Automated Extraction of Mixins in Cascading Style Sheets

Alan Charpentier
Univ. Bordeaux

Jean-Rémy Falleri
Bordeaux INP

Laurent Réveillère
Bordeaux INP

France

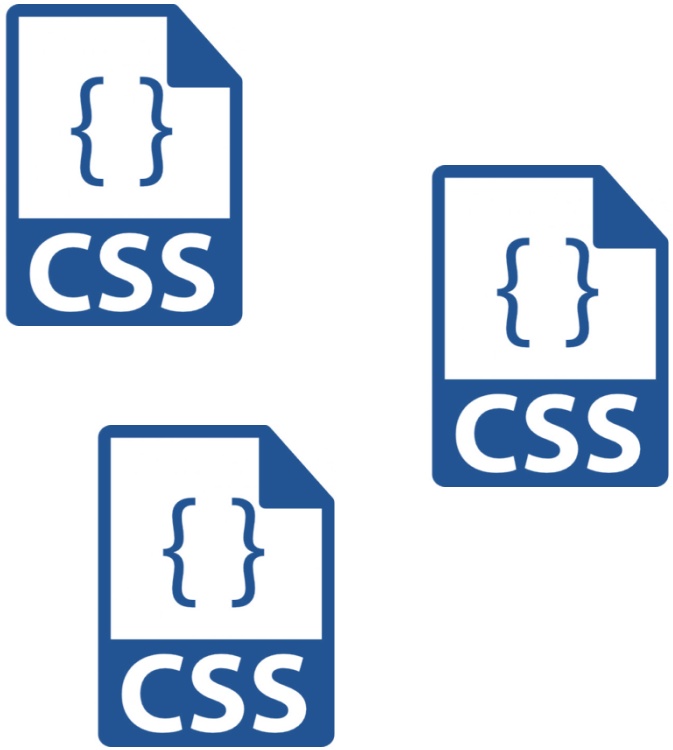
ICSME 2016

What is behind a web page?

- HTML to specify the content

What is behind a web page?

- HTML to specify the content
- CSS to describe **how HTML elements have to be displayed**

