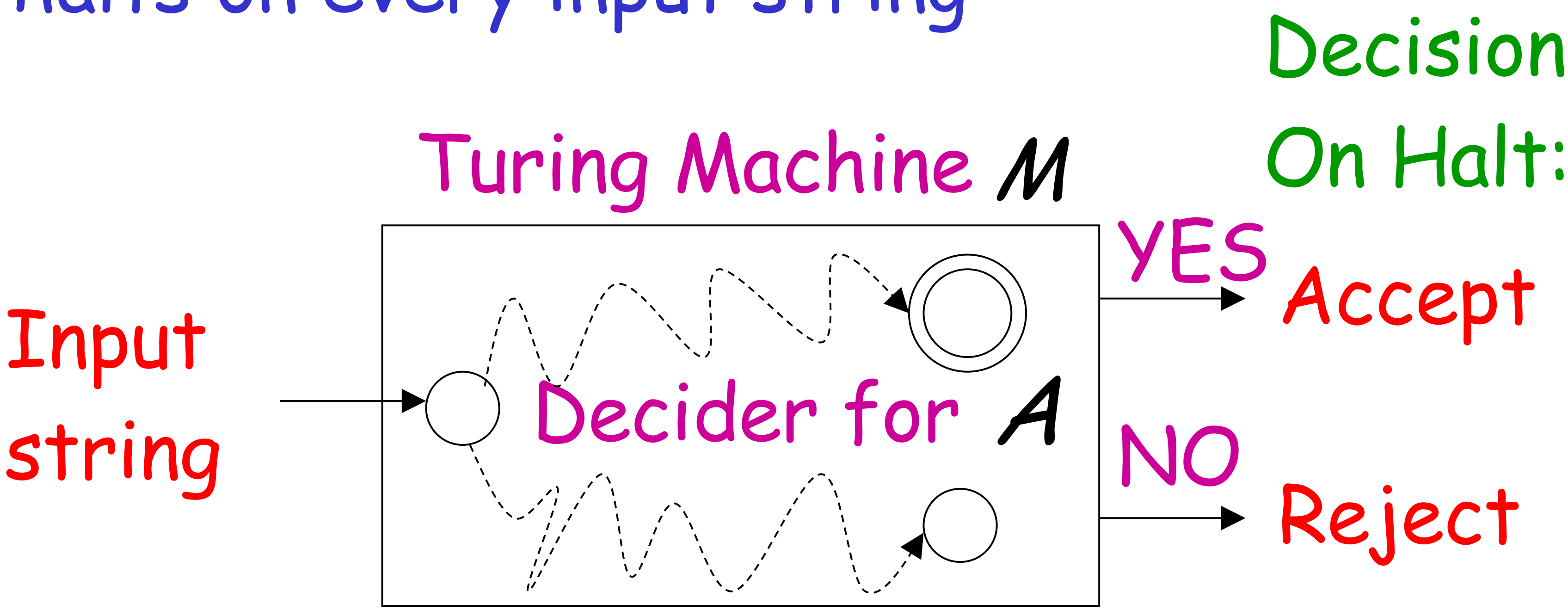
Busch Complexity Lectures:

Decidability and Reductions

Decidable Languages

Recall that:

A language A is decidable,
if there is a Turing machine M (decider)
that accepts the language A and
halts on every input string



A computational problem is decidable
if the corresponding language is decidable

(If so, we also say that the problem is solvable)

Problem: Does DFA M accept string w ?

Corresponding Language: (Decidable)

$A_{DFA} =$

$\{\langle M, w \rangle : M \text{ is a DFA that accepts string } w\}$

Decider for A_{DFA} :

On input string $\langle M,w \rangle$:

Run DFA M on input string w

If M accepts w

<u>Then</u> accept $\langle M,w \rangle$ (and halt)

<u>Else</u> reject $\langle M,w \rangle$ (and halt)

Given Deterministic Finite State Automata (DFA) M:

Problem: Does DFA M accept the empty language $L(M) = \emptyset$?

