Synthesizing Regular Expressions for Introductory Automata Assignments

Mina Lee, **Sunbeom So**, and Hakjoo Oh

Programming Research Laboratory
Korea University

31 October 2016
GPCE 2016 @ Amsterdam, Netherlands

Quiz

Strings have exactly one pair of consecutive 0s. ($\Sigma=\{0,1\}$)

Give it a try!

Quiz

Strings have exactly one pair of consecutive 0s. (Σ={0,1})

Positive Example

00
1001
010010
1011001110

Negative Example

01
11
000
00100

Quiz

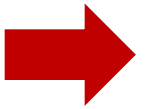
Strings have exactly one pair of consecutive 0s. (Σ={0,1})

Positive Example

00
1001
010010
1011001110

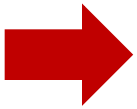
Negative Example

01
11
000
00100



Alpha Regex

1.0s



(0?1)*00(10?)*

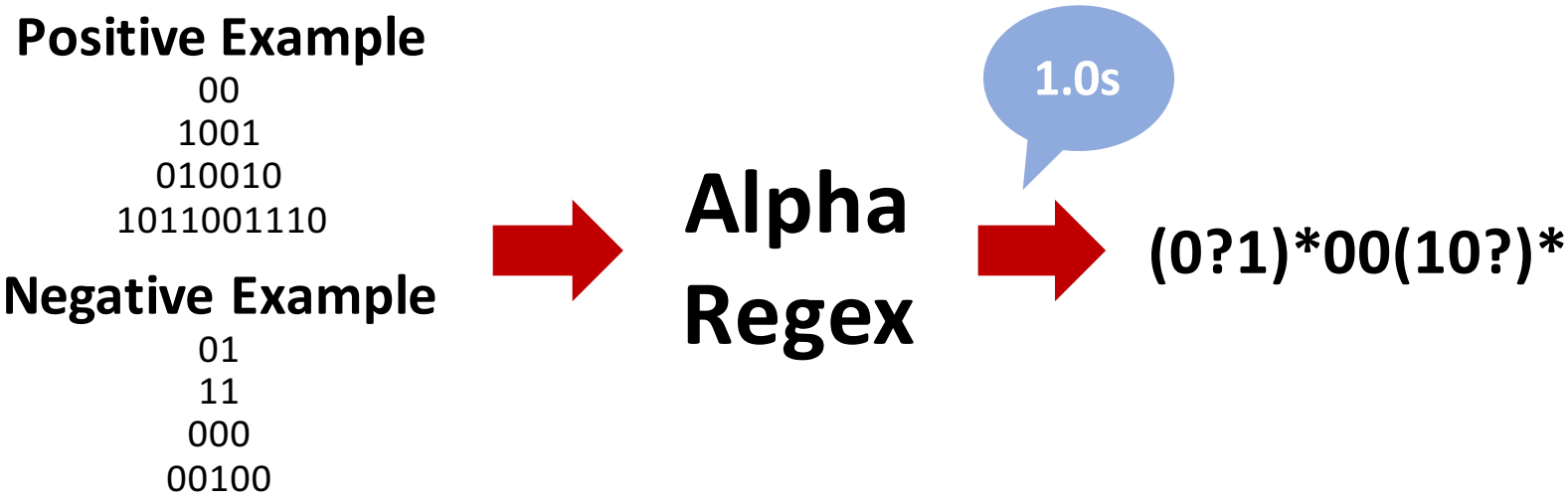
Motivation

To help the students who learn REs

Motivation

Strings have exactly one pair of consecutive 0s. (Σ={0,1})

- **Most students : "I don't know how to start."**



Motivation

Strings have exactly one pair of consecutive 0s. (Σ={0,1})

- **Some students : "Is there compact any solution?"**

Students

$$1^*(01)^*001^*(10)^*1^*$$

(0?1)*00(10?)*

AlphaRegex

Another example 1

The number of 0s is divisible by 3. (Σ={0,1})

Positive Example

1
11
111
000
000000
01010
0110110

Negative Example

0
00
100
010
001
0000
00000
0000000

Another example 1

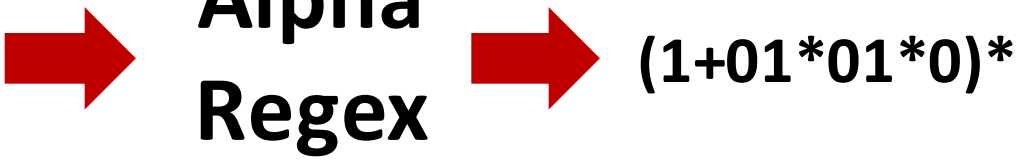
The number of 0s is divisible by 3. (Σ={0,1})

Positive Example

1
11
111
000
000000
01010
0110110

Negative Example

0
00
100
010
001
0000
00000
0000000



Another example 2

0 and 1 alternate. (Σ={0,1})

Positive Example

0
1
01
10
101
010
1010
0101
10101

Negative Example

00
11
011
001
110
1001
101101
010010

Another example 2

0 and 1 alternate. (Σ={0,1})

Positive Example

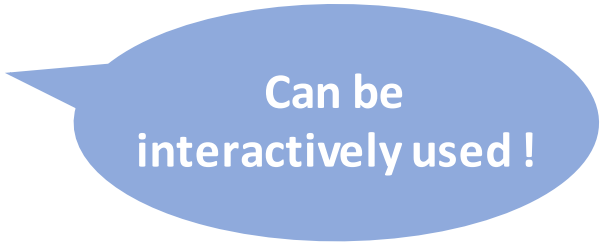
0
1
01
10
101
010
1010
0101
10101

Negative Example

00
11
011
001
110
1001
101101
010010

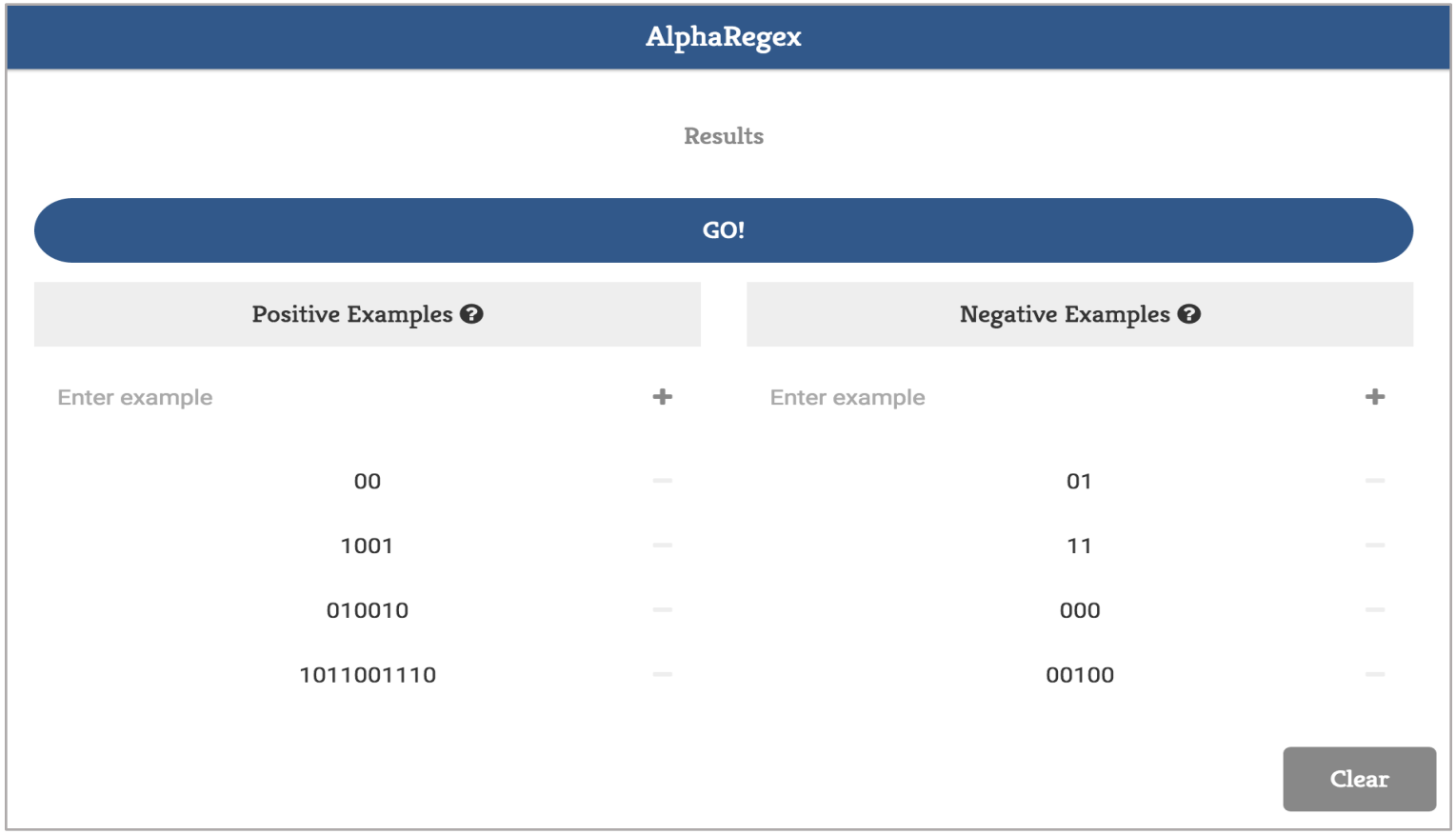
Contribution

- **Efficient Synthesis Algorithms for REs.**
  - **6.7 seconds on average.**



- **Provide AlphaRegex, publicly available.**
  - http://prl.korea.ac.kr/AlphaRegex

