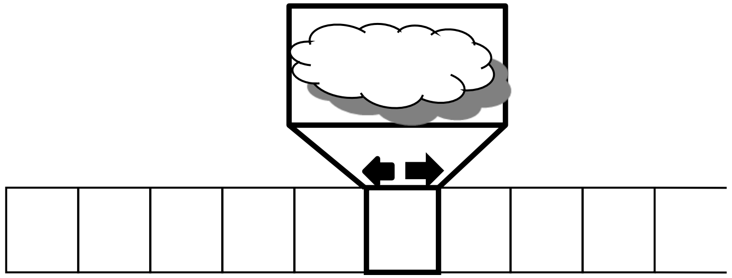
# CS3102 Theory of Computation

# Decidable

$w \rightarrow$
Input



Accept & halt | Reject & halt | Never runs forever

A language is decidable iff it is exactly the set of strings accepted by some always-halting TM.

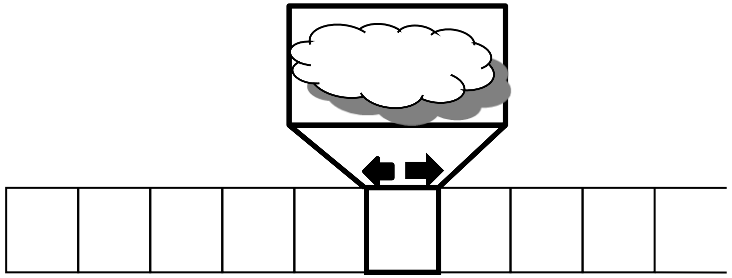
| $w \in \Sigma^*$ | a | b | aa | ab | ba | bb | aaa | aab | aba | abb | baa | bab | bba | bbb | aaaa | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M(w) \Rightarrow$ | ✓ | × | ✓ | × | × | × | ✓ | × | × | × | × | × | × | × | ✓ | … |
| $L(M) =$ | $\{\ a,$ | | $aa,$ | | | | $aaa,$ | | | | | | | | $aaaa$ | $\ldots\}$ |

$M$ must always halt on every input.

# Recognizable

$w \rightarrow$

Input



$\Rightarrow$ ✓ Accept & halt | × Reject & halt $\equiv$ $\infty$ Run forever

A language is Turing-recognizable iff it is exactly the set of strings accepted by some Turing machine.

| $w \in \Sigma^*$ | $a$ | $b$ | $aa$ | $ab$ | $ba$ | $bb$ | $aaa$ | $aab$ | $aba$ | $abb$ | $baa$ | $bab$ | $bba$ | $bbb$ | $aaaa$ | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M(w) \Rightarrow$ | ✓ | × | ✓ | $\infty$ | × | $\infty$ | ✓ | $\infty$ | $\infty$ | × | × | × | $\infty$ | × | ✓ | … |
| $L(M) =$ | $\{\ a,$ | | $aa,$ | | | | $aaa,$ | | | | | | | | $aaaa$ | $\ldots\}$ |

$M$ can run forever on an input, which is implicitly a reject (since it is not an accept).

# Computability

- Generally: Computable = Decideable

# An Undecidable Problem/Language

- Acceptance Problem
- Given a Turing Machine description $M$ (e.g. a program, states+transitions, etc.) and a string $w$, does $M$ accept the input $w$?
- $A_{TM}(M, w) = \{\langle M, w \rangle | M(w) \text{ accepts}\}$

# Acceptance  problem is undecidable

- Assume toward reaching a contradiction that $M_{acc}$ is a Decider for $A_{TM}$.
- Consider a new Turing Machine $M'$ which receives as input a turing machine description $M$.
- If $\langle M, M \rangle \in A_{TM}$ then $M'$ will reject, else $M'$ accepts

# Pseudocode for $M'$

```
public static boolean mPrime(String m){
        return !accept(m,m);
}
```

What does **mPrime(source(mPrime))** return?

