CSS JIT: Optimizing CSS Selector Matching with Just-in-Time Compilation

Yusuke Suzuki

Twitter: @Constellation

<utatane.tea@gmail.com>

Outline

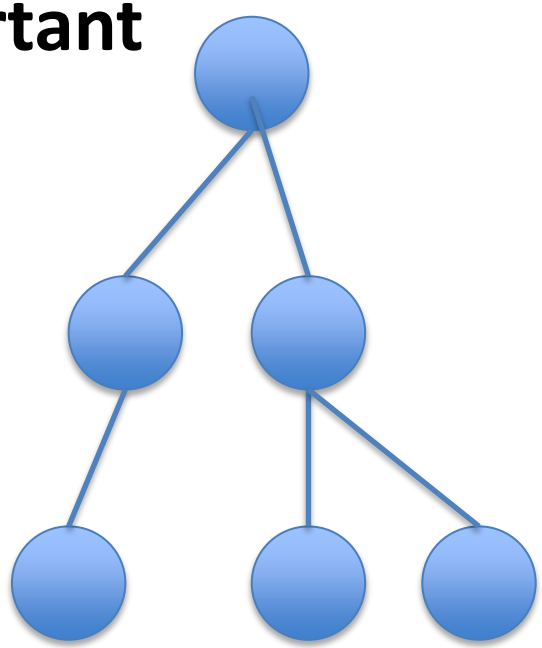
- Motivation & Goals
- Existing CSS Selector Implementation
- CSS Selector JIT
- Conclusion

Motivation

- CSS Selector is executed frequently
- Style Resolution / QuerySelector
  - For each node, check selector match/fail over node
- Speed of selector matching is **important** for Rendering / JS (Selectors API)

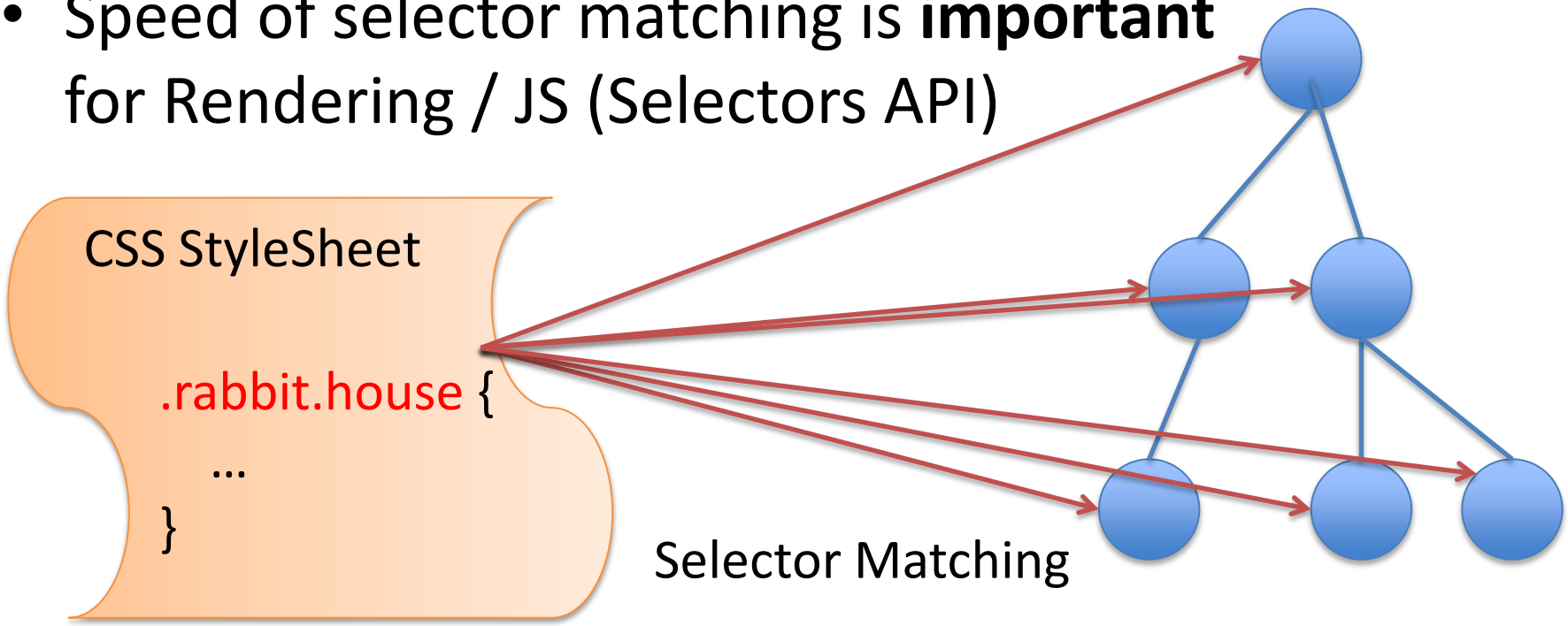
CSS StyleSheet

```
.rabbit.house {
    …
}
```



Motivation

- CSS Selector is executed frequently
- Style Resolution / QuerySelector
  - For each node, check selector match/fail over node
- Speed of selector matching is **important** for Rendering / JS (Selectors API)



QuerySelector implementation

- *root.querySelectorAll(/* selector */)*
  1. For each *element* under *root*
     1. Do selector matching with *selector* and *element* (SelectorChecker in WebKit / Blink)
     2. if matched
        1. *result.append(element)*
  2. Return *result*
- querySelectorAll executes selector matching the number of elements times

Goals

- Provide faster selector matching
  - Make style resolution faster
  - Make JS code using QuerySelector faster

```
    }

    void matchAllRules(bool matchAuthorAndUserStyles, bool includeSMILProperties);
    void matchUARules();
    void matchAuthorRules(bool includeEmptyRules);
    void matchUserRules(bool includeEmptyRules);

    void setMode(SelectorChecker::Mode mode) { m_mode = mode; }
    void setPseudoStyleRequest(const PseudoStyleRequest& request) { m_pseudoStyleRequest = request; }
    void setSameOriginOnly(bool f) { m_sameOriginOnly = f; }
    void setRegionForStyling(RenderRegion* regionForStyling) { m_regionForStyling = regionForStyling; }
    void setMedium(const MediaQueryEvaluator* medium) { m_isPrintStyle = medium->mediaTypeMatchSpecific("print");

    bool hasAnyMatchingRules(RuleSet*);

    StyleResolver::MatchResult& matchedResult();
    const Vector<RefPtr<StyleRule>>& matchedRuleList() const;
```

Outline

- Motivation & Goals
- Existing CSS Selector Implementation
- CSS Selector JIT
- Conclusion

Existing CSS Selector Implementation

- Highly tuned C++ implementation (WebKit/Blink)

- SelectorChecker::match
  - SelectorChecker::matchRecursively
    - SelectorChecker::checkOne
    - … recursively call markRecursively (backtracking is needed)

```
158
159 // Recursive check of selectors and combinators
160 // It can return 4 different values:
161 // * SelectorMatches          - the selector matches the element e
162 // * SelectorFailsLocally     - the selector fails for the element e
163 // * SelectorFailsAllSiblings - the selector fails for e and any sibling of e
164 // * SelectorFailsCompletely  - the selector fails for e and any sibling or ancestor of e
165 SelectorChecker::Match SelectorChecker::matchRecursively(const SelectorCheckingContext& context, PseudoId& dynamicPs
166 {
167     // The first selector has to match.
168     if (!checkOne(context))
169         return SelectorFailsLocally;
170
171     if (context.selector->m_match == CSSSelector::PseudoElement) {
172         if (context.selector->isCustomPseudoElement()) {
173             if (ShadowRoot* root = context.element->containingShadowRoot()) {
174                 if (context.element->shadowPseudoId() != context.selector->value())
```

CSS Selector Structure & Terminology

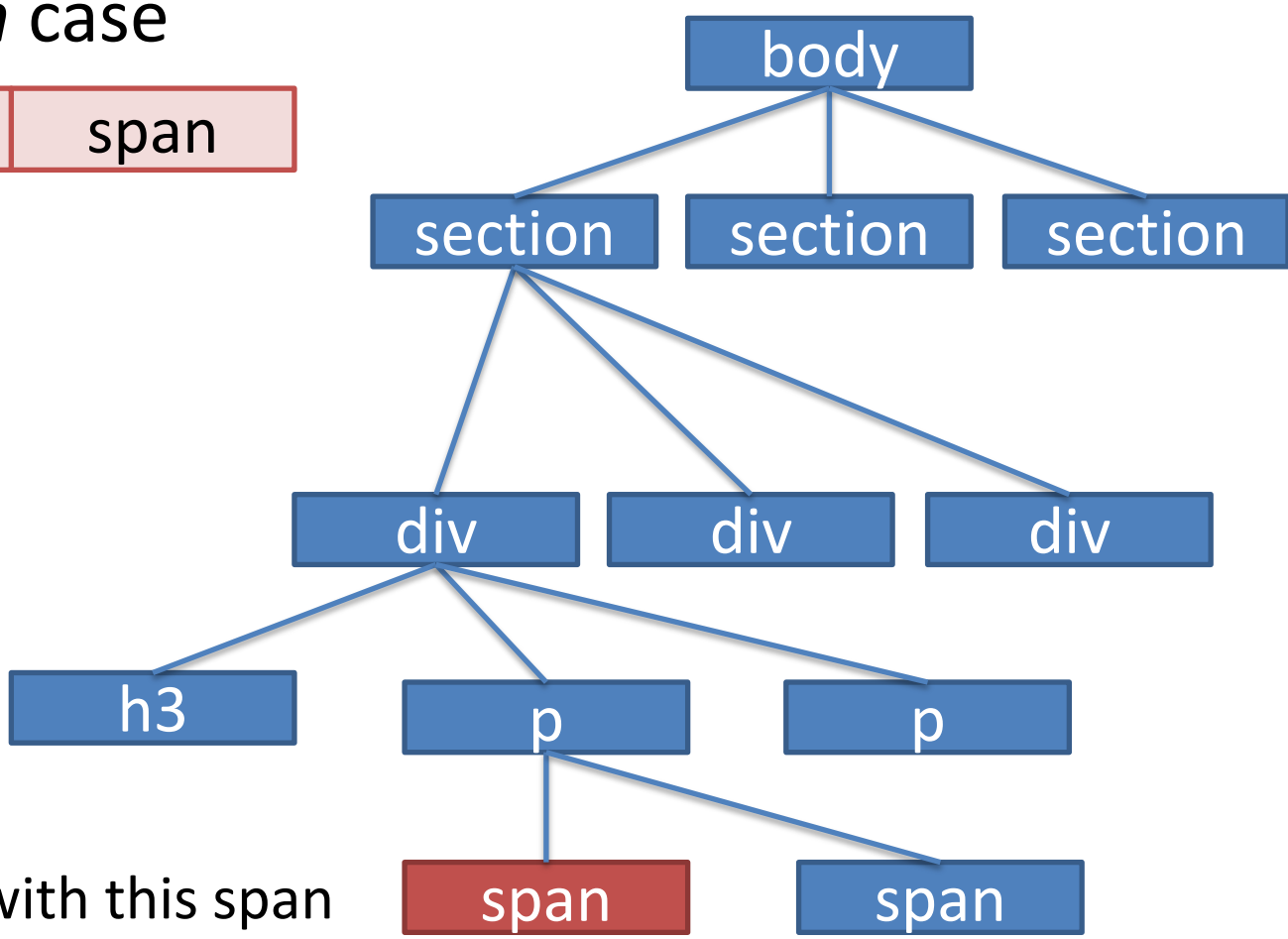
- Composed of *compound selectors*
  - *simple selector* (e.g. div) + *combinator* (e.g. >)
- Selector is evaluated **from-right-to-left**

compound selectors (div+ div~ div div> span)

