

SynPEG: Synthesizing Parsing Expression Grammars via Analysis-based Pruning

Seongmin Ko and Jihyeok Park



KOREA
UNIVERSITY



Parsing Expression Grammars (PEGs)

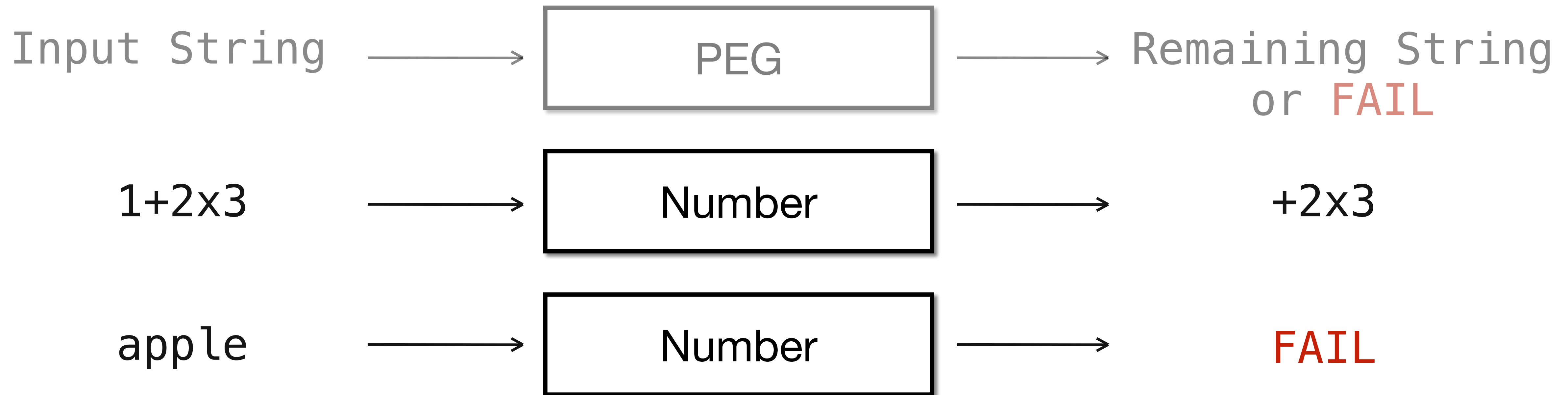
Parsing Expression Grammars: A Recognition-Based Syntactic Foundation

Bryan Ford
Massachusetts Institute of Technology
Cambridge, MA
baford@mit.edu

Bryan Ford, POPL 2004

A formal grammar & A parsing program

Parsing Expression Grammars (PEGs)



PEG in Practice - Python

- Python 3.9 grammar

```
# STARTING RULES
# =====

file: [statements] ENDMARKER
interactive: statement_newline
eval: expressions NEWLINE* ENDMARKER
func_type: '(' [type_expressions] ')' '->' expression NEWLINE* ENDMARKER

# GENERAL STATEMENTS
# =====

statements: statement+

statement:
| compound_stmt
| simple_stmts

single_compound_stmt:
| compound_stmt

statement_newline:
| single_compound_stmt NEWLINE
| simple_stmts
| NEWLINE
| ENDMARKER
```

```
simple_stmts:
| simple_stmt !';' NEWLINE # Not needed, there for speedup
| ';' .simple_stmt+ [';'] NEWLINE

# NOTE: assignment MUST precede expression, else parsing a simple assignment
# will throw a SyntaxError.
simple_stmt:
| assignment
| type_alias
| star_expressions
| return_stmt
| import_stmt
| raise_stmt
| pass_stmt
| del_stmt
| yield_stmt
| assert_stmt
| break_stmt
| continue_stmt
| global_stmt
| nonlocal_stmt
```

PEG in Practice - PEG Libraries

```
import scala.util.parsing.combinator.*  
  
object ArithExpr extends RegexParsers {  
  lazy val number: Parser[Number]    = "[0-9]+".r  
  lazy val product: Parser[Product] = number ~ rep("x" ~> number)  
  lazy val expr: Parser[Expr]        = product ~ rep("+ ~> product)  
}
```



PEG in Practice - PEG Libraries

```
import scala.util.parsing.combinator.*
```

```
object ArithExpr
```

```
  lazy val num
```

```
  lazy val pro
```

```
  lazy val exp
```

```
}
```

1. Expressive and Readable Grammar

2. Deterministic, Unambiguous Parsing

3. Linear Parsing

But there are some pitfalls!

1+2x3



expr



ϵ

Pitfalls in Writing PEGs

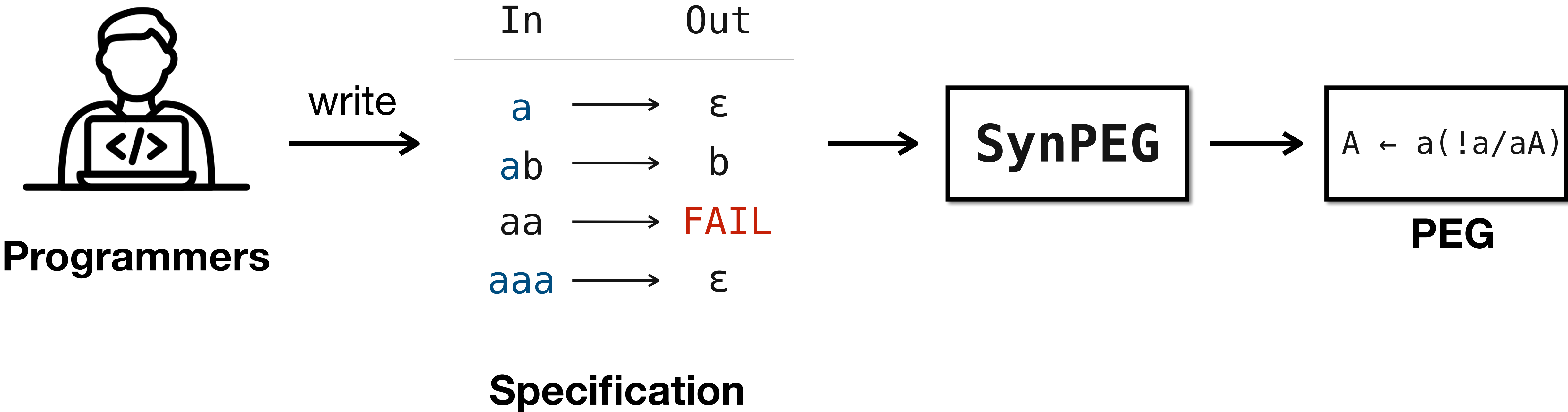
In Python PEG...

```
simple_stmts:  
    | simple_stmt ';' NEWLINE # Not needed, there for speedup  
    | ';' .simple_stmt+ [';'] NEWLINE  
  
# NOTE: assignment MUST precede expression, else parsing a simple assignment  
# will throw a SyntaxError.  
simple_stmt:  
    | assignment  
    | type_alias  
    | star_expressions  
    | return_stmt  
    | import_stmt  
    | raise_stmt  
    | pass_stmt  
    | del_stmt
```



Order can changes
parsing behavior!!

Motivation: Synthesize PEGs automatically



Background: PEG Operators

Basic expressions

A ε a $\cdot$

Optional and repetition

$e?$ e^* e^+

Sequence

$e_1 e_2$

$\&e$ $!e$

Predicates

e_1 / e_2

Ordered choice

Background: Predicates

- Using PEGs as a lookahead **without consuming** input.

And-predicate

`abcd` → `&( ab )` → `abcd`

`aadc` → `&( ab )` → **FAIL**

Not-predicate

`!( ab )`

Background: Predicates

- Using PEGs as a lookahead **without consuming** input.

And-predicate

abcd → $\&(ab)$ → abcd

aadc → $\&(ab)$ → FAIL

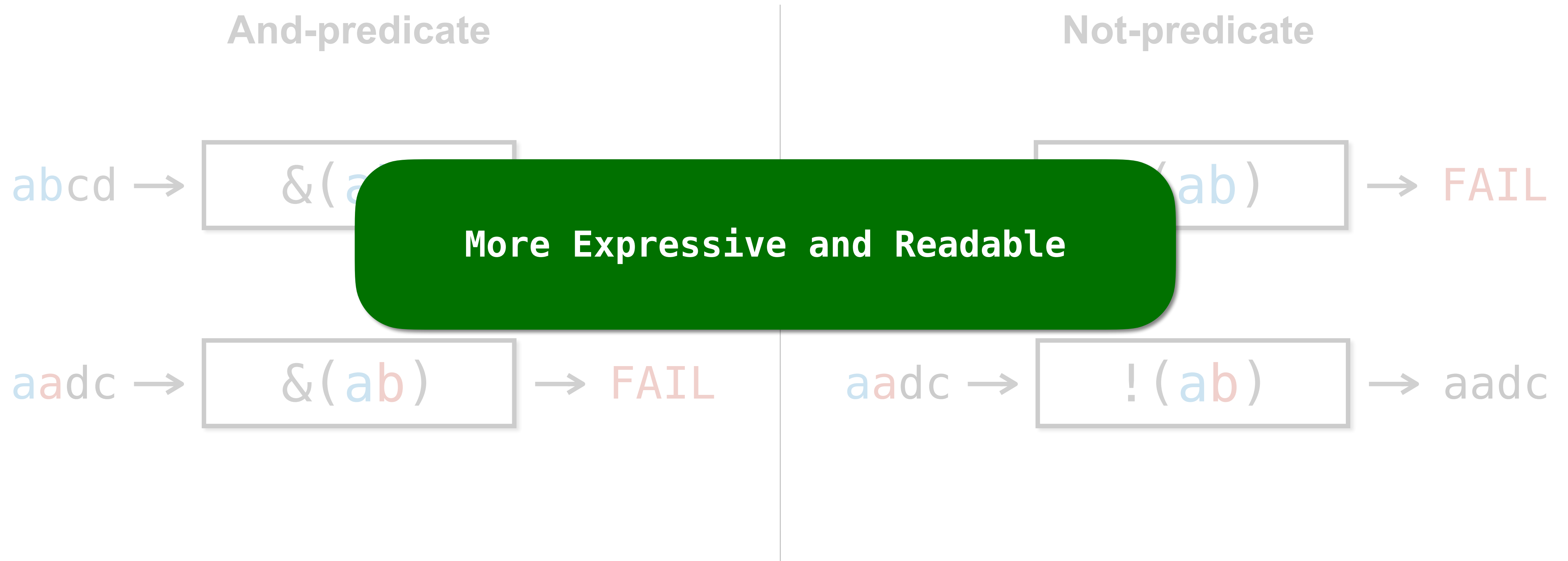
Not-predicate

abcd → $!(ab)$ → FAIL

aadc → $!(ab)$ → aadc

Background: Predicates

- Using PEGs as a lookahead **without consuming** input.