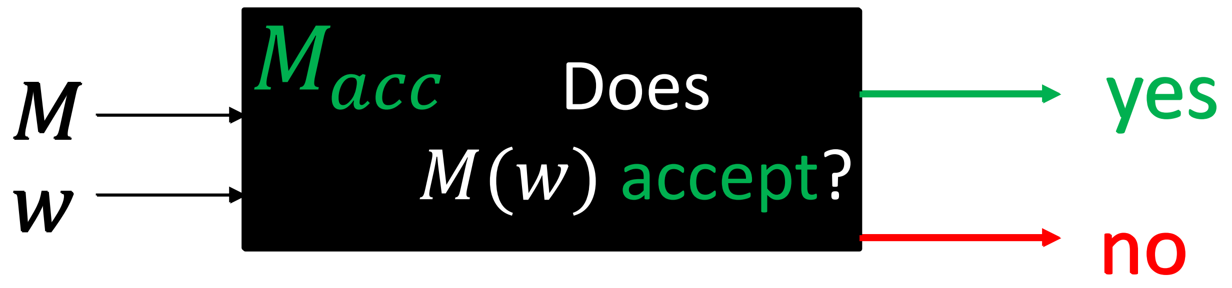
# Acceptance problem is undecidable

- Assume toward reaching a contradiction that $M_{acc}$ is a Decider for $A_{TM}$.
- Consider a new Turing Machine $M'$ which receives as input a turing machine description $M$.
- If $\langle M, M \rangle \in A_{TM}$ then $M'$ will reject, else $M'$ accepts
- Consider $M'(M')$
  - If it accepts, then by definition $M'(M')$ will reject
  - If it rejects, then by definition $M'(M')$ with accept
  - Contradiction!
- Conclusion: $M_{acc}$ cannot exist

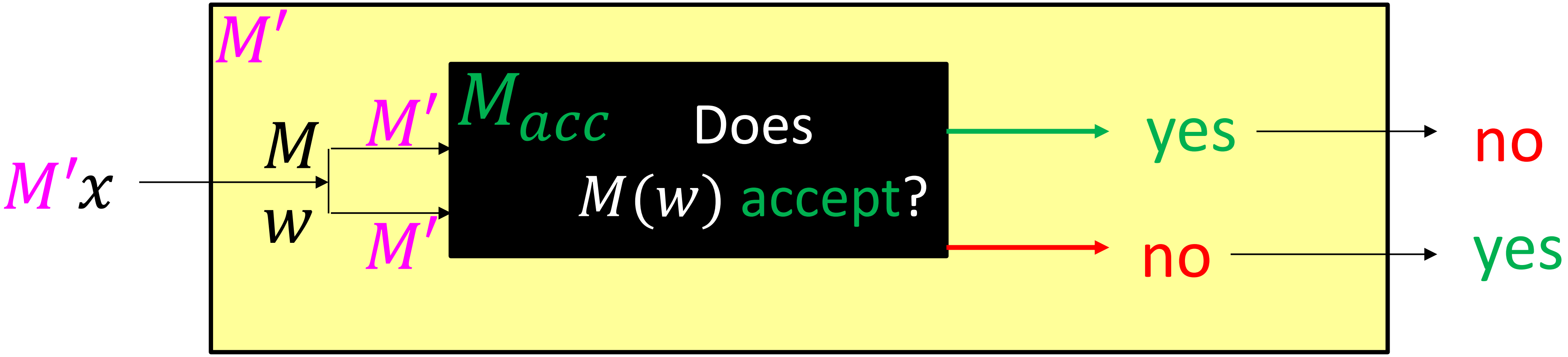
**Theorem**: the acceptance problem ($A_{TM}$) is not decidable

**Proof**: Assume some decider $M_{acc}$ solves $A_{TM}$

always stops with the correct answer for any $M$ and $w$




Use $M_{acc}$, construct a TM $M'$:




$M'(M')$ accepts → $M'(M')$ does not accept
$M'(M')$ does not accept → $M'(M')$ accepts

Contradiction!