Demo Page

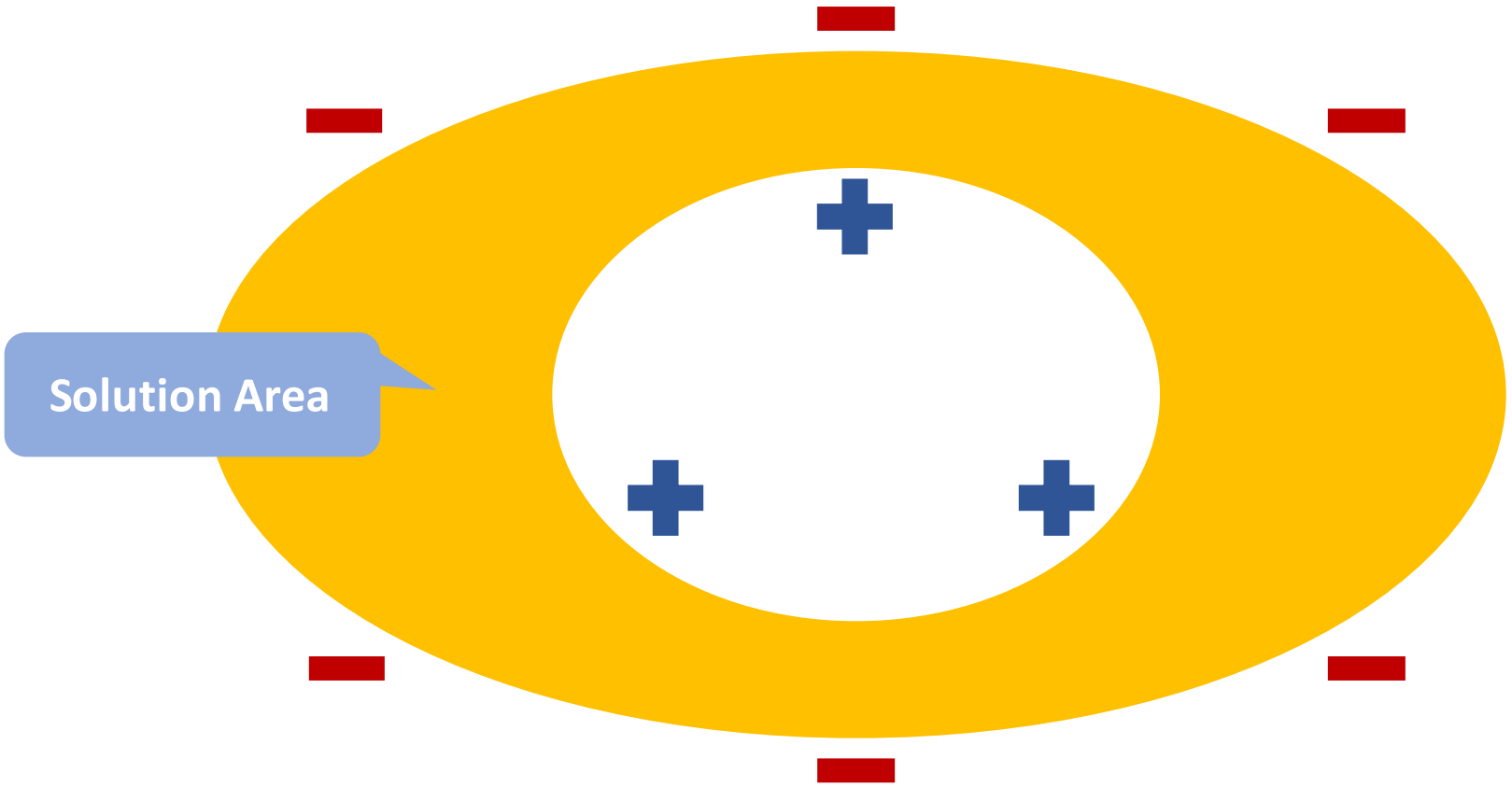

AlphaRegex
Results
GO!
Positive Examples
Negative Examples
Enter example
Enter example
00
1001
010010
1011001110
01
11
000
00100
Clear

RE in Automata Class

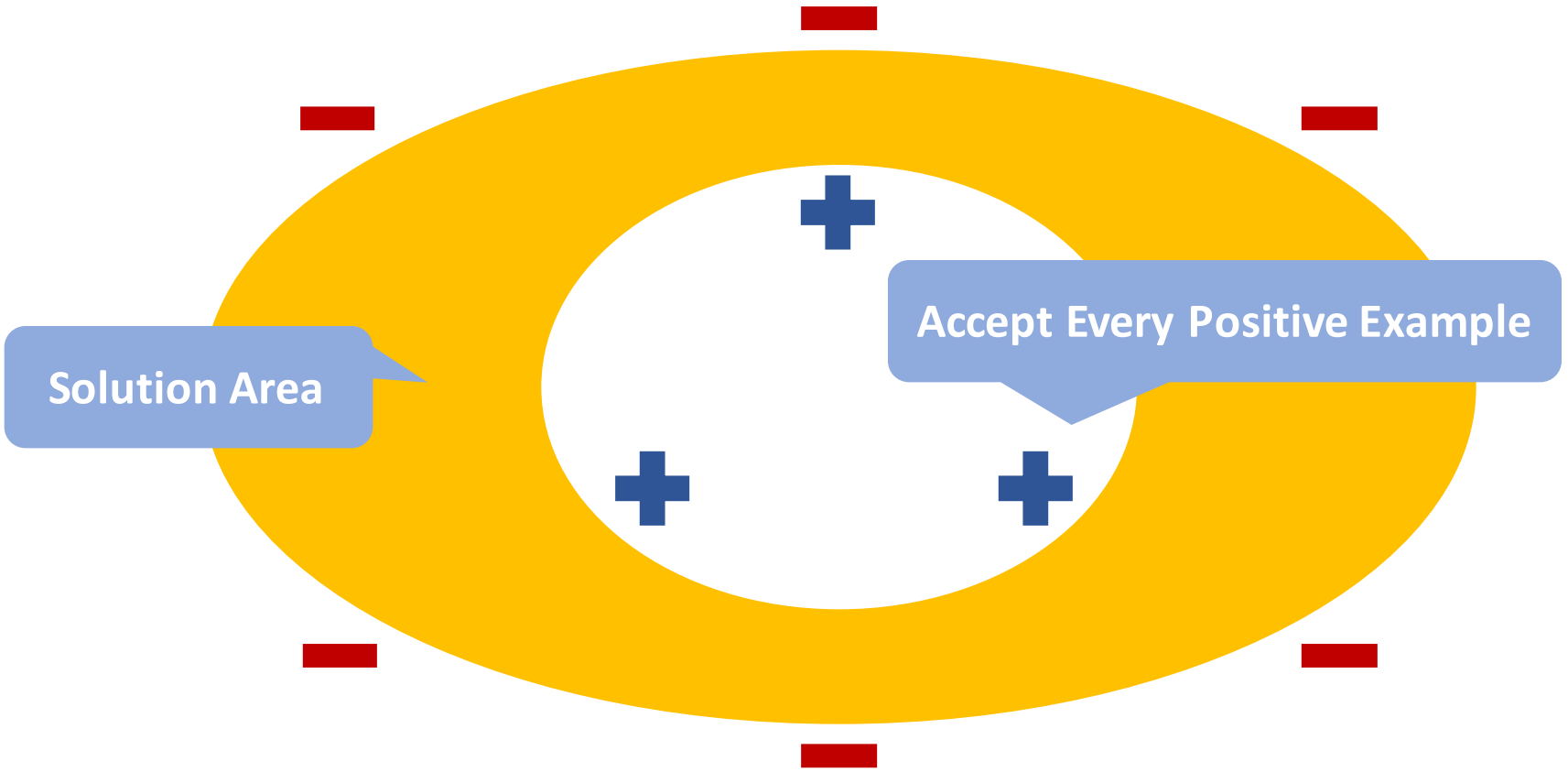
$$e \rightarrow a \in \Sigma \mid \epsilon \mid \emptyset \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e^*$$