http://dev.w3.org/csswg/selectors4/#structure

Evaluation Example

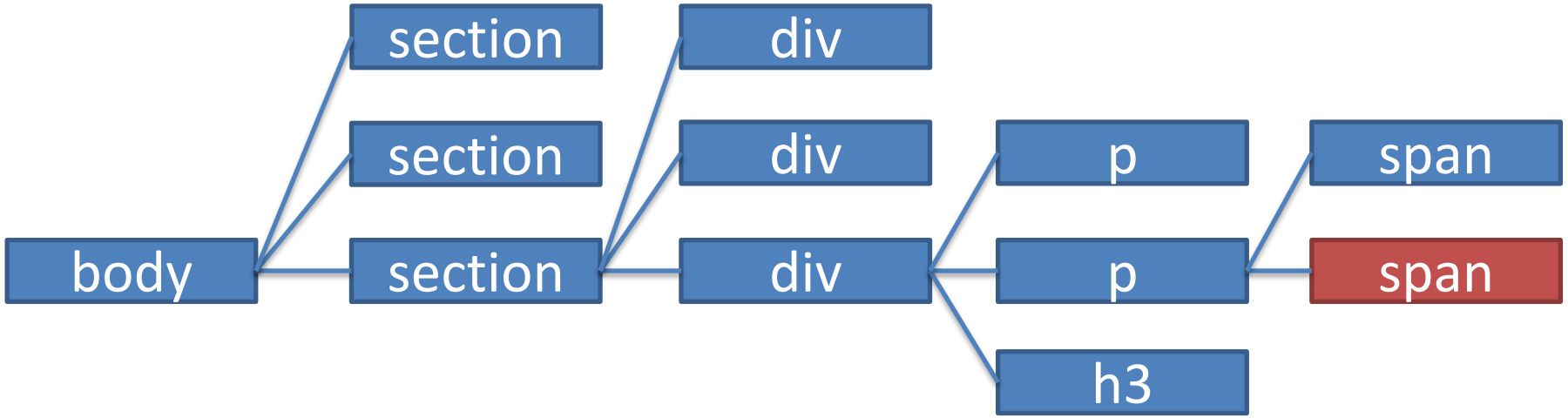
- *body div span* case



Matching starts with this span

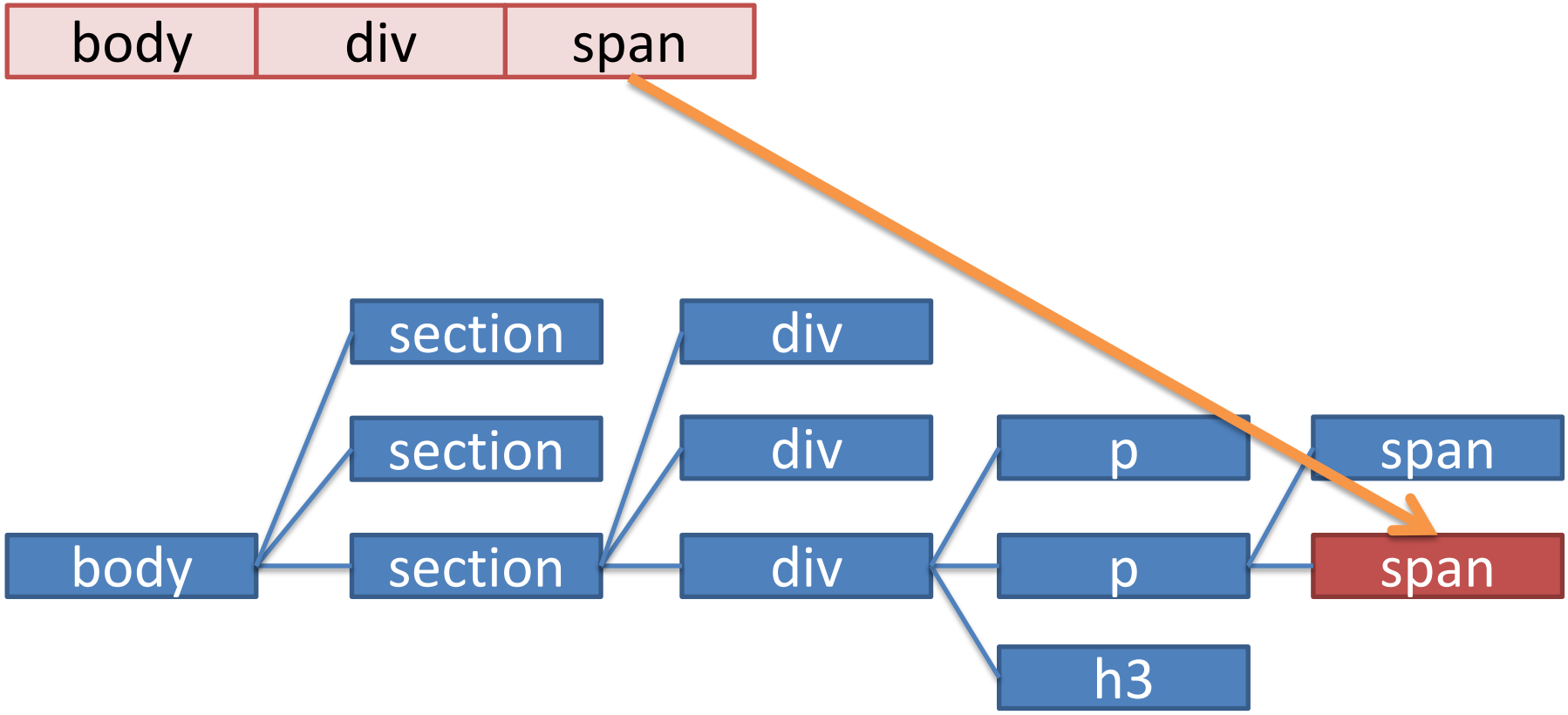
Evaluation Example: Matched

- *body div span* case



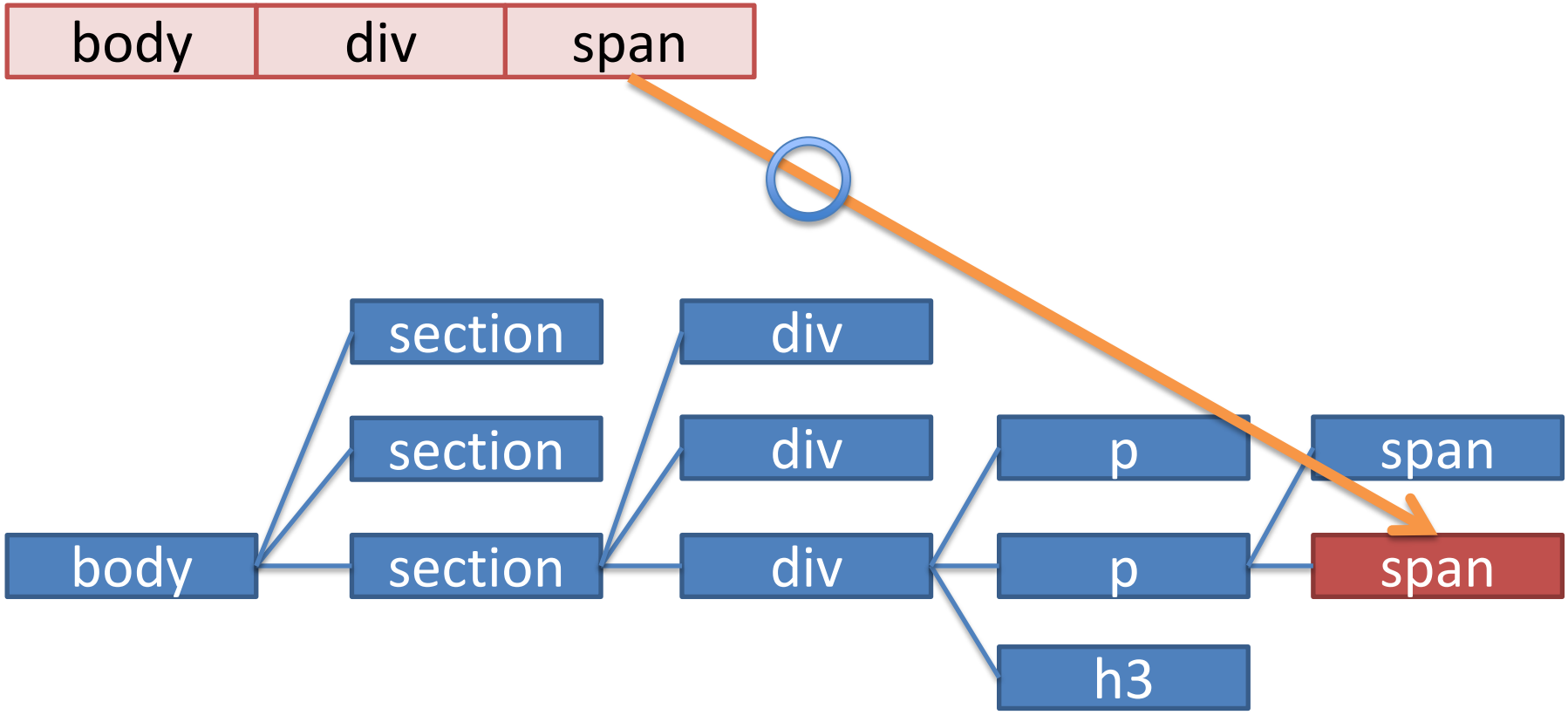
Evaluation Example: Matched

- *body div span* case



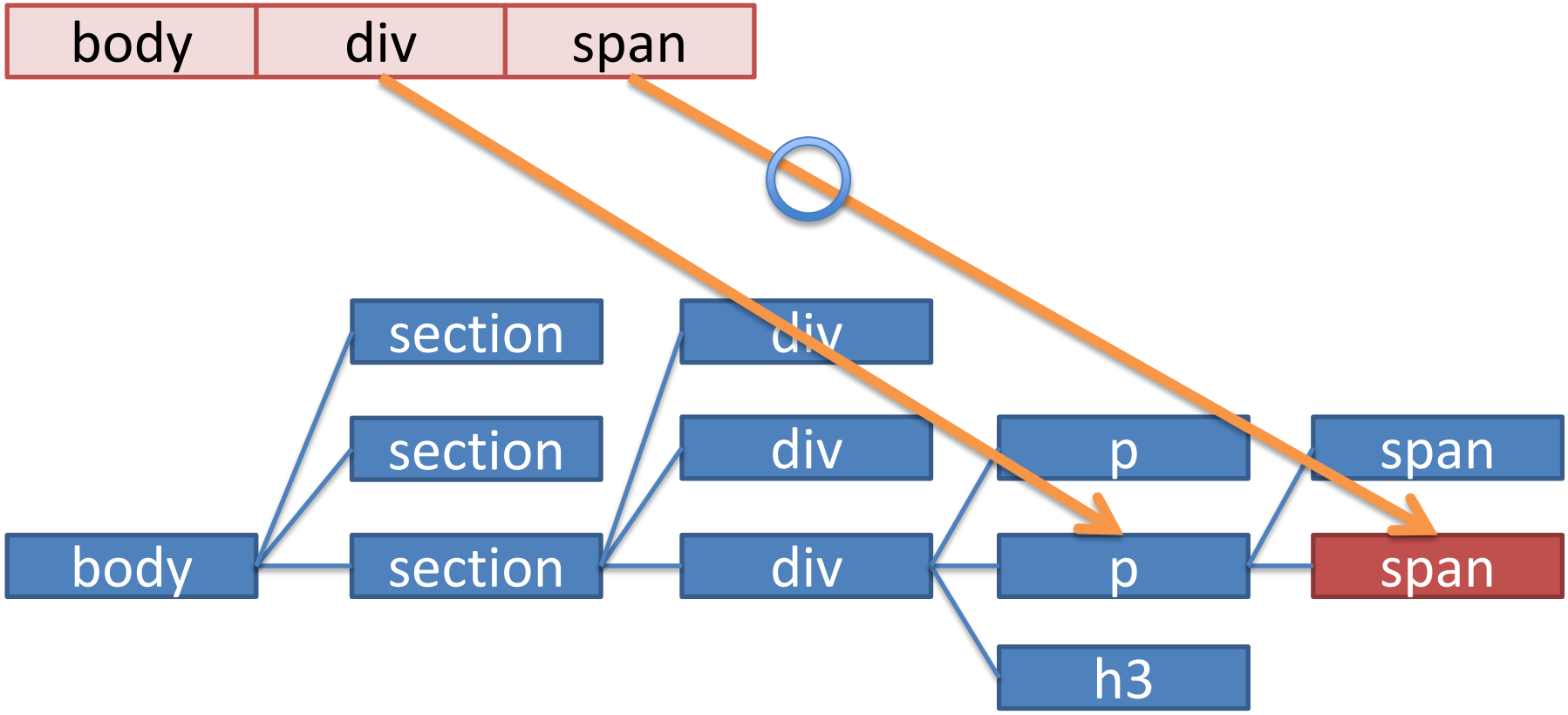
Evaluation Example: Matched

- *body div span* case



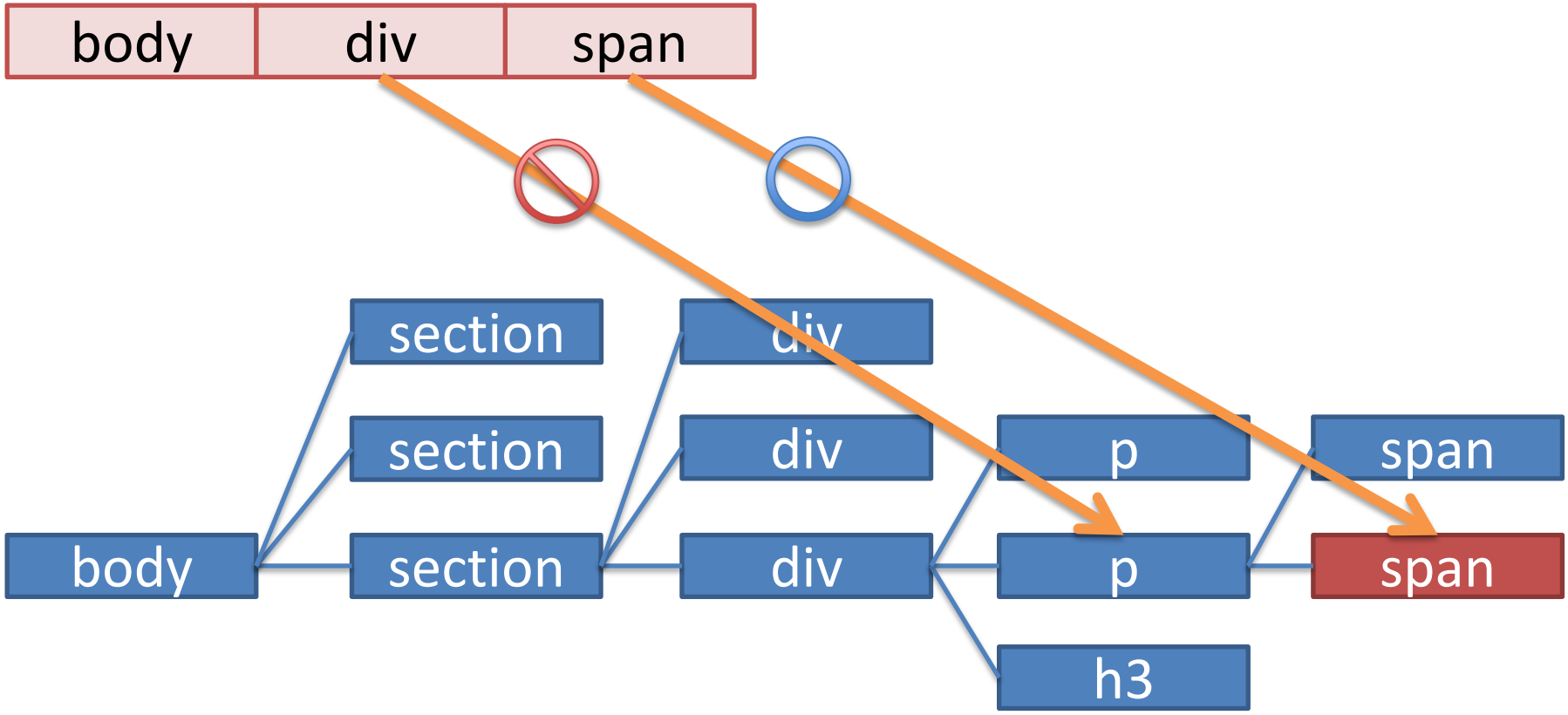
Evaluation Example: Matched

- *body div span* case



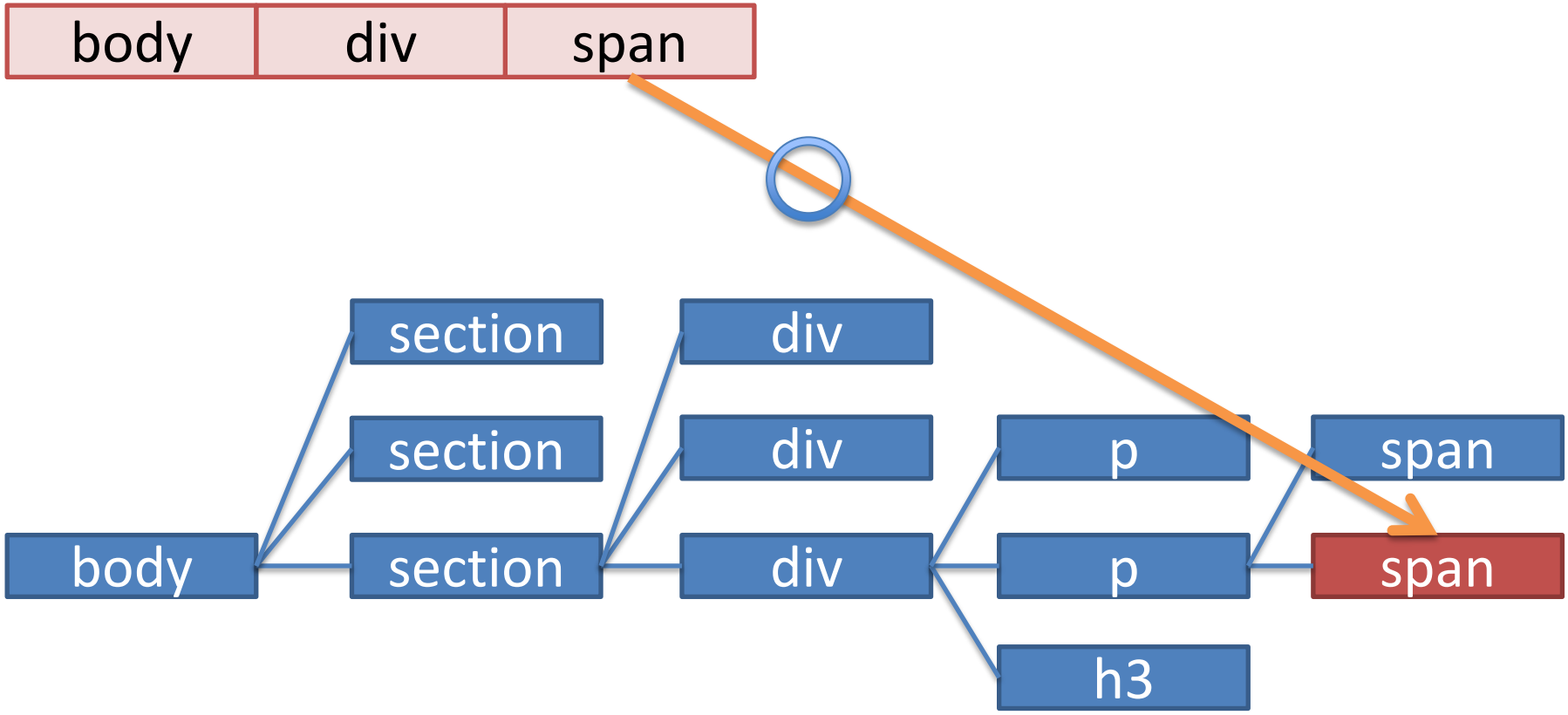
Evaluation Example: Matched

- *body div span* case



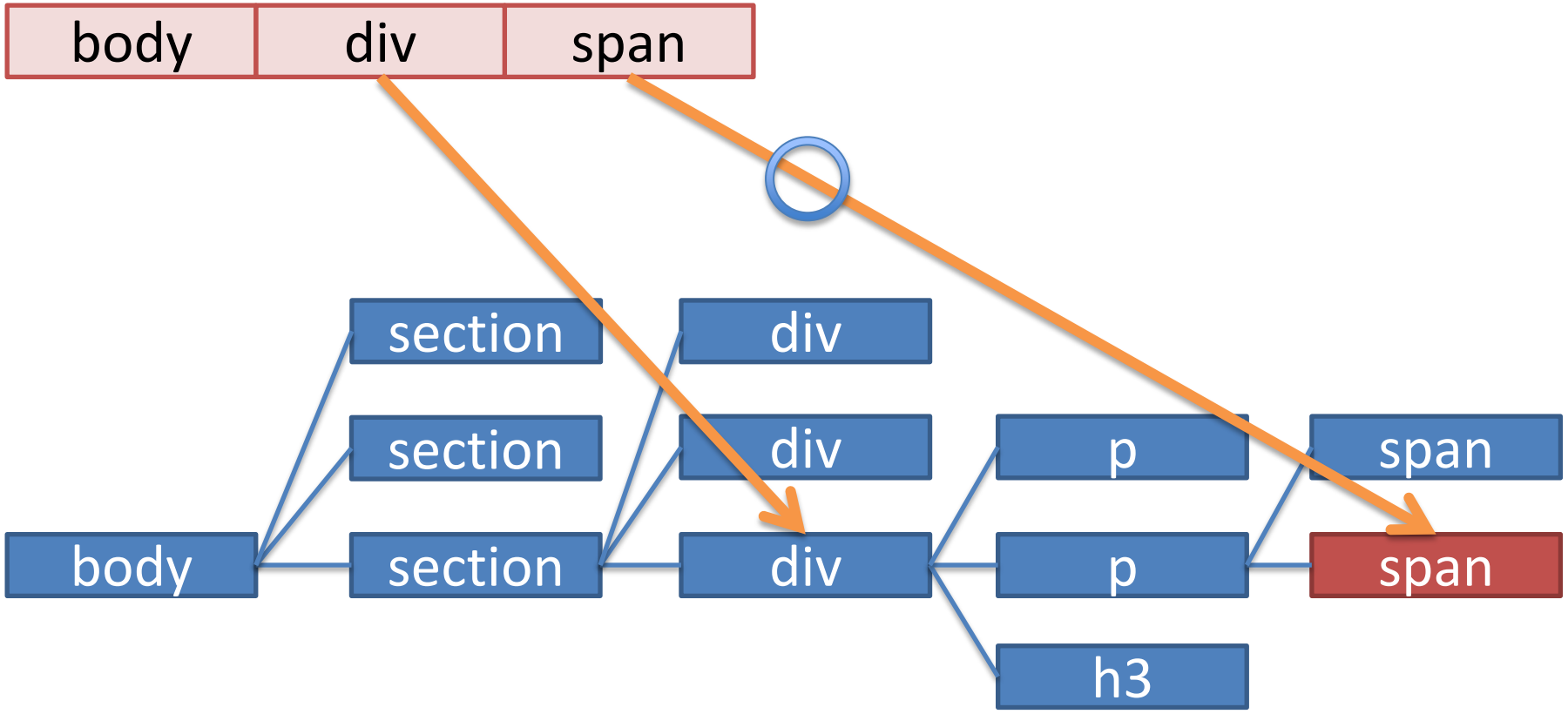
Evaluation Example: Matched

- *body div span* case



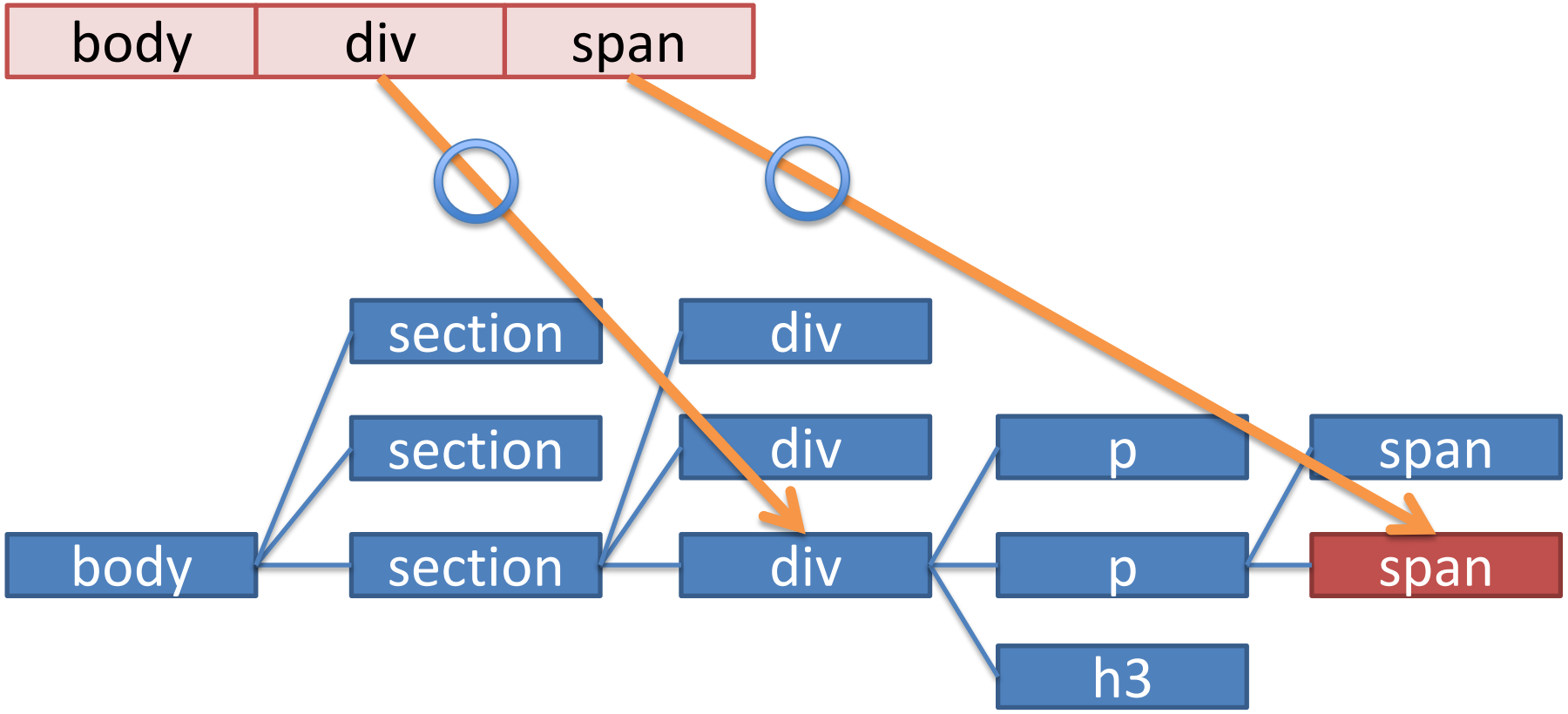
Evaluation Example: Matched

- *body div span* case



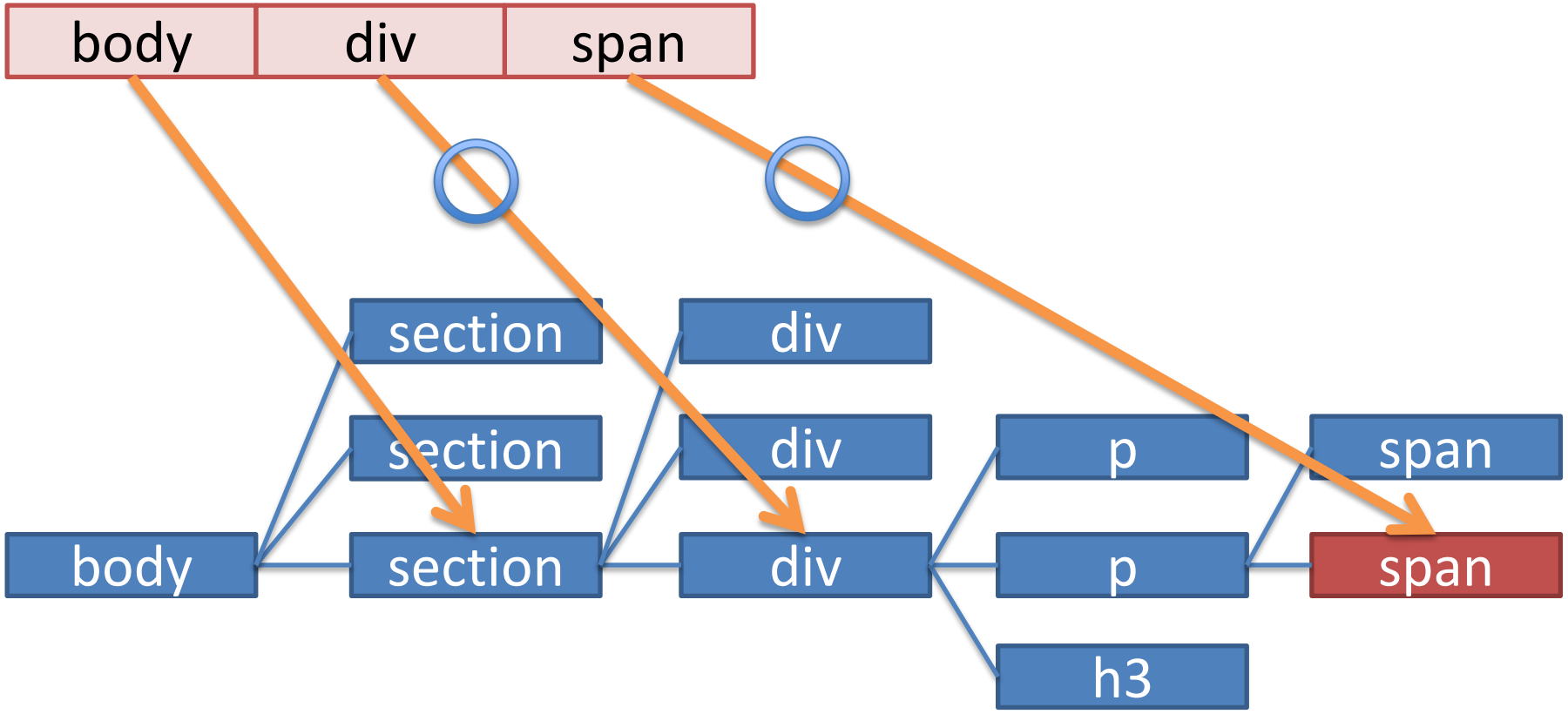
Evaluation Example: Matched

- *body div span* case



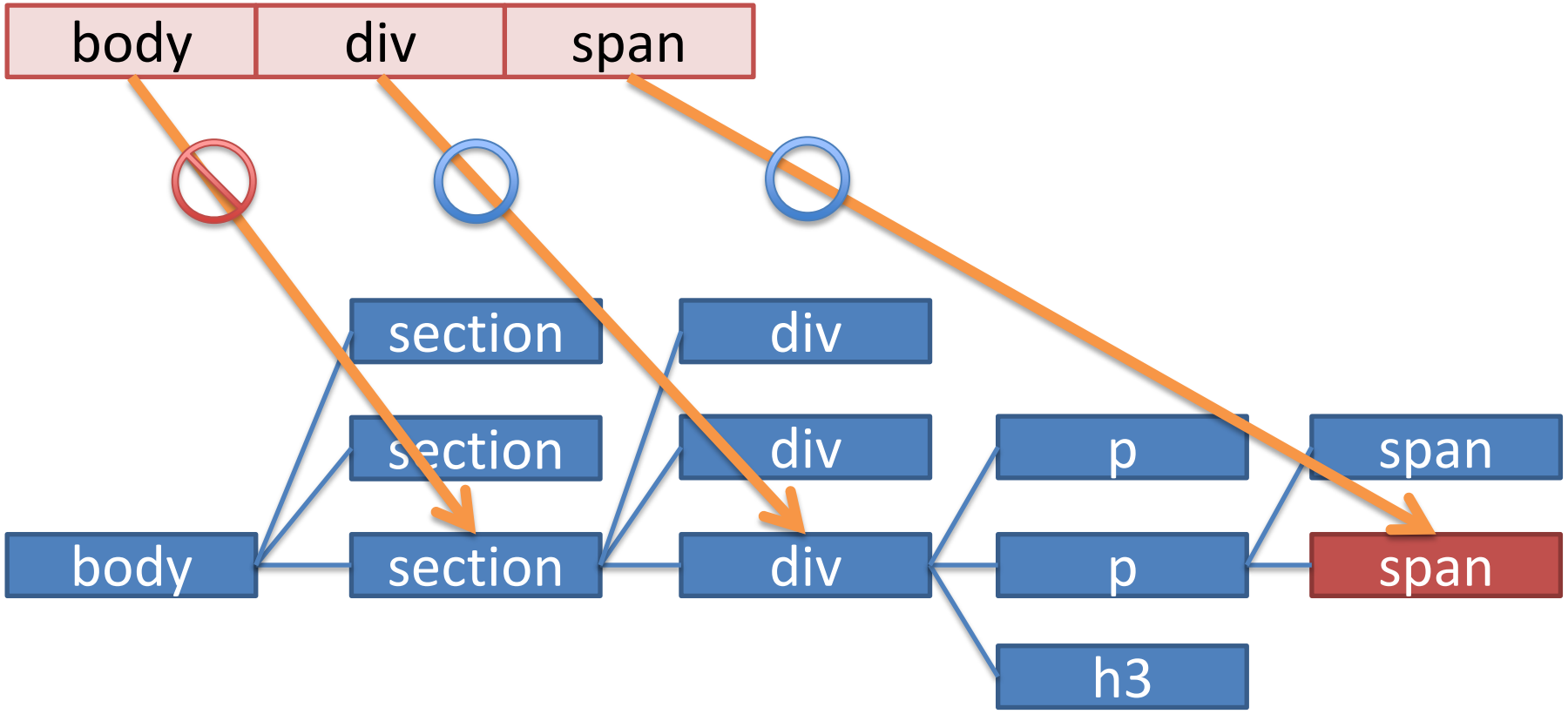
Evaluation Example: Matched

- *body div span* case



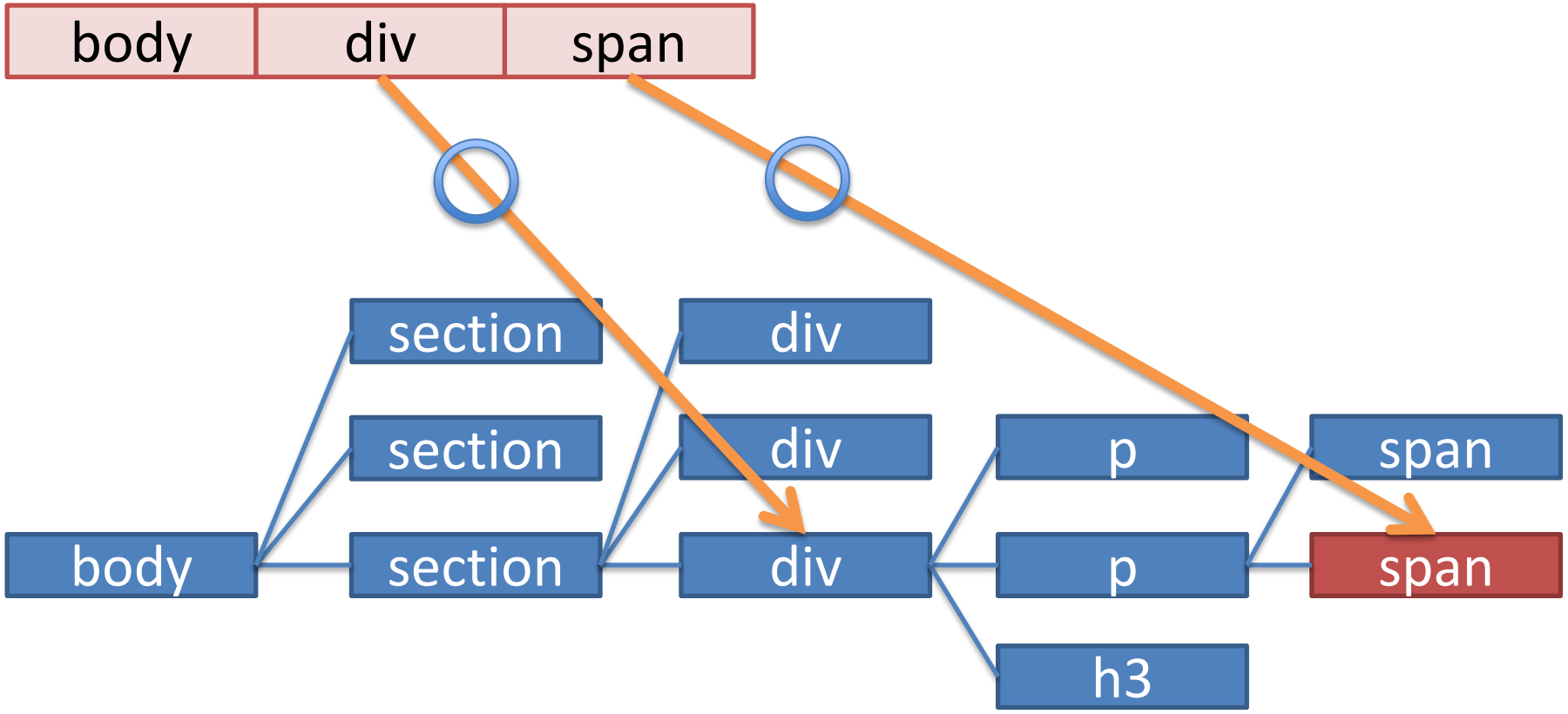
Evaluation Example: Matched

- *body div span* case



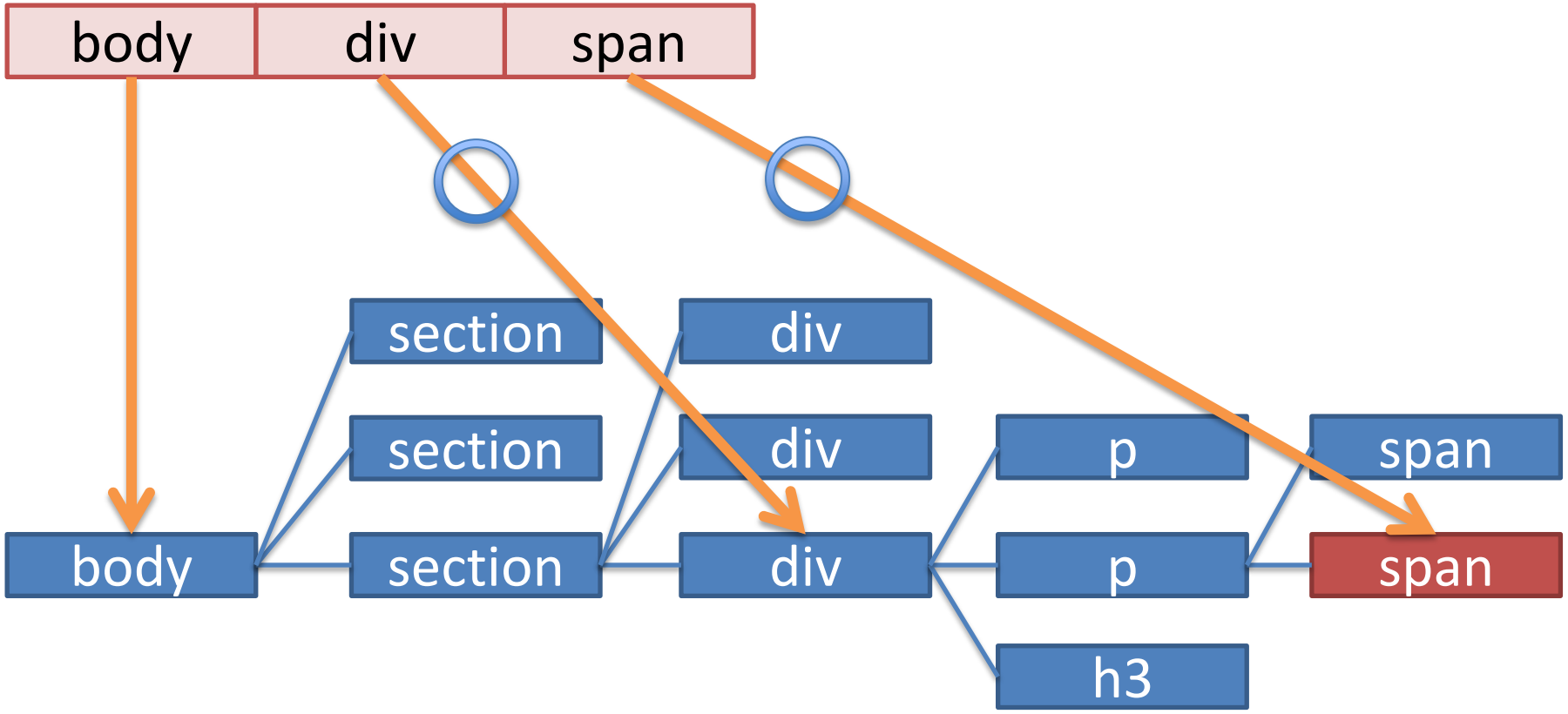
Evaluation Example: Matched

- *body div span* case



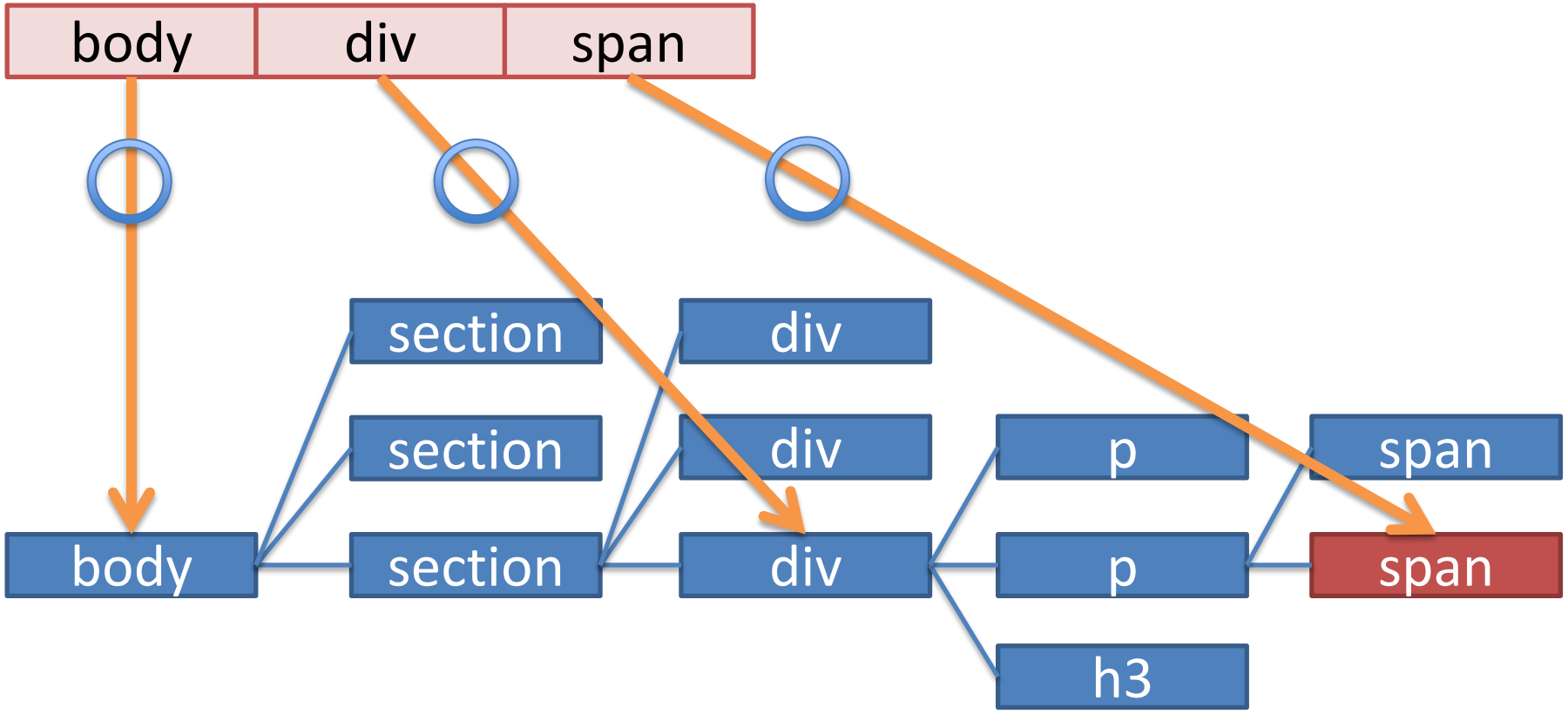
Evaluation Example: Matched

- *body div span* case



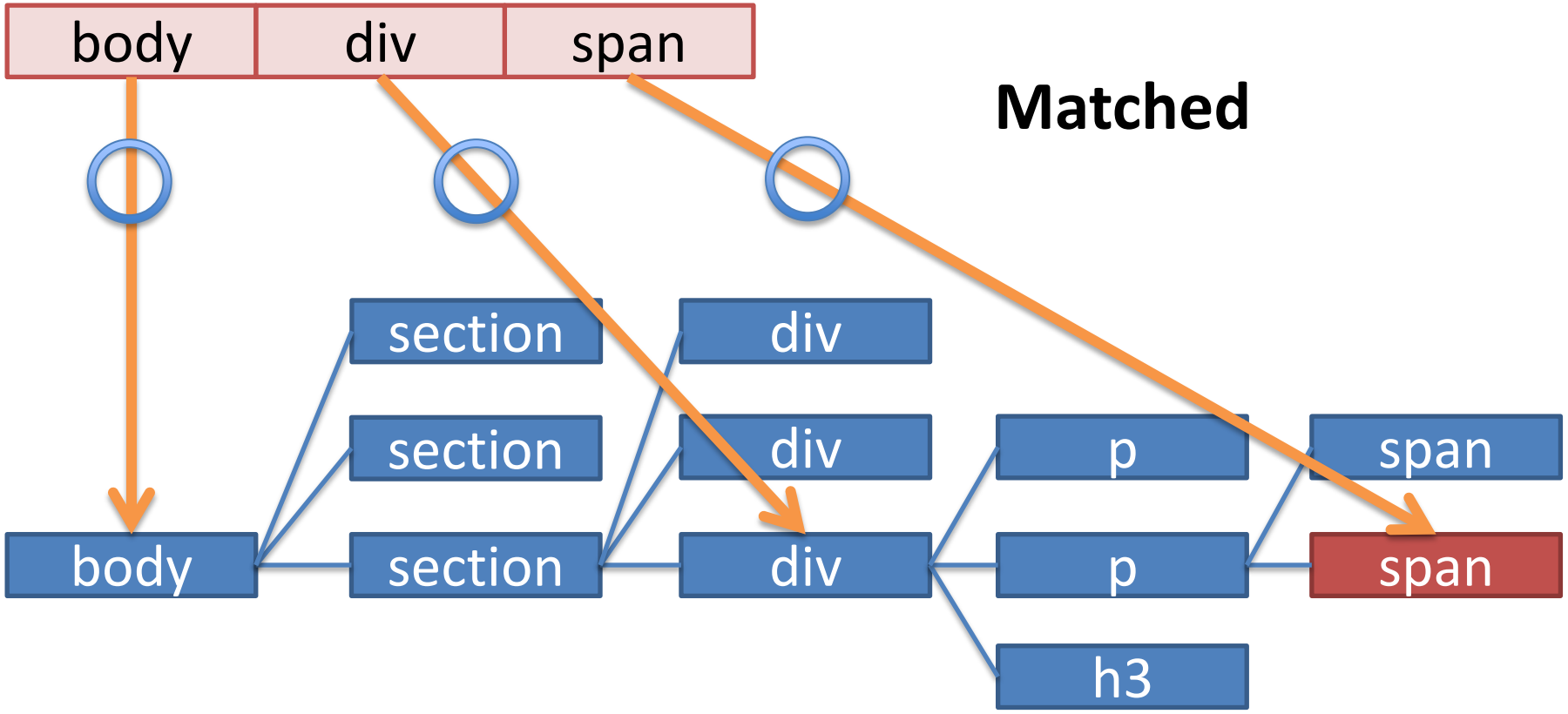
Evaluation Example: Matched

- *body div span* case



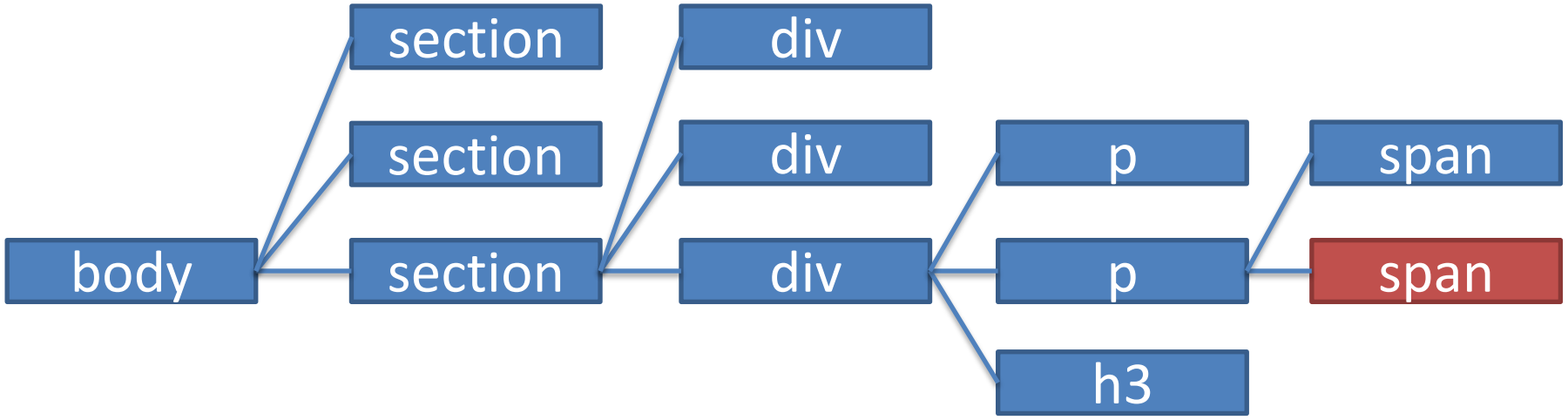