→ $M_{acc}$ cannot exist! (at least as an algorithm / program / TM)

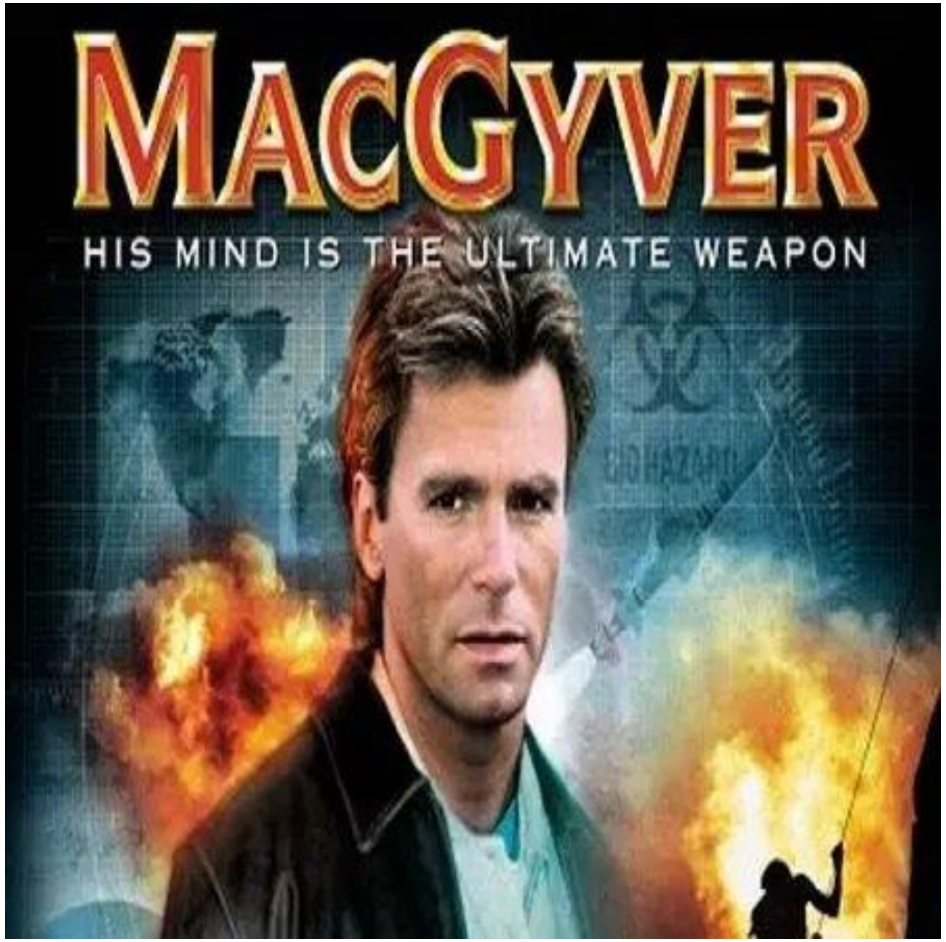
# Proving Other language Undecidable

- Option 1: Closure properties
  - Decidable languages are closed under:
    - Union
    - Intersection
    - Complement
    - Reversal
    - Pretty much everything
- Option 2: Reduction
  - Convert some problem into a known undecidable one to show it’s undecidable.

# Proof by Reduction

Shows how two different problems relate to each other

MOVIE TIME!



# Reduction Proofs

Opening a door

$A$ reduces to $B$

Lighting a fire

Alcohol, wood, matches

$Y$ can be used to make $X$

that can solve **B**

that can solve **A**

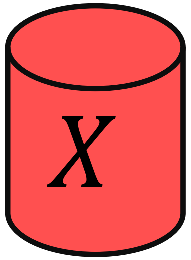
Keg cannon battering ram

**$A$ is not a harder problem than $B$**

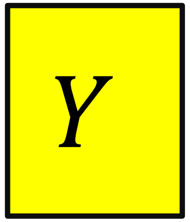
$$A \leq B$$

The name “reduces” is confusing: it is in the *opposite* direction of the making

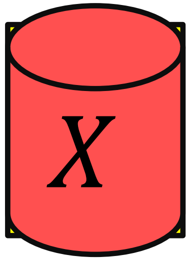
# Proof of Impossibility by Reduction




1. $X$ isn't possible
(e.g., $X$ = some way to open the door)