- **Assume Σ={0,1}.**

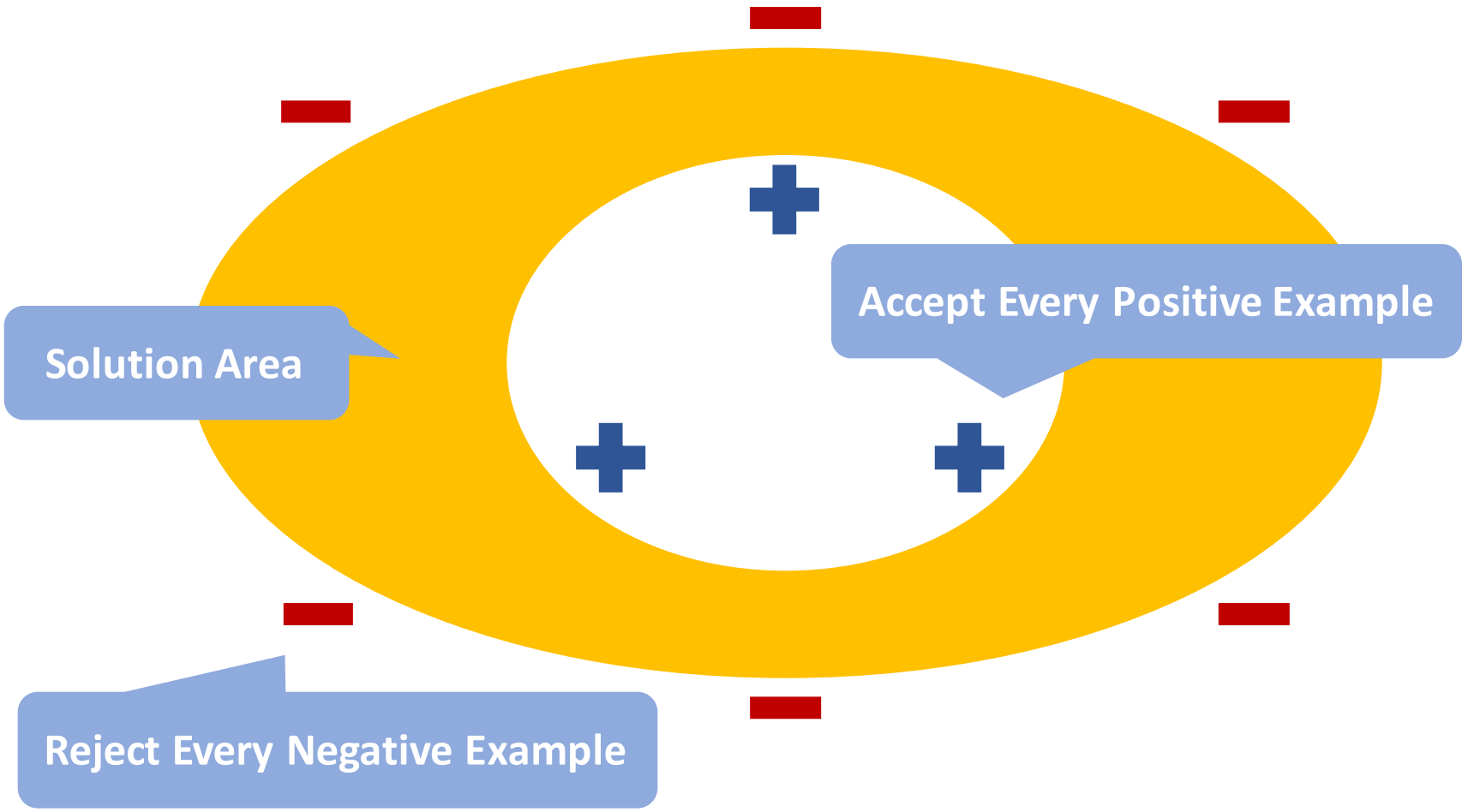
RE Synthesis Goal



RE Synthesis Goal



RE Synthesis Goal

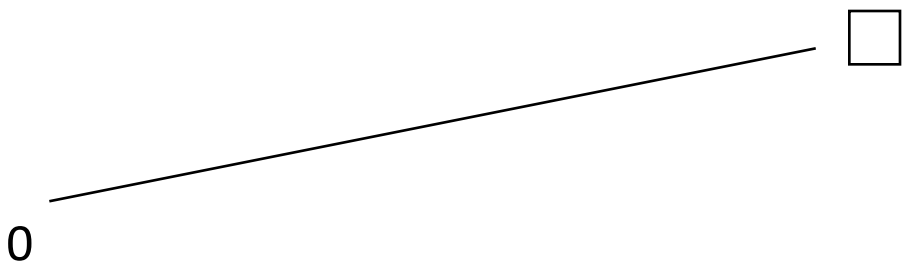


Basic Search Algorithm

□

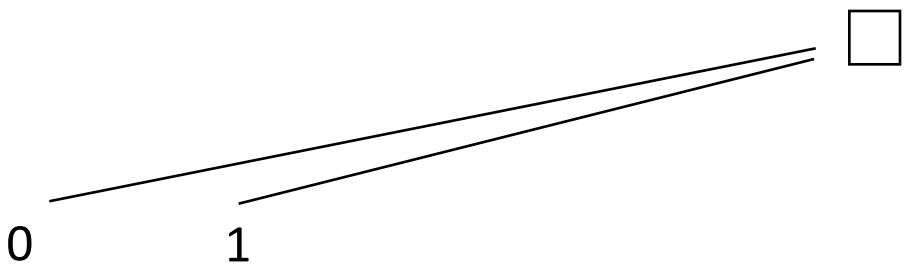
- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Basic Search Algorithm



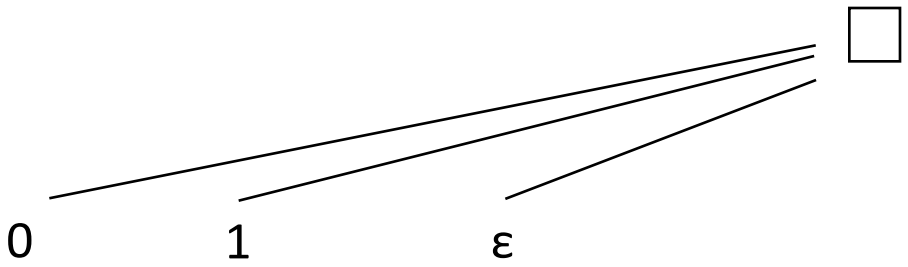
- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Basic Search Algorithm



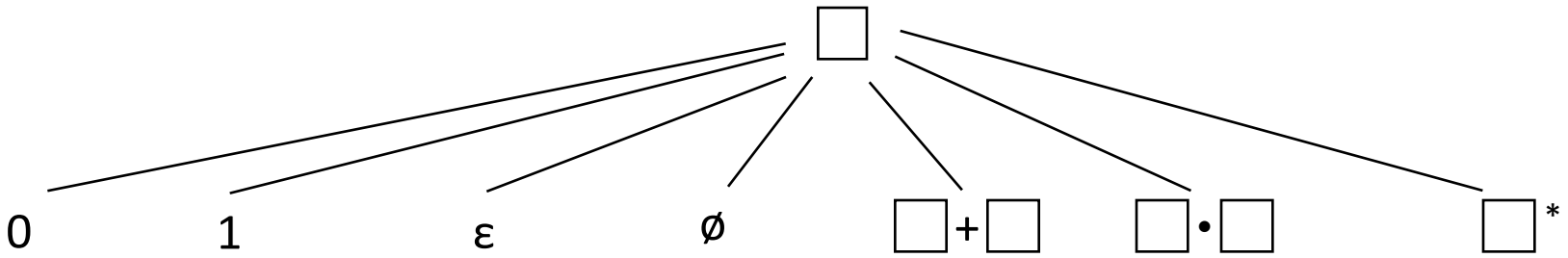
- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Basic Search Algorithm



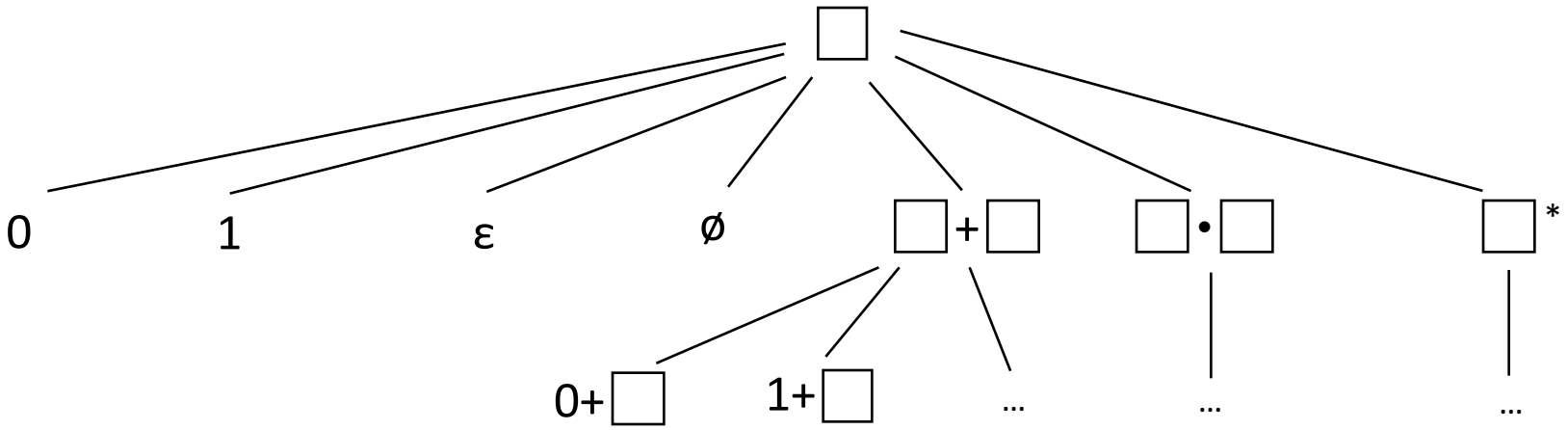
- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Basic Search Algorithm



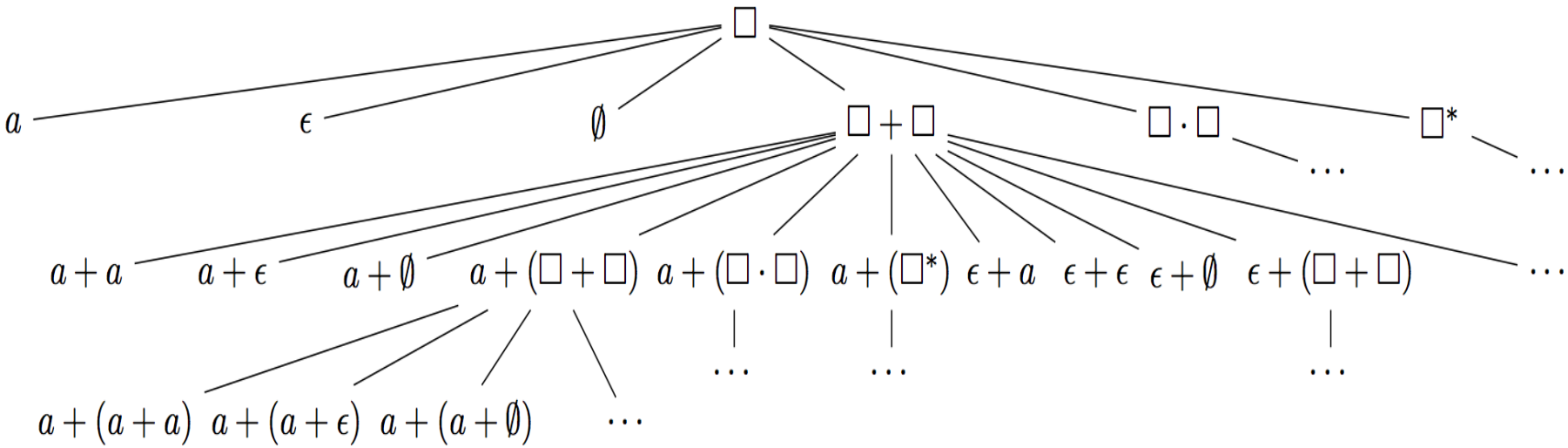
- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Basic Search Algorithm



- **Brute force search**
  - **Start from the initial hole(□).**
  - **Enumerate every possible regular expressions.**
  - **Check consistency with examples for each expression.**

Challenge



$$O(7^{2^d-1})$$

Maximum number of states at depth d.