Background: Ordered Choice

- Try the left side first, and try the right side only if **left side fails**.

acba $\rightarrow$ ab / a $\rightarrow$???

1. Try left

a | c b a

FAIL

Background: Ordered Choice

- Try the left side first, and try the right side only if **left side fails**.

acba $\rightarrow$ ab / a $\rightarrow$ cba

1. Try left

a | c b a

FAIL

2. Left **fails**, then try right

a | c b a
↑

Background: Ordered Choice

- Try the left side first, and try the right side only if **left side fails**.

acba $\rightarrow$ ab / a $\rightarrow$ cba

1. Try left

a | c b a

FAIL

2. Left fails, then try right

a | c b a
↑

abcd $\rightarrow$ a / ab $\rightarrow$ bcd

1. Try left

a | b c d

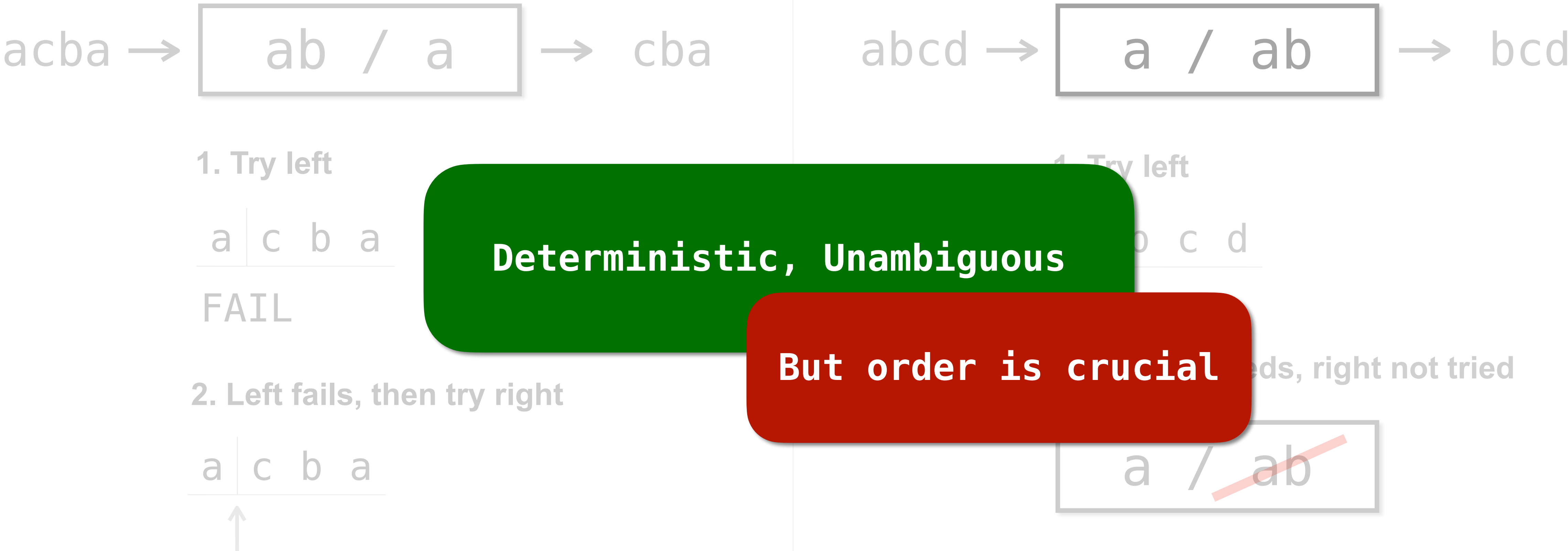


2. Left succeeds, right **not tried**

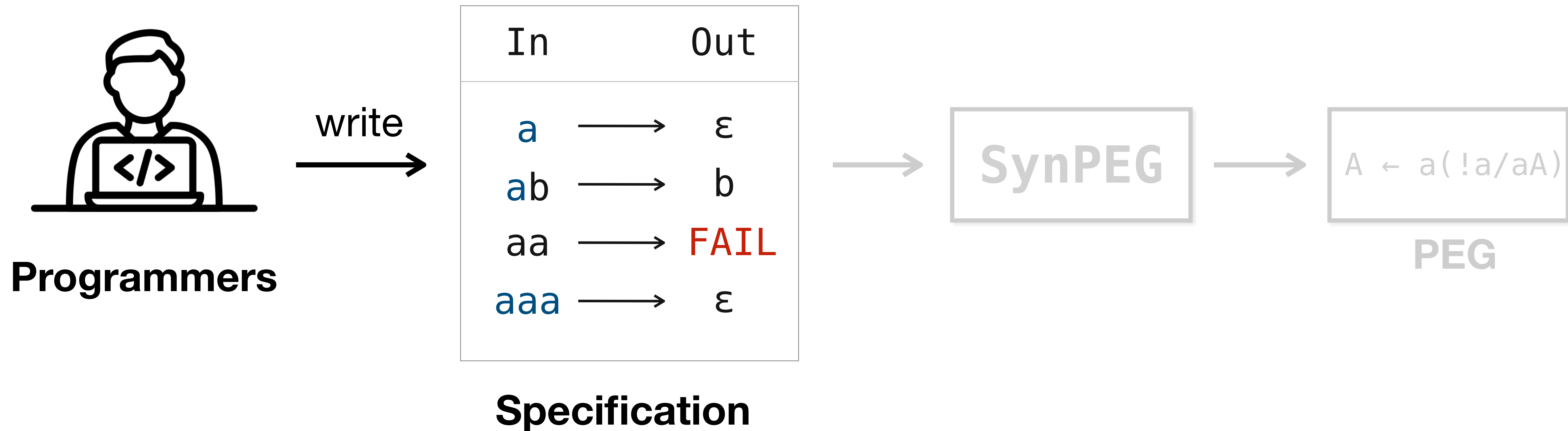
a / ~~ab~~

Background: Ordered Choice

- Try the left side first, and try the right side only if **left side fails**.



Synthesize PEGs from Specification



Running Example: Specification

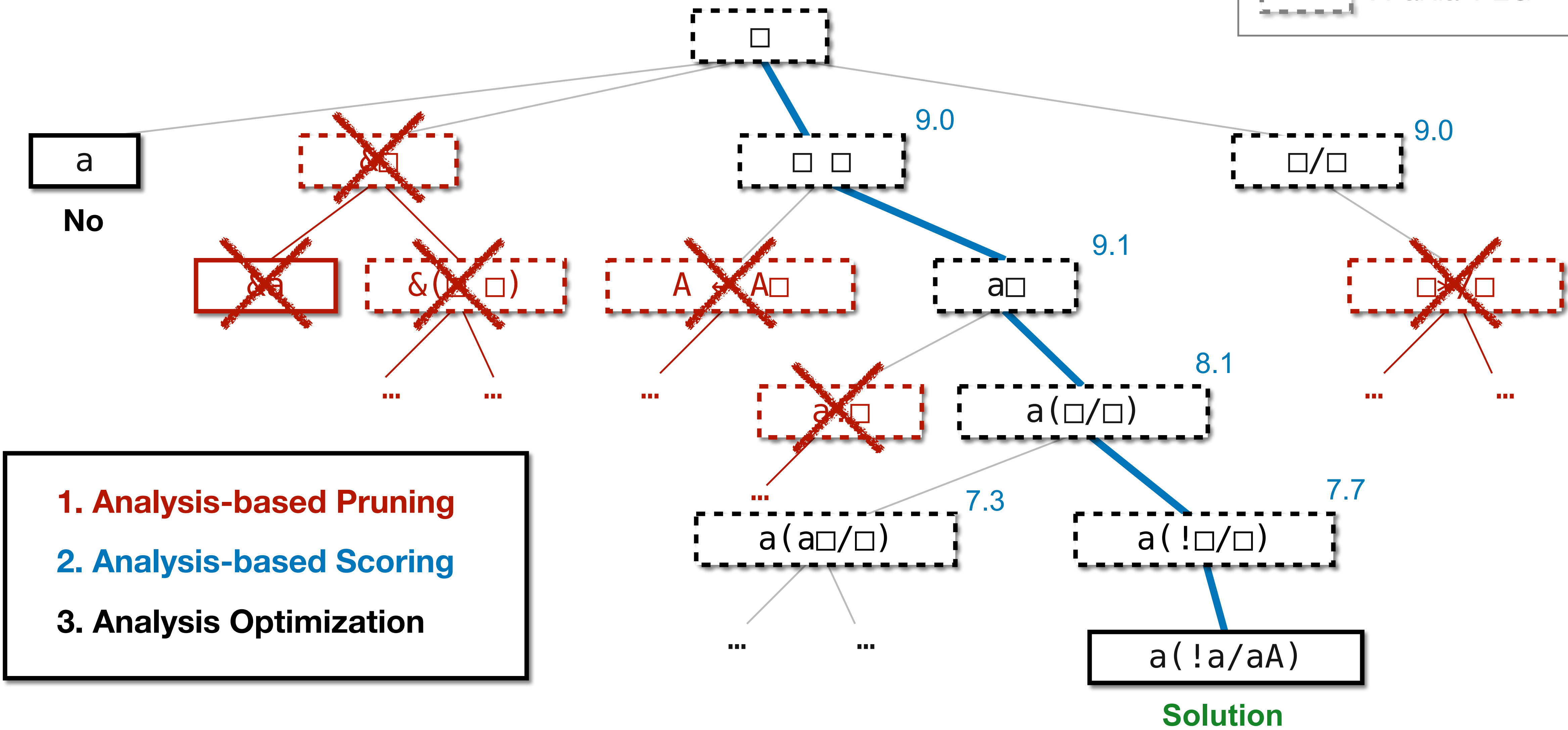
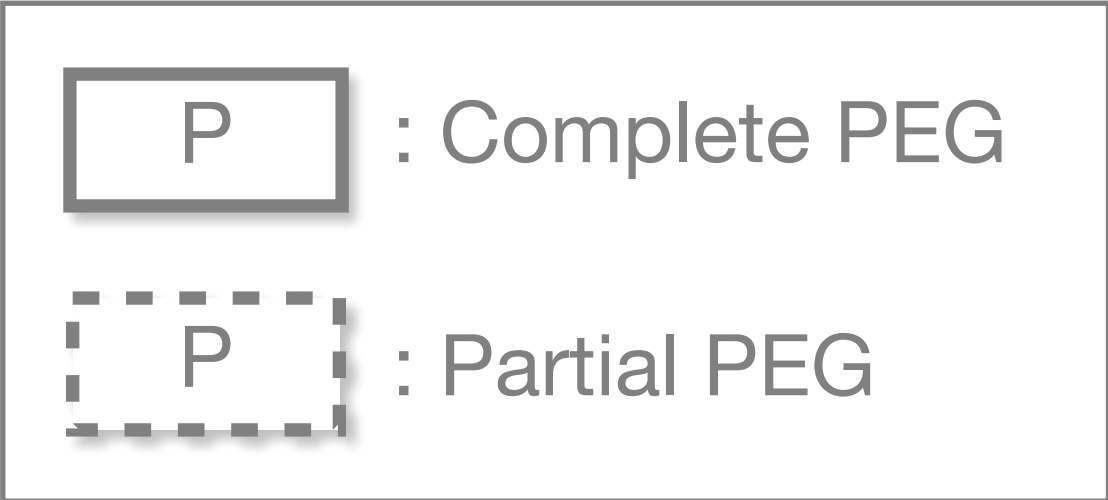
Which PEG do we want?