Evaluation Example: Matched

- *body div span* case



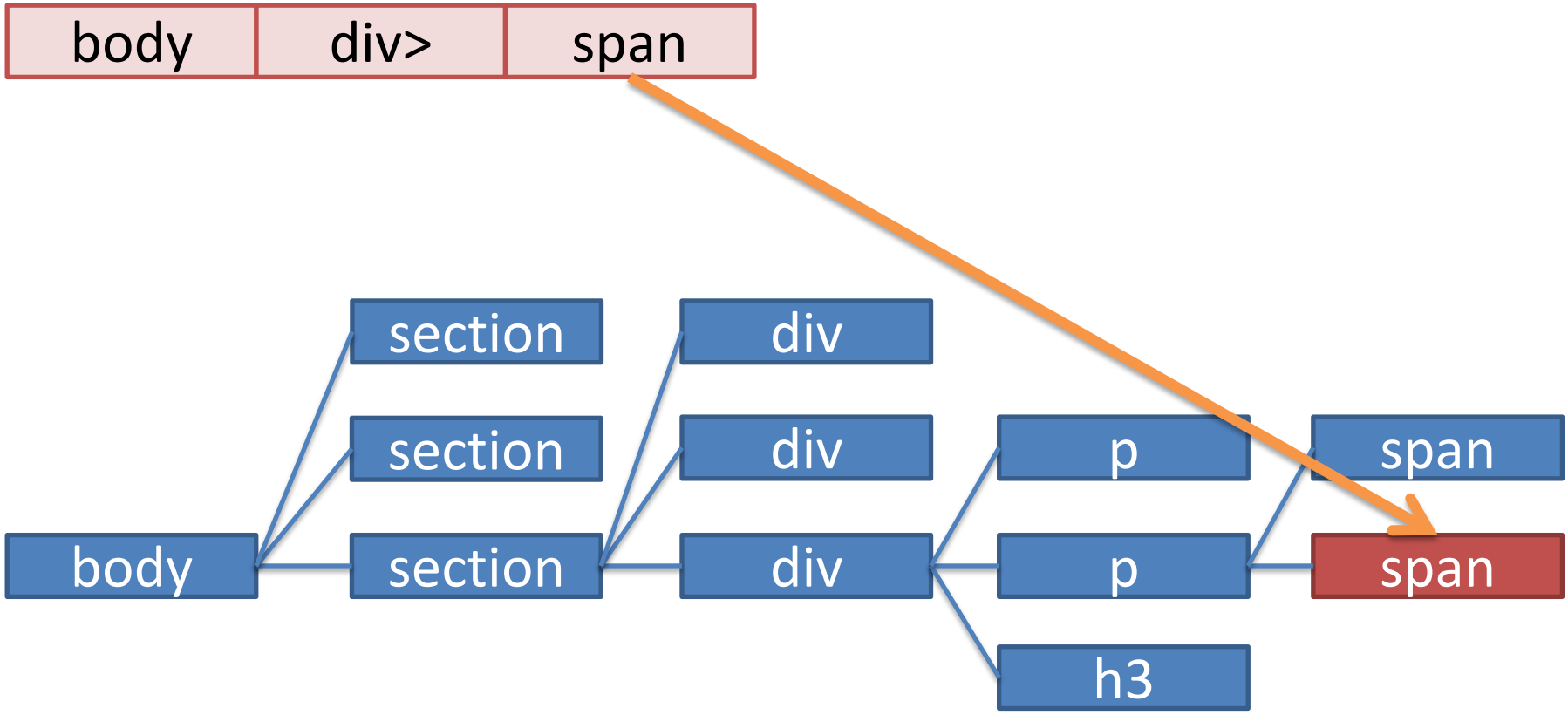
Evaluation Example: Failed

- *body div> span* case



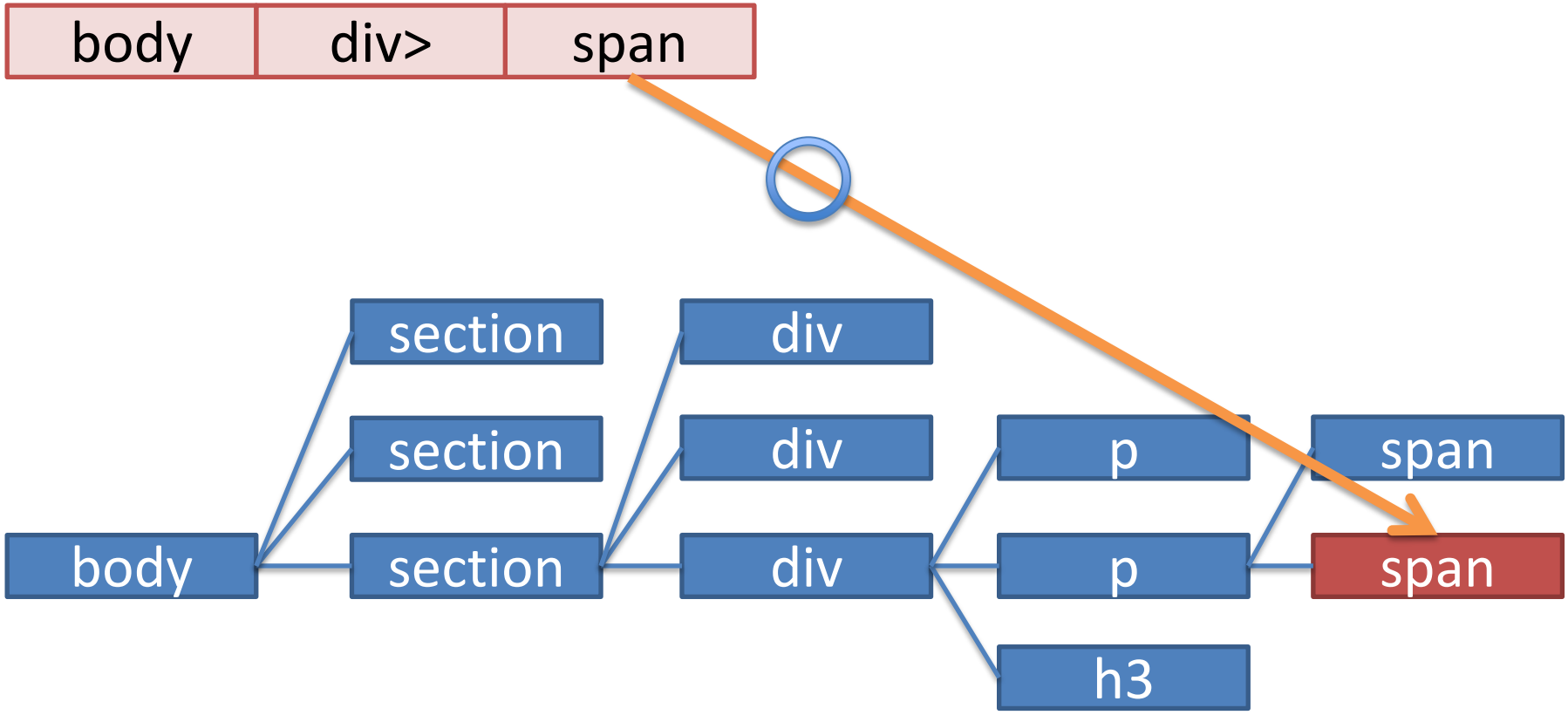
Evaluation Example: Failed

- *body div> span* case



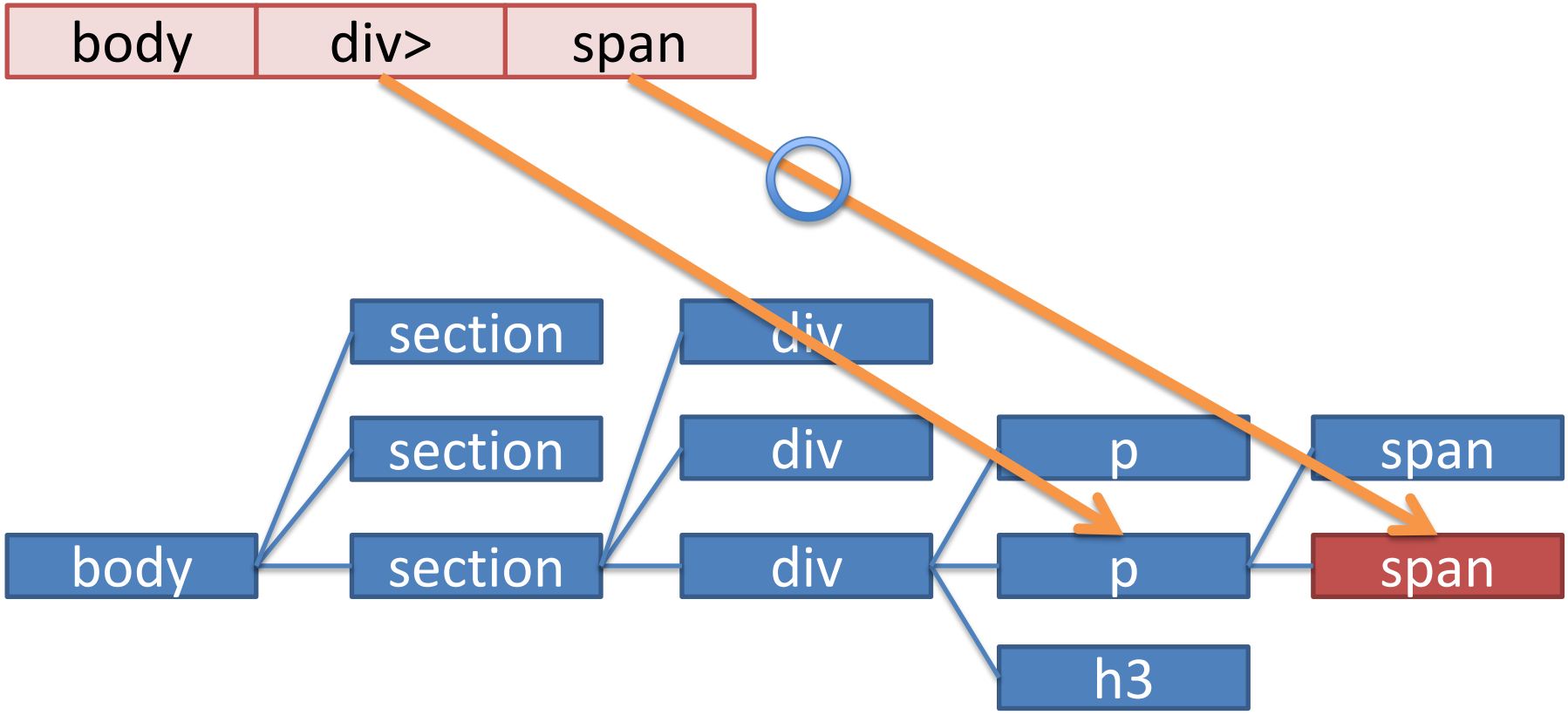
Evaluation Example: Failed

- *body div> span* case



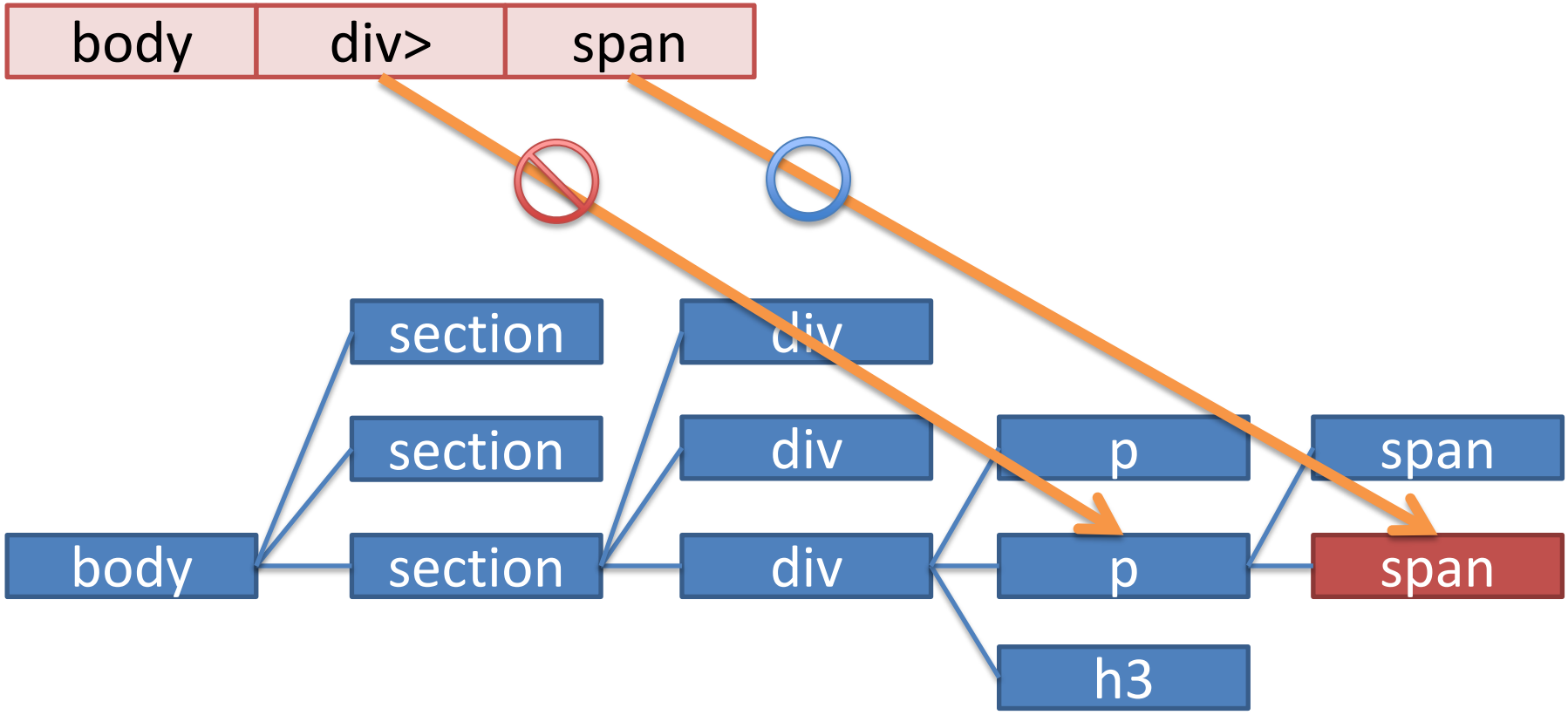
Evaluation Example: Failed

- *body div> span* case



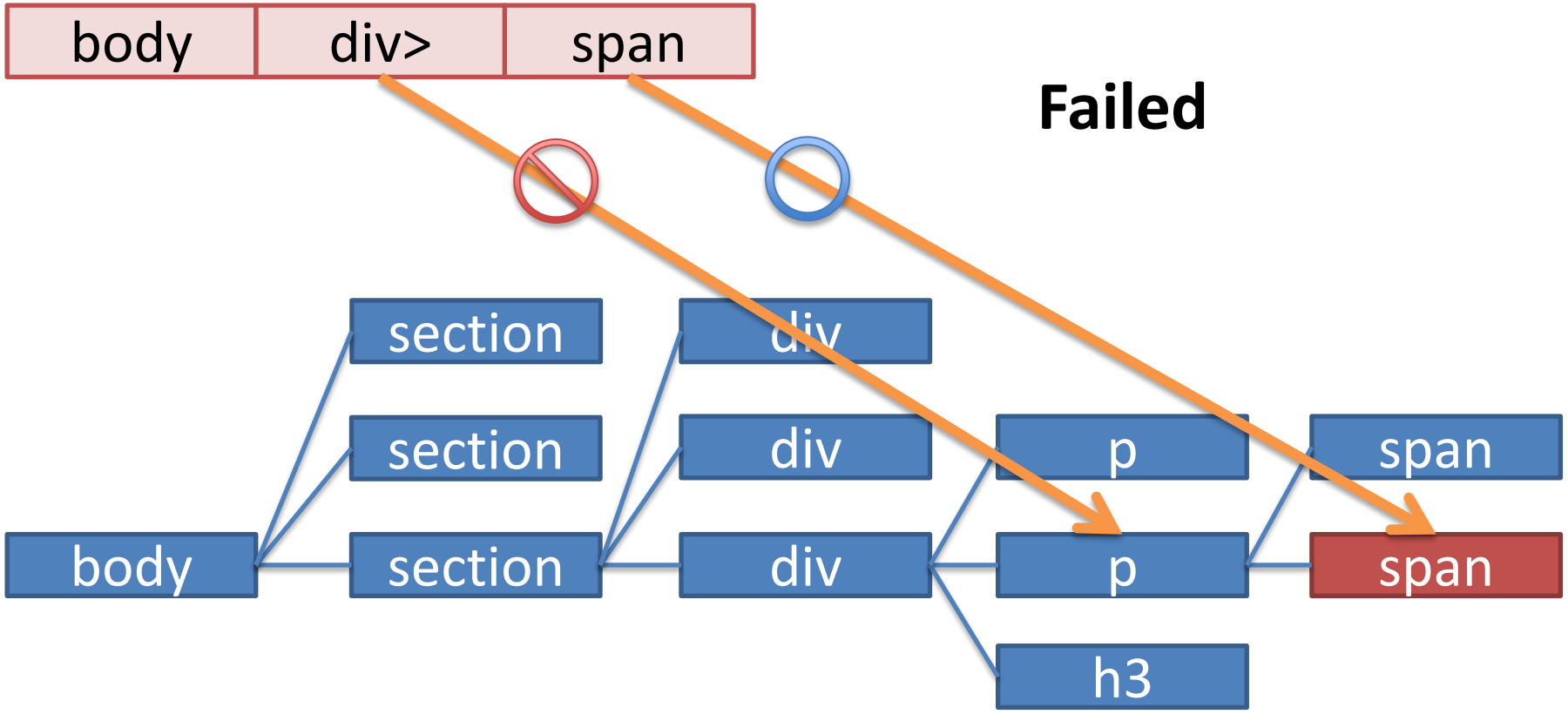
Evaluation Example: Failed

- *body div> span* case



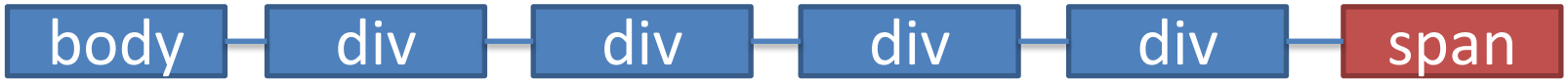
Evaluation Example: Failed

- *body div> span* case



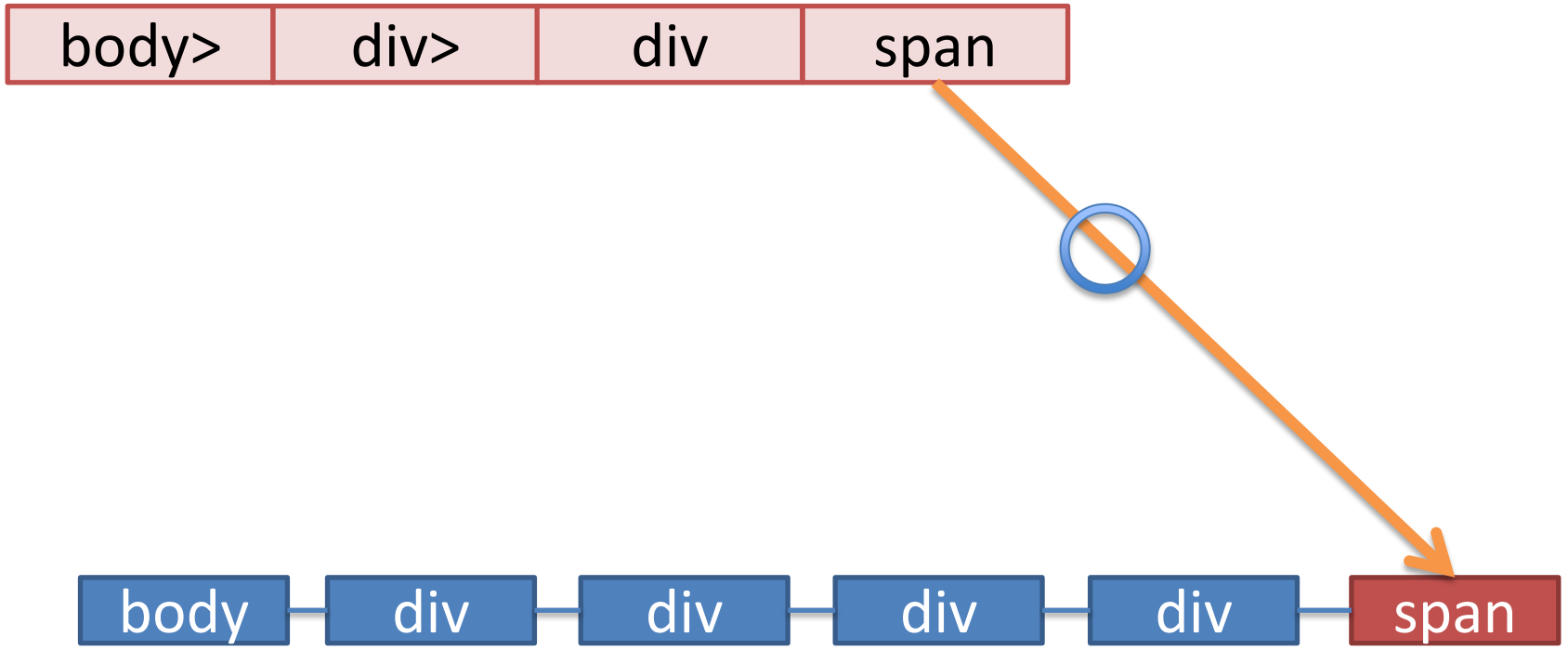
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



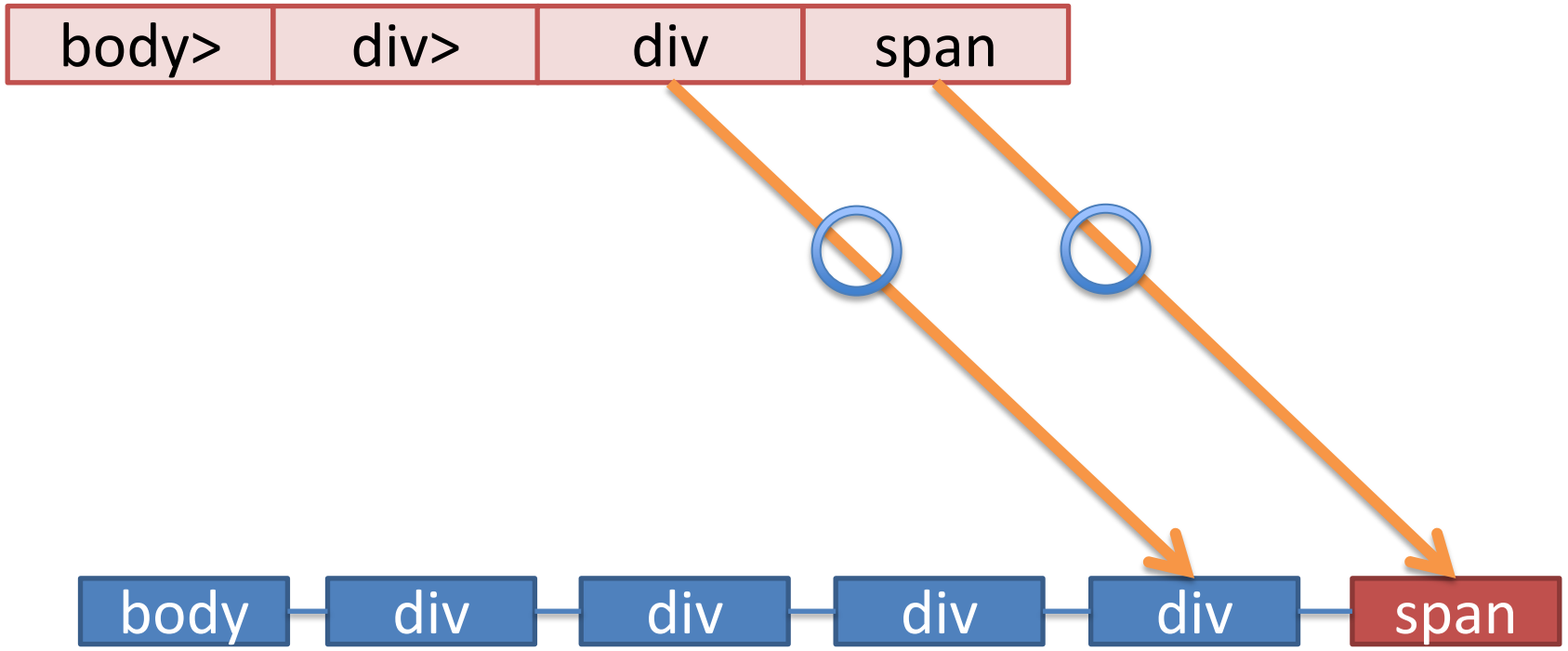
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



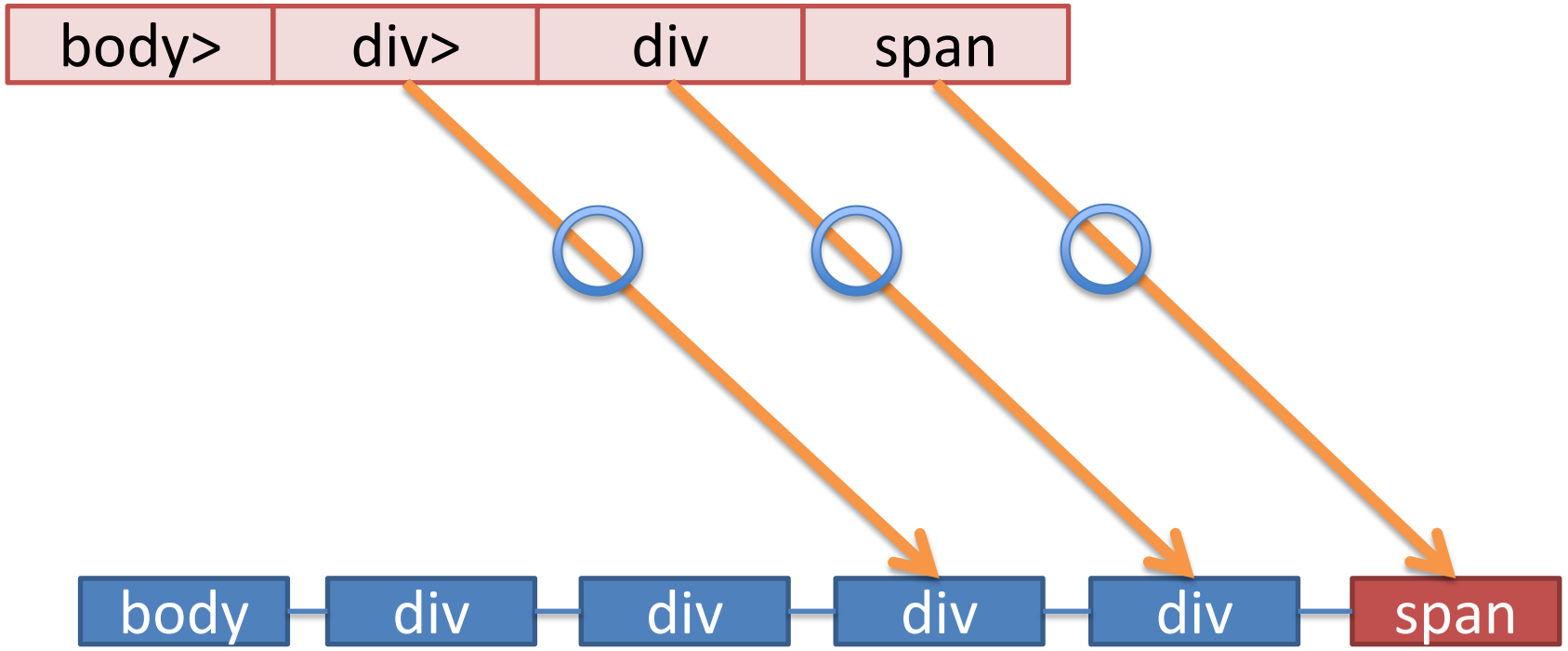
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



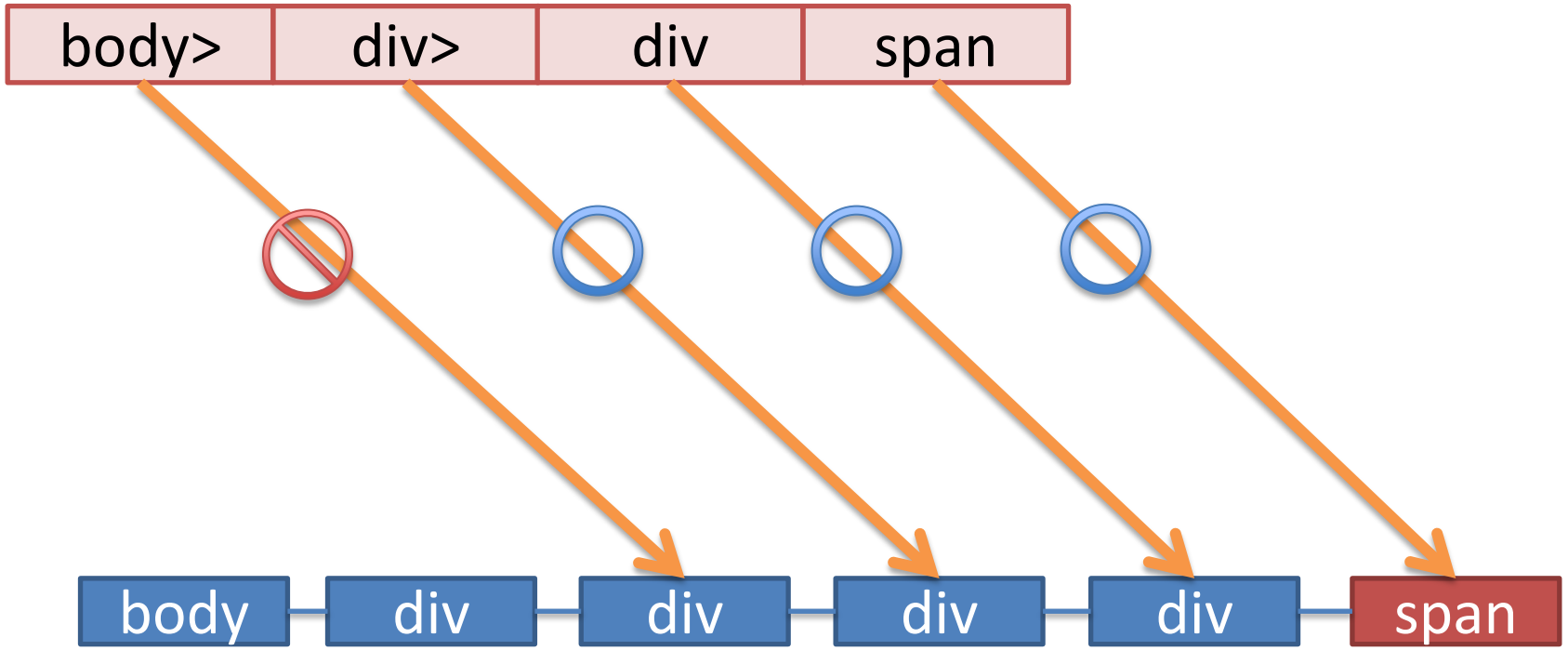
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



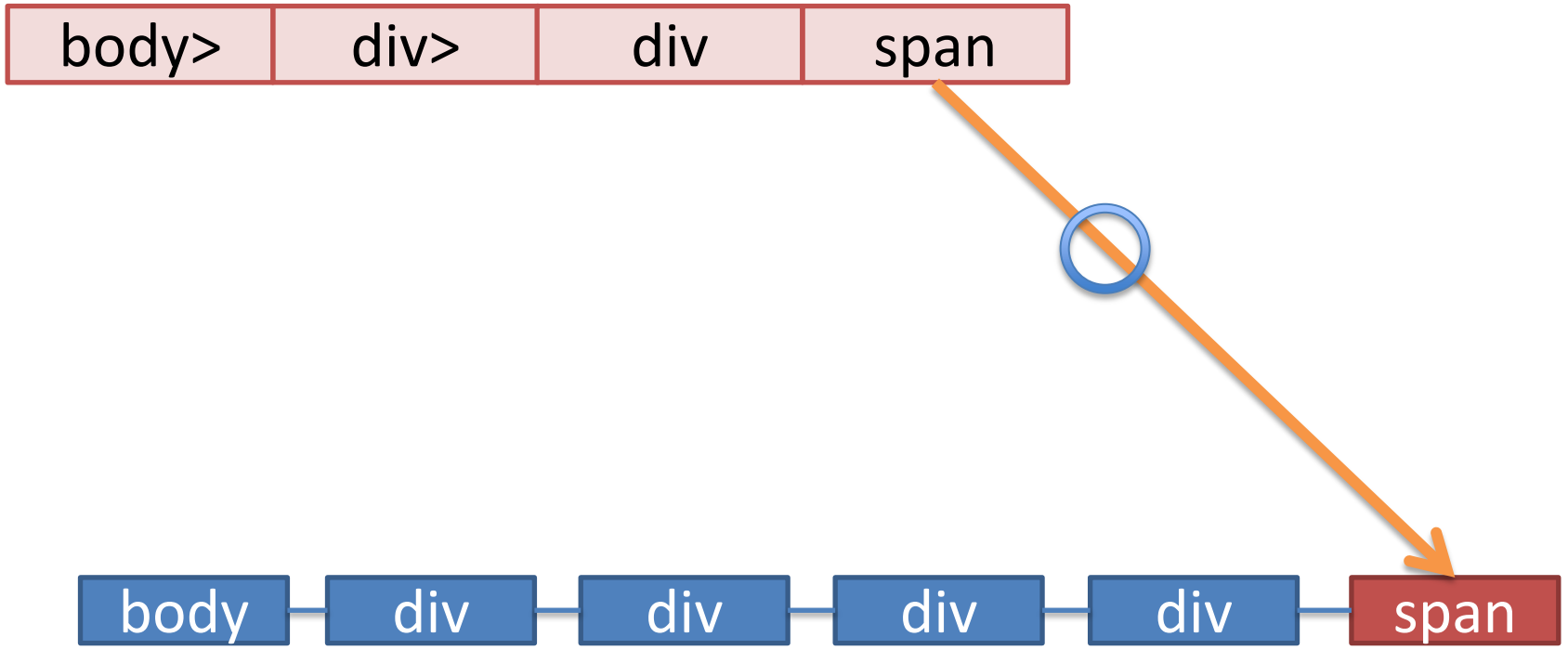
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



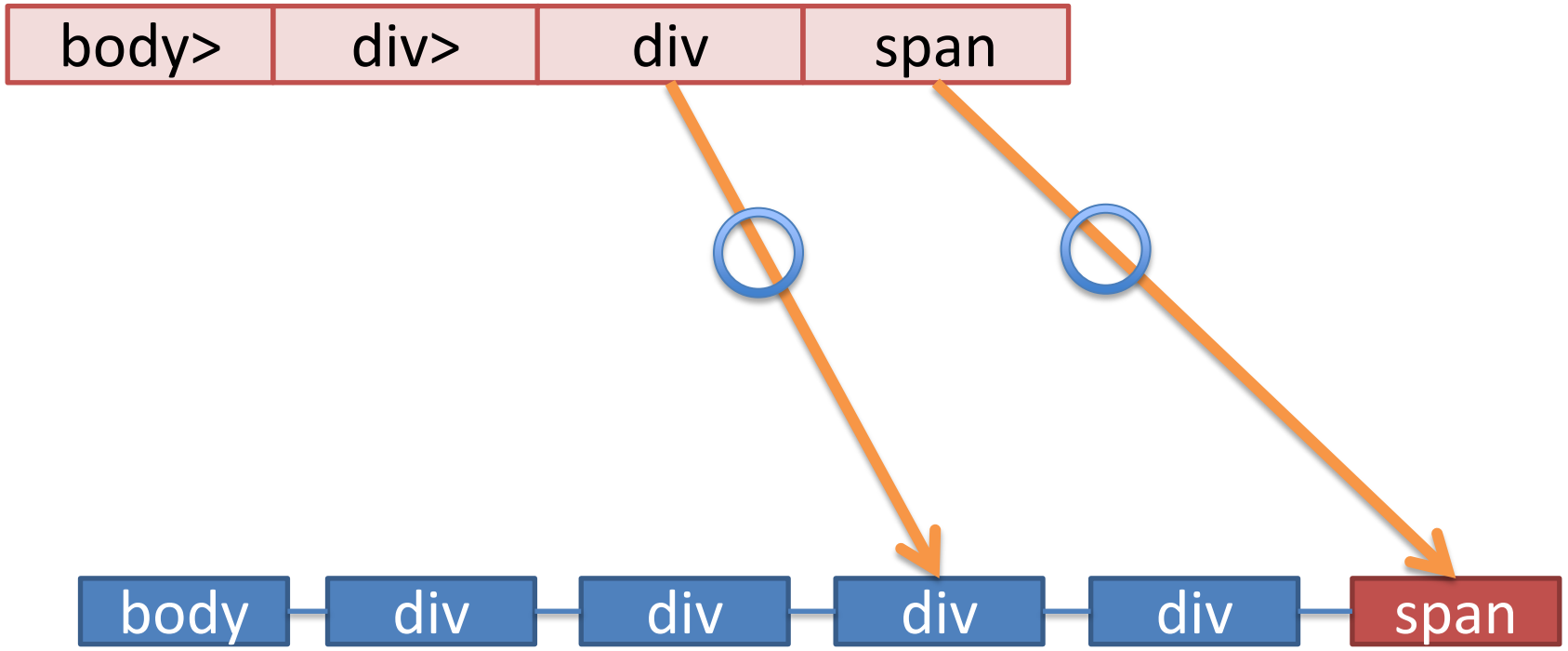
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



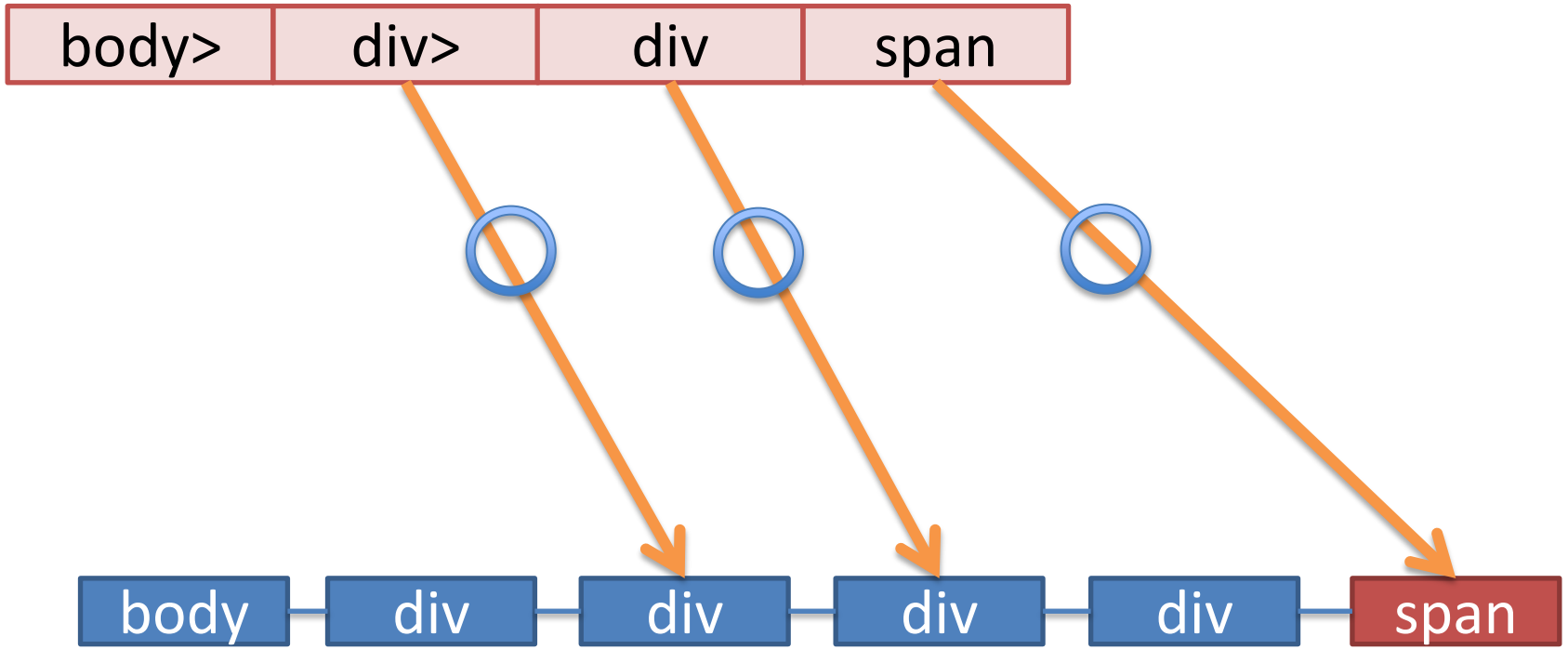
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



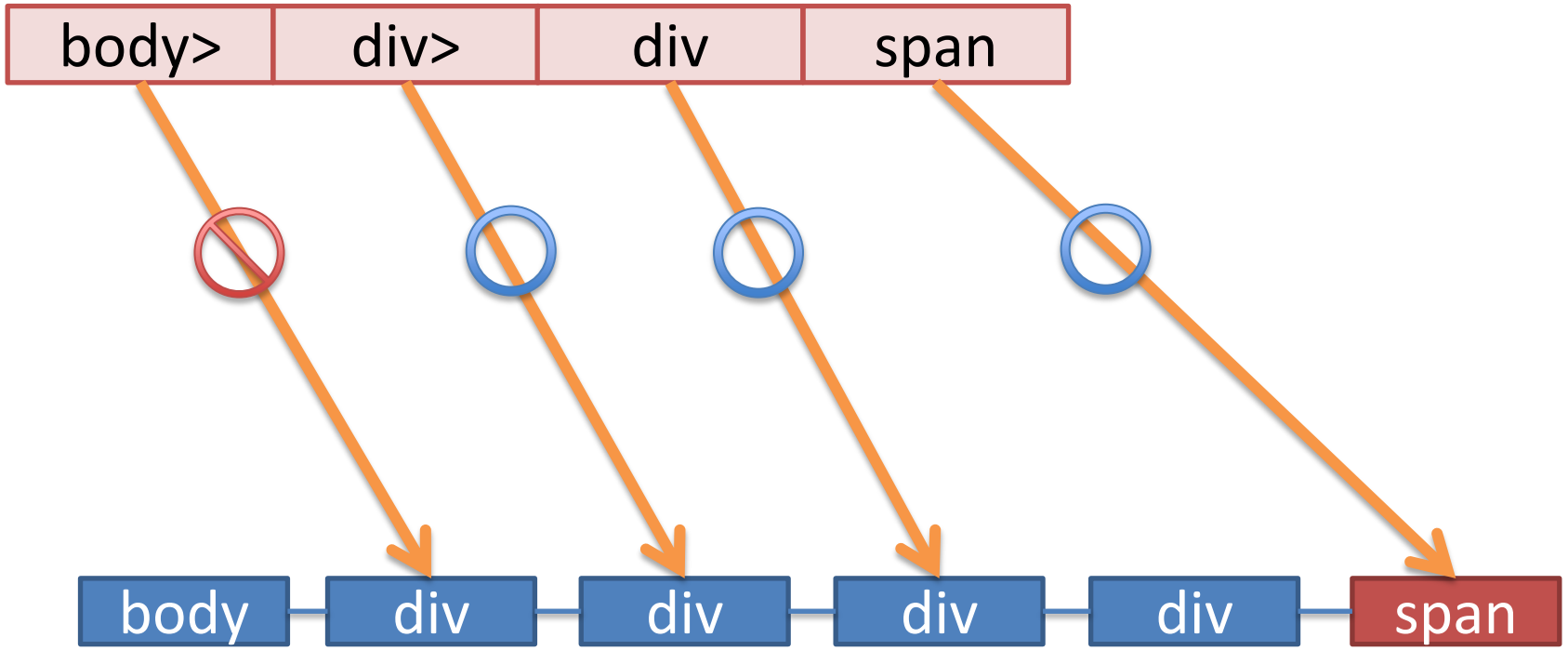
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



