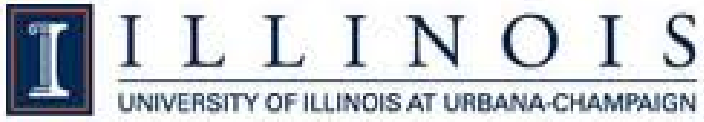
Automated Grading of DFA Constructions

Rajeev Alur (Penn), Loris D'Antoni (Penn), Sumit Gulwani (MSR), Bjoern Hartmann (Berkeley), Dileep Kini (UIUC), Mahesh Viswanathan (UIUC)



Microsoft®
Research

ExCAPE
Expeditions in Computer Augmented Program Engineering

Motivations

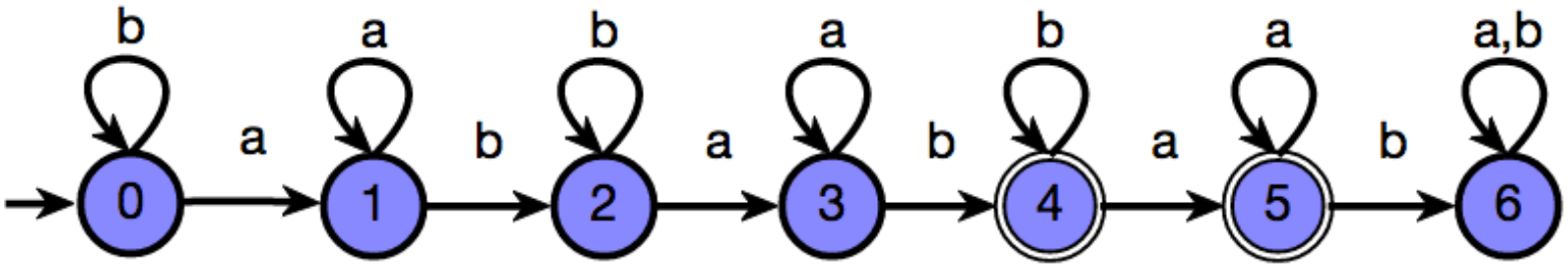
- Grading is a tedious and time consuming task
- Human grades are often inconsistent
- MOOCs (Massive Online Open Courses) admits thousands of students, infeasible to grade manually

The DFA Construction Problem

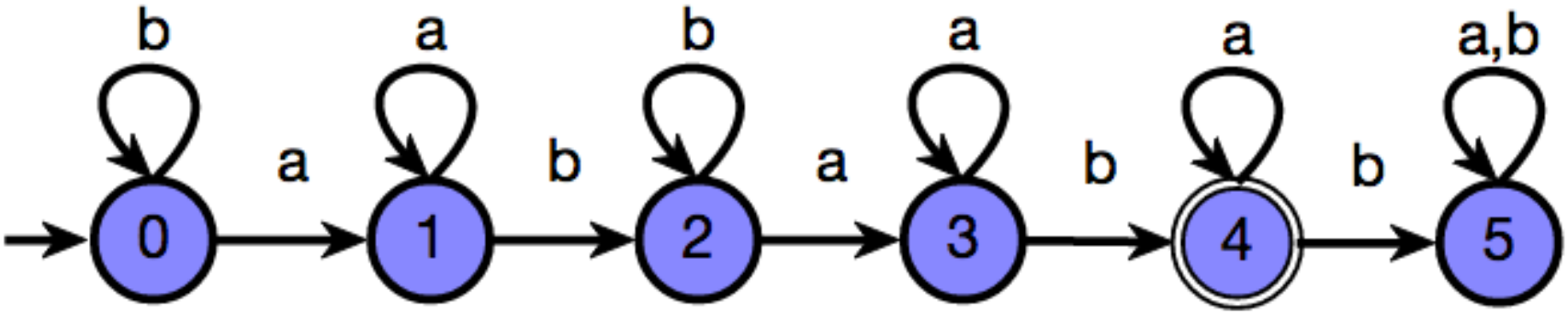
Draw the DFA accepting the language:

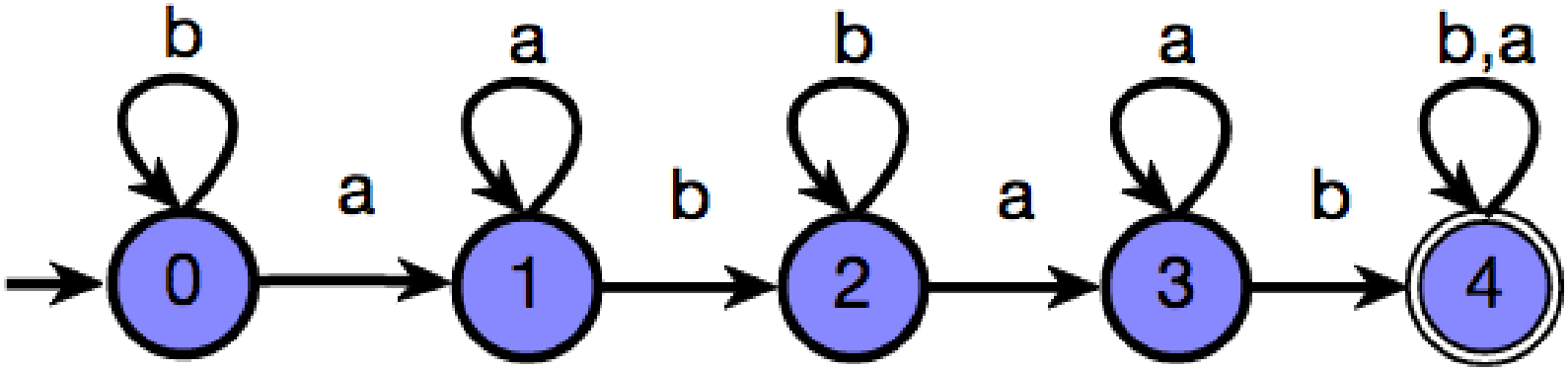
{ *s* | '*ab*' appears in *s* exactly *2* times }

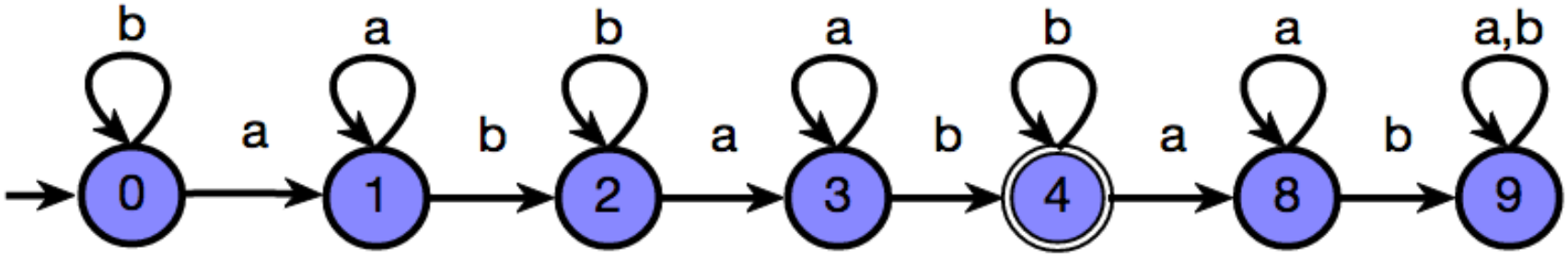
Solution:



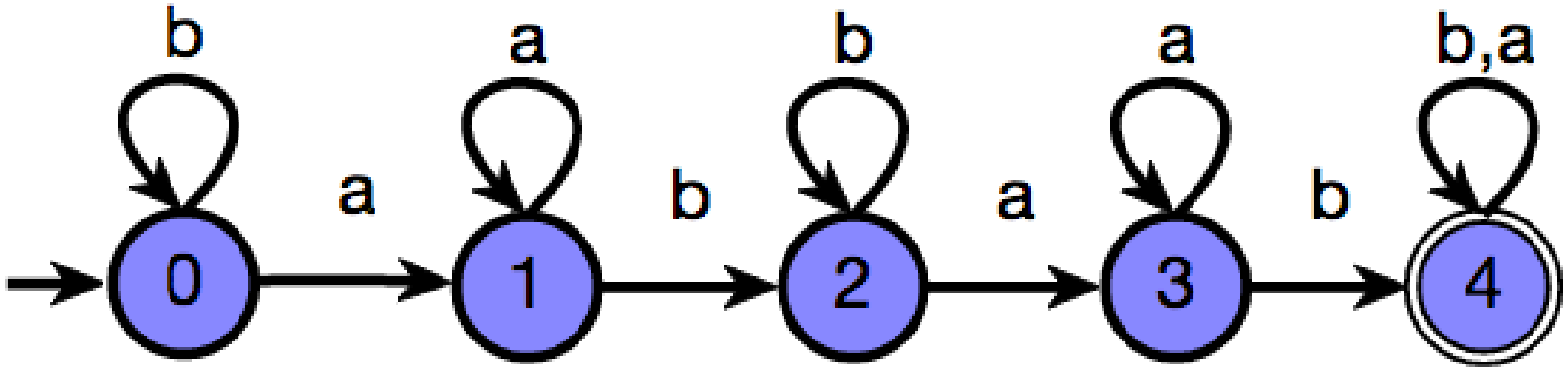
Student Solutions







Problem Syntactic Mistake



The problem description was

{ *s* | '*ab*' appears in *s* exactly *2* times }

The student instead drew DFA for

{ *s* | '*ab*' appears in *s* at least *2* times }

INTUITION: find the distance between the two language descriptions

Mosel: MSO + Syntactic Sugar

Mosel: similar to MSO; predicates describing DFAs

$$\text{sizeOf}(\text{indOf}(\text{‘}ab\text{’}))=2$$

$$\text{sizeOf}(\text{indOf}(\text{‘}ab\text{’}))\geq 2$$

- If we had such descriptions we could use tree edit distance to check how far they are from each other
- ``Easy'' to go from such Mosel predicates to automata (classical MSO to DFA algorithm)
- However, what we need is:

  given a DFA compute a (small) Mosel formula

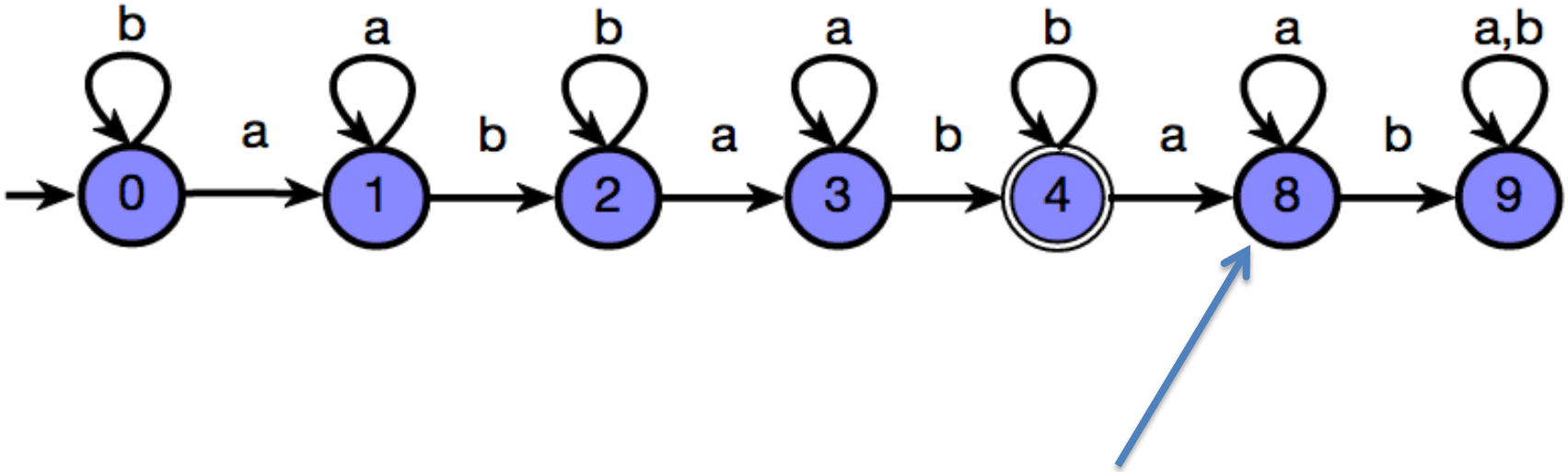
Brute Force Search

Enumerate all the predicates and check for equivalence with target DFA

Search pruning and speeding:

- Avoid trivially equivalent predicates (A V B, B V A)
- Approximate equivalence using set of test strings:
  - Generate sets of positive and negative examples that distinguish each state in the target DFA
  - One can prove all such strings are enough to prove inequivalent all DFAs of smaller size than target DFA