- Consumes as many a's as possible if the input **starts with an odd number of a's**.
- Fails otherwise (i.e., if the input starts with an even numbers of a's).

Specification (Input-output examples)

a	→	ε	aa	→	FAIL
ab	→	b	bba	→	FAIL
aaa	→	ε	aaaa	→	FAIL
aaabbaba	→	bbaba	aabbab	→	FAIL
...			...		

Synthesize a PEG

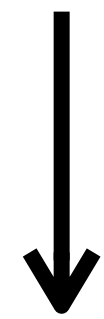


Our Approach

- **Analysis-based Pruning**
 1. **Grammar Analysis**
 - 1) Reducible PEG
 - 2) Left recursive PEG
 2. **Example Analysis**
 - 1) Invalid PEG by a single example
 - 2) Inconsistent PEG by multiple examples
- **Analysis-based Scoring**
- **Analysis Optimization**

Grammar Analysis-based Pruning

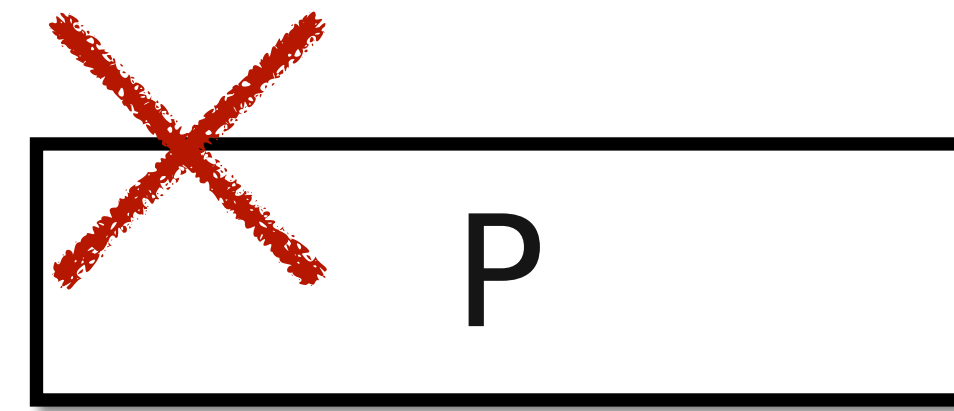
1. Reducible PEG



Reduced to smaller PEG



2. Left-recursive PEG



Stuck in an infinite loop

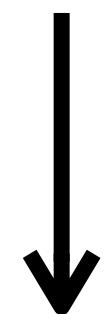
Grammar Analysis-based Pruning

1. Reducible PEG

Can be pruned!

$(\square_1) ? / \square_2$

Always succeed!



Semantically
Equivalent

$(\square) ?$

Must-Pass/Fail Analysis

2. Left-recursive PEG

$A \leftarrow A\square$

Grammar Analysis-based Pruning

1. Reducible PEG

Can be pruned!

$(\square_1) ? / \square_2$

Always succeed!



Semantically
Equivalent

$(\square) ?$

Must-Pass/Fail Analysis

2. Left-recursive PEG

$A \leftarrow A\square$

Can be pruned!

Stuck in an infinite loop

Indirect? → Nullable Analysis

Our Approach

- **Analysis-based Pruning**

1. Grammar Analysis

- 1) Reducible PEG

- 2) Left recursive PEG

2. **Example Analysis**

- 1) Invalid PEG by a single example

- 2) Inconsistent PEG by multiple examples

- Analysis-based Scoring

- Analysis Optimization

Invalid PEG by a single example

Given spec

aaabbaba → bbaba

Partial PEG P_3

$A \leftarrow a\&(\square)$

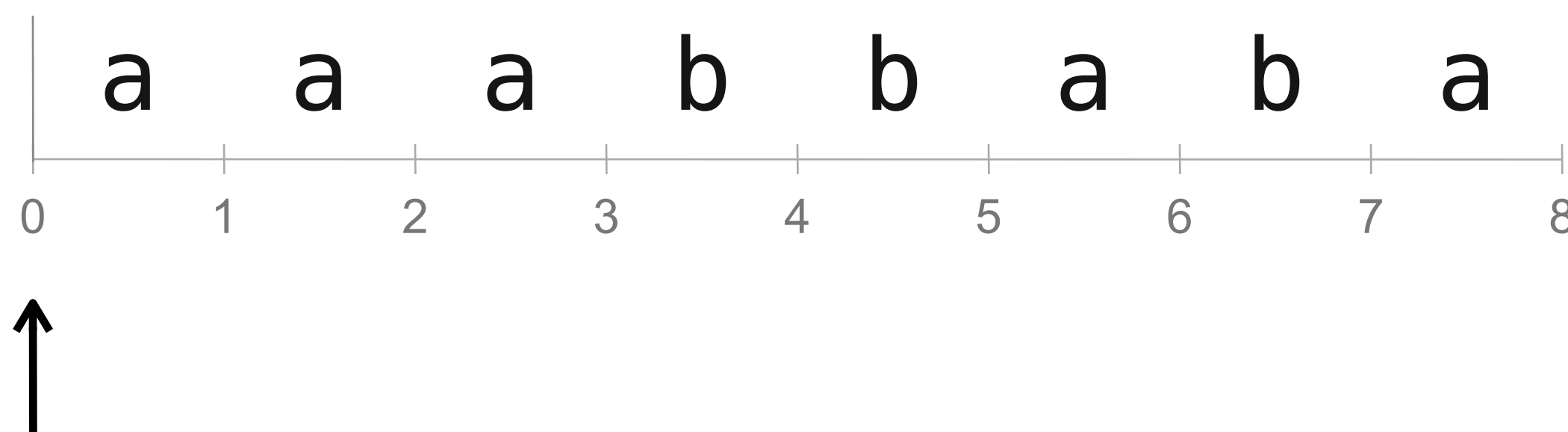
Question

If no expression in the hole can satisfy the examples, can't we just **prune** it?

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

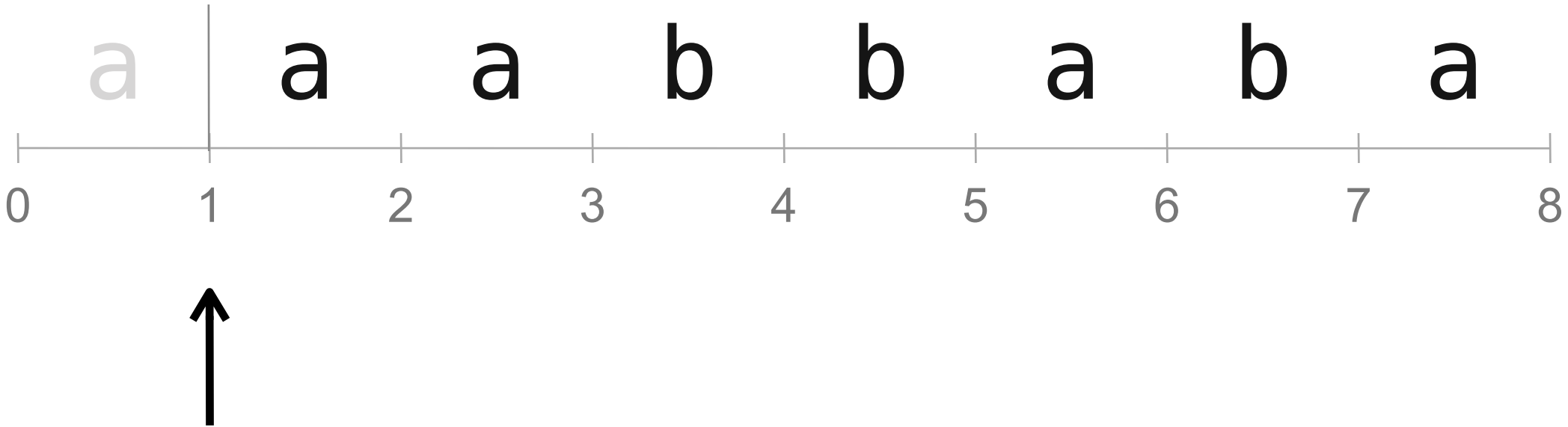
Partial PEG P_3
$A \leftarrow a\&(\square)$ ↑

Parsing Step
 Run until the hole

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

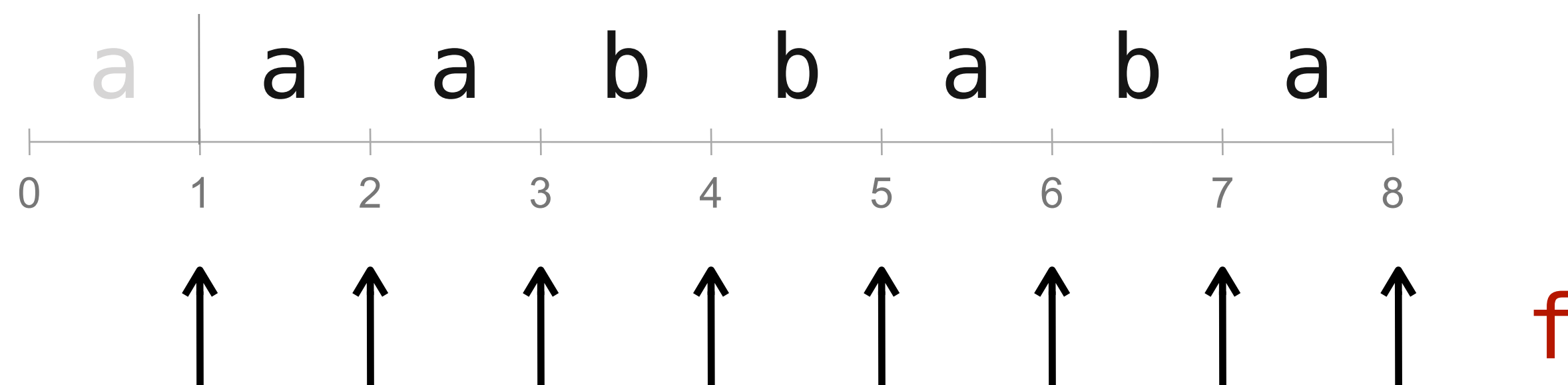
Partial PEG P_3
$A \leftarrow a \& (\square)$

Parsing Step

One a consumed. What happens next?

