

AlphaRegex

정규표현식 자동 합성기

1. 연구 동기

계산이론 수업을 듣다가: 정규식 합성을 자동으로 할 수 없을까?

$\Sigma = \{0, 1\}$ 에 대해, 다음 언어에 대한 정규식을 찾으시오.

$L = \{w \in \{0, 1\}^* \mid w$ 는 정확히 한 쌍의 연속인 0들을 갖는다. $\}$



옳은 예	틀린 예
00,	01,
1001,	11,
0101001010	000,
1111001111	00100

2. 문제 및 목표

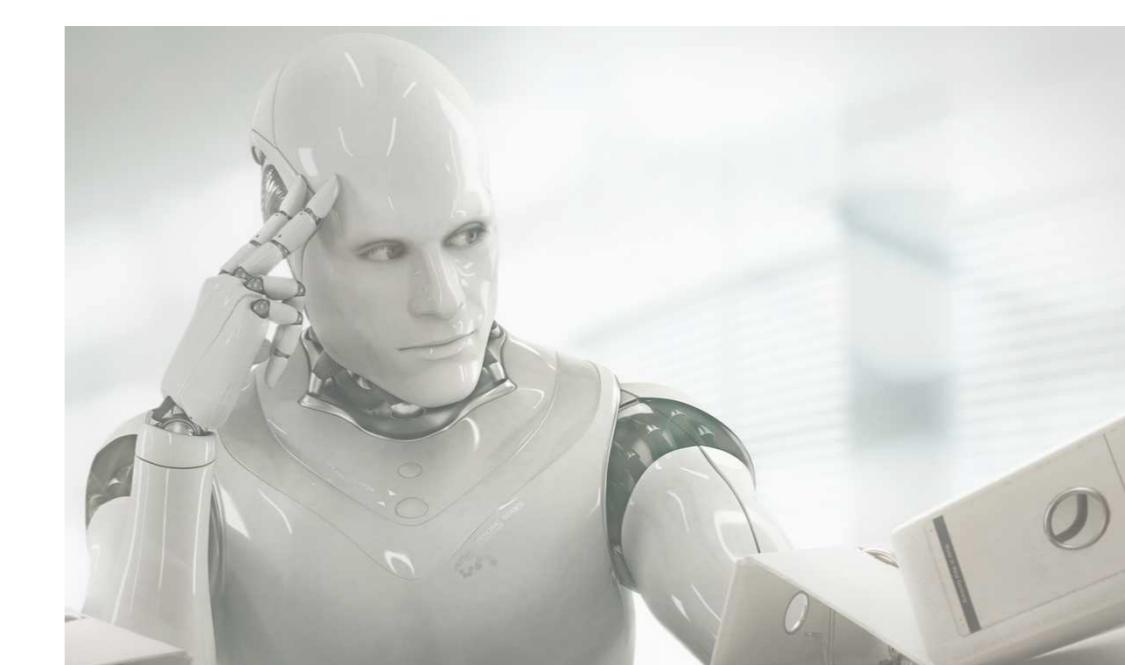
주어진 예제를 만족하는 정규식을 자동 합성하기:

옳은 예(Positive examples)

00,
1001,
0101001010
1111001111

틀린 예(Negative examples)

01,
11,
000,
00100



자동 합성된 정규표현식

$(0?1)^*00(10?)^*$
(in 0.5s)