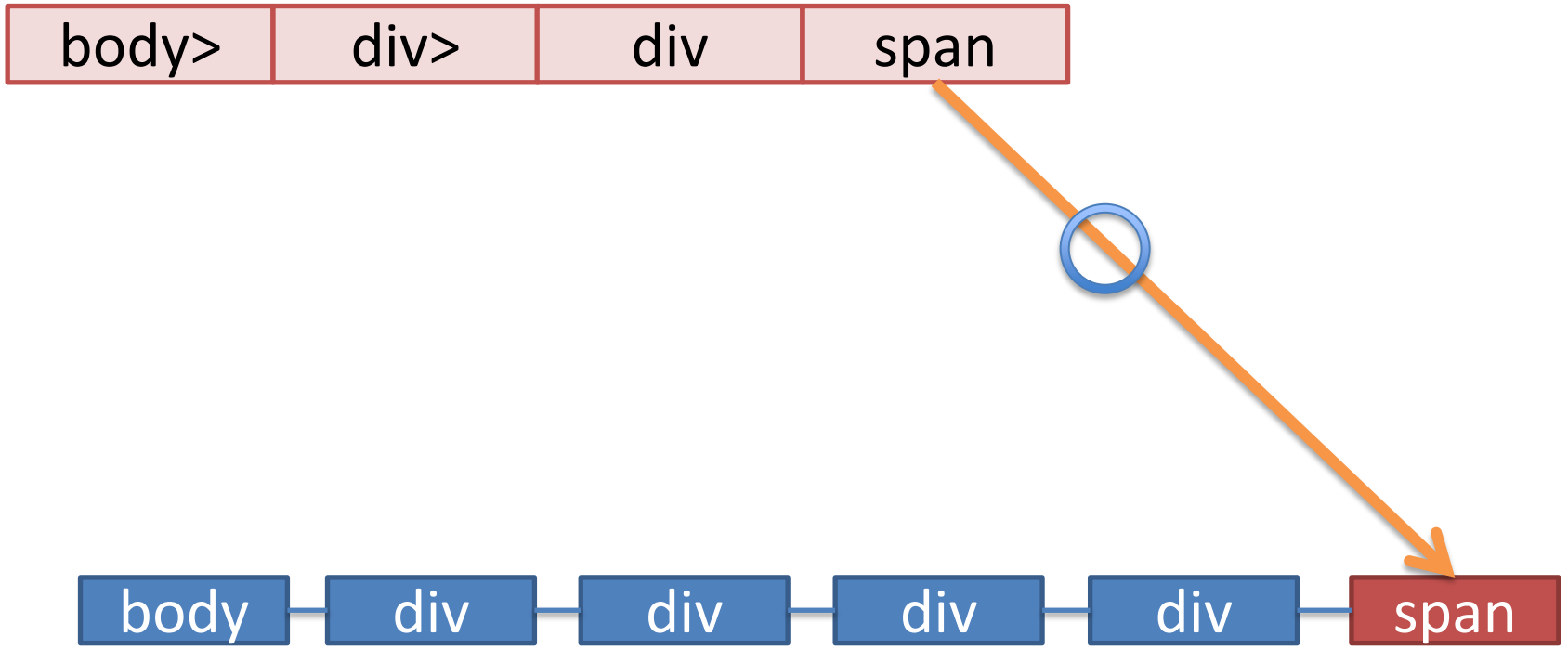
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



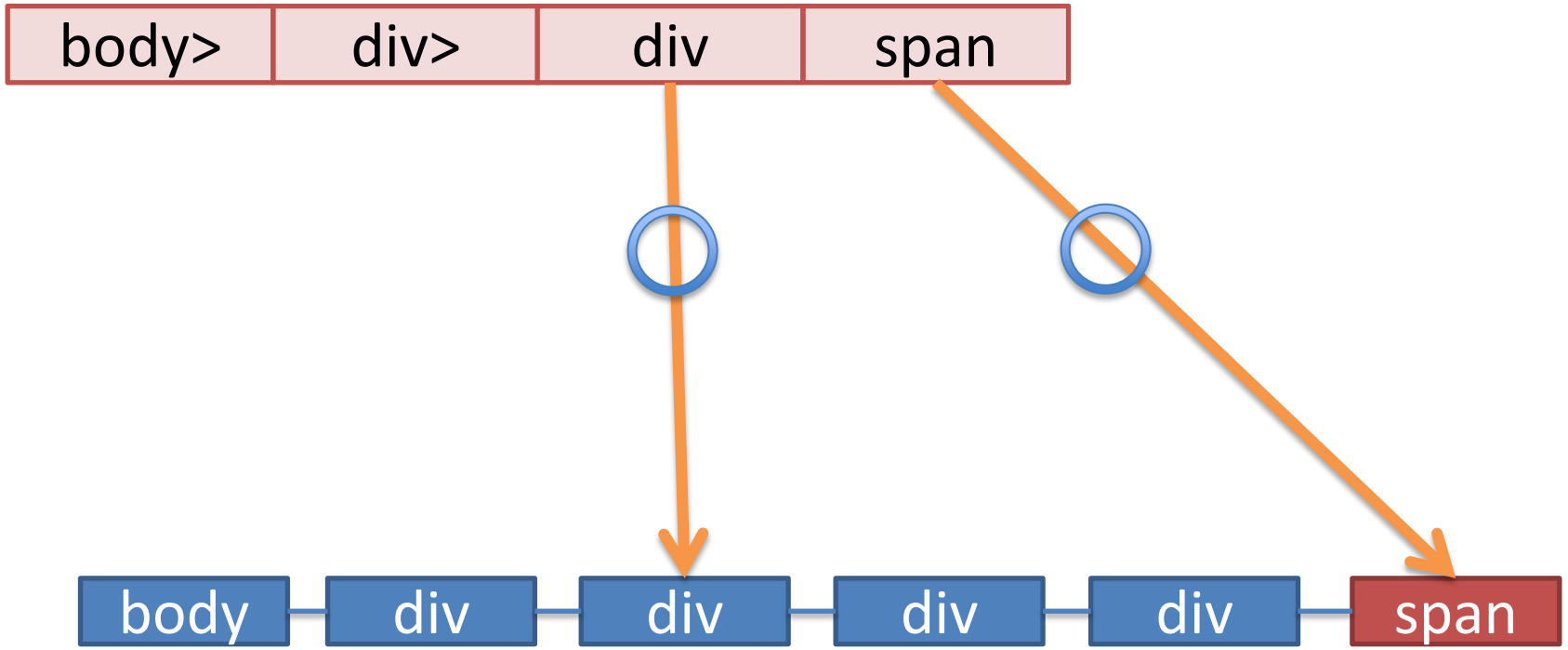
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



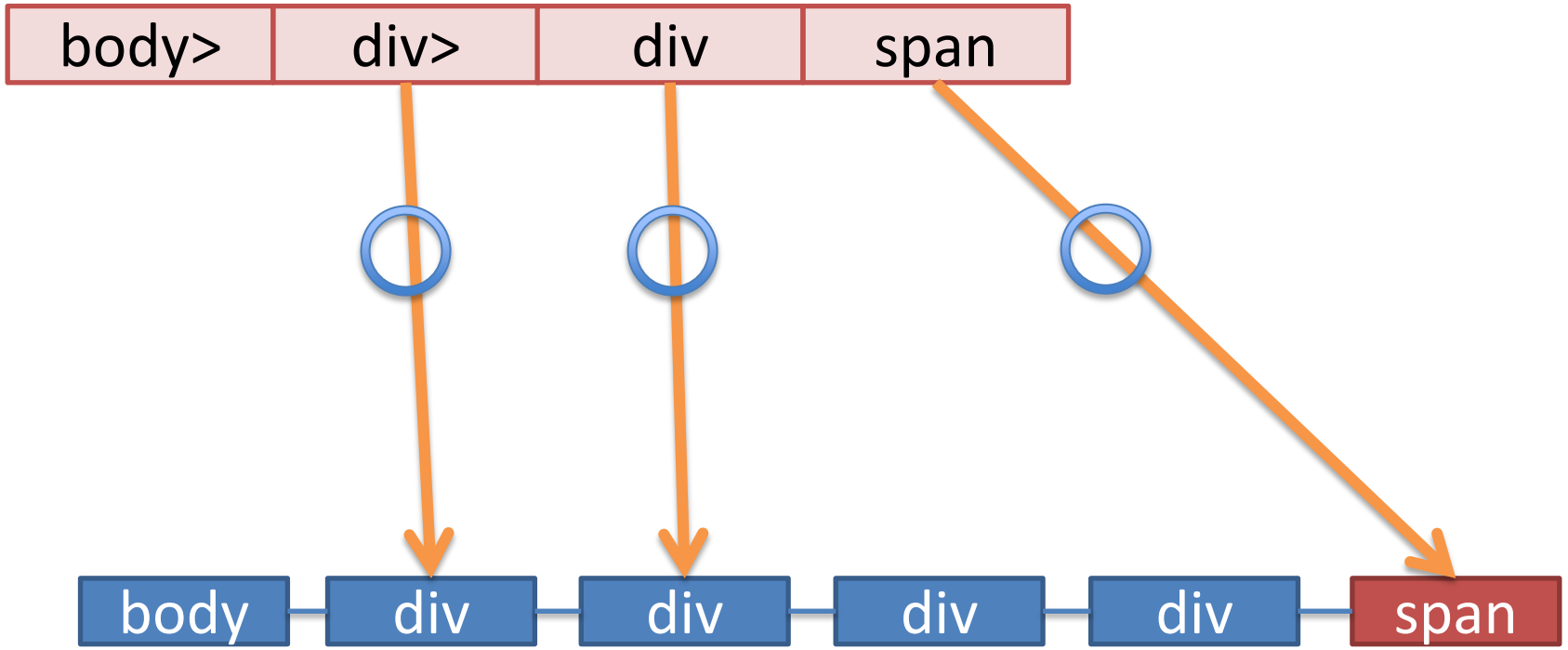
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



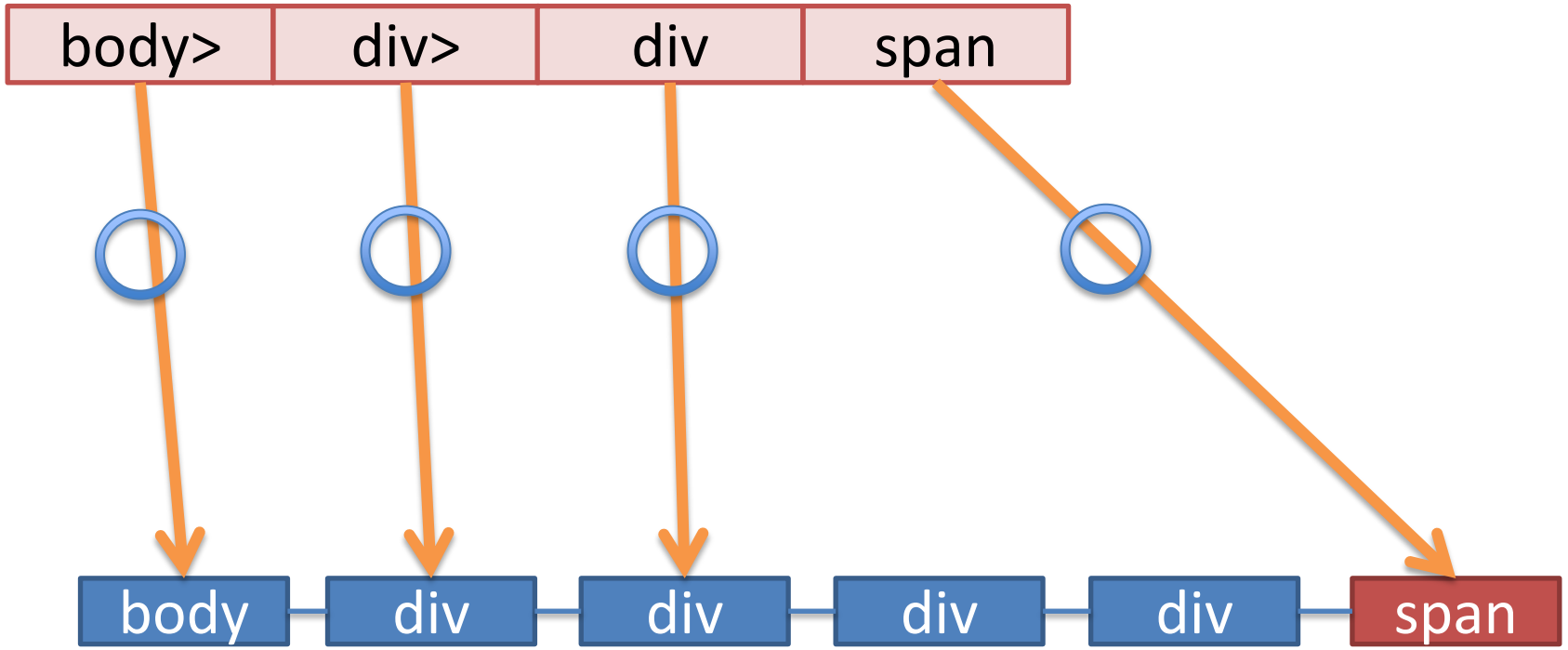
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



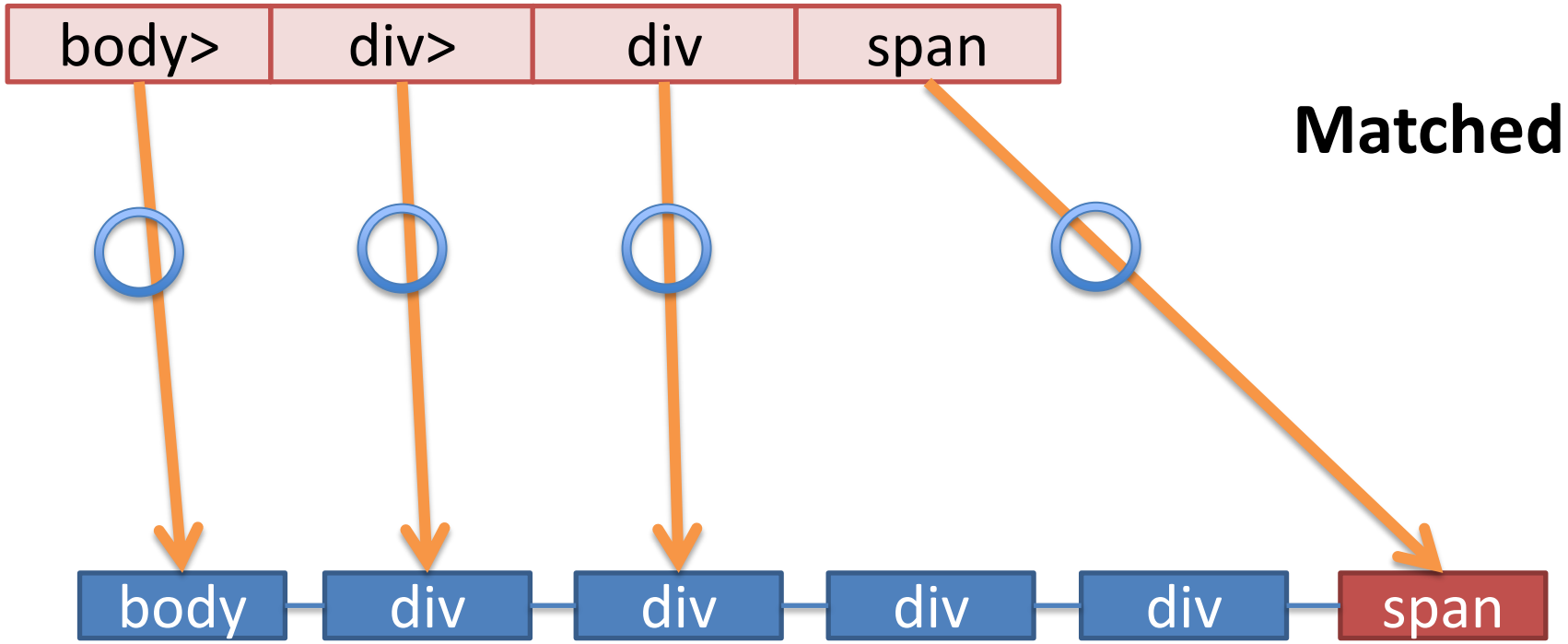
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



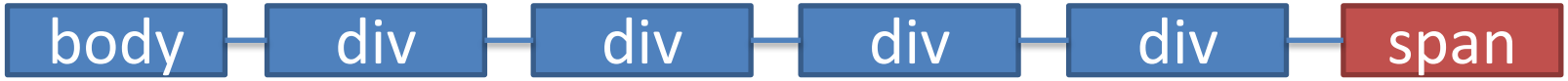
Evaluation Example: Backtracking

- *body> div> div span* case
- When child(xxx>) is failed,
  backtracking with the closest descendant(xxx)



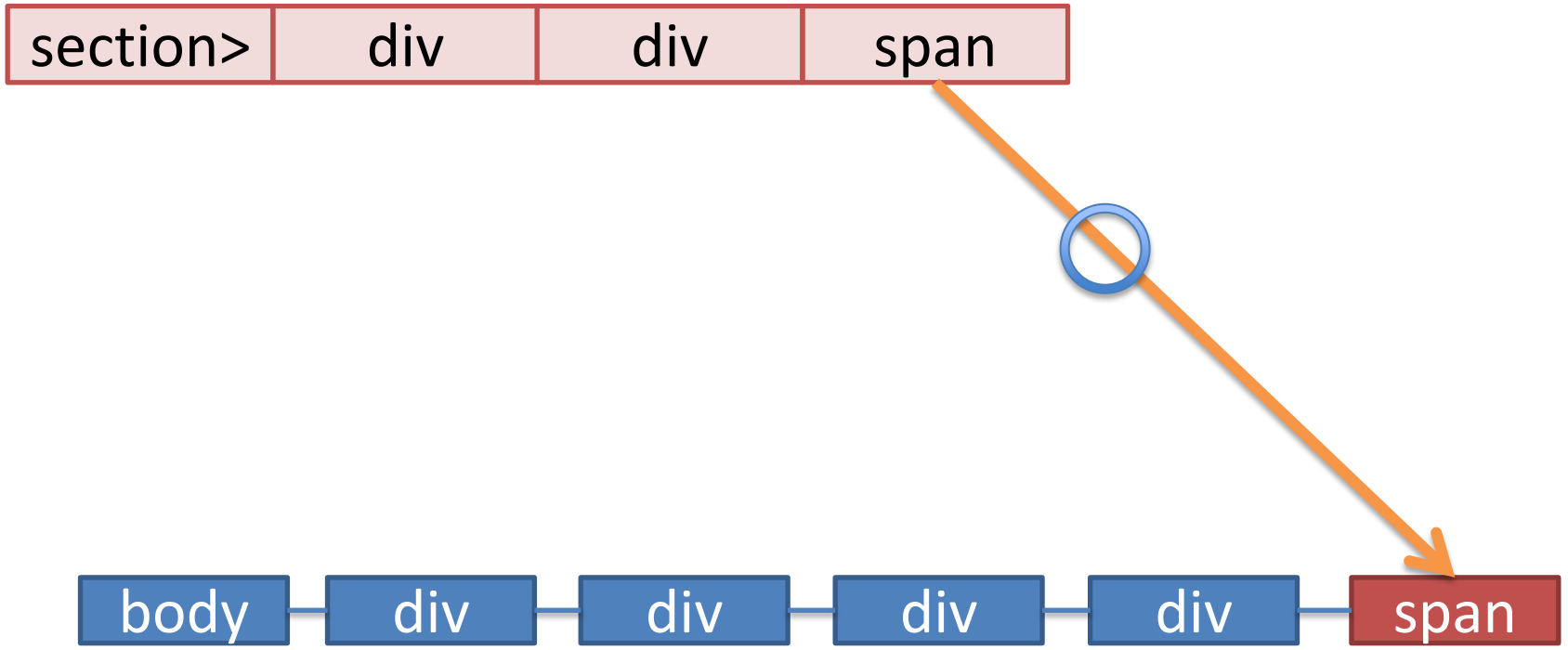
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



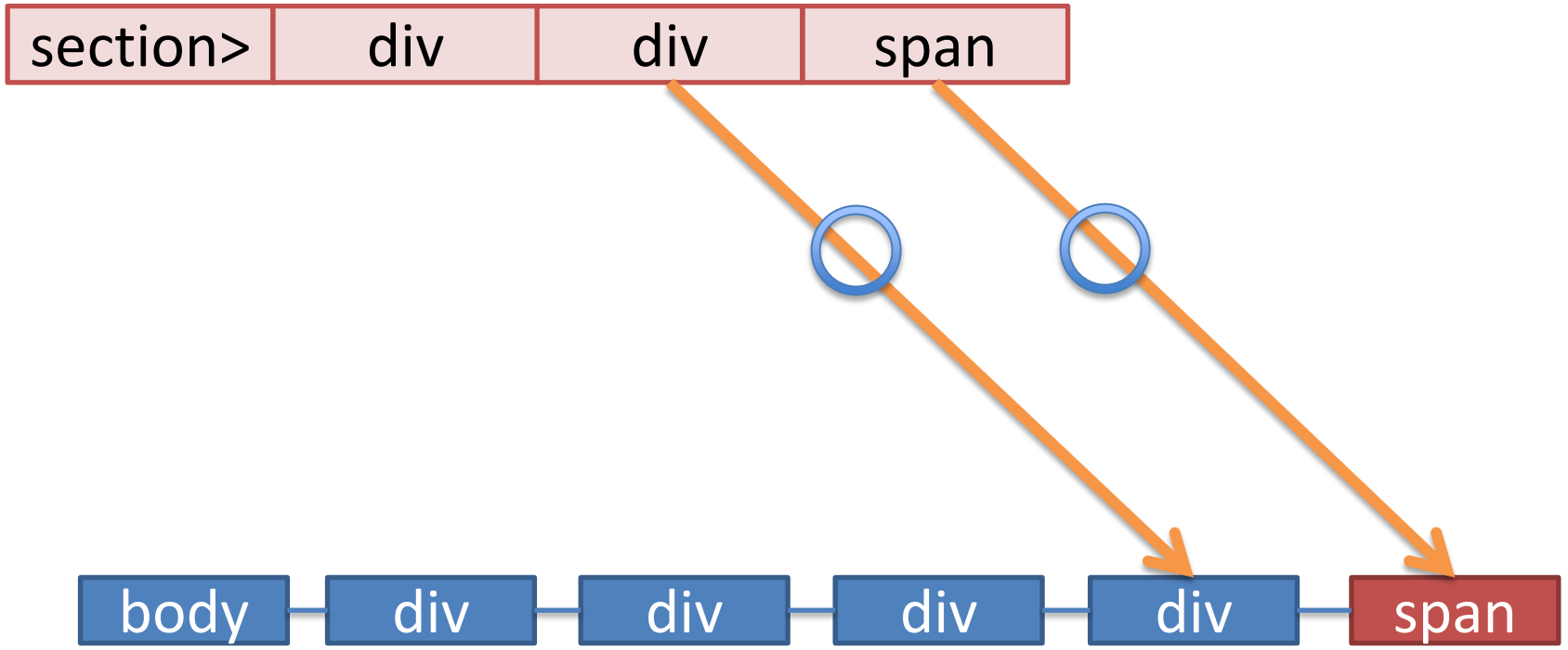
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



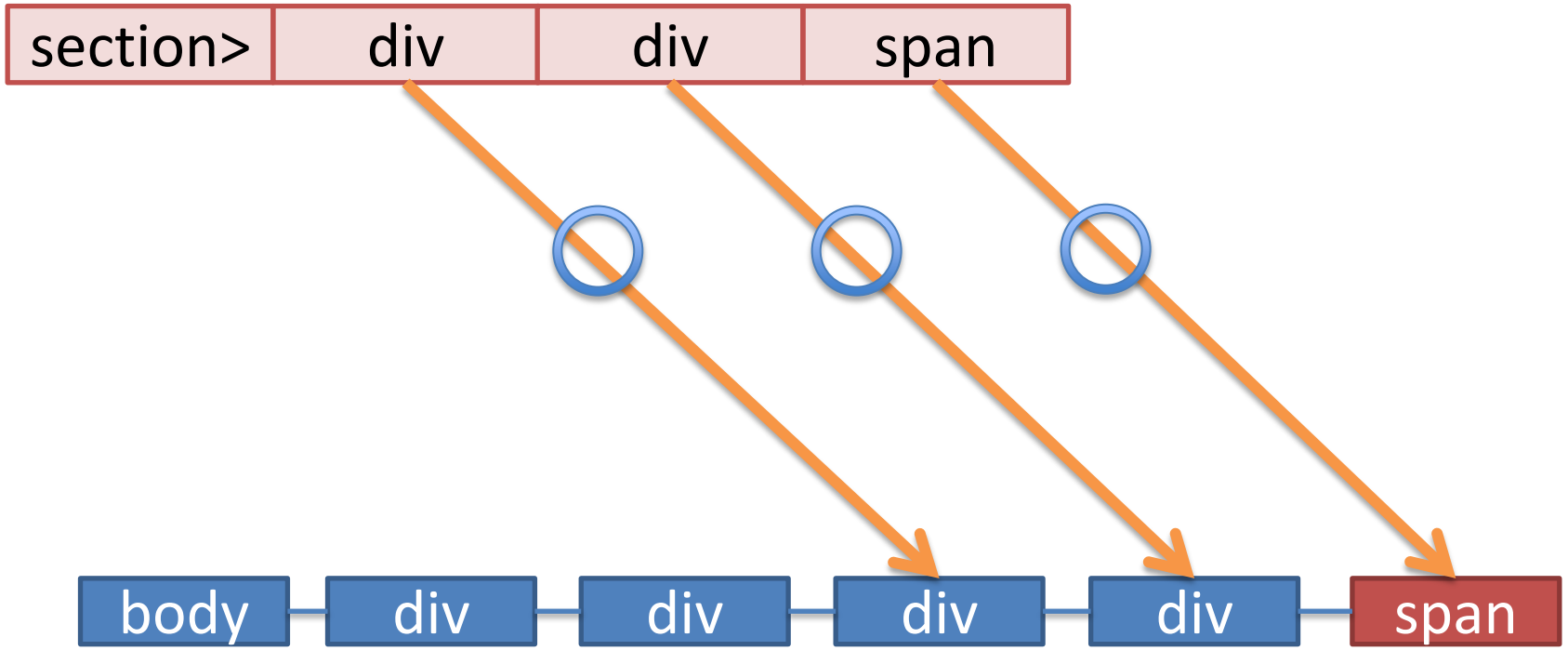
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



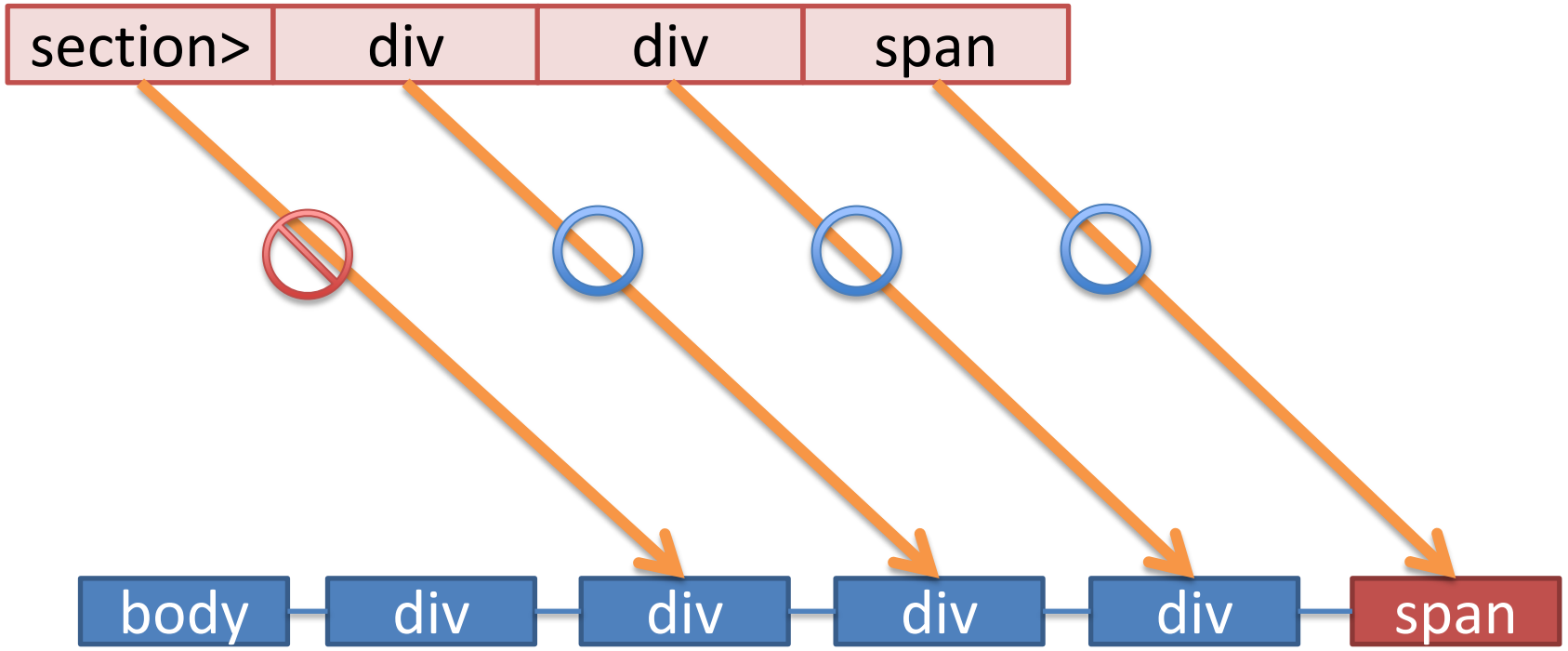
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



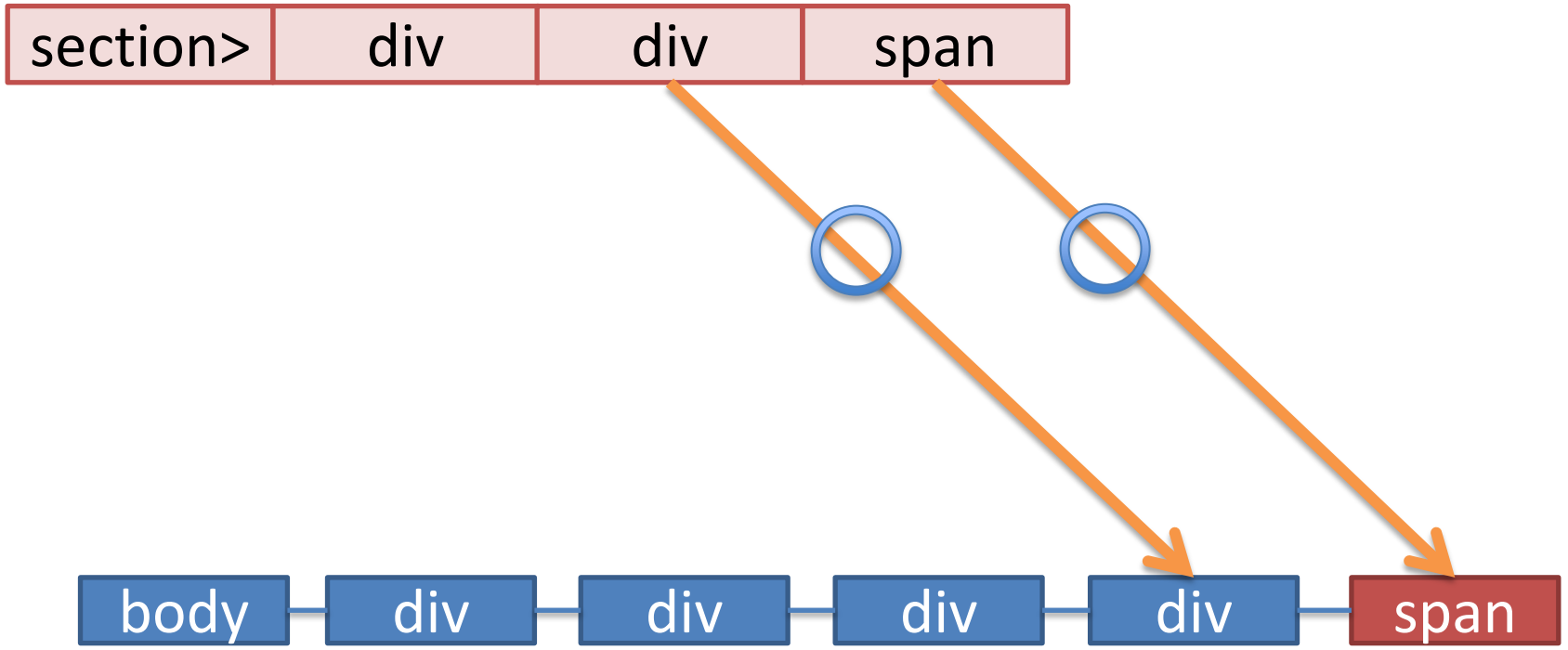
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



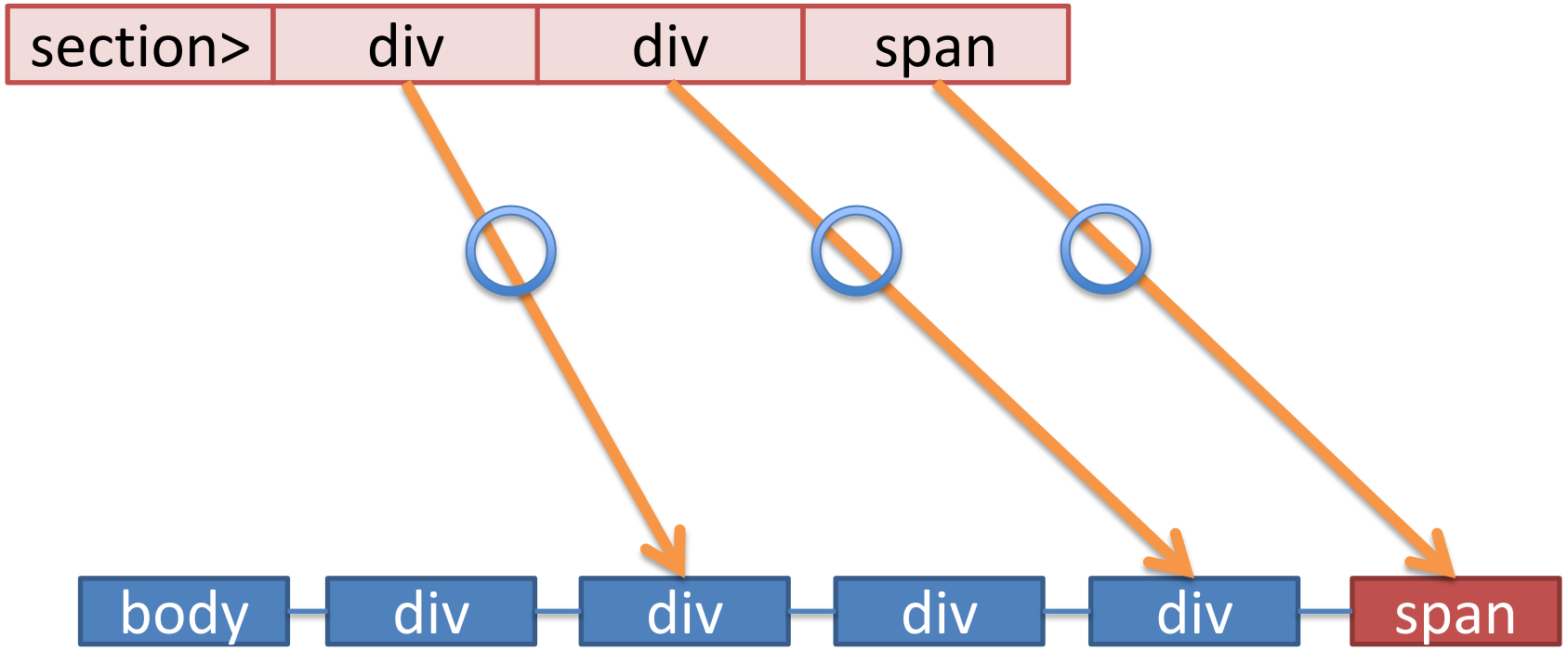
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