Solution Syntactic Mistake



The student forgot
one final state

INTUITION: find the smallest number of syntactic modification to fix solutions

DFA Edit Difference

Compute DFA edit distance:

- Number of edits necessary to transform the DFA into **a** correct one

An edit is

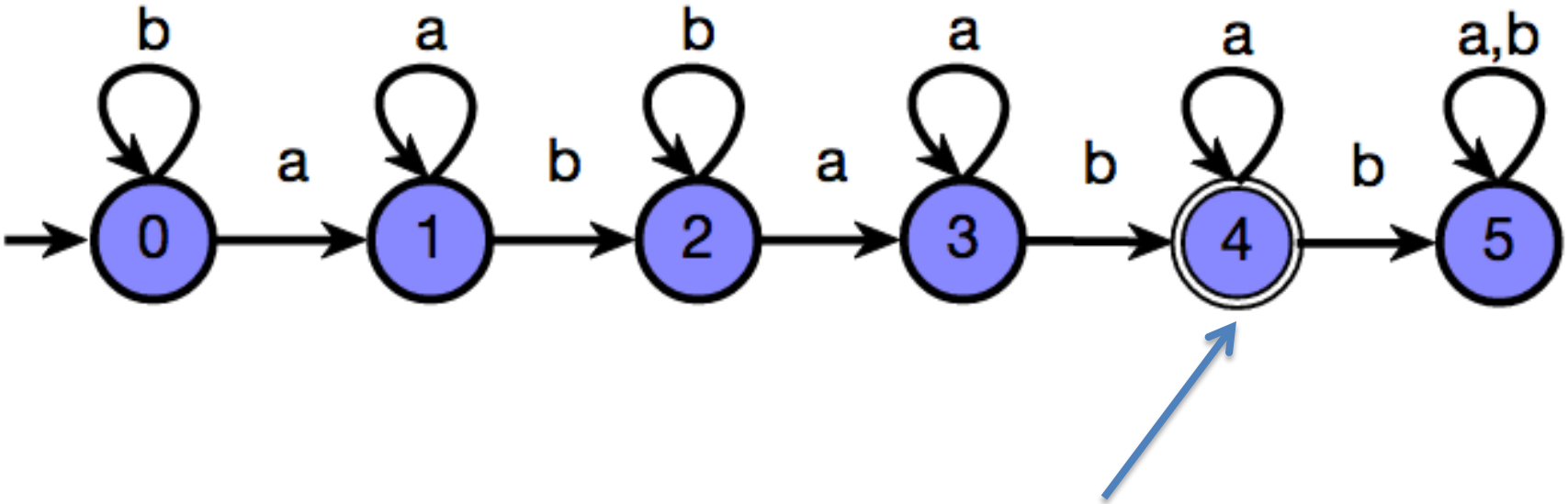
- Make a state (non)final
- Add a new state
- Redirect a transition

DFA Edit Difference: How to compute it?

We try every possible edit and check for equivalence

- Speed up equivalence by using test set
- The problem of finding DFAED is in NP (is it NP-hard?)

Solution Semantic Mistake



The student didn't see that the 'a' loop might not be traversed

INTUITION: find on how many strings the student is wrong

Approximate Density

S = correct solution A = student attempt

Compute Symmetric Difference: $D = S \setminus A \cup A \setminus S$

- Measure relative size of D with respect to S

$$Size(D,S) = \lim_{n \to \infty} D^n / S^n$$

- Size(D,S) is not computable in general (the limit oscillates)
- Approximate the limit to finite n