Our Pruning Techniques

1. Identifying Semantically Equivalent States

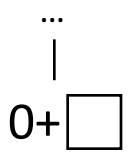
2. Over Approximations

3. Under Approximations

4. Identifying Redundant States

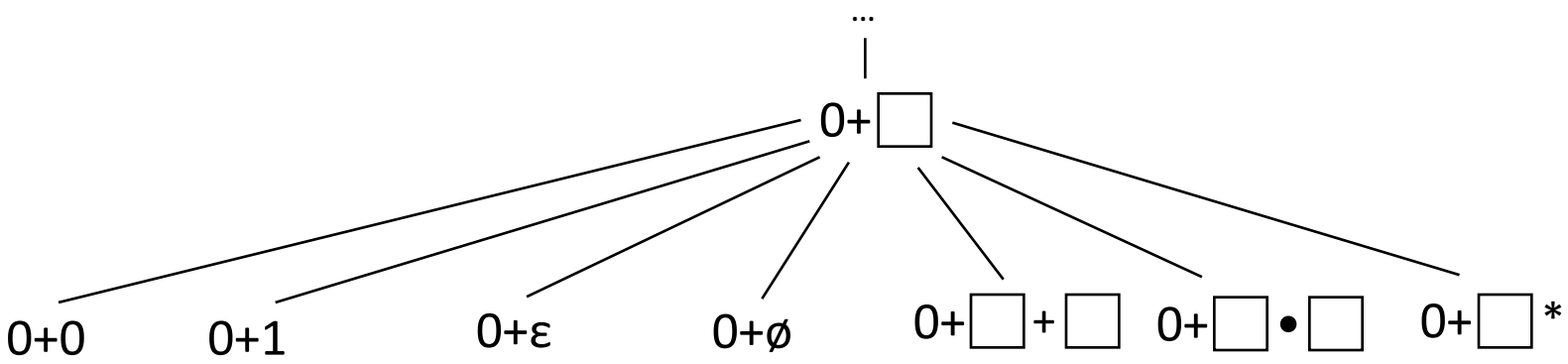
1. Equivalent states

Do not explore more if the equivalent state was explored.



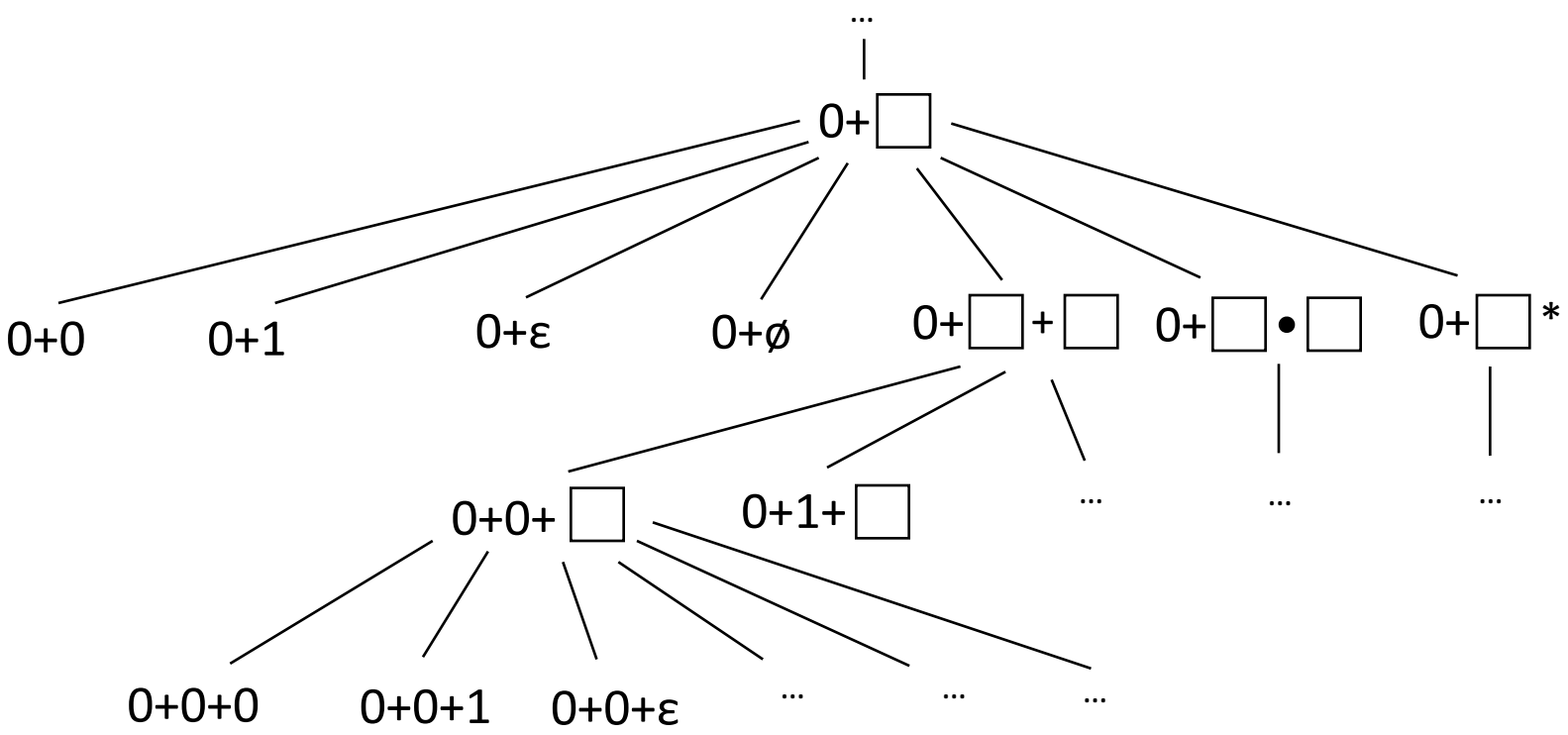
1. Equivalent states

Do not explore more if the equivalent state was explored.



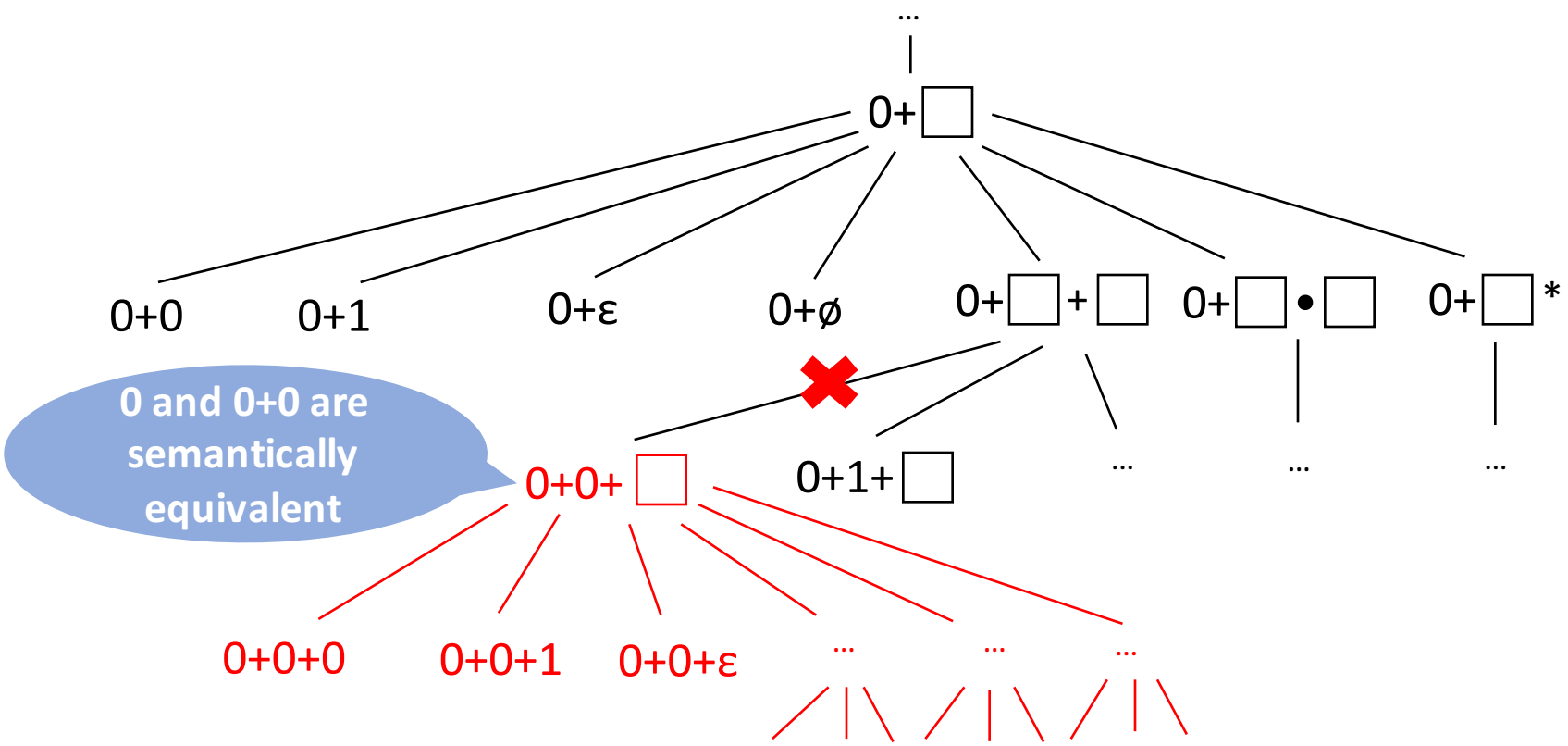
1. Equivalent states

Do not explore more if the equivalent state was explored.



1. Equivalent states

Do not explore more if the equivalent state was explored.



1. Equivalent states

Do not explore more if the equivalent state was explored.