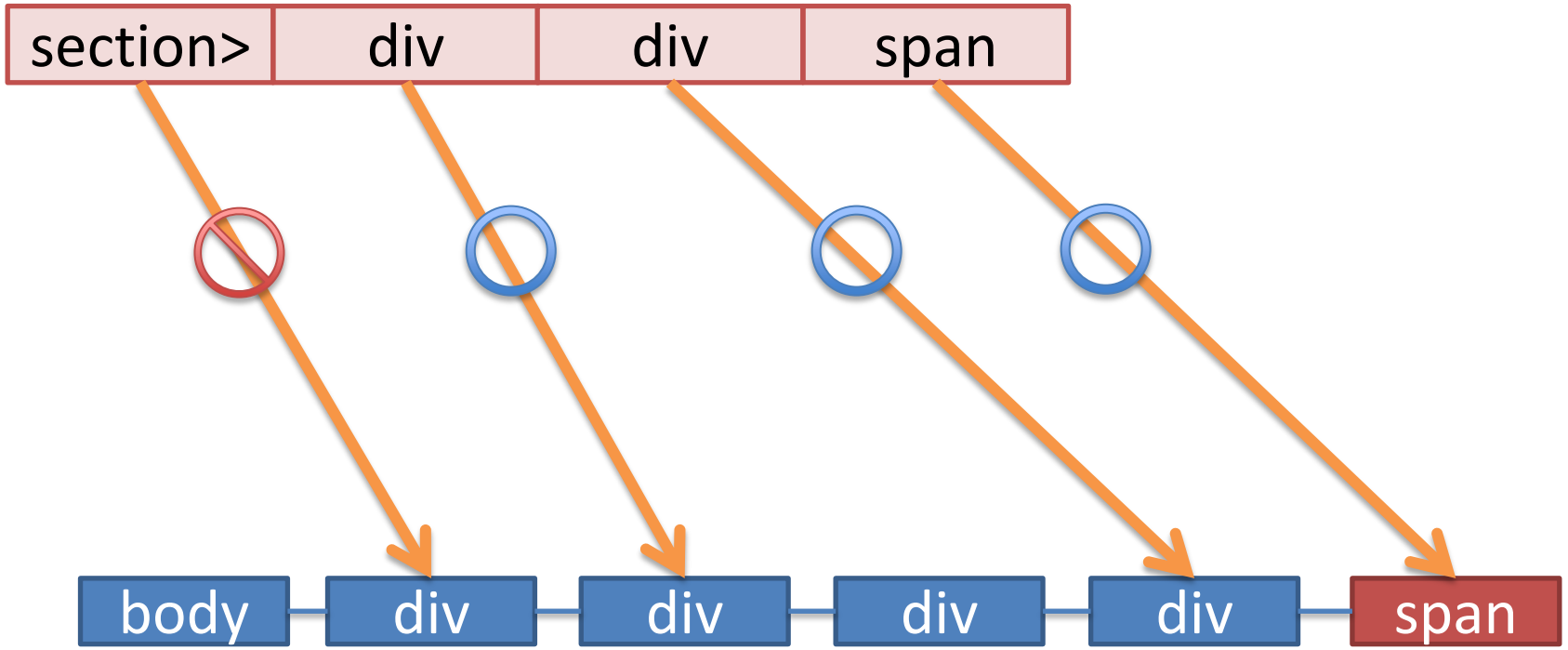
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



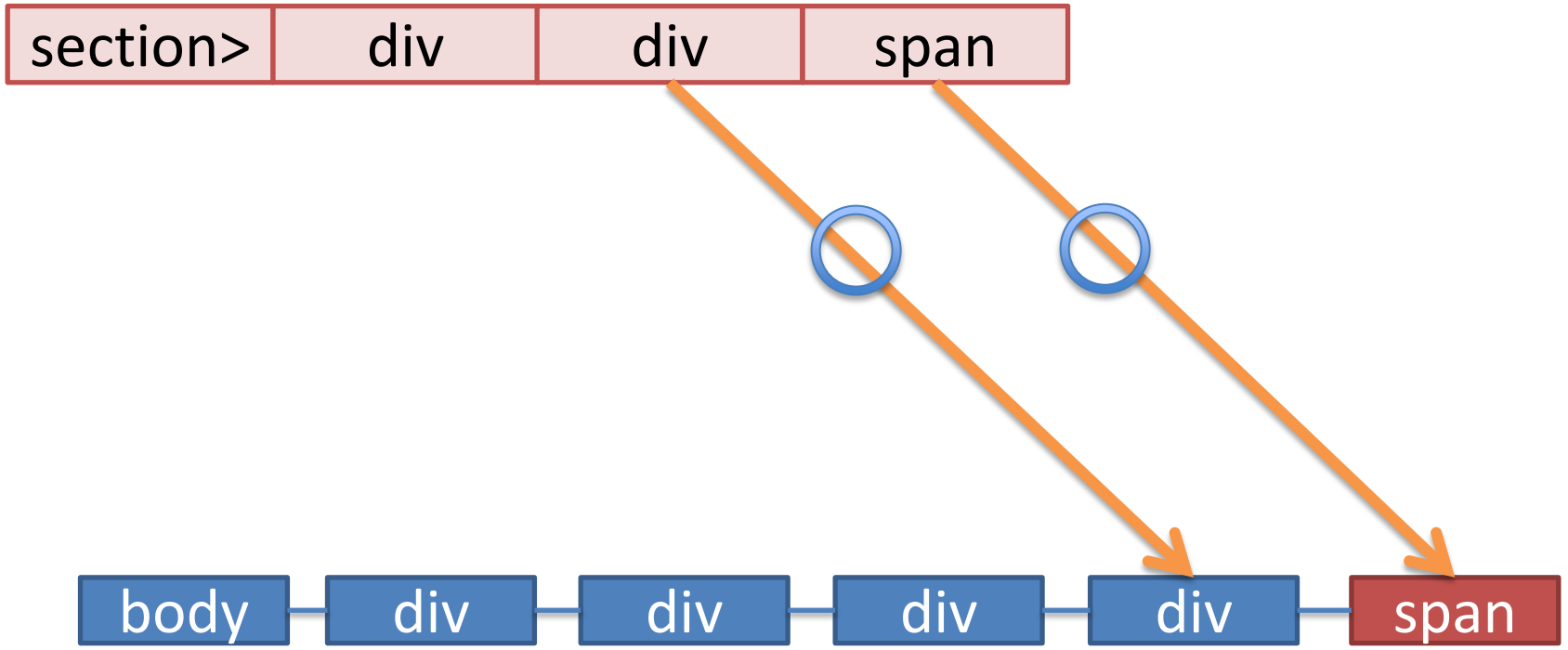
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



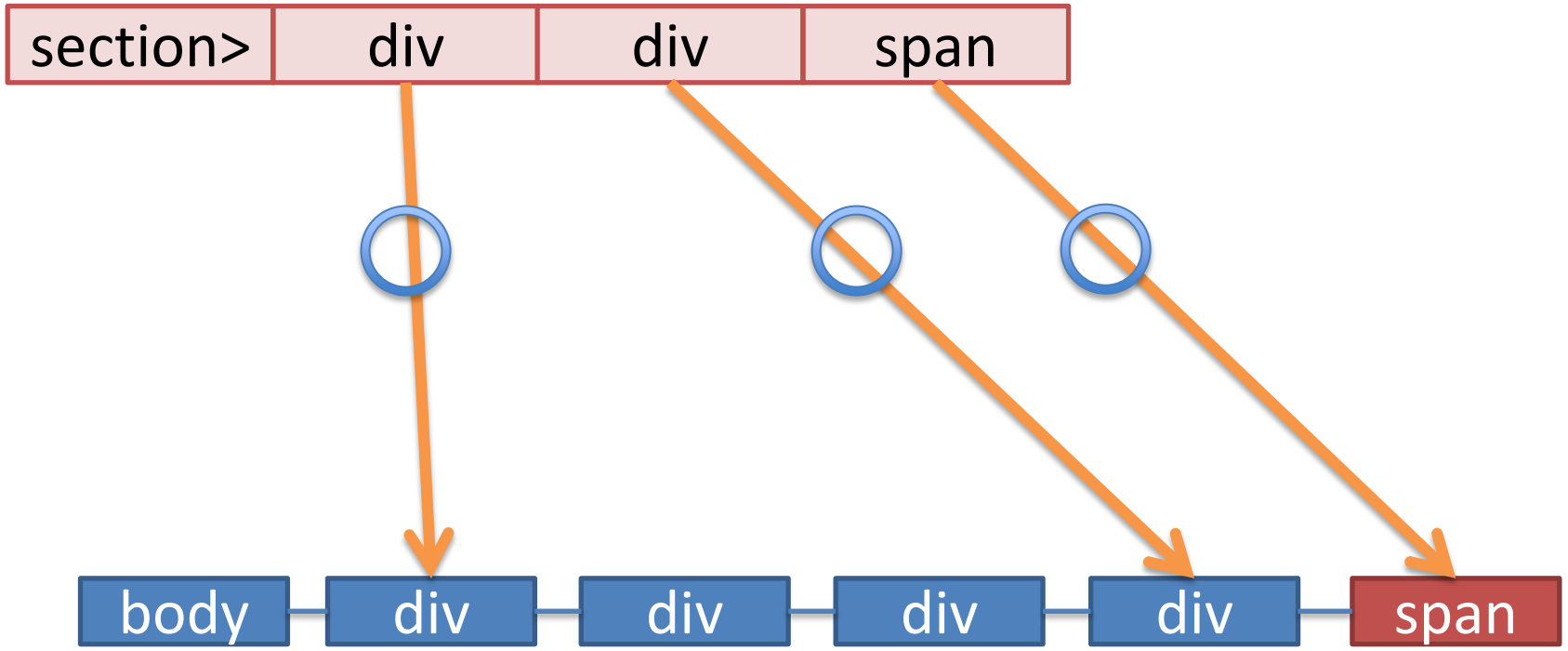
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



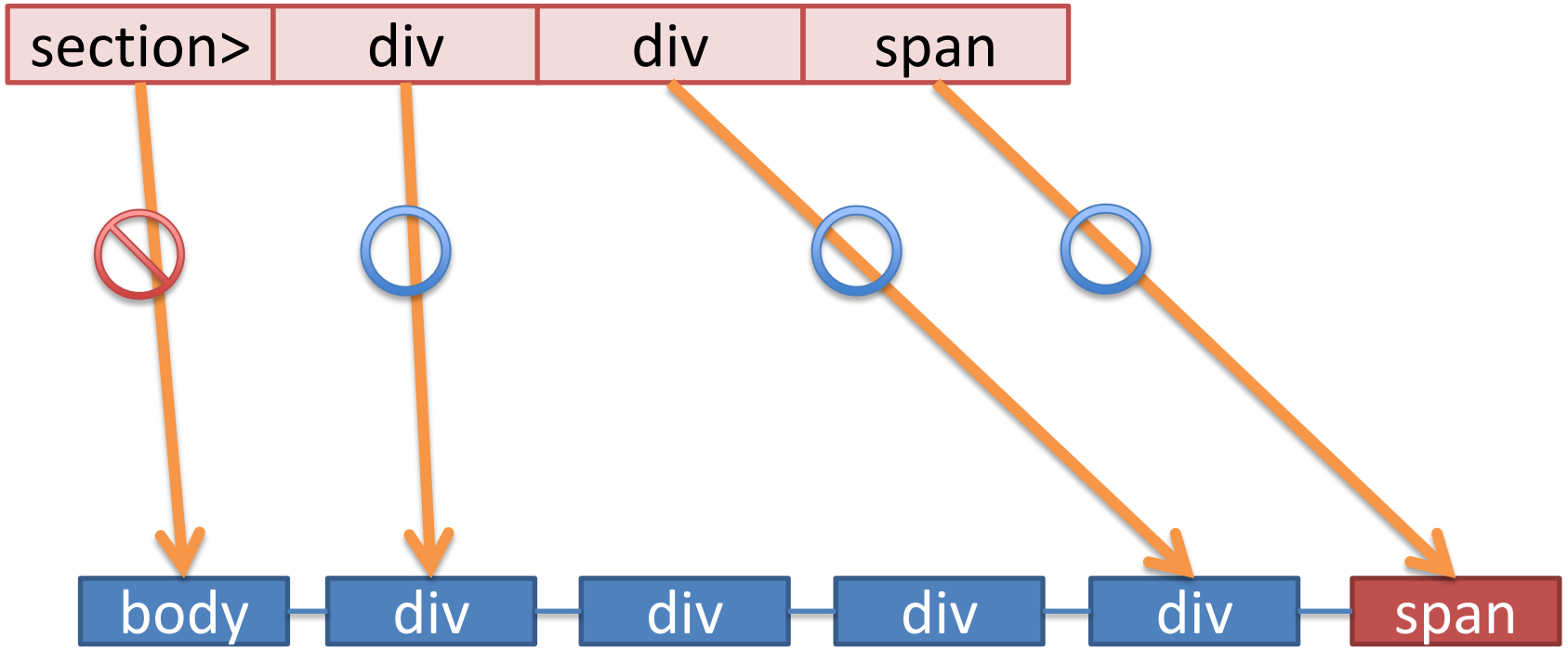
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



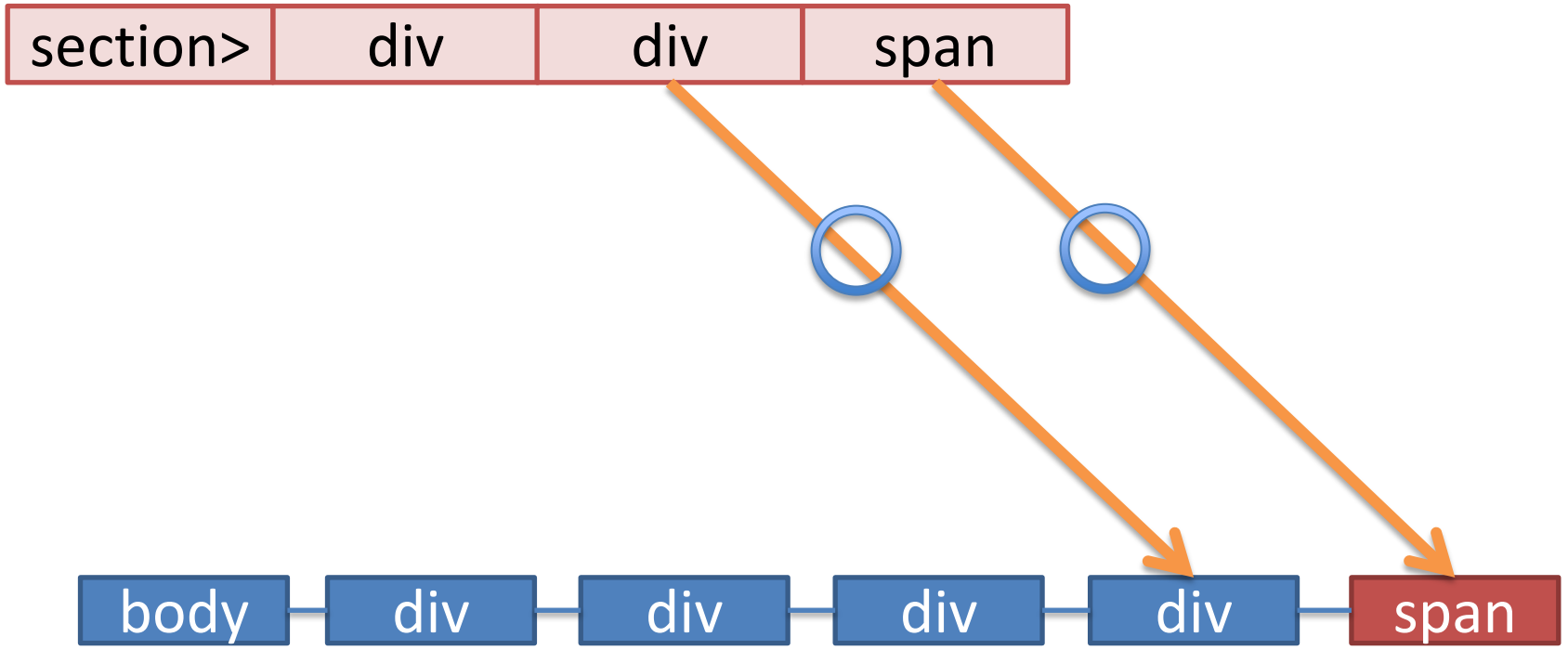
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



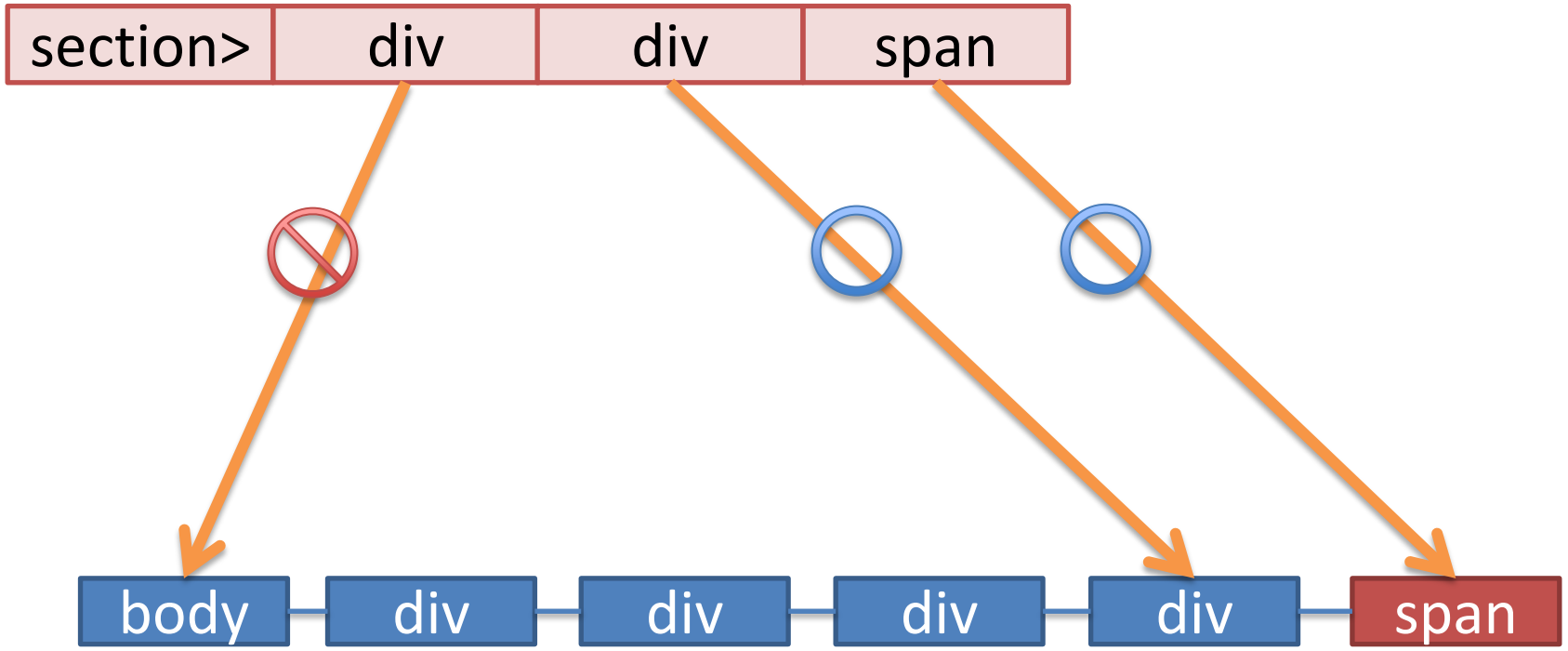
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



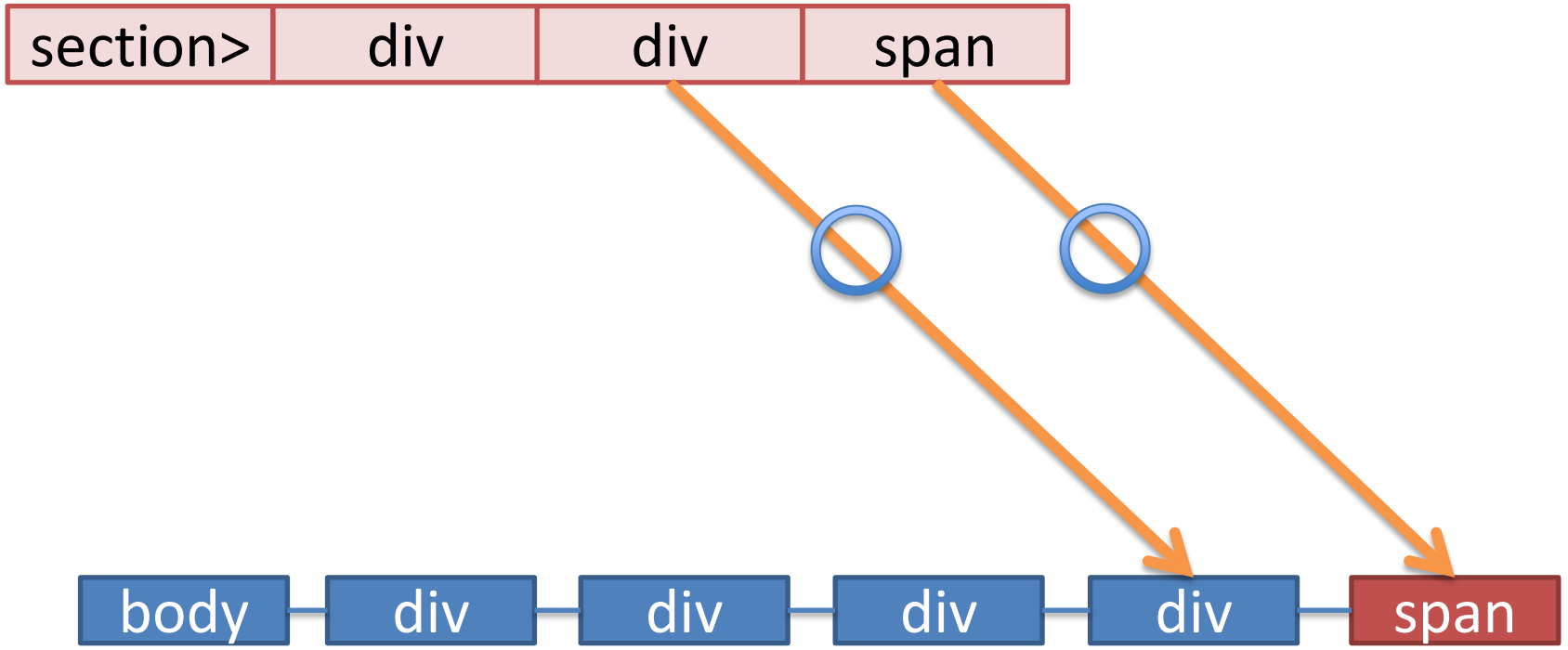
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



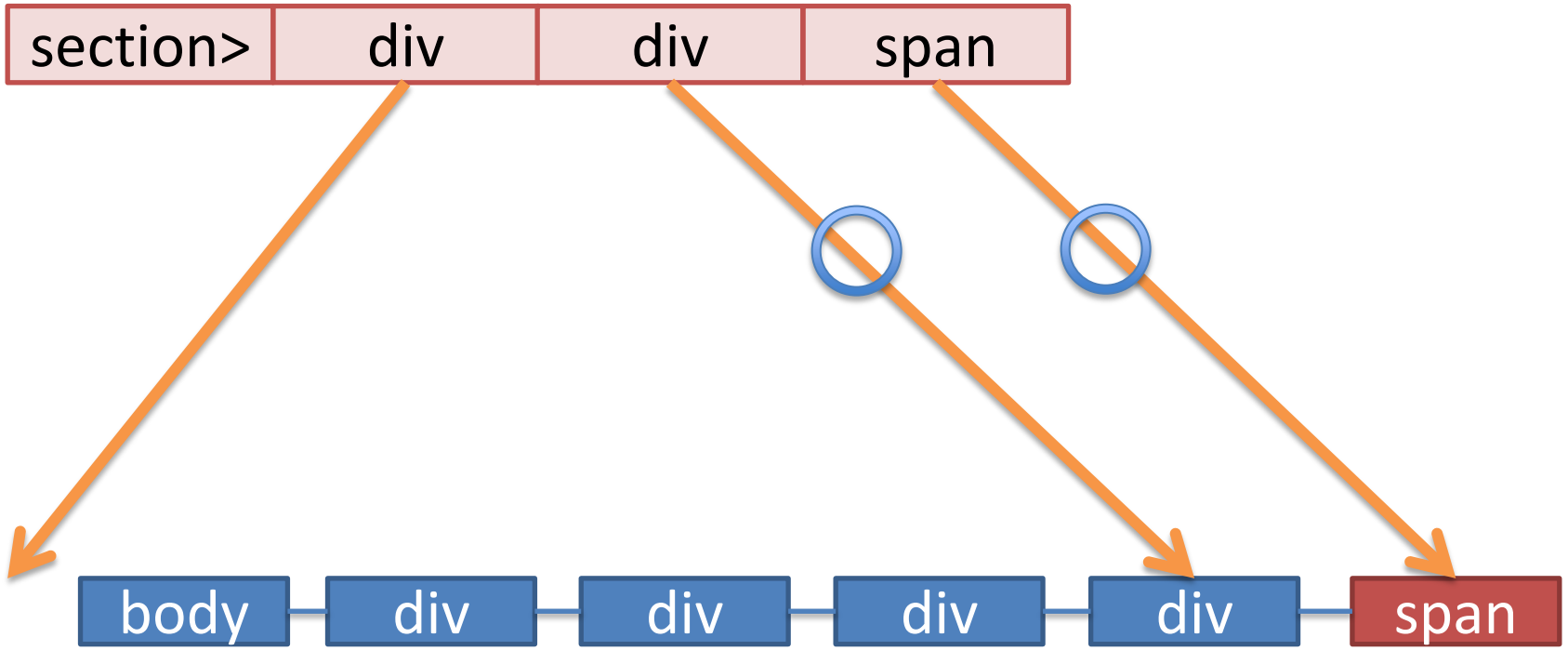
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



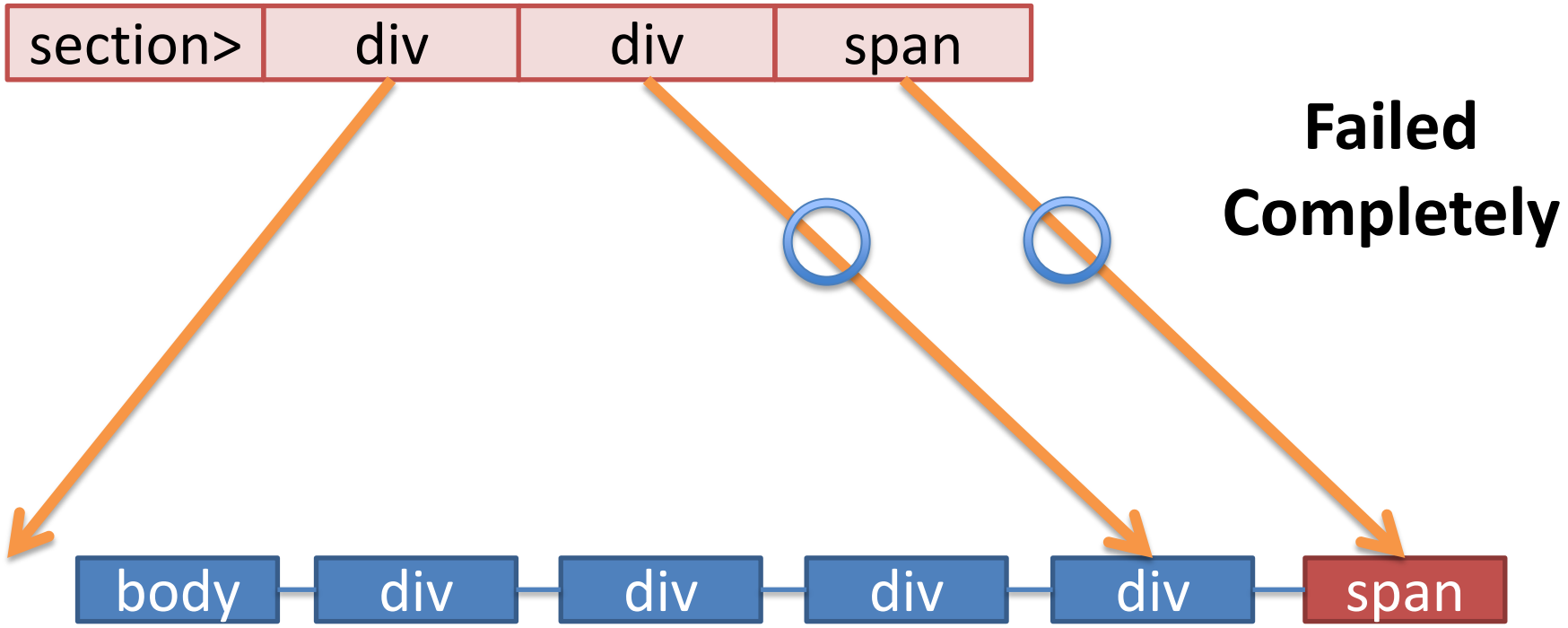
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



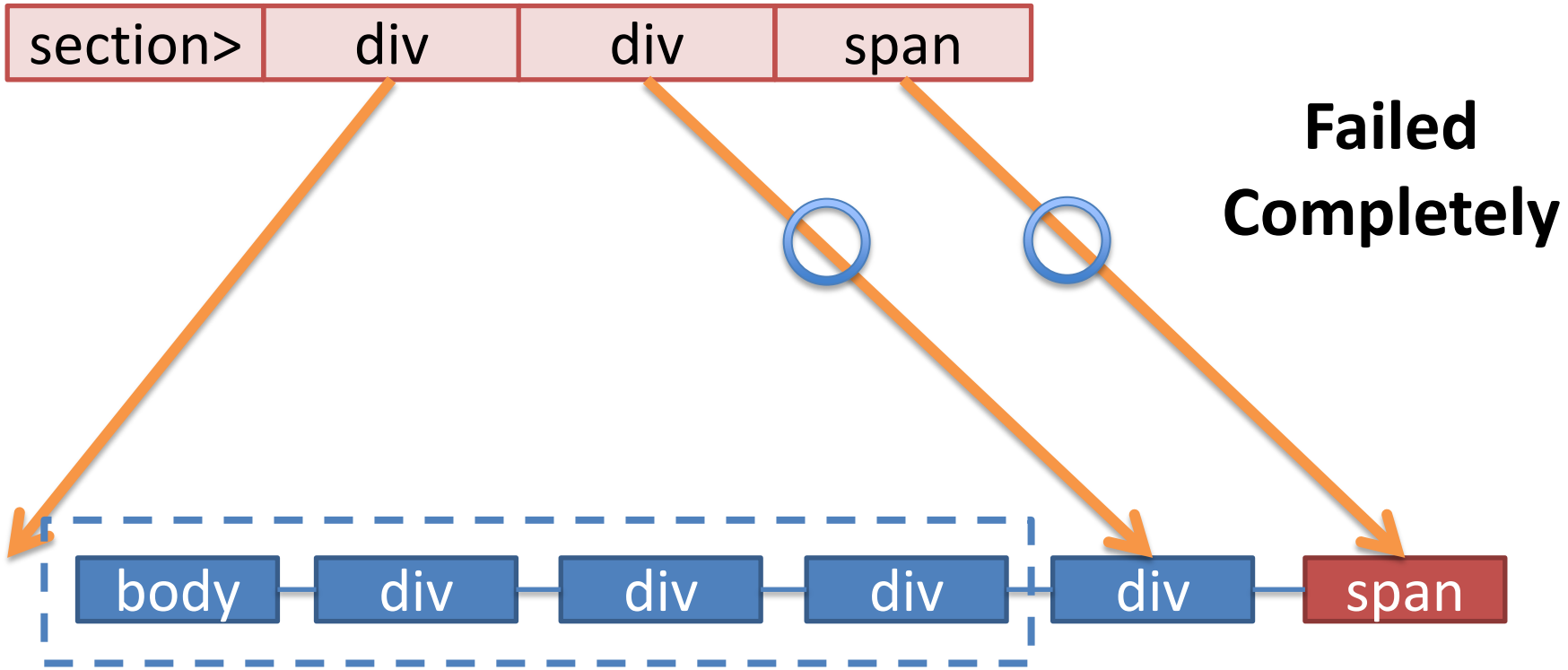
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



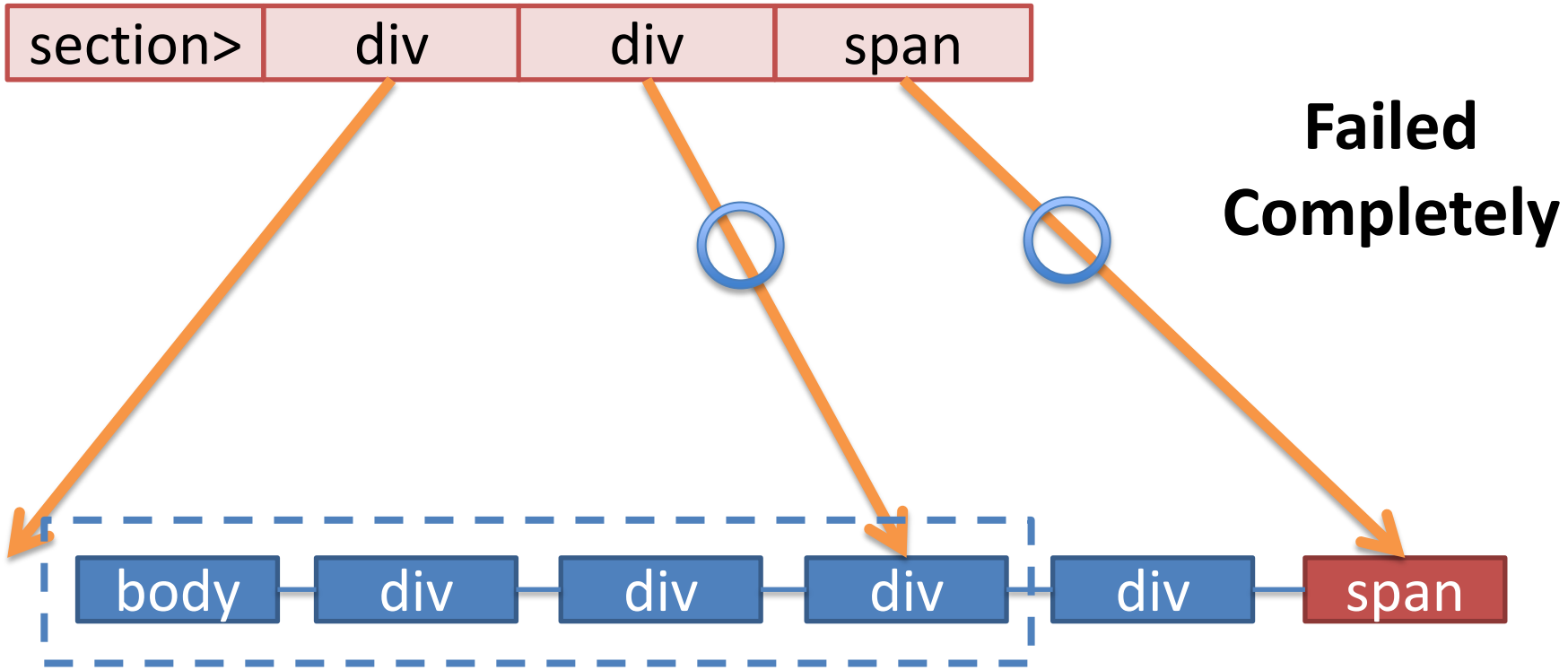
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



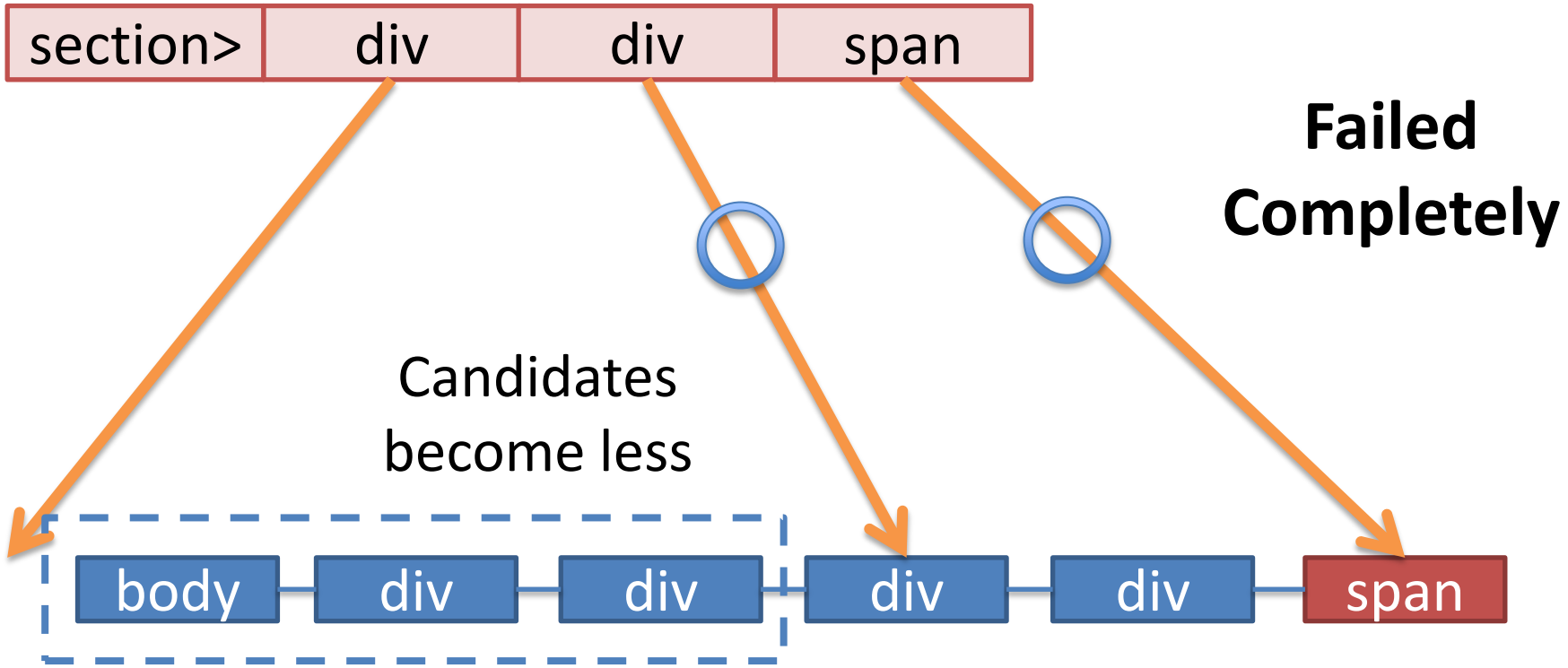
Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



Evaluation Example: Backtracking

- *section> div div span* case
- When descendant(xxx) is failed, selector matching completely failes.



