COMP 481

John Greiner

DFA Minimization

Reading: [Linz3] Section 2.4

1

Introduction

Look at one important algorithm on DFAs:

DFA Minimization

- Will look at other algorithms later in the course.
- This one is interesting now because it provides another tool for showing languages regular or not.

2

DFA Minimization: The Idea

Background:

- Have seen that single RL has many DFAs.
- Have seen that can simplify DFAs, REs, etc., by equivalences.

Questions:

- Is there a unique simplest DFA for a RL?
- If so, can we construct it?

3

DFA Minimization: The Idea

Since it's today's topic…

Yes, there is a unique minimal DFA & can construct it.

"Minimal"?

Minimal number of states.

"Unique"?

Unique up to renaming of states.

I.e., has same shape. Isomorphic.

Will show algorithm & detailed example.

Will outline why algorithm is correct.

4

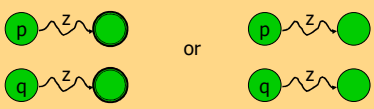
DFA Minimization: Algorithm Idea

Equate & collapse states having same behavior.

Build equivalence relation on states:

$p \equiv q \quad \leftrightarrow \quad (\forall z \in \Sigma^*,\ \hat{\delta}(p,z) \in F \ \leftrightarrow\ \hat{\delta}(q,z) \in F)$

I.e., iff for every string z, one of the following is true:



5

DFA Minimization: Algorithm

Build table to compare each unordered pair of distinct states p,q.

Each table entry has

- a "mark" as to whether p & q are known to be not equivalent, and
- a list of entries, recording dependences: "If this entry is later marked, also mark these."

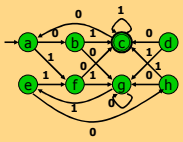
6

DFA Minimization: Algorithm

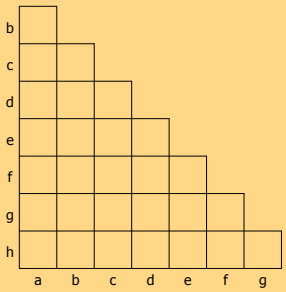
1. Initialize all entries as unmarked & with no dependences.
2. Mark all pairs of a final & nonfinal state.
3. For each unmarked pair p,q & input symbol a:
   1. Let r=δ(p,a), s=δ(q,a).
   2. If (r,s) unmarked, add (p,q) to (r,s)'s dependences,
   3. Otherwise mark (p,q), and recursively mark all dependences of newly-marked entries.
4. Coalesce unmarked pairs of states.
5. Delete inaccessible states.