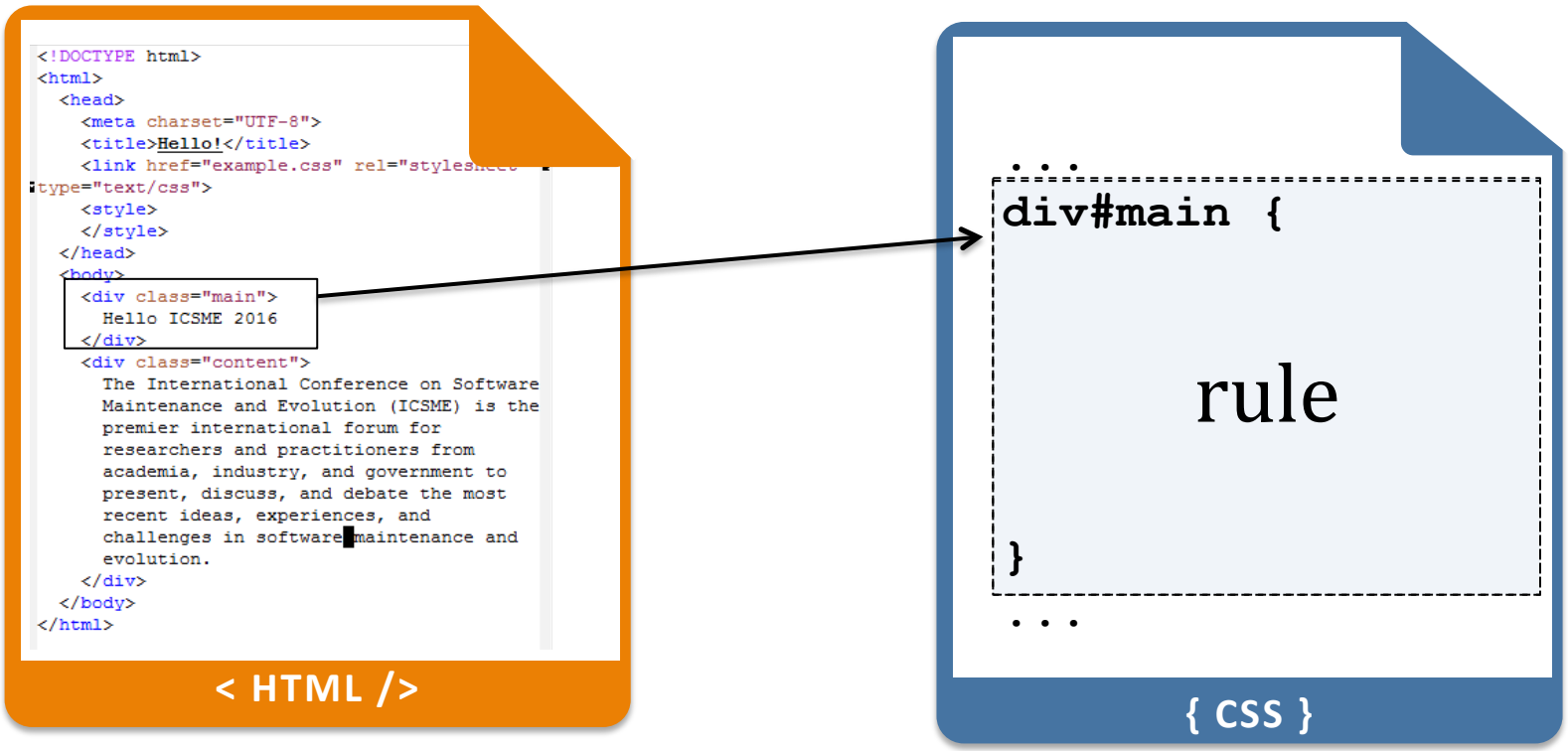


Graphical rendering

- HTML to specify the content
- CSS to des **displayed**

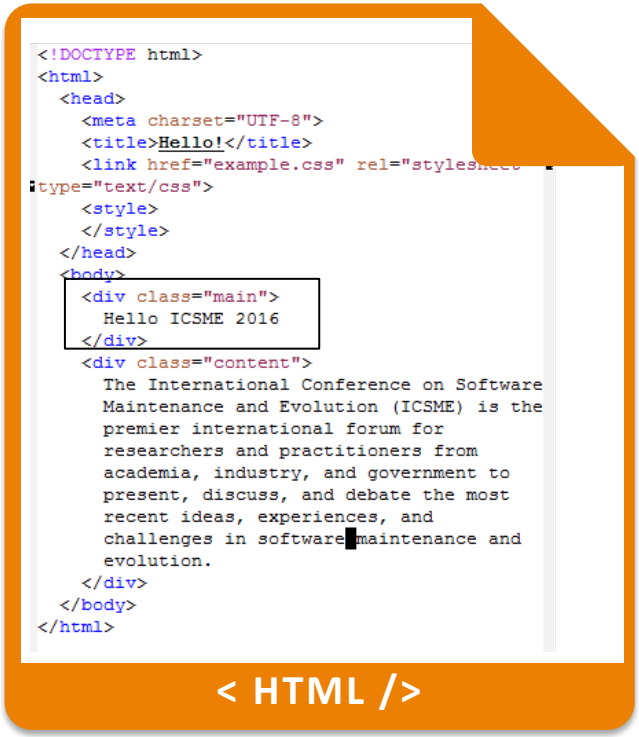
CSS in a Nutshell

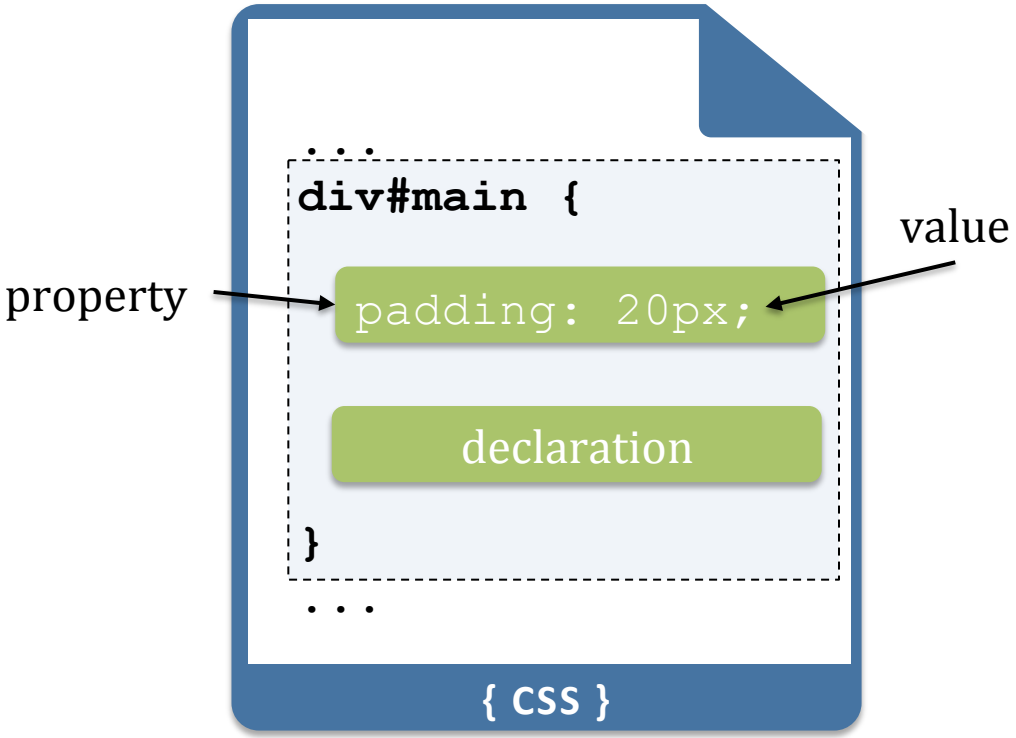
- CSS file : Set of rules
- Rule : Selector points to the HTML elements to style



CSS in a Nutshell

- Declaration definition of the style
  - Couple property & value





CSS nowadays

- One of the core web technologies with HTML and JavaScript
- Almost everywhere
  - More than 90% of existing websites rely on CSS

- PROS
  - Language is *quite* simple
    - Few constructs : selectors, properties, values

CSS nowadays

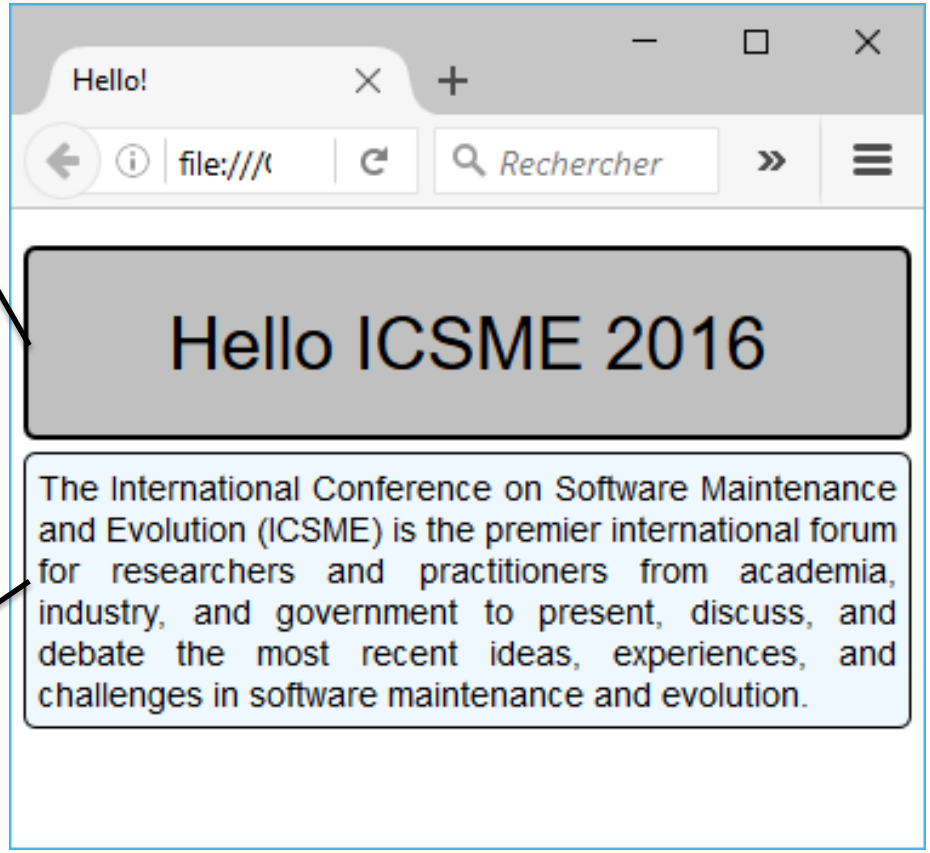
- One of the core web technologies with HTML and JavaScript
- Almost everywhere
  - More than 90% of existing websites rely on CSS