Outline

- Motivation & Goals
- Existing CSS Selector Implementation
- CSS Selector JIT
- Conclusion

CSS Selector JIT

- Just-in-Time compile Selector Matching predicate
  - Provide faster selector matching
- Leverage selector's static data
  to generate highly optimized machine code

Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

Example 1: div> div span

Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

Example 1: div> div span

span

SUCCESS

FAIL

Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

Example 1: div> div span

Compiling machine code

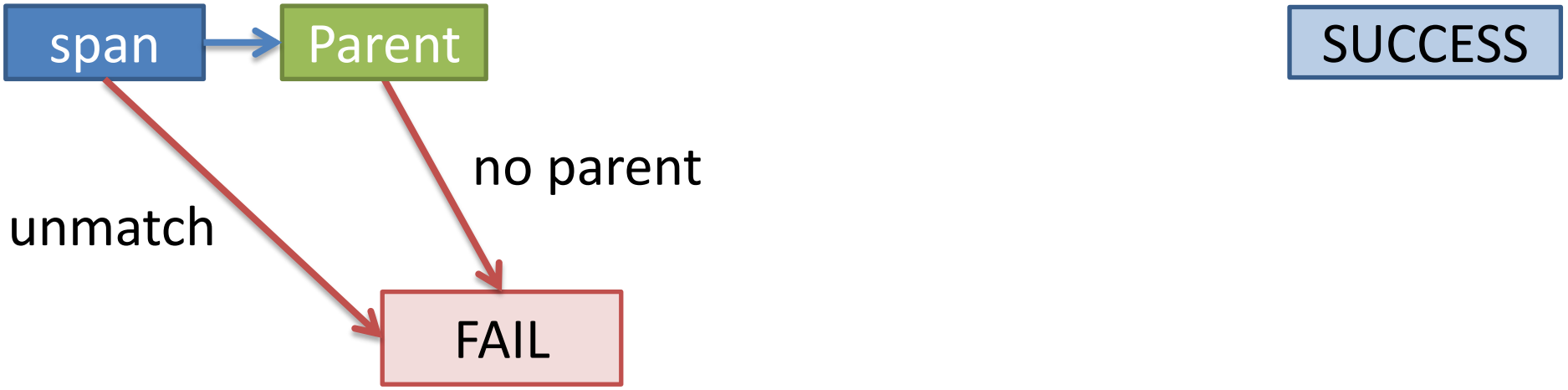
- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

Example 1: div> div span

Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

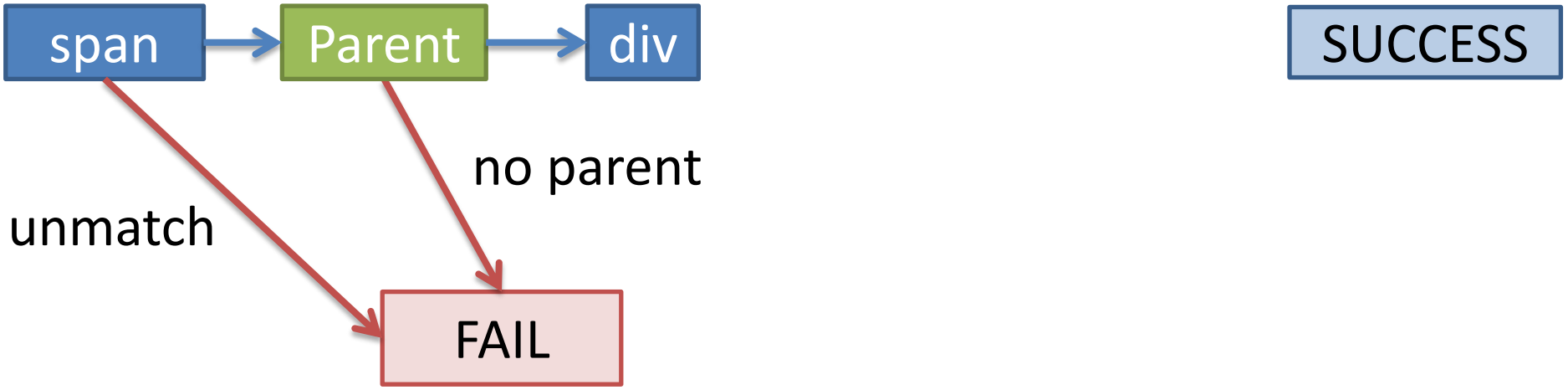
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

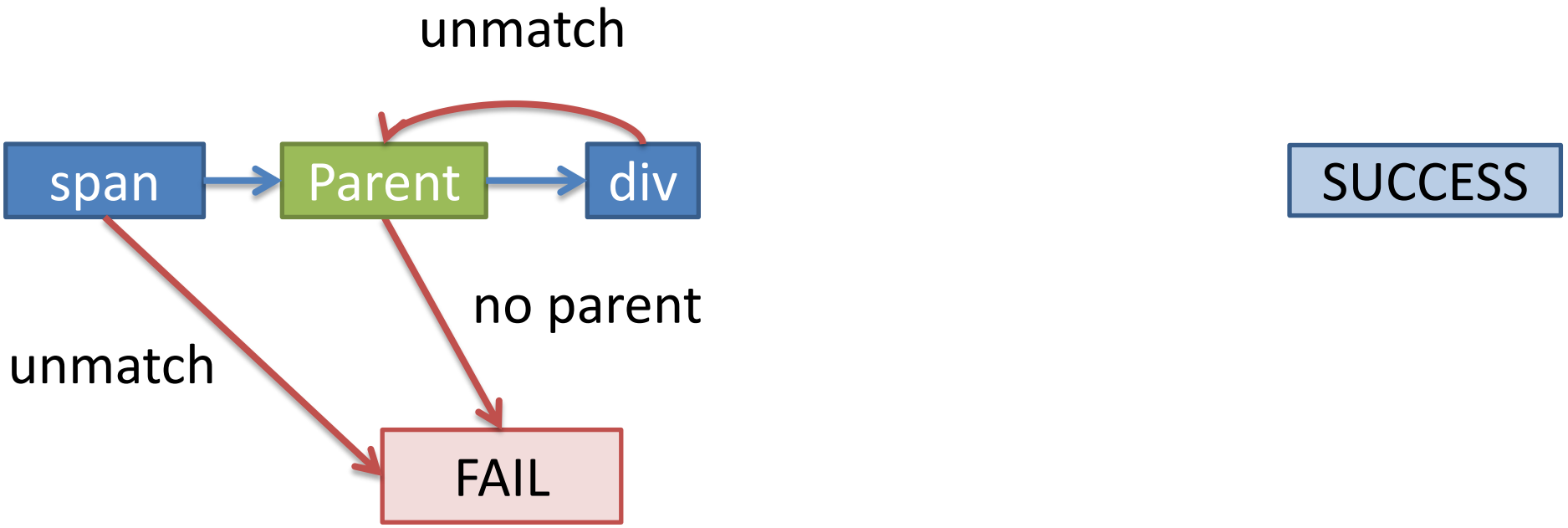
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

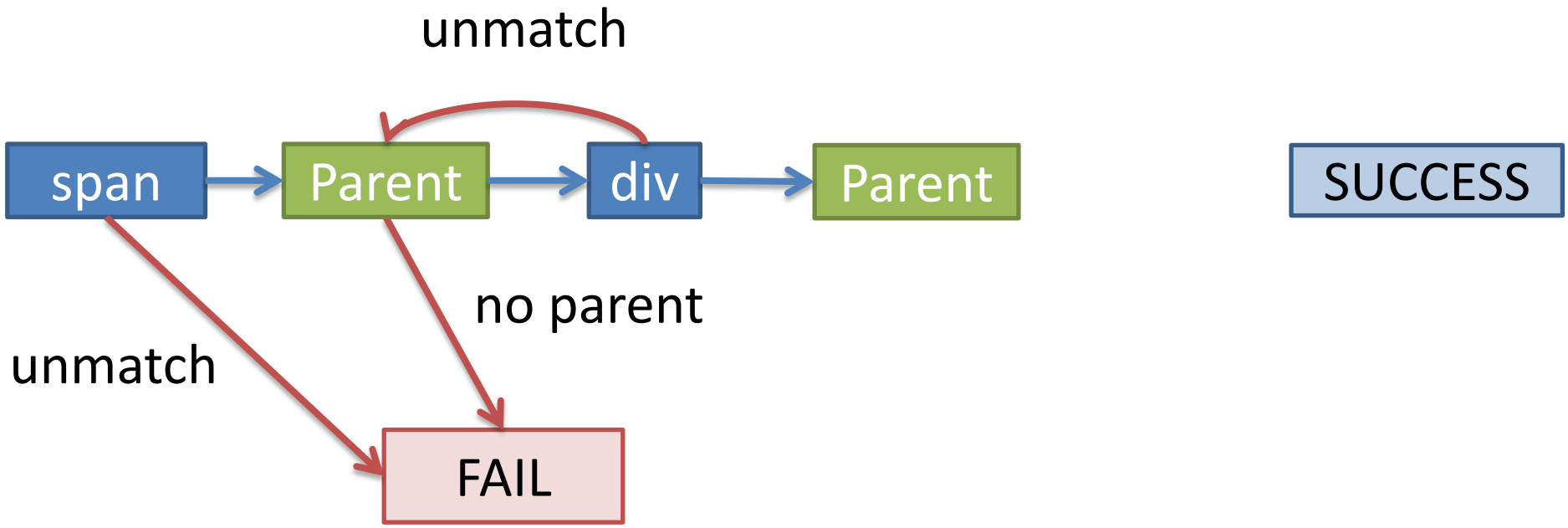
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

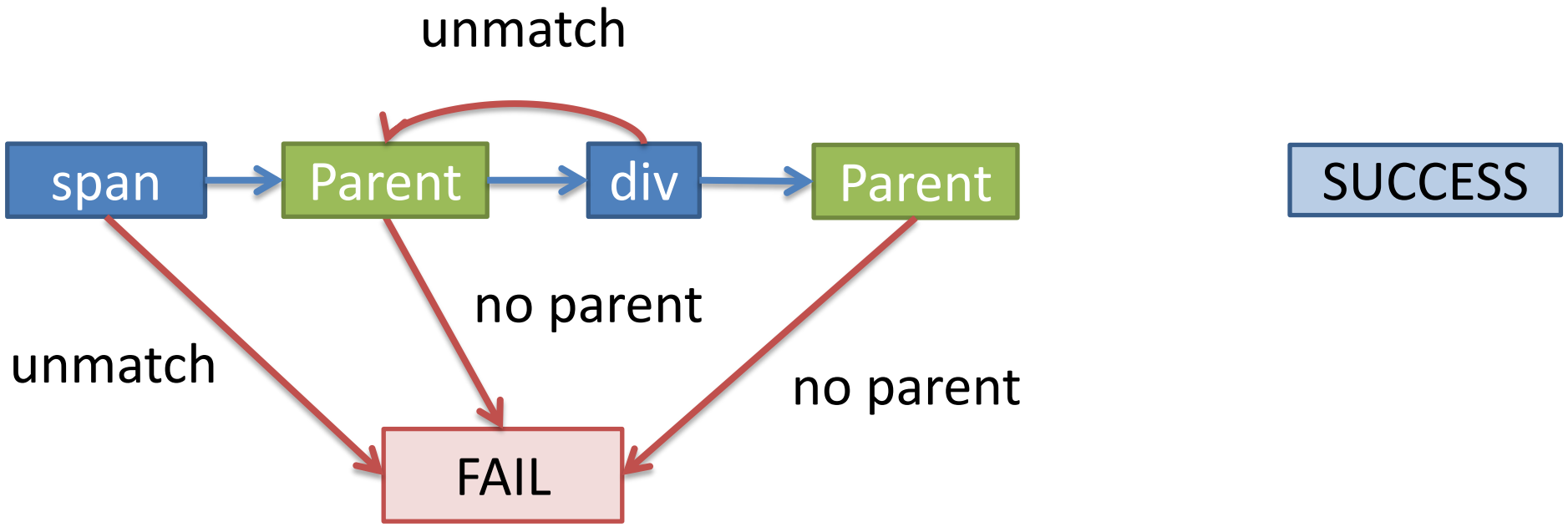
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

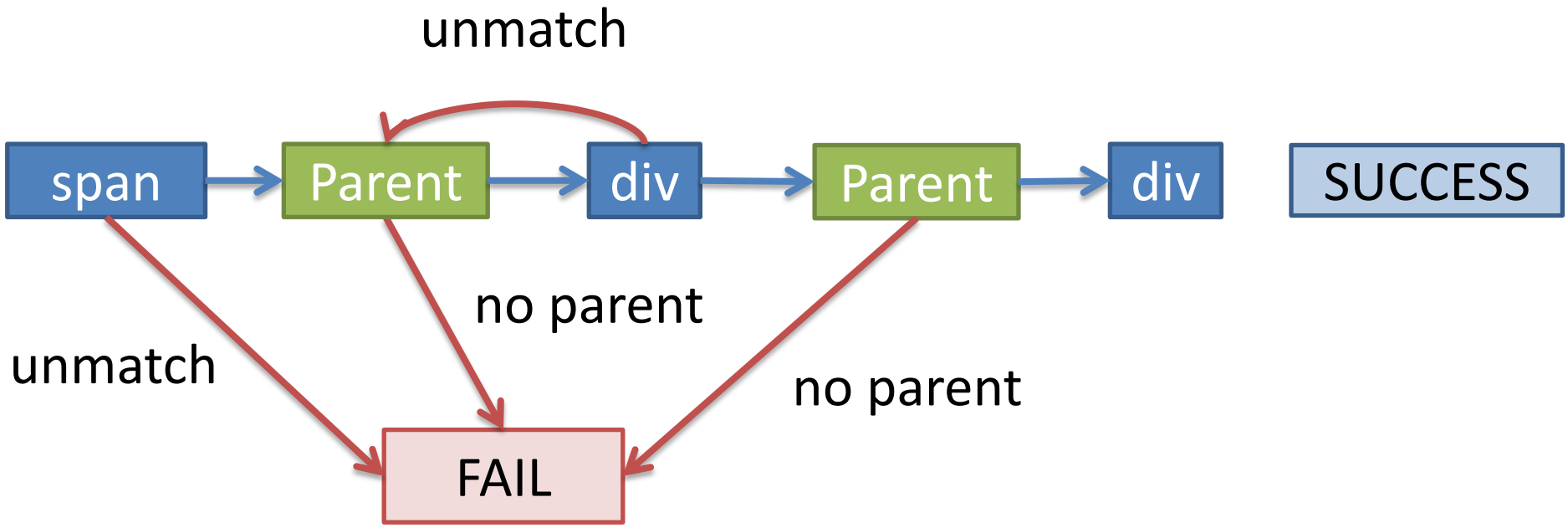
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

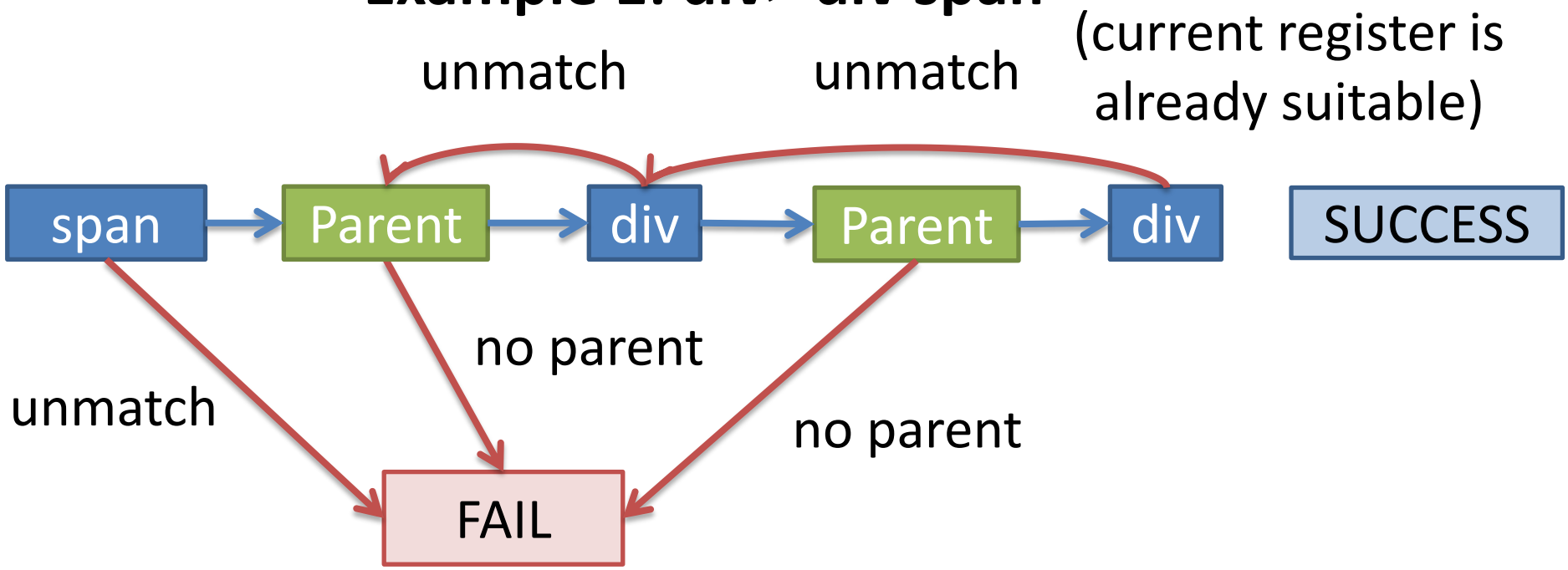
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

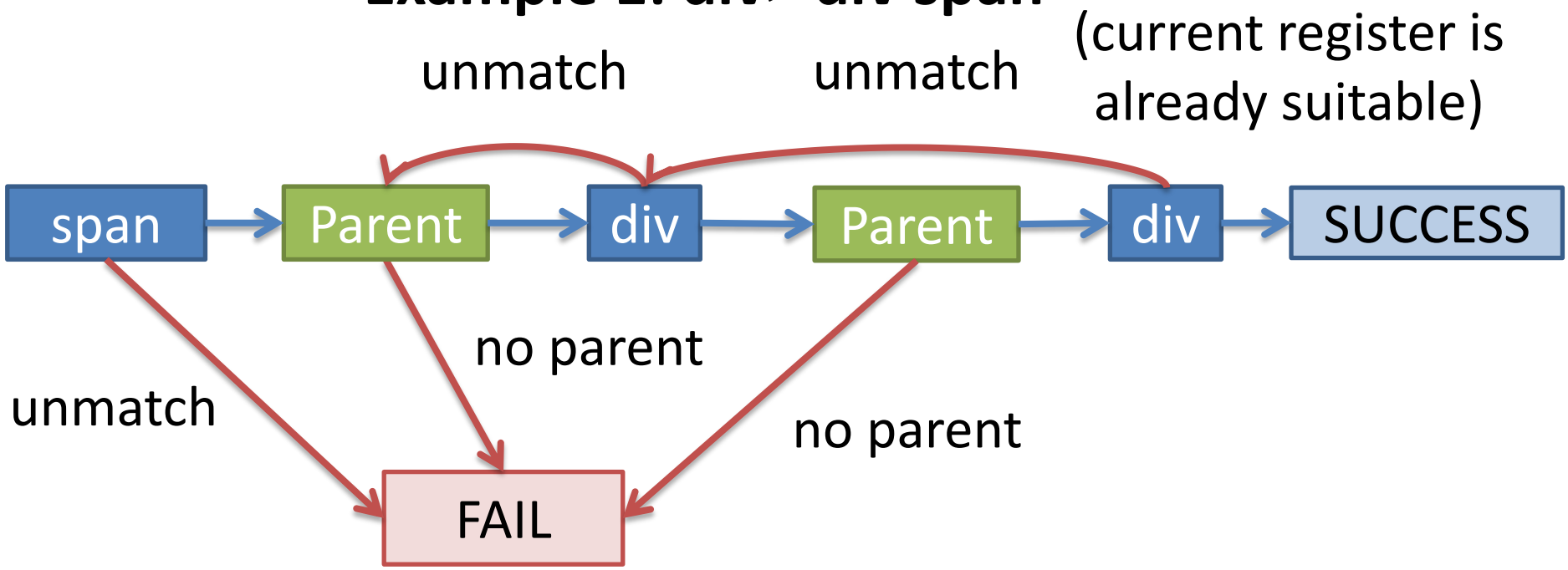
Example 1: div> div span



Compiling machine code

- Represent CSS Selector Matching with code & jumps
  - Avoid recursive function call (NO CALLS!)
  - Storing current element to register

Example 1: div> div span



Compiling machine code

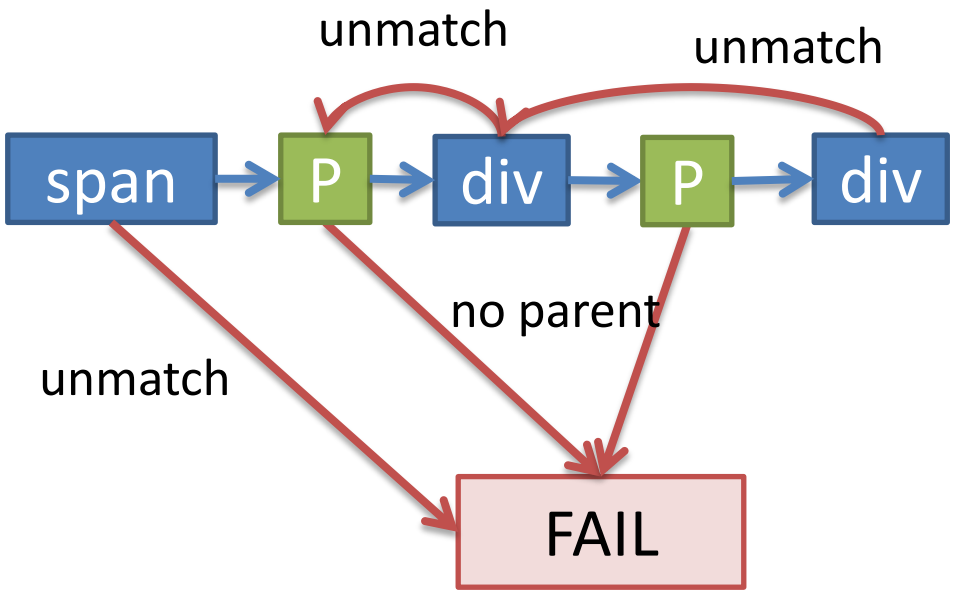
- Storing backtracking element to one register

Example 2: div> div> div span

Compiling machine code

- Storing backtracking element to one register

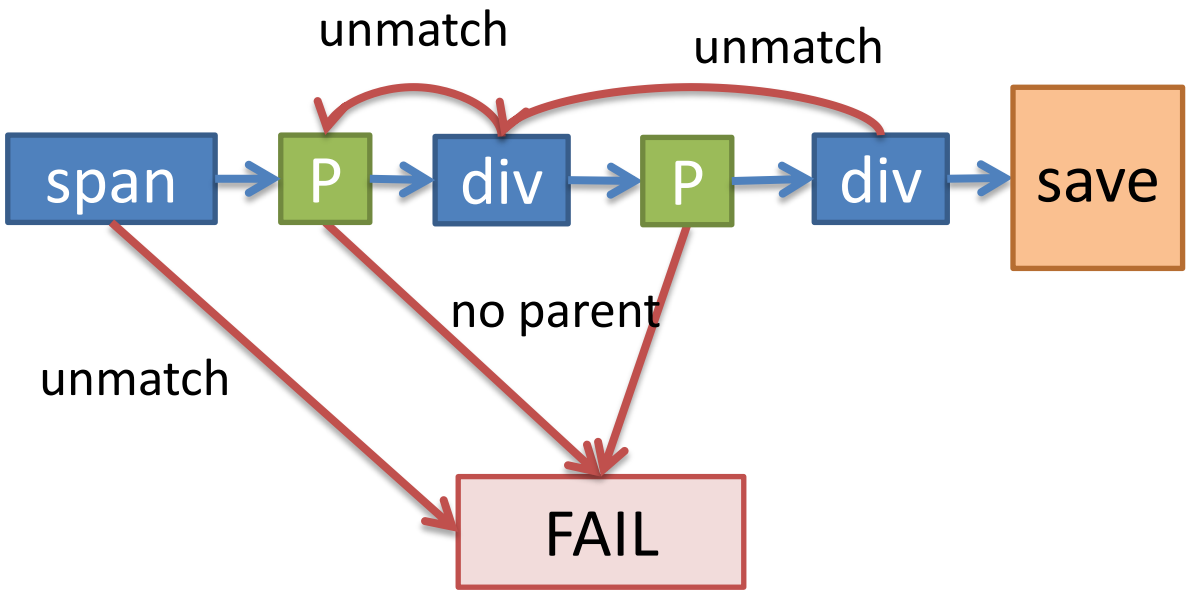
Example 2: div> div> div span



Compiling machine code

- Storing backtracking element to one register

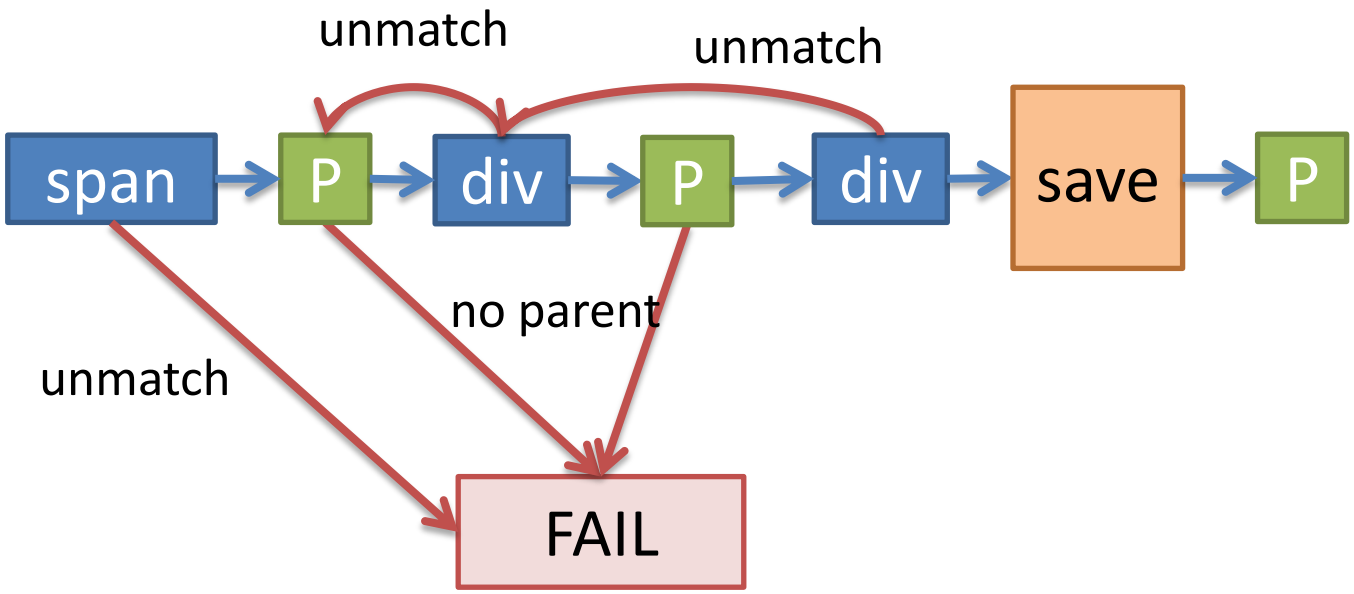
Example 2: div> div> div span



Compiling machine code

- Storing backtracking element to one register

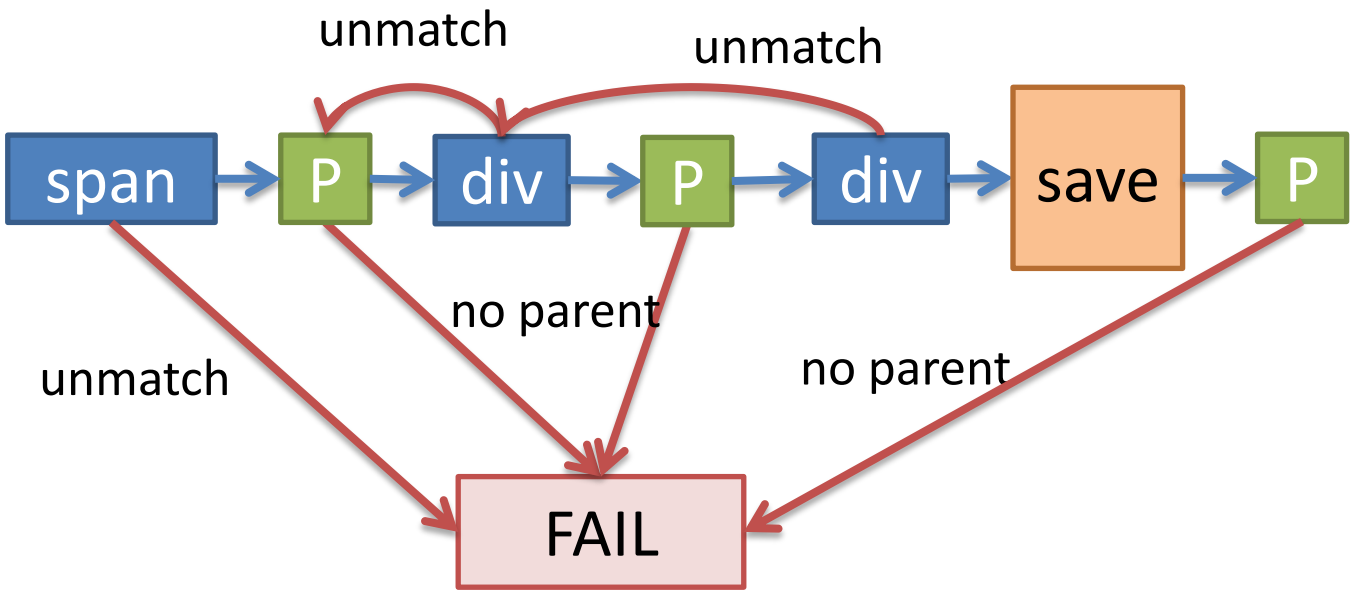
Example 2: div> div> div span



Compiling machine code

- Storing backtracking element to one register

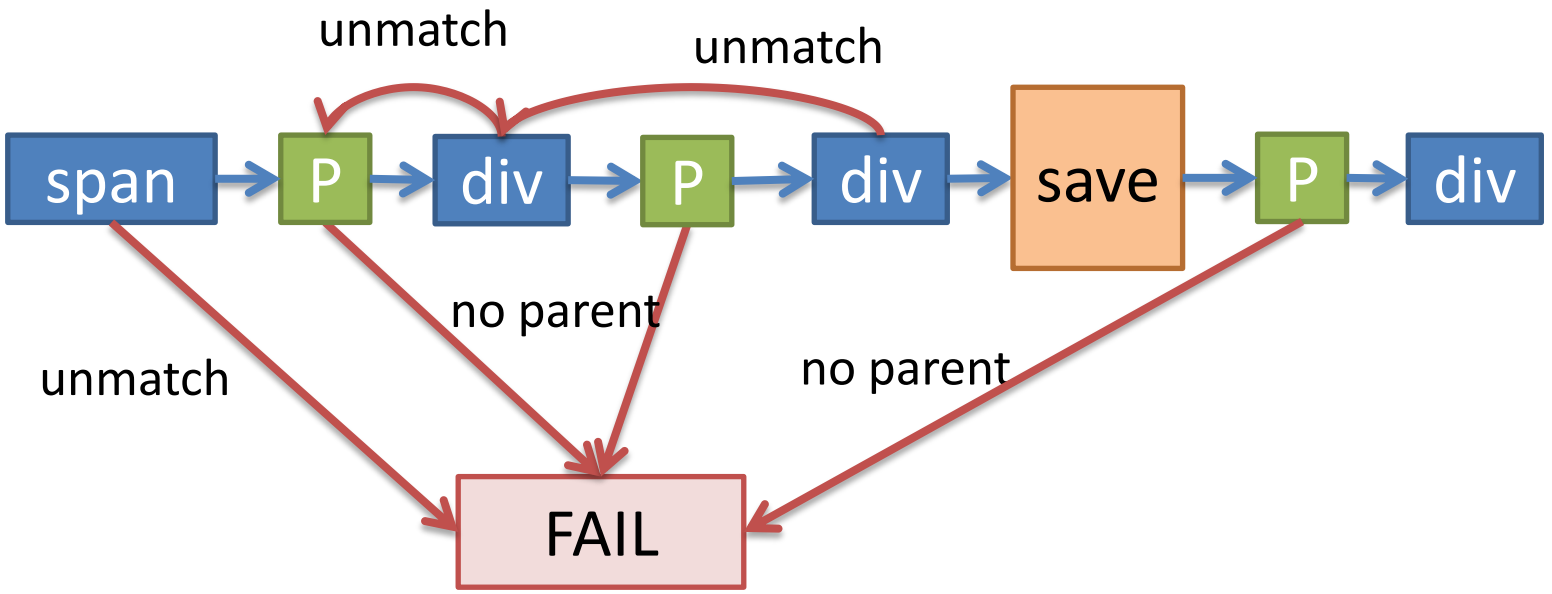
Example 2: div> div> div span



Compiling machine code

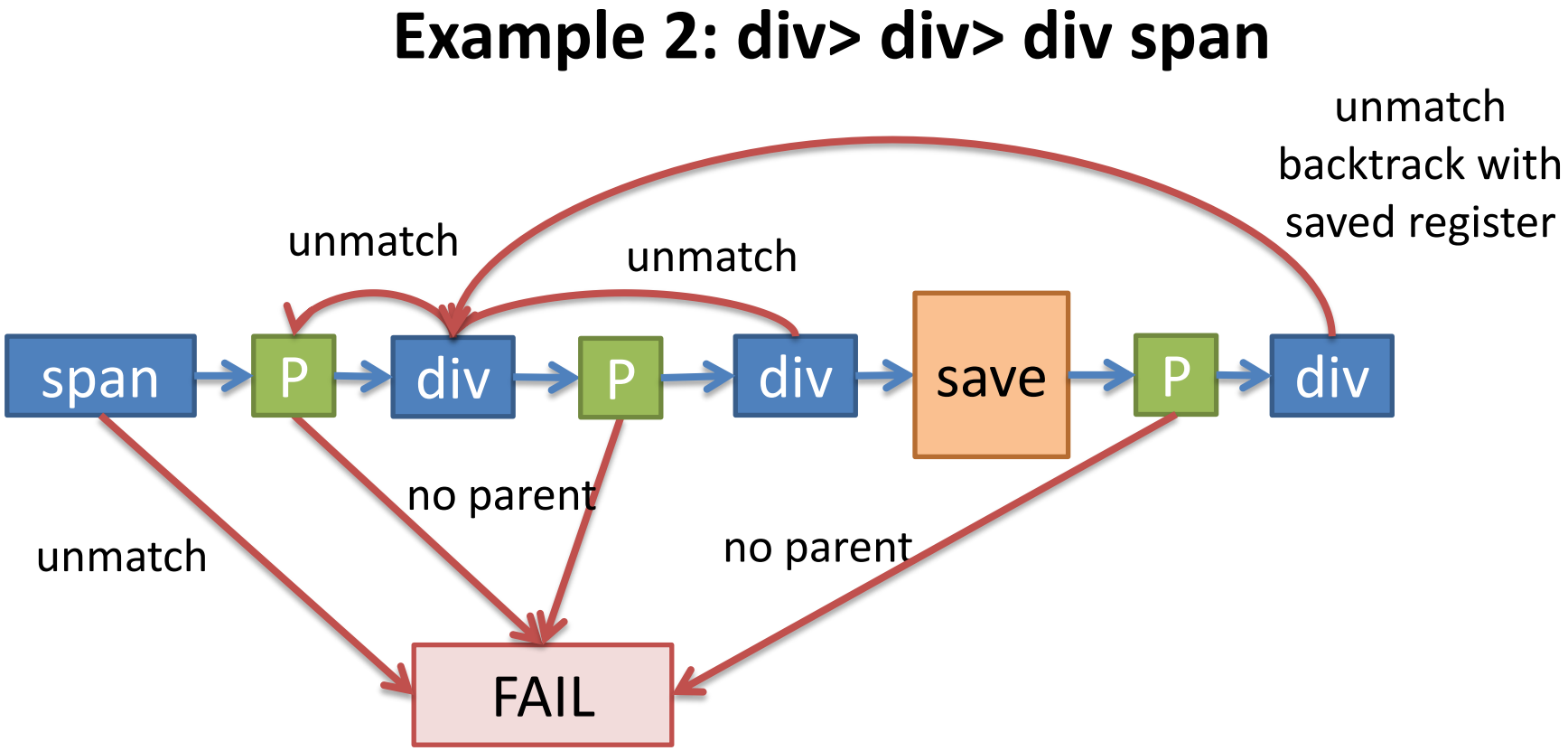
- Storing backtracking element to one register