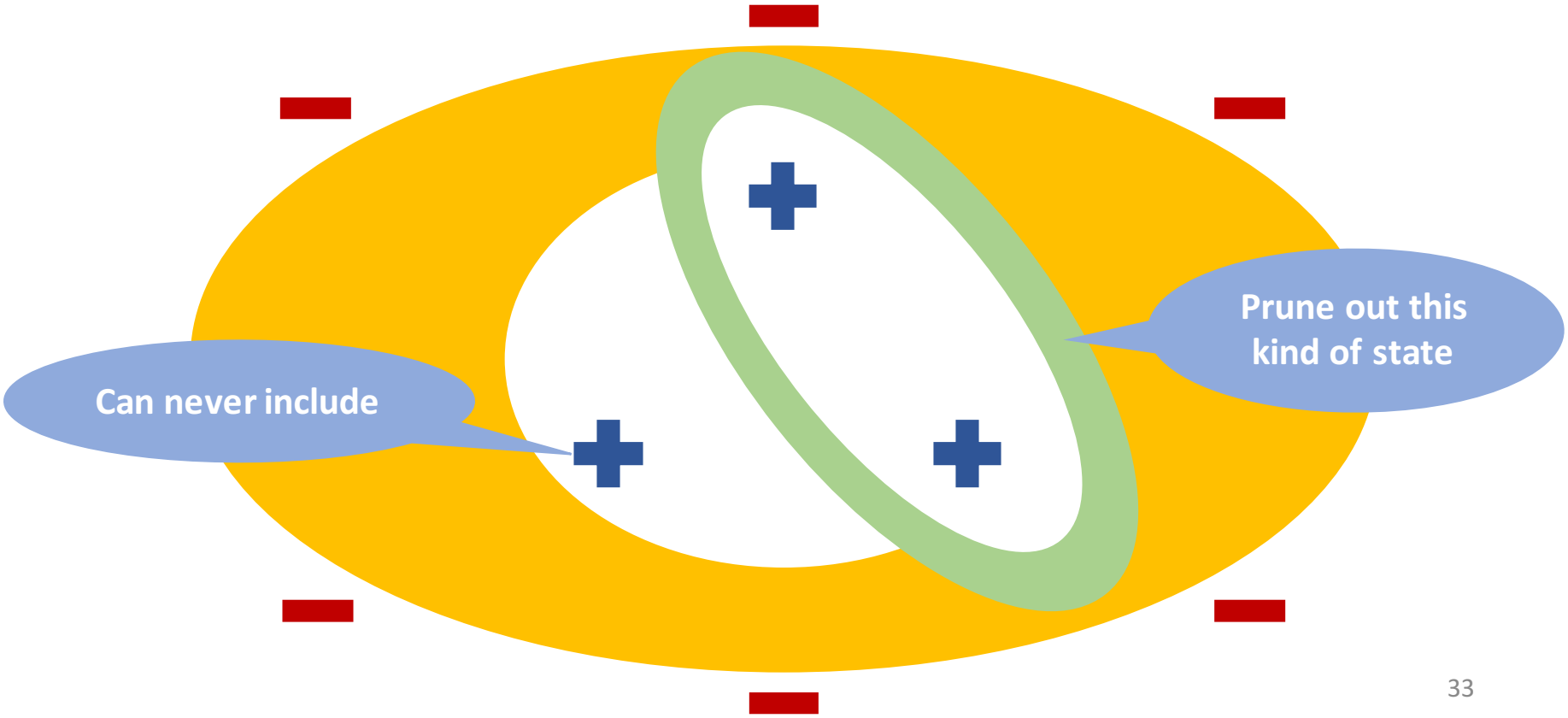
- **How can we identify semantically equivalent states?**

  - Semantics-preserving transformation rules.

  eg. e+e -> e, (e*)* -> e*

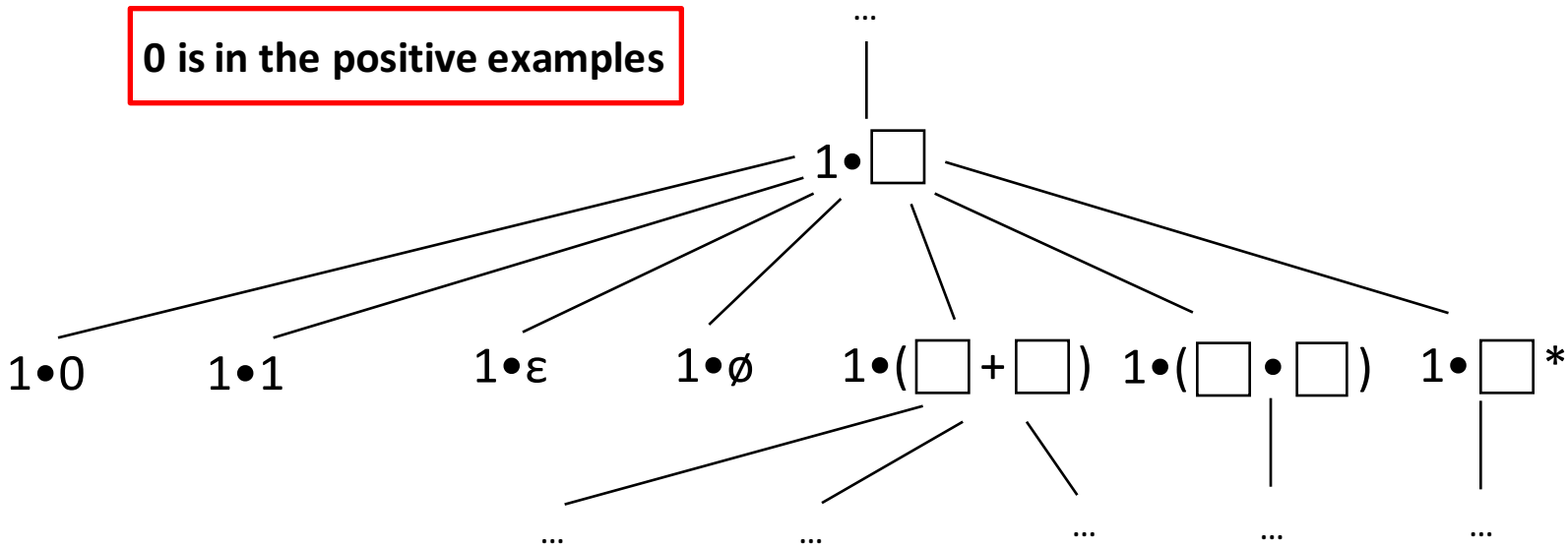
2. Over Approximations

Prune out the state guaranteed to
reject some positive examples.



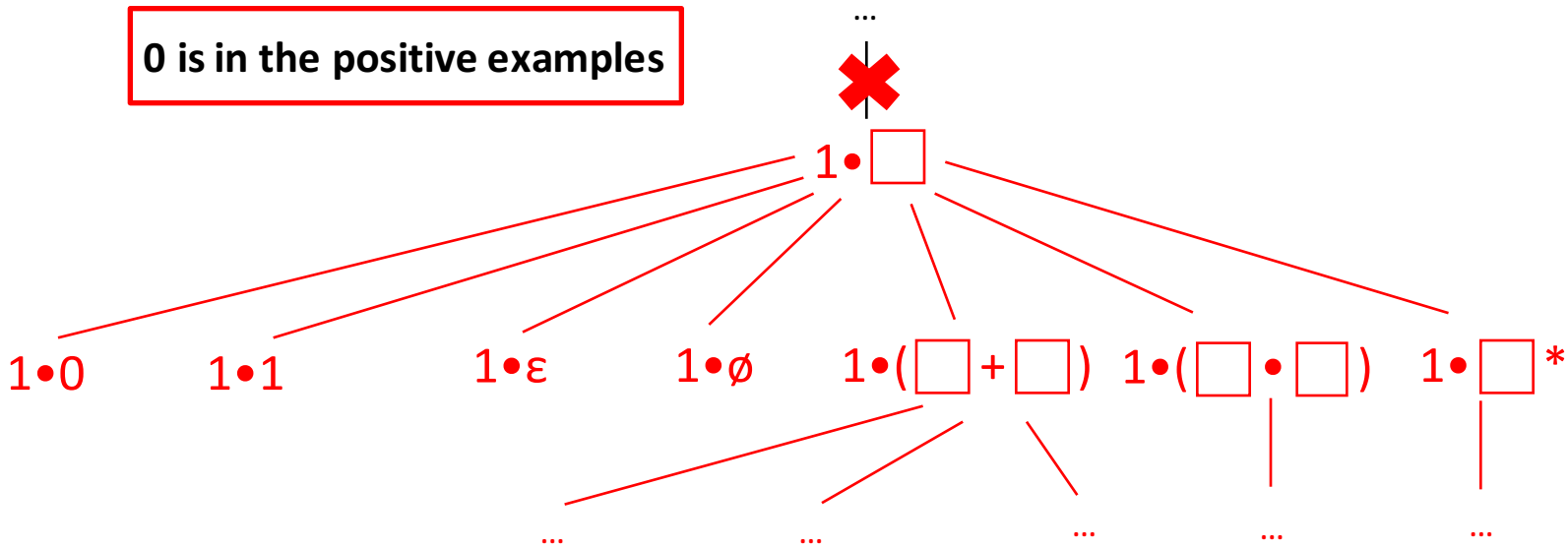
2. Over Approximations

Prune out the state guaranteed to reject some positive examples.



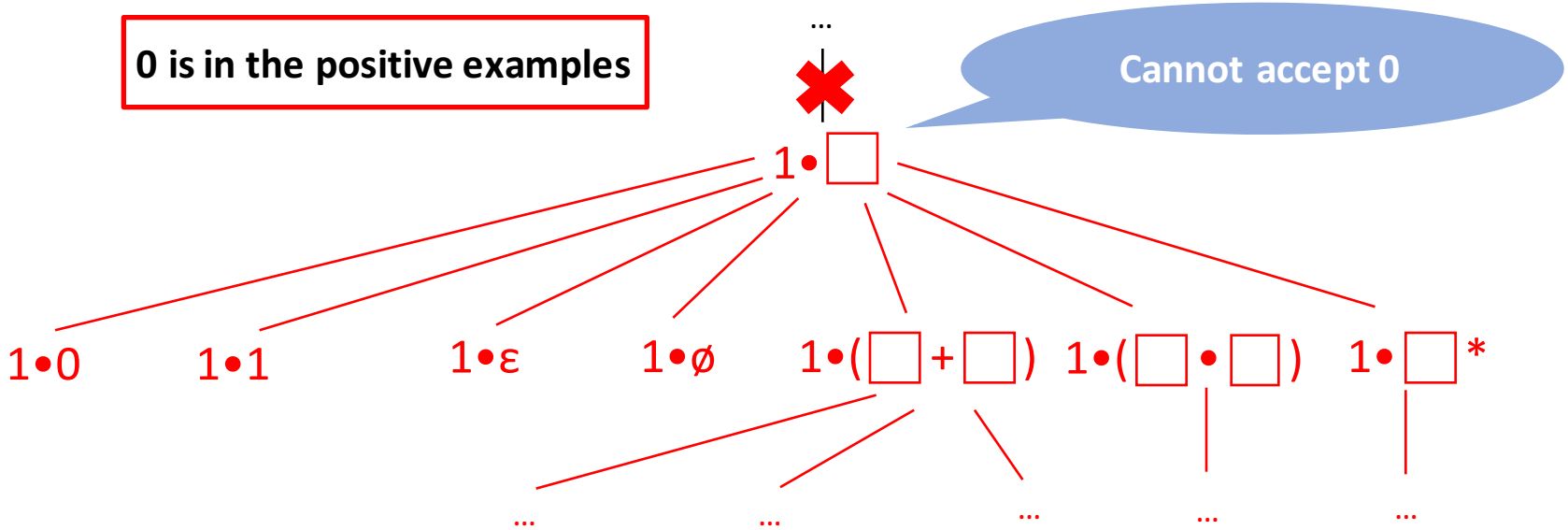
2. Over Approximations

Prune out the state guaranteed to reject some positive examples.