- CONS
  - Up to thousands of rules for one single CSS file
  - Language is **too** simple
  - No advanced constructs to avoid code duplication
    - Bad for reuse
    - Bad for maintenance
    - Bad for evolution
    - In other words, it is not so good …

Toy example illustrating code duplication

```css
div#main {
  border-radius: 5px;
  border-color: black;
  border-width: 2px;
  border-style: solid;
  background-color: Silver;
  text-align: center;
  margin: 5px;
  padding: 20px;
  font-size: 30px;
}

div#content {
  border-radius: 5px;
  border-color: black;
  border-width: 1px;
  border-style: solid;
  background-color:
AliceBlue;
  text-align: justify;
  margin: 5px;
  padding: 5px;
}
```

{ CSS }



Raising the level of abstraction of CSS

- CSS Pre-processors (LESS, Sass, …)
  - Compile into low-level CSS
  - Offer high level reuse mechanisms (variables, functions, mixins)
- Increasingly used by developers
  - More than 50% use a CSS pre-processor
  - Several big projects : Bootstrap, Foundations, …
- One of the main reuse mechanism
  - *mixins*

The use of mixins in Sass

```
div#main {
  border-radius: 5px;
  border-color: black;
  border-width: 2px;
  border-style: solid;
  background-color: Silver;
  text-align: center;
  margin: 5px;
  padding: 20px;
  font-size: 30px;
}

div#content {
  border-radius: 5px;
  border-color: black;
  border-width: 1px;
  border-style: solid;
  background-color: AliceBlue;
  text-align: justify;
  margin: 5px;
  padding: 5px;
}
```

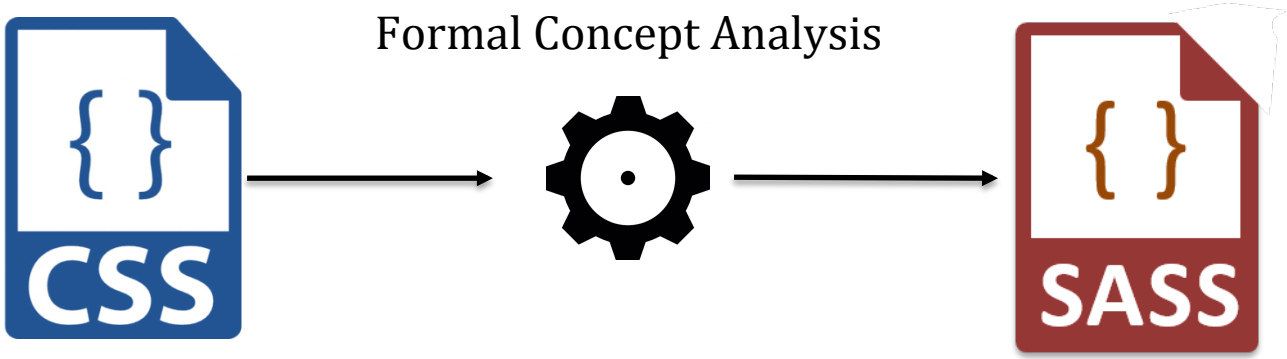
{ CSS }

```
@mixin box($w, $p, $c, $a) {
  border-radius: 5px;
  border-color: black;
  border-width: $w;
  border-style: solid;
  background-color: $c;
  text-align: $a;
  margin: 5px;
  padding: $p;
}

div#main {
  @include box(2px, 20px, Silver, center);
  font-size: 30px;
}

div#content {
  @include box(1px, 5px, AliceBlue, justify);
}
```

{ Sass }

Issues

- Legacy CSS code
  - Large code base
- Need for migration to CSS pre-processors
  - Requires manual identification of mixin opportunities
  - Requires manual code refactoring
  - ➔ Laborious and error-prone
- Our approach



Simplified use case

```css
div#main {
  border-radius: 5px;
  border-color: black;
  border-width: 2px;
  border-style: solid;
  background-color: Silver;
  text-align: center;
  margin: 5px;
  padding: 20px;
  font-size: 30px;
}

div#content {
  border-radius: 5px;
  border-color: black;
  border-width: 1px;
  border-style: solid;
  background-color:
AliceBlue;
  text-align: justify;
  margin: 5px;
  padding: 5px;
}
```

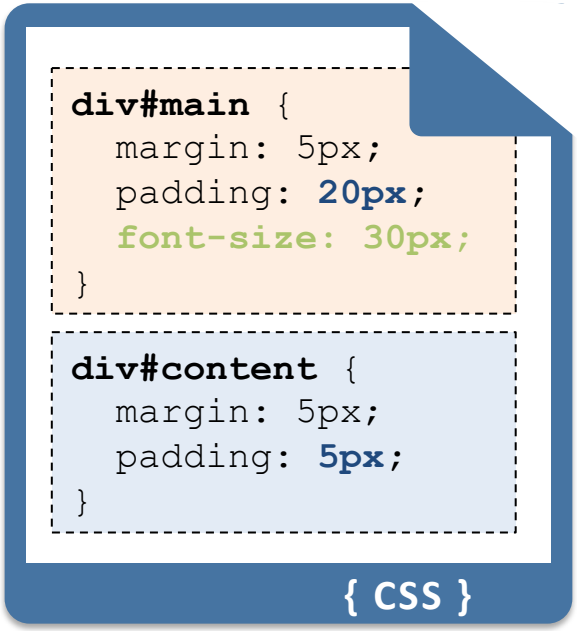
{ CSS }

Simplified use case

```
div#main {

  margin: 5px;
  padding: 20px;
  font-size: 30px;
}

div#content {

  margin: 5px;
  padding: 5px;
}
```