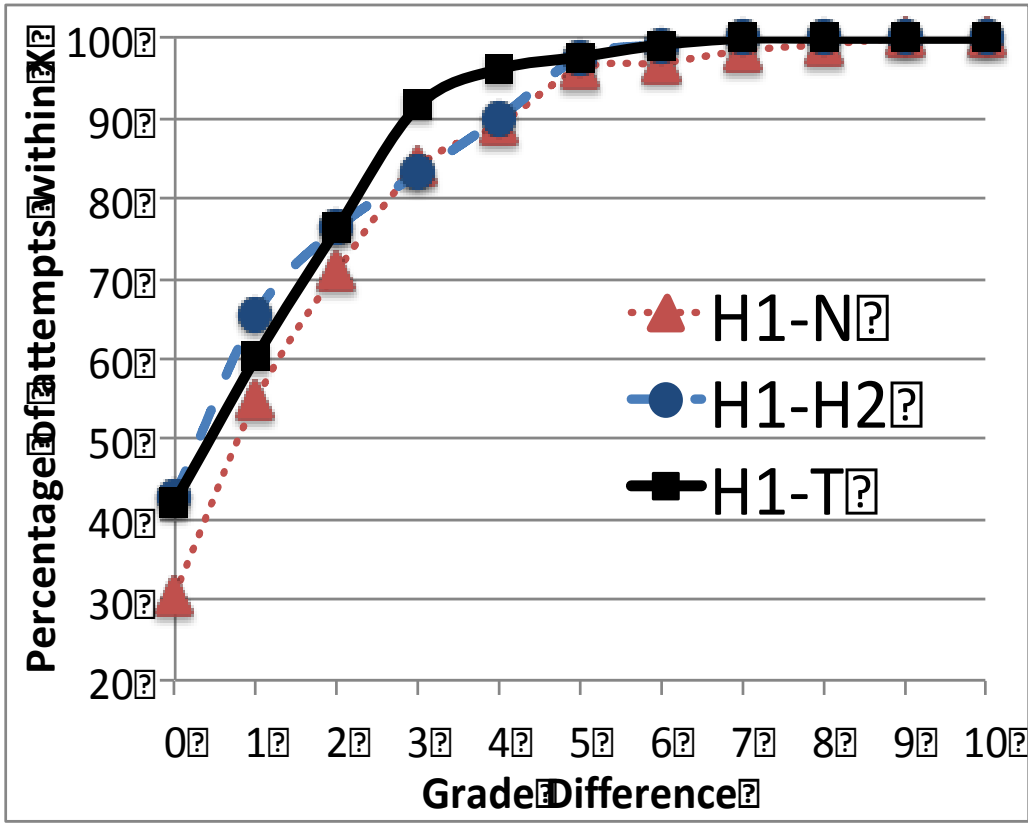
Evaluation 1/2

H1, H2 = human graders

N =naïve grader

T = tool

Tool is closer to humans than humans are to each other



Evaluation 2/2

H1, H2 = human graders

N =naïve grader

T = tool

Tool and humans look indistinguishable

Problem	Attempts		Average			Standard Deviation			Pearson Correlation		
	Tot.	Dis.	H_1-H_2	H_1-T	H_1-N	H_1-H_2	H_1-T	H_1-N	H_1-H_2	H_1-T	H_1-N
$L_1 = \{s \mid s$ starts with a and has odd number of ab substrings$\}$	131	108	0.99	0.54	0.22	2.06	1.99	2.62	0.87	0.83	0.65
$L_2 = \{s \mid s$ has more than 2 a's or more than 2 b's$\}$	110	100	-0.66	0.85	0.26	1.80	2.44	2.71	0.90	0.80	0.75
$L_3 = \{s \mid s$ where all odd positions contain the symbol $a\}$	96	75	-0.52	0.86	-1.38	1.61	2.67	3.84	0.90	0.74	0.31
$L_4 = \{s \mid s$ begins with ab and $\|s\|$ is not divisible by 3$\}$	92	68	0.40	1.32	0.36	1.68	2.78	2.48	0.81	0.71	0.61
$L_5 = \{s \mid s$ contains the substring ab exactly twice$\}$	52	46	0.02	0.19	0.29	2.01	1.88	3.23	0.71	0.79	0.49
$L_6 = \{s \mid s$ contains the substring aa and ends with $ab\}$	38	31	-0.50	-1.34	-1.5	2.42	2.90	3.70	0.76	0.63	0.34

Pro's and Cons

Pros:

- On disagreeing cases, human grader often realized that his grade was inaccurate
- Identical solutions receive same grades and correct attempts awarded max score (unlike human)

Cons:

- For now limited to small DFAs
- When two types of mistakes happen at same time, the tool can't figure it out

Conclusions

AutomataTutor: a tool that grades DFA constructions fully automatically

Few new automata problems:

- How to compute DFA edit difference?
- How to synthesize Mosel formulas in a better way?
- How to compute language sizes in a way that is always defined and accurate?

Questions?

Thank you!