Invalid PEG by a single example

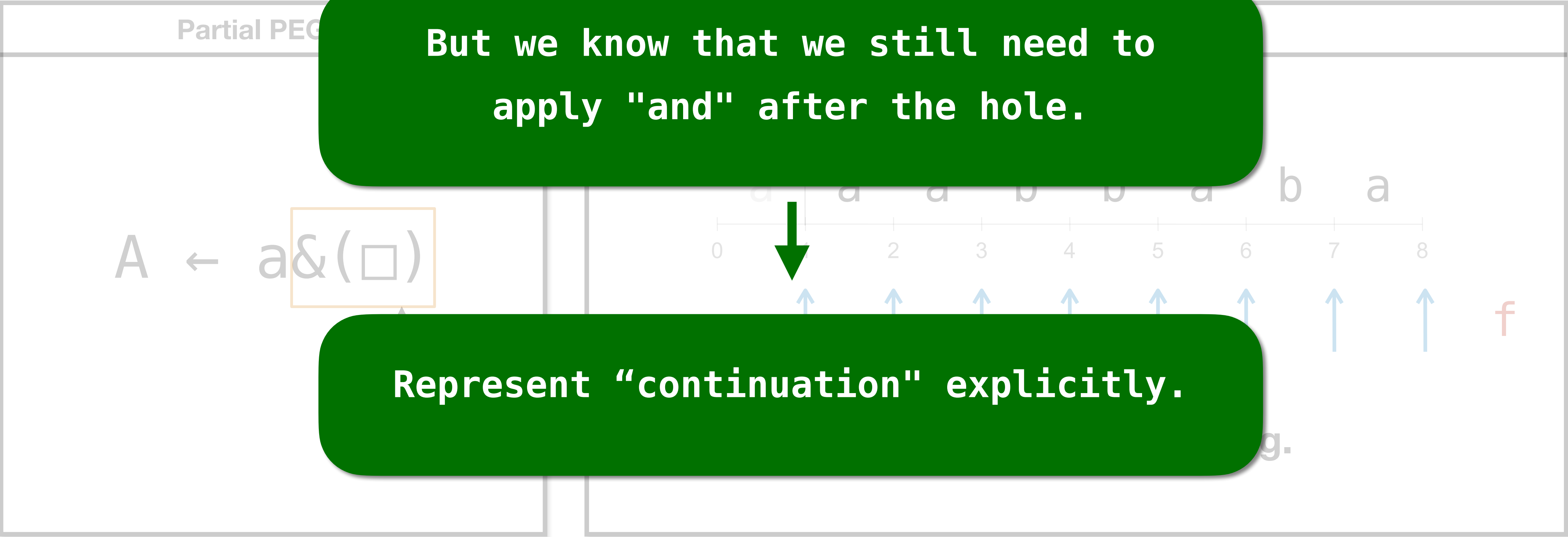
Given spec
aaabbaba → bbaba

Partial PEG P ₃
$A \leftarrow a \& (\square)$

Parsing Step
 Hole can be anything.

Invalid PEG by a single example

Given spec
aaabbaba → bbaba



Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB ↑₀ B ← &(□)
Control point ↑ active ↓ suspended

Analysis State	
call stack :	
mark index :	↓ ₀
input :	a a a b b a b a
possible outputs :	↑
spec :	↑

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB ↑₀ B ← &(□)
Control point ↑ active ↓ suspended

Analysis State	
call stack :	
mark index :	↓ ₀
input :	a a a b b a b a
	0 1 2 3 4 5 6 7 8
possible outputs :	↑
spec :	↑

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB ↑₀ B ← &(□)
Control point ↑ active ↓ suspended

Analysis State
cs : w : a a a b b a b a 0 1 2 3 4 5 6 7 8 po : ↑ spec :

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P_3
$A \leftarrow aB$ $\uparrow_0$ $B \leftarrow \&(\square)$
Control point $\uparrow$ active $\downarrow$ suspended

Analysis State
cs : w : a a a b b a b a 0 1 2 3 4 5 6 7 8 po : ↑ spec : ↑

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P₃

A ← aB

↓₀

B ← &(□)

↑₁

Control point ↑ active ↓ suspended

Analysis State

cs :

A₀

w :

a

a

a

b

b

a

b

a

0

1

2

3

4

5

6

7

8

po :

↑

spec :

↑

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB B ← & (□) ↓₀ ↑₁
Control point ↑ active ↓ suspended

Analysis State									
cs :	A₀								
w :	a	a	a	b	b	a	b	a	
	0	1	2	3	4	5	6	7	8
po :		↑							
spec :				↑					

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB B ← & (□) ↓₀ ↑₁
Control point ↑ active ↓ suspended

Analysis State

cs :

A₀

w :

a

a

a

b

b

a

b

a

po :

↑

↑

↑

↑

↑

↑

↑

↑

f

spec :

↑

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB B ← &(□) ↓₀ ↑₁
Control point ↑ active ↓ suspended

Analysis State									
cs :	A₀								
w :	a a a b b a b a								
po :	↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ success → revert mark f fail → fail								
spec :									

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P₃

A ← aB

B ← &(□)

↓₀

↑₁

Control point ↑ active ↓ suspended

Analysis State

cs :

A₀

w :

a

a

a

b

b

a

b

a

0

1

2

3

4

5

6

7

8

po :

↑

spec :

↑

f

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P ₃
A ← aB ↑₀ B ← &(□) Control point ↑ active ↓ suspended

Analysis State
cs : w : a a a b b a b a 0 1 2 3 4 5 6 7 8 po : ↑ spec : ↑ f

Invalid PEG by a single example

Given spec
aaabbaba → bbaba

Normal Form of P_3

$A \leftarrow aB$

$B \leftarrow \&(\square)$

Pruning

$\uparrow_0$

Control point $\uparrow$ active $\downarrow$ suspended

Analysis State

cs :

w :

a | a a b b a b a

0 1 2 3 4 5 6 7 8

po :

$\uparrow$

$\times$

f

spec :

$\uparrow$

Find Possible Outputs via Analysis

Prune if the spec's output
is **not in** the possible outputs.

Given spec
aaabbaba → bbaba

A ← aB
B ← &(□)

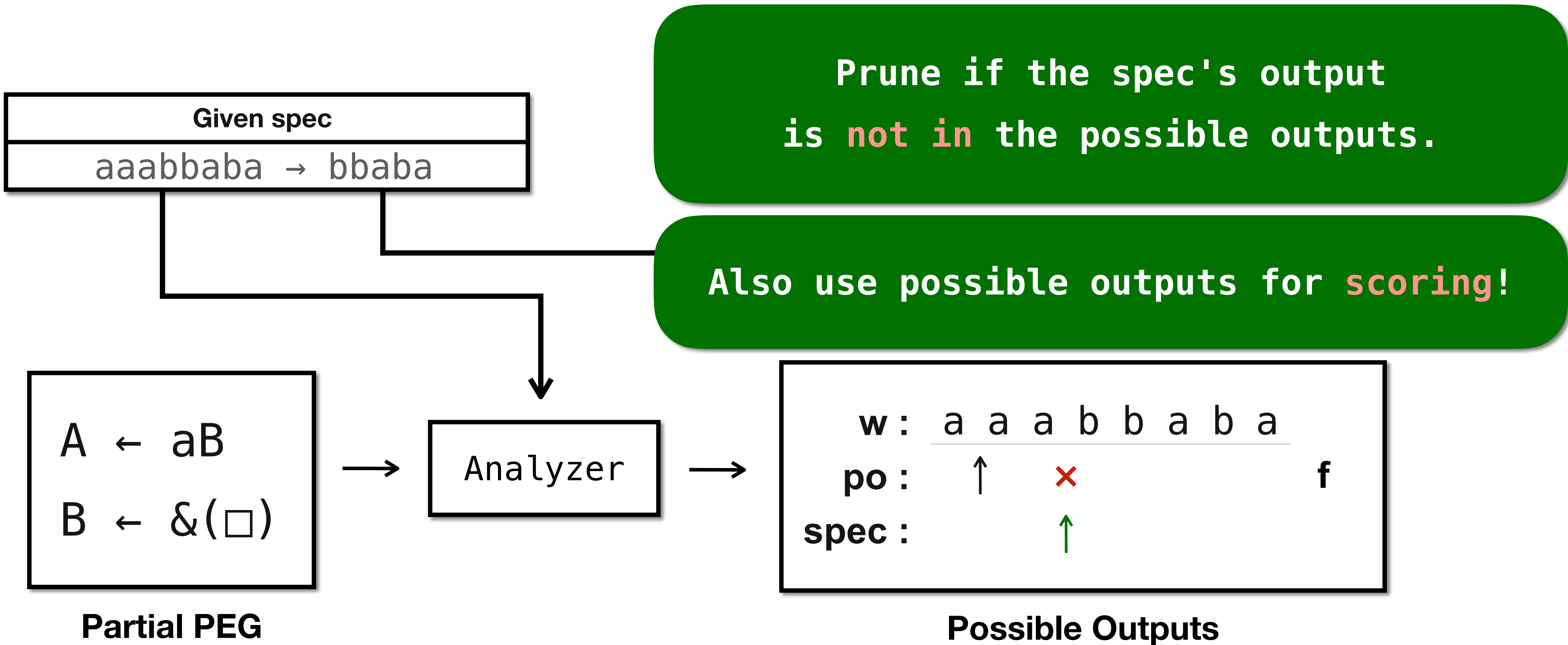
Partial PEG

Analyzer

w :	a	a	a	b	b	a	b	a	
po :	↑			×					f
spec :				↑					

Possible Outputs

Find Possible Outputs via Analysis



Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL aaabbab → bbab
Partial PEG P ₄
A ← BC
B ← (a)*
C ← &(D)
D ← b□
Control point ↑ active ↓ suspended

Question
Can any expressions satisfy both examples at the same time ?

Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

Partial PEG P ₄
A ← BC
B ← (a)*
C ← &(D)
D ← b□
Control point ↑ active ↓ suspended

State for example 1
cs :
w : a a b b a b
po : ↑
spec : FAIL

State for example 2
cs :
w : a a a b b a b
po : ↑
spec :

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

A ← BC

↑₀

B ← (a)*

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs :

w : a a b b a b

po : ↑

spec : FAIL

State for example 2

cs :

w : a a a b b a b

po : ↑

spec : ↑

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↑₁

B ← (a)*

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs : A₀

w : a a b b a b

po : ↑

spec : FAIL

State for example 2

cs :

w : a a a b b a b

po : ↑

spec : ↑

PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↑₁

B ← (a)*

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs : A₀

w : a a b b a b

po : ↑

spec : FAIL

State for example 2

cs :

w : a a a b b a b

po : ↑

spec : ↑

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P_4

$A \leftarrow BC$
 $\uparrow_0$

$B \leftarrow (a)^*$

$C \leftarrow \&(D)$

$D \leftarrow b\Box$

Control point

$\uparrow$ active $\downarrow$ suspended

State for example 1

cs :

w :

a a | b b a b

po :

$\uparrow$

spec :

FAIL

State for example 2

cs :

w :

a a a b b a b

po :

$\uparrow$

spec :

$\uparrow$

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↑₁

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs : A₀

w : a a b b a b

po : ↑

spec : FAIL

State for example 2

cs :

w : a a a b b a b

po : ↑

spec : ↑

PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

B ← (a)*

C ← &(D)

D ← b□

↑₁

Control point

↑ active

↓ suspended

State for example 1

cs : A₀

↓₁

w : a a b b a b

↑

po :

spec : FAIL

State for example 2

cs :

w : a a a b b a b

↑

po :

spec :

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↓₁

B ← (a)*

↓₁

C ← &(D)

↑₂

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs :

A₀

C₁

w :

a

a

b

b

a

b

po :

spec : FAIL

State for example 2

cs :

w :

a

a

a

b

b

a

b

po :

spec :

 PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↓₁

B ← (a)*

↓₁

C ← &(D)

↑₂

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs :

A₀

C₁

w :

a

a

b

b

a

b

po :

spec : FAIL

State for example 2

cs :

w :

a

a

a

b

b

a

b

po :

spec :

 PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

A ← BC

↑₀

B ← (a)*

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs : A₀ C₁

↓¹

w : a a b b a b

po : ↑

spec : FAIL

State for example 2

cs :

w : a a a b b a b

po : ↑

spec : ↑

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↓₁

B ← (a)*

↓₁

C ← &(D)

↑₂

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs :

A₀

C₁

w : a a b b a b

po :

spec : FAIL

State for example 2

cs :

A₀

C₁

w : a a a b b a b

po :

spec :

PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

Partial PEG P ₄
↓₀ A ← BC ↓₁ B ← (a)* ↓₁ C ← &(D) ↑₂ D ← b□ Control point ↑ active ↓ suspended

State for example 1

cs :

A ₀	C ₁	
----------------	----------------	--

↓¹

w :

a	a	b	b	a	b
---	---	---	---	---	---

po :

↑

spec : FAIL

State for example 2

cs :

A ₀	C ₁	
----------------	----------------	--

w :

↓¹

a a a b b a b

↑

po :

spec :

↑

Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

Partial PEG P_4

```

graph TD
    A[A] -- l0 --> B[B]
    B --> C[C]
    C --> D[D]
    D --> S[ ]
    style A fill:#ccc,stroke:#333,stroke-width:1px
    style B fill:#ccc,stroke:#333,stroke-width:1px
    style C fill:#ccc,stroke:#333,stroke-width:1px
    style D fill:#ccc,stroke:#333,stroke-width:1px
    style S fill:#fff,stroke:#333,stroke-width:1px
  
```

Control point ↑ active ↓ suspended

State for example 1

cs :

A_0	C_1	
-------	-------	--

w :

a	a	b	b	a	b
---	---	---	---	---	---

po :

a	a	b	b	a	b
---	---	---	---	---	---

spec : FAIL

State for example 2

The diagram illustrates the state for example 2. It consists of four rows:

- cs :** A sequence of cells. The first cell contains A_0 , the second contains C_1 , and the rest are empty.
- w :** A sequence of characters: a, a, a, b, b, a, b. The first three 'a's are enclosed in a red box. A blue arrow labeled '1' points down to the first 'b'.
- po :** A sequence of markers (vertical lines) corresponding to the positions of the characters in 'w'.
- spec :** A sequence of markers. A green arrow points up to the first 'b' in 'w', and a black arrow points up to the second 'b' in 'w'.

Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

Partial PEG P ₄
↓₀ A ← BC ↓₁ B ← (a)* ↓₁ C ← &(D) ↑₂ D ← b□ Control point ↑ active ↓ suspended

State for example 1
cs : A₀ C₁ w : ↓₁ b b a b po : ↑ spec : FAIL

State for example 2
cs : A₀ C₁ w : ↓₁ b b a b po : ↑ spec : ↑

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

A ← BC

B ← (a)*

C ← &(D)

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs : A₀ C₁

↓¹

w : b b a b

They have same continuations!

State for example 2

cs : A₀ C₁

↓¹

w : b b a b

po : ↑

spec : ↑

PLRG

16 / 27

Inconsistent PEG by multiple examples

Given spec

aabbab → FAIL
aaabbab → bbab

Partial PEG P₄

↓₀

A ← BC

↓₁

B ← (a)*

↓₁

C ← &(D)

↑₂

D ← b□

Control point

↑ active

↓ suspended

State for example 1

cs :

A ₀	C ₁	
----------------	----------------	--

w :
po :

↓₁

b

b

a

b

↑

spec : FAIL

cs :

A ₀	C ₁	
----------------	----------------	--

w :
po :

↓₁

b

b

a

b

↑

spec :

Also have same string and consume states.

PLRG

16 / 27

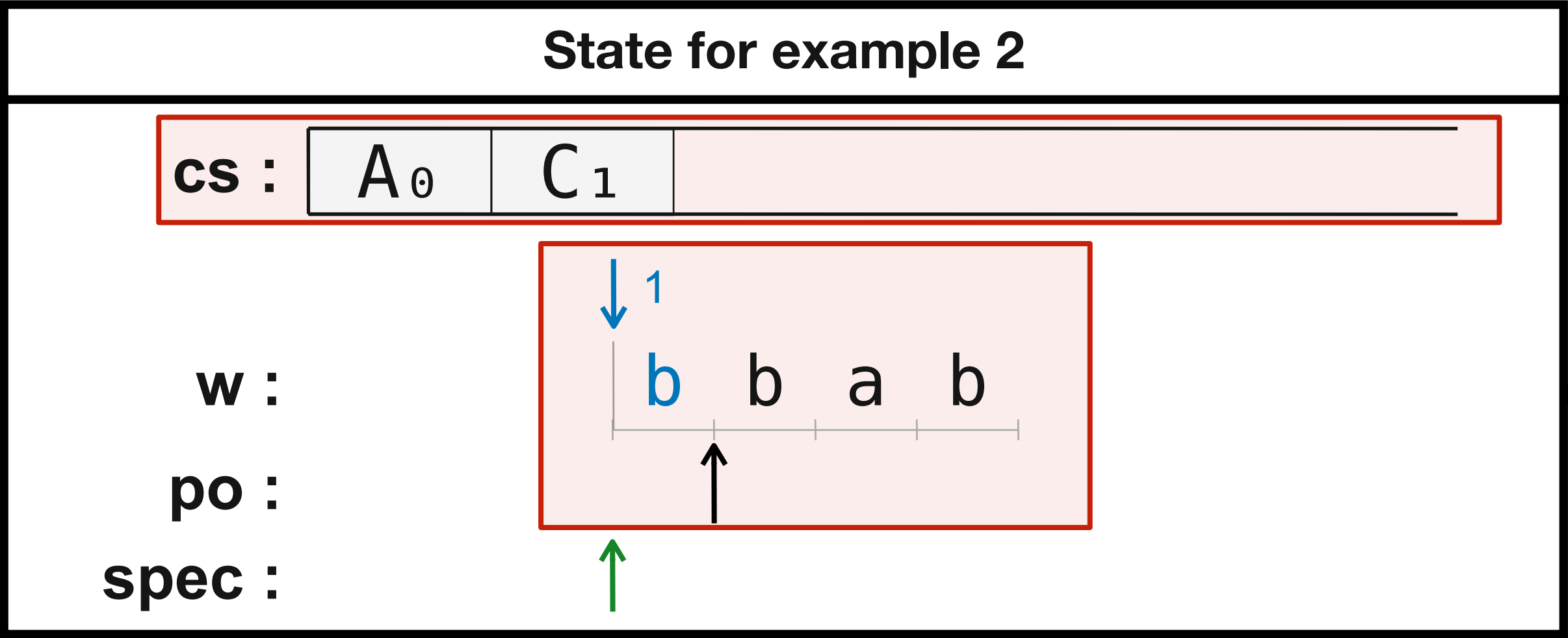
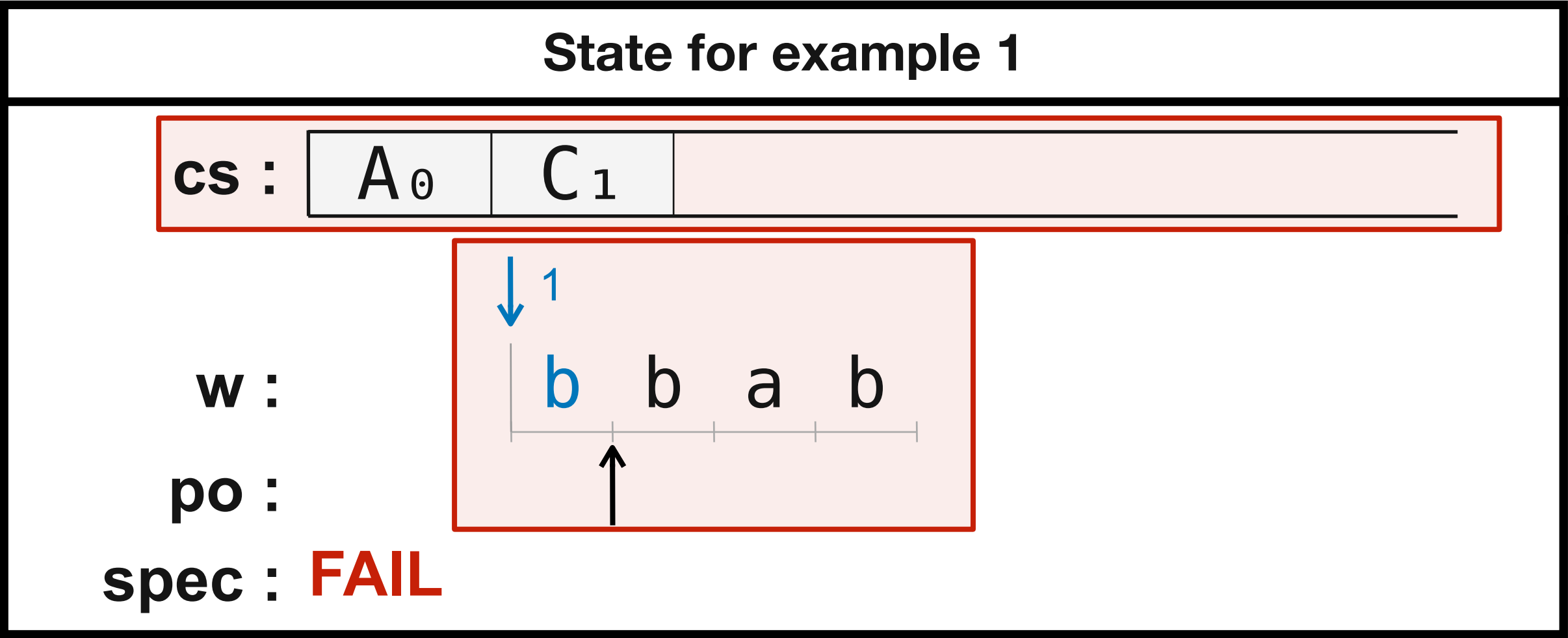
Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