{ CSS }

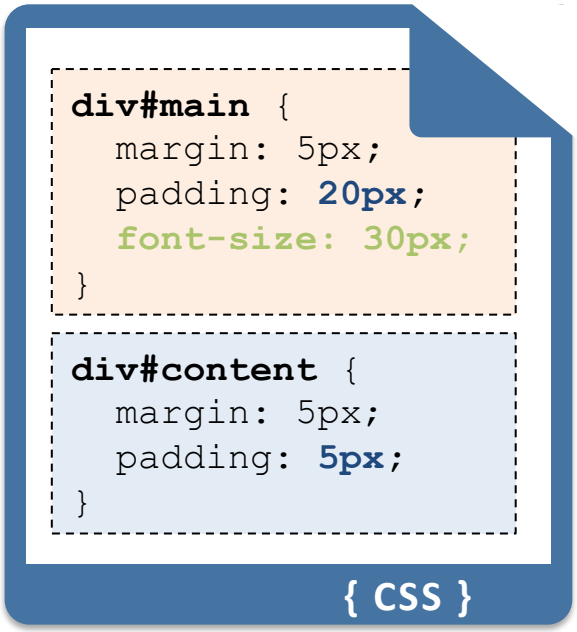
Step 1: formal context



Formal context

Binary relation associating objects with attributes

Step 1: Formal Context



Formal context

attributes

objects	margin	padding	margin: 5px	padding: 20px	font-size: 30px	font-size
div#main	✔	✔	✔	✔	✔	✔

Step 1: formal context

```
div#main {
  margin: 5px;
  padding: 20px;
  font-size: 30px;
}

div#content {
  margin: 5px;
  padding: 5px;
}
```

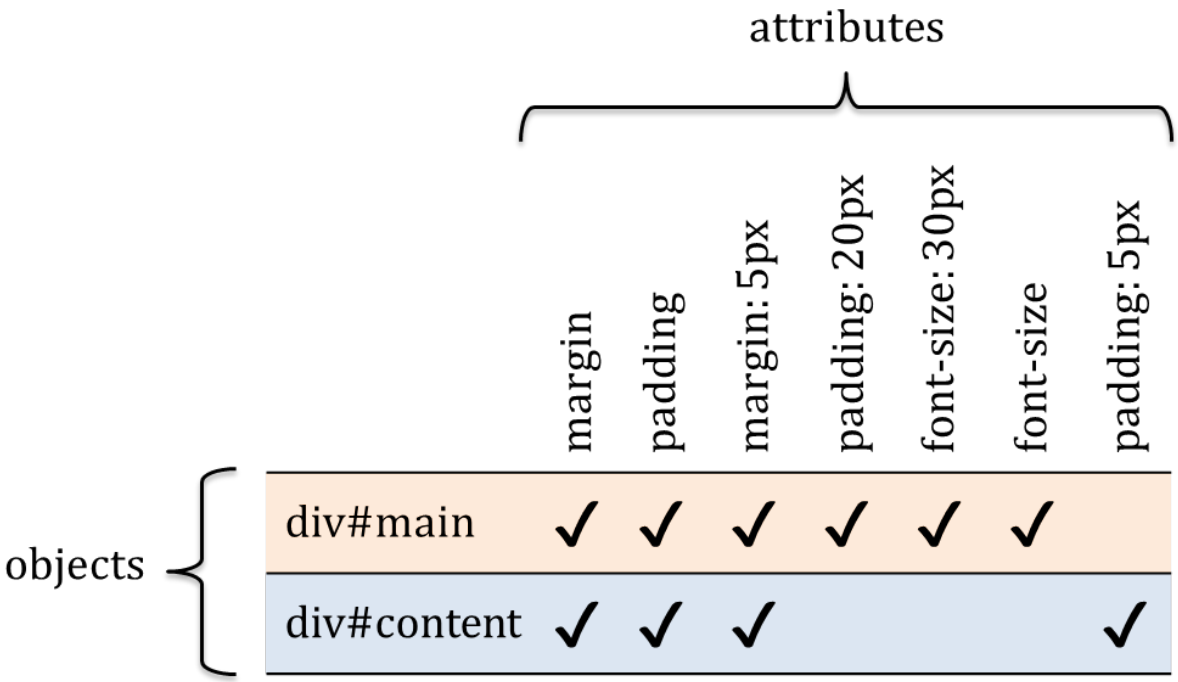
{ CSS }

Formal context

objects	margin	padding	margin: 5px	padding: 20px	font-size: 30px	font-size	padding: 5px
div#main	✔	✔	✔	✔	✔	✔	
div#content	✔	✔	✔				✔

Step 2 : formal concept

- Maximal collection of objects sharing common attributes
- Formal concept ≡ (*extent, intent*)
  - Extent : all objects that share the attributes in the *intent*
  - Intent : all attributes shared by the objects in the *extent*