2. Over Approximations

Prune out the state guaranteed to reject some positive examples.



2. Over Approximations

Prune out the state guaranteed to reject some positive examples.

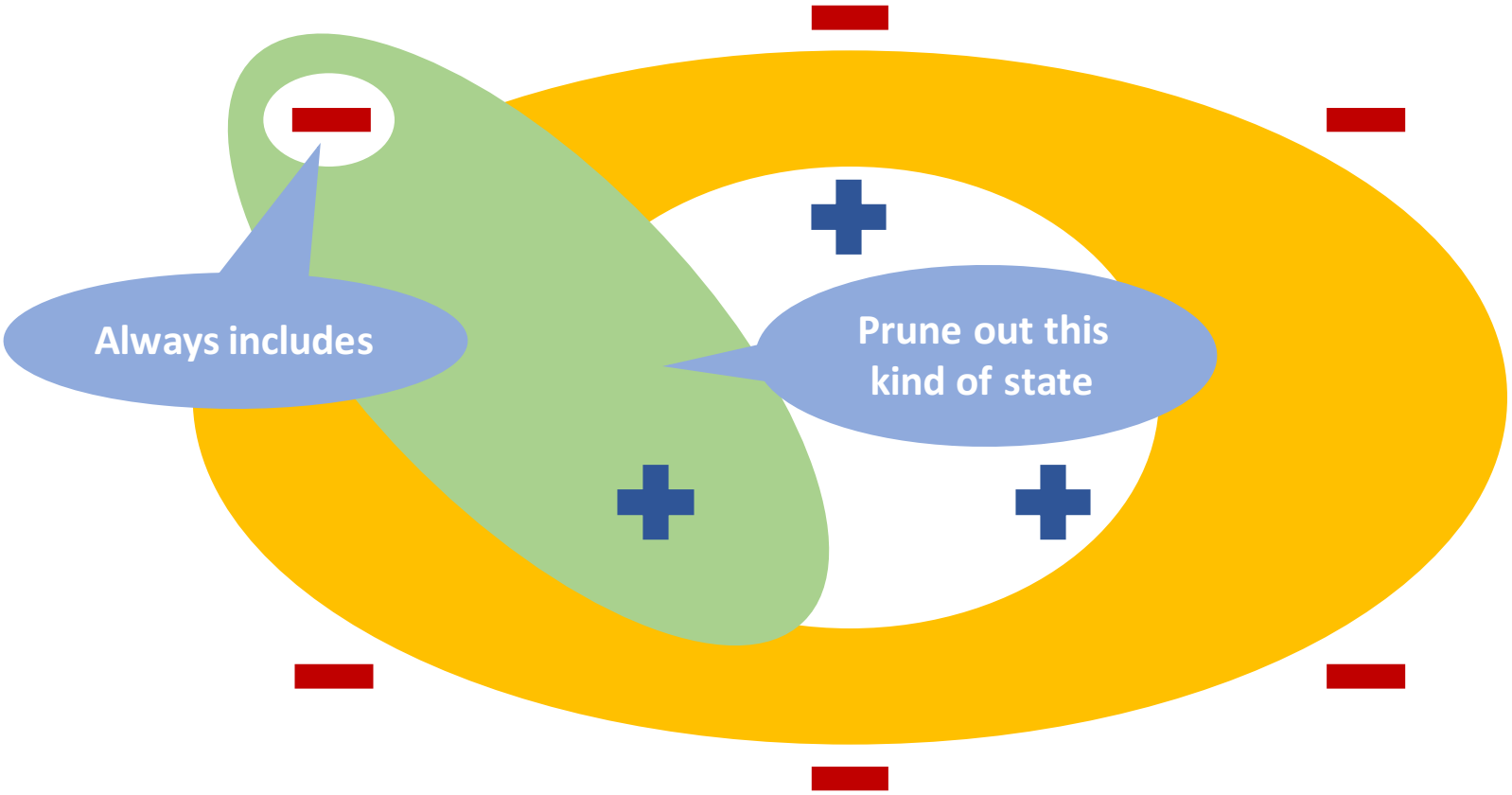
- **How can we over approximate the states?**

  1. Replace all holes(□) with Σ*(= (0+1)*).

  2. Check if it can accept every positive example.

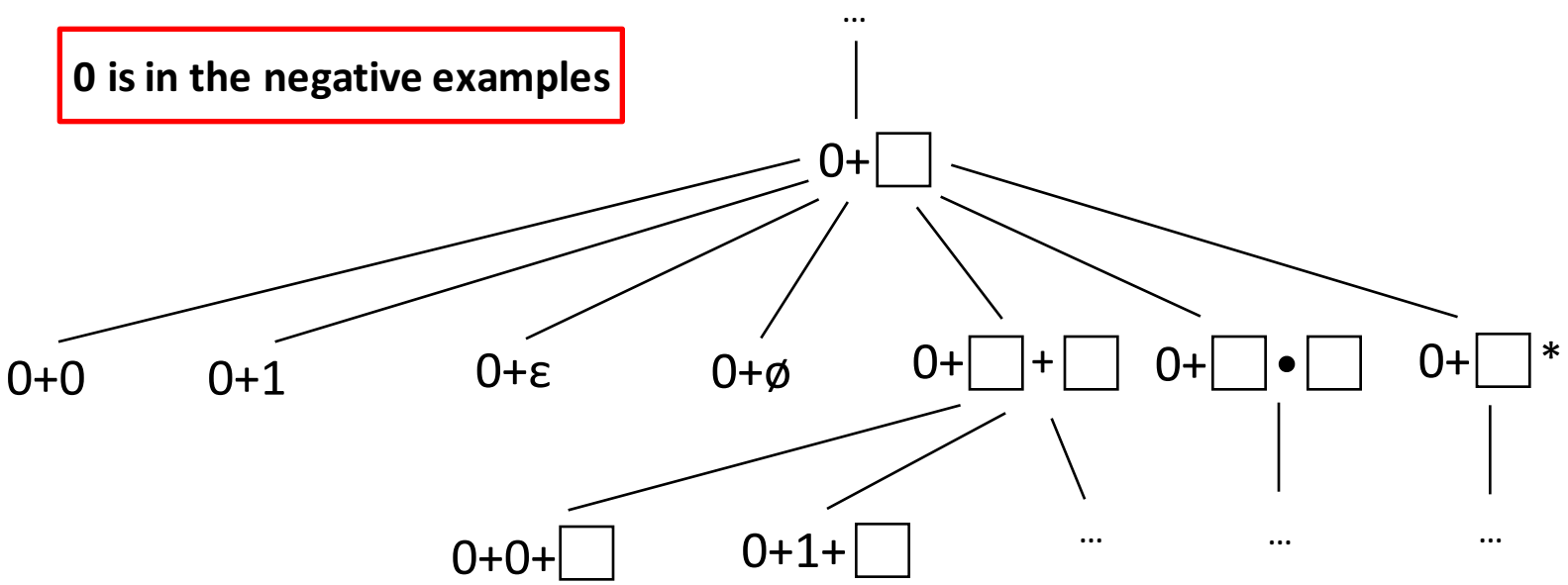
3. Under Approximations

Prune out the state guaranteed to accept some negative examples.



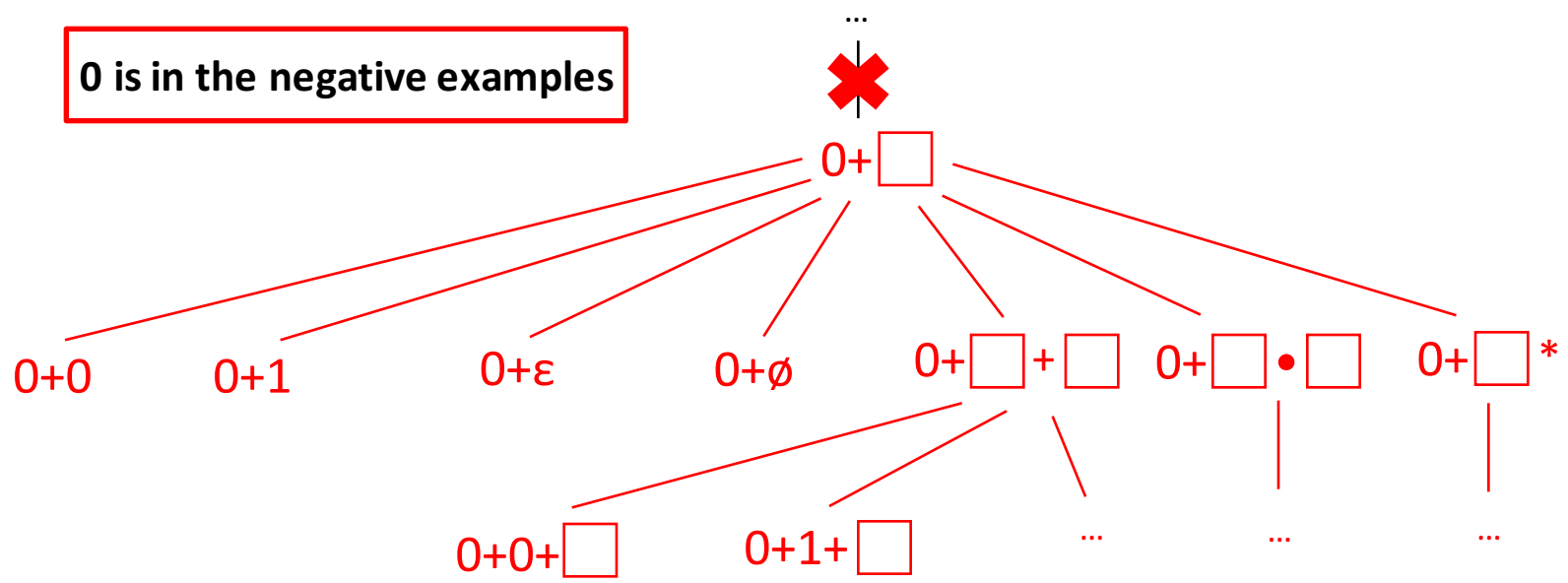
3. Under Approximations

Prune out the state guaranteed to accept some negative examples.



