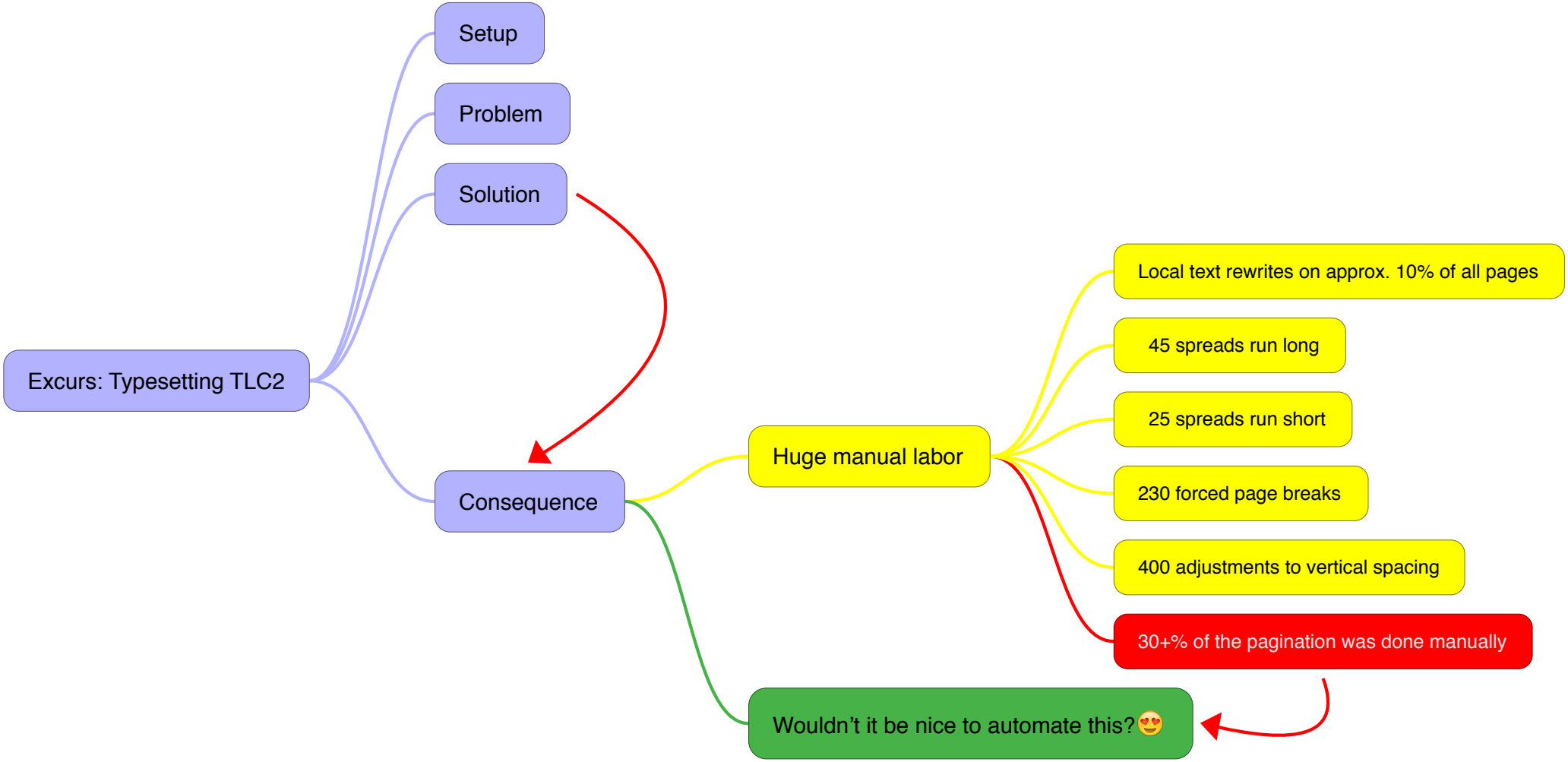

Introduction
What's this all about?
Example: Typesetting Alice
So why do we do all this?
Excurs: Typesetting TLC2


Excurs: Typesetting TLC2
Setup
Problem
Solution
Consequence

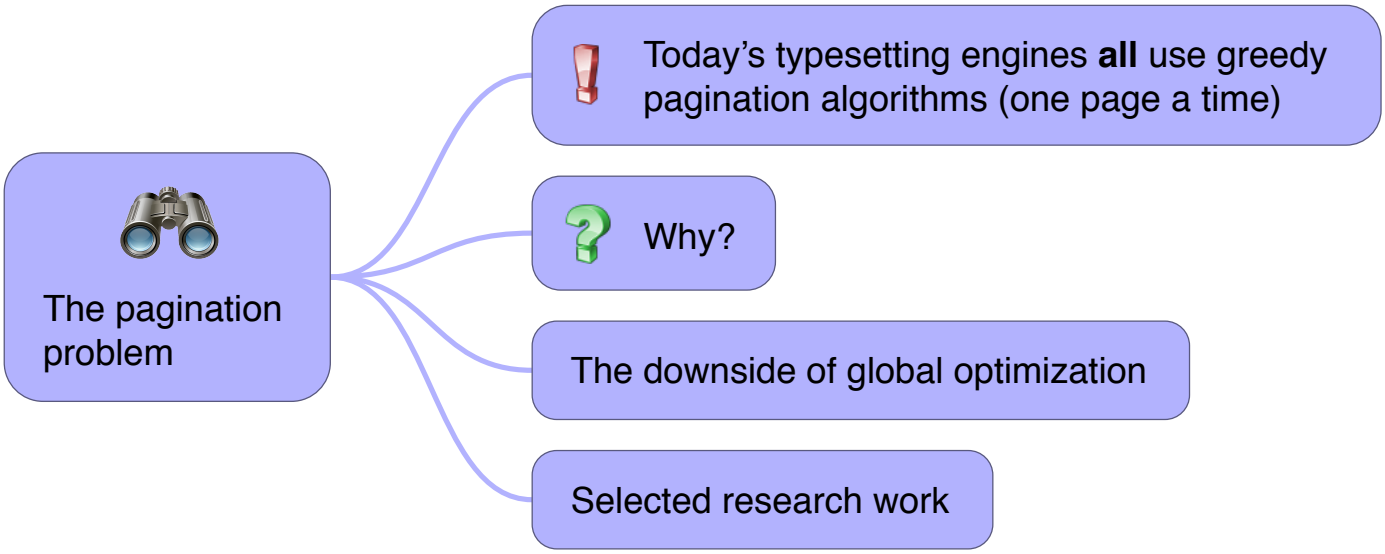

Excurs: Typesetting TLC2
Setup
Problem
Solution
Consequence
The LaTeX Companion has about 1000 examples
(forming unbreakable blocks) on roughly 1000 pages

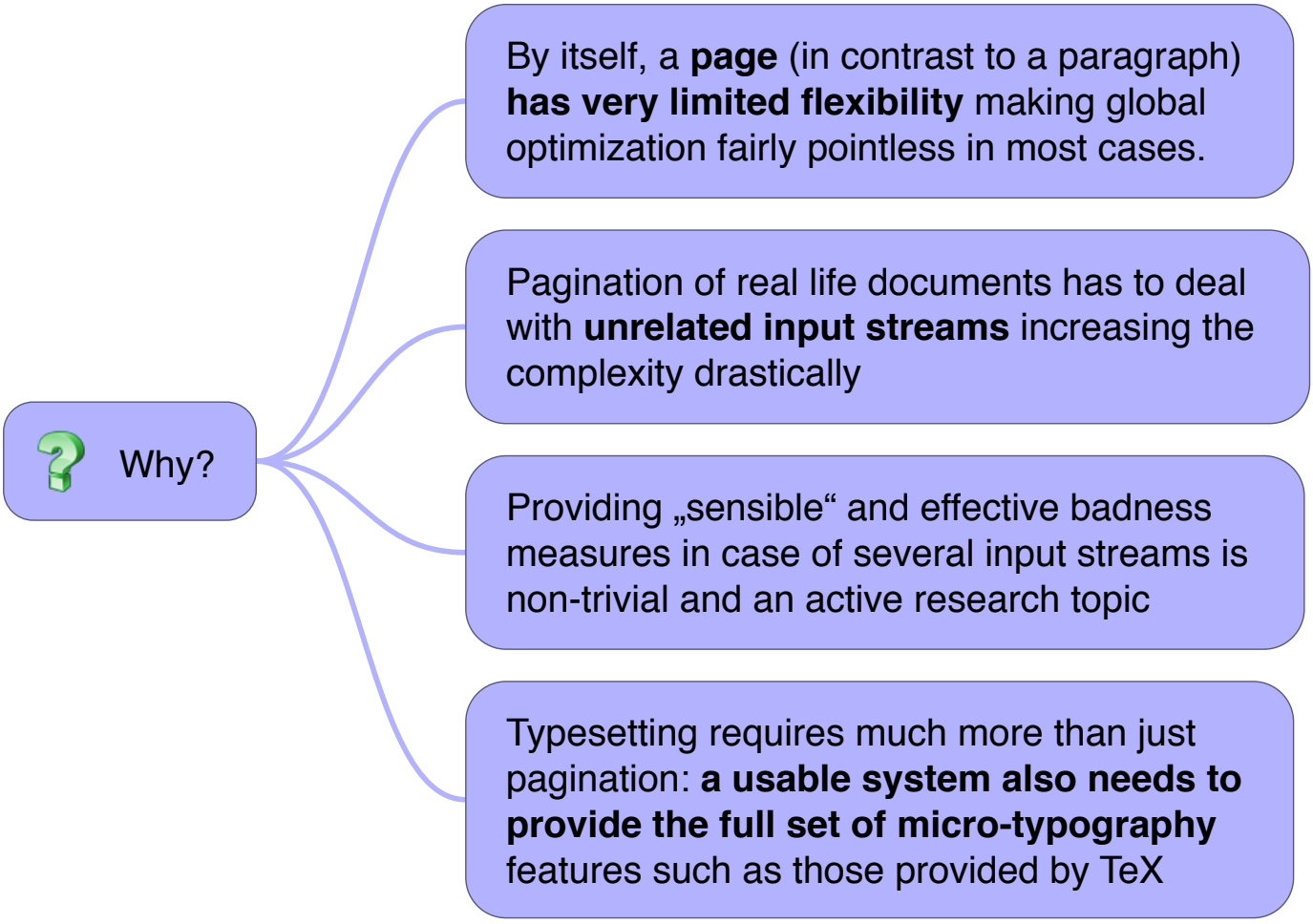

Excurs: Typesetting TLC2
Setup
Problem
This means a lot of places where page breaking is impossible
Using floating examples was tried with TLC1 and is useless
Solution
Consequence

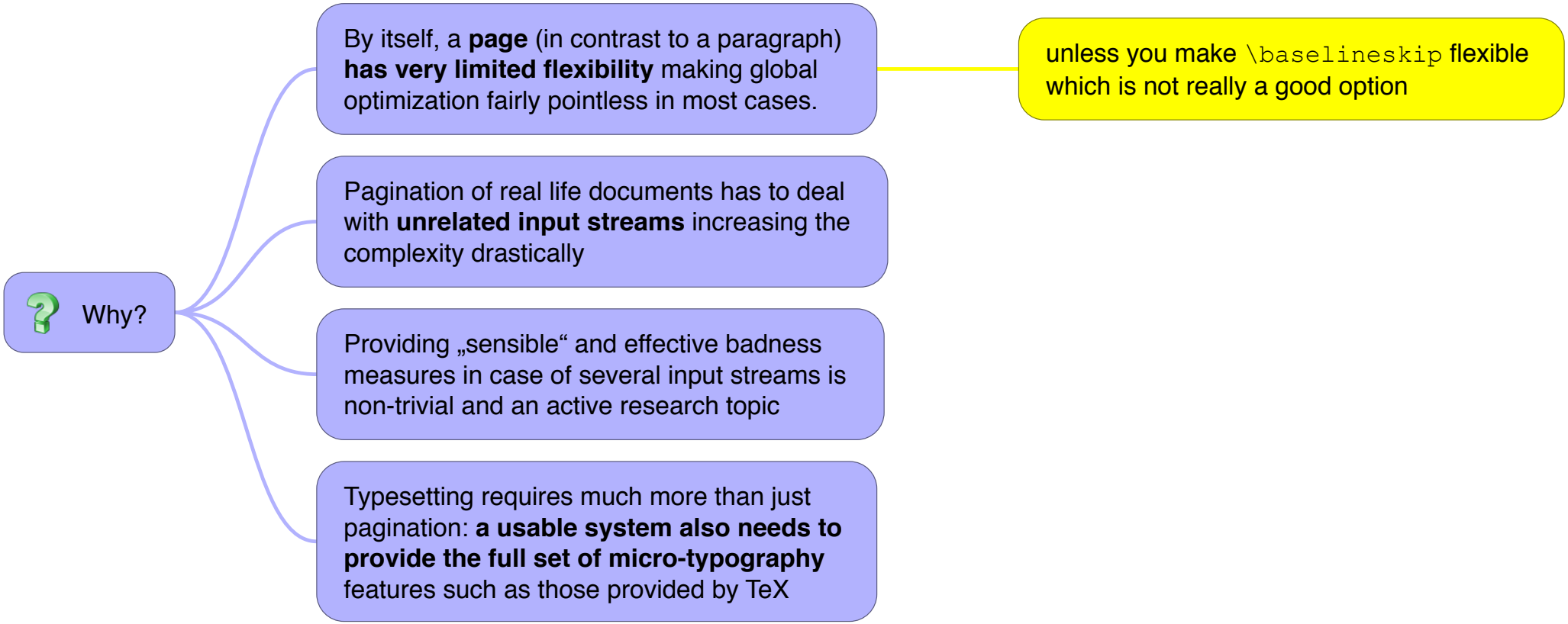

Excurs: Typesetting TLC2
Setup
Problem
Solution
Consequence
local rewrites (alternative text)
\enlargethispage (on double spreads)
forced page breaks in strategic places

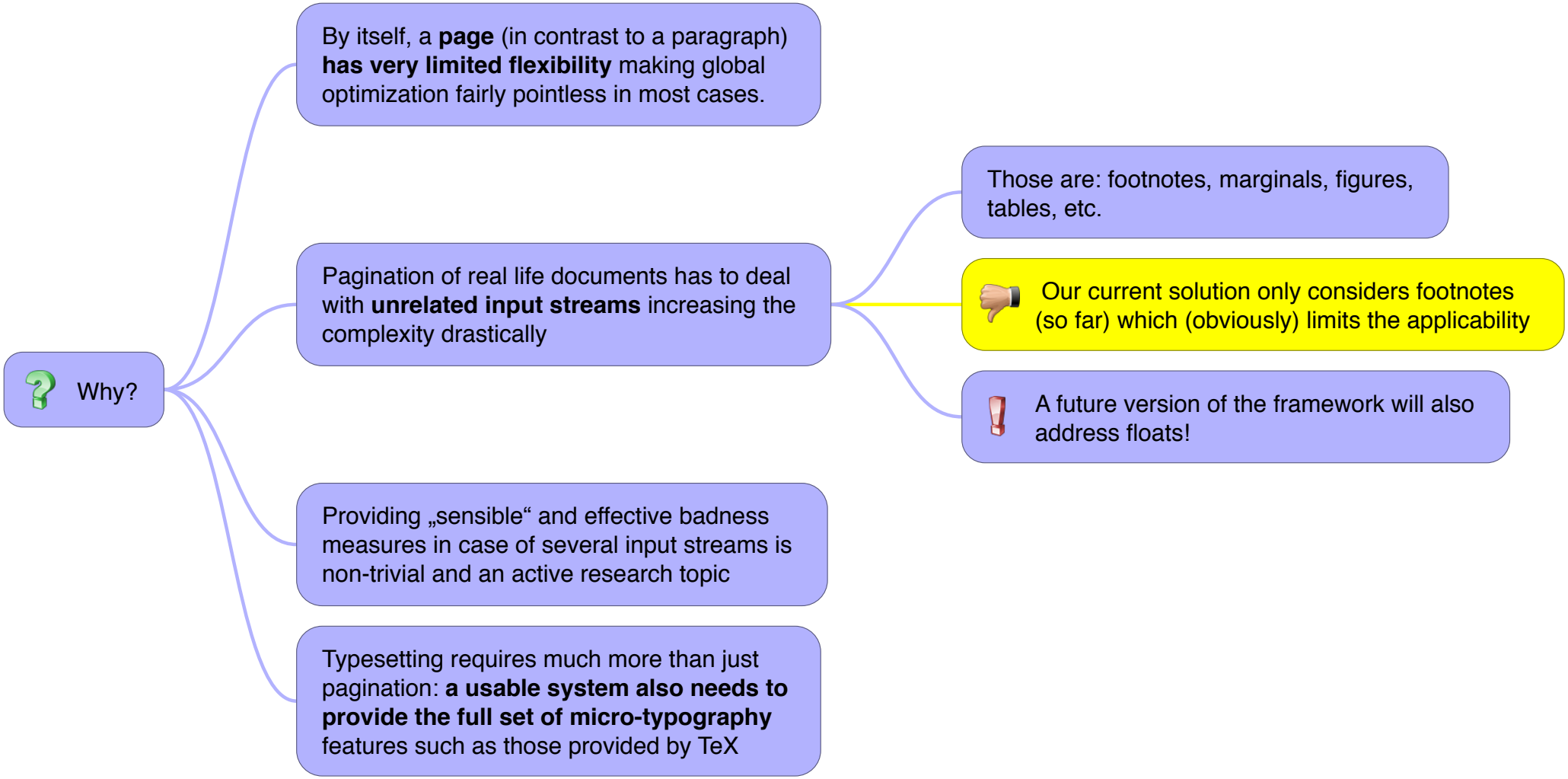

Excurs: Typesetting TLC2
Setup
Problem
Solution
Consequence
Huge manual labor
Local text rewrites on approx. 10% of all pages
45 spreads run long
25 spreads run short
230 forced page breaks
400 adjustments to vertical spacing
30+% of the pagination was done manually
Wouldn’t it be nice to automate this?😍

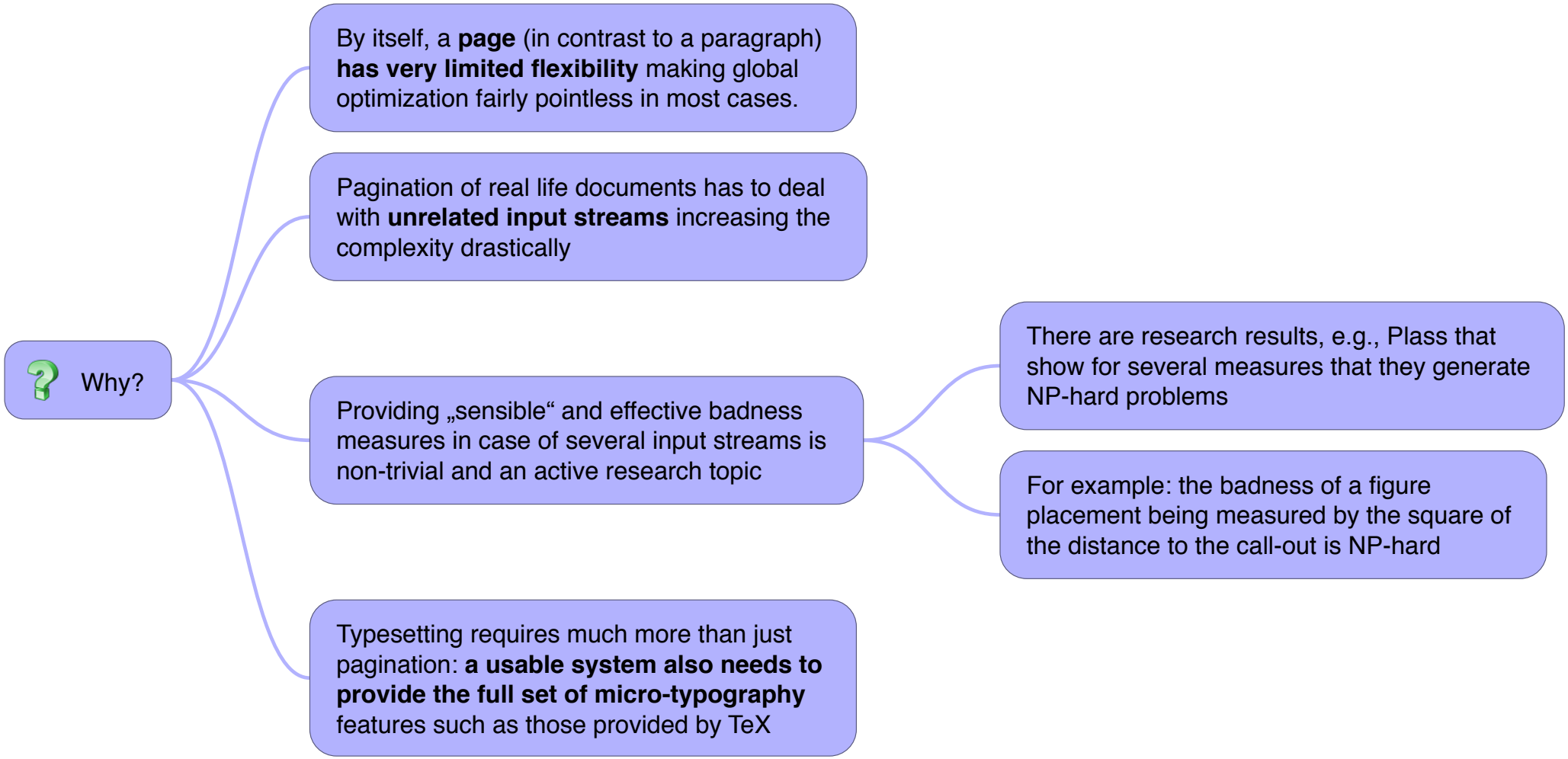
The pagination problem

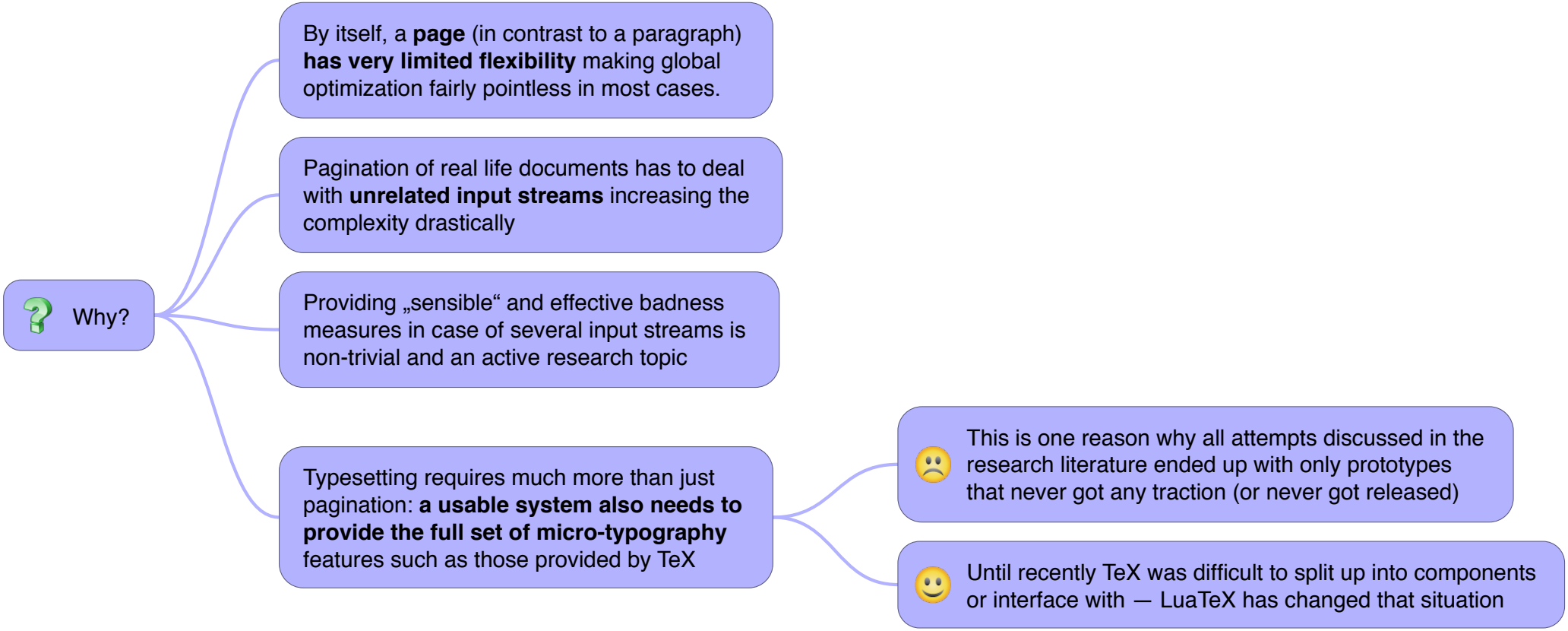

The pagination problem
Today's typesetting engines **all** use greedy pagination algorithms (one page a time)
Why?
The downside of global optimization
Selected research work

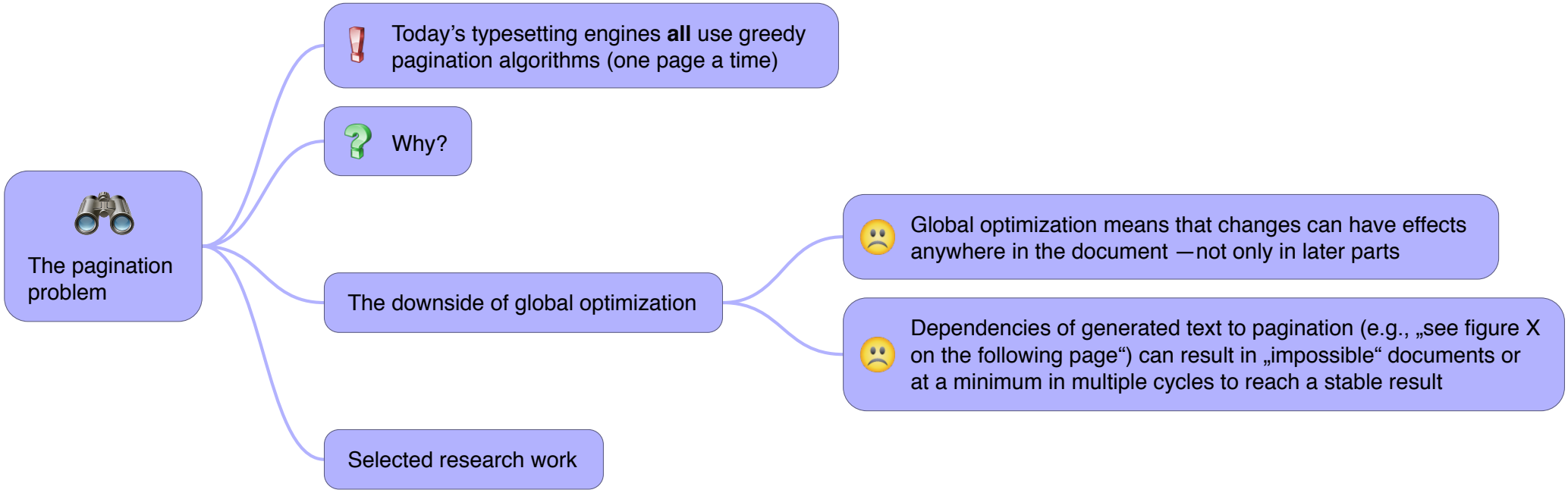

Why?
By itself, a **page** (in contrast to a paragraph) **has very limited flexibility** making global optimization fairly pointless in most cases.
Pagination of real life documents has to deal with **unrelated input streams** increasing the complexity drastically
Providing „sensible“ and effective badness measures in case of several input streams is non-trivial and an active research topic
Typesetting requires much more than just pagination: **a usable system also needs to provide the full set of micro-typography** features such as those provided by TeX

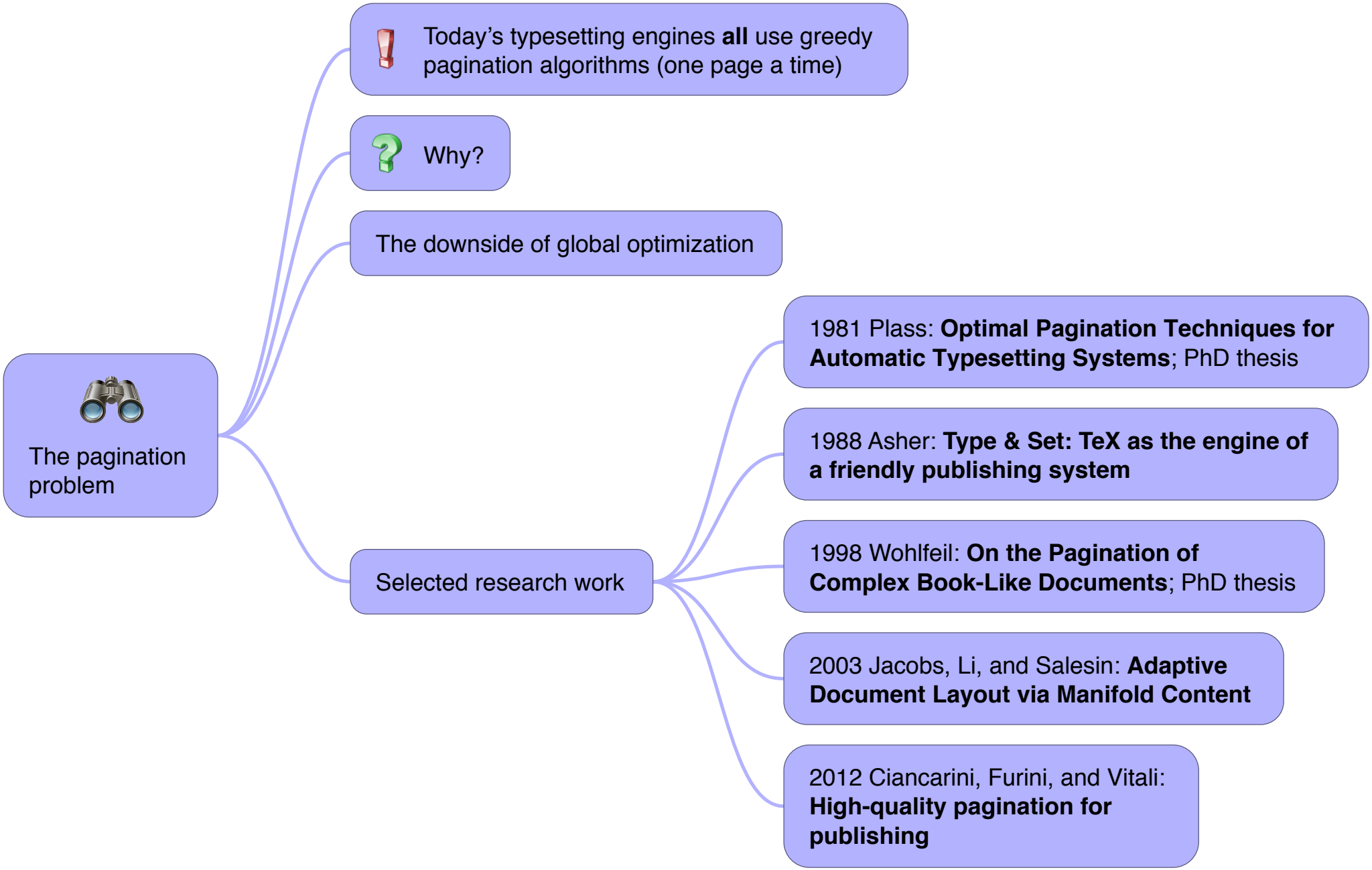

Why?
By itself, a **page** (in contrast to a paragraph) **has very limited flexibility** making global optimization fairly pointless in most cases.
unless you make `\baselineskip` flexible which is not really a good option
Pagination of real life documents has to deal with **unrelated input streams** increasing the complexity drastically
Providing „sensible“ and effective badness measures in case of several input streams is non-trivial and an active research topic
Typesetting requires much more than just pagination: **a usable system also needs to provide the full set of micro-typography** features such as those provided by TeX


Why?
By itself, a **page** (in contrast to a paragraph) **has very limited flexibility** making global optimization fairly pointless in most cases.
Pagination of real life documents has to deal with **unrelated input streams** increasing the complexity drastically
Those are: footnotes, marginals, figures, tables, etc.
Our current solution only considers footnotes (so far) which (obviously) limits the applicability
A future version of the framework will also address floats!
Providing „sensible“ and effective badness measures in case of several input streams is non-trivial and an active research topic
Typesetting requires much more than just pagination: **a usable system also needs to provide the full set of micro-typography** features such as those provided by TeX


Why?
By itself, a **page** (in contrast to a paragraph) **has very limited flexibility** making global optimization fairly pointless in most cases.
Pagination of real life documents has to deal with **unrelated input streams** increasing the complexity drastically
Providing „sensible“ and effective badness measures in case of several input streams is non-trivial and an active research topic
There are research results, e.g., Plass that show for several measures that they generate NP-hard problems
For example: the badness of a figure placement being measured by the square of the distance to the call-out is NP-hard
Typesetting requires much more than just pagination: **a usable system also needs to provide the full set of micro-typography** features such as those provided by TeX

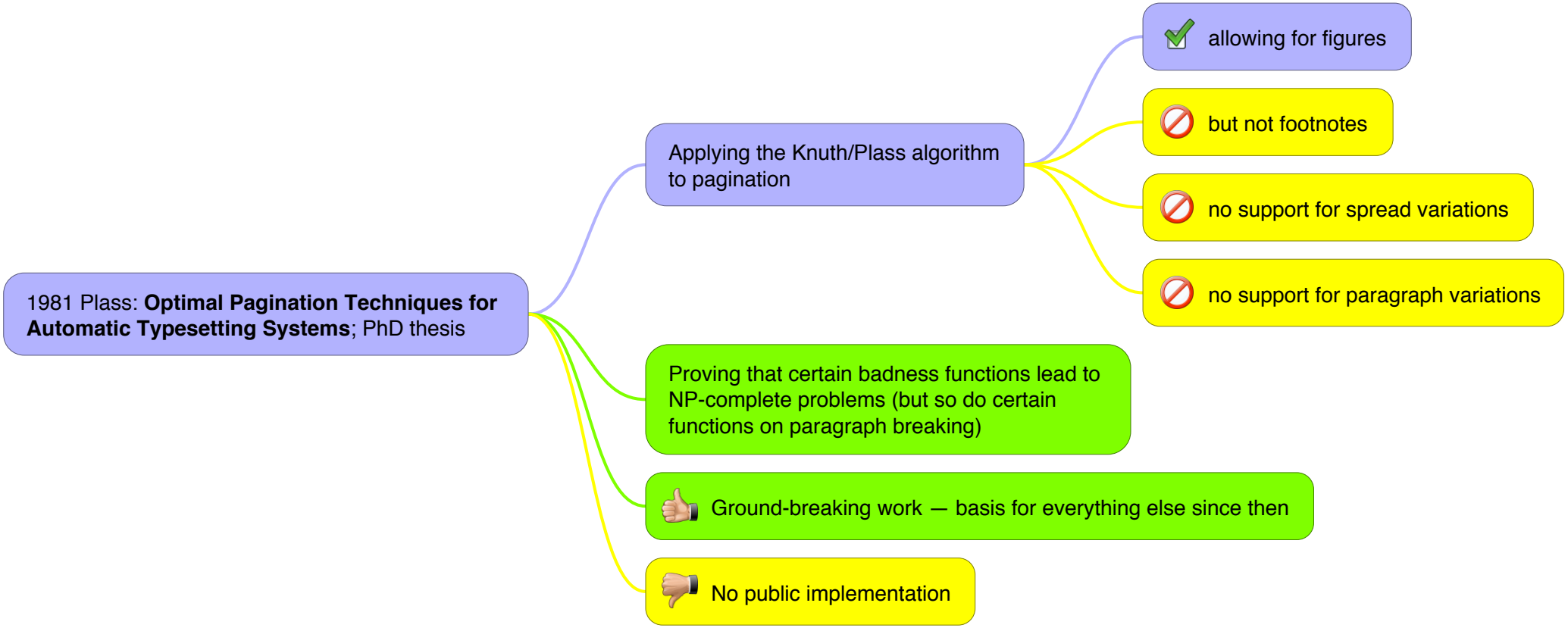

Why?
By itself, a **page** (in contrast to a paragraph) **has very limited flexibility** making global optimization fairly pointless in most cases.
Pagination of real life documents has to deal with **unrelated input streams** increasing the complexity drastically
Providing „sensible“ and effective badness measures in case of several input streams is non-trivial and an active research topic
Typesetting requires much more than just pagination: **a usable system also needs to provide the full set of micro-typography** features such as those provided by TeX
This is one reason why all attempts discussed in the research literature ended up with only prototypes that never got any traction (or never got released)
Until recently TeX was difficult to split up into components or interface with — LuaTeX has changed that situation


The pagination problem
Today's typesetting engines **all** use greedy pagination algorithms (one page a time)
Why?
The downside of global optimization
Global optimization means that changes can have effects anywhere in the document —not only in later parts
Dependencies of generated text to pagination (e.g., „see figure X on the following page“) can result in „impossible“ documents or at a minimum in multiple cycles to reach a stable result
Selected research work


The pagination problem
Today's typesetting engines **all** use greedy pagination algorithms (one page a time)
Why?
The downside of global optimization
Selected research work
1981 Plass: **Optimal Pagination Techniques for Automatic Typesetting Systems**; PhD thesis
1988 Asher: **Type & Set: TeX as the engine of a friendly publishing system**
1998 Wohlfeil: **On the Pagination of Complex Book-Like Documents**; PhD thesis
2003 Jacobs, Li, and Salesin: **Adaptive Document Layout via Manifold Content**
2012 Ciancarini, Furini, and Vitali: **High-quality pagination for publishing**

1981 Plass: **Optimal Pagination Techniques for Automatic Typesetting Systems**; PhD thesis

- Applying the Knuth/Plass algorithm to pagination
- Proving that certain badness functions lead to NP-complete problems (but so do certain functions on paragraph breaking)
- 👍 Ground-breaking work — basis for everything else since then
- 👎 No public implementation