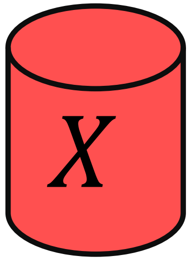

2. Assume $Y$ is possible
($Y$ = some way to light a fire)



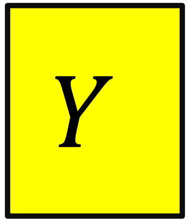

3. Show how to use $Y$ to perform $X$.

4. $X$ isn't possible, but $Y$ could be used to perform $X$
conclusion: $Y$ must not be possible either

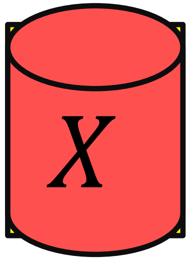
# Proof of Impossibility by Reduction




1. $X$ does not exist.
(e.g., $X$ = some TM that decides $A_{TM}$)




2. Assume $Y$ exists.
($Y$ = some TM that decides $B$)




3. Show how to use $Y$ to perform $X$.

4. $X$ doesn't exist, but $Y$ could be used to make $X$

conclusion: $Y$ must not exist either

# Converse?

**$A$ reduces to $B$**

$Y$

that can solve **B**
**(Lighting a fire)**

can be used to make

$X$

that can solve **A**
**(opening a door)**

**$A$ is not harder to solve than $B$**

$$A \leq B$$

Does this mean $B$ is equally as hard as $A$? $\quad A = B$

**No!**
Solving $Y$ is only one way to solve $X$
There may be an easier way

# Common Reduction Traps

- Be careful: the direction matters a great deal
  - Using a solver for $B$ to solve $A$ shows $A$ is not harder than $B$

$\underline{\quad A \quad}$ Reduces to $\underline{\quad B \quad}$

- The transformation must use only things you ***can do***:
  - Otherwise it may be that $B$ exists, but some other step doesn't!

# What “Can Do” Means

- Tools used in a reduction are limited by what you are proving
- Undecidability:
  - You are proving something about all TMs:
  - The transformation “can do” things a terminating TM “can do”

Spoiler alert!

- Complexity:
  - You are proving something about time required:
  - The time it takes to do the transformation is limited

# The Halting Language

$A_{TM} = \{ < M, w > \ | \ M$ is a TM description and $M$ accepts input $w$ $\}$
All machine description, input pairs in which the machine accepts that input

$HALT_{TM} = \{ < M, w > \ | \ M$ is a TM description and $M$ halts on input $w$ $\}$
All machine description, input pairs in which the machine halts on that input

Every $< M, w >$which halts at all belongs to $HALT_{TM}$
$< M, w >$ belongs to $A_{TM}$ if it both halts and accepts