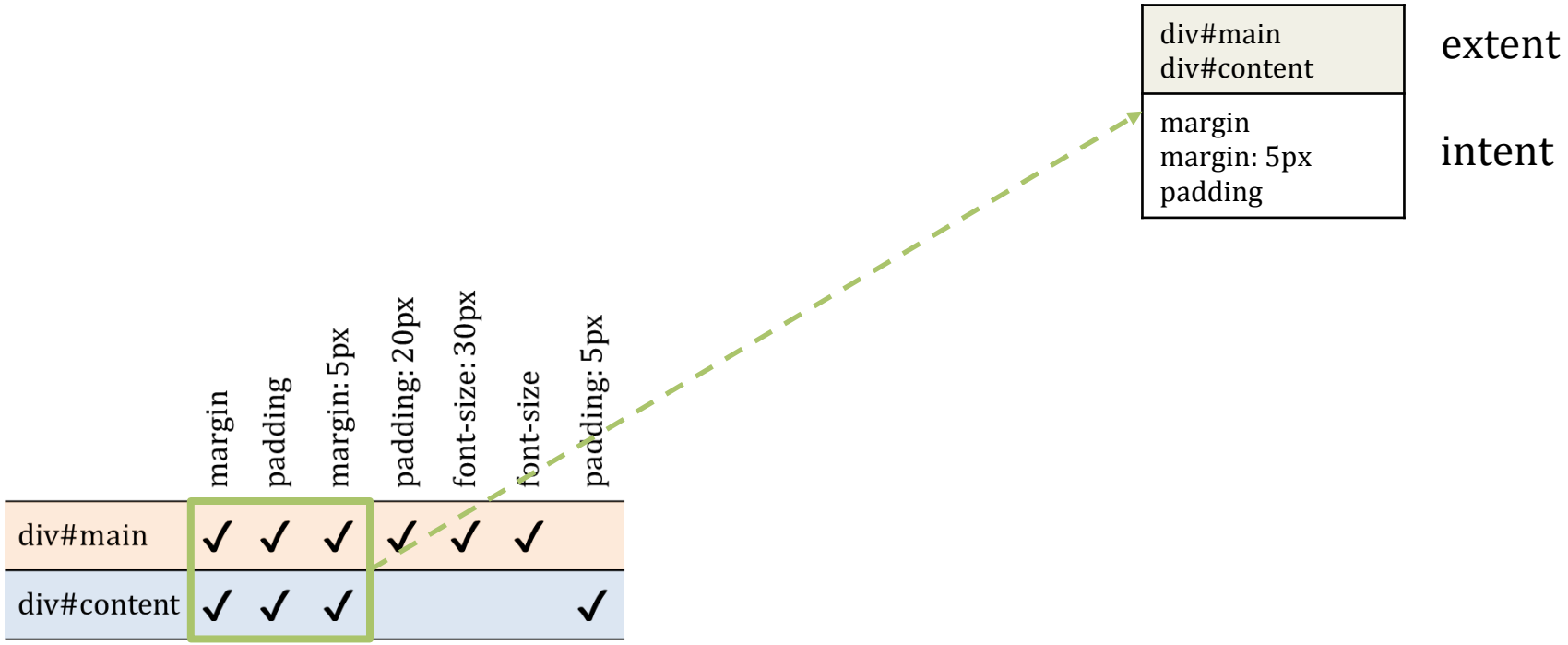


Step 2 : identifying formal concepts

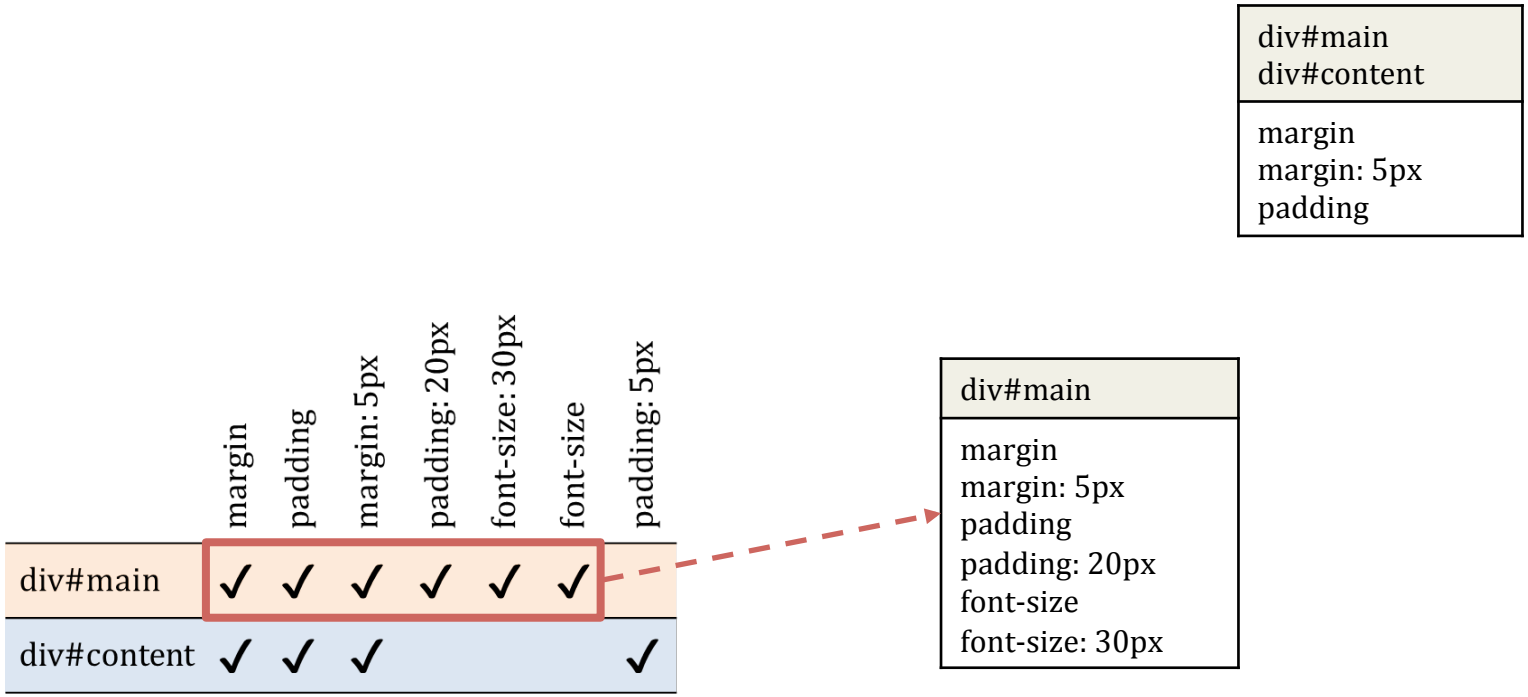
- Searching for maximal submatrix of elements
  - no other column contains a check in all the highlighted rows
  - no other row contains a check in both of the highlighted columns

	margin	padding	margin: 5px	padding: 20px	font-size: 30px	font-size	padding: 5px
div#main	✓	✓	✓	✓	✓	✓	
div#content	✓	✓	✓				✓

Step 2 : identifying formal concepts

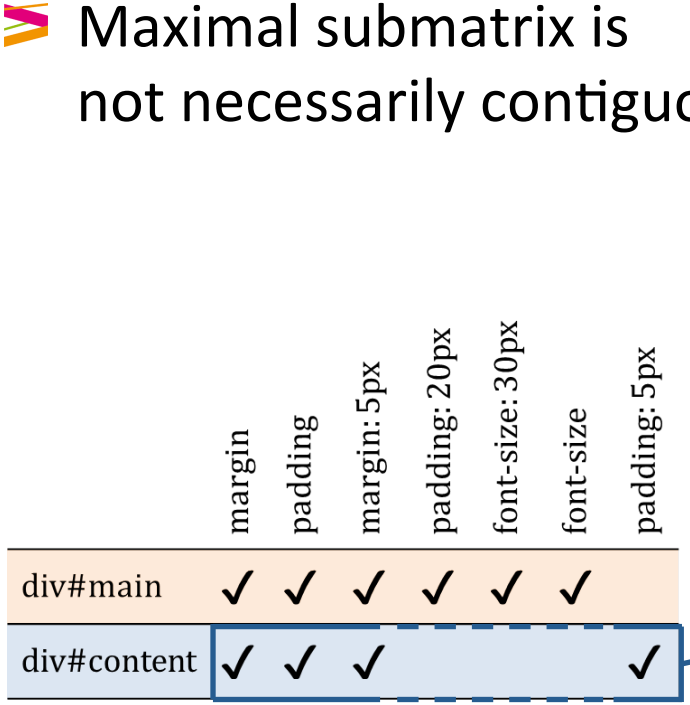


Step 2 : identifying formal concepts



Step 2 : identifying formal concepts

- Maximal submatrix is not necessarily contiguous



	margin	padding	margin: 5px	padding: 20px	font-size: 30px	font-size	padding: 5px
div#main	✓	✓	✓	✓	✓	✓	
div#content	✓	✓	✓				✓