Example 2: div> div> div span



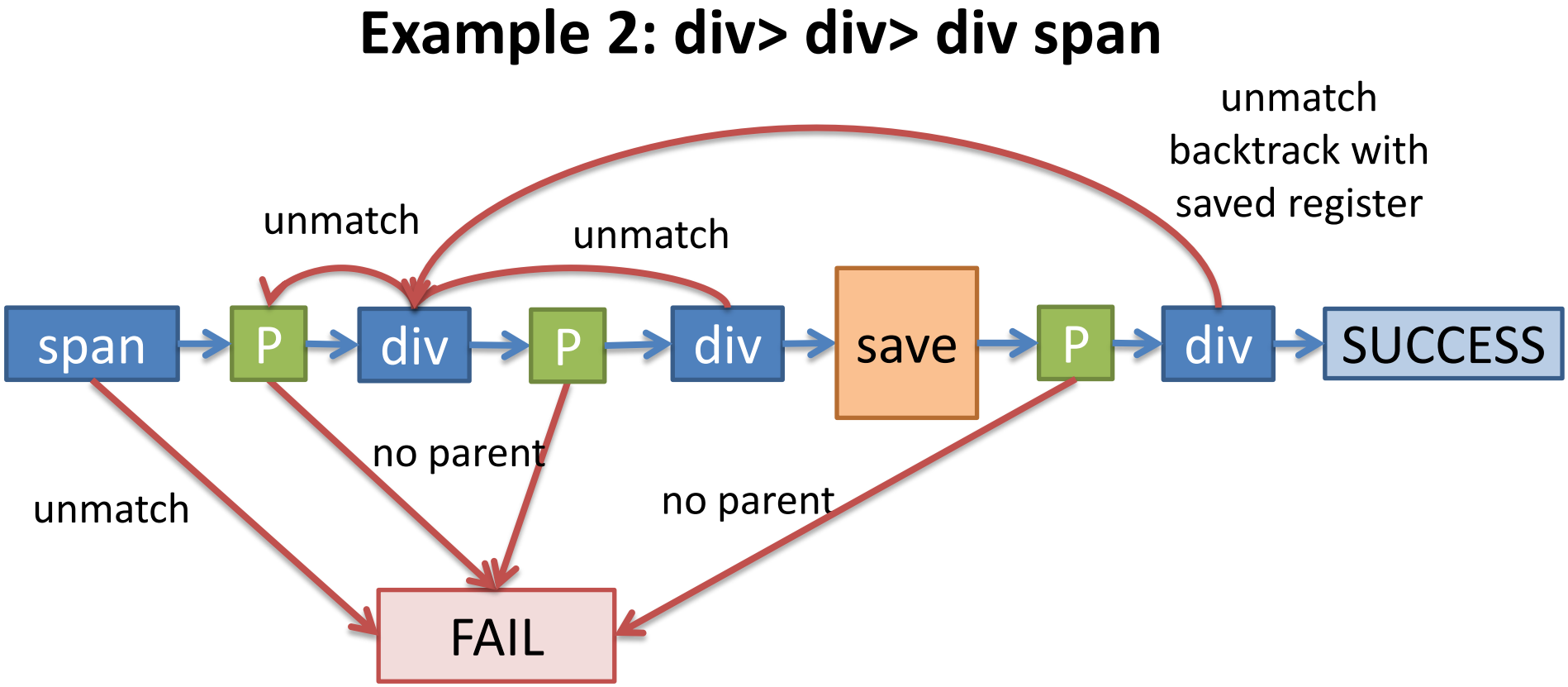
Compiling machine code

- Storing backtracking element to one register



Compiling machine code

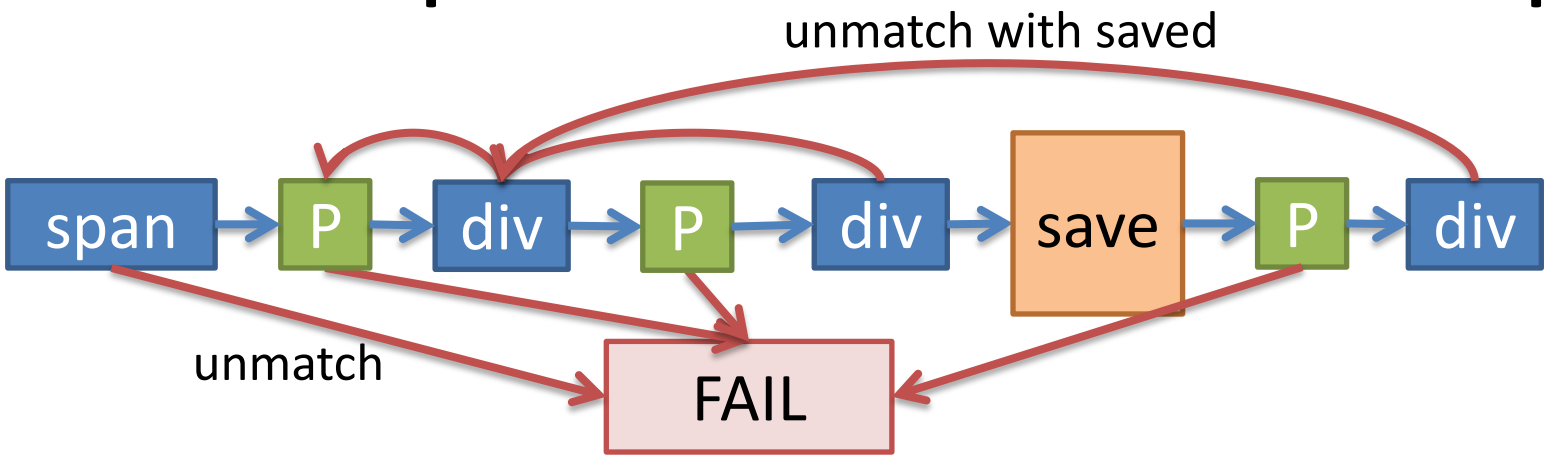
- Storing backtracking element to one register



Compiling machine code

- Storing backtracking element to **one** register
  - Since only one descendant should be considered

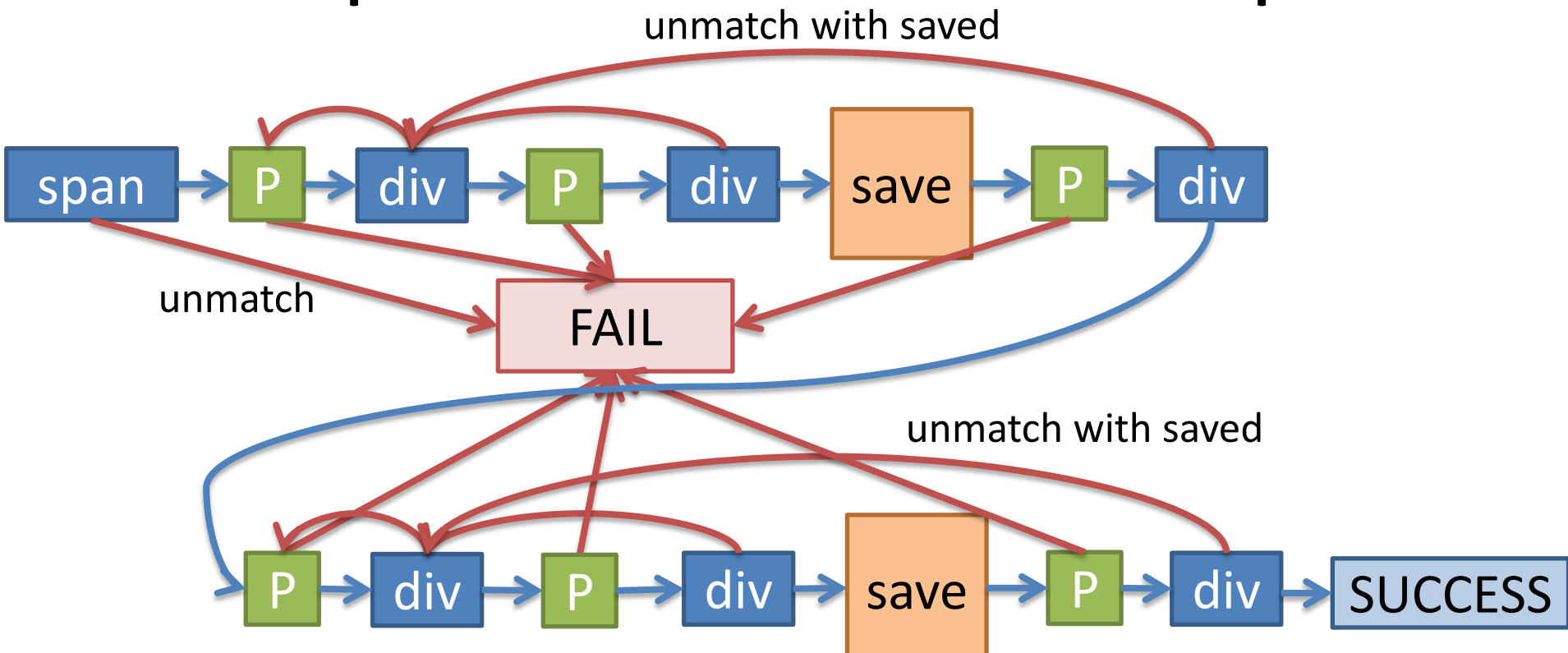
Example 3: div> div> div div> div> div span



Compiling machine code

- Storing backtracking element to **one** register
  - Since only one descendant should be considered

Example 3: div> div> div div> div> div span



div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd031187005: mov $0x7fd08c30ec50, %rcx
0x7fd03118700f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd03118701d: test %rdi, %rdi
0x7fd031187020: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd031187034: mov $0x7fd08c30f670, %rsi
0x7fd03118703e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd03118704c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd031187063: mov $0x7fd08c30f670, %r9
0x7fd03118706d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd[illegible]5: mov $0x7fd08c30ec50, %rcx
0x7fd[illegible]f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd03118701d: test %rdi, %rdi
0x7fd031187020: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd031187034: mov $0x7fd08c30f670, %rsi
0x7fd03118703e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd03118704c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd031187063: mov $0x7fd08c30f670, %r9
0x7fd03118706d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd[illegible]5: mov $0x7fd08c30ec50, %rcx
0x7fd[illegible]f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd03118701d: test %rdi, %rdi
0x7fd031187020: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd031187034: mov $0x7fd08c30f670, %rsi
0x7fd03118703e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd03118704c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd031187063: mov $0x7fd08c30f670, %r9
0x7fd03118706d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

FAIL

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd031187005: mov $0x7fd08c30ec50, %rcx
0x7fd03118700f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd03118701d: test %rdi, %rdi
0x7fd031187020: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd031187034: mov $0x7fd08c30f670, %rsi
0x7fd03118703e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd03118704c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd031187063: mov $0x7fd08c30f670, %r9
0x7fd03118706d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

parent

parent

FAIL

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd                5: mov $0x7fd08c30ec50, %rcx
0x7fd                f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd                d: test %rdi, %rdi
0x7fd                0: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd                4: mov $0x7fd08c30f670, %rsi
0x7fd                e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd                c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd031187063: mov $0x7fd08c30f670, %r9
0x7fd03118706d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

parent

div

parent

FAIL

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd[illegible]5: mov $0x7fd08c30ec50, %rcx
0x7fd[illegible]f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd[illegible]d: test %rdi, %rdi
0x7fd[illegible]0: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd[illegible]4: mov $0x7fd08c30f670, %rsi
0x7fd[illegible]e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd[illegible]c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd[illegible]3: mov $0x7fd08c30f670, %r9
0x7fd[illegible]d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
0x7fd031187077: mov $0x1, %eax
0x7fd03118707c: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

parent

div

parent

div

FAIL

div> div span Code

```
Generated JIT code for CSS Selector JIT
for "div > div span":
Code at [0x7fd031187000, 0x7fd0311870a0):
0x7fd031187000: push %rbp
0x7fd031187001: mov 0x58(%rdi), %rax
0x7fd[illegible]5: mov $0x7fd08c30ec50, %rcx
0x7fd[illegible]f: cmp %rcx, 0x18(%rax)
0x7fd031187013: jnz 0x7fd031187081
0x7fd031187019: mov 0x20(%rdi), %rdi
0x7fd[illegible]d: test %rdi, %rdi
0x7fd[illegible]0: jz 0x7fd031187081
0x7fd031187026: test $0x4, 0x1c(%rdi)
0x7fd03118702a: jz 0x7fd031187081
0x7fd031187030: mov 0x58(%rdi), %rdx
0x7fd[illegible]4: mov $0x7fd08c30f670, %rsi
0x7fd[illegible]e: cmp %rsi, 0x18(%rdx)
0x7fd031187042: jnz 0x7fd031187019
0x7fd031187048: mov 0x20(%rdi), %rdi
0x7fd[illegible]c: test %rdi, %rdi
0x7fd03118704f: jz 0x7fd031187081
0x7fd031187055: test $0x4, 0x1c(%rdi)
0x7fd031187059: jz 0x7fd031187081
0x7fd03118705f: mov 0x58(%rdi), %r8
0x7fd[illegible]3: mov $0x7fd08c30f670, %r9
0x7fd[illegible]d: cmp %r9, 0x18(%r8)
0x7fd031187071: jnz 0x7fd031187019
[illegible]: mov $0x1, %eax
[illegible]: jmp 0x7fd031187083
0x7fd031187081: xor %eax, %eax
0x7fd031187083: pop %rbp
0x7fd031187084: ret
```

span

parent

div

parent

div

SUCCESS

FAIL

More intelligent backtracking

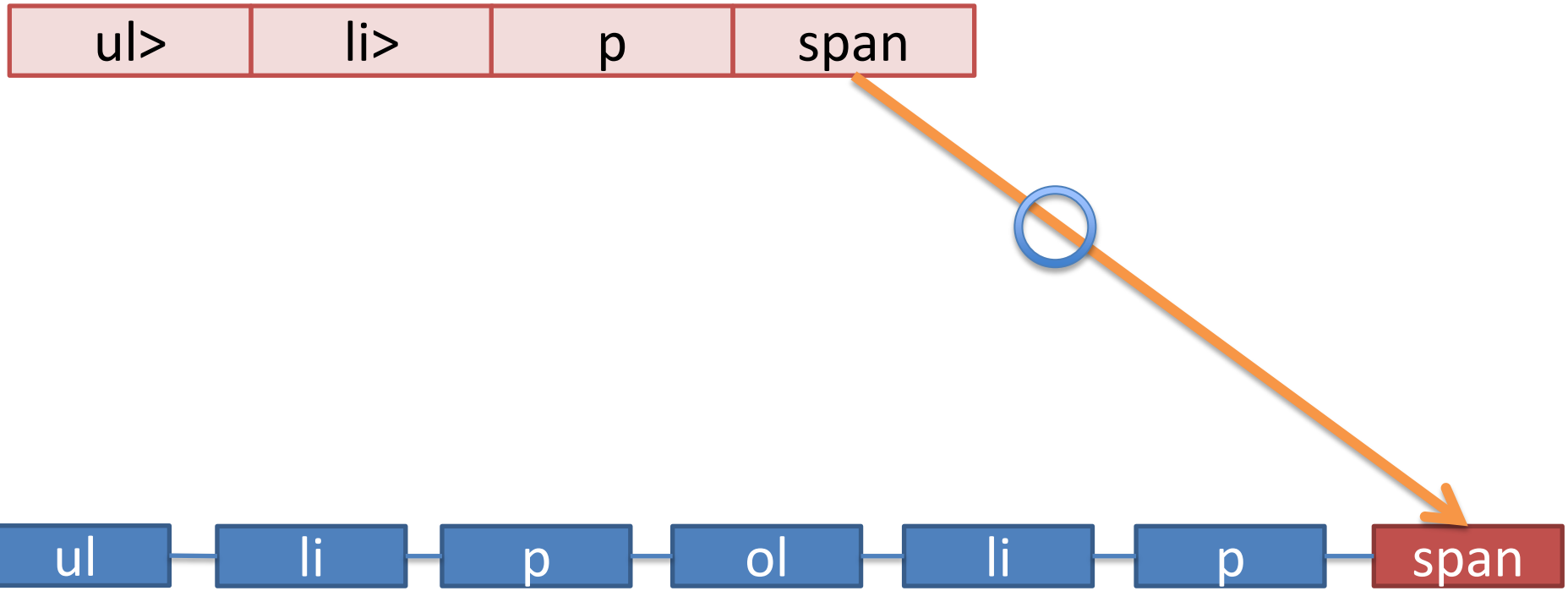
- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

Example 4: ul> li> p span

More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

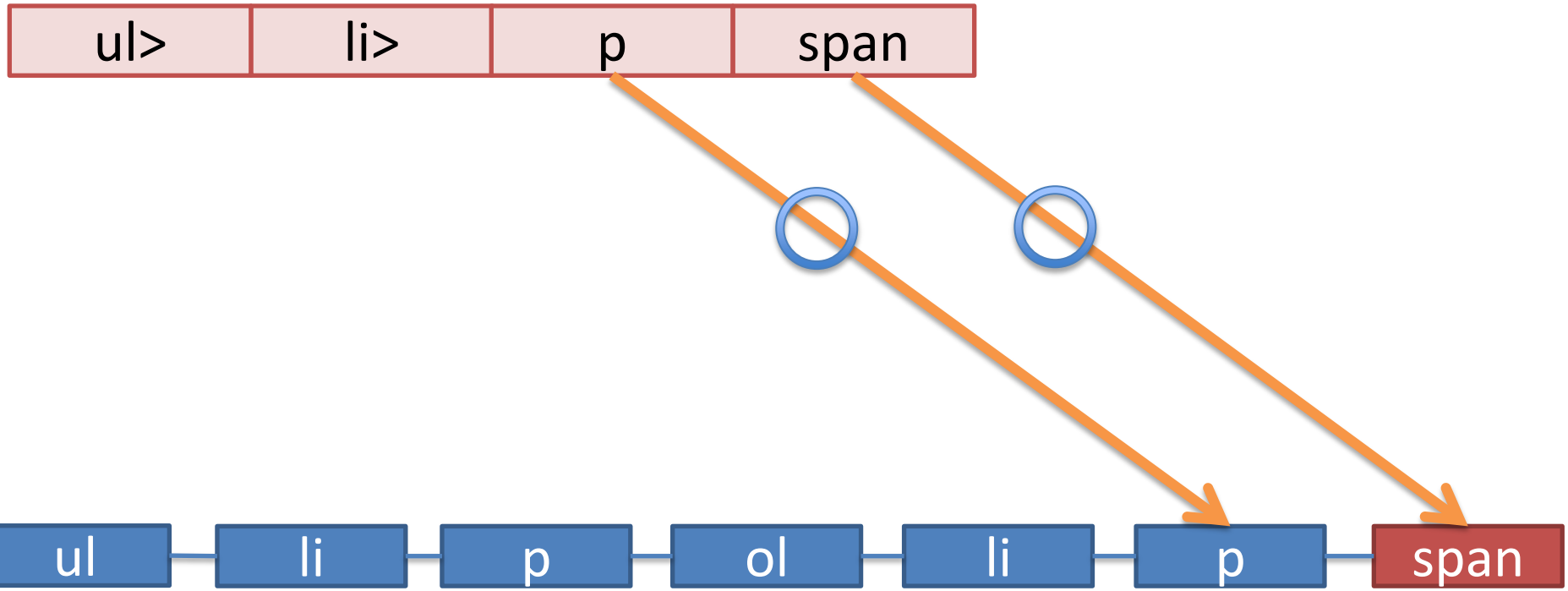
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

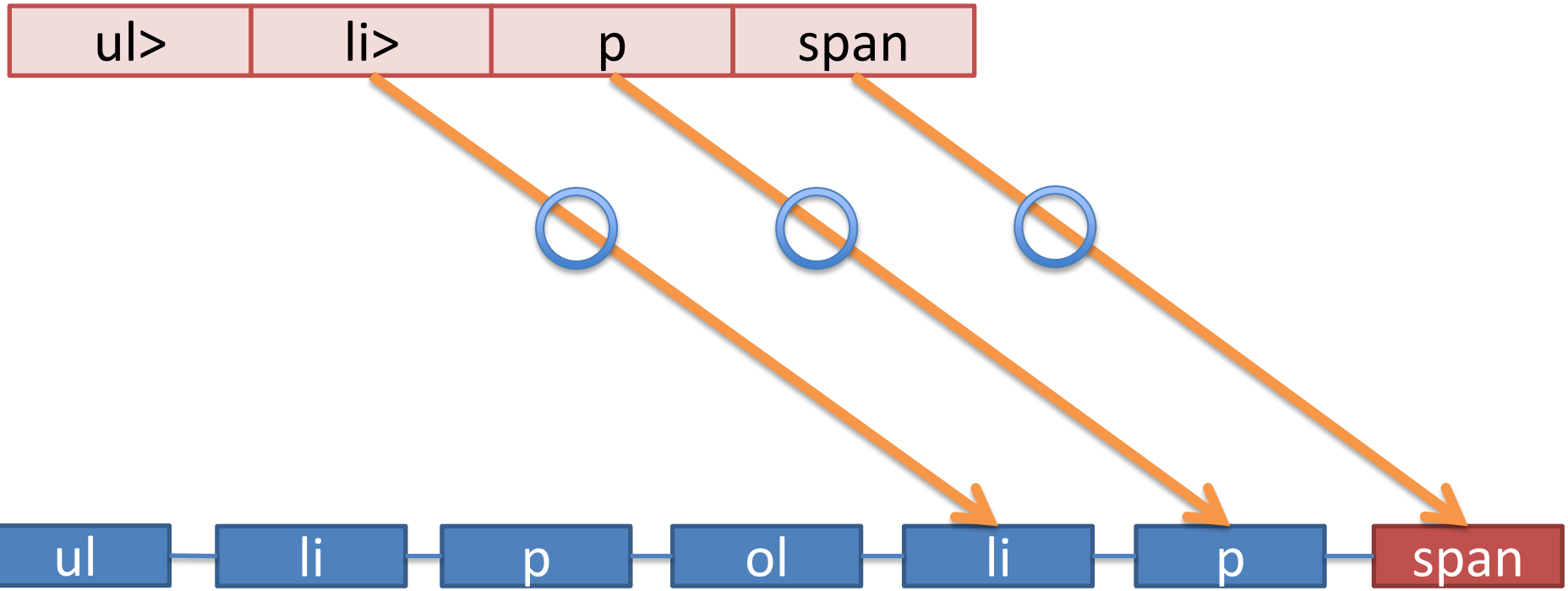
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

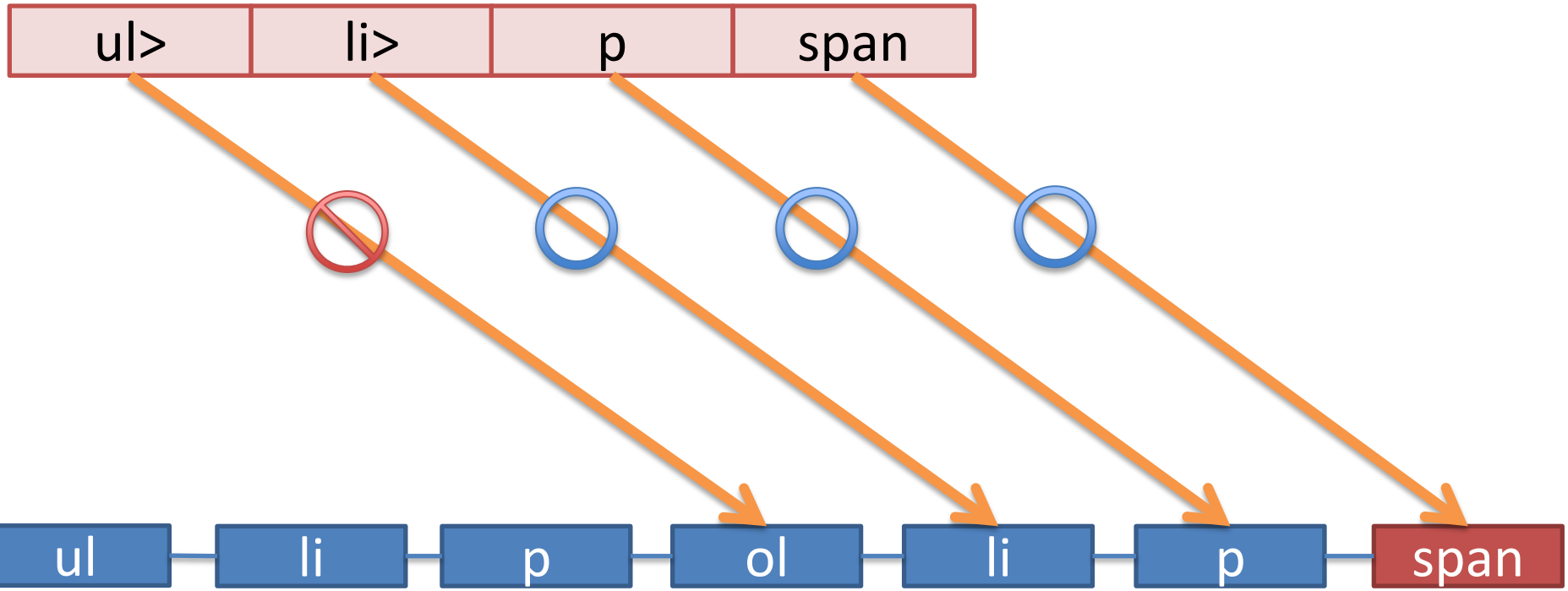
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

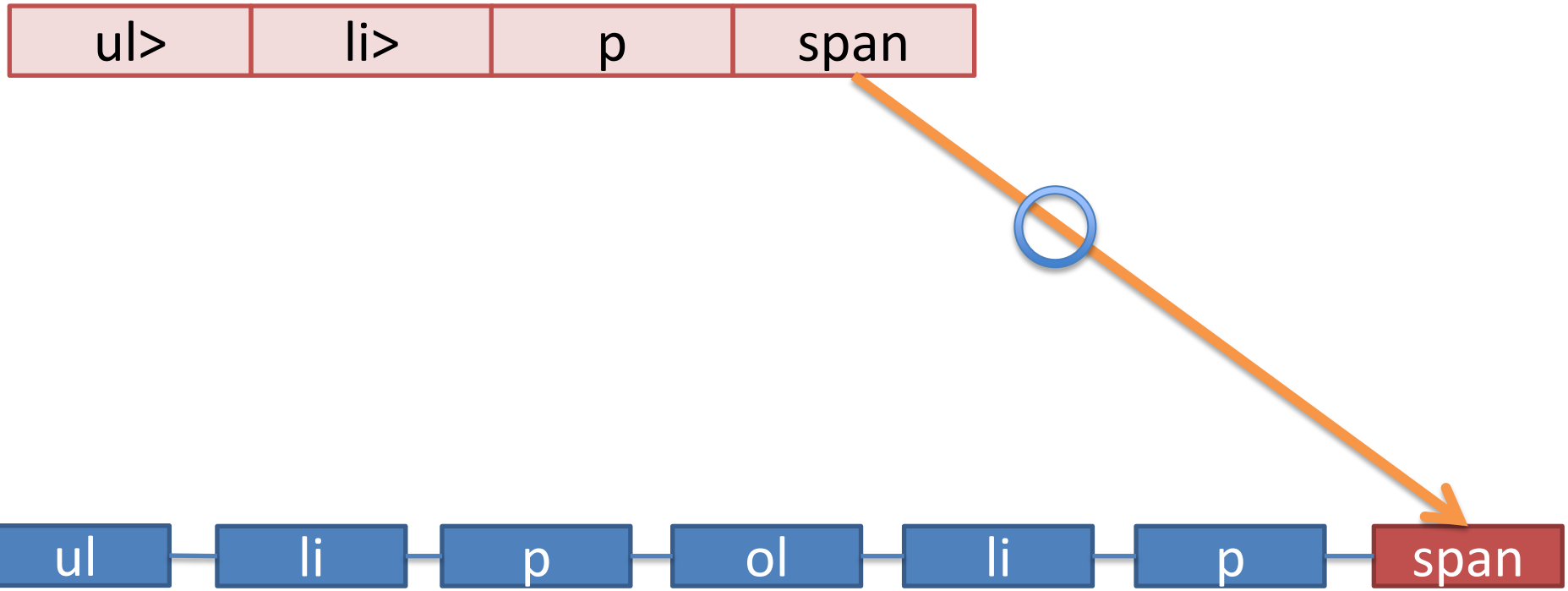
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

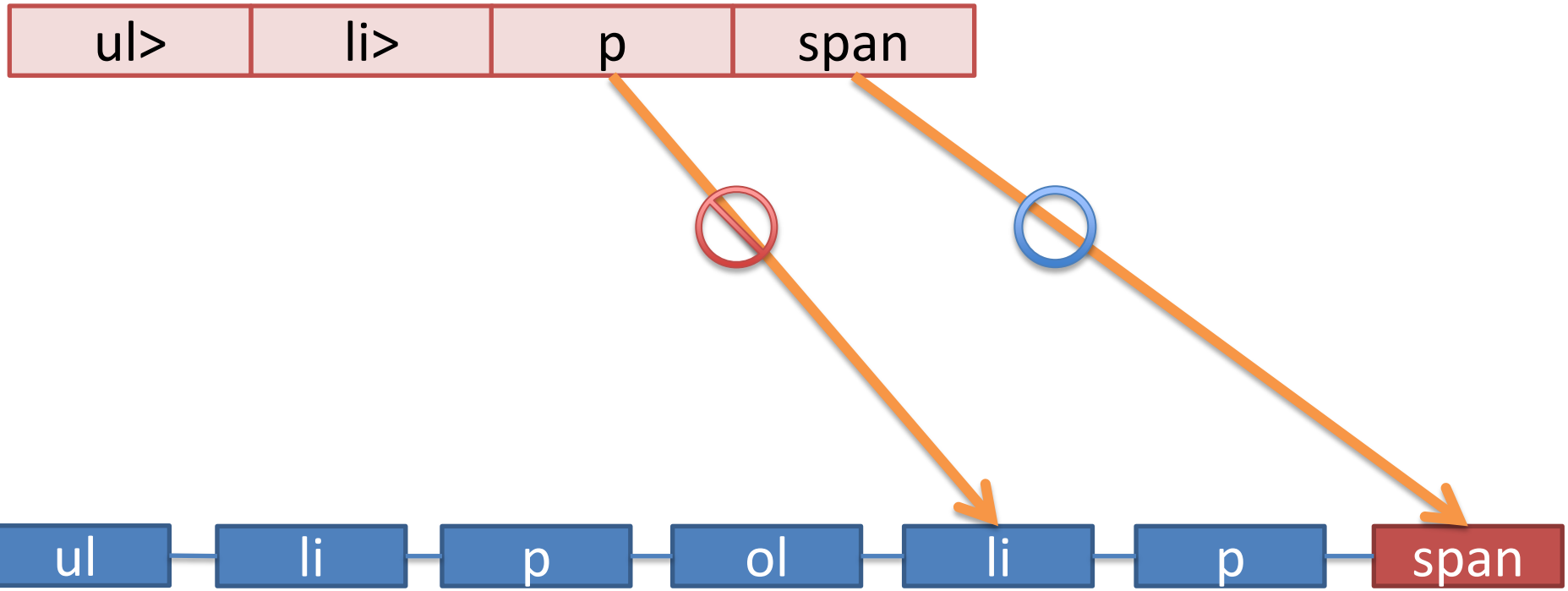
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