Corresponding Language: (Decidable)

$EMPTY_{DFA} =$

$\{\langle M \rangle : M \text{ is a DFA that accepts empty language } \emptyset\}$

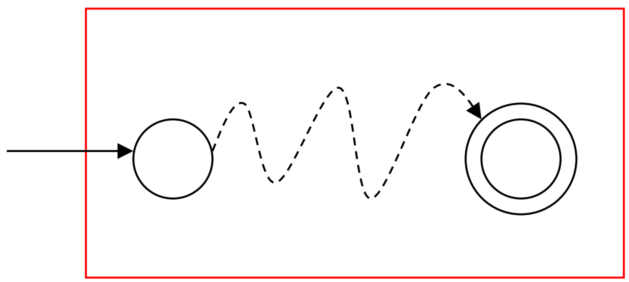
Description of DFA M as a string
(For example, we can represent M as a binary string, as we did for Turing machines)

Decider for $EMPTY_{DFA}$:

On input $\langle M \rangle$:

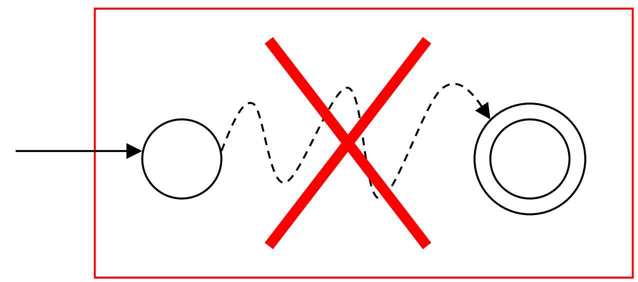
Determine whether there is a path from the initial state to any accepting state

DFA M



$L(M) \neq \emptyset$

DFA M



$L(M) = \emptyset$

Decision: Reject $\langle M \rangle$ Accept $\langle M \rangle$

Problem: Does DFA M accept a finite language?

Corresponding Language: (Decidable)

$FINITE_{DFA} =$

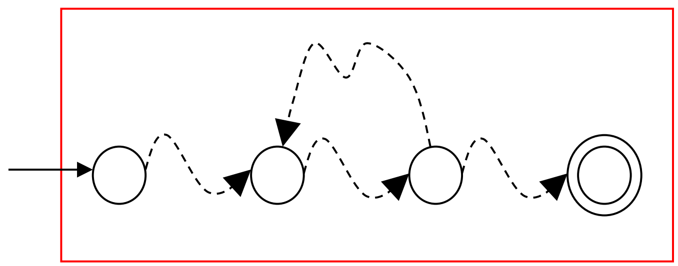
$\{\langle M \rangle : M \text{ is a DFA that accepts a finite language}\}$

Decider for $FINITE_{DFA}$:

On input $\langle M \rangle$:

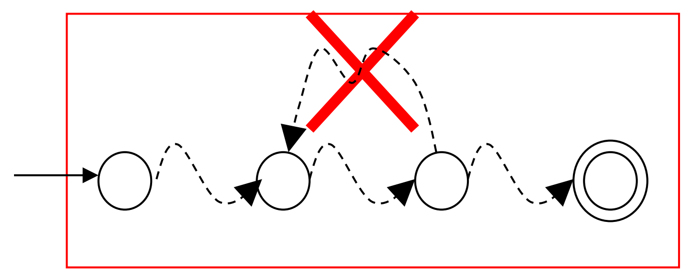
Check if there is a walk with a cycle from the initial state to an accepting state

DFA M



infinite

DFA M



finite

Decision: Reject $\langle M \rangle$ (NO) Accept $\langle M \rangle$ (YES)

Recursive Reductions Between Problems

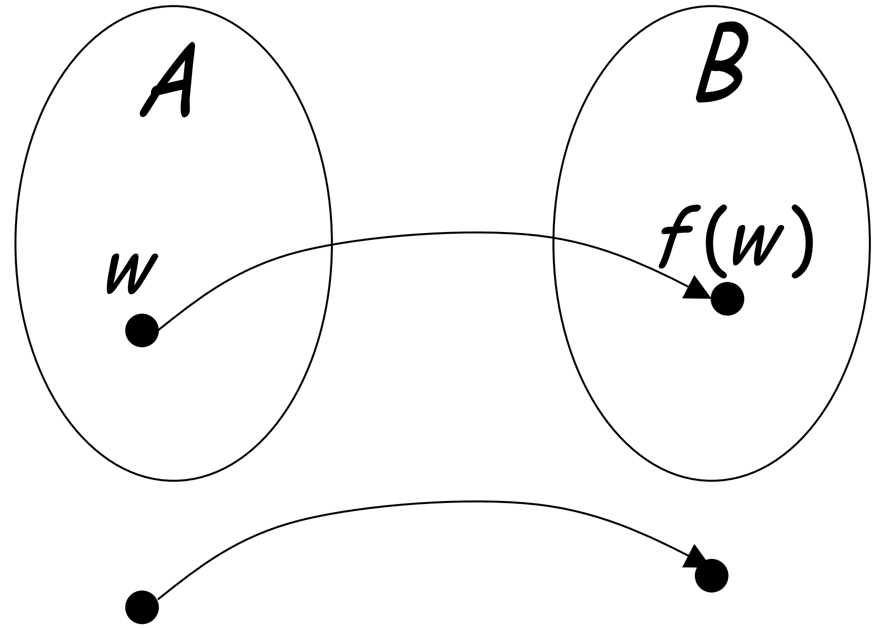
Problem X is recursively reduced
to problem Y