# $HALT_{TM}$ is Undecidable

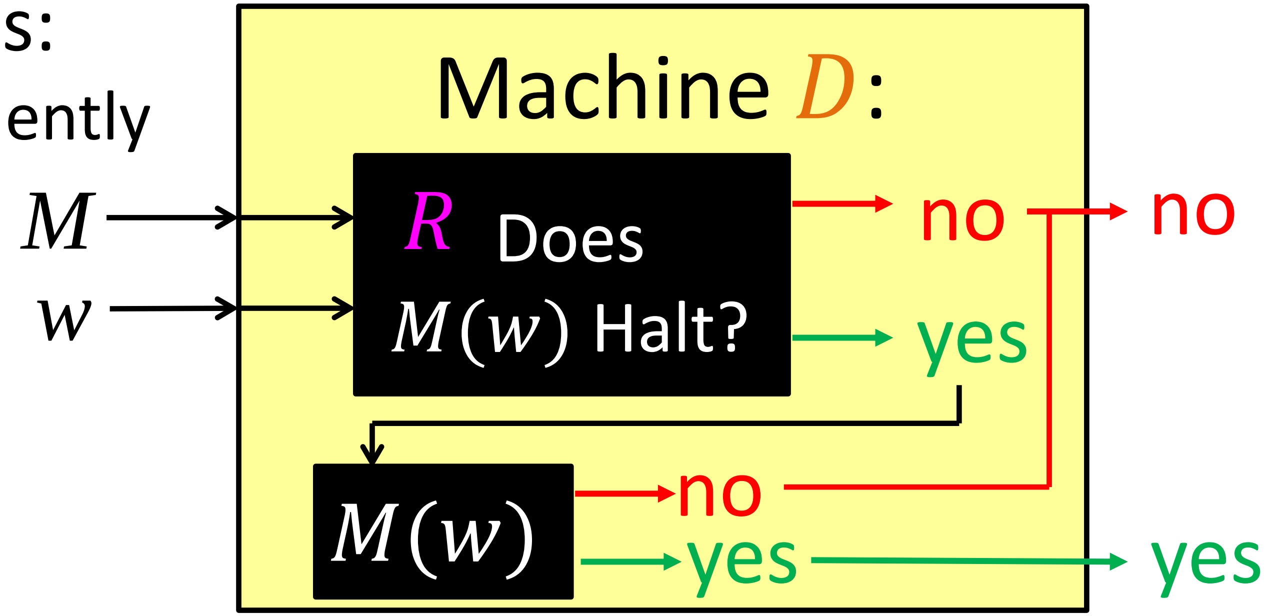
- $HALT_{TM} = \{< M, w > | M$ is a TM description and $M$ halts on input $w\}$
  - All machine description input pairs in which the machine halts on input
- To show $HALT_{TM}$ is undecidable show $A_{TM}$ isn't harder than $HALT_{TM}$
- Want to use a solver for $\underline{\quad HALT_{TM} \quad}$ to build a solver for $\underline{\quad A_{TM} \quad}$
- $\underline{\quad A_{TM} \quad}$ reduces to $\underline{\quad HALT_{TM} \quad}$

$A_{TM}$

$HALT_{TM}$

# Deciding $A_{TM}$ with $\mathrm{HALT}_{\mathrm{TM}}$

- Assume $HALT_{TM}$ is decidable.
- Then some TM $R$ can decide $HALT_{TM}$.
- We can use $R$ to build a machine $D$ that decides $A_{TM}$:
  - Call $R$ on $< M , w >$
  - If $R$ rejects, it means $M$ doesn't halt: **reject**
  - If $R$ accepts, it means $M$ halts:
    - Call $M$ on $w$, respond equivalently




**Any** TM that decides $HALT_{TM}$ could be used to build a TM that decides $A_{TM}$ (which is impossible) thus no TM exists that can decide $HALT_{TM}$

# Another example: $REG_{TM}$

- $REG_{TM} = \{M \mid L(M) \text{ is regular}\}$
- How do we show that $REG_{TM}$ is undecidable?
  – Reduce some language we already know is undecidable to $REG_{TM}$
  – Use $REG_{TM}$ to solve $HALT_{TM}$

# $REG_{TM} \geq HALT_{TM}$

- Given a potential instance of $HALT_{TM}$ (i.e. $M, w$), create a new turing machine $M'$ whose language is regular if and only if $M(w)$ halts
- If I knew whether or not $L(M')$ was regular, I knew whether or not $M(w)$ halted

# Psuedocode for $M'$

```
public static boolean mPrime(String x){
        if(x∈ a^n b^n){
                return true;
        }
        y = M(w);      If this line terminates
        return true;   We can only get to this line
}
```

If $M(w)$ halts then $L(M') = \Sigma^*$

If $M(w)$ runs forever then $L(M') = a^n b^n$

# How to solve $HALT_{TM}$ with $REG_{TM}$

```
public static boolean halt(M,w){
        mPrime = make_mPrime(M,w);
        return reg(mPrime);
}
```