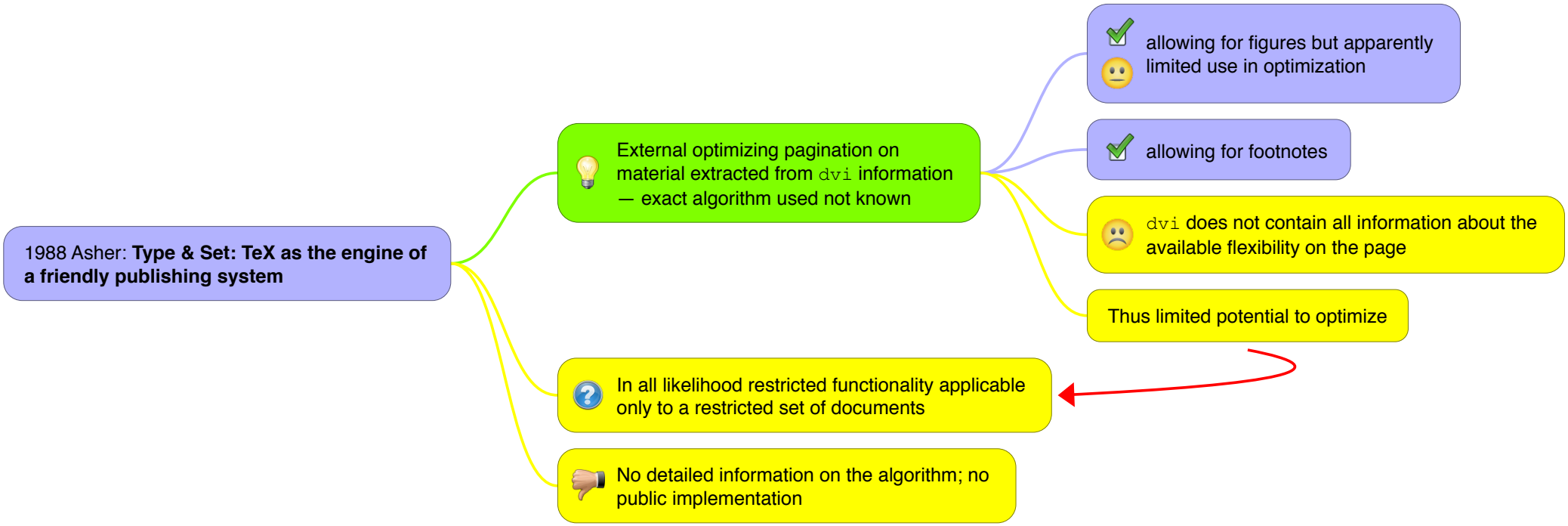

1981 Plass: **Optimal Pagination Techniques for Automatic Typesetting Systems**; PhD thesis
Applying the Knuth/Plass algorithm to pagination
allowing for figures
but not footnotes
no support for spread variations
no support for paragraph variations
Proving that certain badness functions lead to NP-complete problems (but so do certain functions on paragraph breaking)
Ground-breaking work — basis for everything else since then
No public implementation

1988 Asher: **Type & Set: TeX as the engine of a friendly publishing system**

External optimizing pagination on material extracted from `dvi` information — exact algorithm used not known

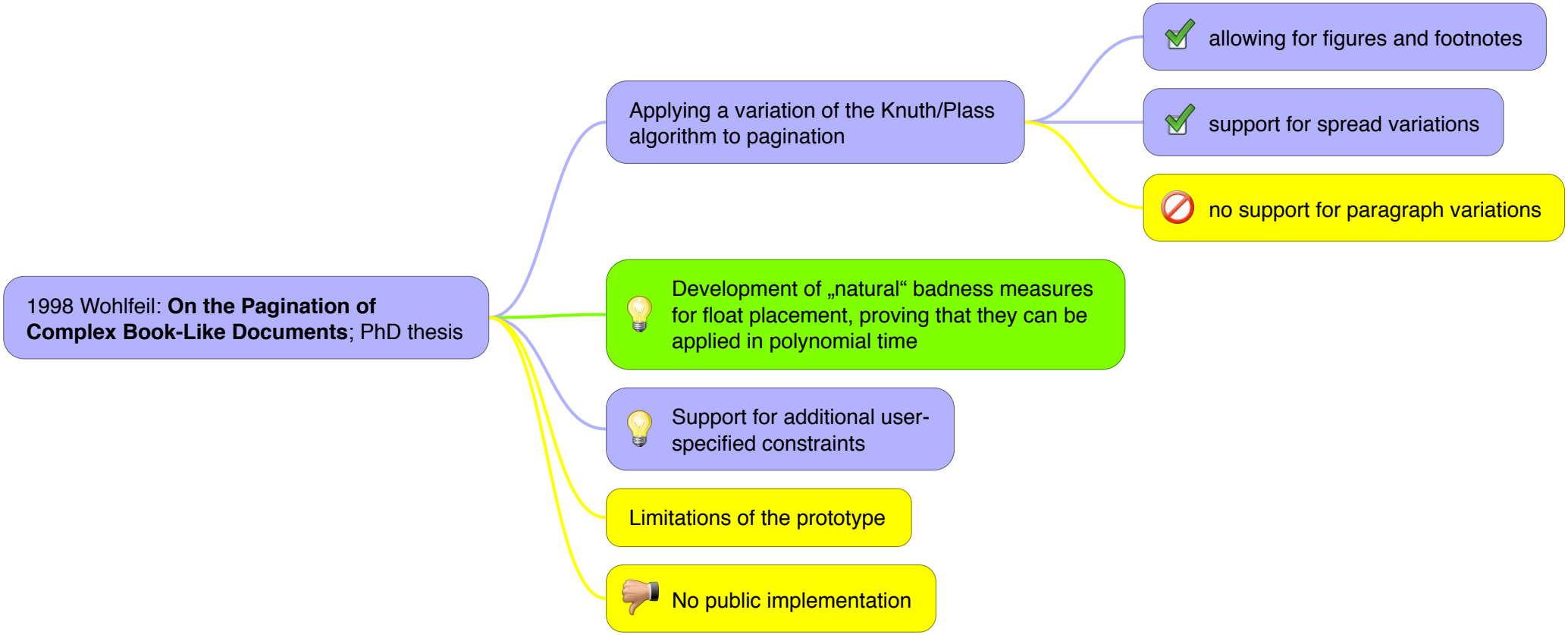
In all likelihood restricted functionality applicable only to a restricted set of documents

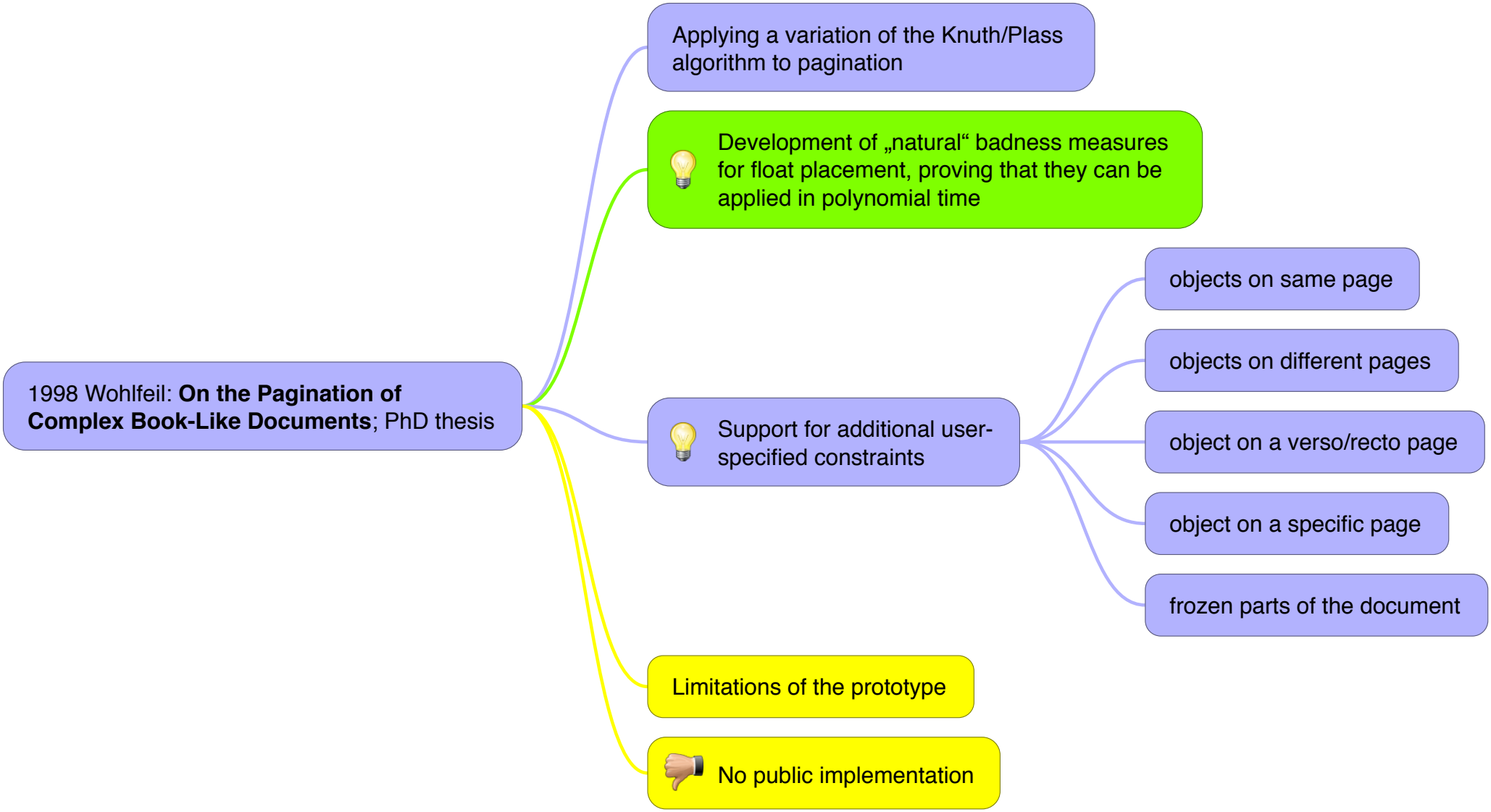
No detailed information on the algorithm; no public implementation

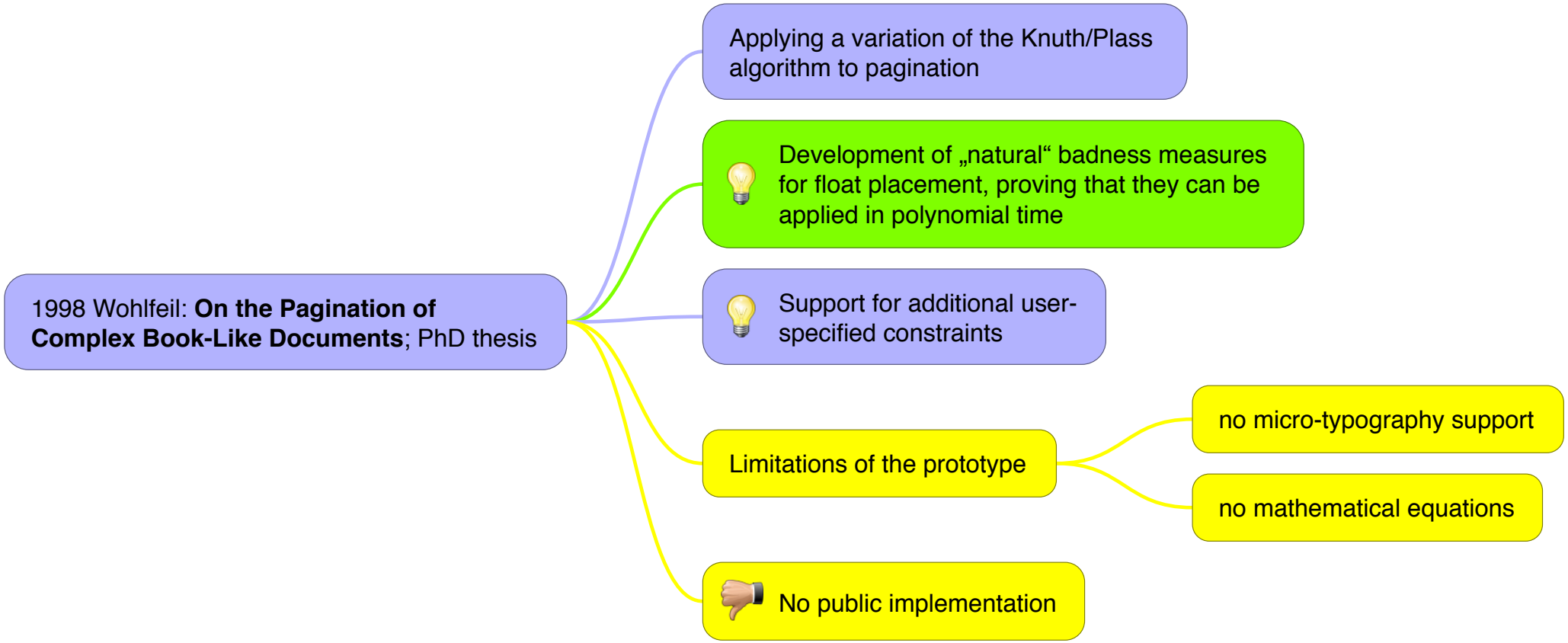

1988 Asher: **Type & Set: TeX as the engine of a friendly publishing system**
External optimizing pagination on material extracted from dvi information — exact algorithm used not known
allowing for figures but apparently limited use in optimization
allowing for footnotes
dvi does not contain all information about the available flexibility on the page
Thus limited potential to optimize
In all likelihood restricted functionality applicable only to a restricted set of documents
No detailed information on the algorithm; no public implementation

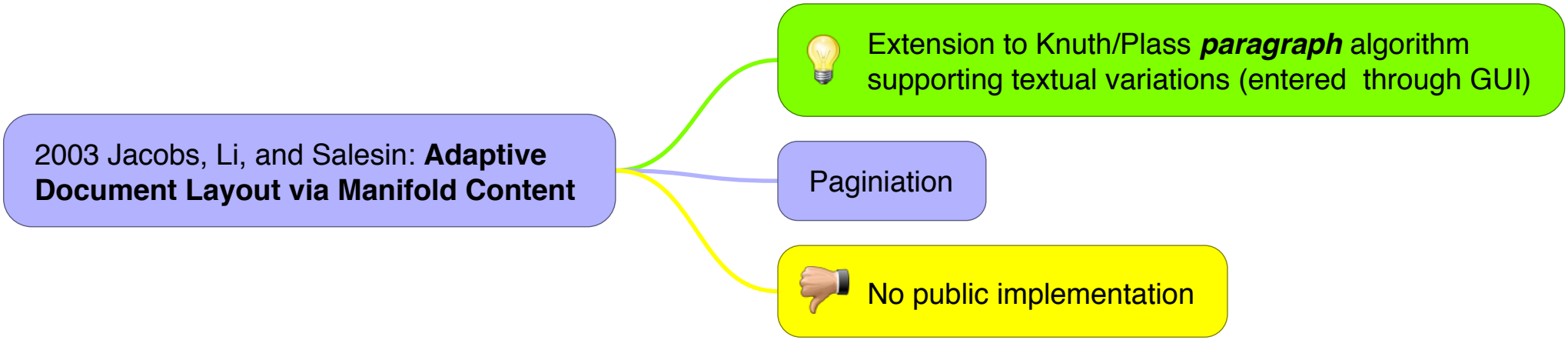
1998 Wohlfeil: **On the Pagination of Complex Book-Like Documents**; PhD thesis

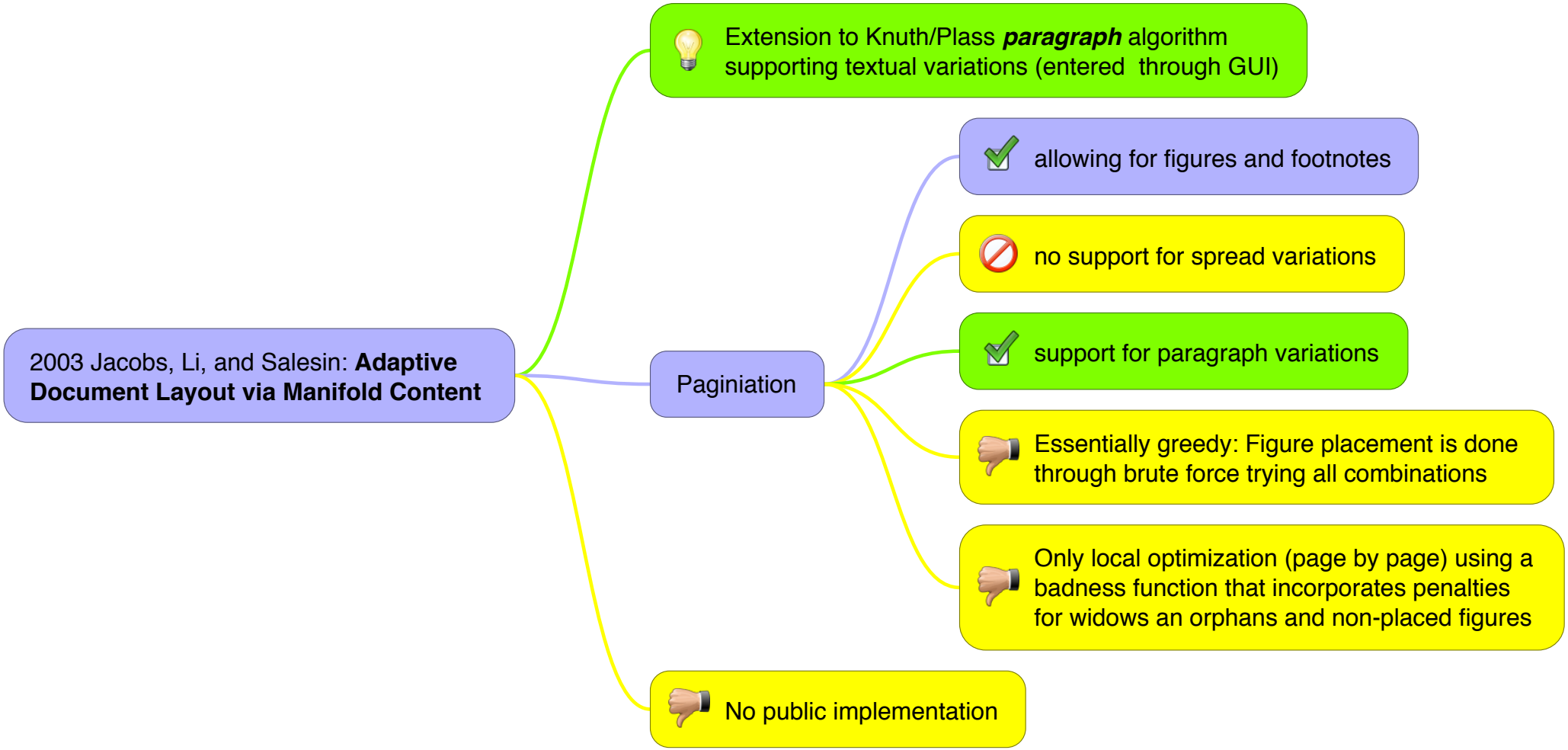
- Applying a variation of the Knuth/Plass algorithm to pagination
-  Development of „natural“ badness measures for float placement, proving that they can be applied in polynomial time
-  Support for additional user-specified constraints
- Limitations of the prototype
-  No public implementation

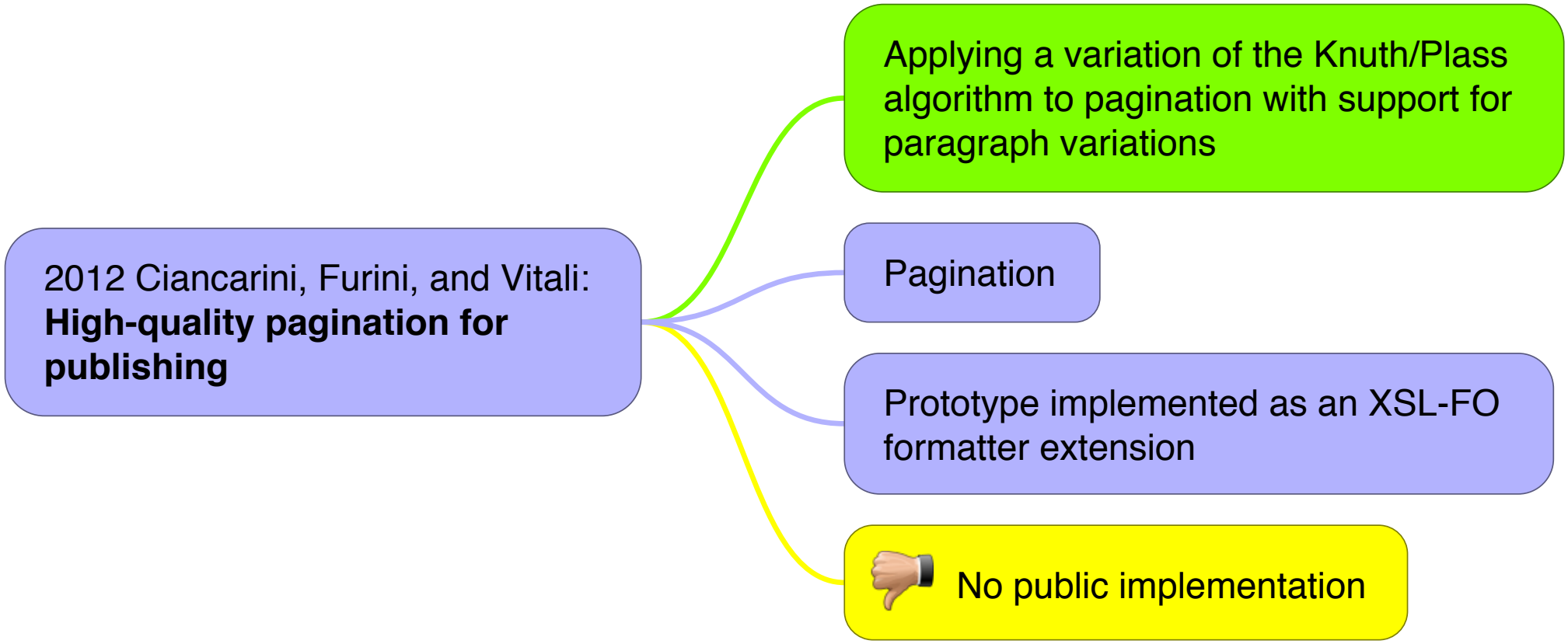

1998 Wohlfeil: **On the Pagination of Complex Book-Like Documents**; PhD thesis
Applying a variation of the Knuth/Plass algorithm to pagination
allowing for figures and footnotes
support for spread variations
no support for paragraph variations
Development of „natural“ badness measures for float placement, proving that they can be applied in polynomial time
Support for additional user-specified constraints
Limitations of the prototype
No public implementation

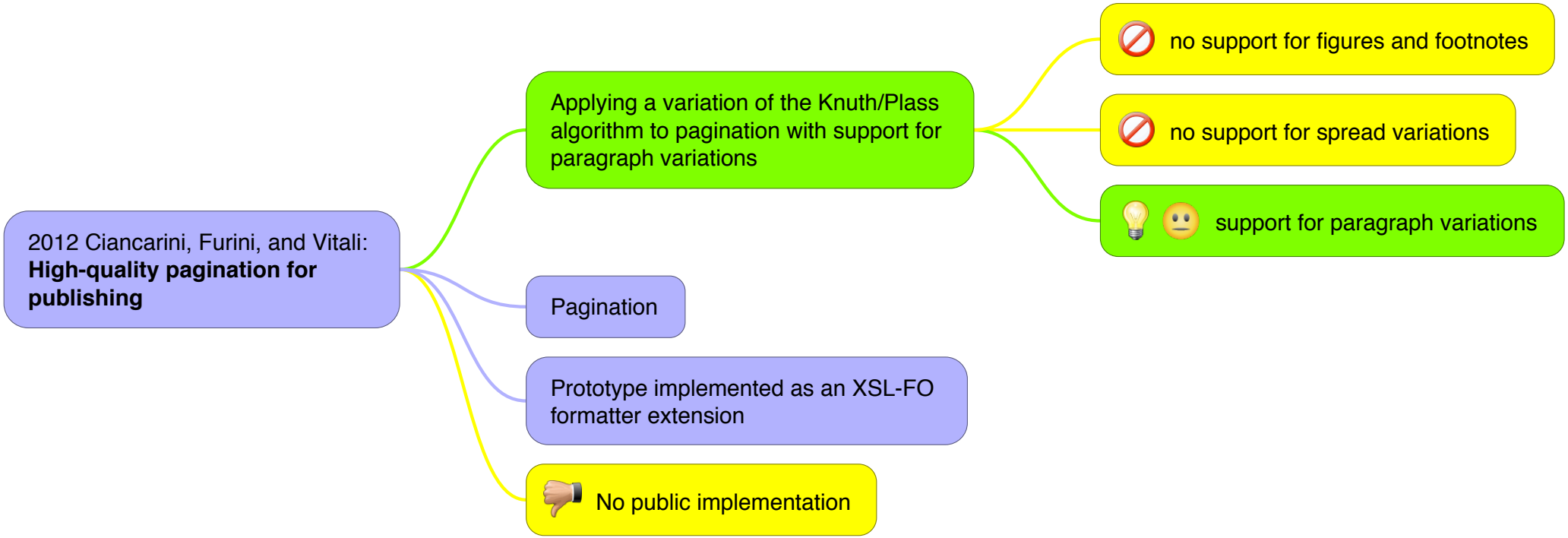

1998 Wohlfeil: **On the Pagination of Complex Book-Like Documents**; PhD thesis
Applying a variation of the Knuth/Plass algorithm to pagination
Development of „natural“ badness measures for float placement, proving that they can be applied in polynomial time
Support for additional user-specified constraints
objects on same page
objects on different pages
object on a verso/recto page
object on a specific page
frozen parts of the document
Limitations of the prototype
No public implementation

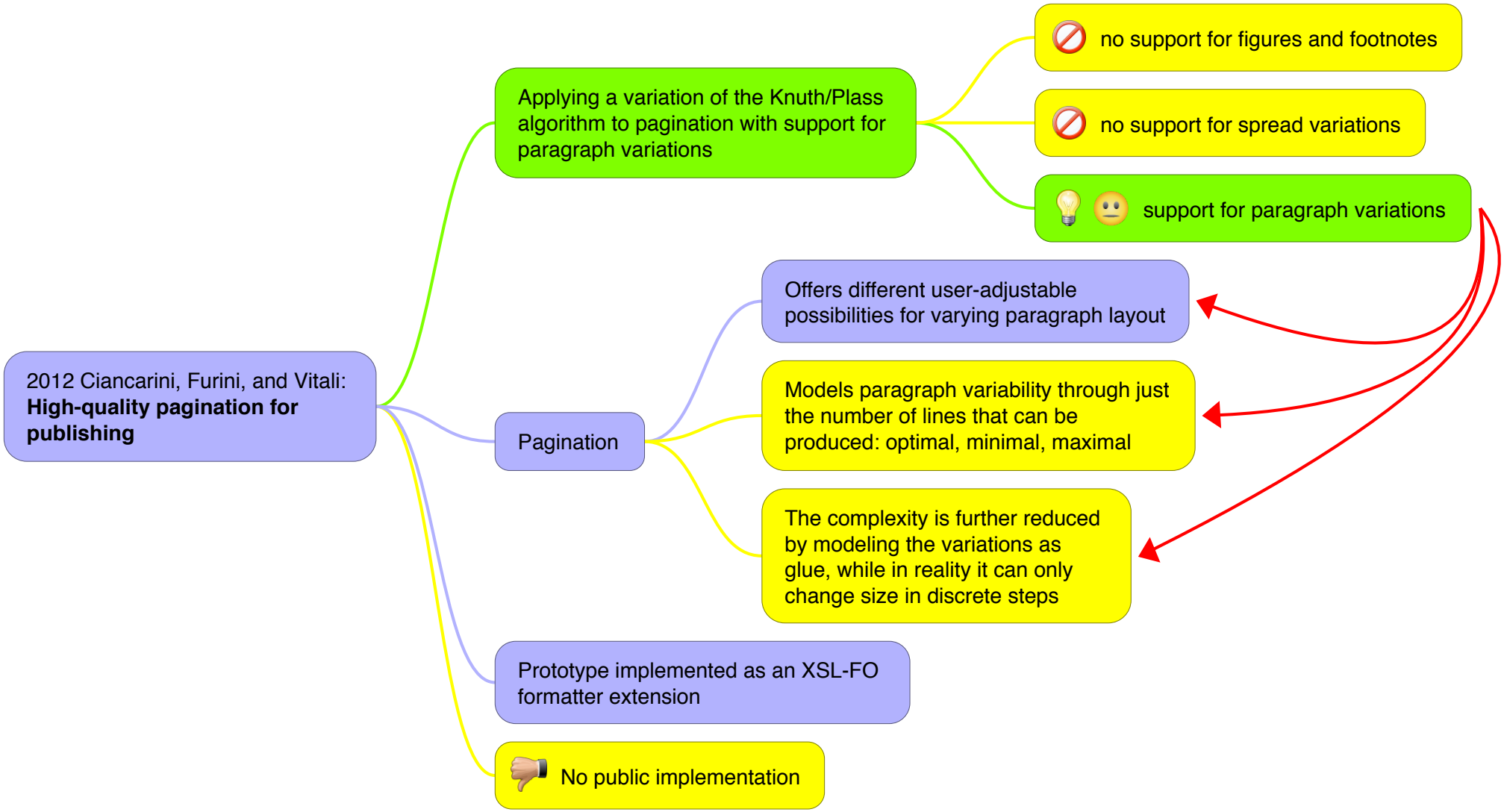

1998 Wohlfeil: **On the Pagination of Complex Book-Like Documents**; PhD thesis
Applying a variation of the Knuth/Plass algorithm to pagination
Development of „natural“ badness measures for float placement, proving that they can be applied in polynomial time
Support for additional user-specified constraints
Limitations of the prototype
no micro-typography support
no mathematical equations
No public implementation

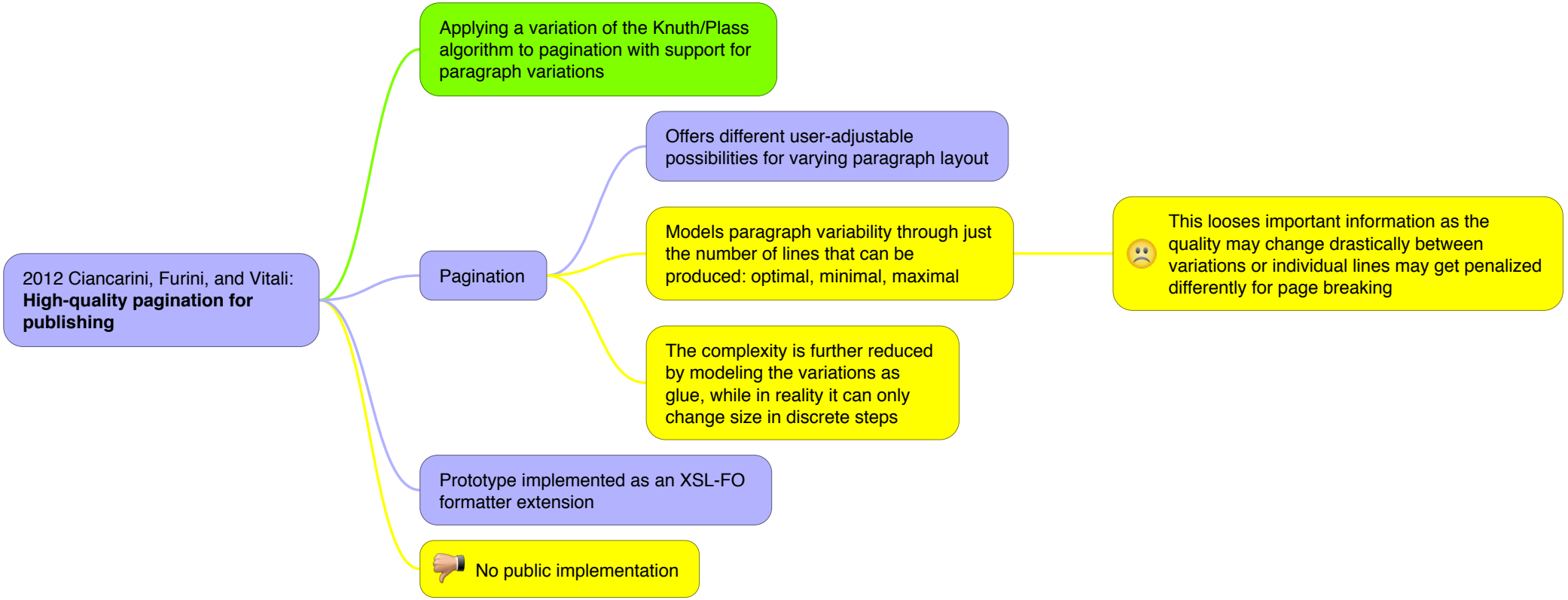

2003 Jacobs, Li, and Salesin: **Adaptive Document Layout via Manifold Content**
Extension to Knuth/Plass ***paragraph*** algorithm supporting textual variations (entered through GUI)
Paginiation
No public implementation

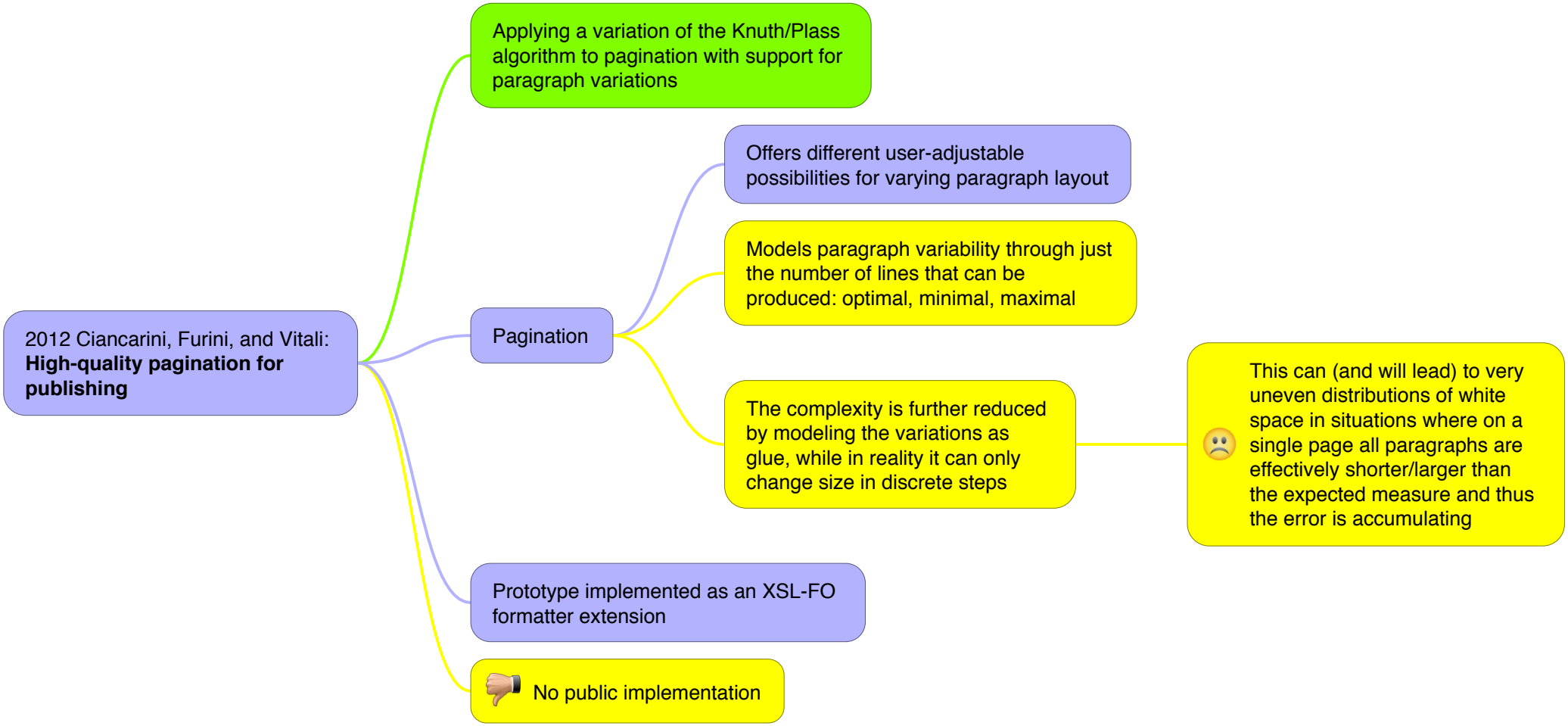

2003 Jacobs, Li, and Salesin: **Adaptive Document Layout via Manifold Content**
Extension to Knuth/Plass ***paragraph*** algorithm supporting textual variations (entered through GUI)
Paginiation
allowing for figures and footnotes
no support for spread variations
support for paragraph variations
Essentially greedy: Figure placement is done through brute force trying all combinations
Only local optimization (page by page) using a badness function that incorporates penalties for widows an orphans and non-placed figures
No public implementation

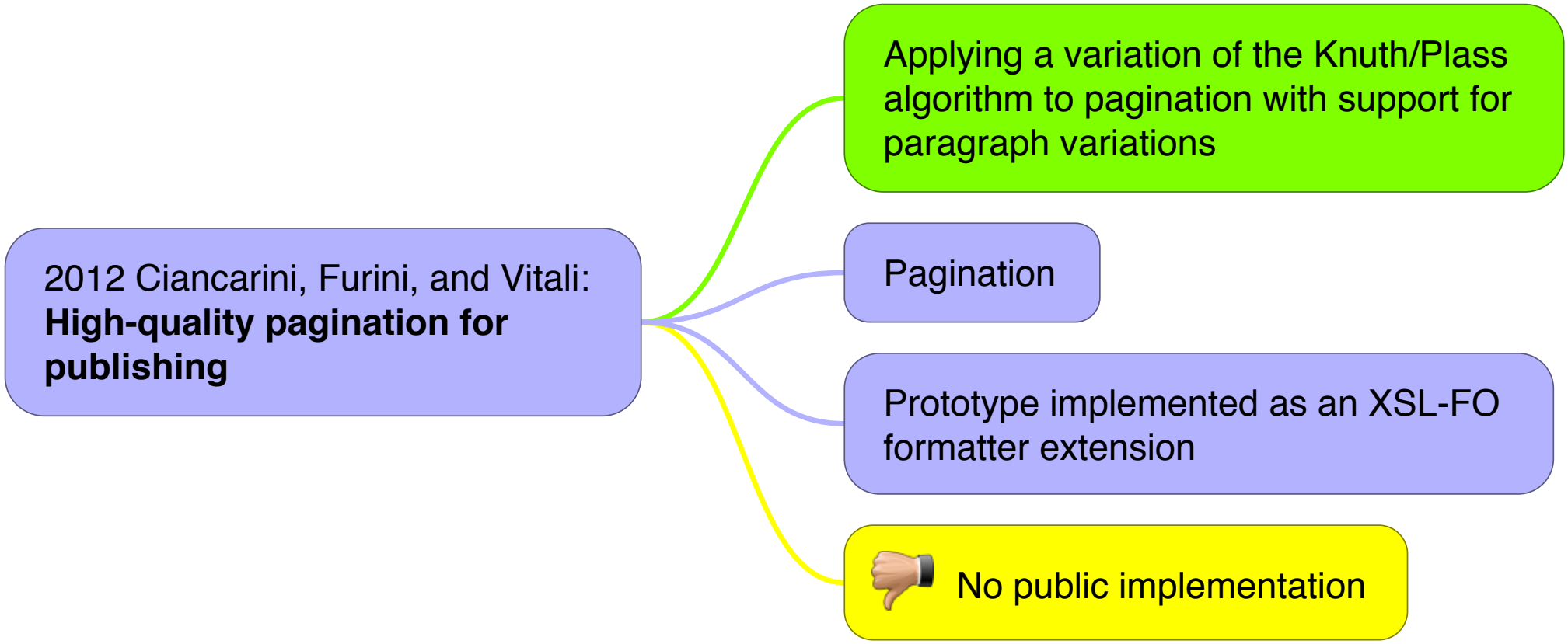

2012 Ciancarini, Furini, and Vitali:
High-quality pagination for publishing
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
Pagination
Prototype implemented as an XSL-FO formatter extension
No public implementation


2012 Ciancarini, Furini, and Vitali: **High-quality pagination for publishing**
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
no support for figures and footnotes
no support for spread variations
support for paragraph variations
Pagination
Prototype implemented as an XSL-FO formatter extension
No public implementation