Partial PEG P ₄

So, two states are equivalent!!

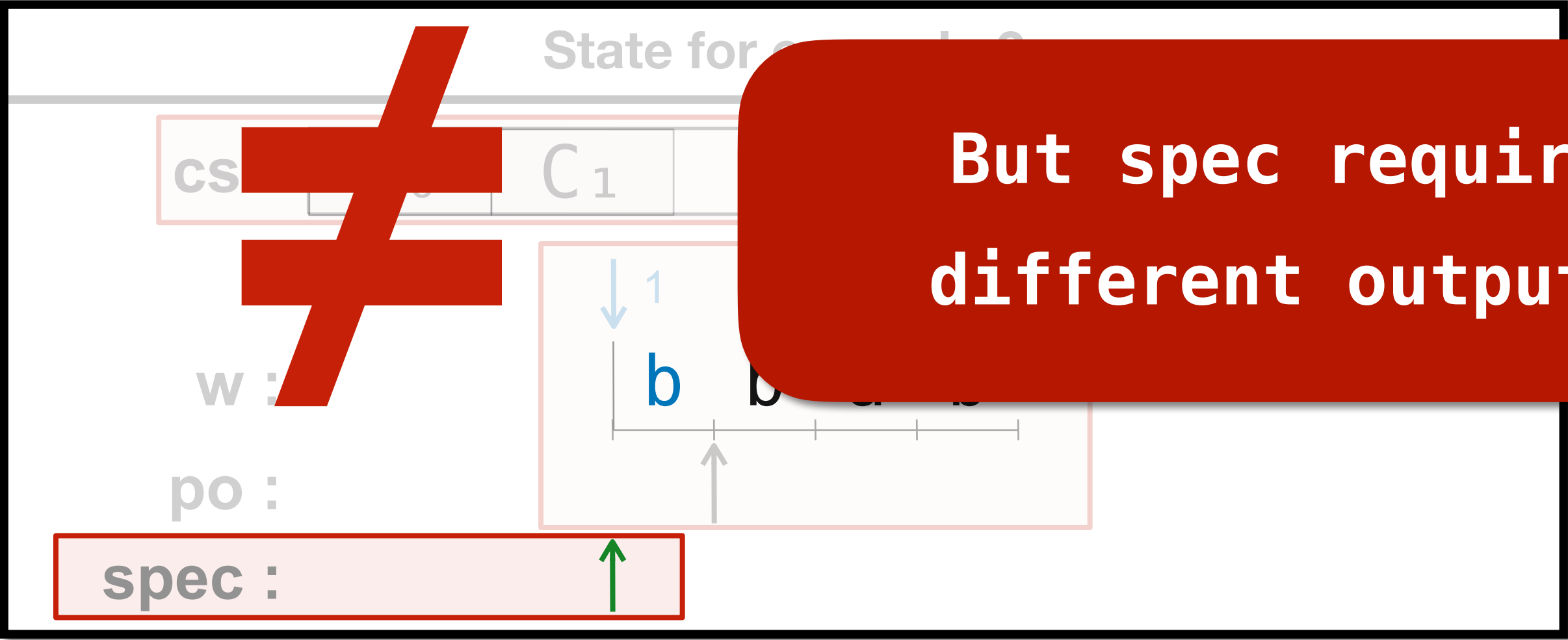
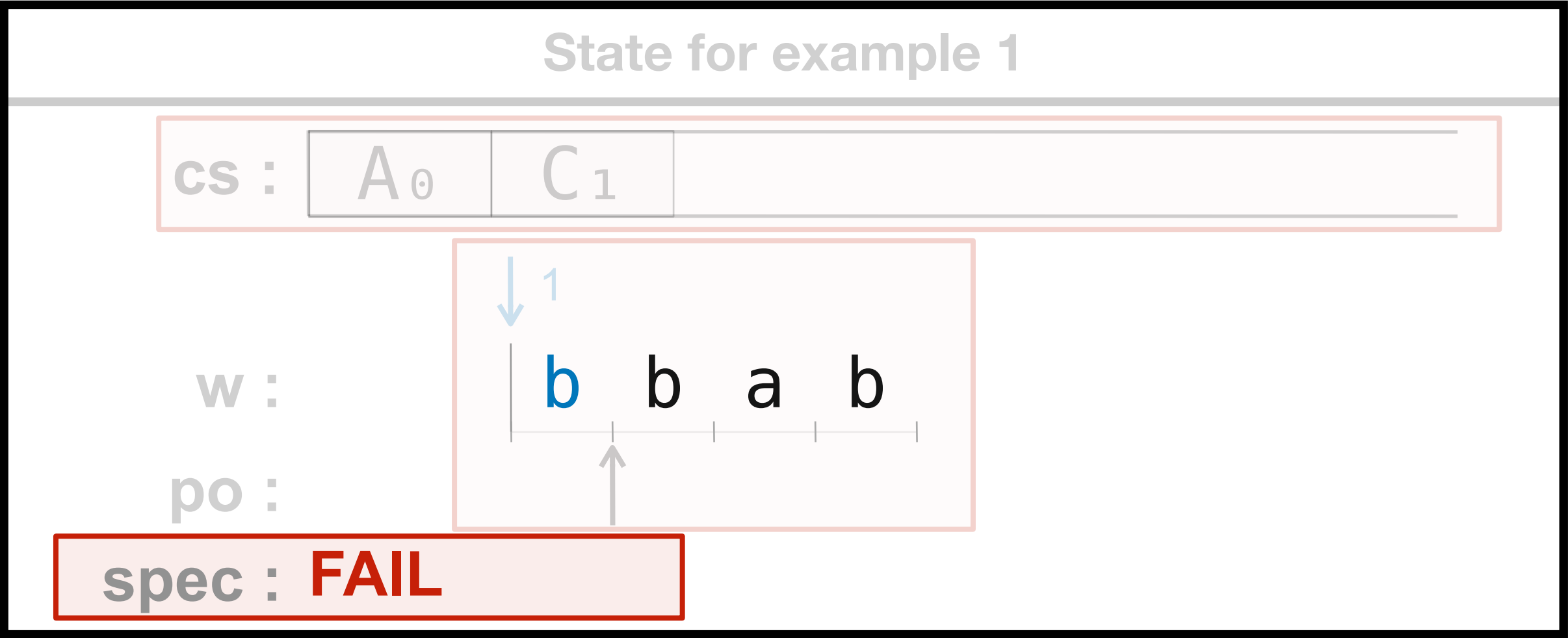
They must end with the same output.