div#main div#content
margin margin: 5px padding

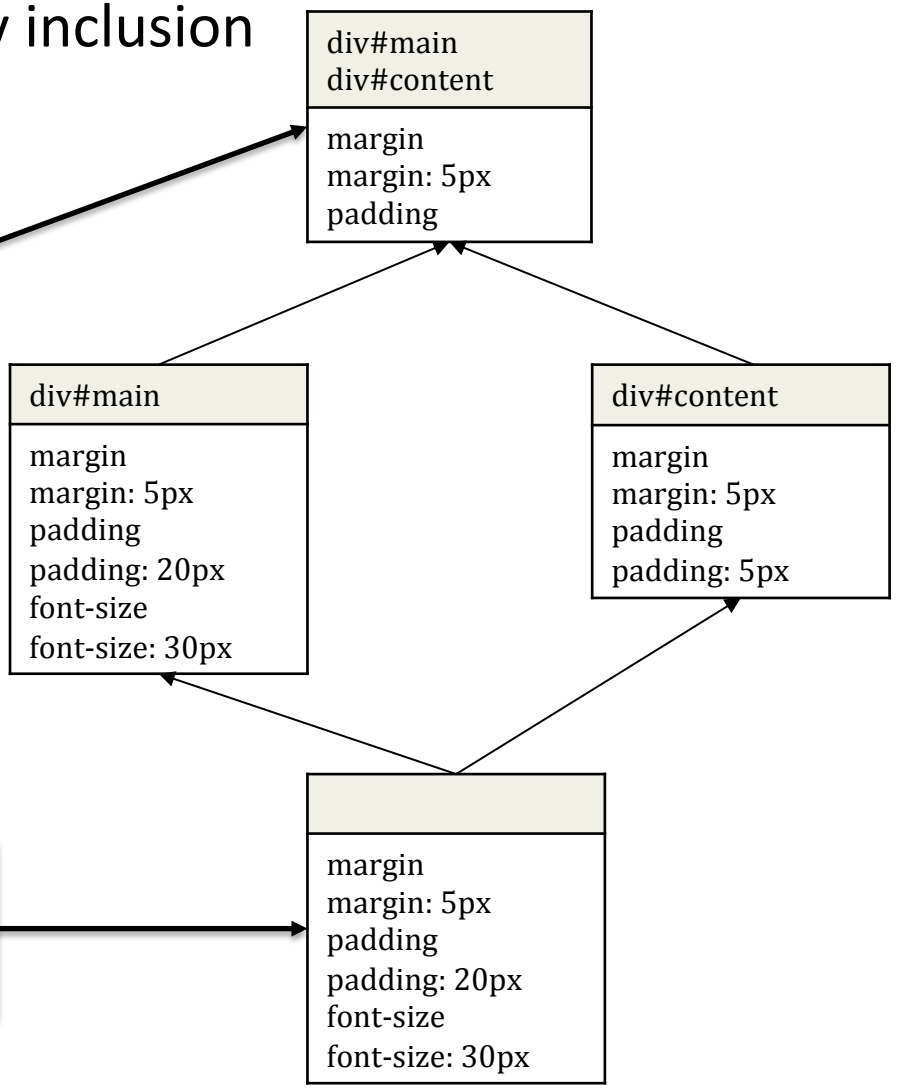
div#main
margin margin: 5px padding padding: 20px font-size font-size: 30px

div#content
margin margin: 5px padding padding: 5px

Step 2 : Concept lattice

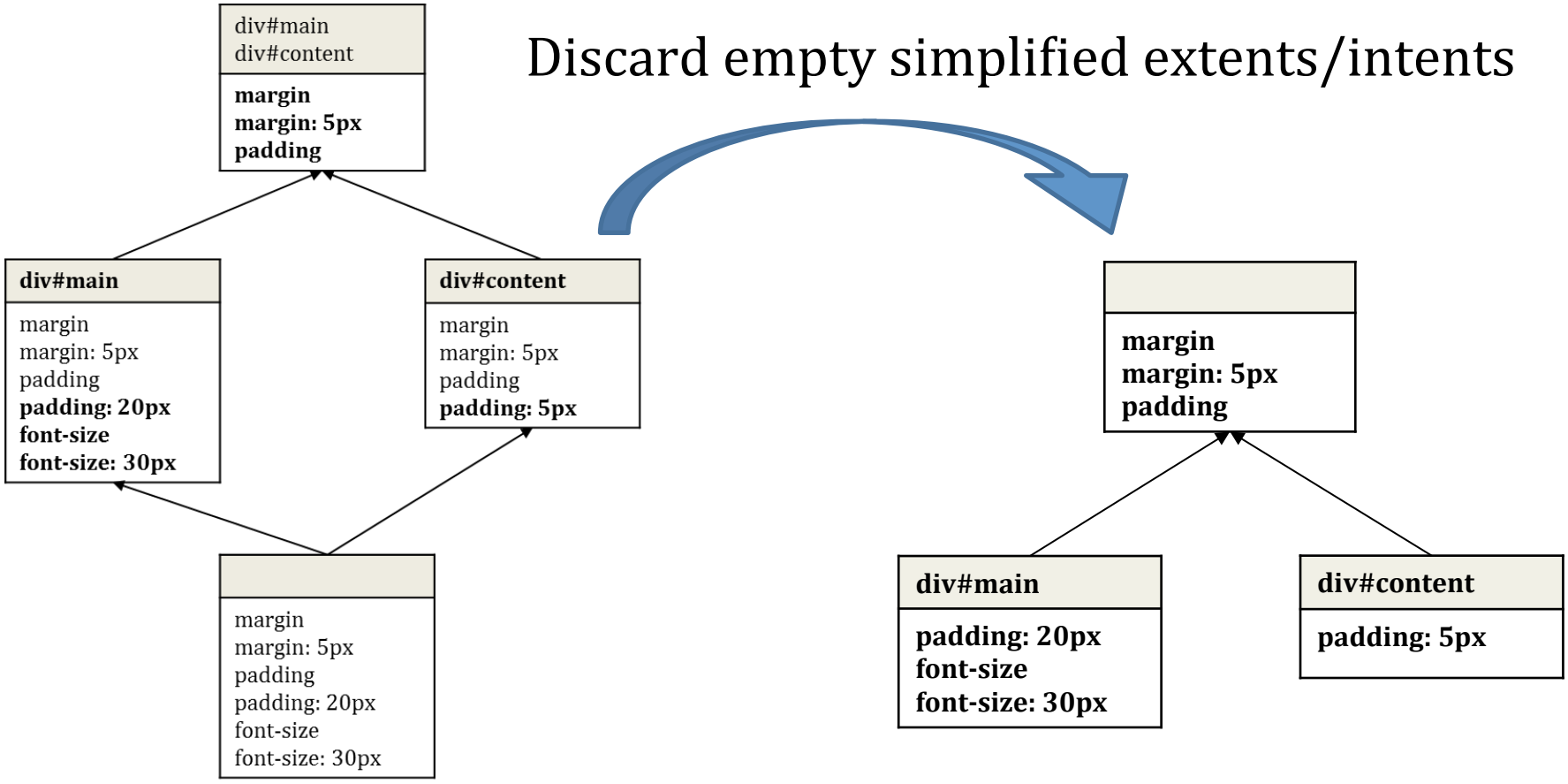
- Concepts are partially ordered by inclusion