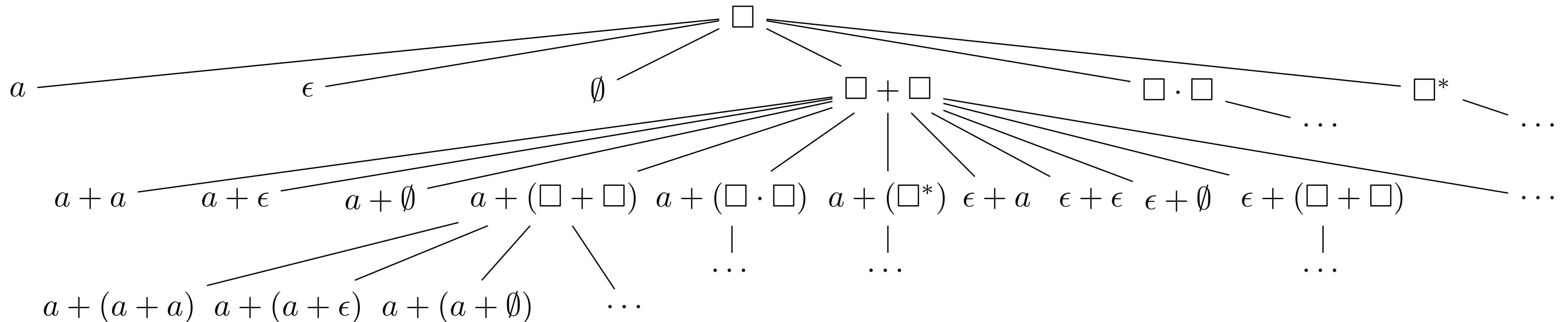
AlphaRegex

목표: 계산이론 수강생과 교수님보다 똑똑하게!

3. 정규식 합성 알고리즘

기본 알고리즘 정규식 문법으로 생성되는 모든 상태공간 탐색

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

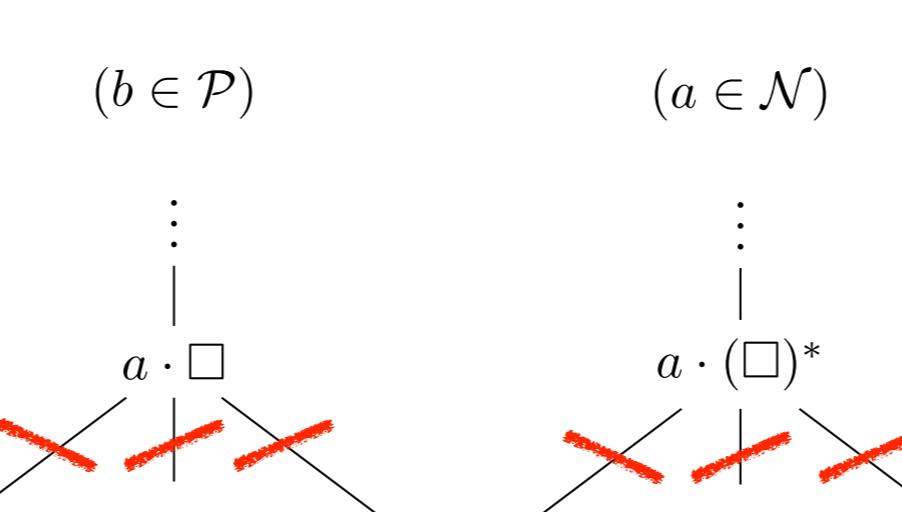


해결 방법 효율적인 공간 탐색 고안

1. 간단한 정규식 우선탐색 (Best-first Enumerative Search)

$$\begin{aligned} C(a) &= C(\epsilon) = C(\emptyset) = c_1 \\ C(\square) &= c_2 (c_2 > c_1) \\ C(e_1 + e_2) &> C(e_1) + C(e_2) \\ C(e_1 \cdot e_2) &> C(e_1) + C(e_2) \\ C(e^*) &> C(e) \end{aligned}$$

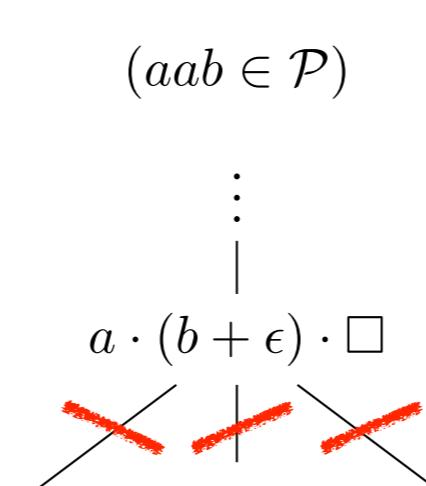
3. 해를 가질수 없는 상태(Dead States) 가지치기



2. 같은 의미 상태(Semantically-Equivalent States) 가지치기

$$\begin{aligned} [s^* s^*] &= [s^*] \\ [(s + s)] &= [s] \\ [(s \cdot s^*)^*] &= [s^*] \end{aligned}$$

4. 불필요한 상태(Redundant States) 가지치기



고안한 가지치기 기법들은 프로그래밍 언어 이론에 기반하여 결과의 안전성(Soundness)을 보장.