Inconsistent PEG by multiple examples

Given spec
aabbab → FAIL
aaabbab → bbab

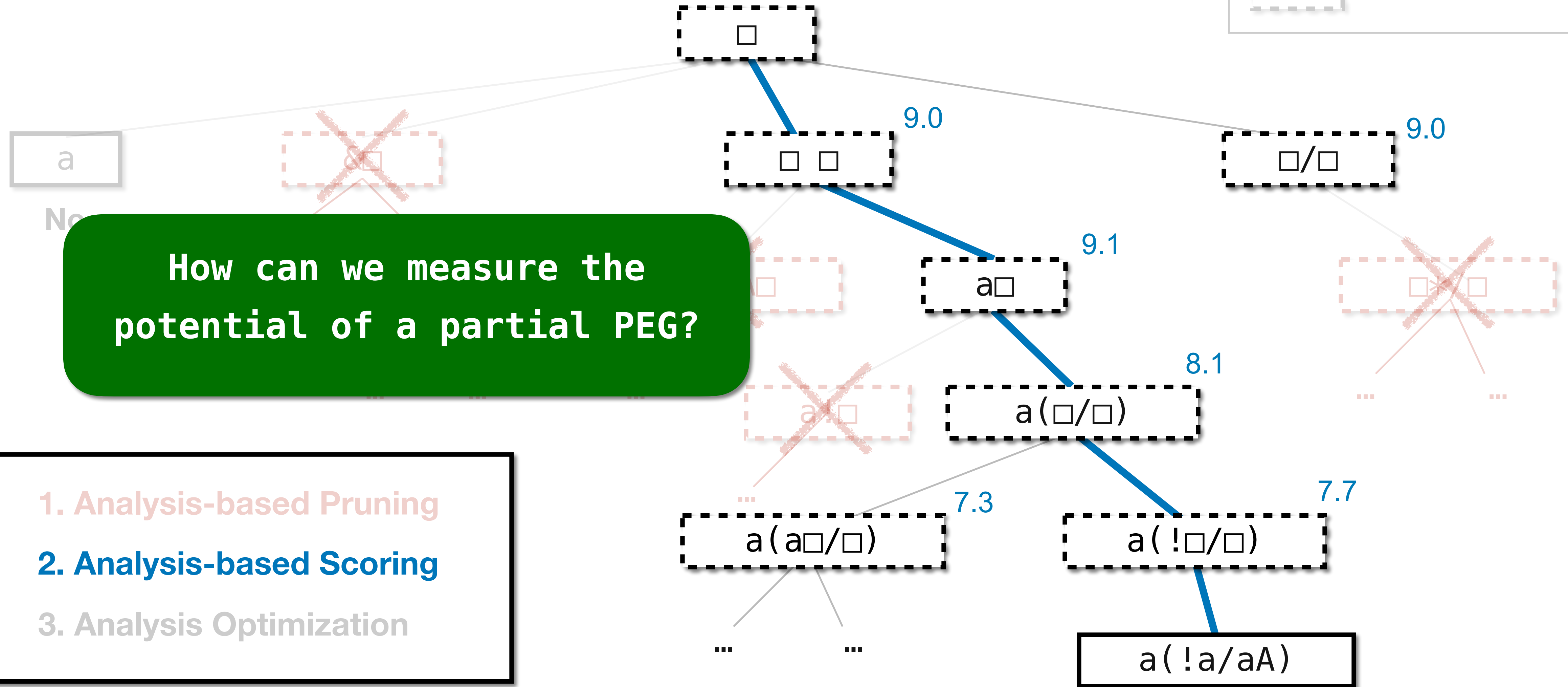
Partial PEG P ₄
$A \leftarrow BC$ $B \leftarrow (a)^*$ $C \leftarrow \&(D)$ $D \leftarrow b\Box$ Pruning Control point ↑ active ↓ suspended



Our Approach

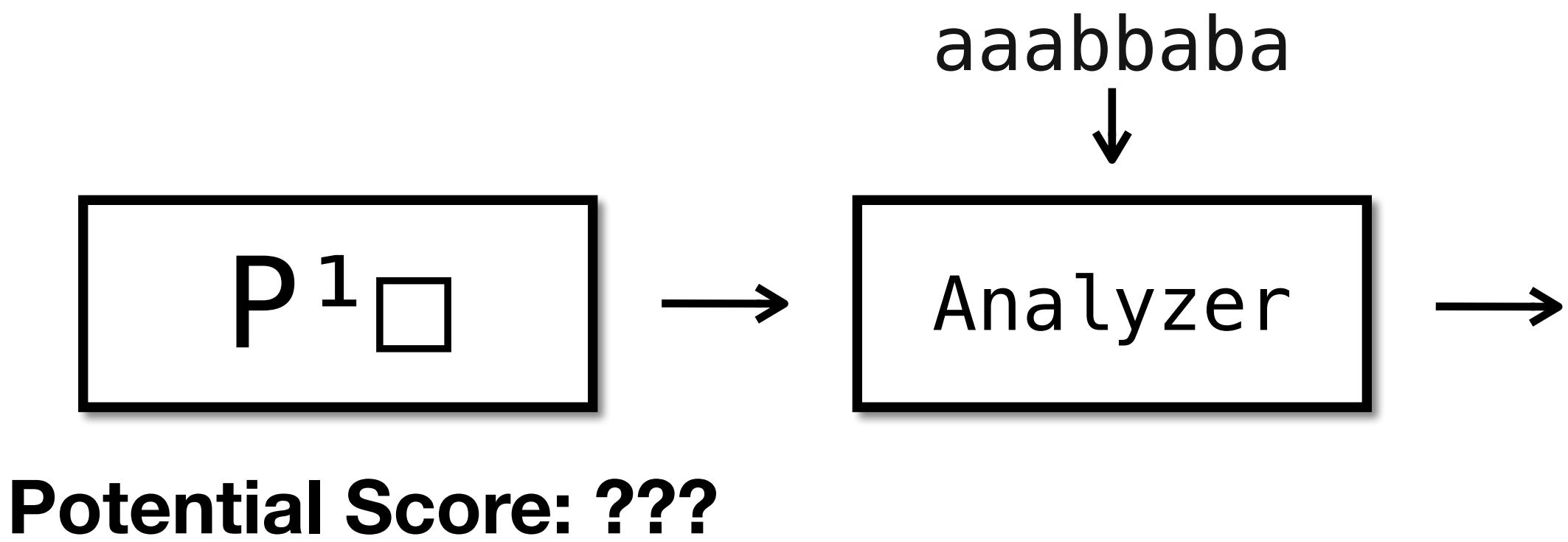
- Analysis-based Pruning
 1. Grammar Analysis
 - 1) Reducible PEG
 - 2) Left recursive PEG
 2. Example Analysis
 - 1) Invalid PEG by a single example
 - 2) Inconsistent PEG by multiple examples
- **Analysis-based Scoring**
- Analysis Optimization

Analysis-based Scoring

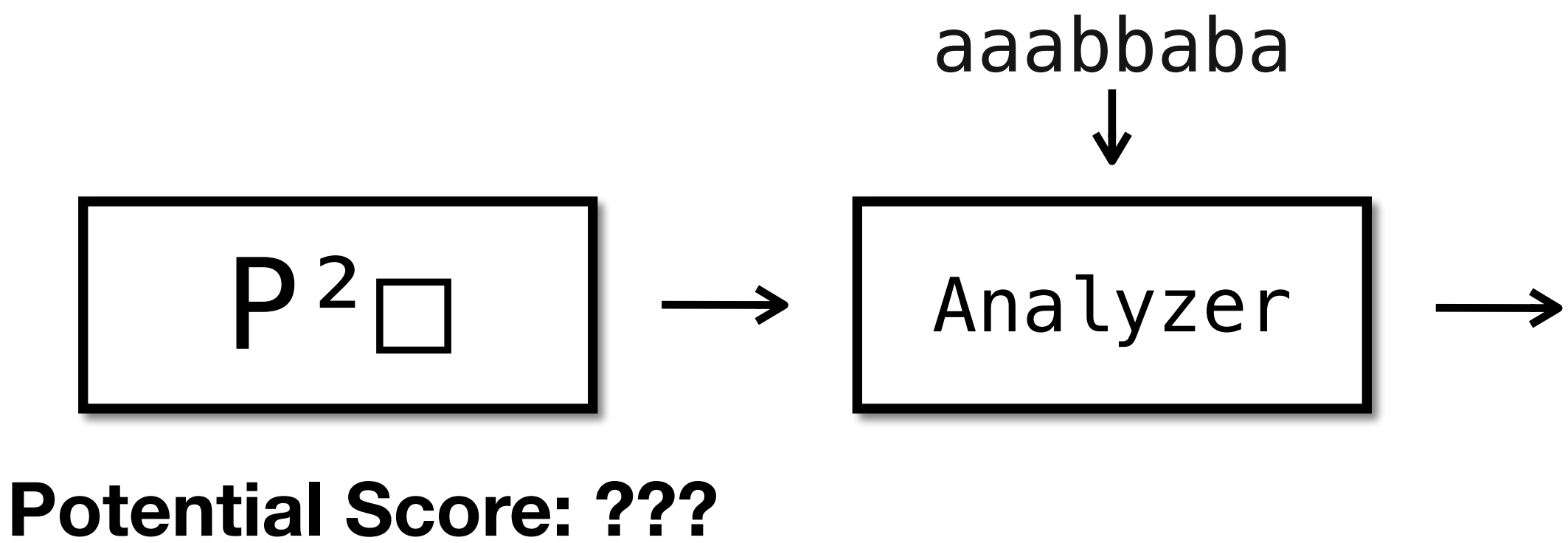


Potential Score

Given spec
aaabbaba → bbaba



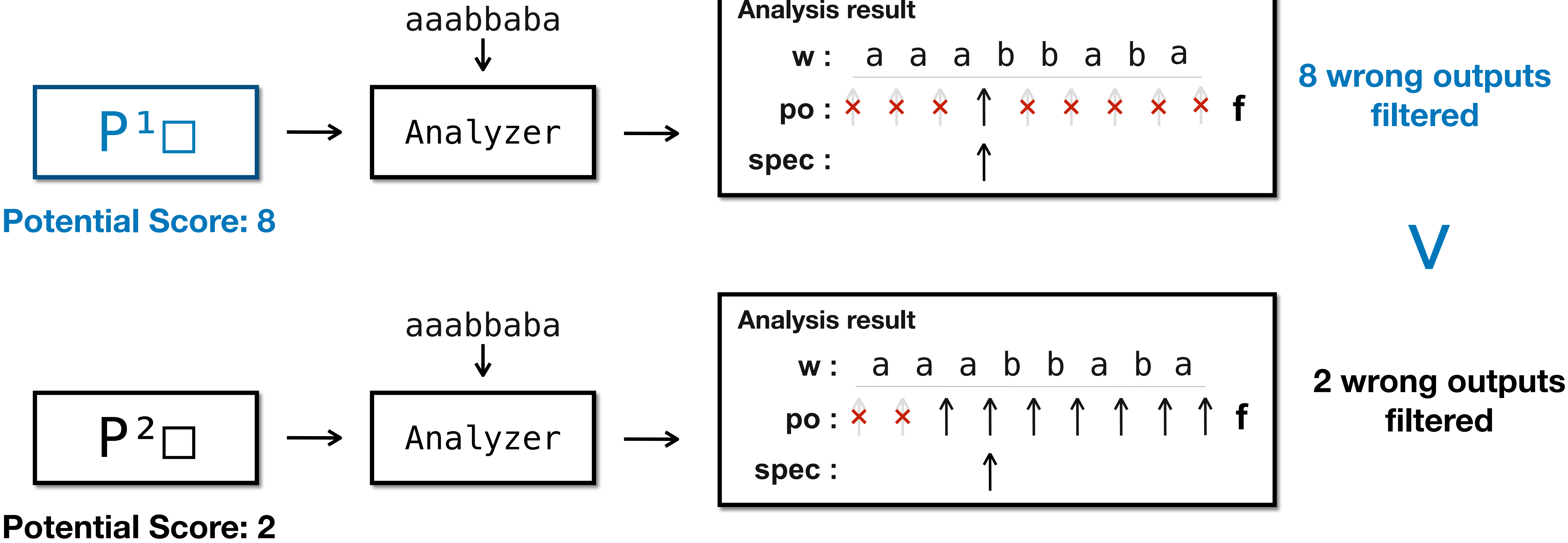
Analysis result									
w :	a	a	a	b	b	a	b	a	
po :				↑					f
spec :				↑					