Top concept : attributes shared by all the objects



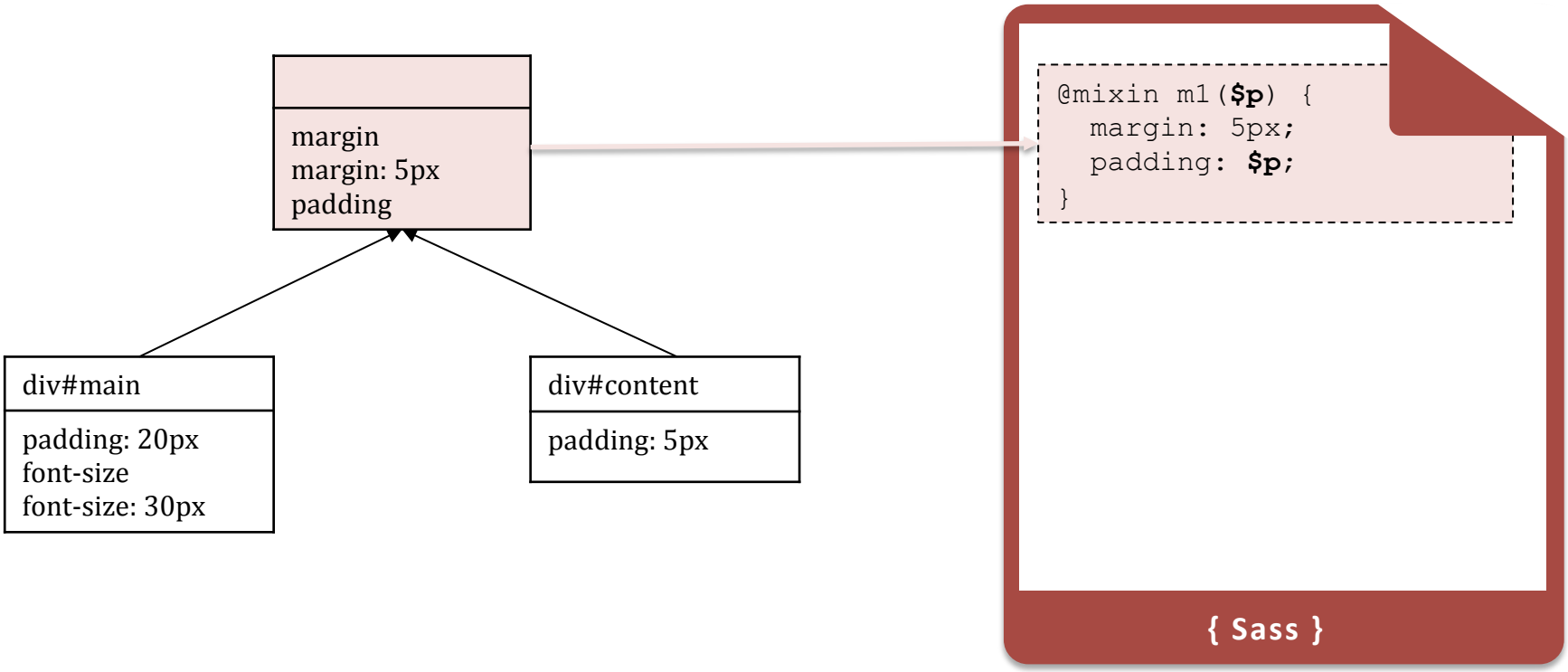
Bottom concept : objects that have all the attributes