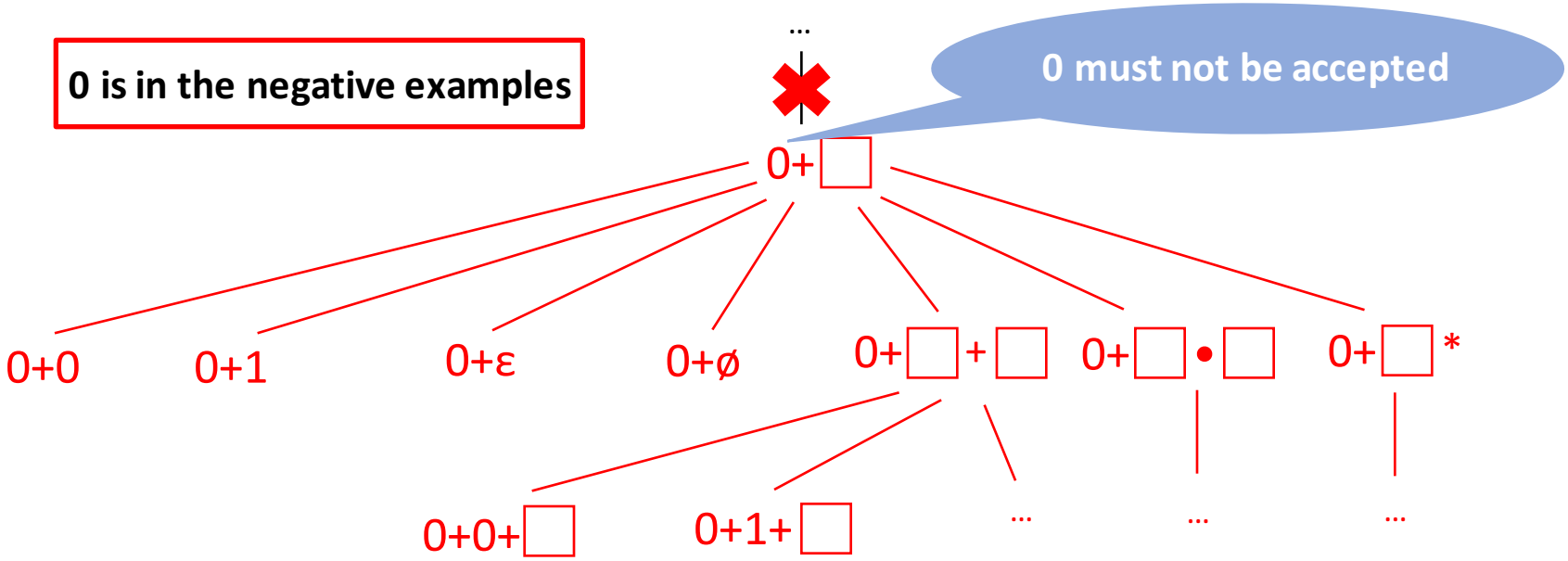
3. Under Approximations

Prune out the state guaranteed to accept some negative examples.



3. Under Approximations

Prune out the state guaranteed to accept some negative examples.



3. Under Approximations

Prune out the state guaranteed to accept some negative examples.

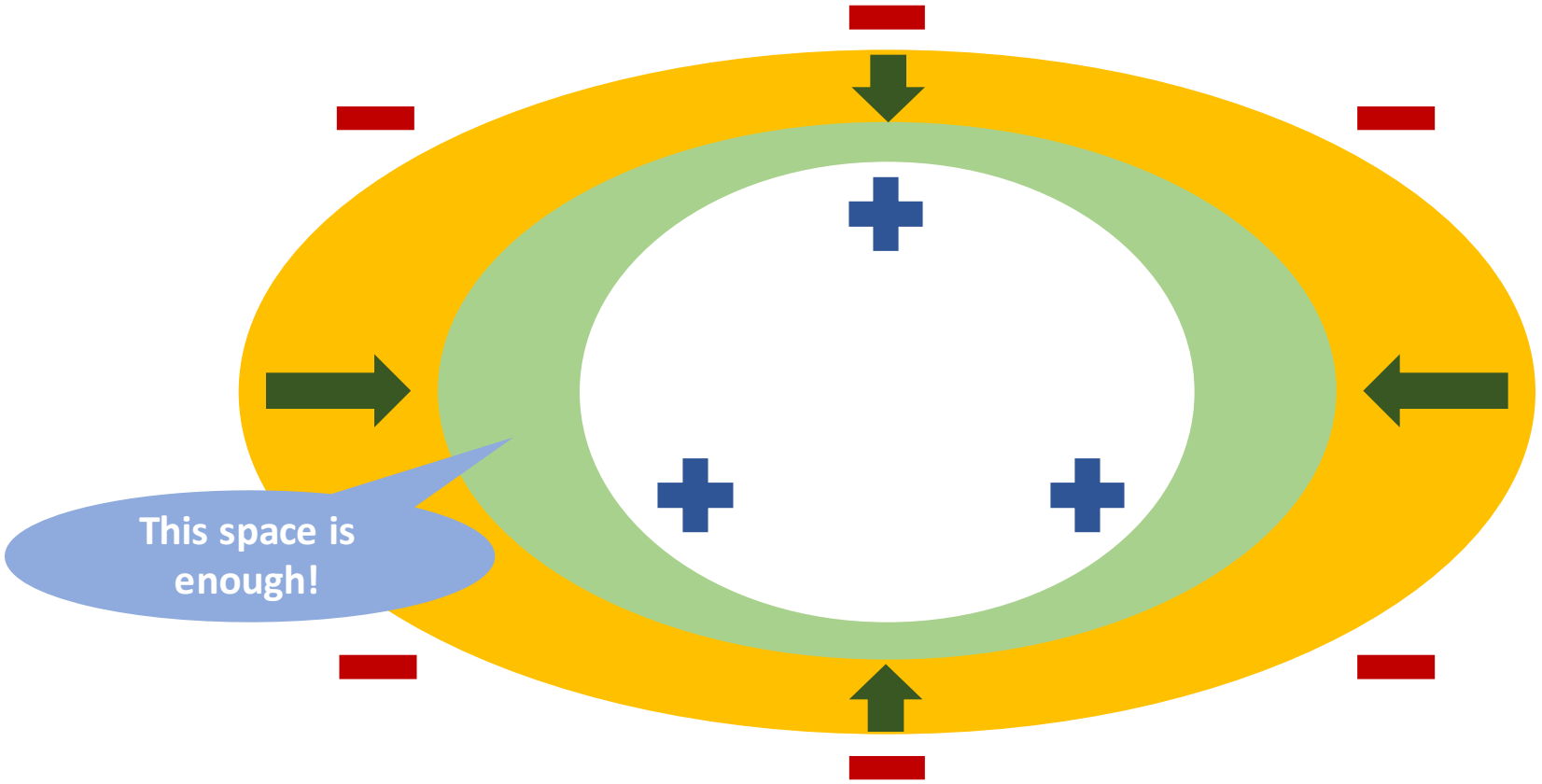
- **How can we under approximate the states?**

  1. Replace all holes(□) with ∅.

  2. Check if it accepts any negative examples.

4. Redundant States

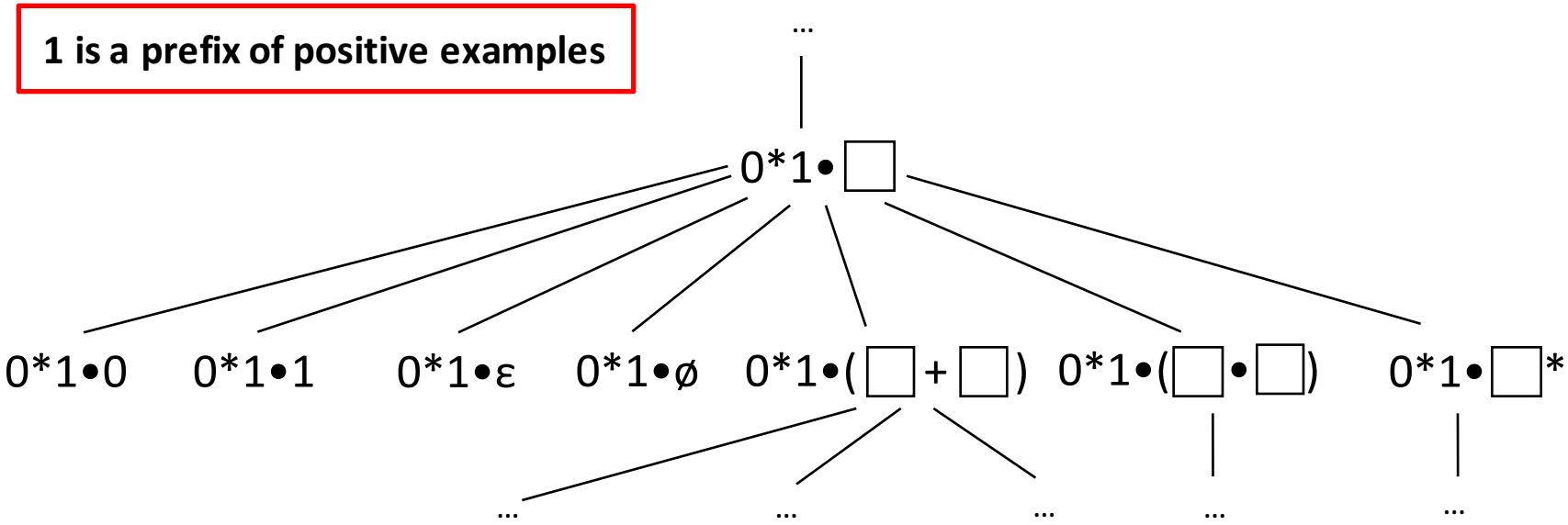
Prune out the state
if the state has redundant symbols.



4. Redundant States

Prune out the state
if the state has redundant symbols.