Definition 1 (Dead States). Let (P, N) be a regular expression problem. We say a state $s \in S$ is dead, denoted $\text{dead}(s)$, iff every closed state s' reachable from s is not a solution:
 $\text{dead}(s) \iff ((s \rightarrow^* s') \wedge s' \not\rightarrow \Rightarrow \neg \text{solution}(s'))$.

Lemma 4. Let s be any state. Then,
 $\text{pdead}(s) \iff \exists p \in P, p \notin [s]$.
Proof. Consider each direction.
• (\Rightarrow) Suppose $\text{pdead}(s)$ holds.
 $s \rightarrow^* s' \wedge s' \not\rightarrow \Rightarrow \exists p \in P, p \notin [s']$. (5)
From (5) and Lemma 6, we obtain $\exists p \in P, p \notin [s]$.
• (\Leftarrow) Suppose $p \notin [s]$. By Lemma 2, we have
 $p \notin \bigcup_{s \rightarrow^* s' \wedge s' \not\rightarrow \Rightarrow} [s']$,
which implies that $p \notin [s']$ for all closed s' reachable from s . \square

Lemma 5. Let s be any state. Then,
 $\text{ndead}(s) \iff \exists n \in N, n \in [s]$.
Proof. Consider each direction.
• (\Rightarrow) Suppose $\text{ndead}(s)$ holds.
 $s \rightarrow^* s' \wedge s' \not\rightarrow \Rightarrow \exists n \in N, n \in [s']$. (6)
From (6) and Lemma 7, we obtain $\exists n \in N, n \in [s]$.
• (\Leftarrow) Suppose $n \in [s]$. By Lemma 3, we have
 $n \in \bigcap_{s \rightarrow^* s' \wedge s' \not\rightarrow \Rightarrow} [s']$,
which implies that $n \in [s']$ for all closed s' reachable from s . \square

4. 실험

- ✓ 840 lines in OCaml
- ✓ 학생들이 어려워하는 정규식 문제를 위주로
- ✓ 탐색 기법을 하나도 적용하지 않은 기본 알고리즘을 비교군으로
- ✓ 탐색 기법을 모두 적용한 알고리즘의 성능 및 향상 평가

5. 결론

- ✓ 적은 수의 예로부터 사람도 풀기 어려워 하는 정규식을 빠르게 합성
- ✓ 효율적으로 상태공간을 탐색하기 위한 다양한 탐색 기법 제시
- ✓ 실제 계산이론 책에 등장하는 고난이도 문제를 통해 성능 입증

문제	예제 개수		합성된 정규식	소요 시간 (초)		속도향상
	P	N		기본 알고리즘	우리 알고리즘	
w는 오른쪽으로부터 5번째 글자가 1이다.	3	3	$(0+1)^*1(0+1)(0+1)(0+1)(0+1)$	148.0	8.2	18x
w는 최대 두 개의 0을 가진다.	8	7	$1^*021^*021^*$	425.0	1.2	354x
w는 0과 1이 번갈아가며 등장한다.	10	11	$0?(10)^*1?$	4073.9	1.6	2546x
w에 있는 0의 개수는 3으로 나누어 떨어진다.	8	7	$(1+01^*01^*0)^*$	> 7200.0	5.9	n/a
w가 0으로 시작하면 짜수의 길이를 가지고, 1로 시작하면 짝수의 길이를 가진다.	5	3	$(0+1(0+1))((0+1)(0+1))^*$	> 7200.0	10.9	n/a
w는 최소 1개의 0과 최대 1개의 1을 가진다.	12	10	$0^*(01?+100^*)$	> 7200.0	7.5	n/a
w는 최대 1쌍의 연속한 1을 가진다.	9	8	$(1+(01?)^*)(0+10^*)$	465.1	24.4	19x

실험 환경 MacBook Pro / OS X El Capitan 10.11.1 / 2.2 GHz Intel Core i7 / 16GB 1600 MHz DDR3