⇓

If we can solve problem Y
then we can solve problem X

Definition:

Language A
is reduced to
language B



There is a computable function f (*reduction*) such that:

$$w \in A \iff f(w) \in B$$

Recall:

Computable function f :

There is a deterministic Turing machine M which for any string w computes $f(w)$

Theorem:

If: a: Language A is reduced to B

b: Language B is decidable

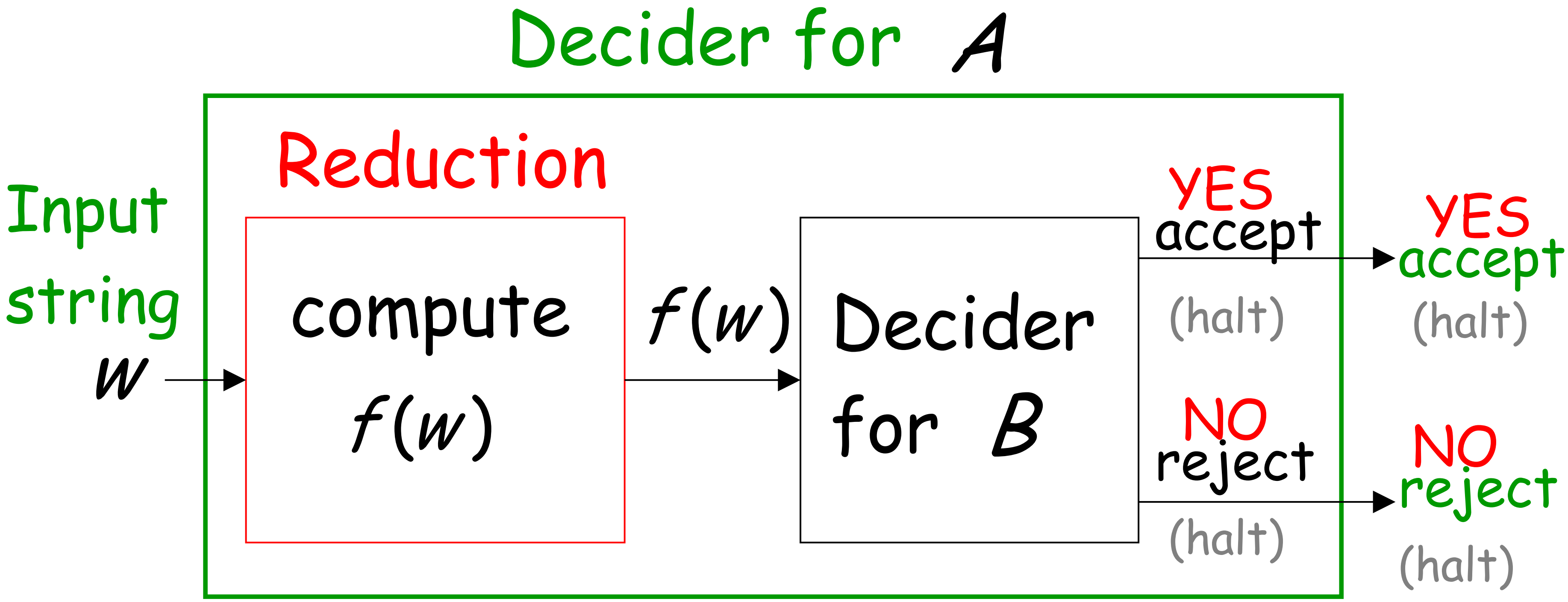
Then: A is decidable

Proof:

Basic idea:

Build the decider for A

using the decider for B



$$w \in A \Leftrightarrow f(w) \in B$$

END OF PROOF

Example:

$$EQUAL_{DFA} = \{\langle M_1, M_2 \rangle : M_1 \text{ and } M_2 \text{ are DFAs that accept the same languages}\}$$

is reduced to:

$$EMPTY_{DFA} = \{\langle M \rangle : M \text{ is a DFA that accepts the empty language } \emptyset\}$$