7

DFA Minimization: Example

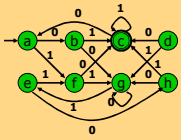


1. Initialize table entries: Unmarked, empty list

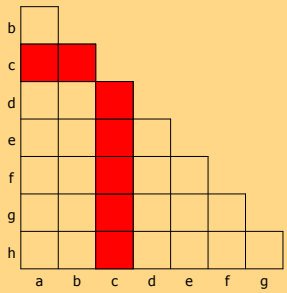


8

DFA Minimization: Example

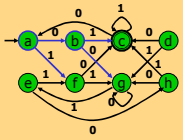


2. Mark pairs of final & nonfinal states

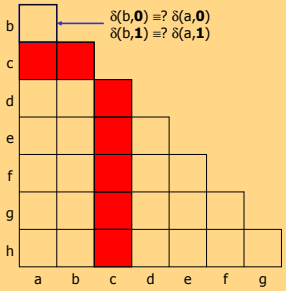


9

DFA Minimization: Example

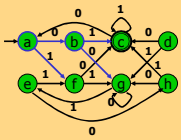


3. For each unmarked pair & symbol, …

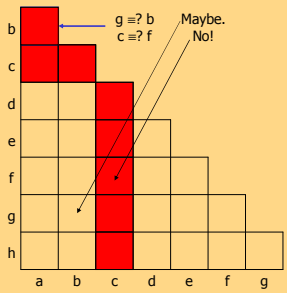


10

DFA Minimization: Example

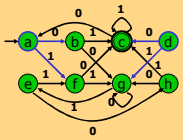


3. For each unmarked pair & symbol, …

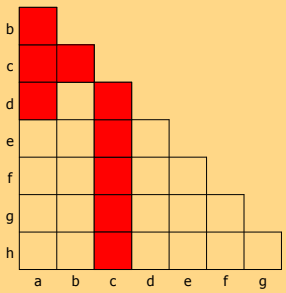


11

DFA Minimization: Example

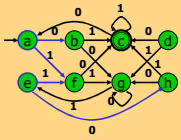


3. For each unmarked pair & symbol, …

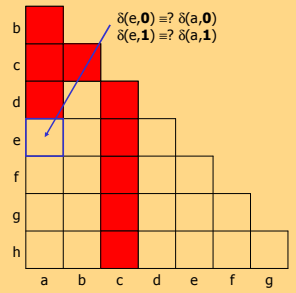


12

DFA Minimization: Example

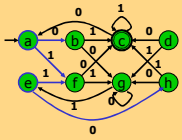


3. For each unmarked pair & symbol, …

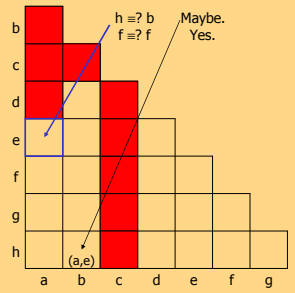


13

DFA Minimization: Example

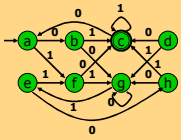


3. For each unmarked pair & symbol, …

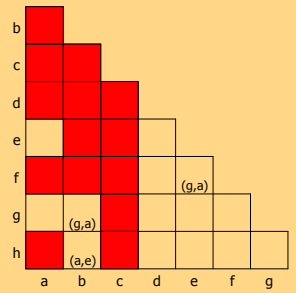


14

DFA Minimization: Example

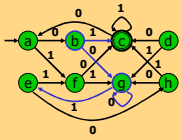


3. For each unmarked pair & symbol, …

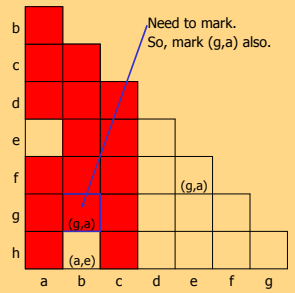


15

DFA Minimization: Example

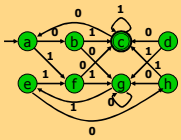


3. For each unmarked pair & symbol, …

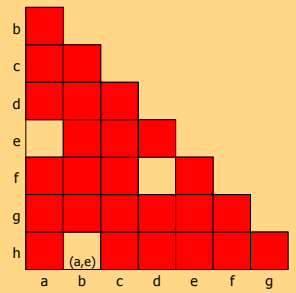


16

DFA Minimization: Example

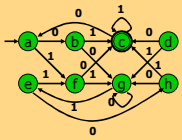


3. For each unmarked pair & symbol, …



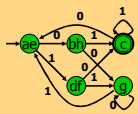
17

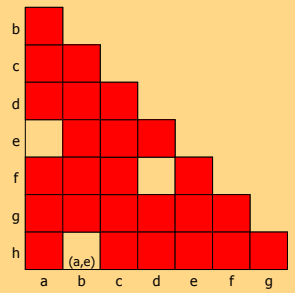
DFA Minimization: Example



4. Coalesce unmarked pairs of states.

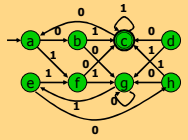
a ≡ e
b ≡ h
d ≡ f





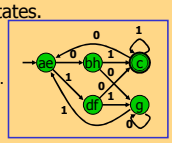
18

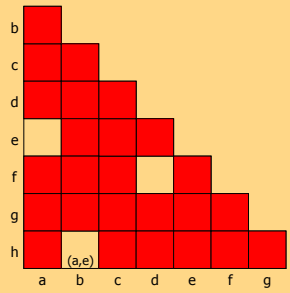
DFA Minimization: Example



5. Delete unreachable states.

None.





19

DFA Minimization: Notes

Order of selecting state pairs was arbitrary.
- All orders give same ultimate result.
- But, may record more or fewer dependences.
- Choosing states by working backwards from known non-equivalent states produces fewest dependences.

This algorithm: Huffman (1954), Moore (1956).

$O(n^2)$ time.
- Constant work per entry: initial mark test & possibly later chasing of its dependences.
- More efficient algorithms exist, e.g., Hopcroft (1971).

Can delete unreachable states initially, instead.

20

DFA Minimization: Correctness

Why is new DFA no larger than old DFA?
- Only removes states, never introduces new states.
- Obvious.