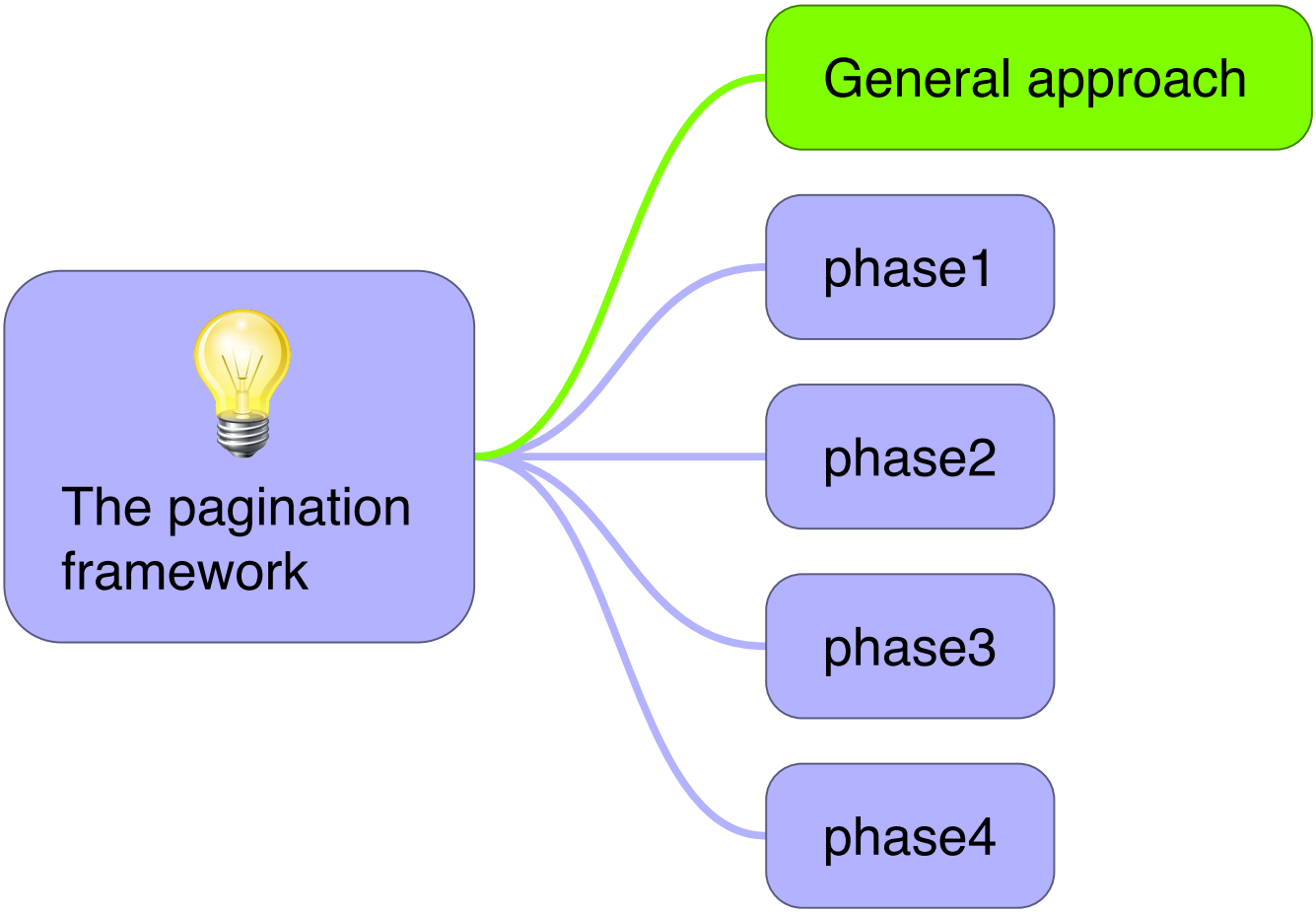

2012 Ciancarini, Furini, and Vitali: **High-quality pagination for publishing**
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
no support for figures and footnotes
no support for spread variations
support for paragraph variations
Pagination
Offers different user-adjustable possibilities for varying paragraph layout
Models paragraph variability through just the number of lines that can be produced: optimal, minimal, maximal
The complexity is further reduced by modeling the variations as glue, while in reality it can only change size in discrete steps
Prototype implemented as an XSL-FO formatter extension
No public implementation

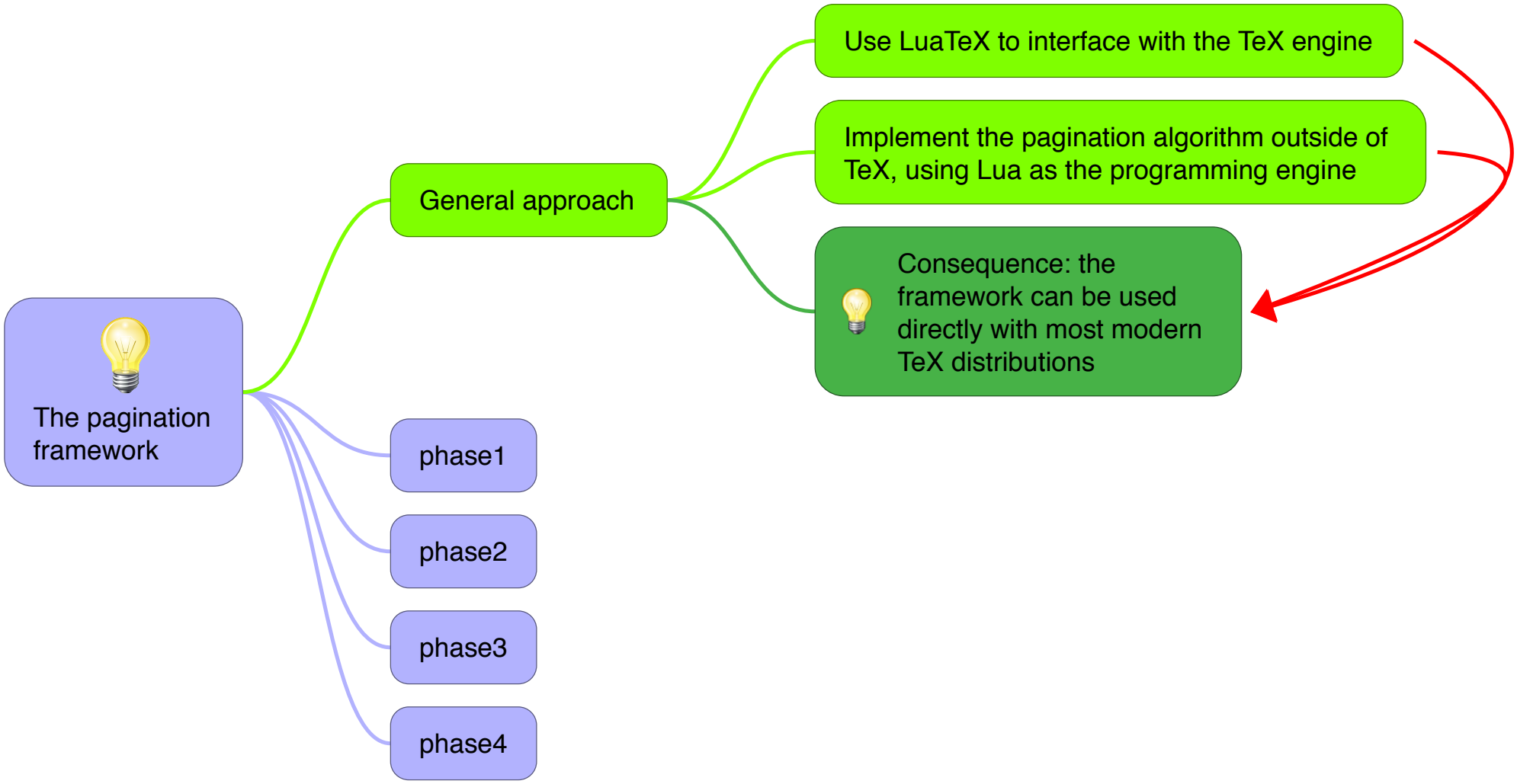

2012 Ciancarini, Furini, and Vitali: **High-quality pagination for publishing**
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
Pagination
Offers different user-adjustable possibilities for varying paragraph layout
Models paragraph variability through just the number of lines that can be produced: optimal, minimal, maximal
☹️
This looses important information as the quality may change drastically between variations or individual lines may get penalized differently for page breaking
The complexity is further reduced by modeling the variations as glue, while in reality it can only change size in discrete steps
Prototype implemented as an XSL-FO formatter extension
👎🏼
No public implementation

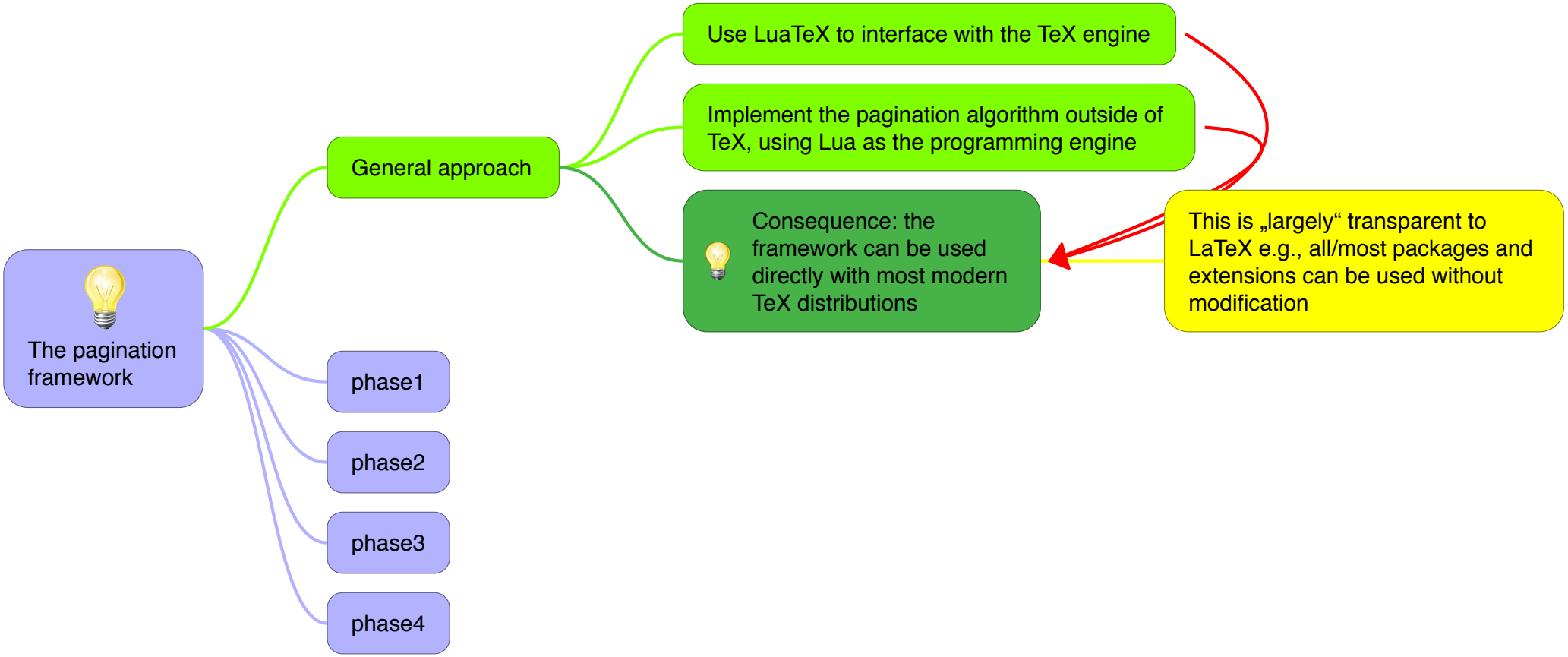

2012 Ciancarini, Furini, and Vitali: **High-quality pagination for publishing**
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
Pagination
Offers different user-adjustable possibilities for varying paragraph layout
Models paragraph variability through just the number of lines that can be produced: optimal, minimal, maximal
The complexity is further reduced by modeling the variations as glue, while in reality it can only change size in discrete steps
☹️
This can (and will lead) to very uneven distributions of white space in situations where on a single page all paragraphs are effectively shorter/larger than the expected measure and thus the error is accumulating
Prototype implemented as an XSL-FO formatter extension
👎🏽
No public implementation

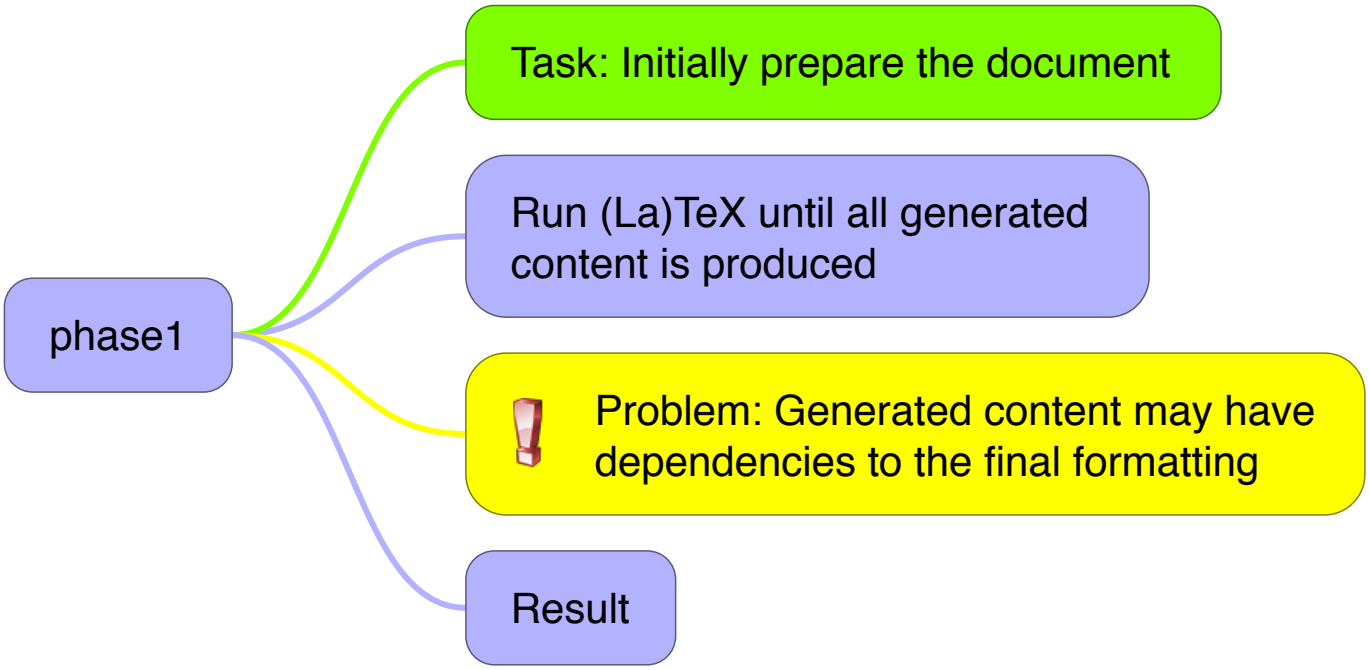

2012 Ciancarini, Furini, and Vitali:
High-quality pagination for publishing
Applying a variation of the Knuth/Plass algorithm to pagination with support for paragraph variations
Pagination
Prototype implemented as an XSL-FO formatter extension
No public implementation

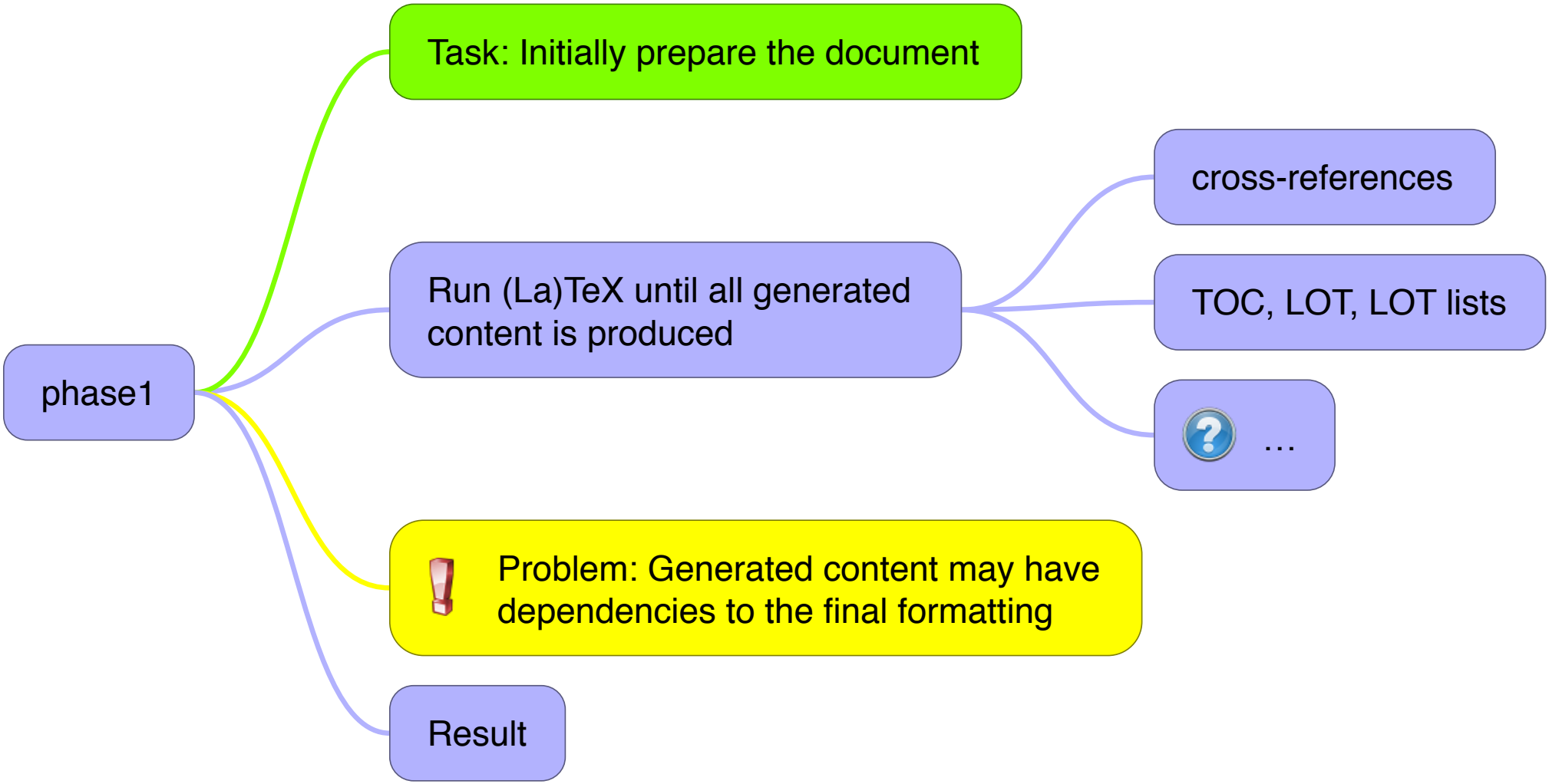
The pagination framework

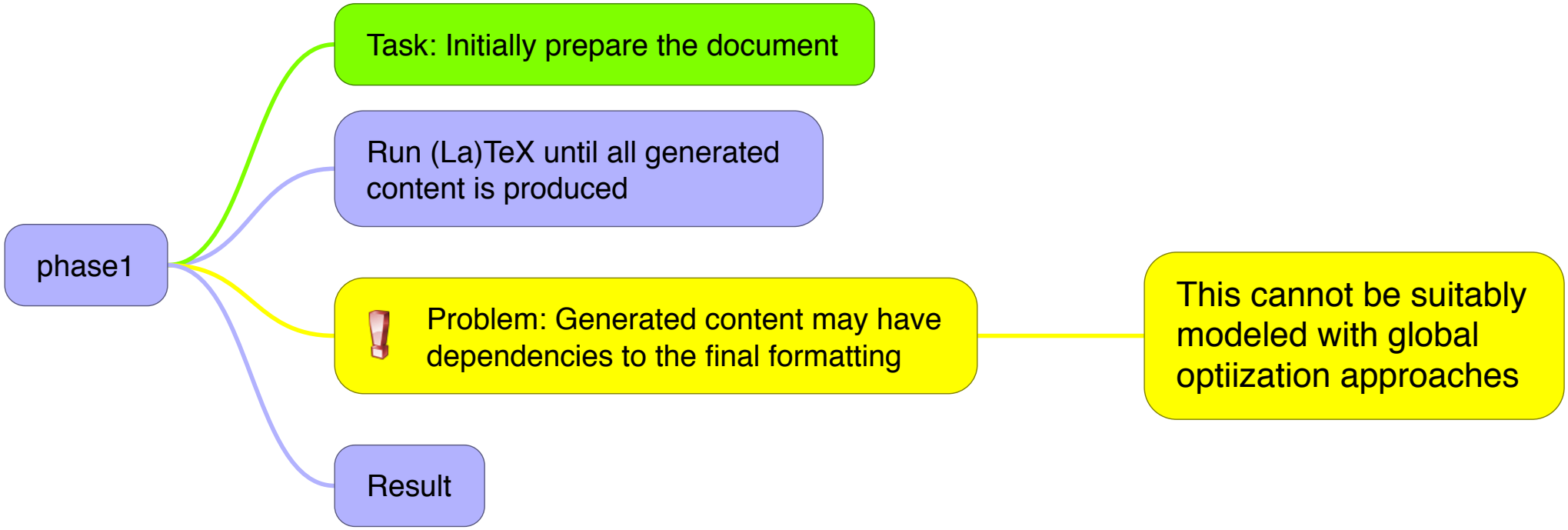

General approach
phase1
phase2
phase3
phase4
The pagination framework

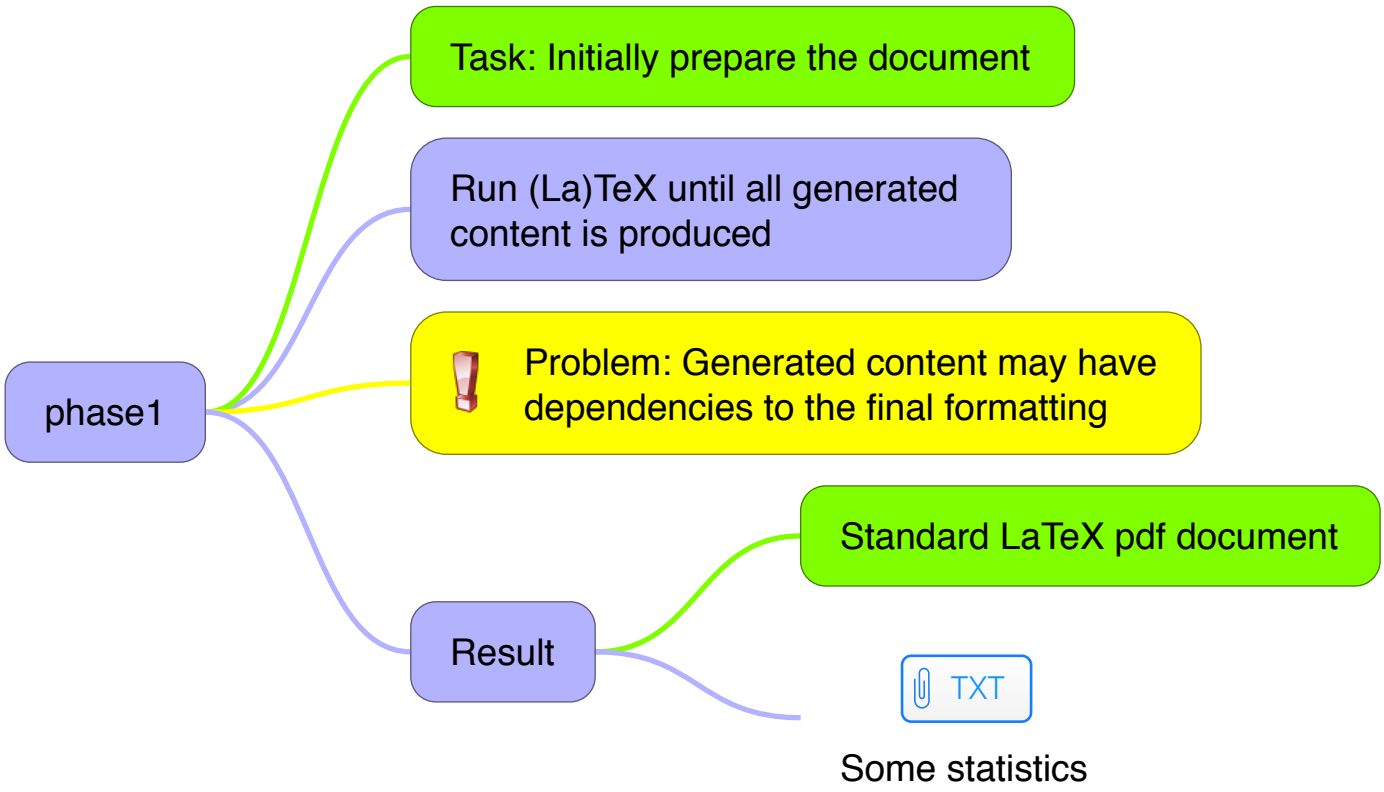

The pagination framework
General approach
Use LuaTeX to interface with the TeX engine
Implement the pagination algorithm outside of TeX, using Lua as the programming engine
Consequence: the framework can be used directly with most modern TeX distributions
phase1
phase2
phase3
phase4


The pagination framework
General approach
Use LuaTeX to interface with the TeX engine
Implement the pagination algorithm outside of TeX, using Lua as the programming engine
Consequence: the framework can be used directly with most modern TeX distributions
This is „largely“ transparent to LaTeX e.g., all/most packages and extensions can be used without modification
phase1
phase2
phase3
phase4

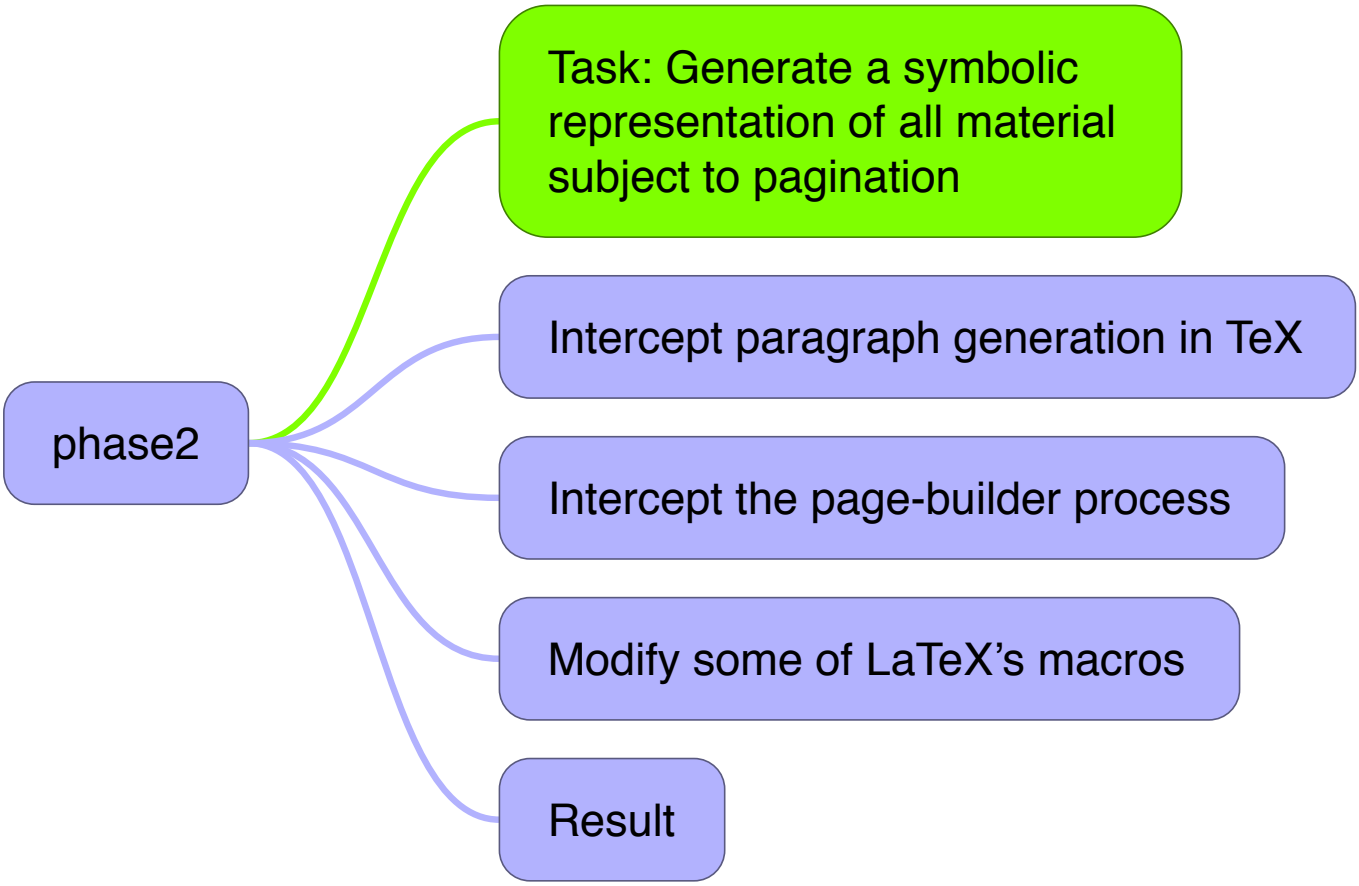

phase1
Task: Initially prepare the document
Run (La)TeX until all generated content is produced
Problem: Generated content may have dependencies to the final formatting
Result

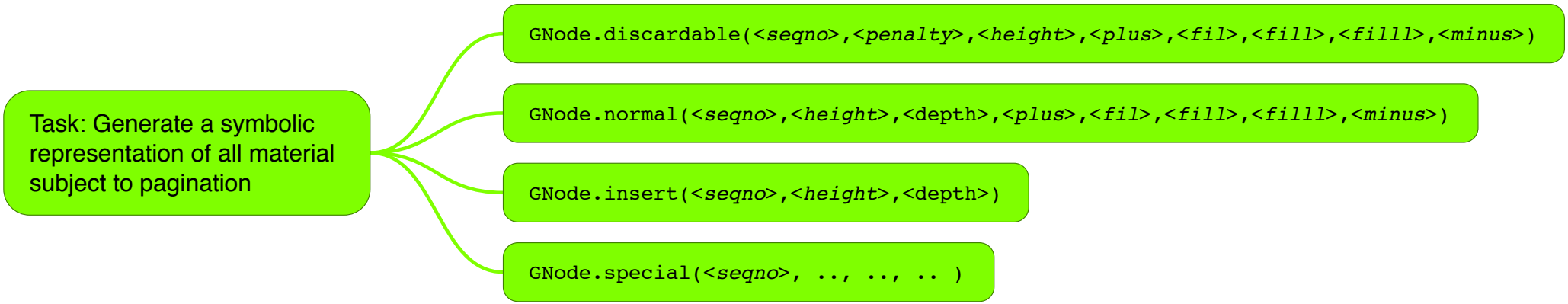

phase1
Task: Initially prepare the document
Run (La)TeX until all generated content is produced
cross-references
TOC, LOT, LOT lists
…
Problem: Generated content may have dependencies to the final formatting
Result

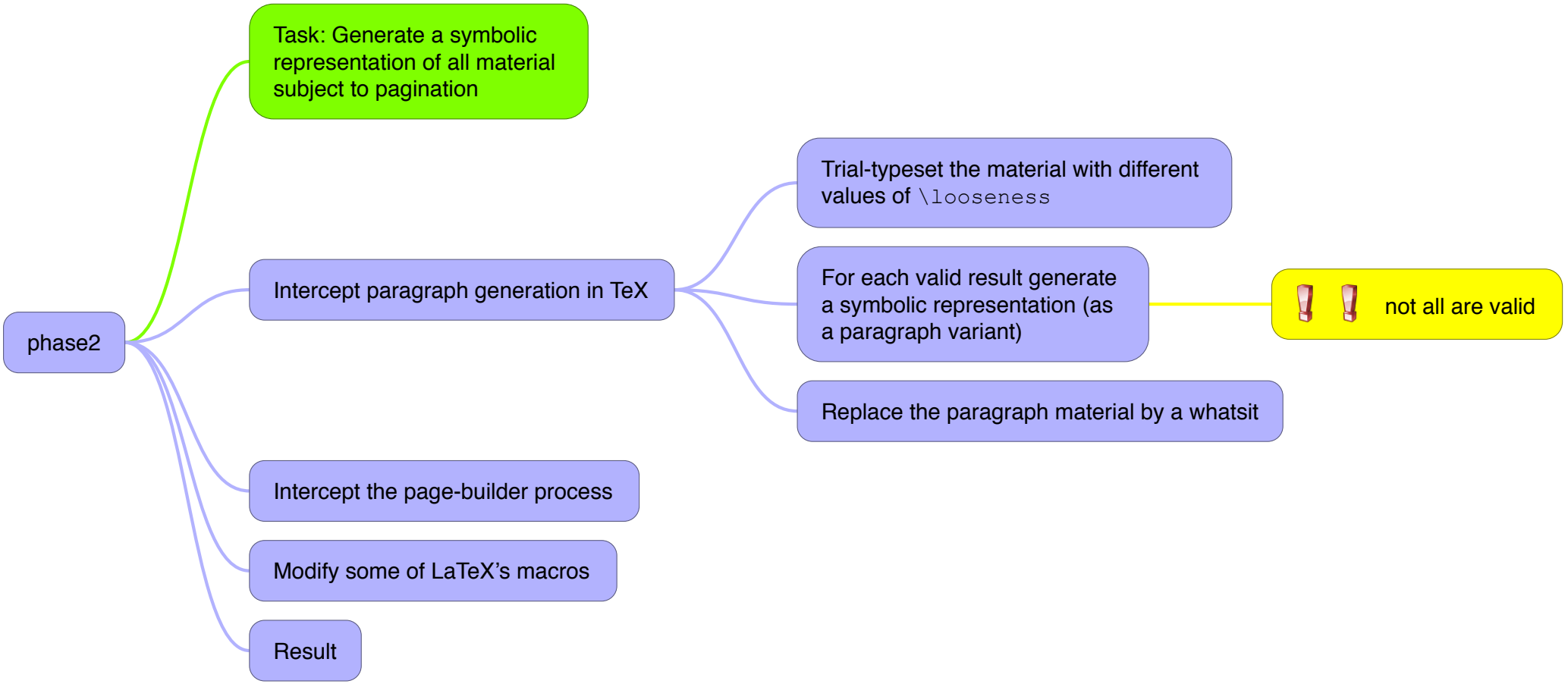

phase1
Task: Initially prepare the document
Run (La)TeX until all generated content is produced
Problem: Generated content may have dependencies to the final formatting
This cannot be suitably modeled with global optiization approaches
Result

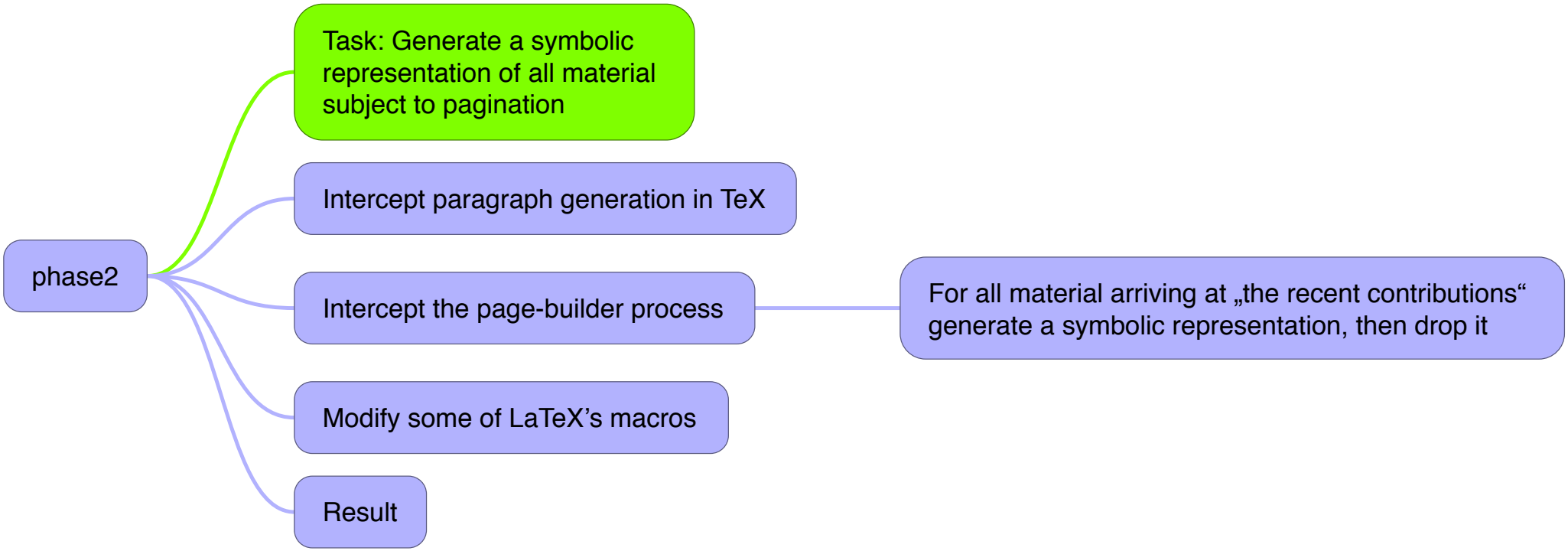

phase1
Task: Initially prepare the document
Run (La)TeX until all generated content is produced
Problem: Generated content may have dependencies to the final formatting
Result
Standard LaTeX pdf document
TXT
Some statistics