Step 3: Galois sub-hierarchy

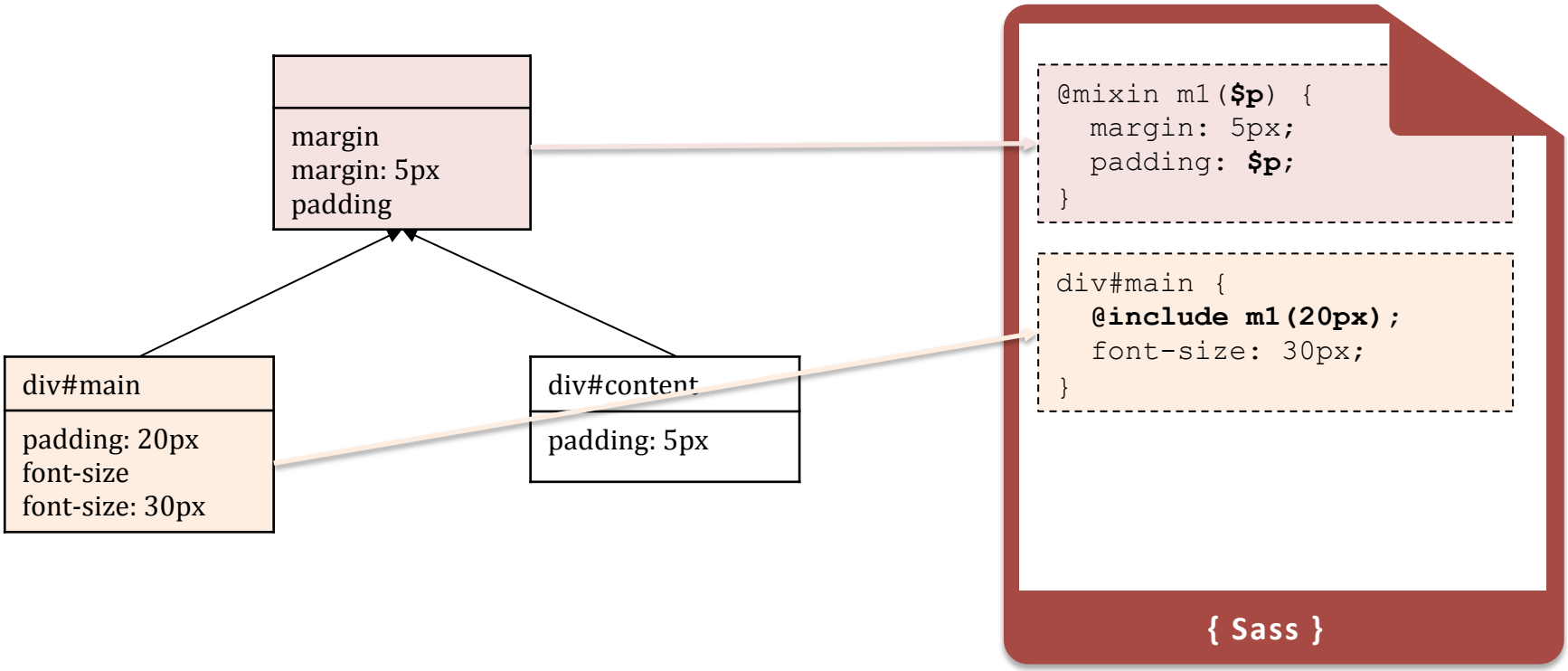


Simplified extent : objects not in the extents of subconcepts
Simplified intent : attributes not in the intents of supconcepts

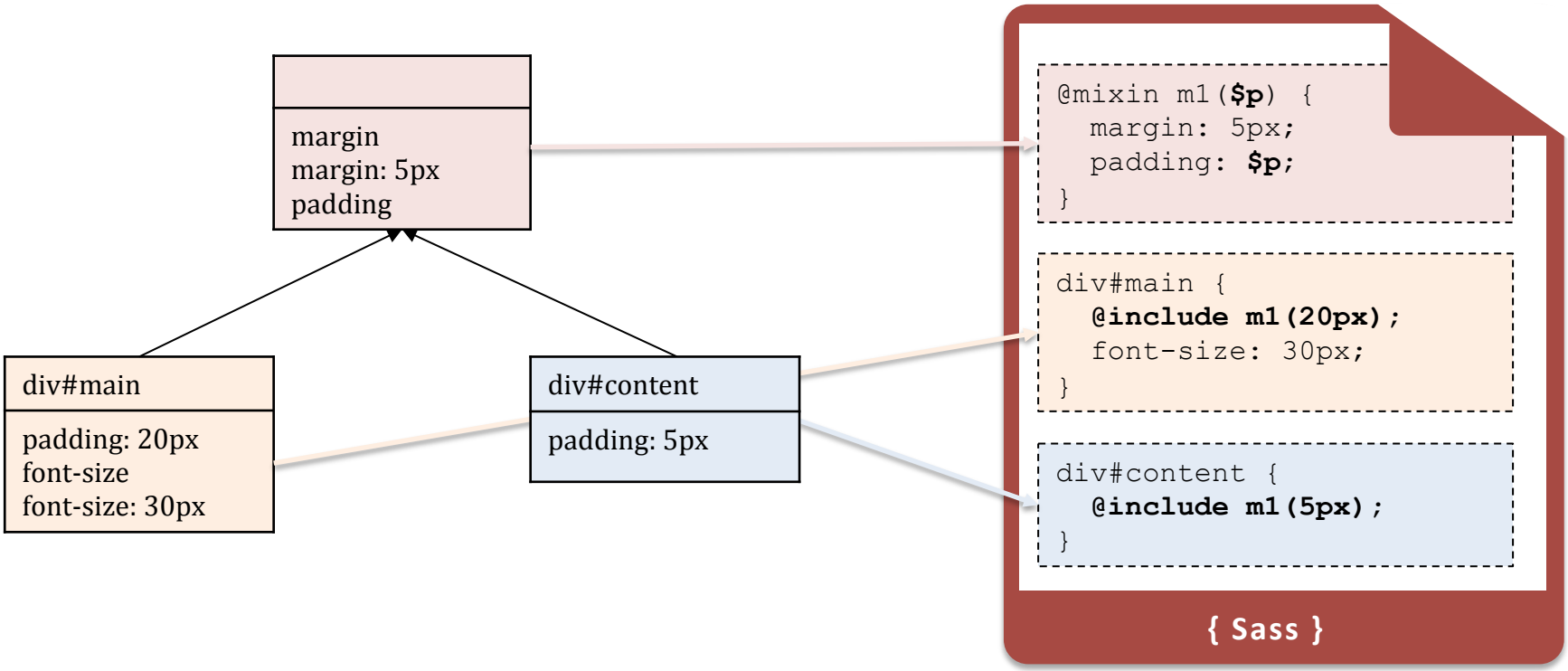
Step 4: Generation of Saas



Step 4: Generation of Saas



Step 4: Generation of Saas



Implementation

- Tooling support : mocss
  - https://github.com/acharpen/mocss
  - Publicaly available and open-source
- Fine-grained control on the generated code
  - Filtering of mixins creation based on user-defined criteria
    - parameters, declarations, usage sites, …
  - Grouping common properties
  - Preserving semantics
- Evaluation
  - 108 CCS files from 38 real-world web applications
  - Performance : less than 2s (up to 11s in the worst case)
  - Relevance : case studies with four web developpers

Conclusion

- CSS
  - Core web technologies with HTML and JavaScript
  - Lack of supprot to avoir code duplication
- Migrating legacy code to pre-processors
  - Challenging, tedious and error-prone
- Our Contribution
  - Automatic identification and extraction of mixins
  - Tooling support for code refactoring

https://github.com/acharpen/mocss