Why is new DFA equivalent to old DFA?
- Only identify states that provably have same behavior.
- Could prove $x \in L(M) \leftrightarrow x \in L(M')$ by inductions on derivations.

Why is "minimal" DFA unique (up to isomorphism)?
- Depends on the uniqueness of minimal equivalence classes of strings in the language.
- Details hard to show...

21

DFA Minimization: Correctness

Myhill-Nerode Theorem (1958):
- A way to show algorithm's result is indeed minimal.
- More generally, a useful tool for showing languages regular or non-regular.

Will only explain the theorem, not the proof.
- Proof is fairly long & not very enlightening.

22

Myhill-Nerode Theorem: 1

The following three statements are equivalent:

1. $L \subseteq \Sigma^*$ is accepted by some DFA M.

   I.e., L is regular.

23

Myhill-Nerode Theorem: 2

The following three statements are equivalent:

2. L is the union of some equivalence classes of some right invariant equivalence relation R of finite index.

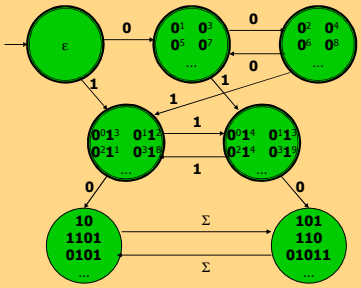
Def'n: Relation R is right invariant $\leftrightarrow$ $x \mathrel{R} y \rightarrow xz \mathrel{R} yz$.
Def'n: Relation R's index is its number of equivalence classes.

Intuition:
- Equivalence classes represent final states of **some** DFA.
- Index = number of minimal DFA states.
- Must be right invariant to mimic $\hat{\delta}$.

24

Myhill-Nerode Theorem: 2 Example

Equivalence classes of one particular R over $\{\mathbf{0},\mathbf{1}\}^*$:



L = $\mathbf{0}^*\mathbf{1}^*$ = Union of some of these.
(Chose an appropriate R.)

Intuition: Corresponds to some DFA for L.

Myhill-Nerode Theorem: 3

The following three statements are equivalent:

3. The following equivalence relation R_L is of finite index:
   $x\ R_L\ y \leftrightarrow \forall z \in \Sigma^*, xz \in L \leftrightarrow yz \in L$

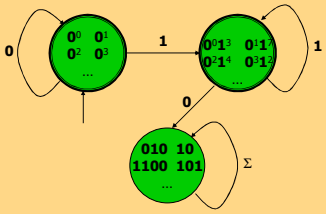
Intuition:
- Similar to #2, but working backwards from final states.
- x,y related by $R_L \leftrightarrow (\hat{\delta}(q_x,z)$ final $\leftrightarrow \hat{\delta}(q_y,z)$ final)

Not immediately clear how R_L and R related.
Not immediately clear if R_L is right invariant, since statement doesn't say anything about $xz\ R_L\ yz$.

Myhill-Nerode Theorem: 3 Example

L = $\mathbf{0}^*\mathbf{1}^*$

Implies R_L has these equivalence classes:



Intuition: Corresponds to minimal DFA for L.

Myhill-Nerode Theorem: Proof

Theorem's proof results in minimization algorithm:
$1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ implies that, given a DFA or RL, can obtain a new DFA that models R_L's equivalence classes.

Minimality depends on the uniqueness of minimal equivalence classes of strings in language (not of states in the DFA).

Can use theorem to show language to be RL or non-RL:
I.e., can show that either property 2 or 3 holds or not.

Myhill-Nerode: Example Use

L = $\{\mathbf{0}^i\ \mathbf{1}^i \mid i \geq 0\}$

What is R_L?
Reminder: $x\ R_L\ y \leftrightarrow \forall z \in \Sigma^*, xz \in L \leftrightarrow yz \in L$

For any prefix x, consider the set of suffixes z such that $xz \in L$.
E.g., $x=\mathbf{0}^5 \rightarrow z \in \mathbf{0}^i\ \mathbf{1}^{i+5}$
Each x has its own unique set. Each equivalence class is a singleton.

How many equivalence classes?
Same as number of prefixes x.
Countably infinite.

Thus, R_L is not of finite index. L is not regular.

NFA Minimization

This algorithm doesn't find a unique minimal NFA.

?

Is there a (not necessarily unique) minimal NFA?

?

Of course.

NFA Minimization

In general, minimal NFA not unique!

Example NFAs for $\mathbf{0}^+$:

Both minimal, but not isomorphic.

31