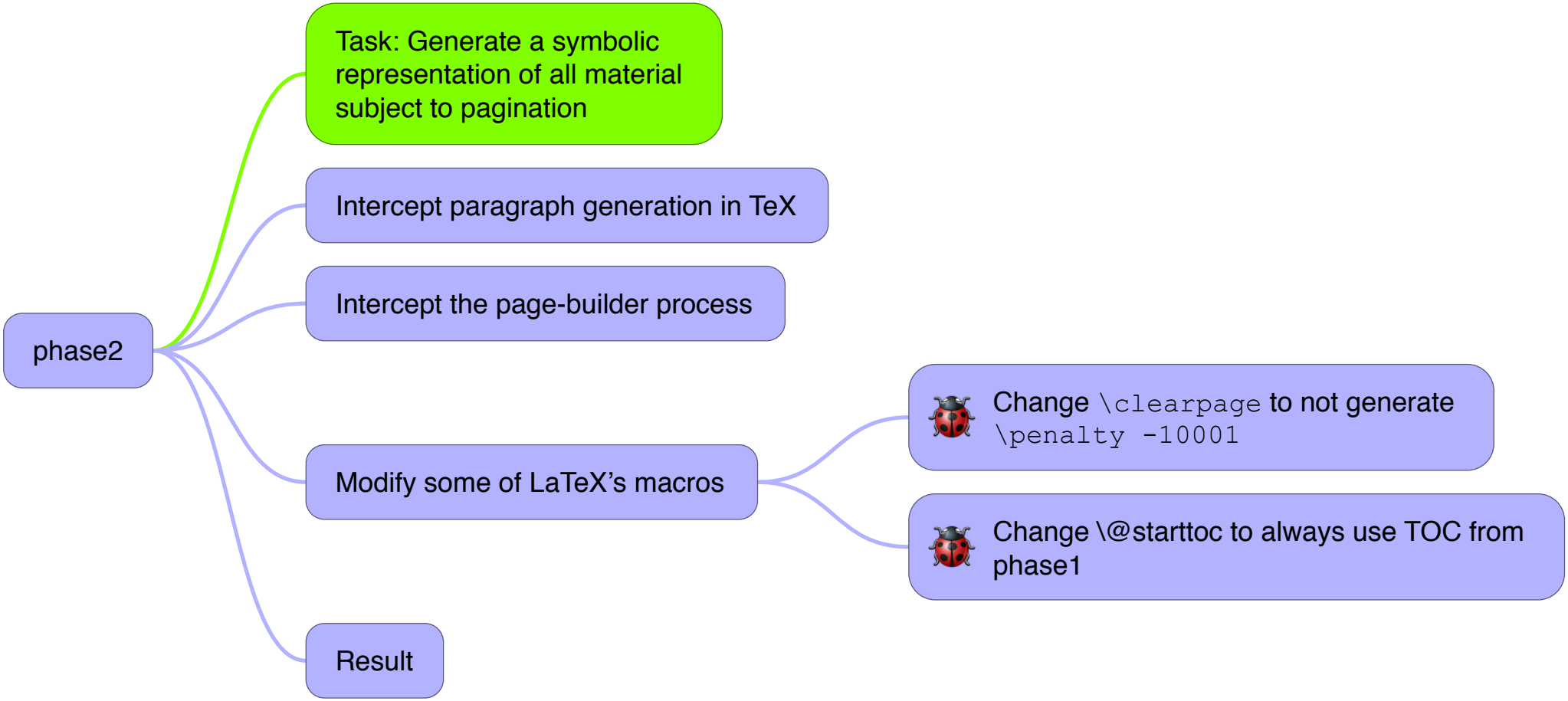
Some statistics

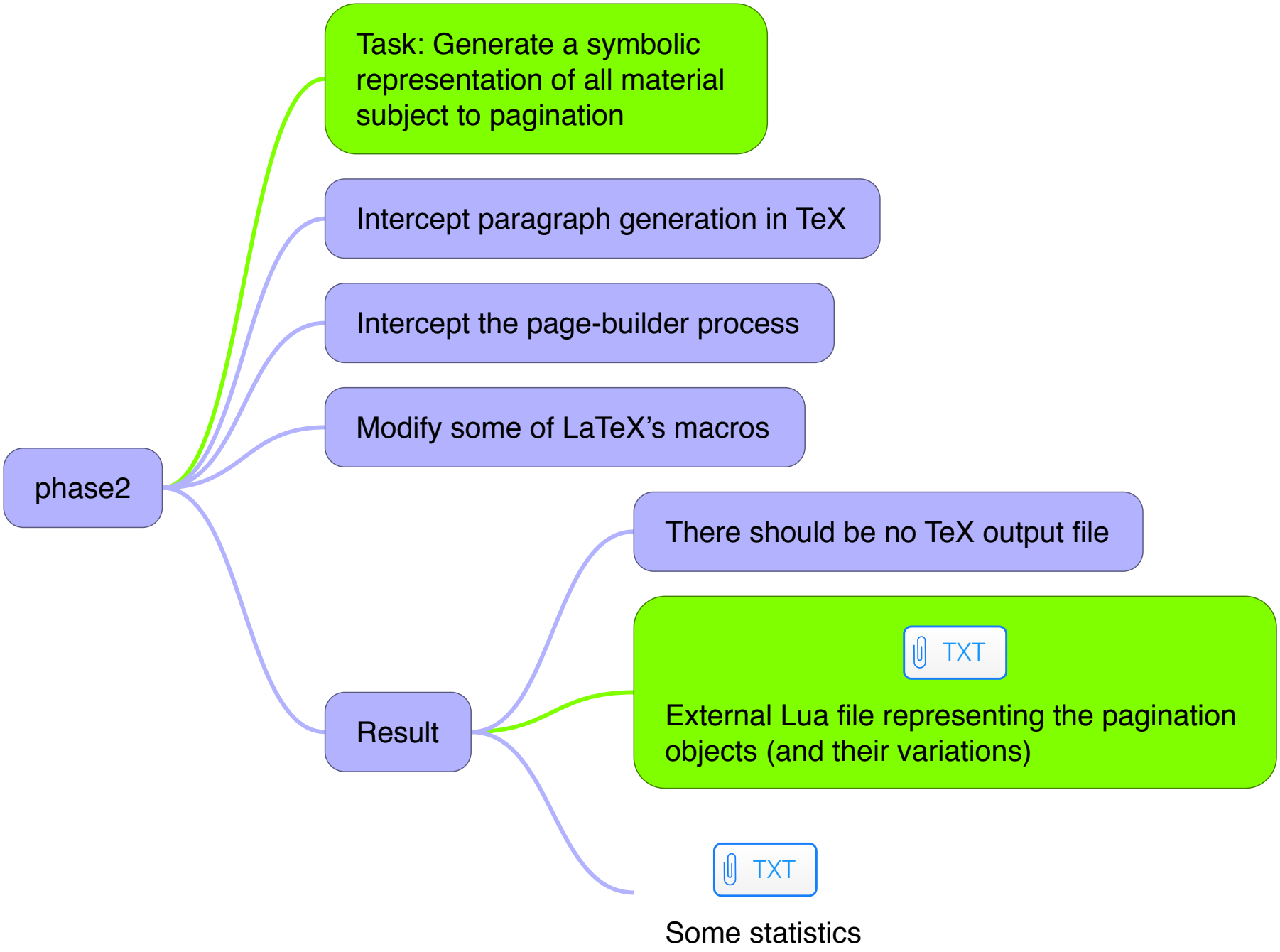

phase2
Task: Generate a symbolic representation of all material subject to pagination
Intercept paragraph generation in TeX
Intercept the page-builder process
Modify some of LaTeX's macros
Result


Task: Generate a symbolic representation of all material subject to pagination
GNode.discardable(<seqno>,<penalty>,<height>,<plus>,<fil>,<fill>,<filll>,<minus>)
GNode.normal(<seqno>,<height>,<depth>,<plus>,<fil>,<fill>,<filll>,<minus>)
GNode.insert(<seqno>,<height>,<depth>)
GNode.special(<seqno>, .., .., ..)


phase2
Task: Generate a symbolic representation of all material subject to pagination
Intercept paragraph generation in TeX
Trial-typeset the material with different values of \looseness
For each valid result generate a symbolic representation (as a paragraph variant)
not all are valid
Replace the paragraph material by a whatsit
Intercept the page-builder process
Modify some of LaTeX's macros
Result

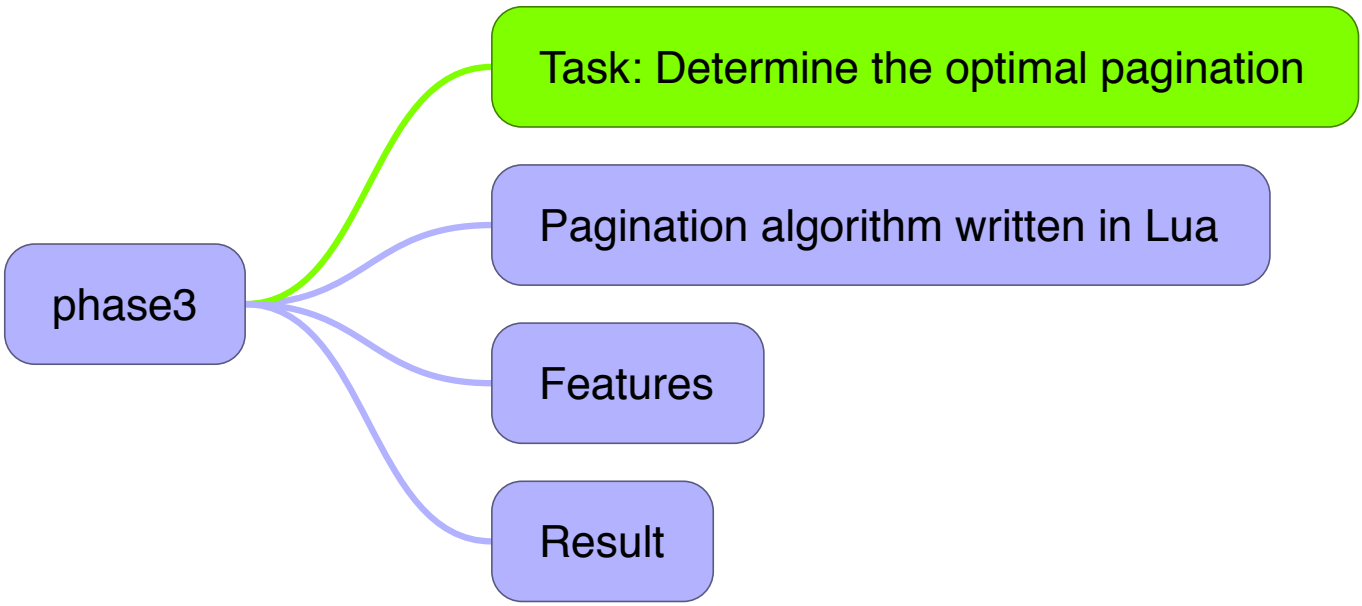

phase2
Task: Generate a symbolic representation of all material subject to pagination
Intercept paragraph generation in TeX
Intercept the page-builder process
For all material arriving at „the recent contributions“ generate a symbolic representation, then drop it
Modify some of LaTeX’s macros
Result

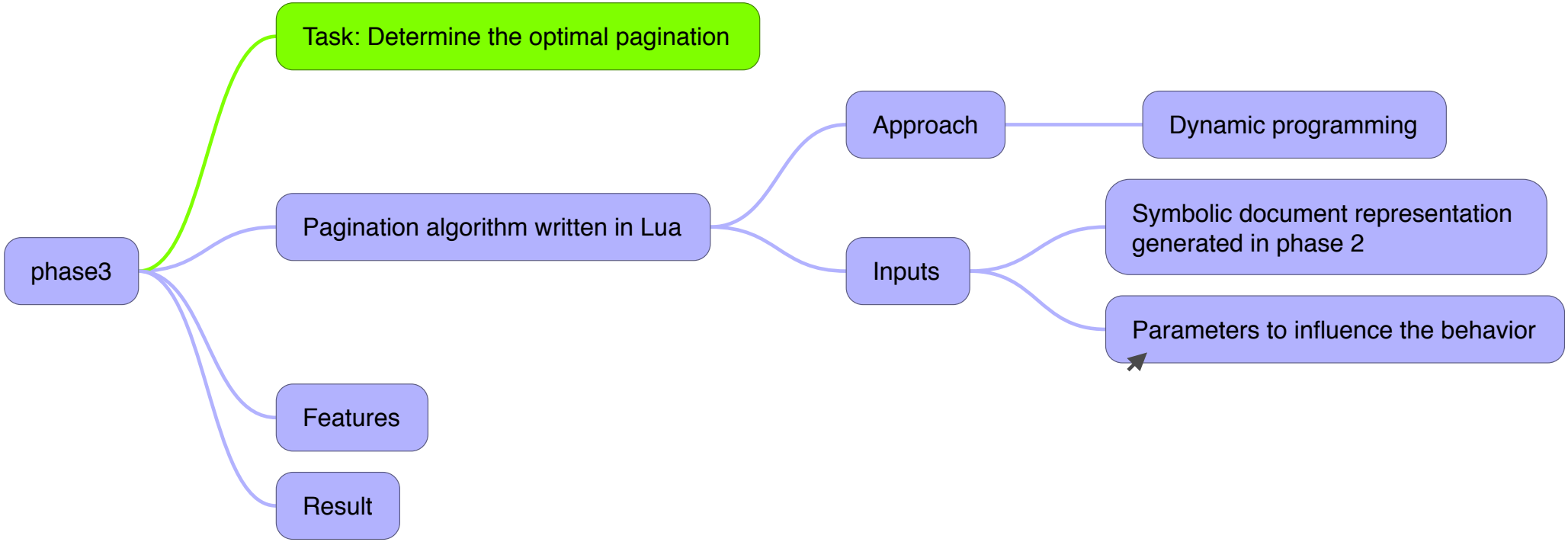

phase2
Task: Generate a symbolic representation of all material subject to pagination
Intercept paragraph generation in TeX
Intercept the page-builder process
Modify some of LaTeX's macros
Change \clearpage to not generate \penalty -10001
Change \@starttoc to always use TOC from phase1
Result

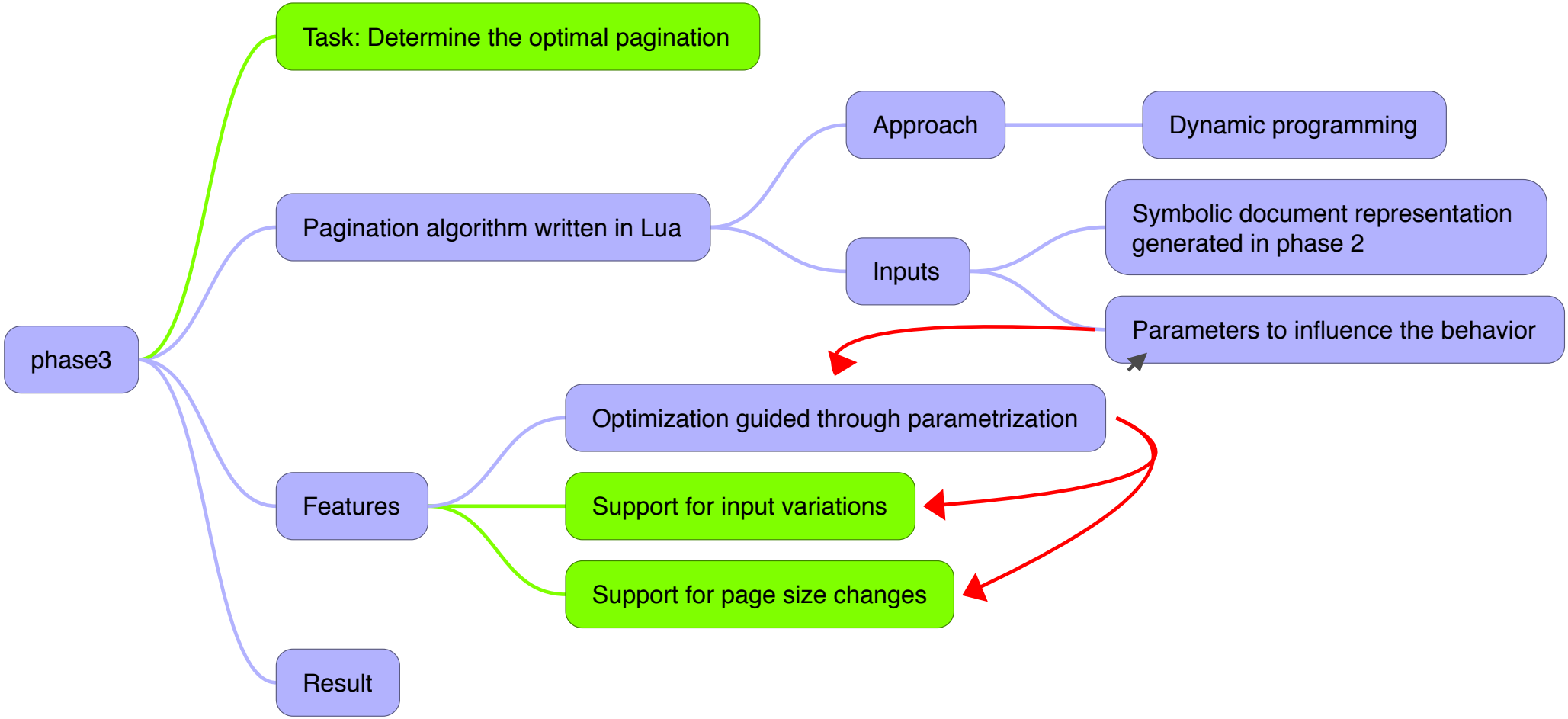

phase2
Task: Generate a symbolic representation of all material subject to pagination
Intercept paragraph generation in TeX
Intercept the page-builder process
Modify some of LaTeX's macros
Result
There should be no TeX output file
TXT
External Lua file representing the pagination objects (and their variations)
TXT
Some statistics

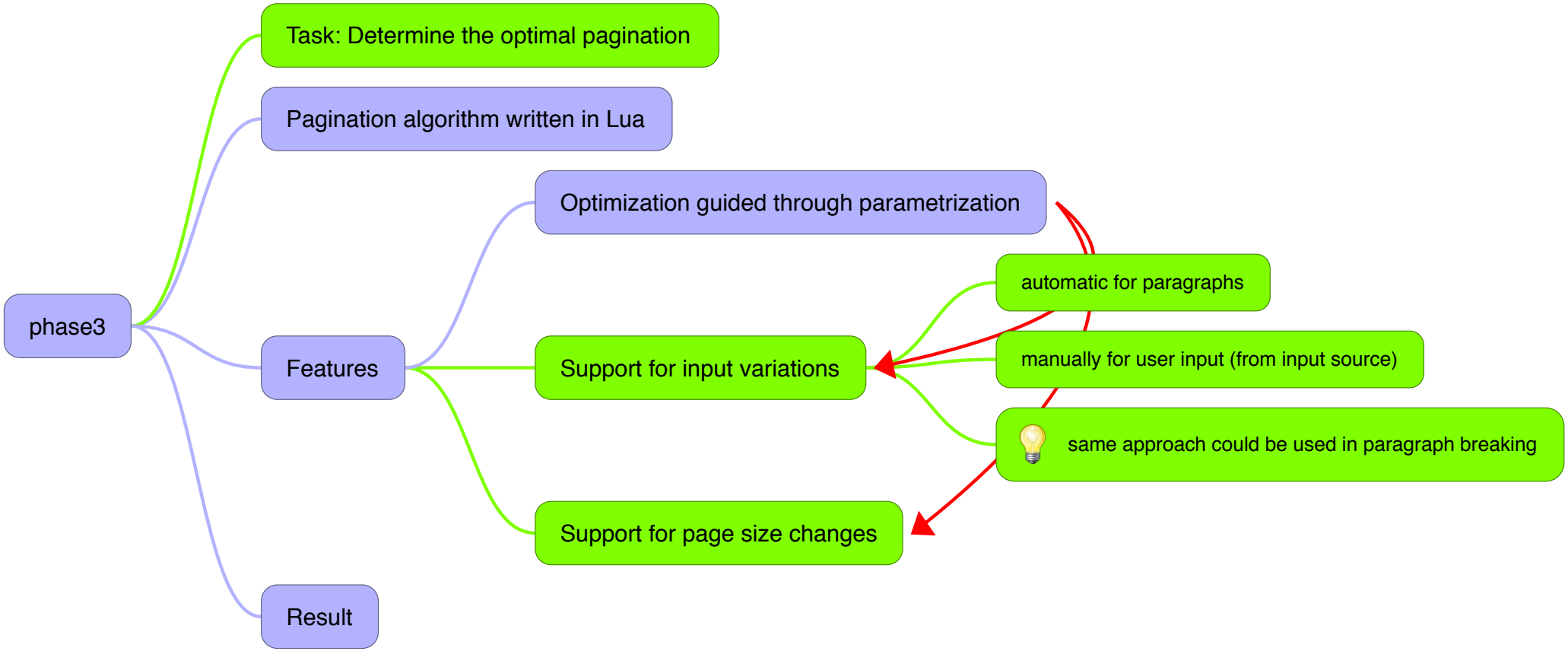
External Lua file representing the pagination objects (and their variations)

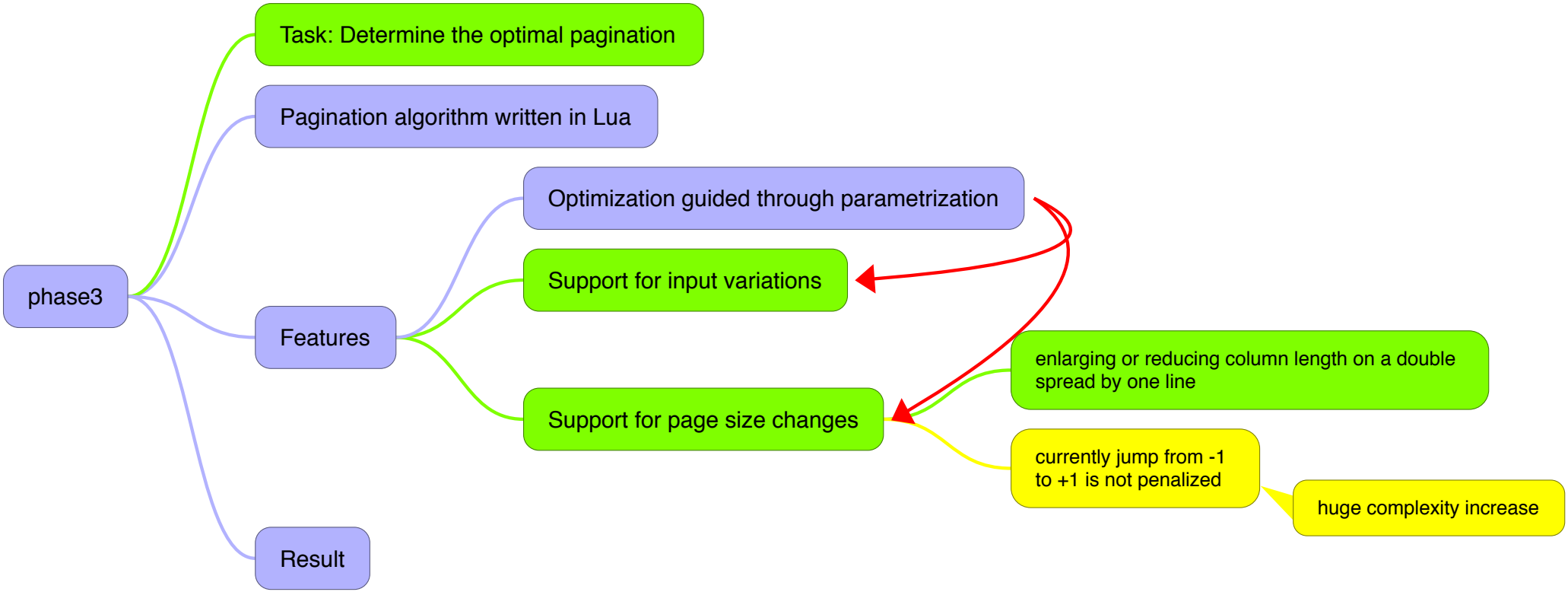
Some statistics

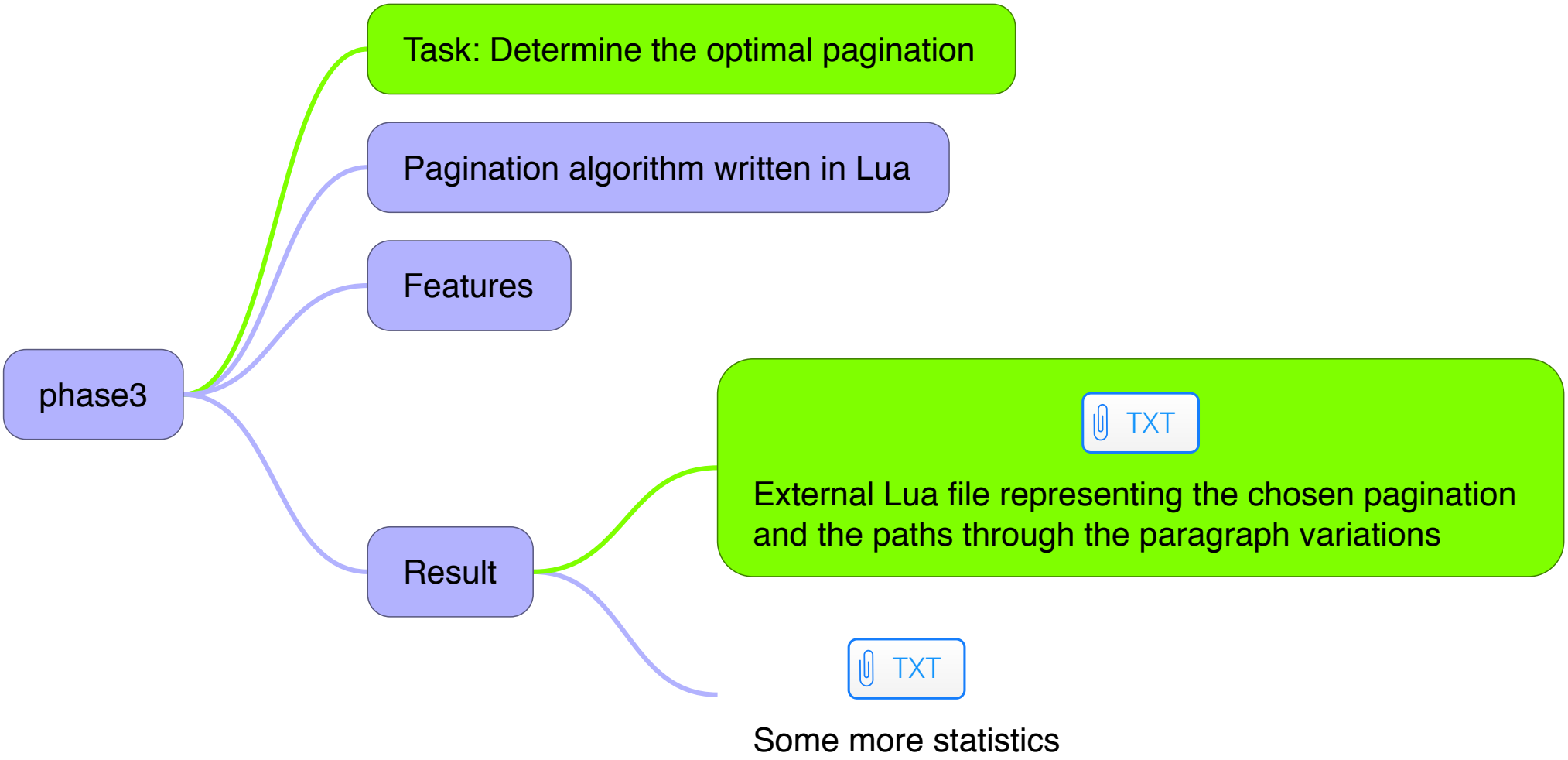

phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Features
Result


phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Approach
Dynamic programming
Inputs
Symbolic document representation generated in phase 2
Parameters to influence the behavior
Features
Result


phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Approach
Dynamic programming
Inputs
Symbolic document representation generated in phase 2
Parameters to influence the behavior
Features
Optimization guided through parametrization
Support for input variations
Support for page size changes
Result

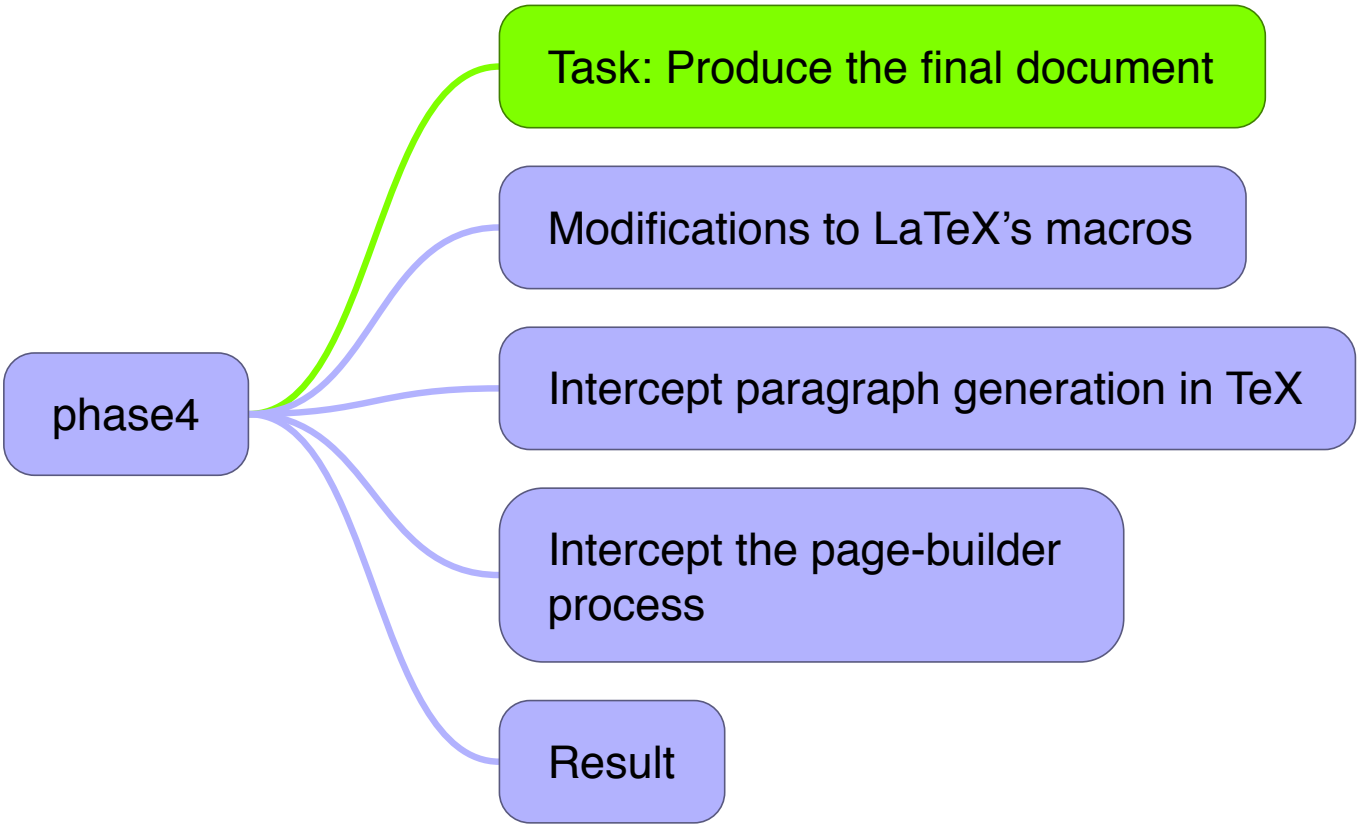

phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Features
Optimization guided through parametrization
Support for input variations
automatic for paragraphs
manually for user input (from input source)
same approach could be used in paragraph breaking
Support for page size changes
Result

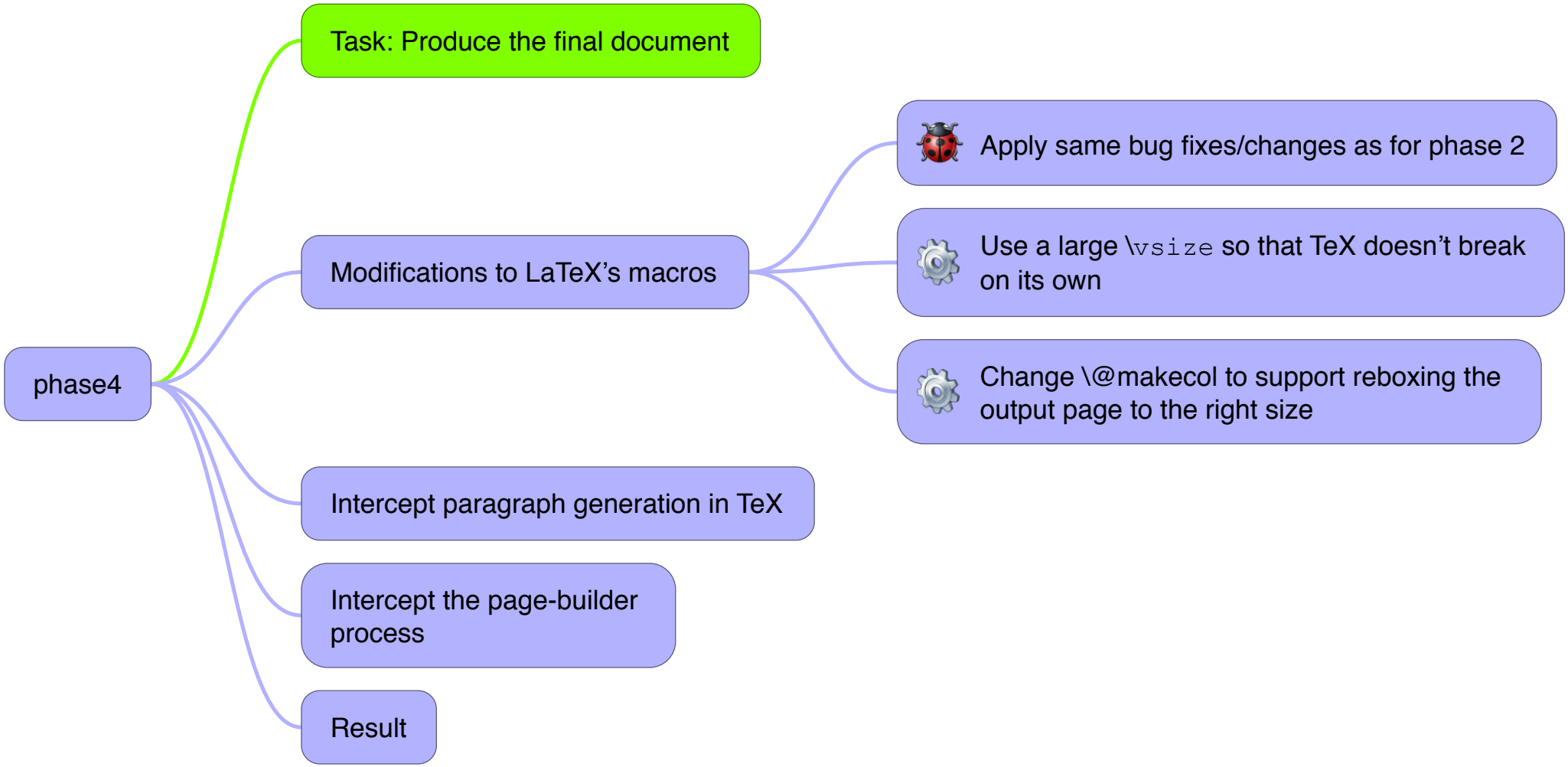

phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Features
Optimization guided through parametrization
Support for input variations
Support for page size changes
enlarging or reducing column length on a double spread by one line
currently jump from -1 to +1 is not penalized
huge complexity increase
Result

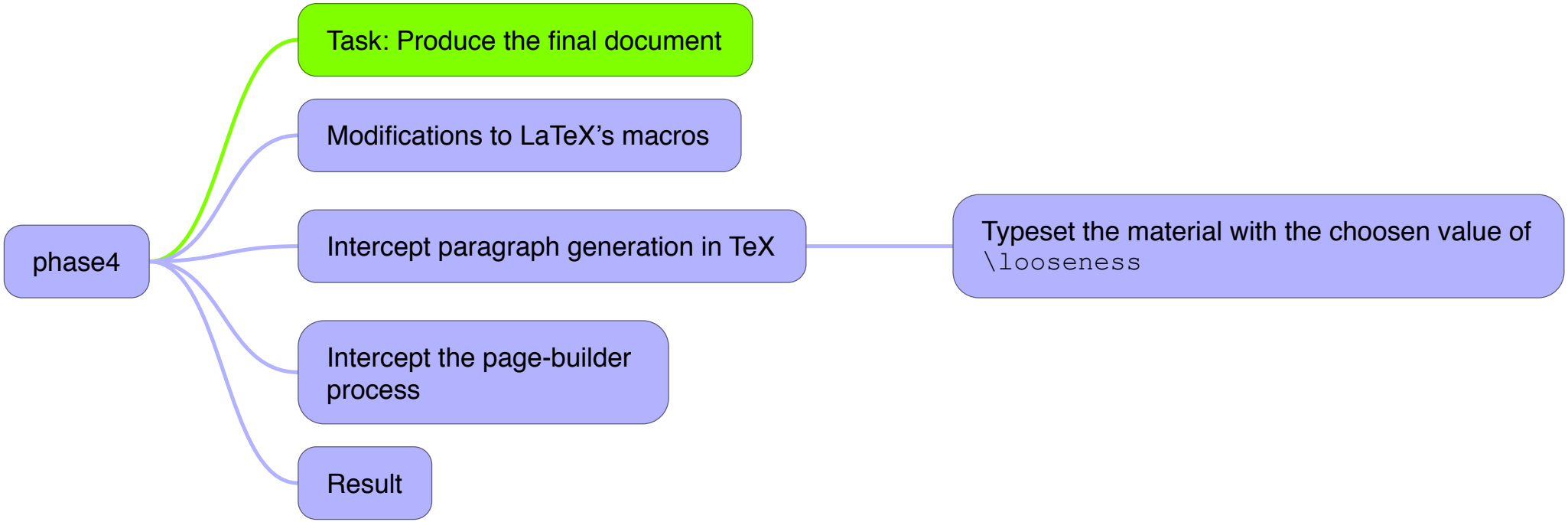

phase3
Task: Determine the optimal pagination
Pagination algorithm written in Lua
Features
Result
TXT
External Lua file representing the chosen pagination and the paths through the paragraph variations
TXT
Some more statistics

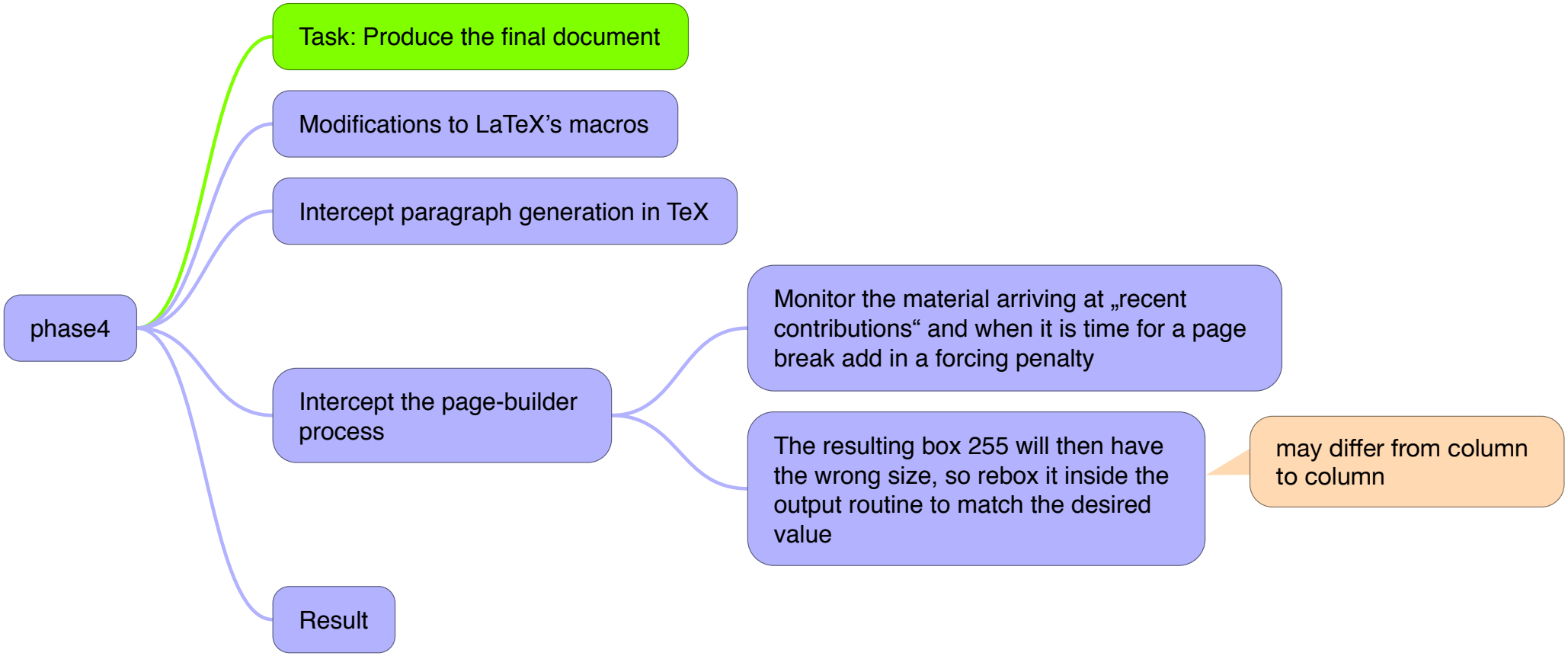
External Lua file representing the chosen pagination and the paths through the paragraph variations