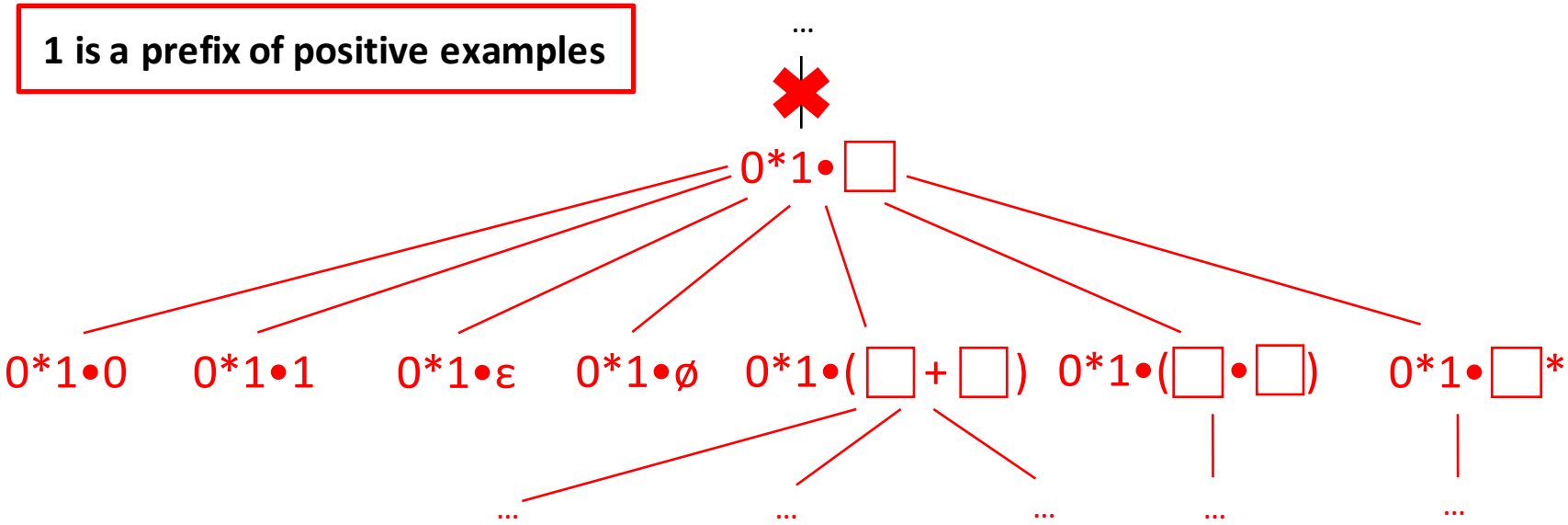
1 is a prefix of positive examples



4. Redundant States

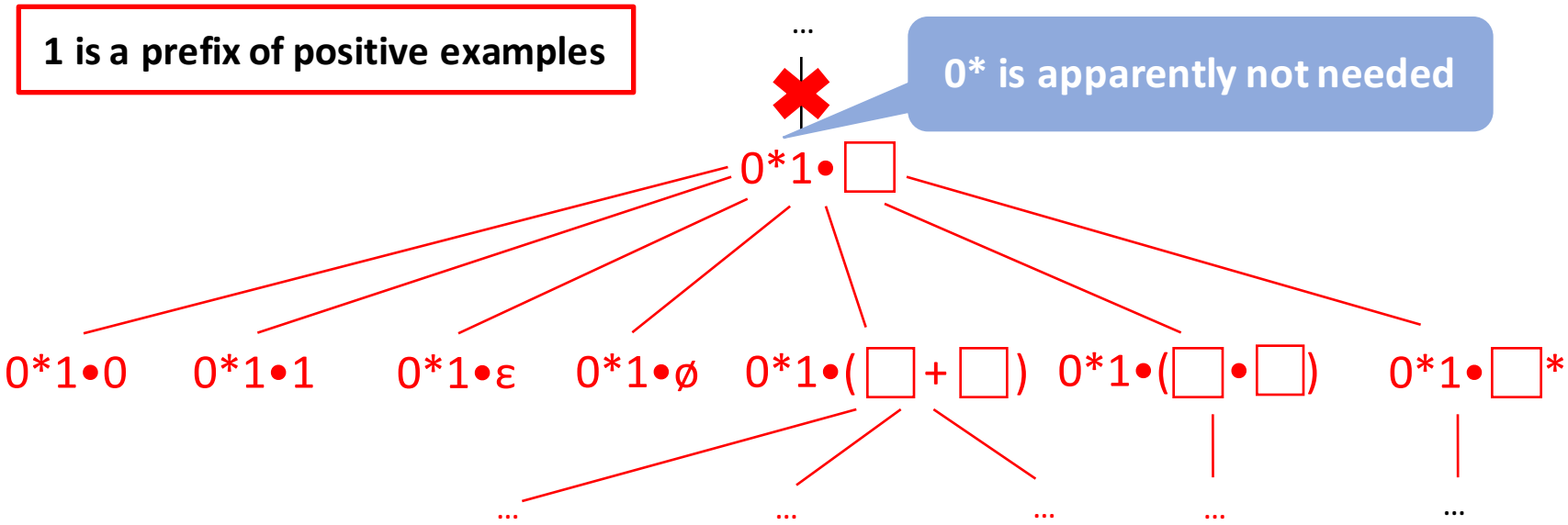
Prune out the state
if the state has redundant symbols.

1 is a prefix of positive examples



4. Redundant States

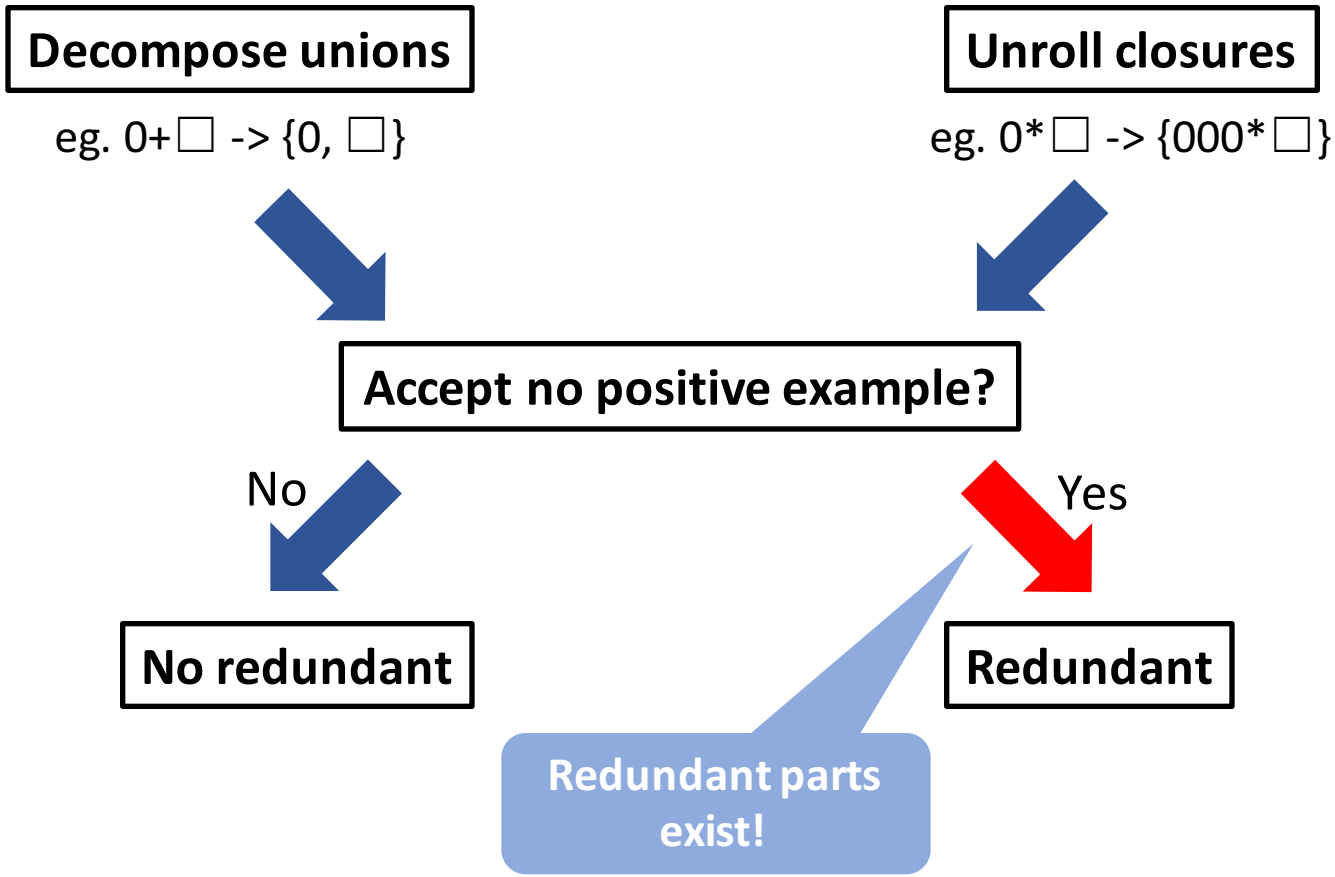
Prune out the state
if the state has redundant symbols.



4. Redundant States

- **How can we identify redundant states?**

  - Identify redundant unions and closures.



Completeness

If there exists a solution for the given examples,

it will be eventually found.

Evaluation

- **25 problems from textbooks.**
- **3 levels of difficulty.**

  - Easy, Normal, Hard
- **Positive, Negative examples are < 10 for each problem, respectively.**

Evaluation

	Full	All-Off	No Apr	No Eq	No Rd
Average (Time : sec)	6.7	> 6599.3	284.8	27.8	18.8

Conducted on an Ubuntu machine with Intel Xeon CPU E5-2630 (2.40GHz).

- **Average runtime : 6.7 sec**

  - Worst case < 1 min

Evaluation

	Full	All-Off	No Apr	No Eq	No Rd
Average (Time : sec)	6.7	> 6599.3	284.8	27.8	18.8

Conducted on an Ubuntu machine with Intel Xeon CPU E5-2630 (2.40GHz).

- **Exhaustive Search Only (All-Off)**

  - 5 of 25 : > 10,000 sec

Evaluation

	Full	All-Off	No Apr	No Eq	No Rd
Average (Time : sec)	6.7	> 6599.3	284.8	27.8	18.8

Conducted on an Ubuntu machine with Intel Xeon CPU E5-2630 (2.40GHz).

- **Effectiveness**
  1. **Over- and Under Approximations (X 42.5 ↑)**
  2. **Identifying Equivalent states (X 4.1 ↑)**
  3. **Identifying Redundant states (X 2.8 ↑)**

Conclusion

- RE synthesizer for automata class
- Exhaustive search with simple pruning
- Complete : Guaranteed to find a solution

Thank you !