We only need to construct:

$\langle M_1, M_2 \rangle \longrightarrow$ Turing Machine for reduction f $\longrightarrow f(\langle M_1, M_2 \rangle)$

$= \langle M \rangle$ DFA

$$\langle M_1, M_2 \rangle \in EQUAL_{DFA} \quad \Leftrightarrow \quad \langle M \rangle \in EMPTY_{DFA}$$

Let L_1 be the language of DFA M_1

Let L_2 be the language of DFA M_2

$\langle M_1, M_2 \rangle \longrightarrow$ Turing Machine for reduction f $\longrightarrow f(\langle M_1, M_2 \rangle) = \langle M \rangle$ DFA

construct DFA M

by combining M_1 and M_2 so that:

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

$$\Updownarrow$$

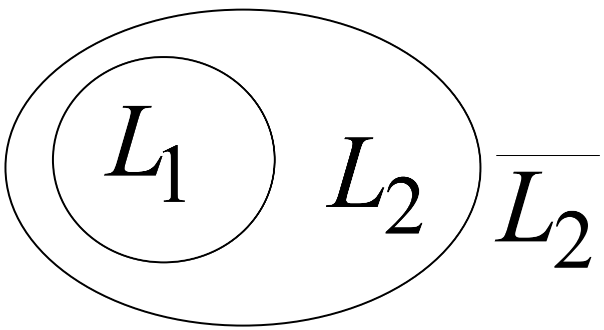
$$L_1 = L_2 \quad \Leftrightarrow \quad L(M) = \emptyset$$

$$\Updownarrow$$

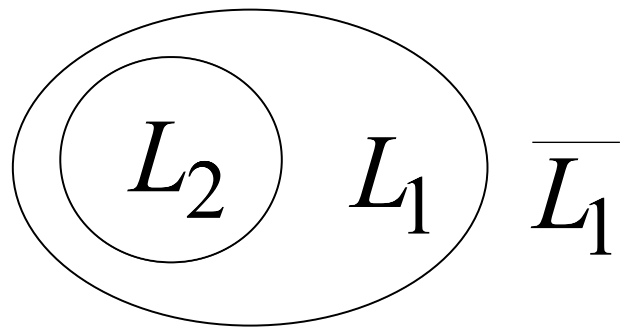
$$\langle M_1, M_2 \rangle \in EQUAL_{DFA} \quad \Leftrightarrow \quad \langle M \rangle \in EMPTY_{DFA}$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

$L_1 \cap \overline{L_2} = \varnothing$ and $\overline{L_1} \cap L_2 = \varnothing$



$L_1 \subseteq L_2$



$L_2 \subseteq L_1$

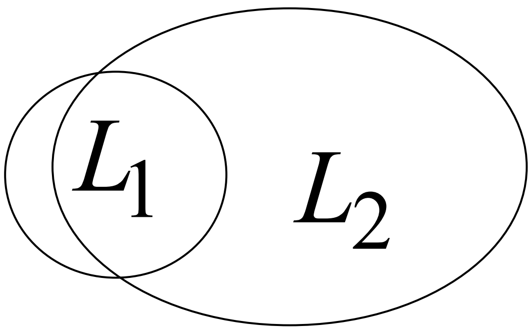
$$L_1 = L_2$$

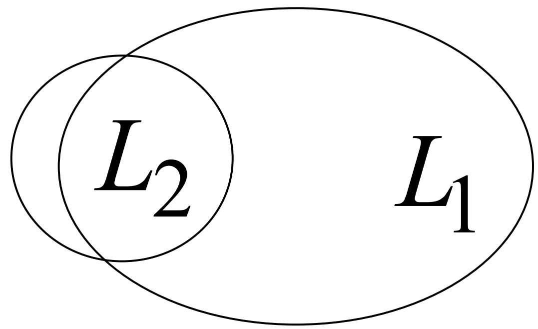
$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$$L_1 \cap \overline{L_2} \neq \emptyset$$

or

$$\overline{L_1} \cap L_2 \neq \emptyset$$





$$L_1 \not\subset L_2$$

$$L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

Therefore, we only need
to determine whether

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

which is a solvable problem for DFAs:

$EMPTY_{DFA}$

Decider for $EQUAL_{DFA}$