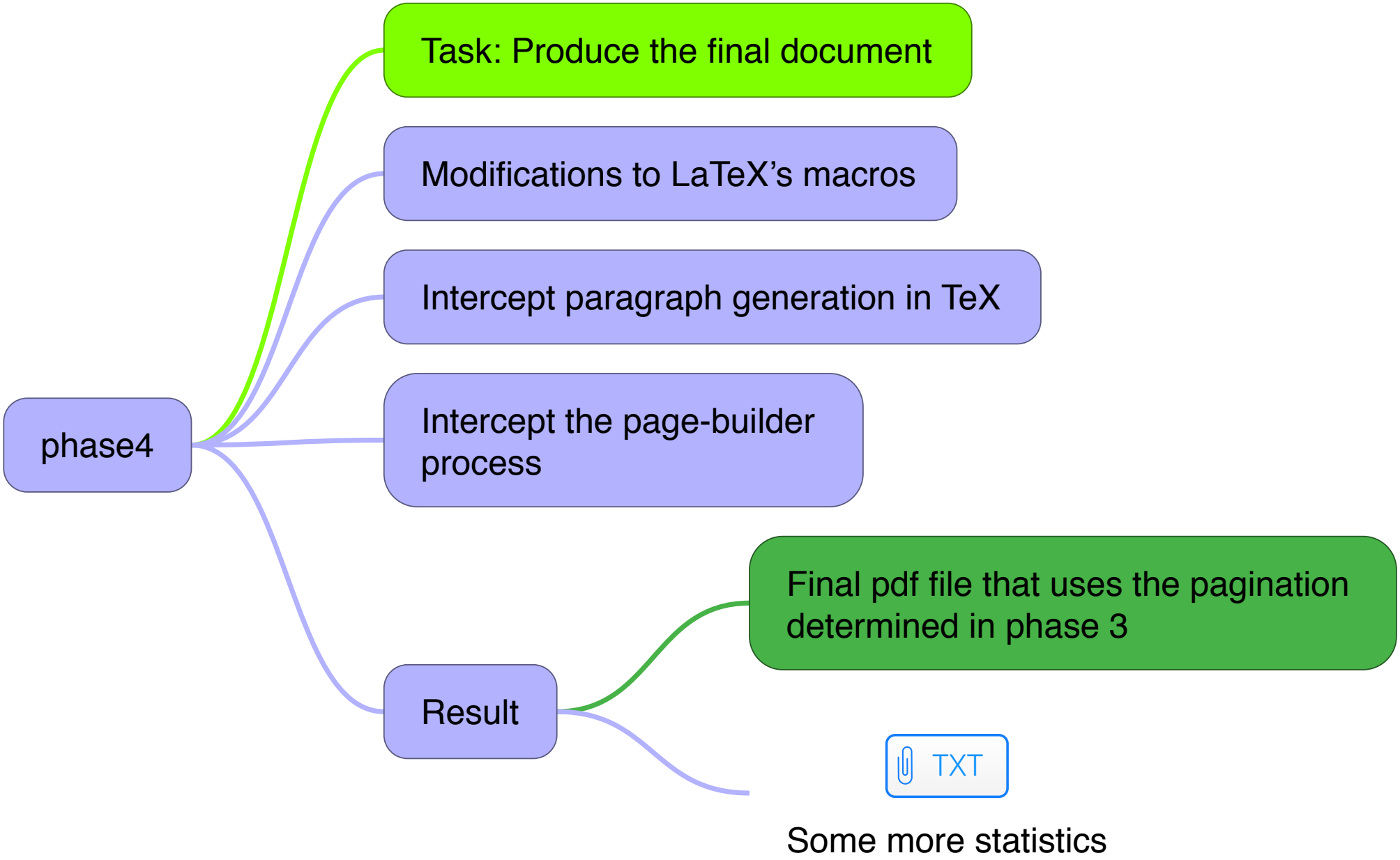
Some more statistics


phase4
Task: Produce the final document
Modifications to LaTeX's macros
Intercept paragraph generation in TeX
Intercept the page-builder process
Result


phase4
Task: Produce the final document
Modifications to LaTeX's macros
Apply same bug fixes/changes as for phase 2
Use a large \vsize so that TeX doesn't break on its own
Change \@makecol to support reboxing the output page to the right size
Intercept paragraph generation in TeX
Intercept the page-builder process
Result

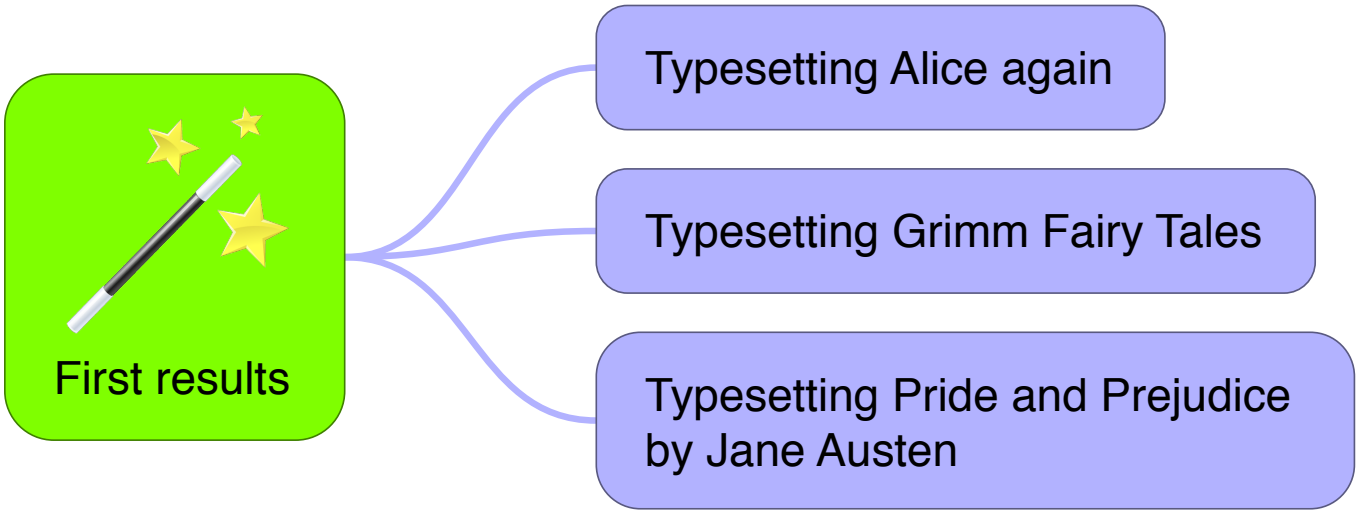

phase4
Task: Produce the final document
Modifications to LaTeX's macros
Intercept paragraph generation in TeX
Typeset the material with the choosen value of \looseness
Intercept the page-builder process
Result


phase4
Task: Produce the final document
Modifications to LaTeX's macros
Intercept paragraph generation in TeX
Intercept the page-builder process
Monitor the material arriving at „recent contributions“ and when it is time for a page break add in a forcing penalty
The resulting box 255 will then have the wrong size, so rebox it inside the output routine to match the desired value
may differ from column to column
Result


phase4
Task: Produce the final document
Modifications to LaTeX's macros
Intercept paragraph generation in TeX
Intercept the page-builder process
Result
Final pdf file that uses the pagination determined in phase 3
TXT
Some more statistics

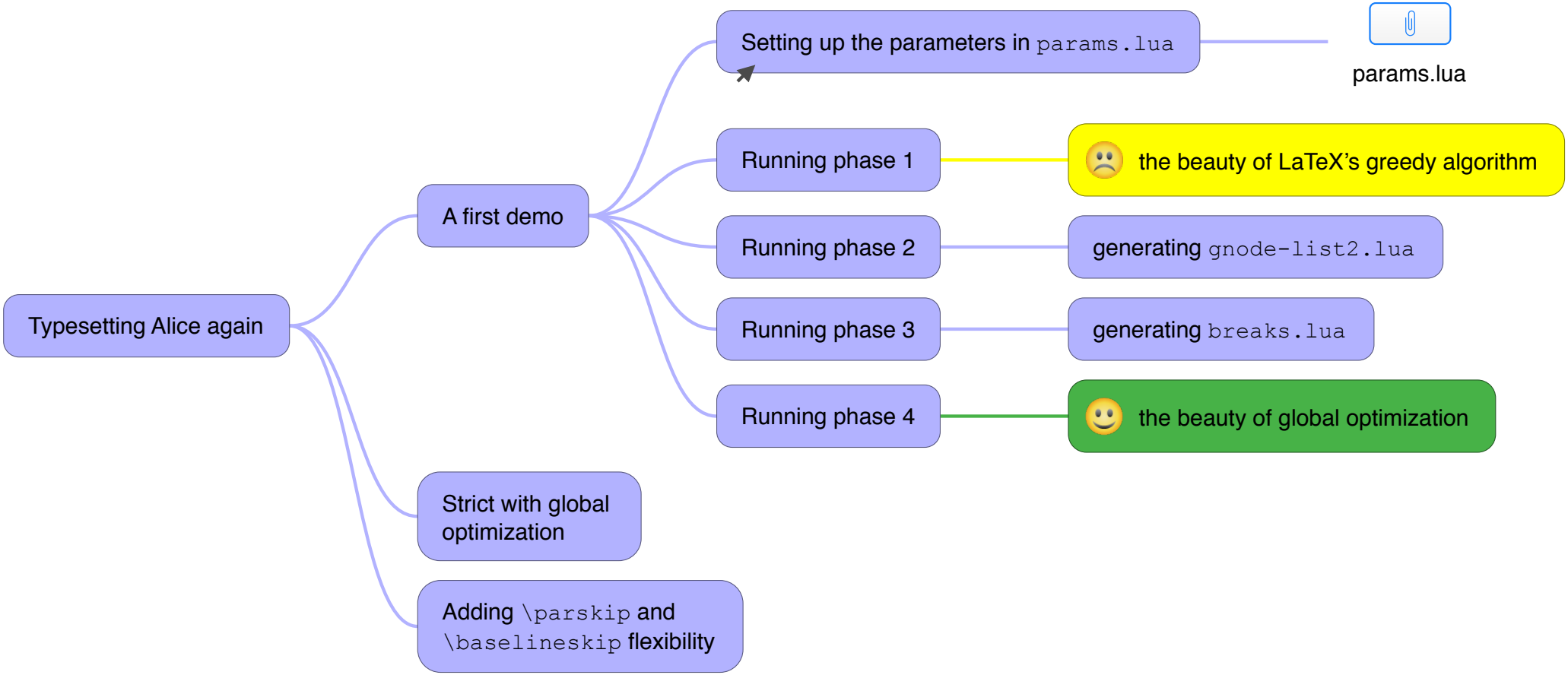
Some more statistics

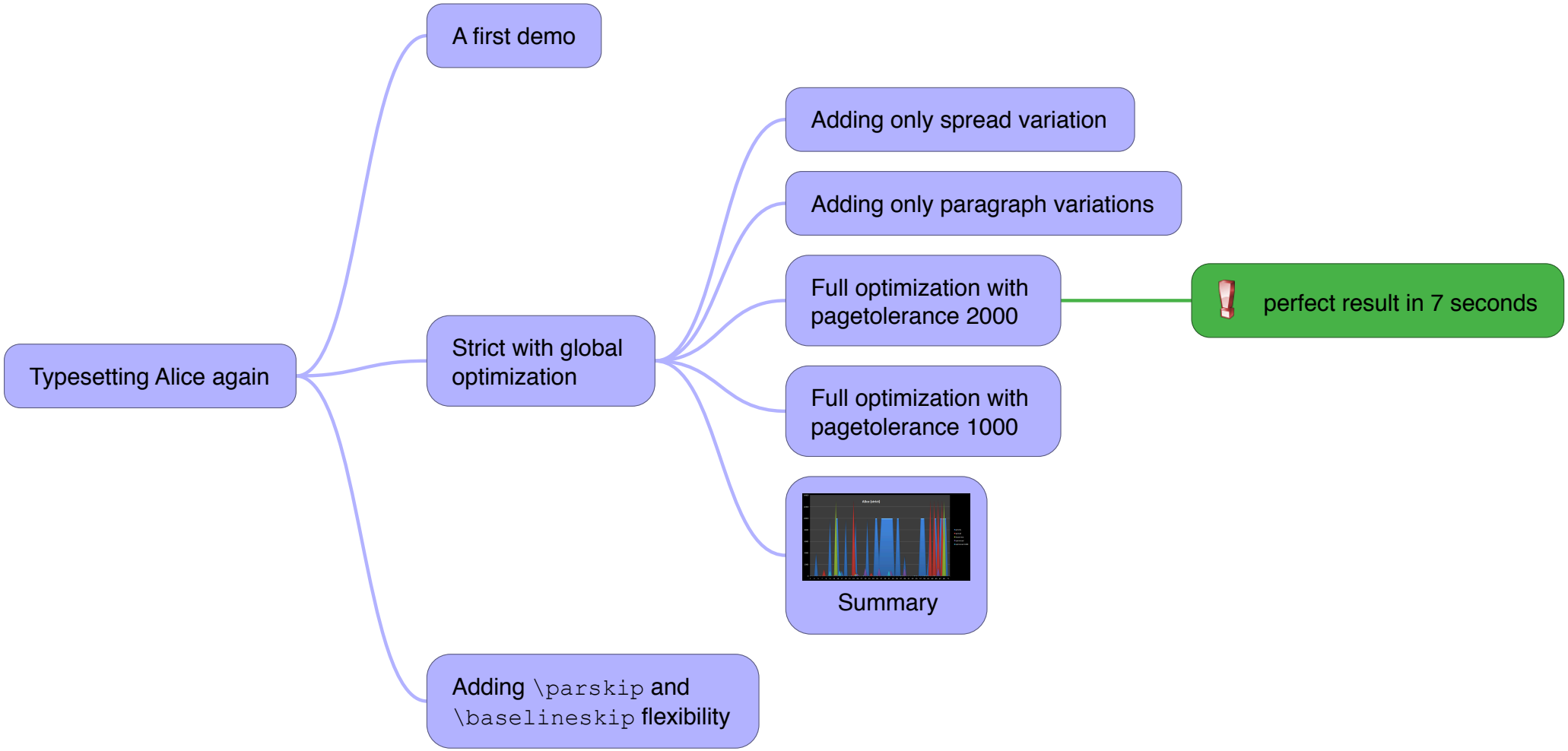

First results

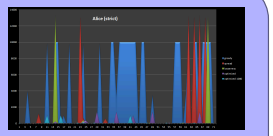

First results
Typesetting Alice again
Typesetting Grimm Fairy Tales
Typesetting Pride and Prejudice by Jane Austen

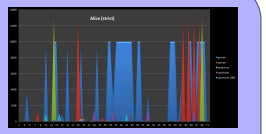
First results

- Typesetting Alice again
  - A first demo
  - Strict with global optimization
  - Adding `\parskip` and `\baselineskip` flexibility
- Typesetting Grimm Fairy Tales
- Typesetting Pride and Prejudice by Jane Austen

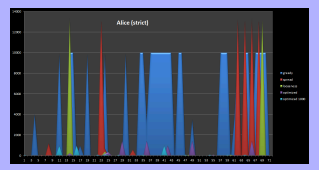

Typesetting Alice again
A first demo
Setting up the parameters in params.lua
params.lua
Running phase 1
the beauty of LaTeX's greedy algorithm
Running phase 2
generating gnode-list2.lua
Running phase 3
generating breaks.lua
Running phase 4
the beauty of global optimization
Strict with global optimization
Adding \parskip and \baselineskip flexibility


Typesetting Alice again
A first demo
Strict with global optimization
Adding only spread variation
Adding only paragraph variations
Full optimization with pagetolerance 2000
perfect result in 7 seconds
Full optimization with pagetolerance 1000
Summary
Adding \parskip and \baselineskip flexibility

- Strict with global optimization
  - Adding only spread variation
    - 4,5 seconds
    - ❗ 5 overfull columns (pages 12, 31, 32, 33, 34)
    - 🙂 Possible manual fix: shorten the paragraph length with `\rlap` for p. 12 and force a widow on p. 31
  - Adding only paragraph variations
  - Full optimization with pagetolerance 2000
  - Full optimization with pagetolerance 1000
  - 
    Summary

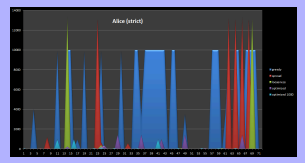
- Strict with global optimization
  - Adding only spread variation
  - Adding only paragraph variations
    - 5,1 seconds
    - ❗ 2 overfull columns (pages 7 and 35)
    - ☹️ Possible manual fix: not easy as p. 7 is really bad
  - Full optimization with pagetolerance 2000
  - Full optimization with pagetolerance 1000
  - 
    Summary

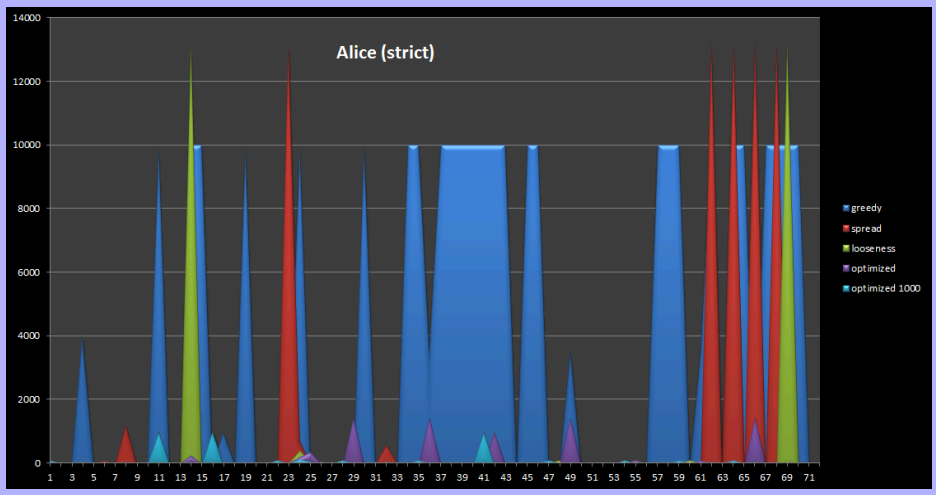
Strict with global optimization

- Adding only spread variation
- Adding only paragraph variations
- Full optimization with pagetolerance 2000
  - perfect result in 7 seconds
- Full optimization with pagetolerance 1000
- 
  Summary

Strict with global optimization

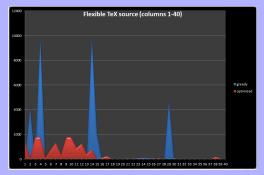
- Adding only spread variation
- Adding only paragraph variations
- Full optimization with pagetolerance 2000
- Full optimization with pagetolerance 1000
  - perfect result in 7 seconds
  - one page (one column) shorter
- Summary



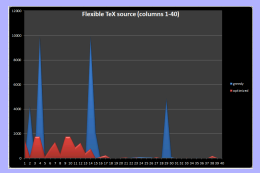

Alice (strict)
14000
12000
10000
8000
6000
4000
2000
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71
greedy
spread
looseness
optimized
optimized 1000

Summary

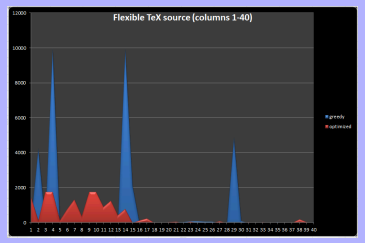
Adding `\parskip` and `\baselineskip` flexibility

- Not really a great idea, as varying `\baselineskip` changes the grey value of the paragraph
  -  It is also an invalid approach if you intend to do grid-based typesetting
- Why does it help?
- Results
- 
  Summary

Adding `\parskip` and `\baselineskip` flexibility

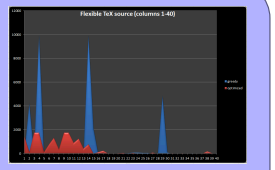
- Not really a great idea, as varying `\baselineskip` changes the grey value of the paragraph
- Why does it help?
  - The normal flexibility of a page increases from 0 to 5,25pt (assuming 3 paragraphs and 46 lines of text)
  - So with a badness of < 800 we already gain a line per column through white space stretching
  - If there are more short paragraphs than gaining two lines within pagetolerance limits is possible
- Results
- 
  Summary

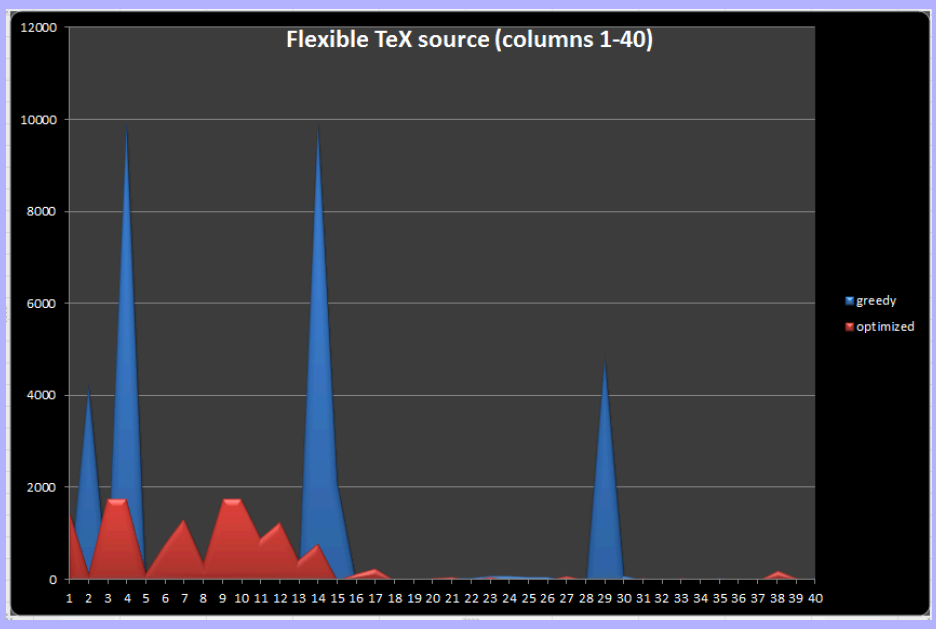
Adding `\parskip` and `\baselineskip` flexibility

- Not really a great idea, as varying `\baselineskip` changes the grey value of the paragraph
- Why does it help?
- Results
  - with standard TeX
  - when optimizing
- 

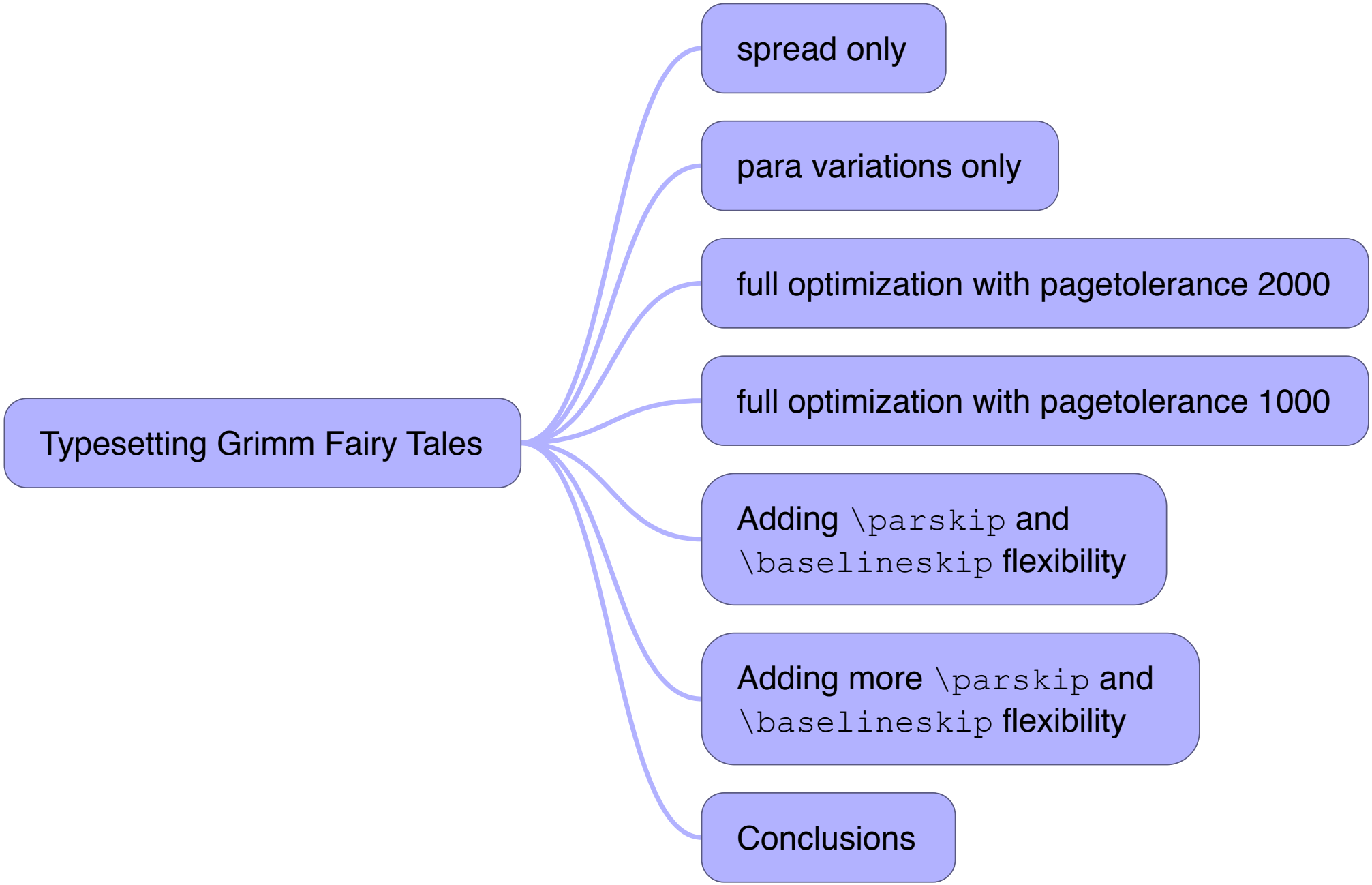
  Summary

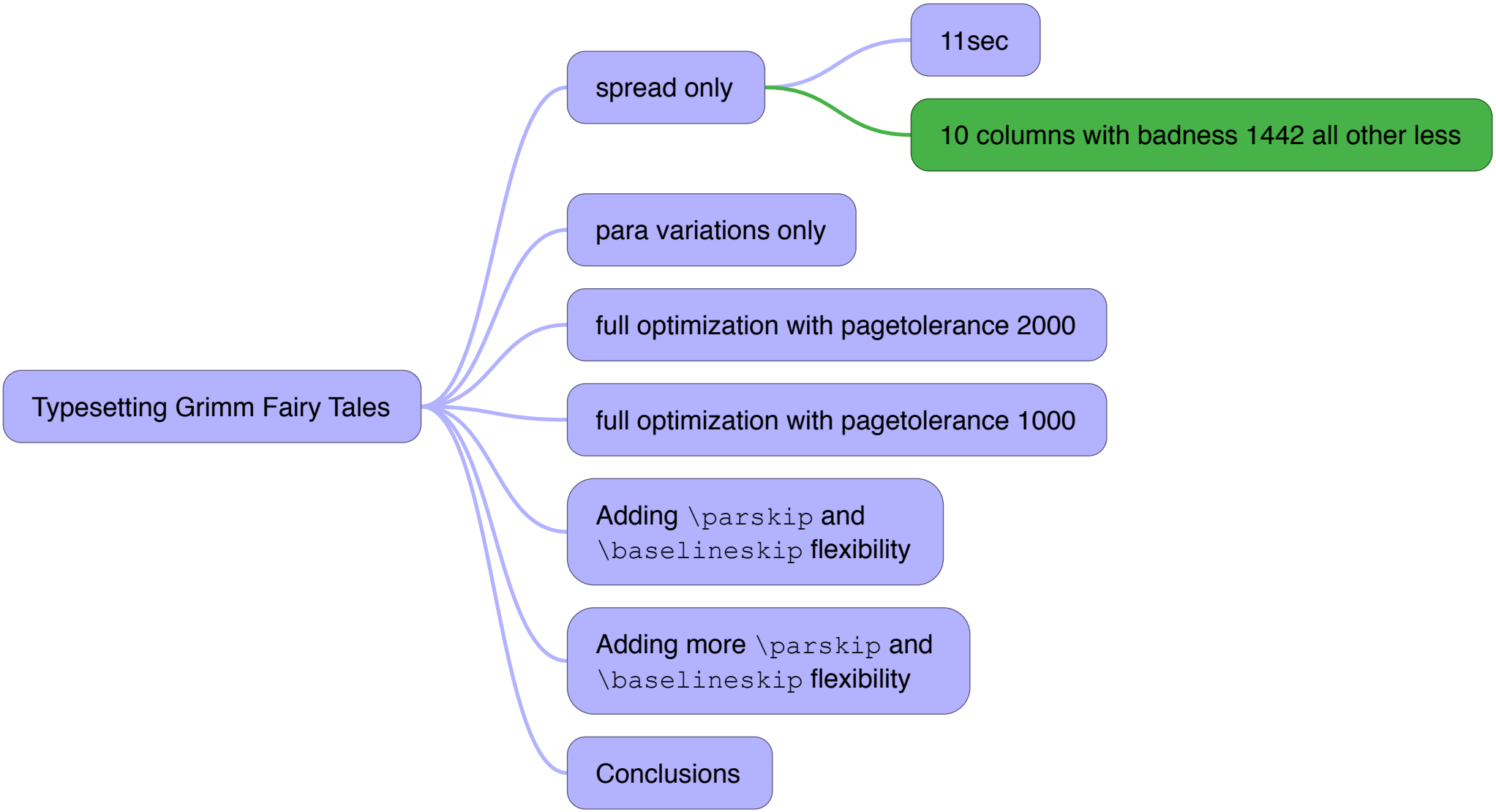
Adding `\parskip` and `\baselineskip` flexibility

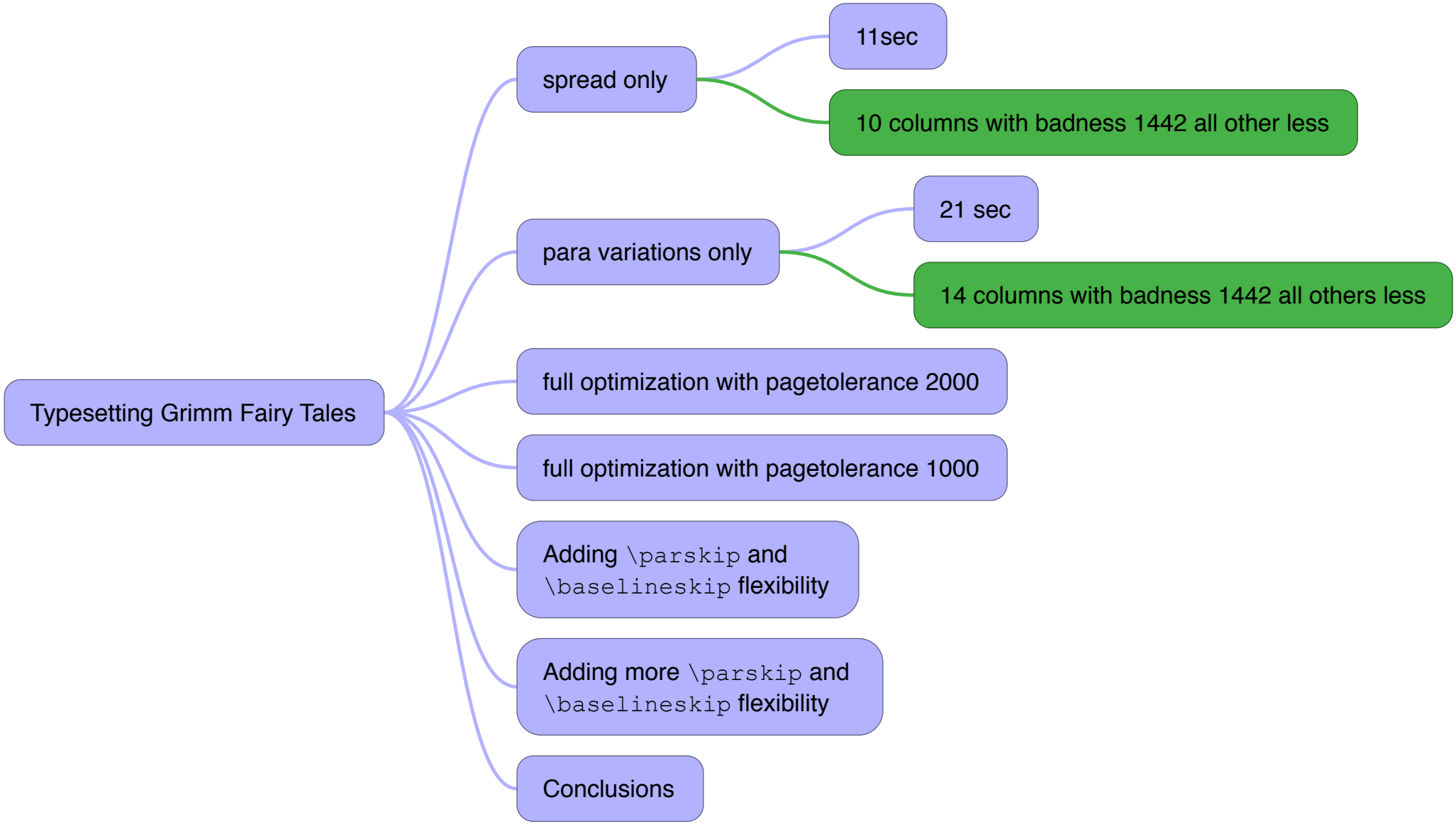
- Not really a great idea, as varying `\baselineskip` changes the grey value of the paragraph
- Why does it help?
- Results
  - with standard TeX
    - much better already but still big problems
  - when optimizing
    - good result in 5 seconds
      - let's run it quickly
- 
  Summary

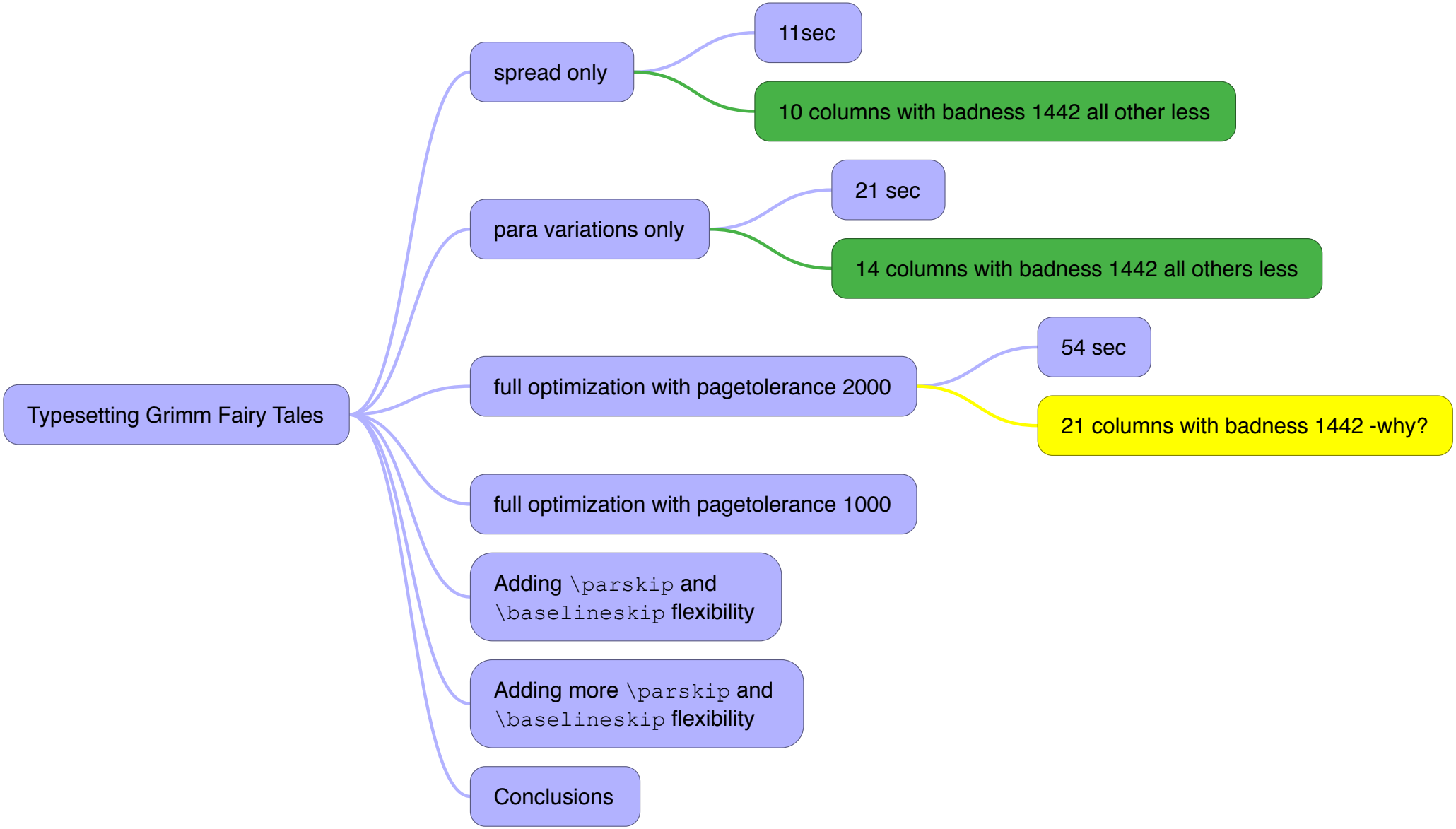

Flexible TeX source (columns 1-40)
12000
10000
8000
6000
4000
2000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
greedy
optimized