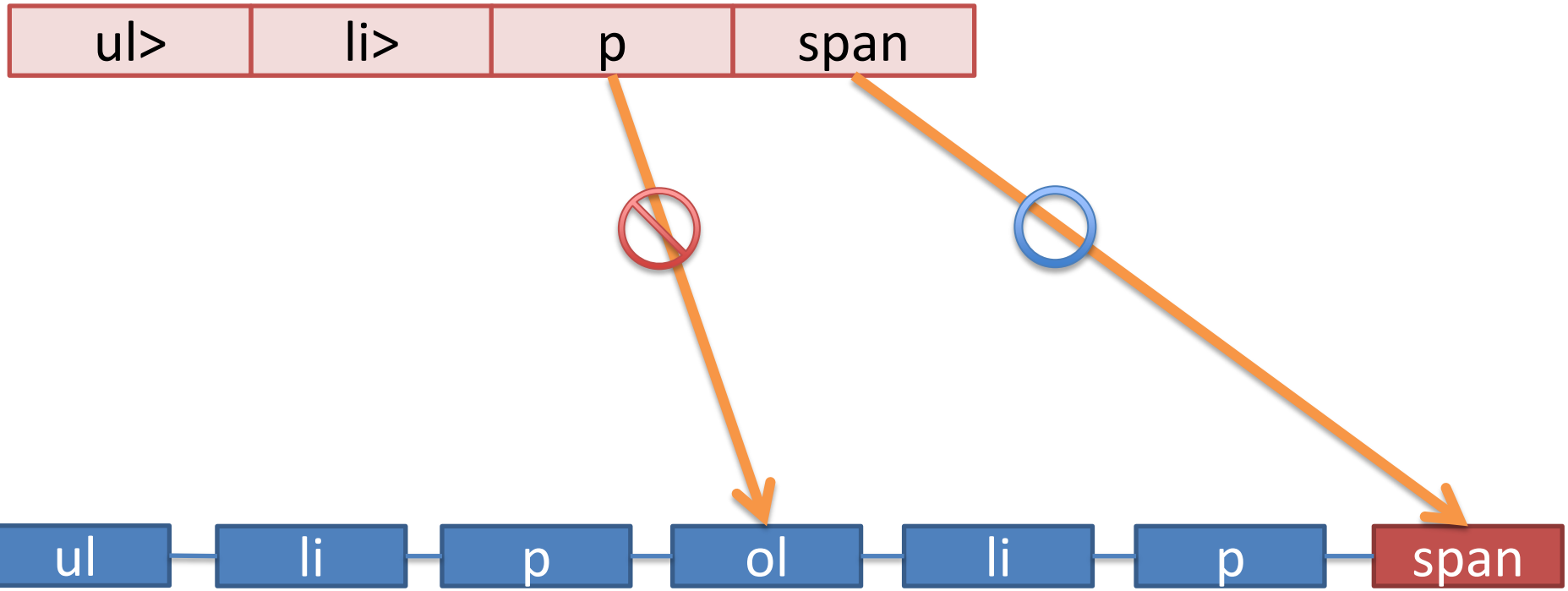
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

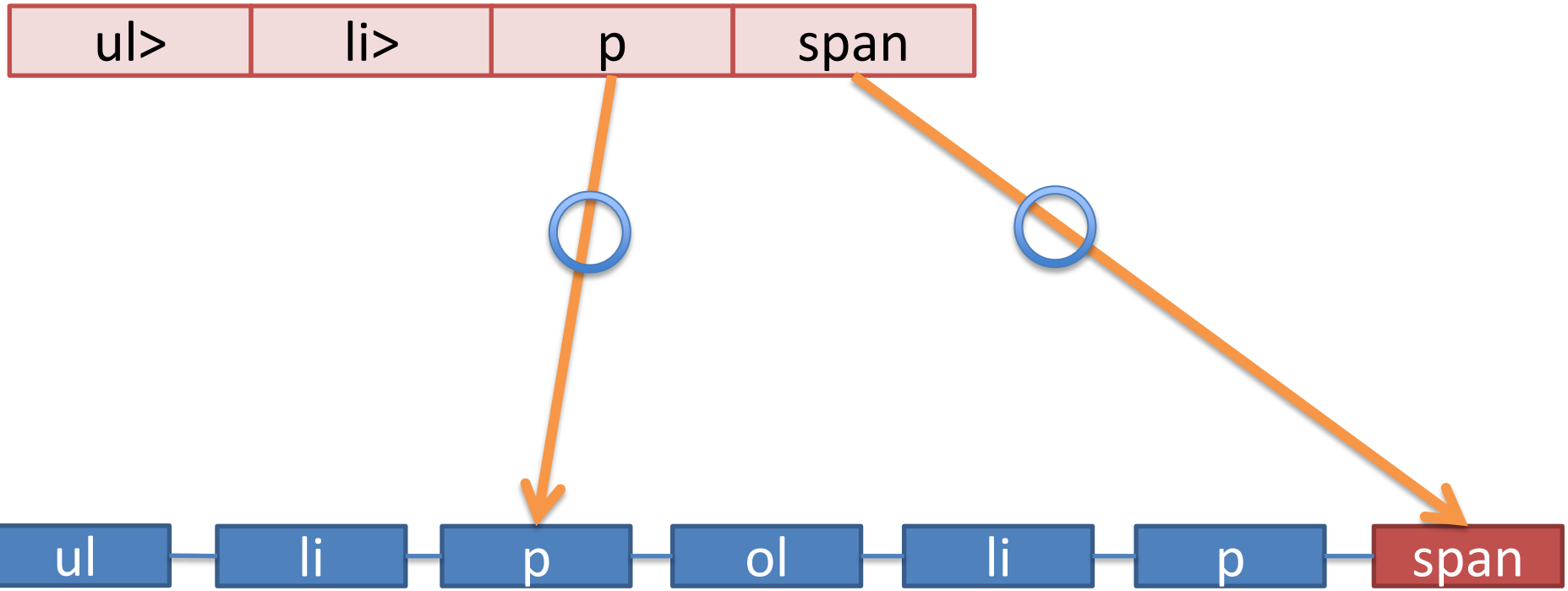
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

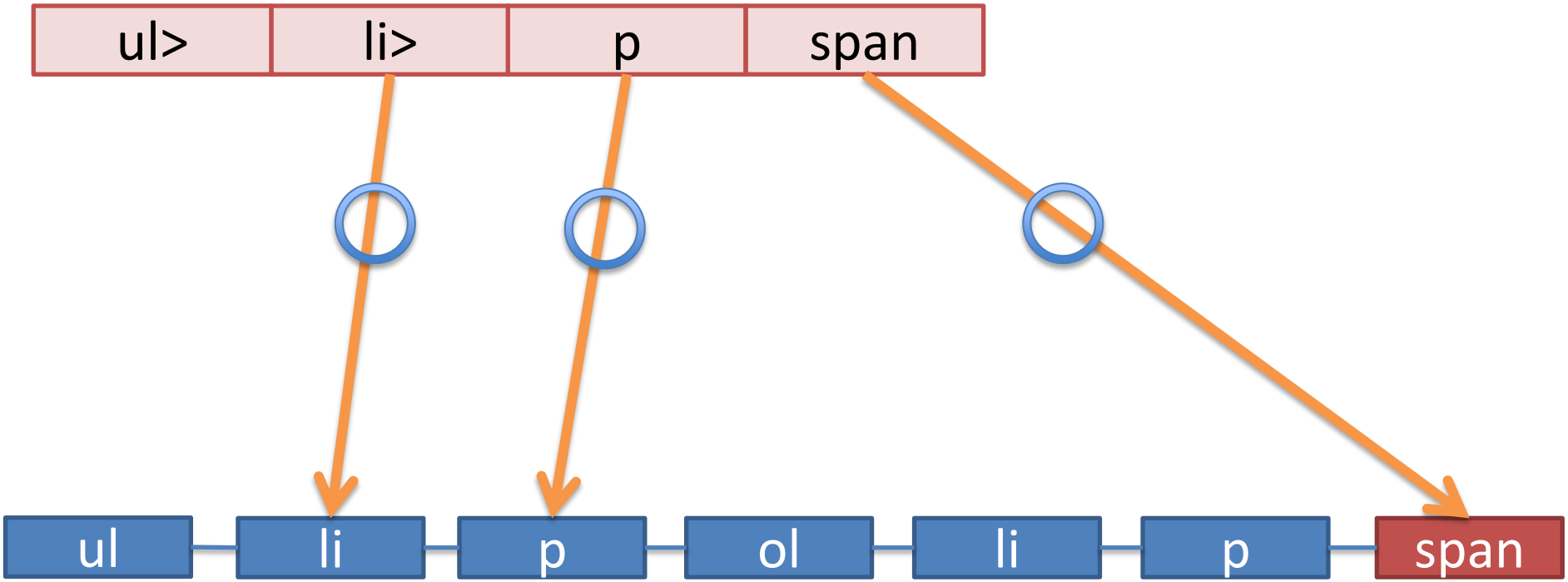
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

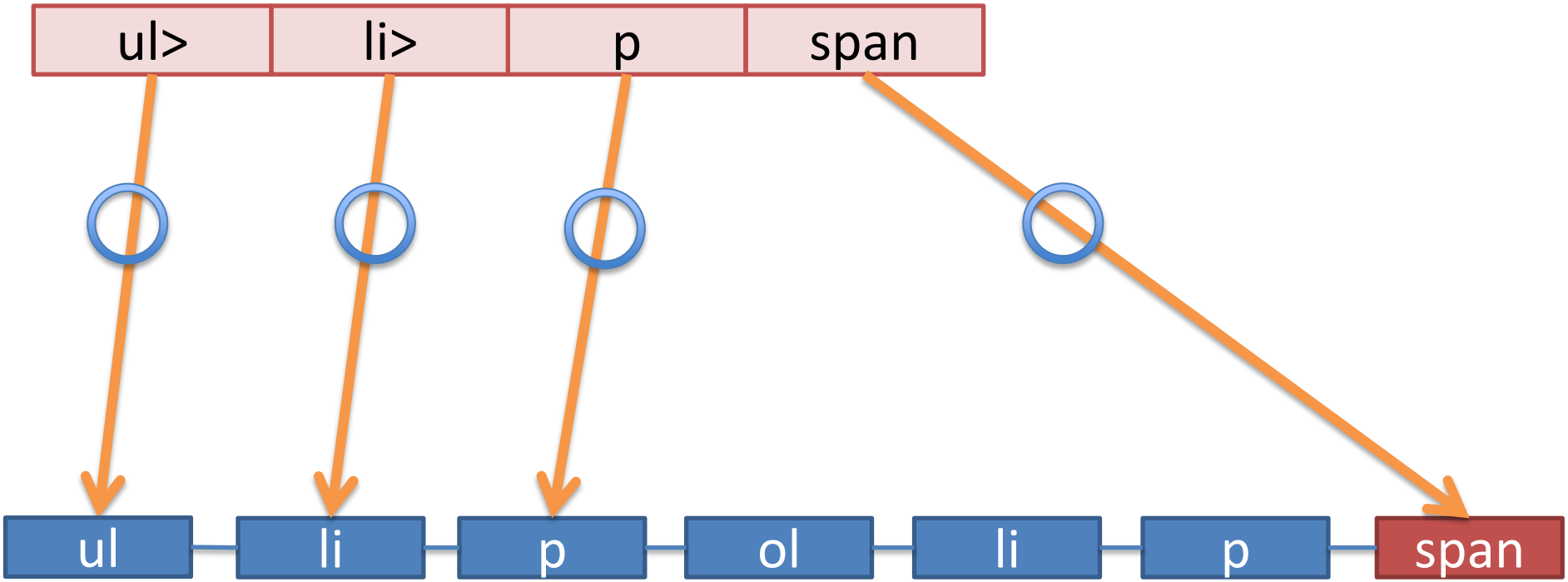
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

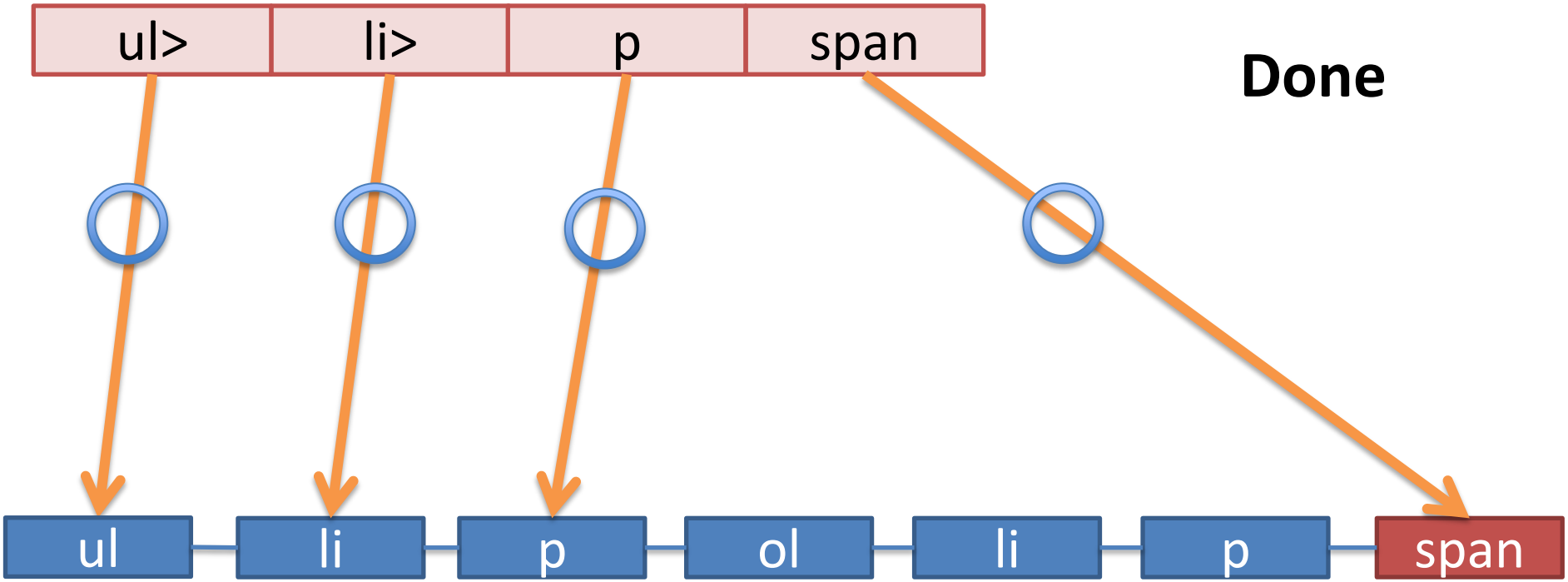
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

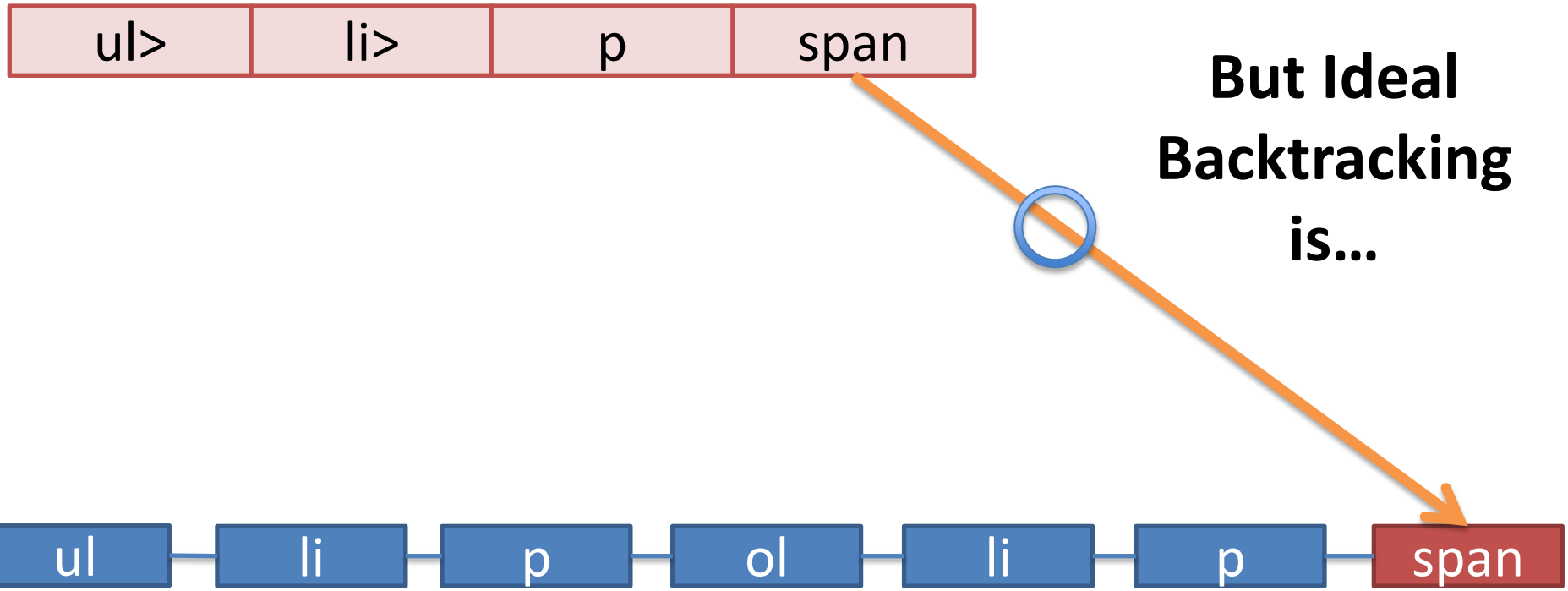
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

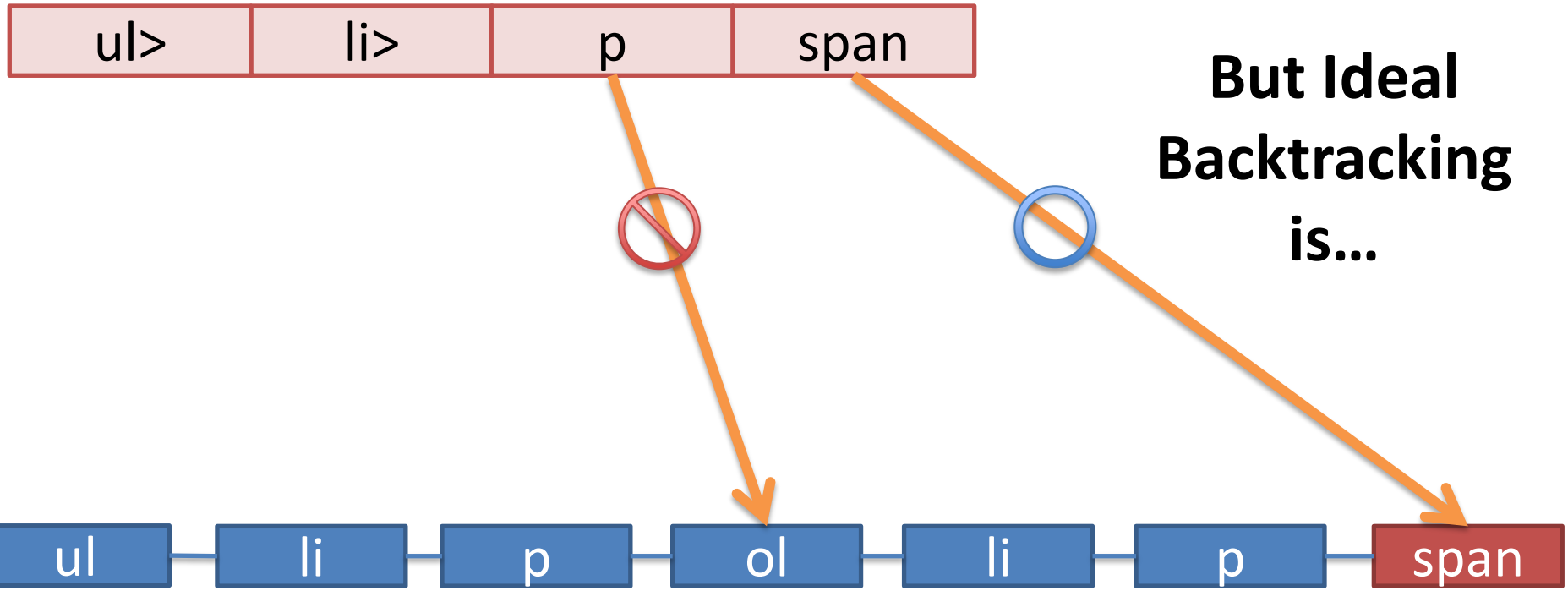
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

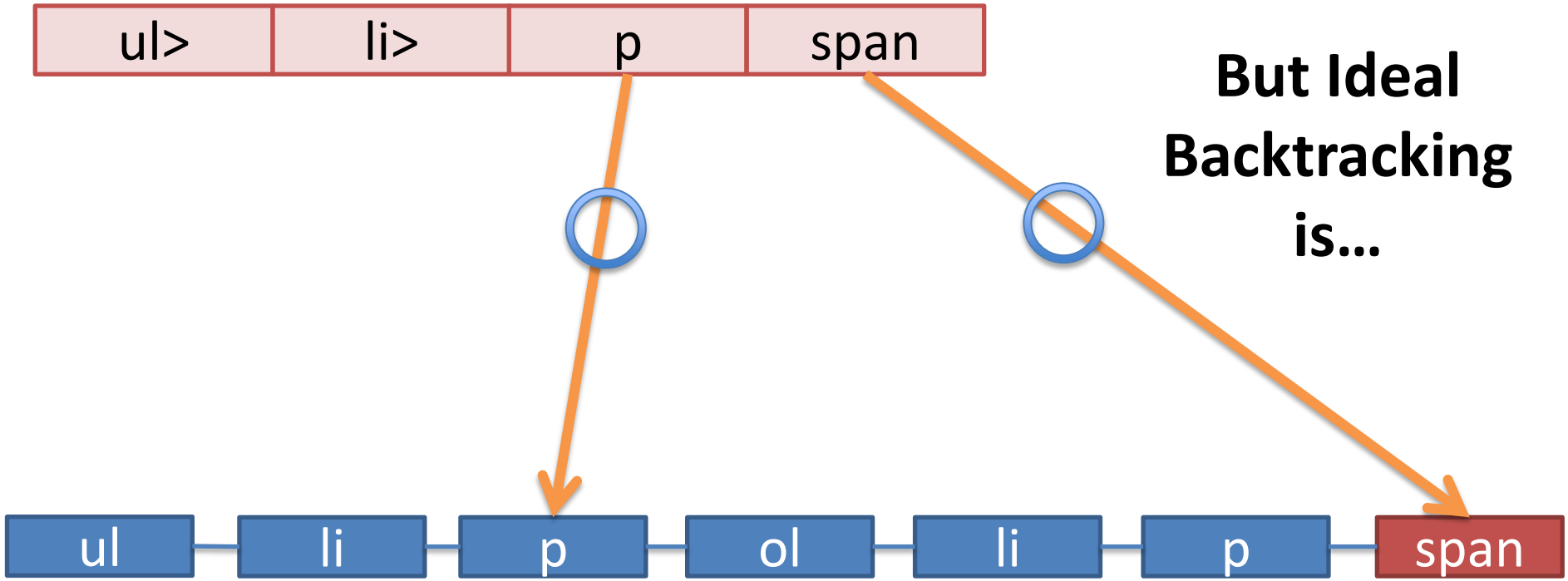
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

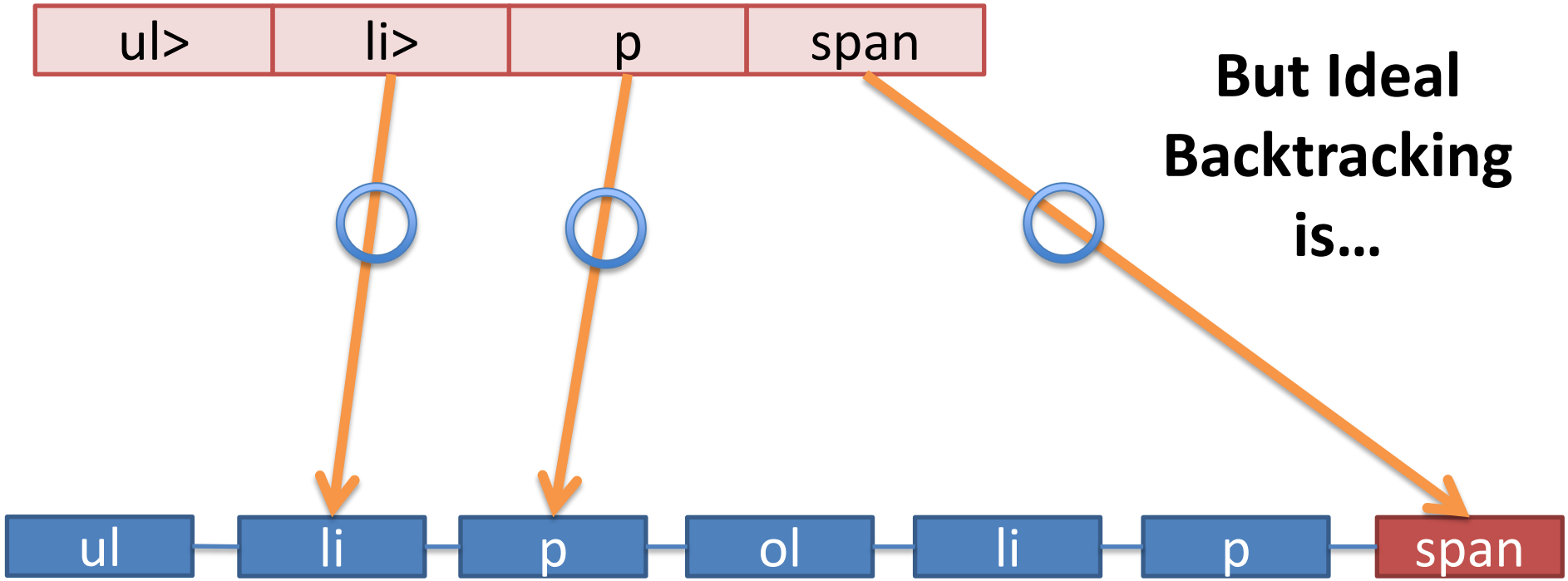
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

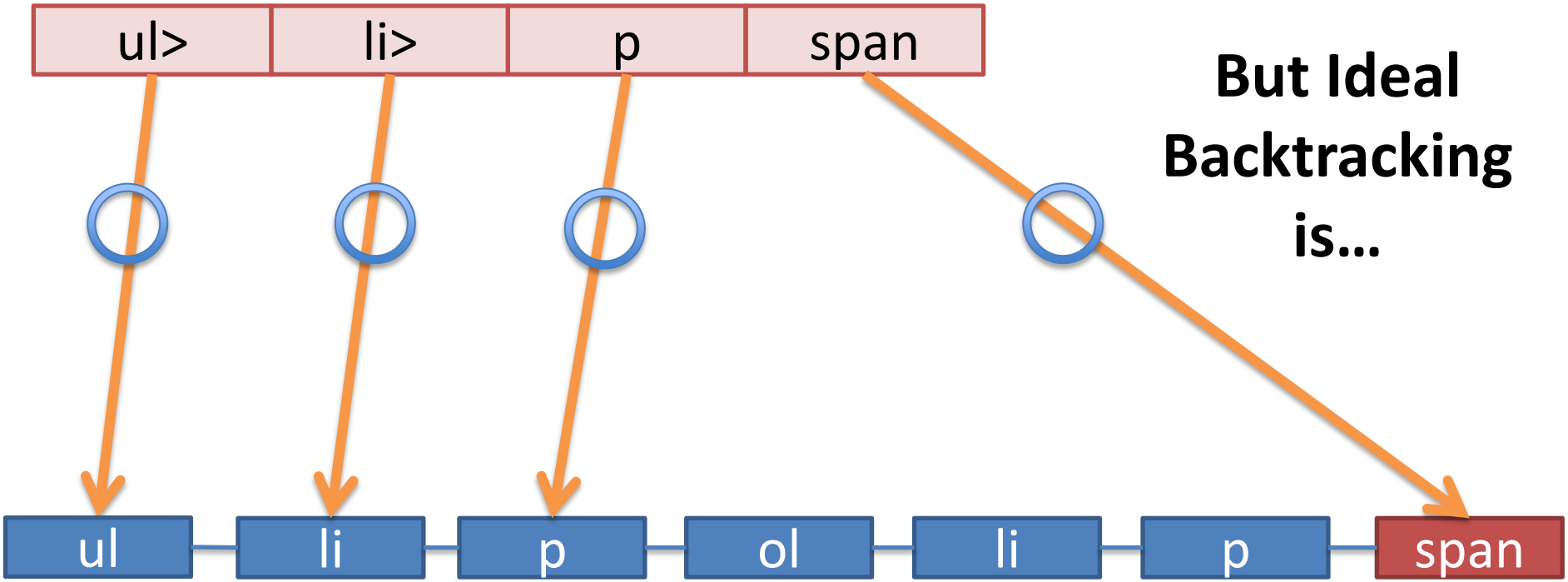
Example 4: ul> li> p span



More intelligent backtracking

- Leveraging this static information, provide more intelligent backtracking
  - This is my largest contribution to WebKit CSS JIT

Example 4: ul> li> p span

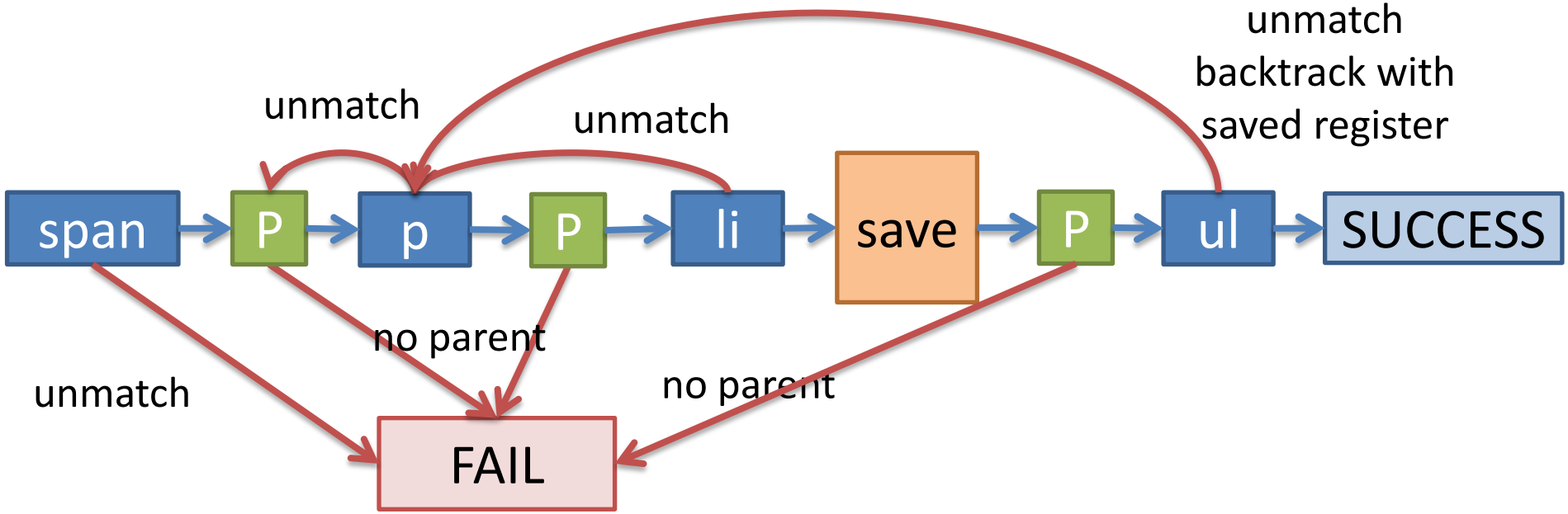


Computing Backtracking height

- Pre-compute appropriate backtracking height

Example 4: ul> li> p span

	ul>	li>	p	span
height	2	1	0	0
pre-tag	2	1	✗	✗

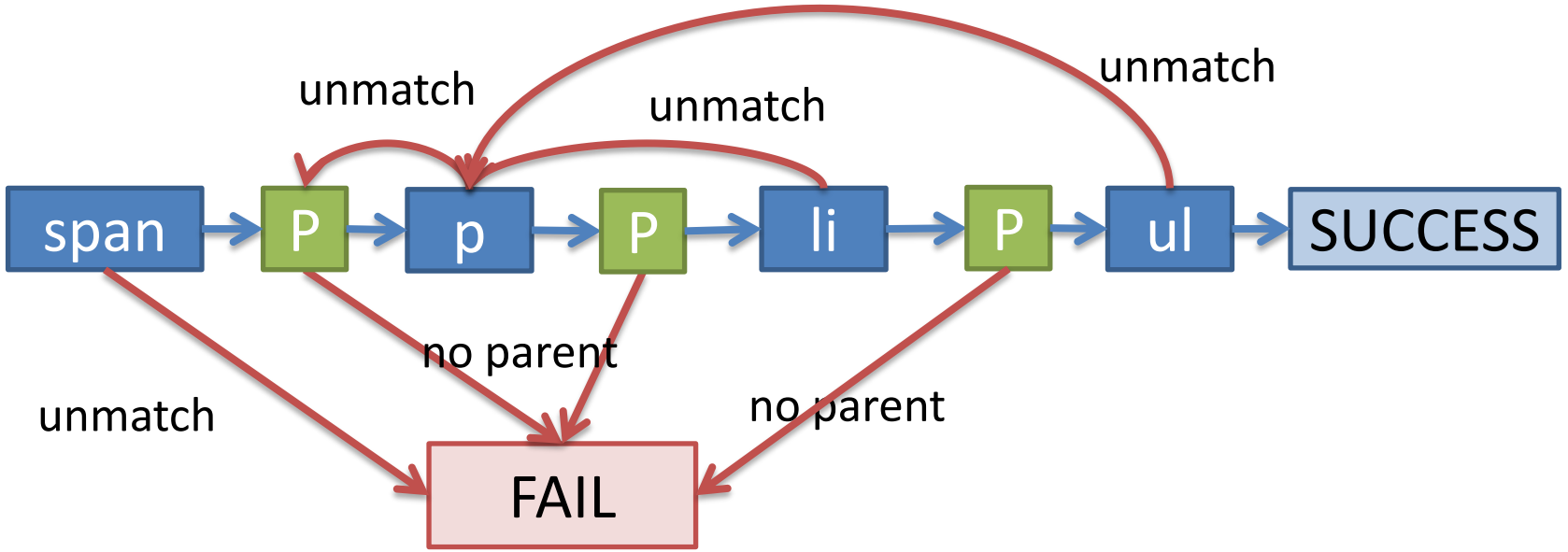


Computing Backtracking height

- Pre-compute appropriate backtracking height

Example 4: ul> li> p span

	ul>	li>	p	span
height	2	1	0	0
pre-tag	2	1	✗	✗



Outline

- Motivation & Goals
- Existing CSS Selector Implementation
- CSS Selector JIT
- Conclusion

Status

- x64 / ARMv7s / ARMv7 / ARM64 backend
- More intelligent backtracking is now planned
  - Most of implementation is done
  - Before implementing it, I'll instrument/profile the impact of this
- Seeing the Safari release branch…
  - Maybe shipped with the next OSX
  - ~~not in iOS8… See you next year, CSS Selector JIT!~~
  - In iOS 8 beta3! Hello CSS Selector JIT in your iPhone!

Conclusion

- Selector matching is crucial
- CSS Selector JIT brings up to the next stage
  - Avoid function calls
  - Store almost all to the registers
  - Provide faster performance
- Intelligent backtracking
  - Reduces register pressure
  - Avoid unnecessary backtraking