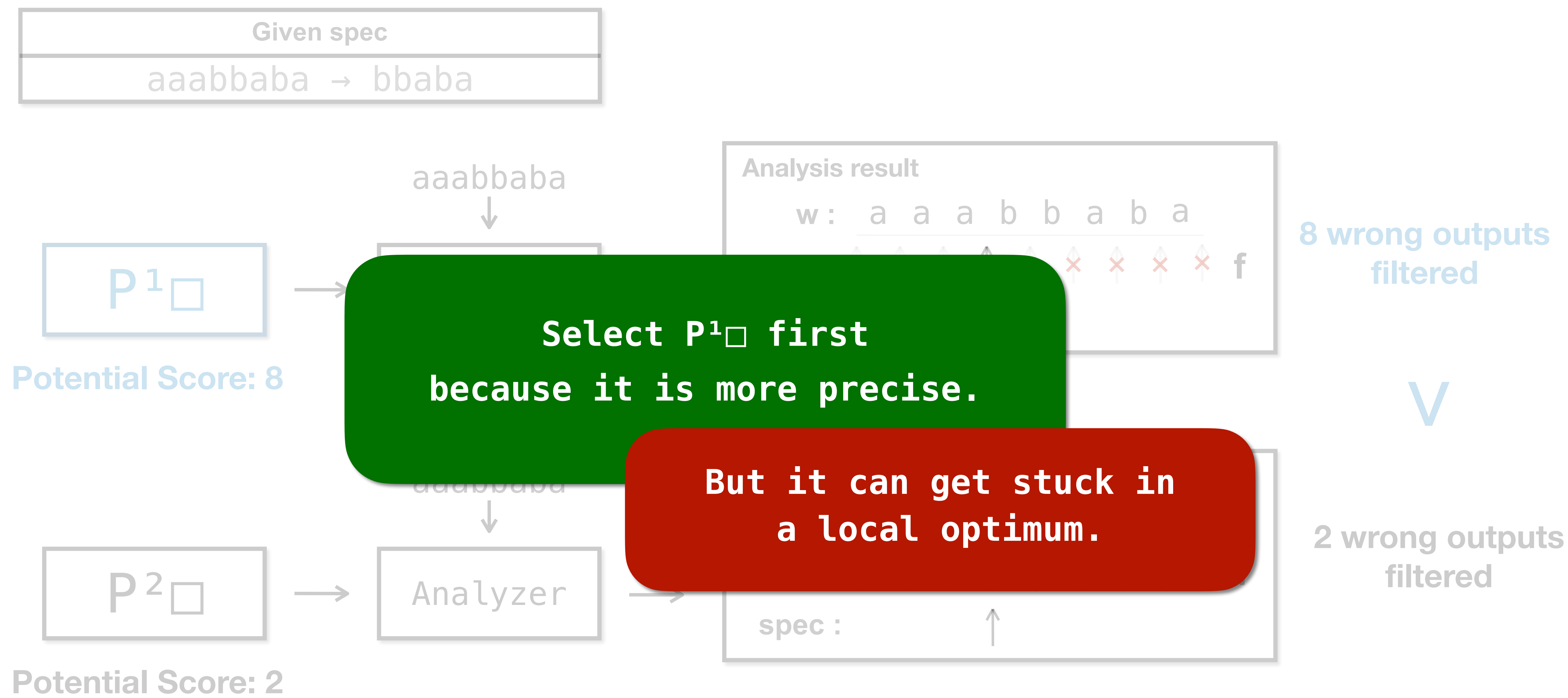
Analysis result									
w :	a	a	a	b	b	a	b	a	
po :		↑	↑	↑	↑	↑	↑	↑	f
spec :			↑						

Potential Score

Given spec
aaabbaba → bbaba



Potential Score



Scoring Function

Scoring function for partial PEG P

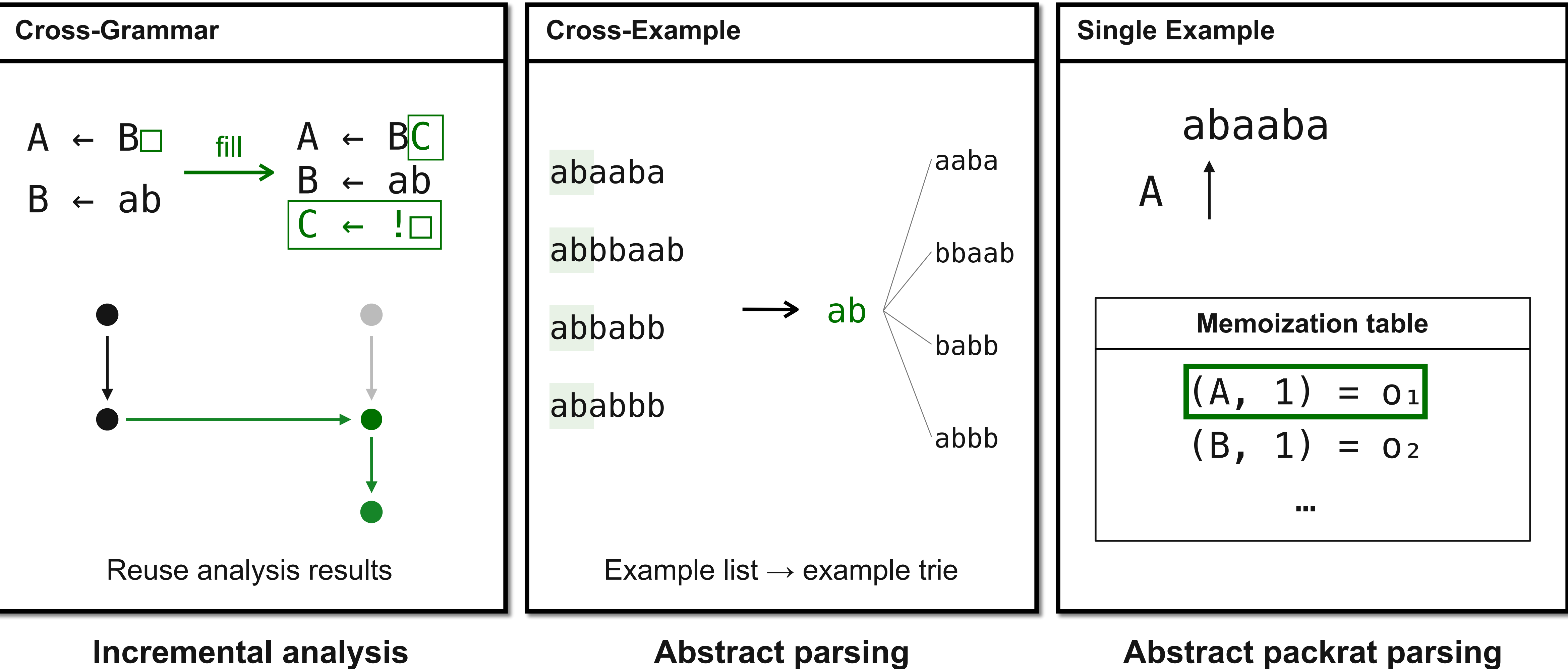
$$f(P) = \alpha \cdot \text{score}(P) - \text{size}(P)$$

Term	Meaning
score(P)	Wrong outputs filtered, normalized over examples
size(P)	Number of operators
α	Weight of the potential score (upper bound of its contribution)

Our Approach

- Analysis-based Pruning
 1. Grammar Analysis
 - 1) Reducible PEG
 - 2) Left recursive PEG
 2. Example Analysis
 - 1) Invalid PEG by a single example
 - 2) Inconsistent PEG by multiple examples
- Analysis-based Scoring
- **Analysis Optimization**

Optimization for Analysis



Benchmark

- 15 tasks, from formal specifications to language fragments

No.	Description	V_T	Predefined Rules	$\#e_{\text{train}}$
1	Parse as many 'a's as possible or 'abb'.	a, b	-	20
2	Parse the leading 'a's only if there is an odd number of 'a's.	a, b	-	20
...				
15	Parse the keyword 'static' only if it is used for static generators in a JavaScript program.	[a-z], (,),*, {,},s	$\begin{aligned} \text{WS} &\leftarrow ' '*; \\ \text{NAME} &\leftarrow \text{ALPHA ALPHA}^*; \\ \text{STATIC} &\leftarrow \text{'static ' WS}; \\ \text{PARAMS} &\leftarrow '()' \\ &\quad / '(' \text{NAME} (', ' \text{NAME})^* ')'; \end{aligned}$	30

- 10-30 examples for training, 10,000 examples for test.

Evaluation: Research Questions

RQ1: How well does SynPEG synthesize PEGs for the benchmarks?

RQ2: How effective are the pruning techniques and the score?

- Search space reduction
- Time speed up

RQ3: How effective are our optimizations?

RQ 1: Synthesis Performance

Method	Size	Time (s)	Accuracy (%)	Solved / 15
LLM	6.3	1.43	75.9	3.4
LLM + desc.	7.5	1.41	74.9	5.4
SynPEG	4.9	0.43	100.0	15 / 15

RQ 2: Effectiveness of Techniques

Configuration	Mean candidates	Mean time (s)	Solved / 15
Baseline	>14.91 M	209.37	5
B + Grammar	>9.86 M	166.87	9
G + Example	>1.79 M	97.54	12
E + Score	17.21 K	0.43	15

99.82%

389x

RQ 3: Effectiveness of Optimizations

Analysis	Without Opt. (s)	With Opt. (s)
Grammar	0.76	0.07
Example	0.24	0.06
Total	1.00	0.13

6.85x

Future Works

1. More precise analysis

- Prune more partial PEGs

2. Extend an existing PEG

- Using examples for the new behavior

Overall of SynPEG