Summary

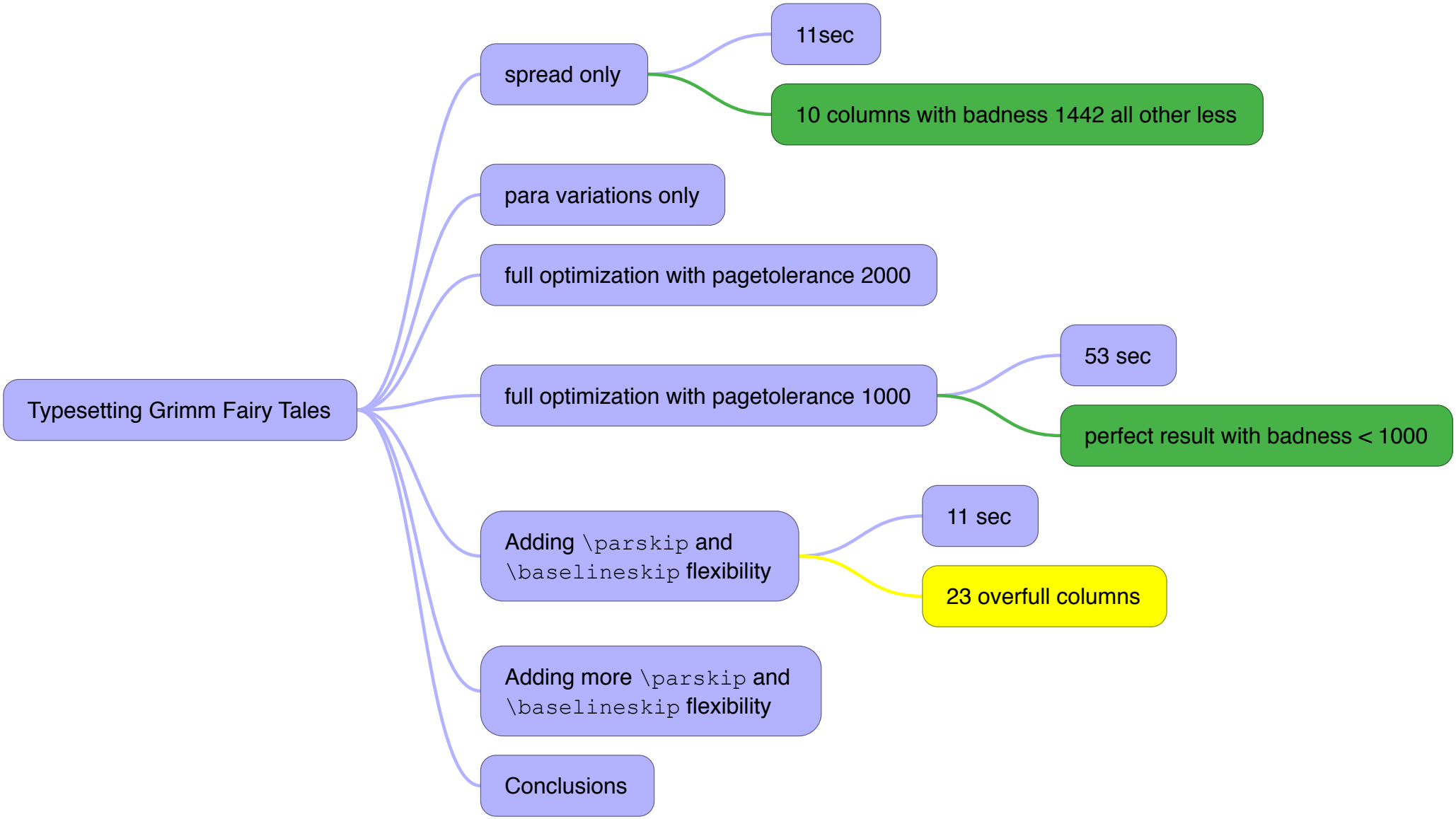

Typesetting Grimm Fairy Tales
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions


Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions


Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
21 sec
14 columns with badness 1442 all others less
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

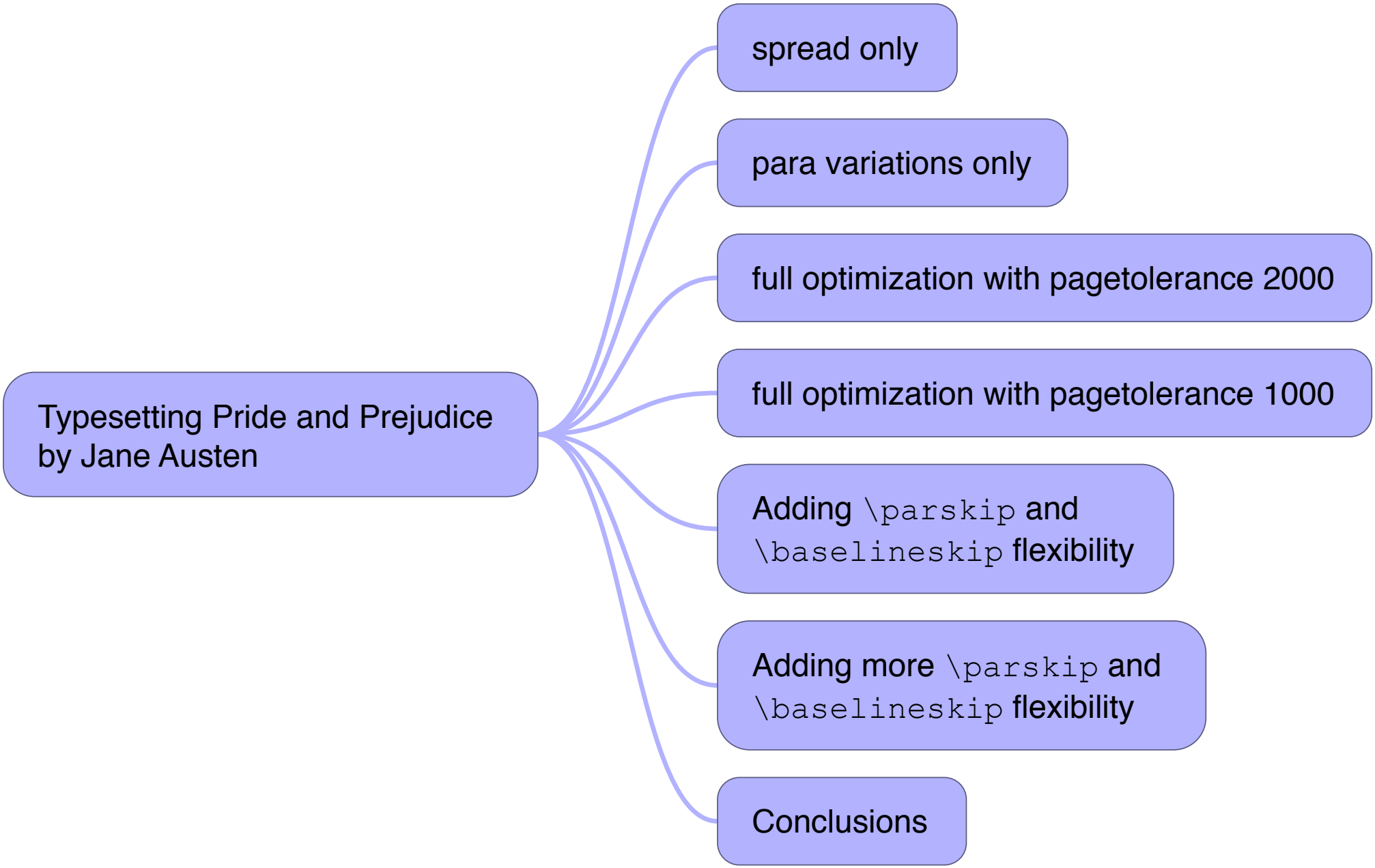

Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
21 sec
14 columns with badness 1442 all others less
full optimization with pagetolerance 2000
54 sec
21 columns with badness 1442 -why?
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

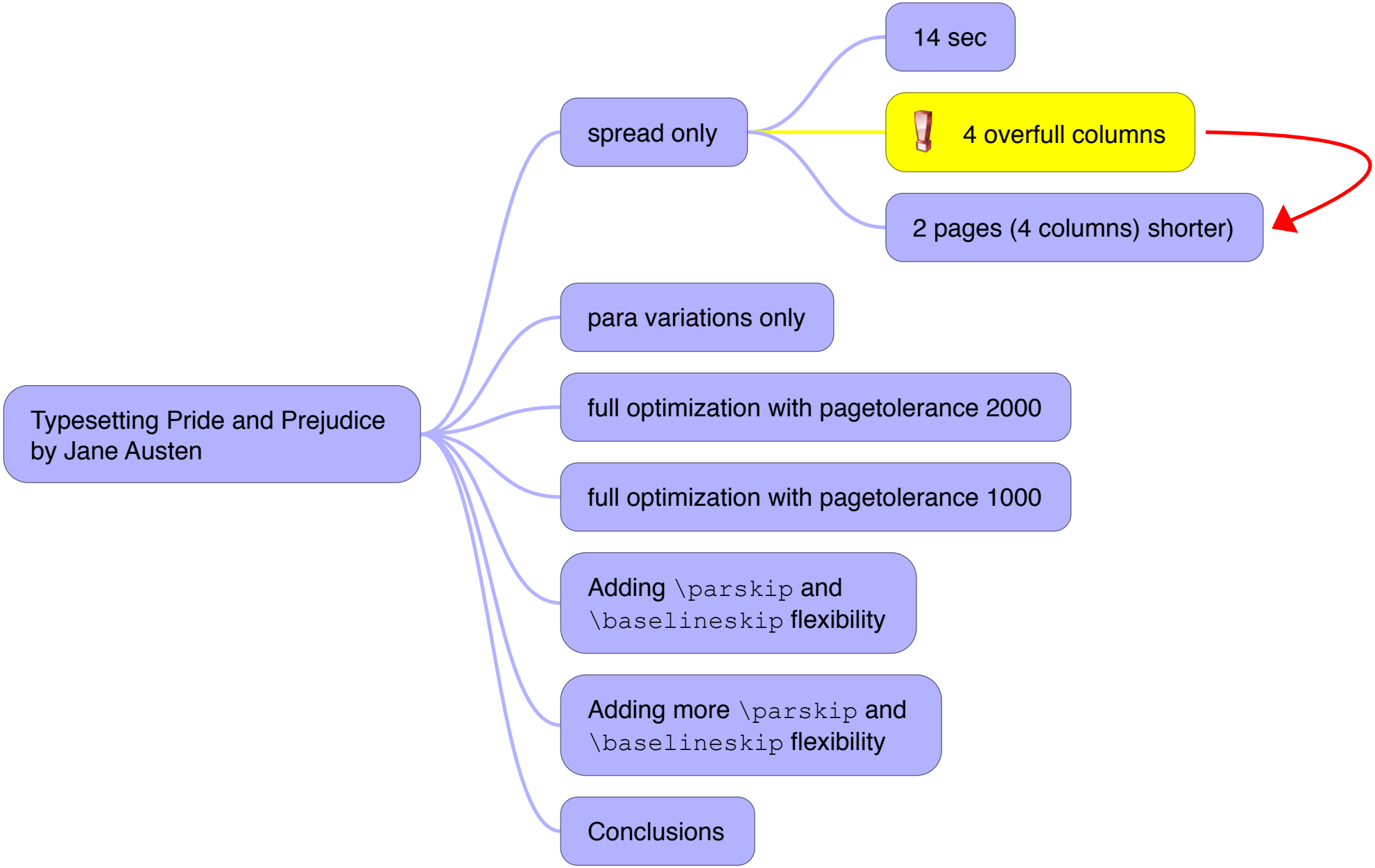

Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
21 sec
14 columns with badness 1442 all others less
full optimization with pagetolerance 2000
54 sec
21 columns with badness 1442 -why?
full optimization with pagetolerance 1000
53 sec
perfect result with badness < 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

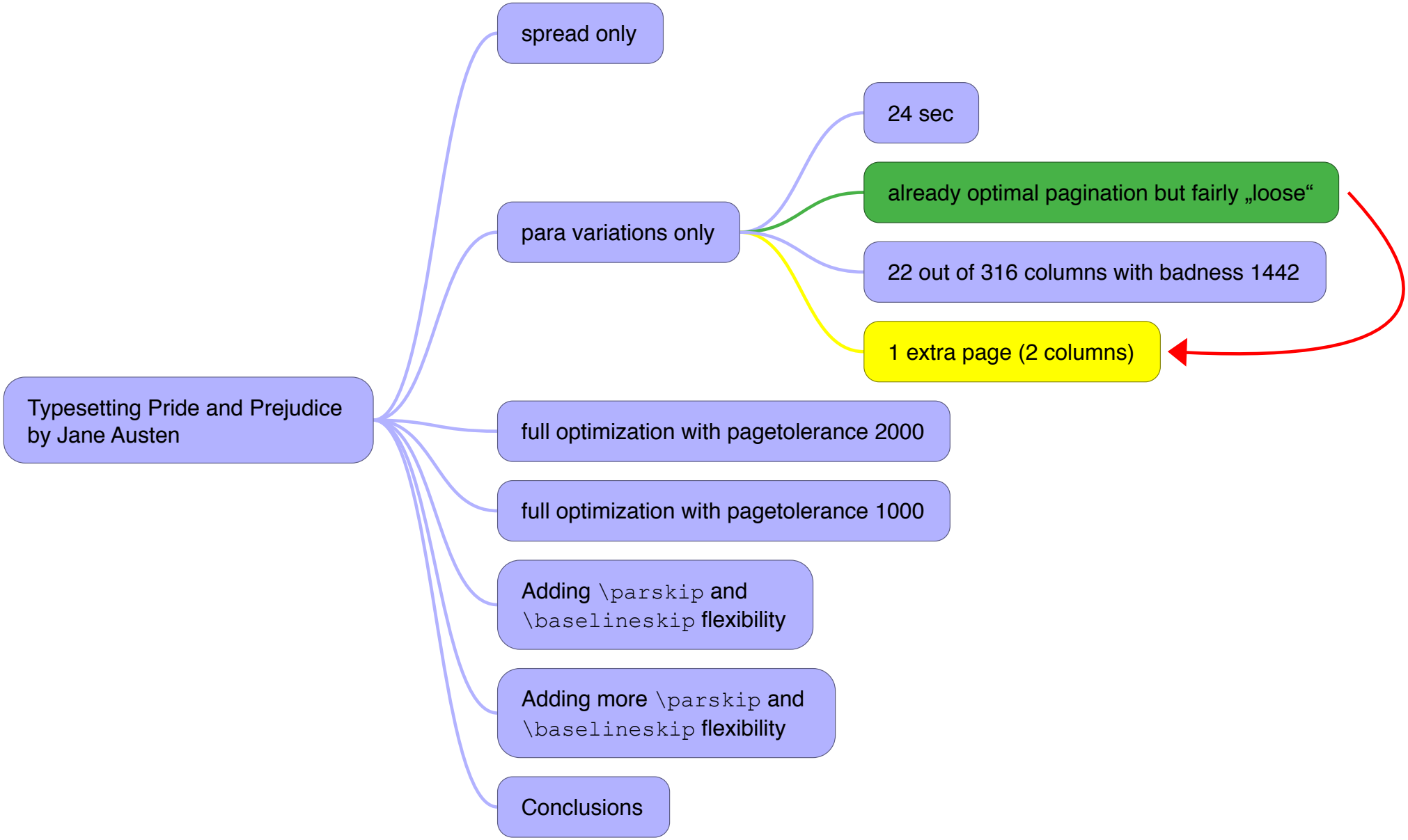

Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
53 sec
perfect result with badness < 1000
Adding \parskip and \baselineskip flexibility
11 sec
23 overfull columns
Adding more \parskip and \baselineskip flexibility
Conclusions

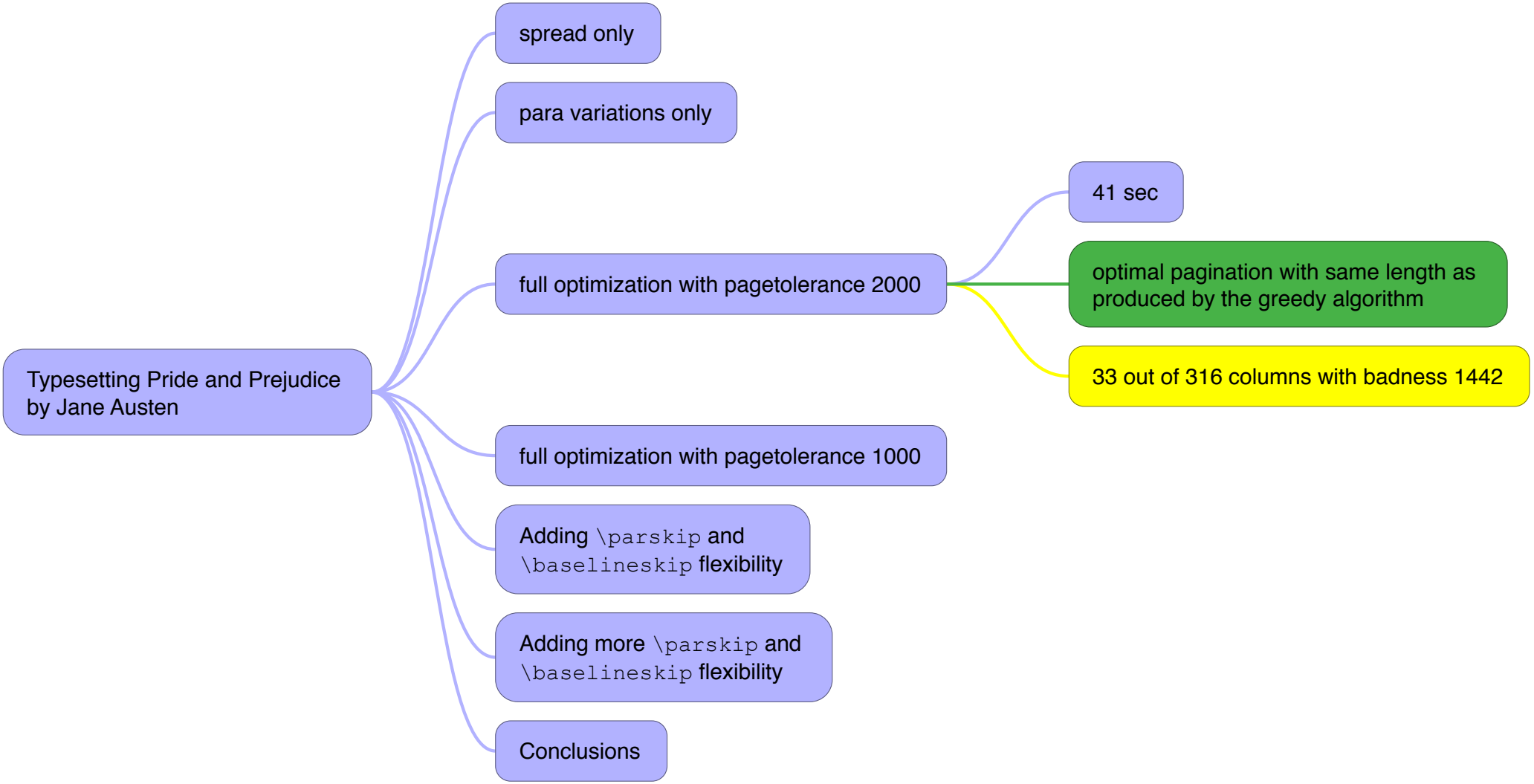

Typesetting Grimm Fairy Tales
spread only
11sec
10 columns with badness 1442 all other less
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
53 sec
perfect result with badness < 1000
Adding \parskip and \baselineskip flexibility
11 sec
23 overfull columns
Adding more \parskip and \baselineskip flexibility
12 sec
optimized result, but somewhat noticeable stretch
Conclusions

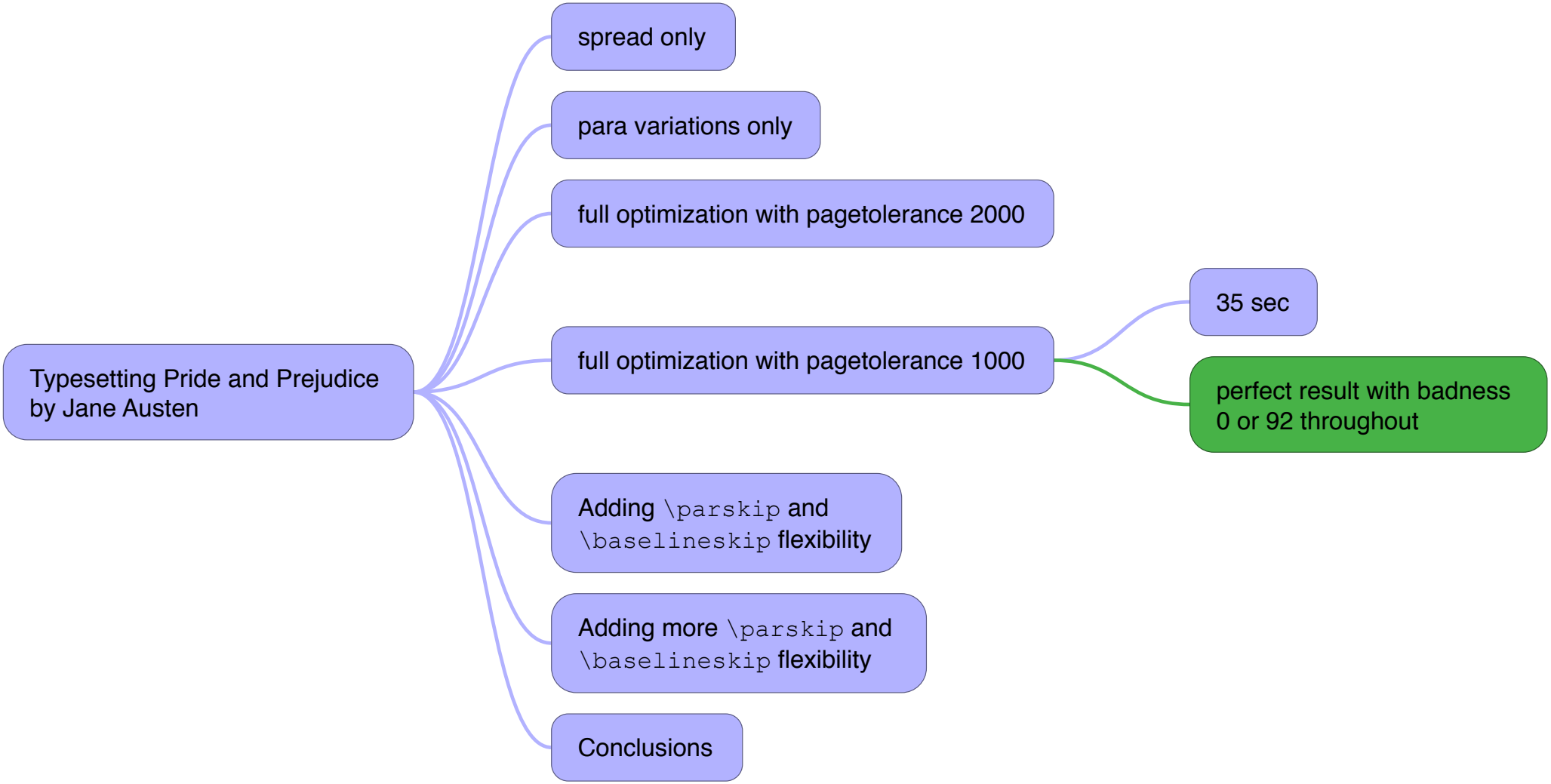

Typesetting Grimm Fairy Tales
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions
Due to the fact that the Fairy Tales contain only headings as mediums sized unbreakable blocks, just a little bit of extra flexibility is required to allow the global optimization to succeed
However, because of the very long paragraphs it is not enough to increase the flexibility on the TeX level (via \parskip and \baselineskip) unless we move to already very visible values

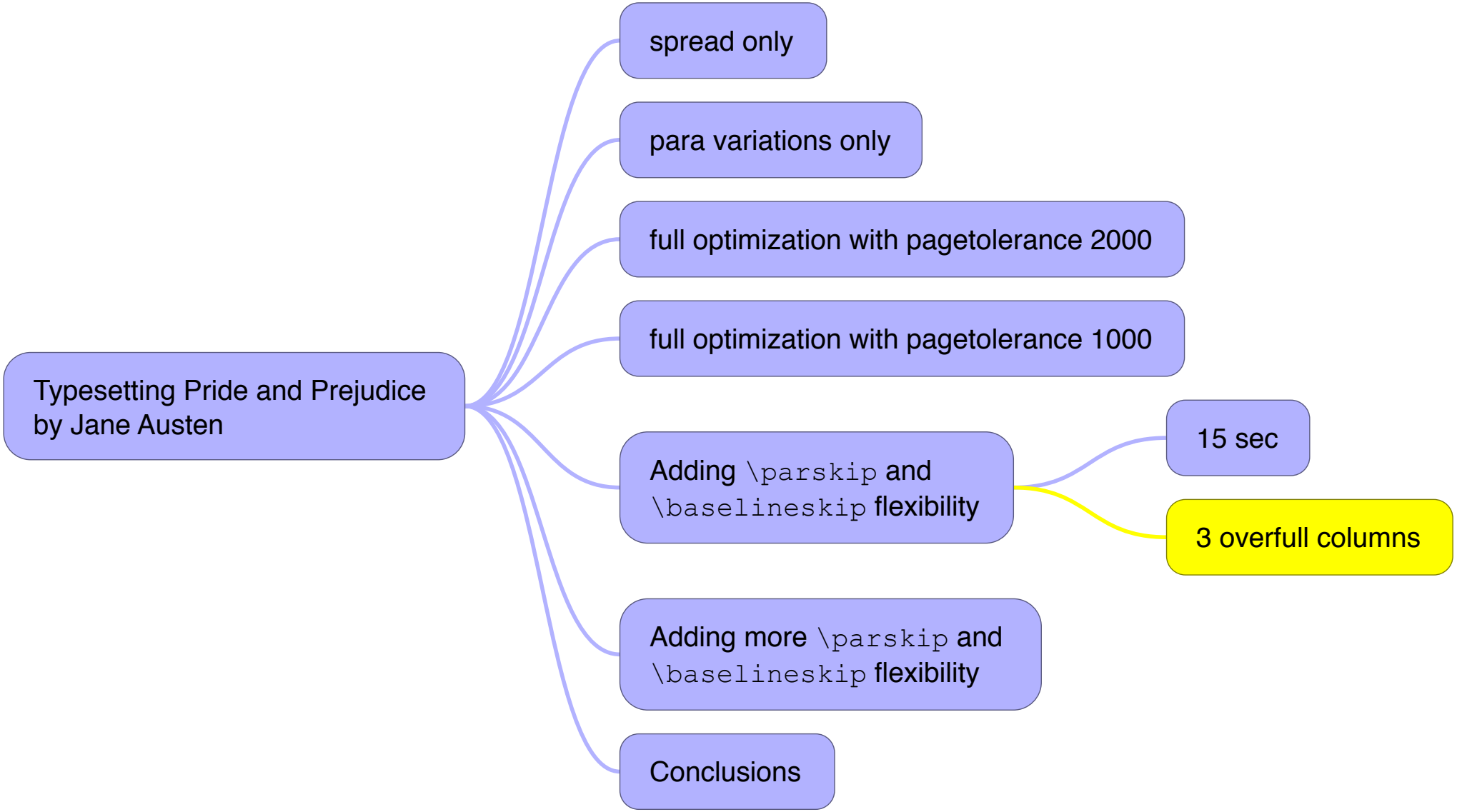

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

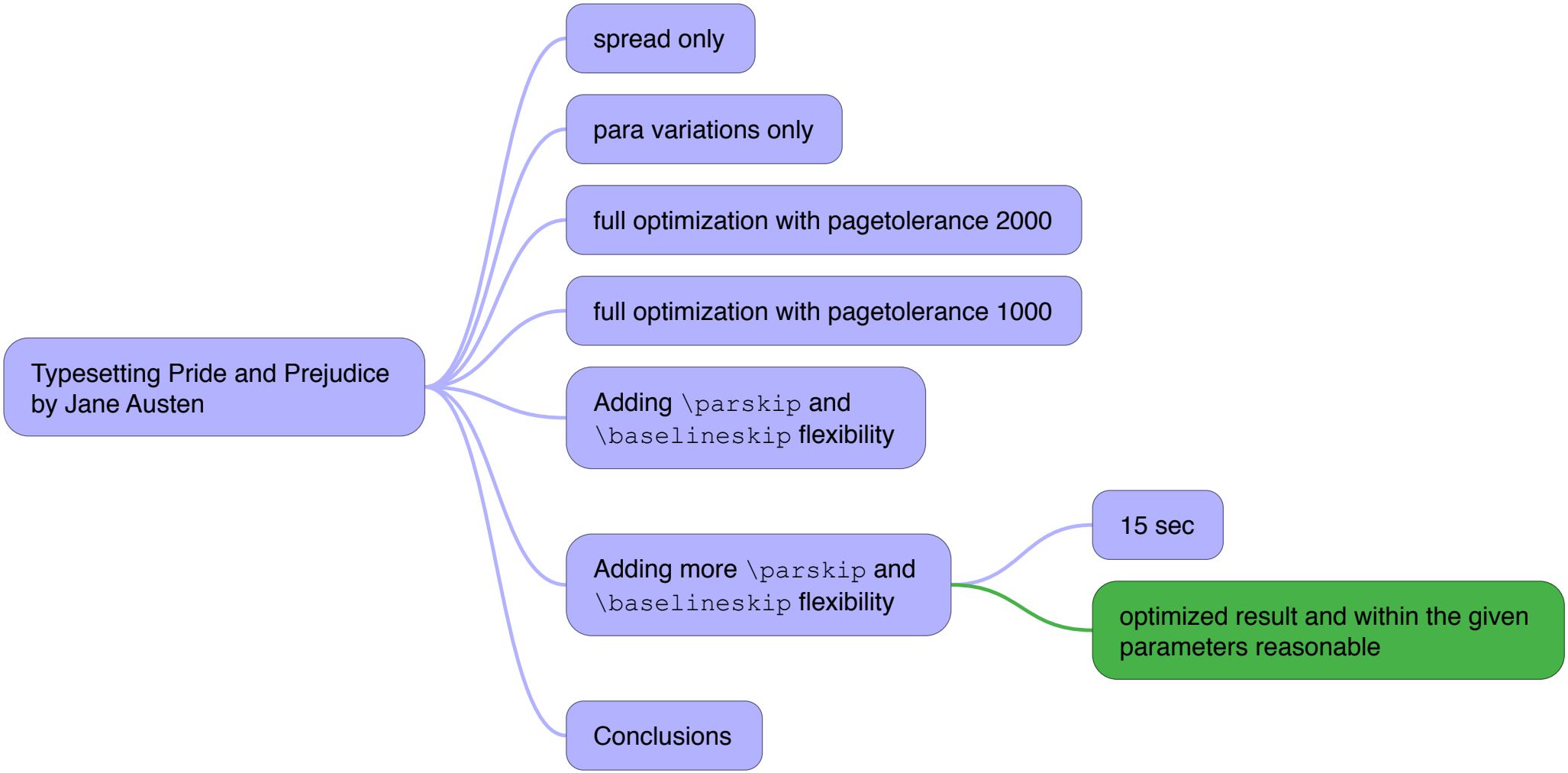

Typesetting Pride and Prejudice by Jane Austen
spread only
14 sec
4 overfull columns
2 pages (4 columns) shorter)
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

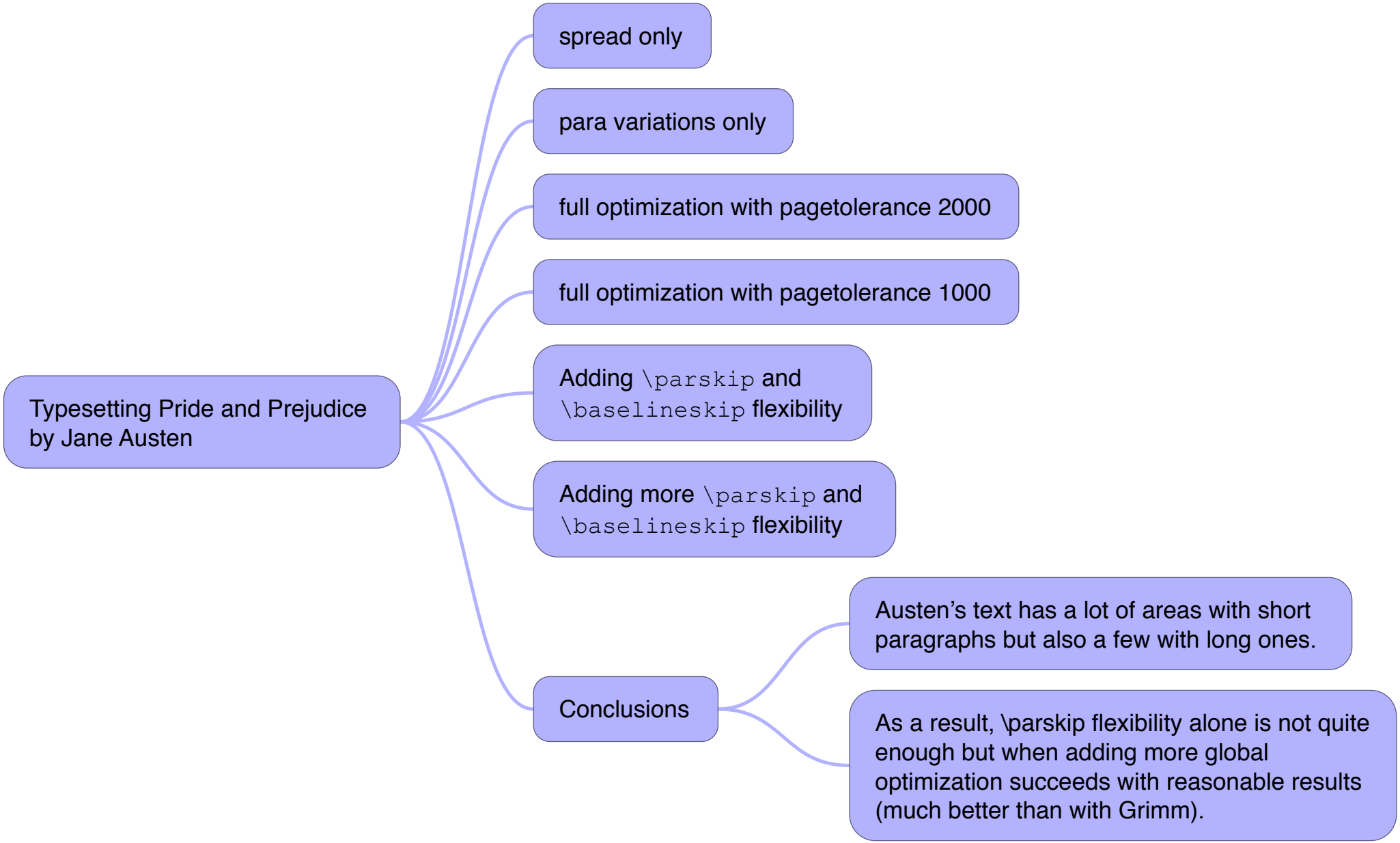

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
24 sec
already optimal pagination but fairly „loose“
22 out of 316 columns with badness 1442
1 extra page (2 columns)
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions


Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
41 sec
optimal pagination with same length as produced by the greedy algorithm
33 out of 316 columns with badness 1442
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

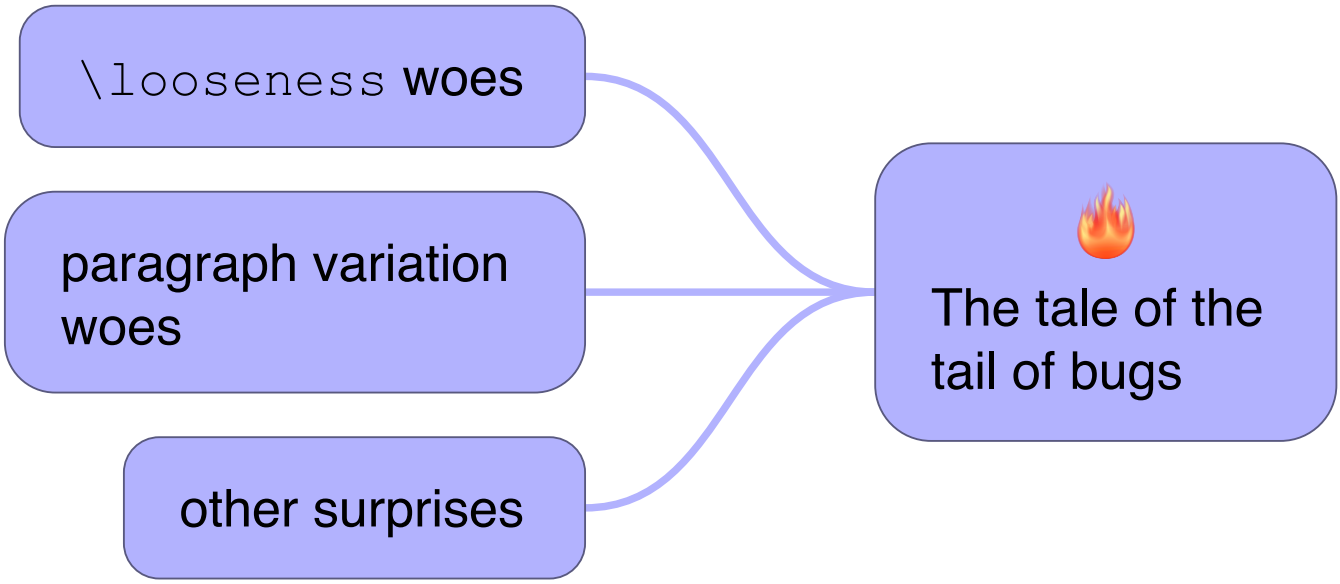

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
35 sec
perfect result with badness 0 or 92 throughout
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions

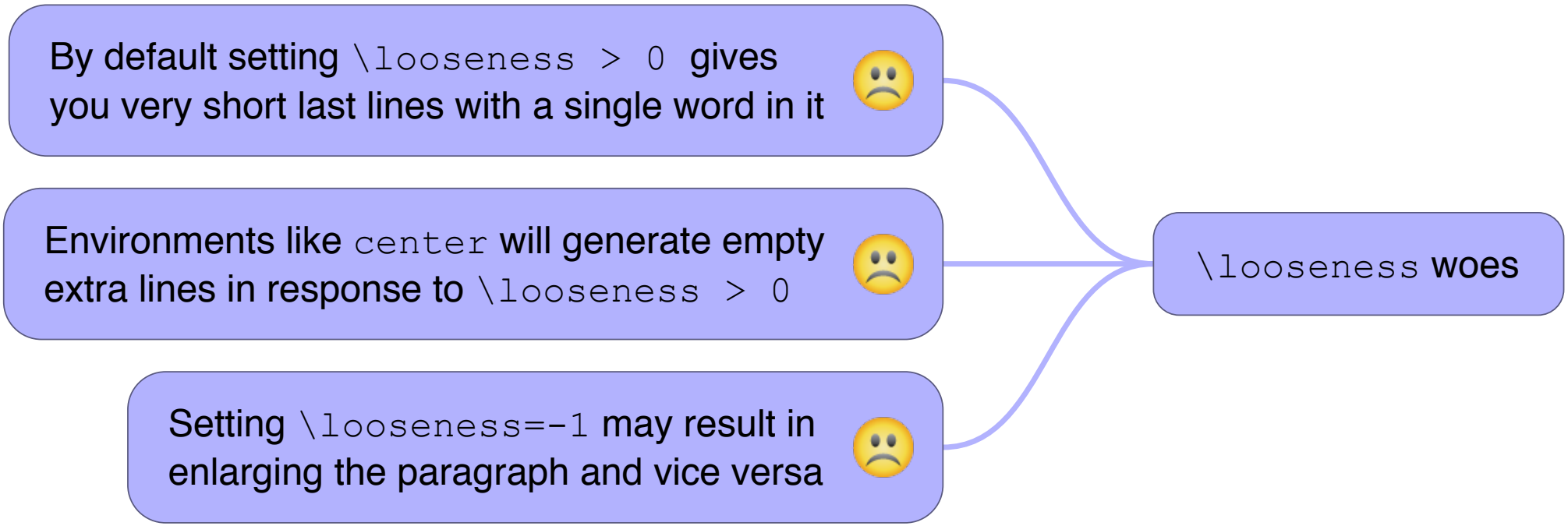

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
15 sec
3 overfull columns
Adding more \parskip and \baselineskip flexibility
Conclusions

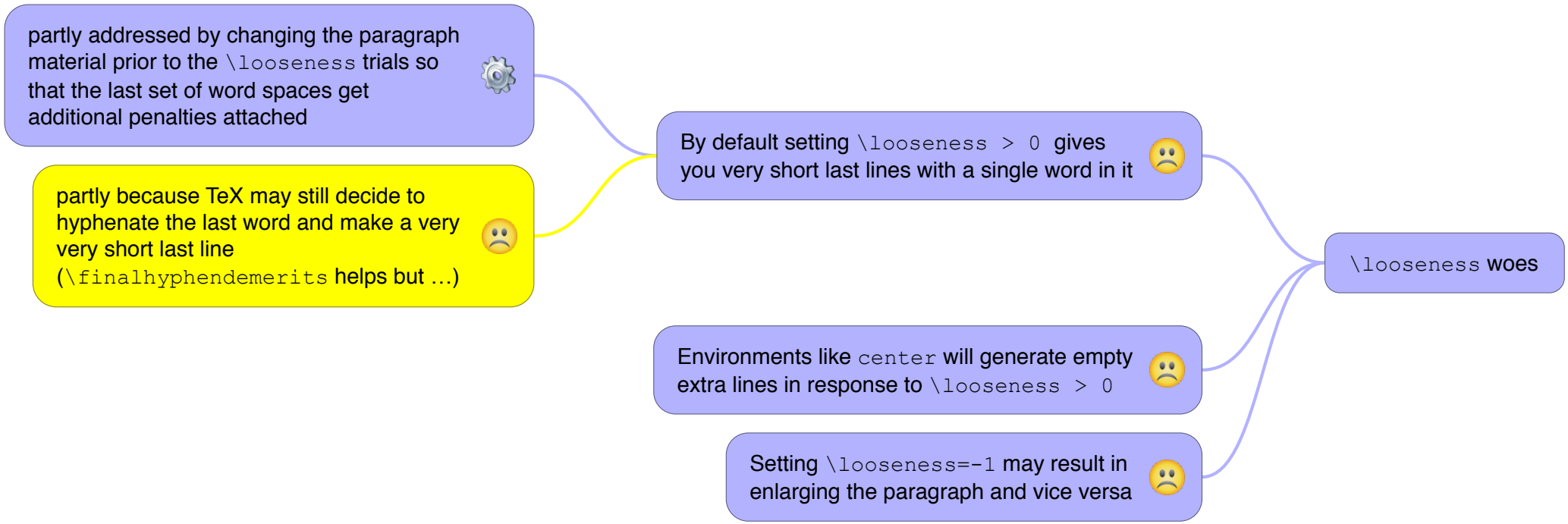

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
15 sec
optimized result and within the given parameters reasonable
Conclusions

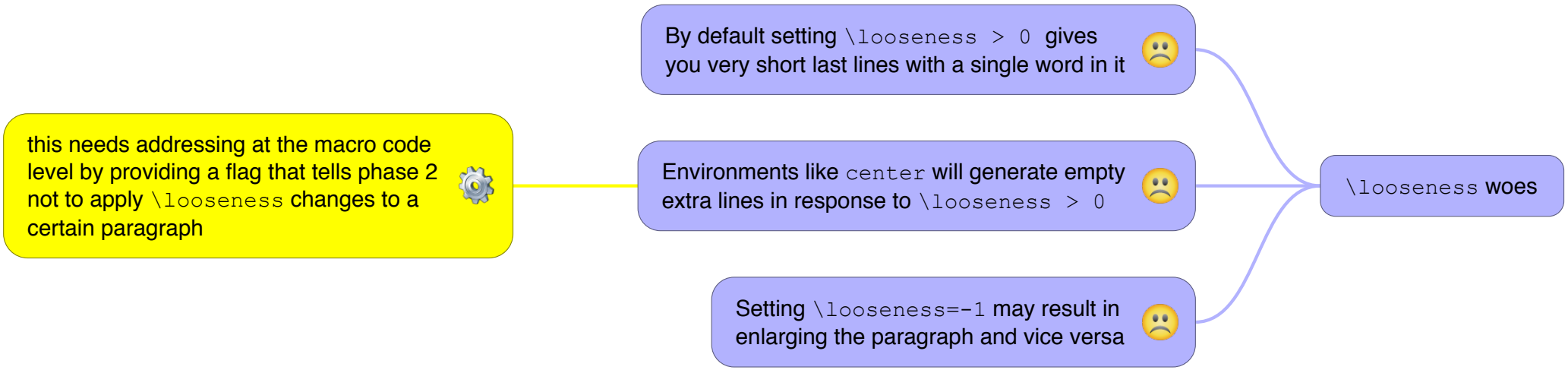

Typesetting Pride and Prejudice by Jane Austen
spread only
para variations only
full optimization with pagetolerance 2000
full optimization with pagetolerance 1000
Adding \parskip and \baselineskip flexibility
Adding more \parskip and \baselineskip flexibility
Conclusions
Austen's text has a lot of areas with short paragraphs but also a few with long ones.
As a result, \parskip flexibility alone is not quite enough but when adding more global optimization succeeds with reasonable results (much better than with Grimm).

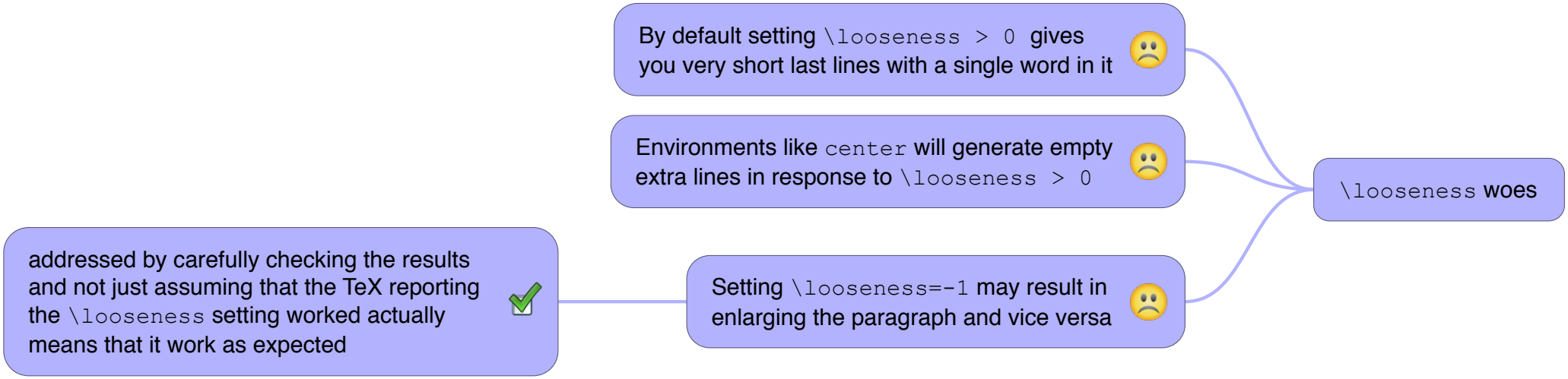
The tale of the tail of bugs

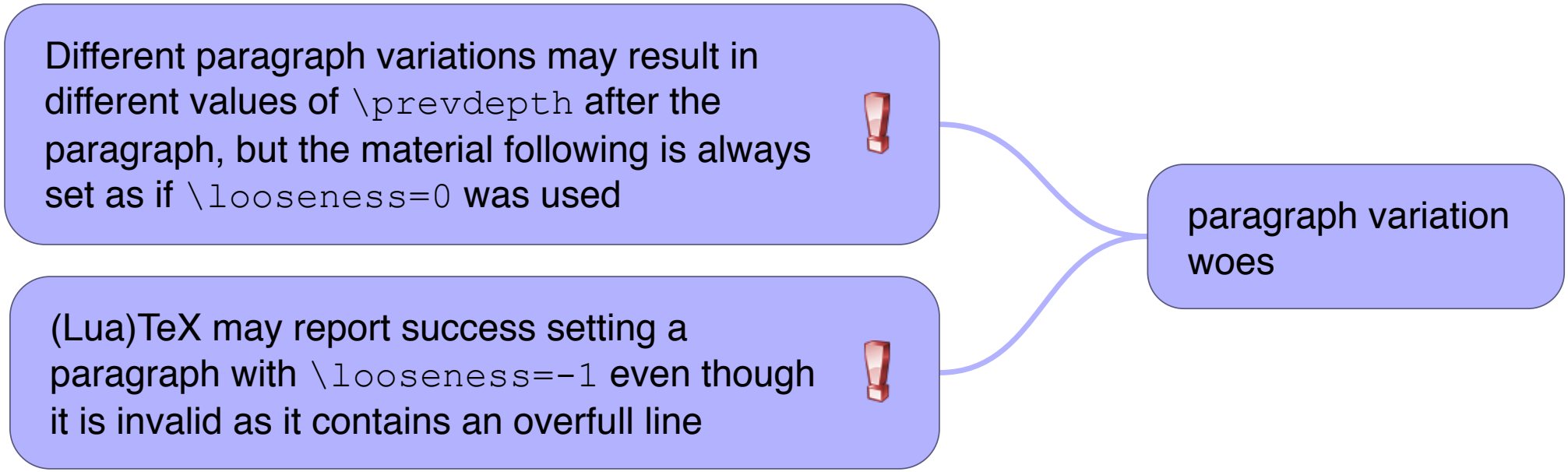

\looseness woes
paragraph variation woes
other surprises
The tale of the tail of bugs

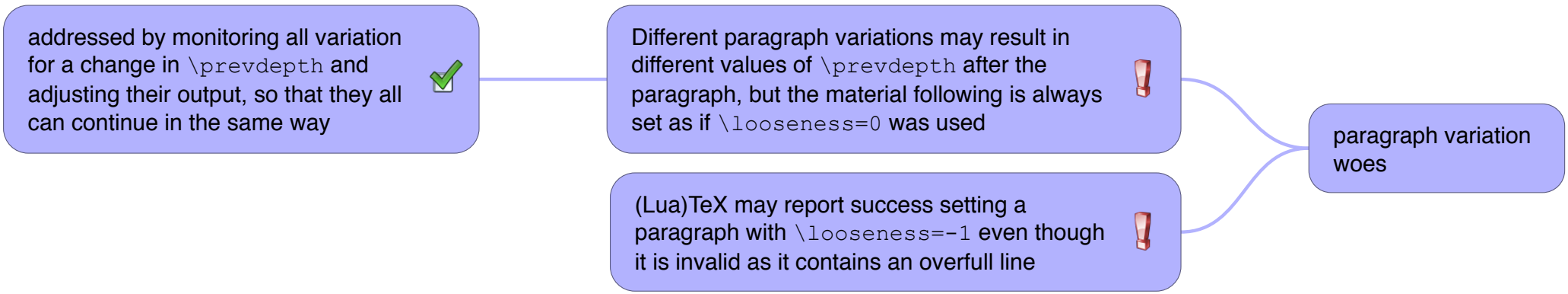

By default setting \looseness > 0 gives you very short last lines with a single word in it
Environments like center will generate empty extra lines in response to \looseness > 0
Setting \looseness=-1 may result in enlarging the paragraph and vice versa
\looseness woes

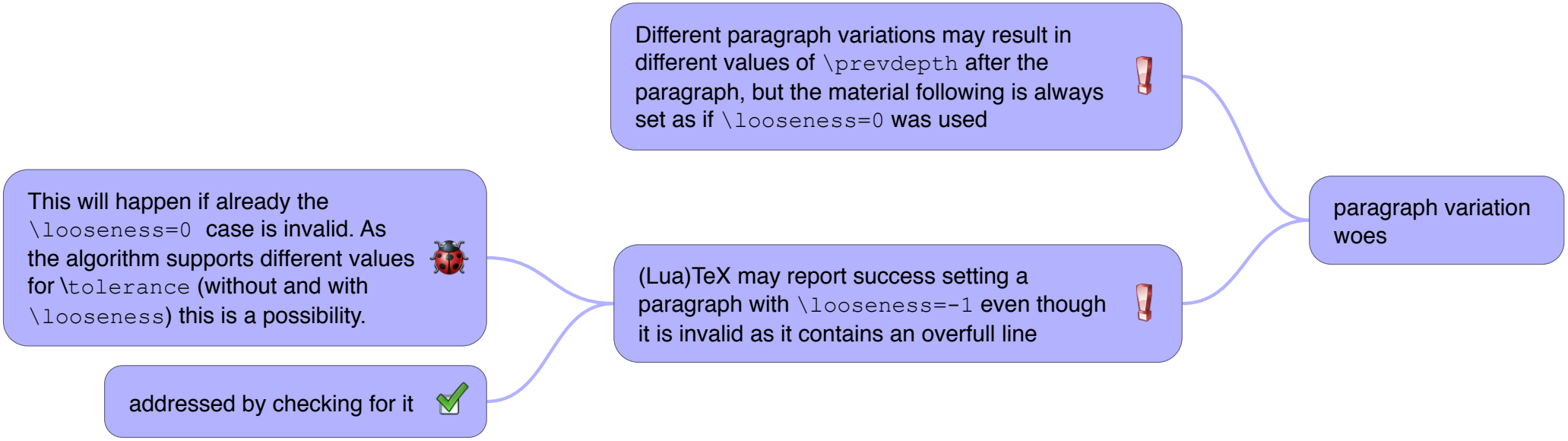

\looseness woes
By default setting \looseness > 0 gives you very short last lines with a single word in it
partly addressed by changing the paragraph material prior to the \looseness trials so that the last set of word spaces get additional penalties attached
partly because TeX may still decide to hyphenate the last word and make a very very short last line (\finalhyphendemerits helps but …)
Environments like center will generate empty extra lines in response to \looseness > 0
Setting \looseness=-1 may result in enlarging the paragraph and vice versa

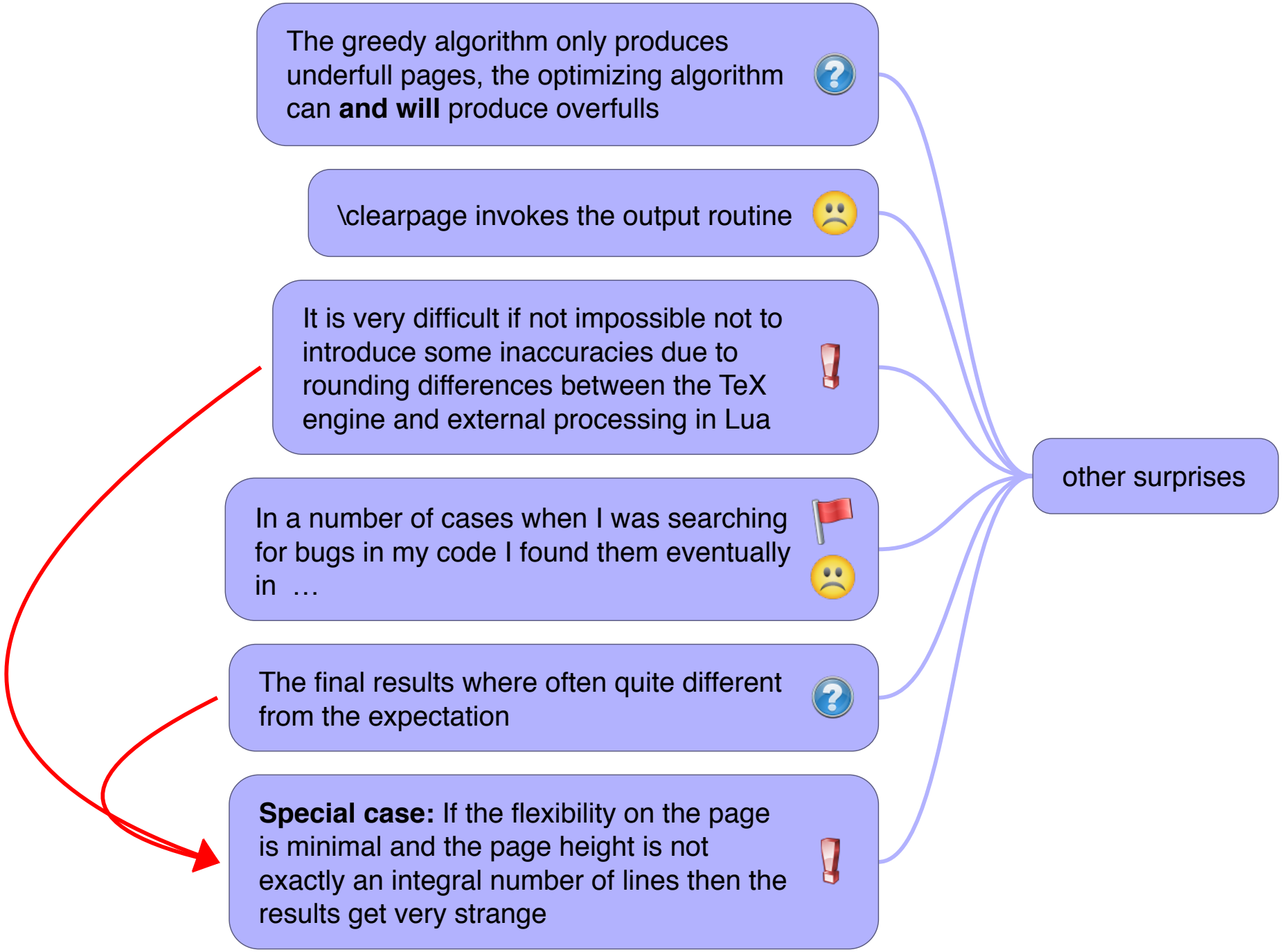

\looseness woes
By default setting \looseness > 0 gives you very short last lines with a single word in it
Environments like center will generate empty extra lines in response to \looseness > 0
Setting \looseness=-1 may result in enlarging the paragraph and vice versa
this needs addressing at the macro code level by providing a flag that tells phase 2 not to apply \looseness changes to a certain paragraph

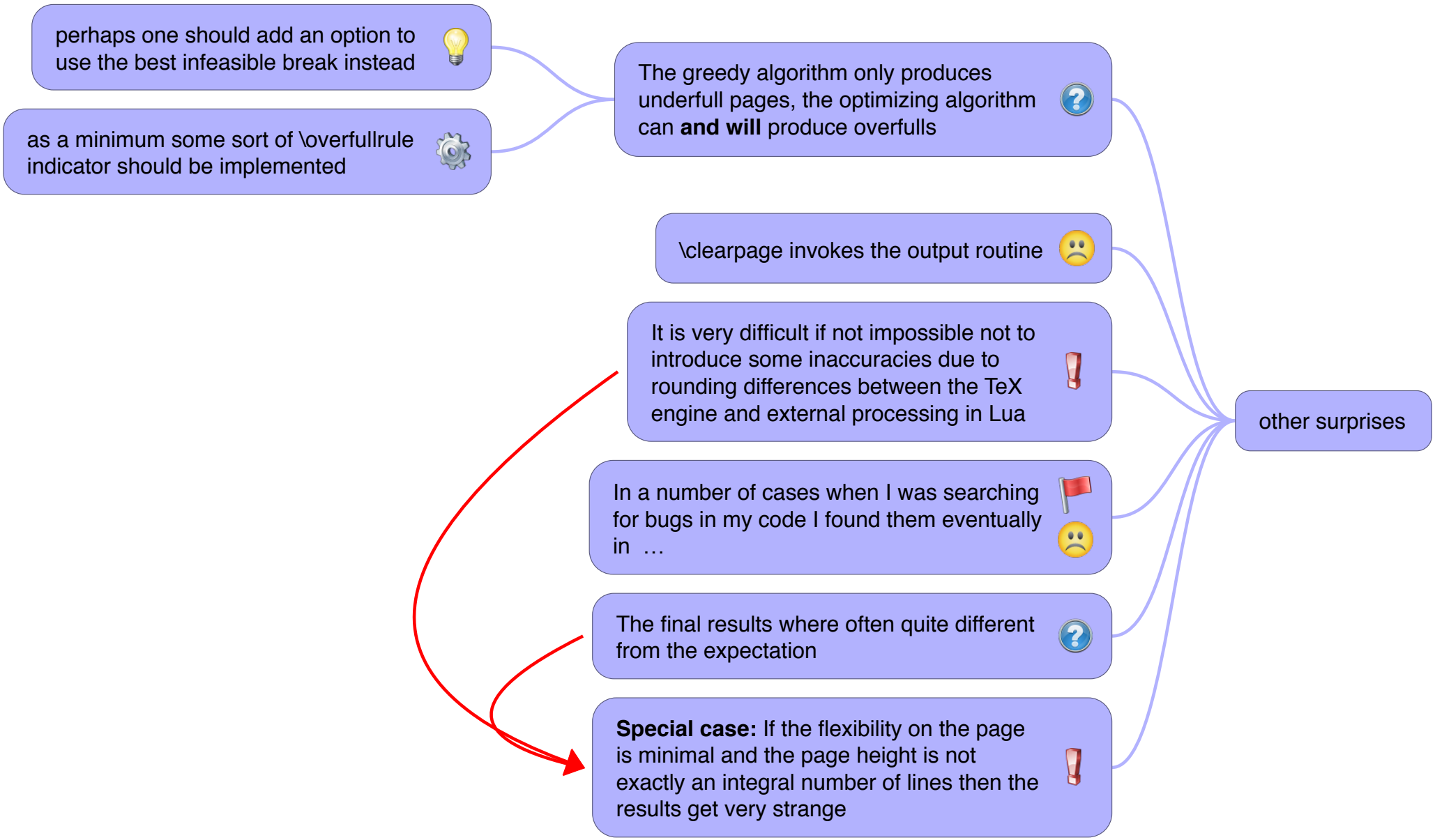

\looseness woes
By default setting \looseness > 0 gives you very short last lines with a single word in it
Environments like center will generate empty extra lines in response to \looseness > 0
Setting \looseness=-1 may result in enlarging the paragraph and vice versa
addressed by carefully checking the results and not just assuming that the TeX reporting the \looseness setting worked actually means that it work as expected

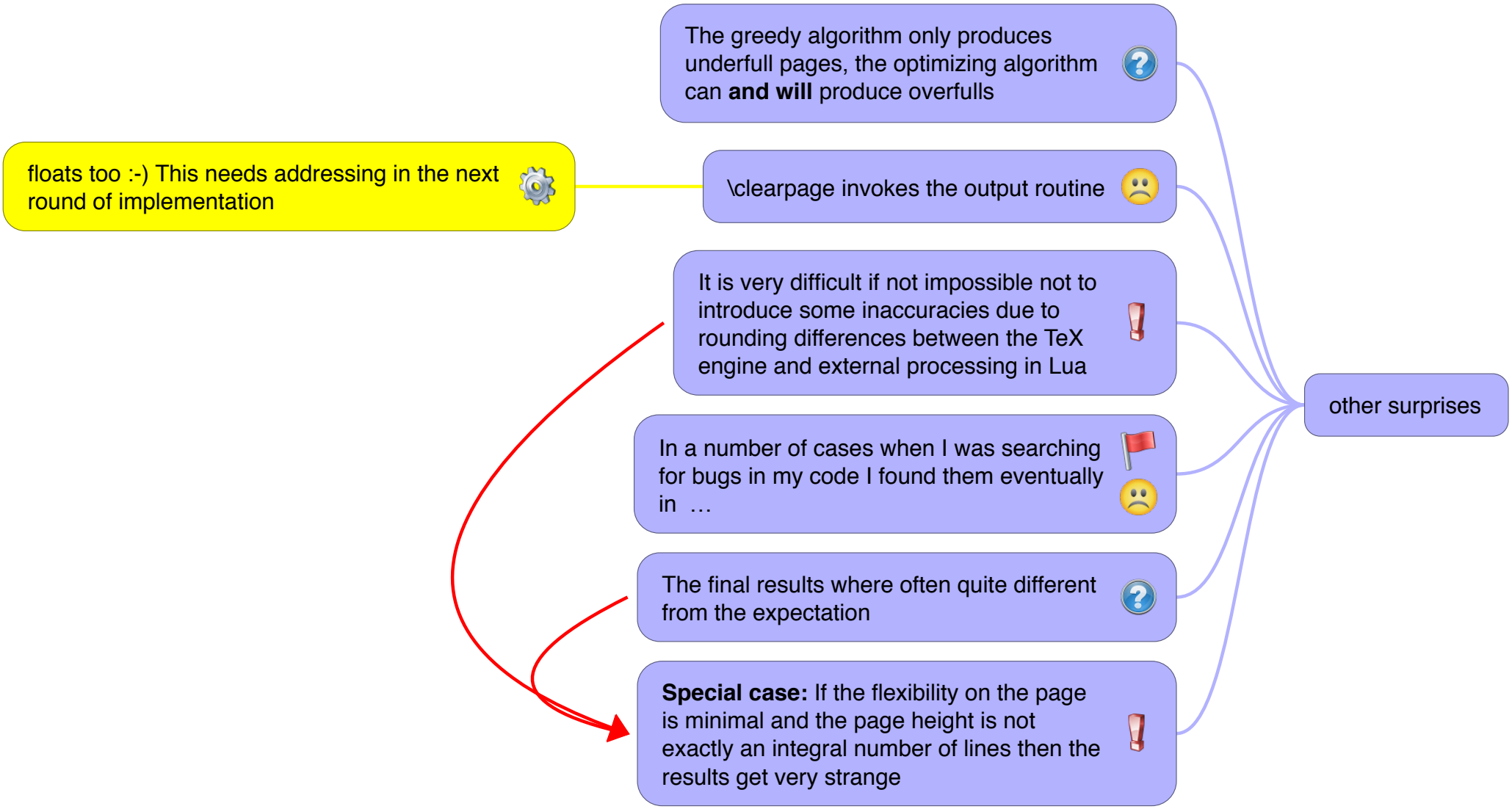

Different paragraph variations may result in different values of \prevdepth after the paragraph, but the material following is always set as if \looseness=0 was used
(Lua)TeX may report success setting a paragraph with \looseness=-1 even though it is invalid as it contains an overfull line
paragraph variation woes

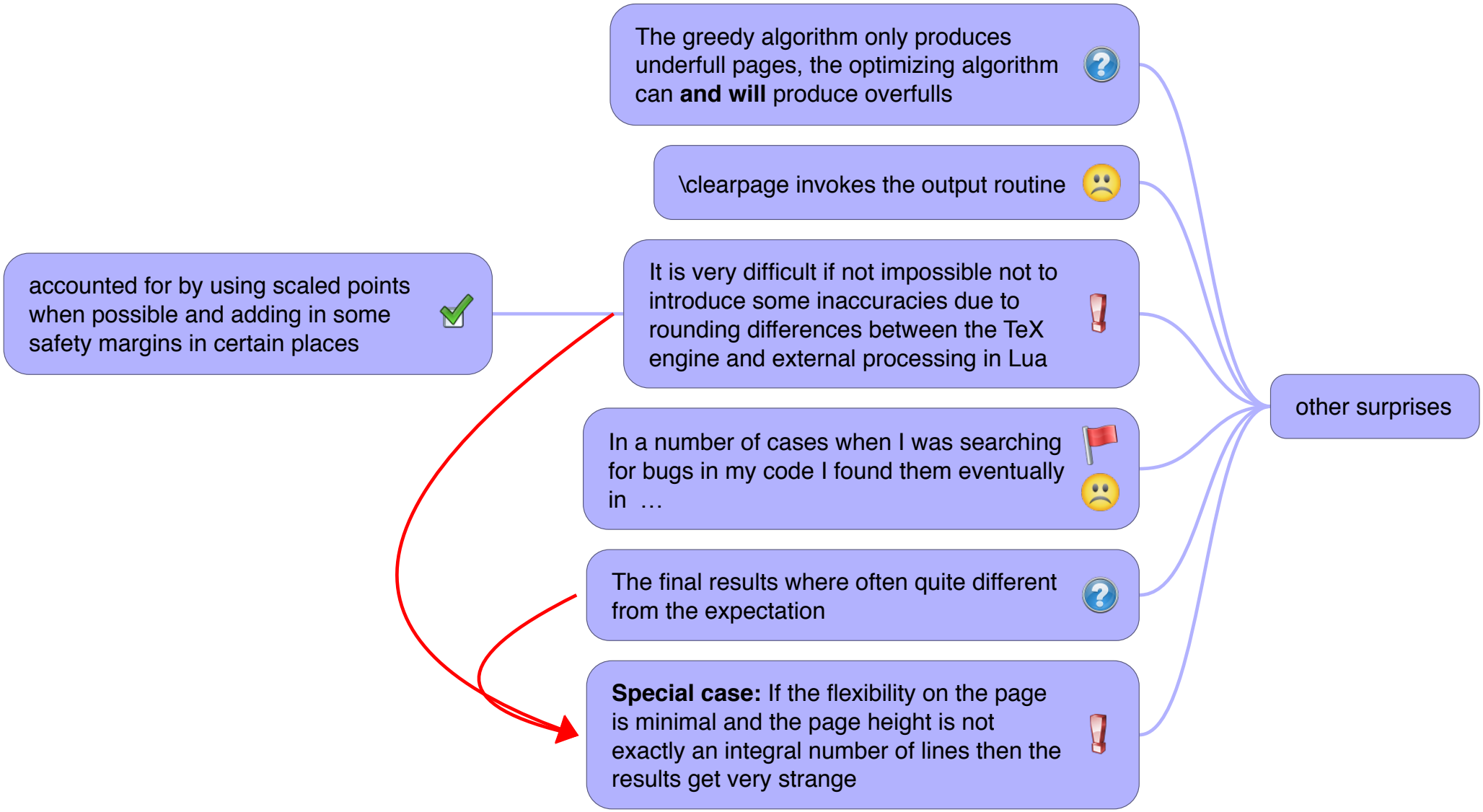

addressed by monitoring all variation for a change in \prevdepth and adjusting their output, so that they all can continue in the same way
Different paragraph variations may result in different values of \prevdepth after the paragraph, but the material following is always set as if \looseness=0 was used
(Lua)TeX may report success setting a paragraph with \looseness=-1 even though it is invalid as it contains an overfull line
paragraph variation woes

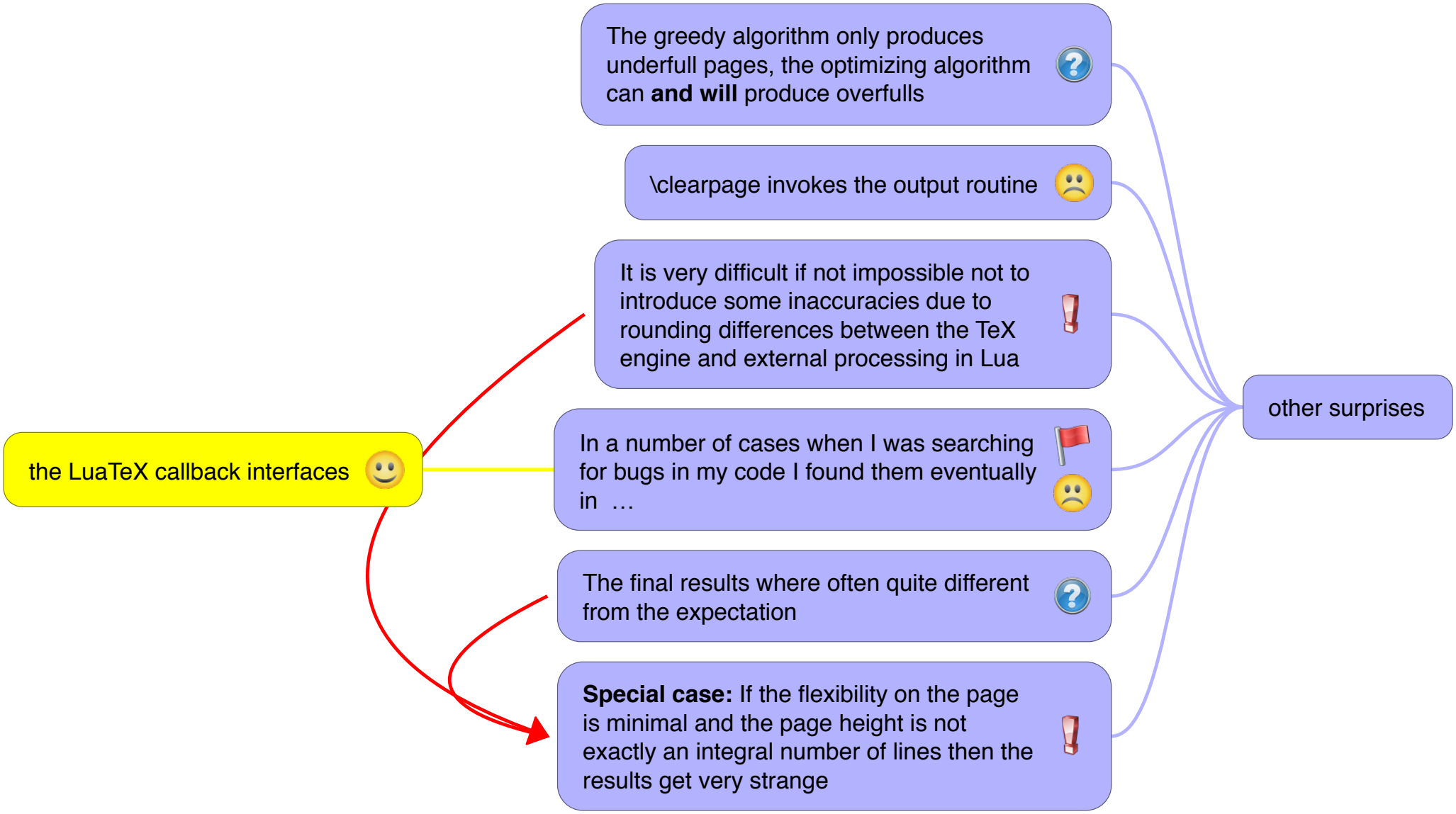

paragraph variation woes
Different paragraph variations may result in different values of \prevdepth after the paragraph, but the material following is always set as if \looseness=0 was used
(Lua)TeX may report success setting a paragraph with \looseness=-1 even though it is invalid as it contains an overfull line
This will happen if already the \looseness=0 case is invalid. As the algorithm supports different values for \tolerance (without and with \looseness) this is a possibility.
addressed by checking for it

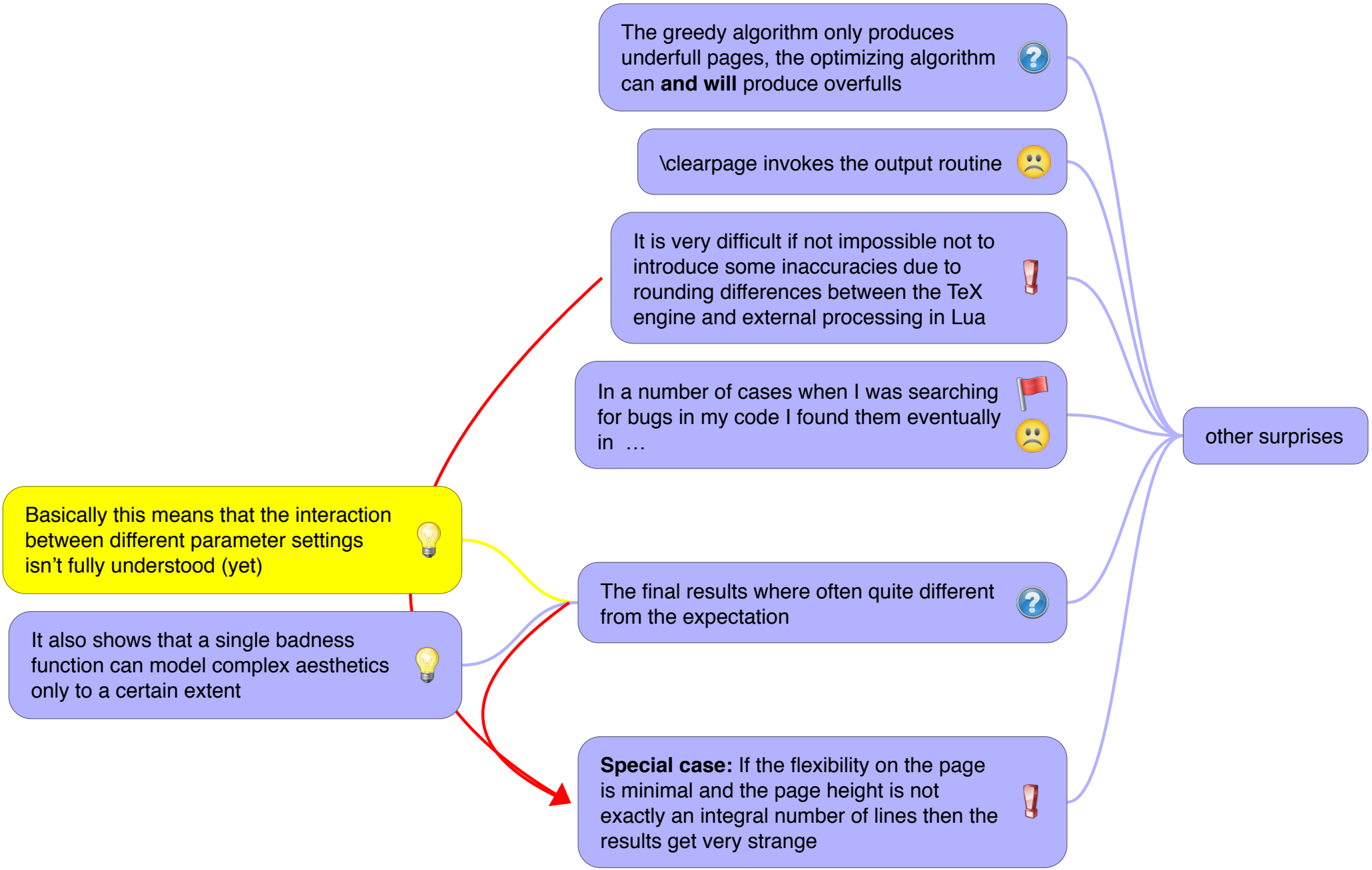

The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange
other surprises


other surprises
The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
perhaps one should add an option to use the best infeasible break instead
as a minimum some sort of \overfullrule indicator should be implemented
\clearpage invokes the output routine
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange


other surprises
The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
floats too :-) This needs addressing in the next round of implementation
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange

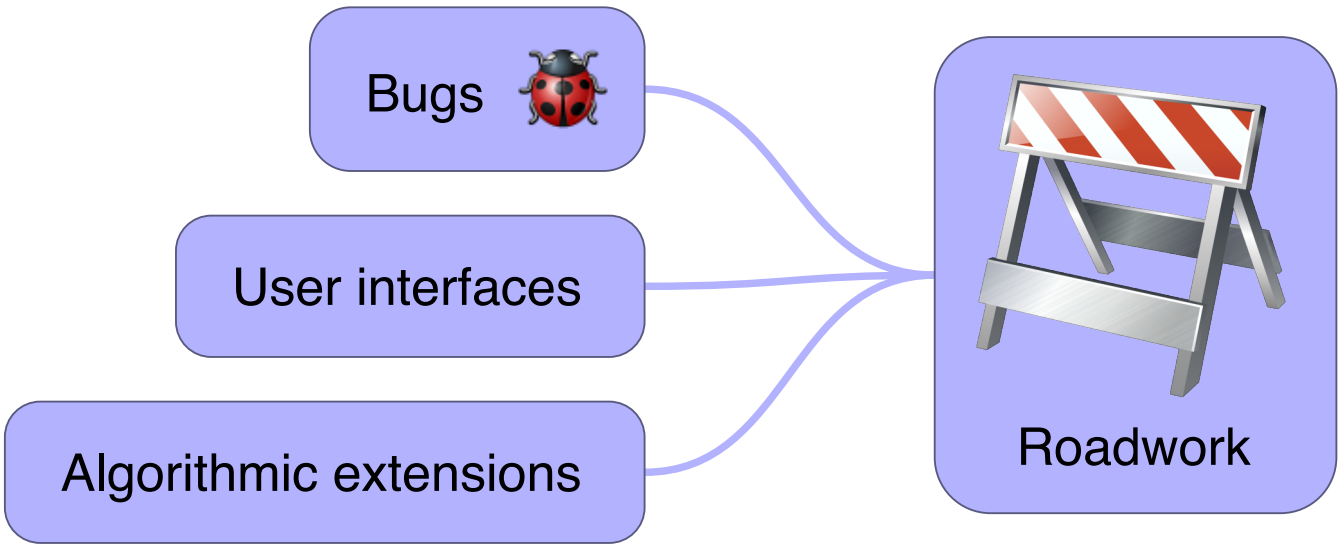

The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
accounted for by using scaled points when possible and adding in some safety margins in certain places
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
other surprises
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange

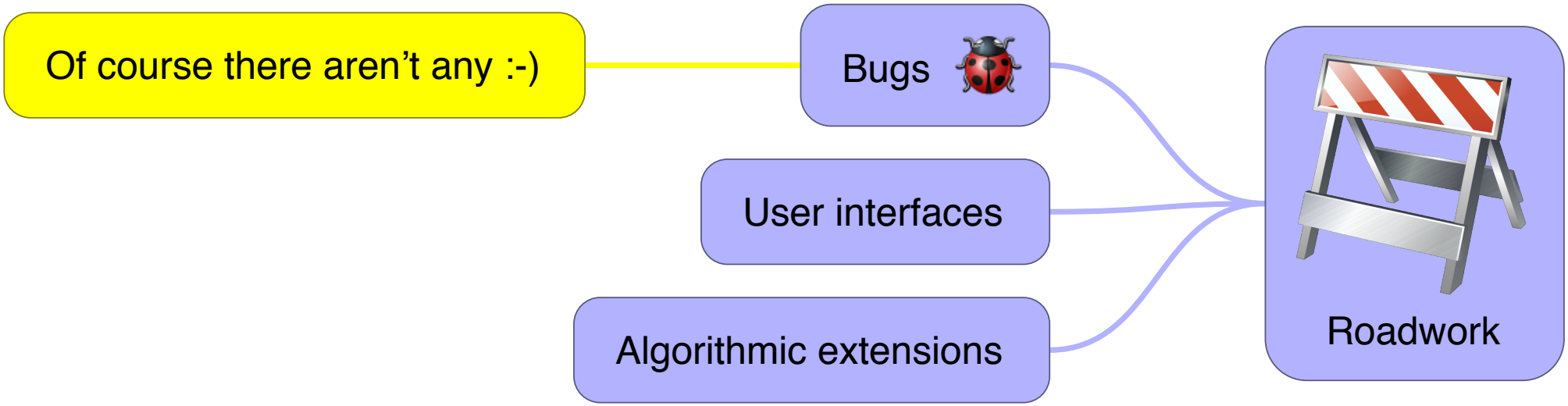

the LuaTeX callback interfaces
The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange
other surprises

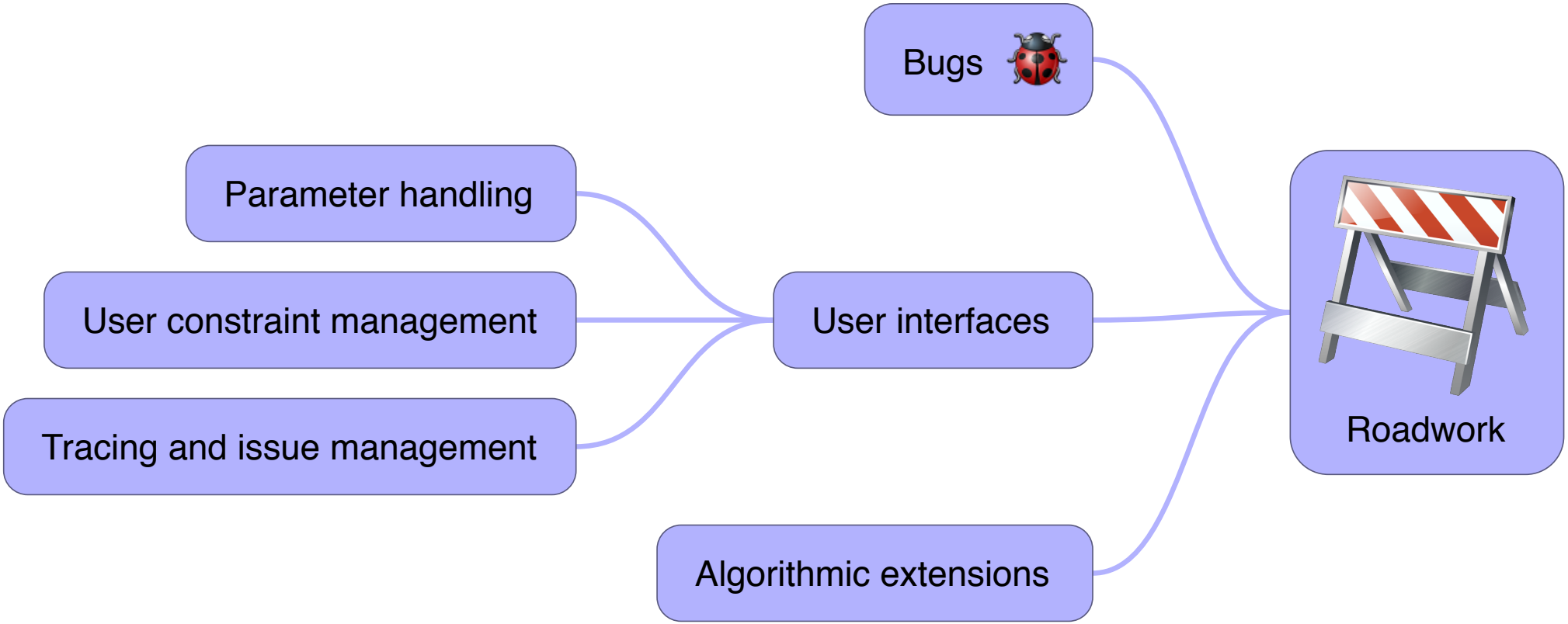

The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
other surprises
Basically this means that the interaction between different parameter settings isn't fully understood (yet)
It also shows that a single badness function can model complex aesthetics only to a certain extent
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange

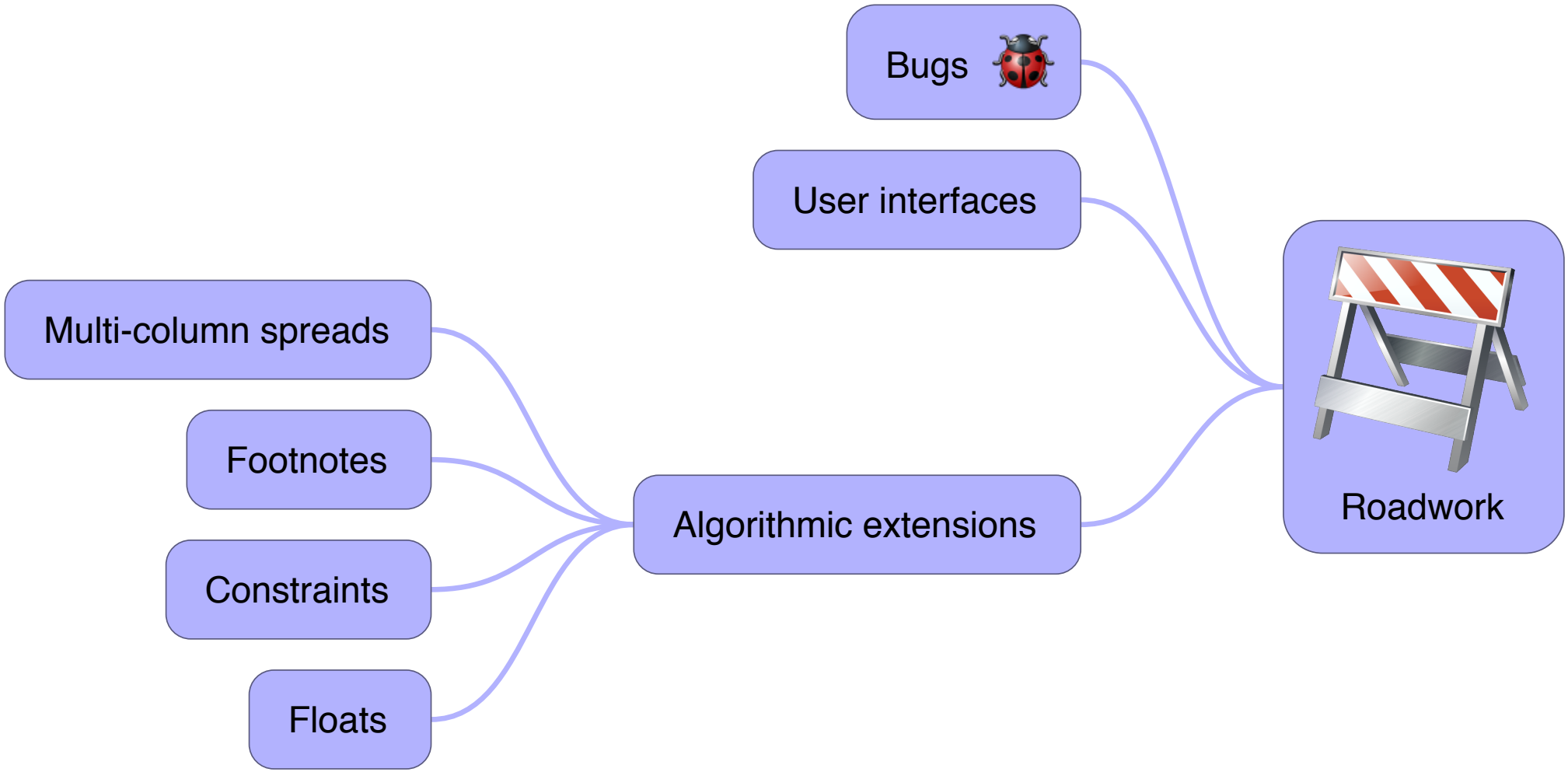

other surprises
The greedy algorithm only produces underfull pages, the optimizing algorithm can **and will** produce overfulls
\clearpage invokes the output routine
It is very difficult if not impossible not to introduce some inaccuracies due to rounding differences between the TeX engine and external processing in Lua
In a number of cases when I was searching for bugs in my code I found them eventually in …
The final results where often quite different from the expectation
Special case: If the flexibility on the page is minimal and the page height is not exactly an integral number of lines then the results get very strange
If one thinks about it is actually quite clear why, but first it got me puzzled a couple of times.

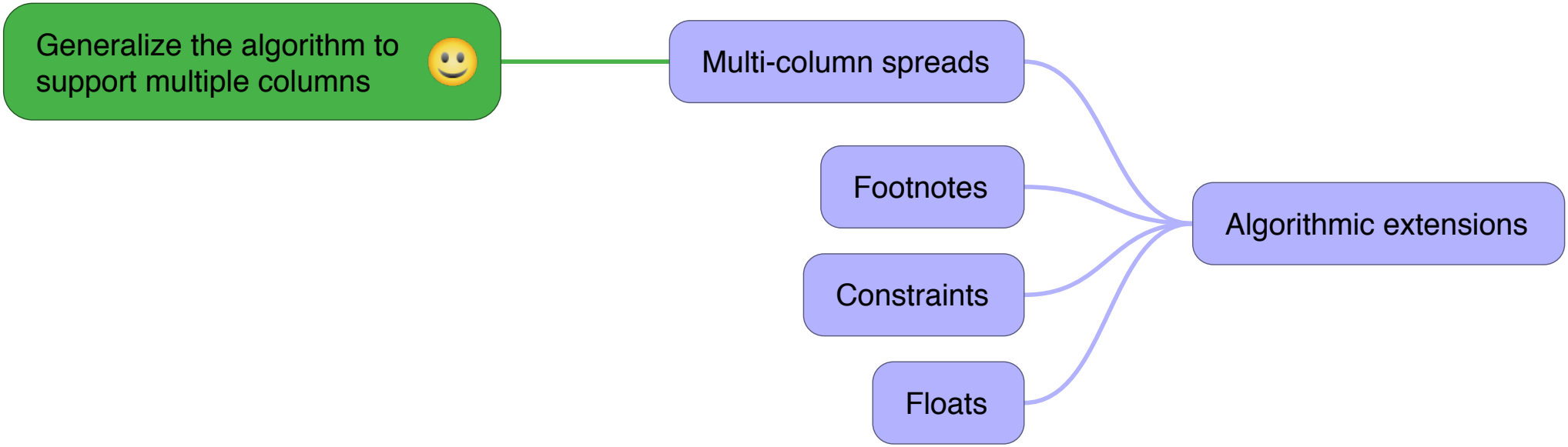
Roadwork

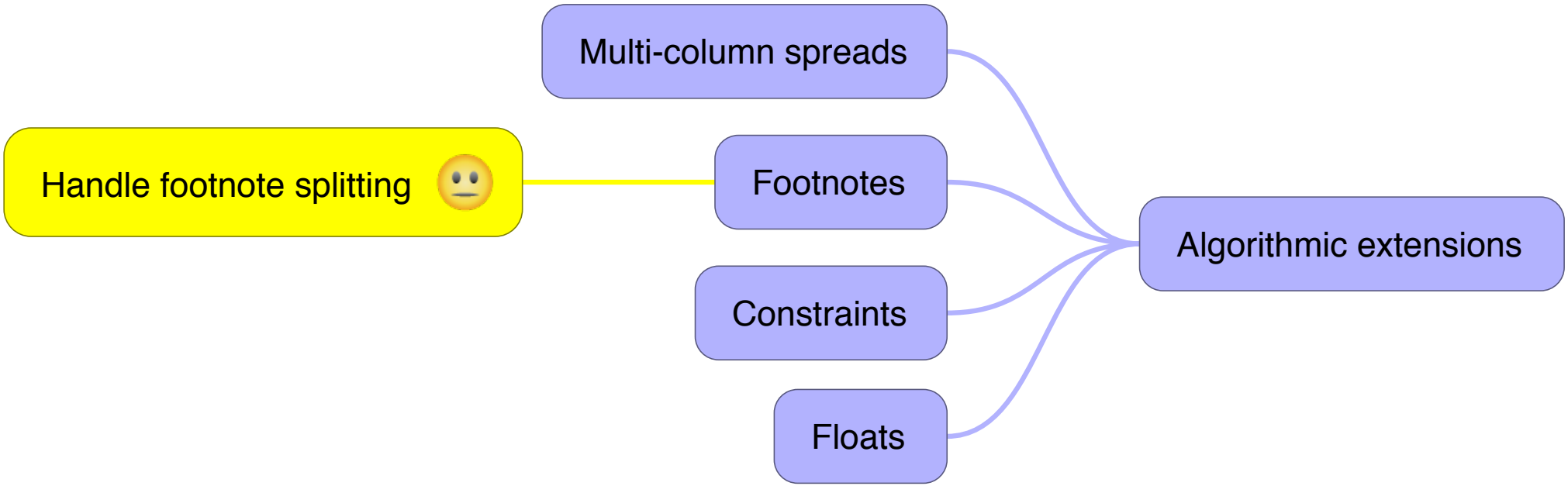

Bugs
User interfaces
Algorithmic extensions
Roadwork

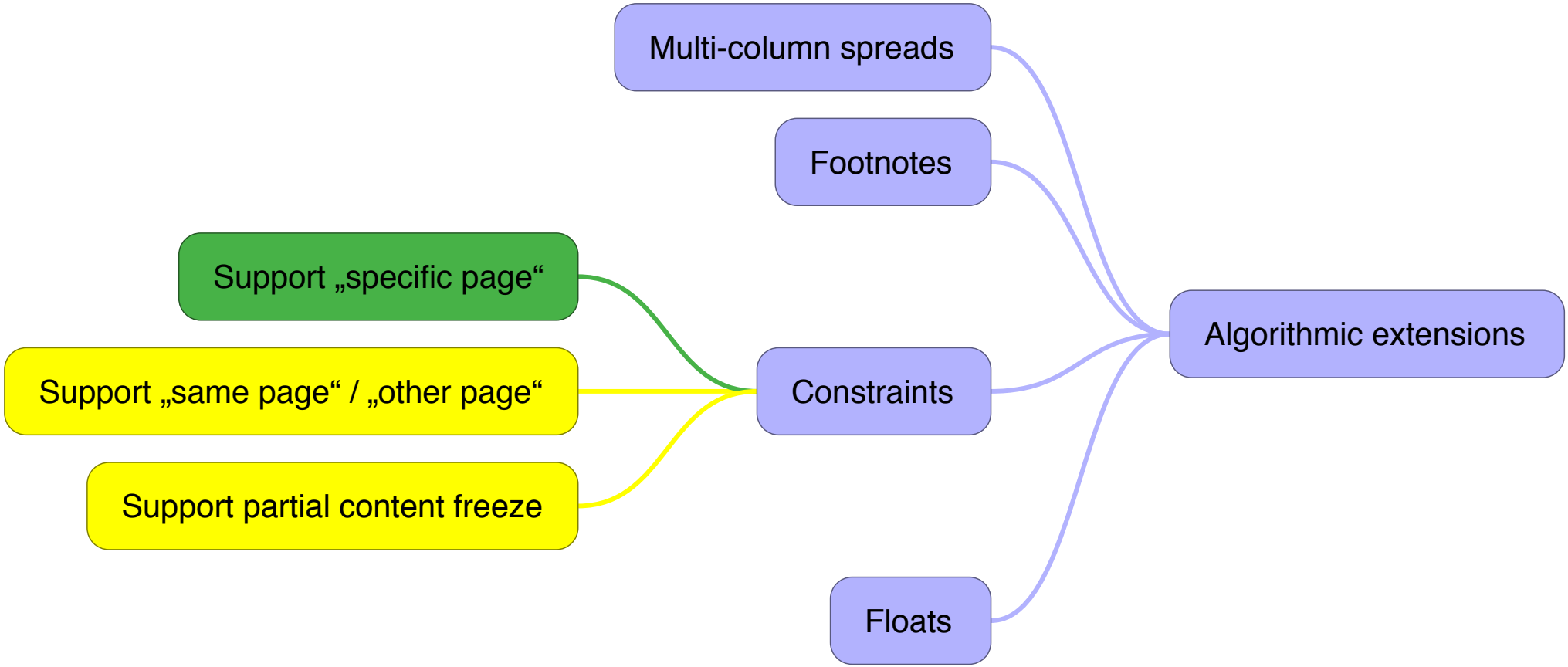

Of course there aren’t any :-)
Bugs
User interfaces
Algorithmic extensions
Roadwork

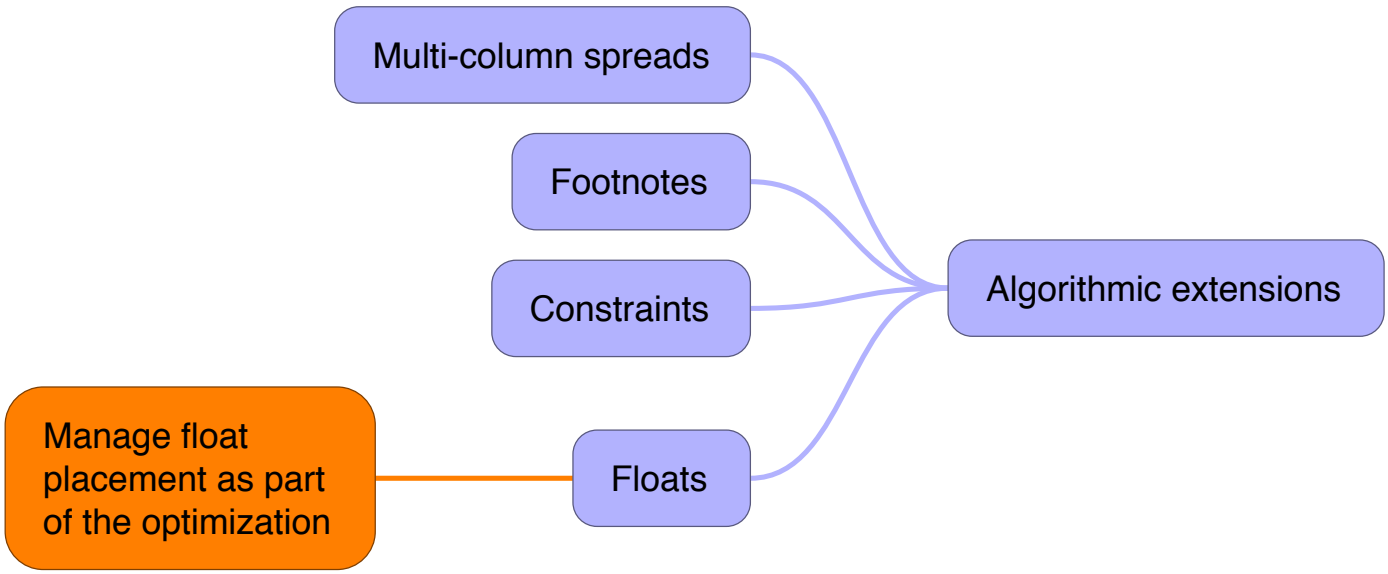

Bugs
Parameter handling
User constraint management
Tracing and issue management
User interfaces
Algorithmic extensions
Roadwork


Bugs
User interfaces
Multi-column spreads
Footnotes
Constraints
Floats
Algorithmic extensions
Roadwork


Generalize the algorithm to support multiple columns
Multi-column spreads
Footnotes
Constraints
Floats
Algorithmic extensions


Multi-column spreads
Handle footnote splitting 😐
Footnotes
Algorithmic extensions
Constraints
Floats


Multi-column spreads
Footnotes
Support „specific page“
Support „same page“ / „other page“
Support partial content freeze
Constraints
Algorithmic extensions
Floats


Multi-column spreads
Footnotes
Constraints
Floats
Algorithmic extensions
Manage float placement as part of the optimization

Manage float placement as part of the optimization

The number of floats that can appear on a page is bounded by a constant

Therefore number of different layouts to examine is bounded

thus comparable to the spread situation only that there are more possibilities

In the algorithm with just spreads we were able to drop all inferior solutions reaching a breakpoint *b* when the solution involves the same page height and the same page type (verso/recto)

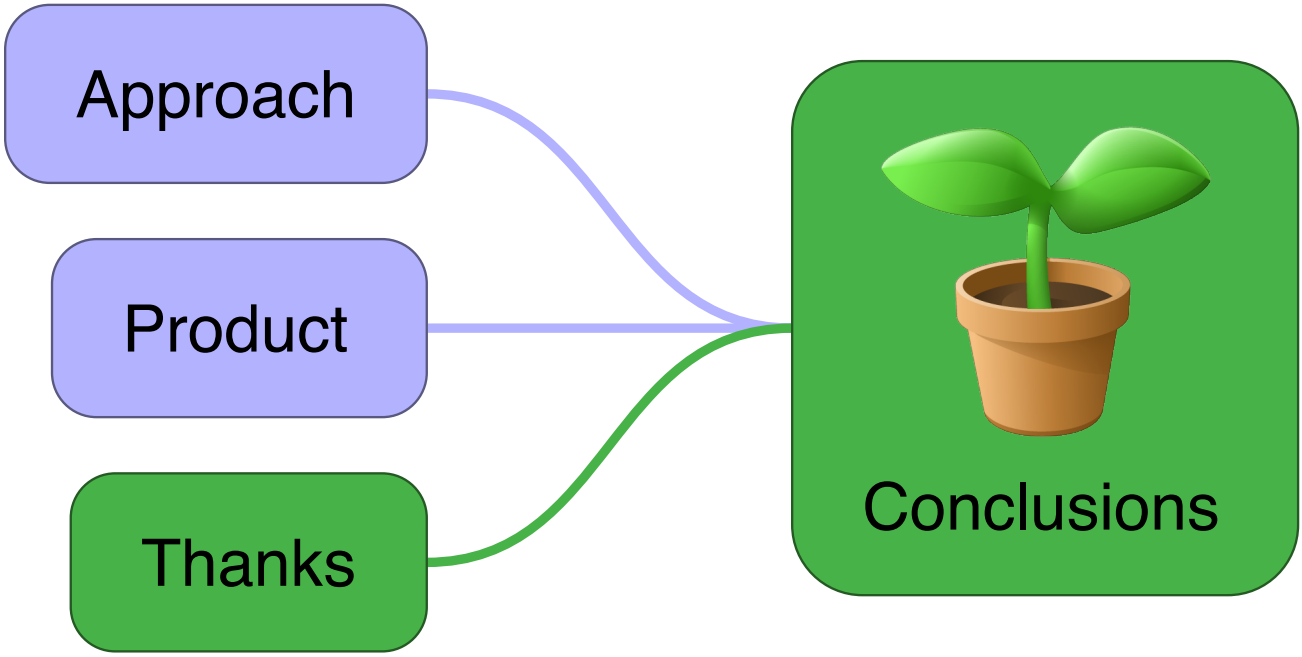
With floats the equivalent class is more complex: additionally we need to have the same floats being typeset on the pages up to this point

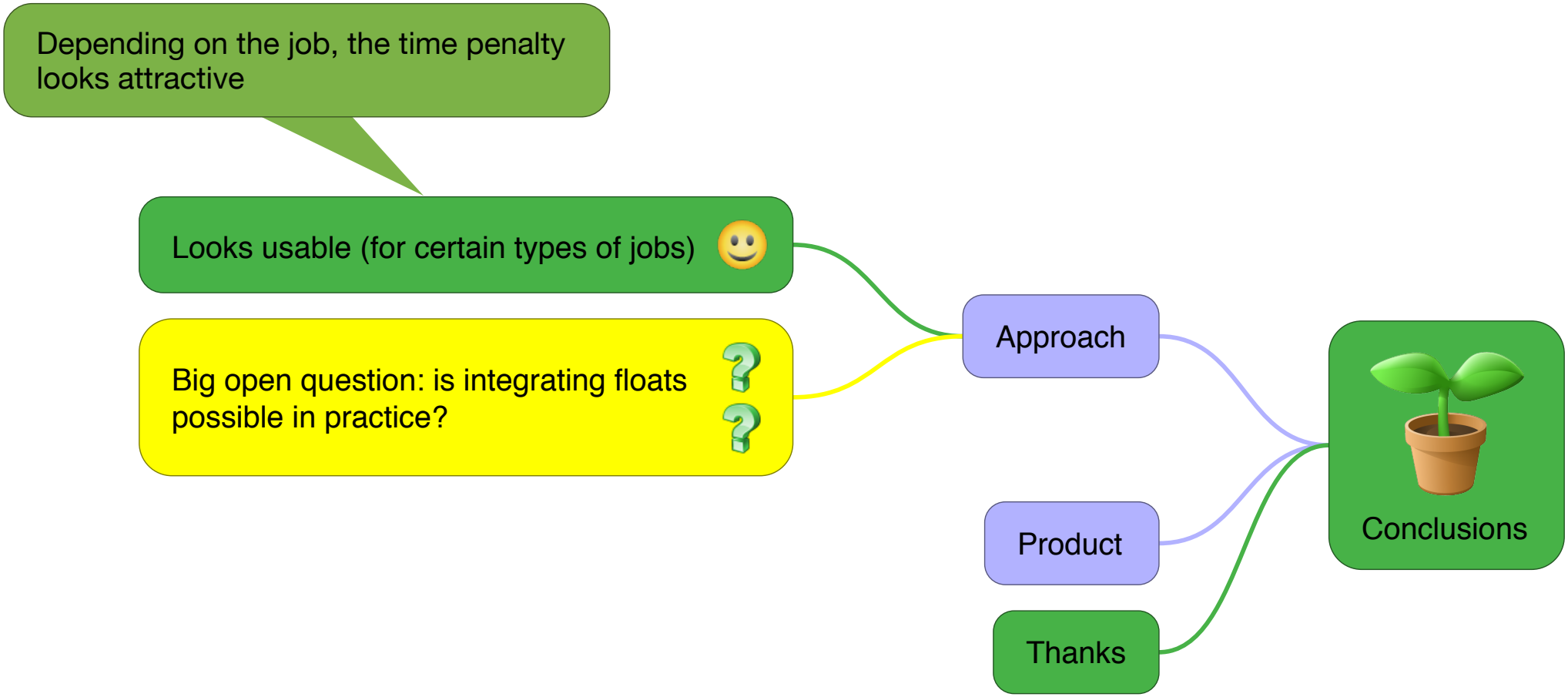
it is not necessary that they are typeset on the same pages though provided we use a sensible badness measure

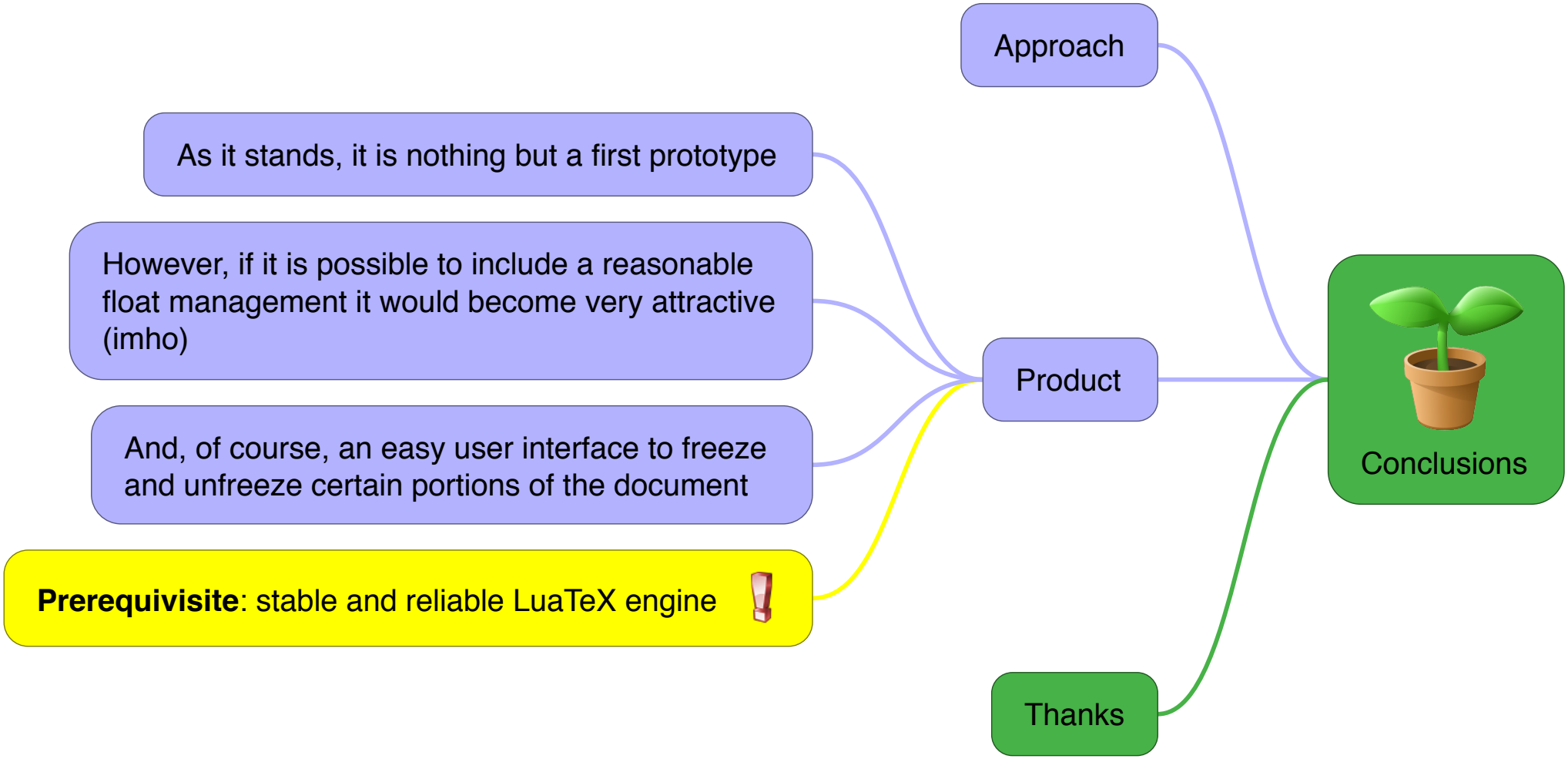
From the complexity perspective we are therefore in the same league and the time penalty should increase by a more or less constant factor compared to the algorithm presented here

Big question though: is this factor too large to make this a usable approach given current processing powers?

Conclusions


Approach
Product
Thanks
Conclusions


Depending on the job, the time penalty looks attractive
Looks usable (for certain types of jobs)
Big open question: is integrating floats possible in practice?
Approach
Product
Thanks
Conclusions


Approach
As it stands, it is nothing but a first prototype
However, if it is possible to include a reasonable float management it would become very attractive (imho)
Product
And, of course, an easy user interface to freeze and unfreeze certain portions of the document
Prerequivisite: stable and reliable LuaTeX engine
Conclusions
Thanks

Approach

Product

To **Hans** and **the LuaTeX team** for the great idea and the work carried out to marry Lua and TeX and allow me to pursue an old dream of mine — experimenting with globally optimized pagination

… and thanks to **you** for patiently listening

Thanks

Conclusions

/Alice in Bachotek/Introduction/Example: Typesetting Alice/General settings

```
\textheight = 550.0pt (46 lines a 12pt)
\textwidth  = 229.5pt (approx 50-55 characters per line)

\clubpenalty      = 10000    % no orphans
\widowpenalty     = 10000    % no widows
\parskip          = 0pt      % no paragraph separation
flexibility
\@beginparpenalty = 9999     % strongly discourage breaks in
front of
                             % „verse“ and similar environments
\@secpenalty      = -9000    % strongly encourage section breaks

\tolerance        = 4000     % allow fairly loose paragraphs
```

/Alice in Bachotek/Introduction/Example: Typesetting Alice/Typeset result/Medium desaster

— 36 pages (72 columns)
— 27 columns underfull with badness 10000 and 4 with badness > 3567 = **37% bad columns!**
— one heading (out of 11) at column start
— time to process < 1second … time to fix **???**

/Alice in Bachotek/The pagination framework/phase1

The document, which consists of standard TEX files, is processed by a TEX engine without any modification
until all implicit content (e.g., table of content, bibliography, etc.) is generated and all cross-references are resolved.

The engine is modified to interact with TEX's way of filling the main vertical list (from which, in an asynchronous way, TEX later cuts column material for pagination).
In particular, whenever TEX is ready to move new vertical material to the main vertical list this material is intercepted and analyzed. Information about each block (vertical size, depth, stretchability if any and penalty of a breakpoint) is then gathered and written out to an external file. If possible, data is accumulated, e.g., several objects in a row without any possibility for breaking them up are written out as a single data point to reduce later processing.
The modification is also able to interpret special flags (implemented as new types of "whatsit nodes" in TEX engine lingo) that can signal the start/end or switch of an explicit variation in the input source. This information is then used to structure the corresponding data in the output file for later processing.
The second modification to the engine is to intercept the generation of paragraphs targeted for the main galley prior to TEX applying line breaking:

- For each horizontal list that is passed to the line-breaking algorithm the framework algorithm then determines the number of acceptable variations in "looseness" within the specified parameter settings.

- For each possible variation it then does a paragraph breaking trial to determine the exact sequence of lines, vertical spaces and associated penalties under a specific "looseness" value.

- The results of each trial is externally recorded together with the associated "looseness" value of the variation.

- Finally, instead of adding a vertical list representing the paragraph to the main vertical list, a single special node is passed so that the paragraph material is not collected again by the first modification described above.

As the result of this phase the external file will hold an abstraction of the document galley material including marked up variations for each paragraph.

/Alice in Bachotek/The pagination framework/phase3

The result of phase 2 is used as input to a global optimizing algorithm modeled after the Knuth/Plass algorithm for line breaking that uses dynamic programming to determine an optimal sequence of page/ column breaks throughout the whole document. Com- pared to the line-breaking algorithm this page-breaking algorithm provides the following additional features:

- Support for variations within the input: This is used to automatically manage variant break sequences resulting from different paragraph breakings calculated in phase 2, but could also be used to support, for example, variations of figures in different size or similar applications.

- Support for shortening or lengthening the vertical size of double spreads to enable better columns/ page breaks across the whole document.

- Global optimization is guided by parameters that allow a document designer to balance the importance of individual aspects (e.g., avoiding widows against changing the page length or using sub- optimal paragraphs) against each other.

The result of this phase will be a sequence of optimal page break positions within the input together with length information for all pages/columns for which it applies. Also recorded is which of the variants have been chosen when selecting the optimal sequence.

/Alice in Bachotek/The pagination framework/phase3/Pagination algorithm written i.../Inputs/ Parameters to influence the be...

/Alice in Bachotek/First results/Typesetting Alice again/Adding \parskip and \baselines...

```
\baselineskip = 12pt plus 0.05pt
\parskip      =  2pt plus 1pt minus 1pt

All other values the same.
```

/Alice in Bachotek/First results/Typesetting Alice again/A first demo/Setting up the parameters in p...

/Alice in Bachotek/First results/Typesetting Alice again/Adding \parskip and \baselines.../Results/with standard TeX

```
Page 1  — second column badly spaced out (because of orphan/
widow settings)
Page 2  — second column very badly spaced out (because of *
block following)
Page 7  — second column very badly spaced out (because of mouse
tail following)
Page 8  — first column spaced out (because of orphan/widow
settings)
Page 15 — first column spaced out (because of heading following)

In total:

  — 37 pages (74 columns)
  — 5 columns underfull with badness above 2100 and twice 10000
= 5,4% bad columns!
  — one heading (out of 11) at column start
  — time to process < 1second  … time to fix ???
```

/Alice in Bachotek/First results/Typesetting Alice again/Adding \parskip and \baselines.../Results/when optimizing

No bad pages!

In total:
— 37 pages (74 columns)
— all columns with badness < 2000 (no bad columns)
— 3 headings (out of 11) at column start

/Alice in Bachotek/First results/Typesetting Grimm Fairy Tales/Adding \parskip and \baselines...

```
\baselineskip = 12pt plus 0.05pt
\parskip      =  2pt plus 1pt minus 1pt

All other values the same.
```

/Alice in Bachotek/First results/Typesetting Grimm Fairy Tales/Adding more \parskip and \base...

```
\baselineskip = 12pt plus 0.2pt
\parskip      =  2pt plus 2pt minus 1pt

All other values the same.
```

/Alice in Bachotek/First results/Typesetting Grimm Fairy Tales/Conclusions/However, because of the very l...

```
\baselineskip = 12pt plus 0.2pt
\parskip      =  2pt plus 2pt minus 1pt
```

/Alice in Bachotek/First results/Typesetting Pride and Prejudic.../Adding \parskip and \baselines...

```
\baselineskip = 12pt plus 0.05pt
\parskip      =  2pt plus 1pt minus 1pt

All other values the same.
```

/Alice in Bachotek/First results/Typesetting Pride and Prejudic.../Adding more \parskip and \base...

```
\baselineskip = 12pt plus 0.2pt
\parskip      =  2pt plus 2pt minus 1pt

All other values the same.
```