The Big Picture

Announcements

- Problem Set 9 due right now. We'll release solutions right now.
  - ***Congratulations – you're done with CS103 problem sets!***
- ***Please evaluate this course on Axess!*** Your feedback really does make a difference.

Final Exam Logistics

- Our final exam is this Friday from 3:30PM – 6:30PM. Rooms are divvied up by last (family) name:
  - `Abb` – `Kan`: Go to Bishop Auditorium.
  - `Kar` – `Zuc`: Go to Cemex Auditorium.
- Exam is cumulative and all topics from the lectures and problem sets are fair game.
- The exam focus is roughly 50/50 between discrete math topics (PS1 – PS5) and computability/complexity topics (PS6 – PS9).
- As with the midterms, the exam is closed-book, closed-computer, and limited-note. You can bring a single, double-sided, 8.5” × 11” sheet of notes with you to the exam.

The Big Picture

The Big Picture

Cantor's Theorem: $|S| < |\wp(S)|$

Corollary: Unsolvable problems exist.

What problems can
be solved by computers?

First, we need to learn how to prove results with certainty.

Otherwise, how can we know for sure that we're right about anything?

We also need a way to precisely pin down key terms and definitions.

Let's add some logic into the mix.

Let's study a few common discrete structures.

That way, we know how to model connected structures and relationships.

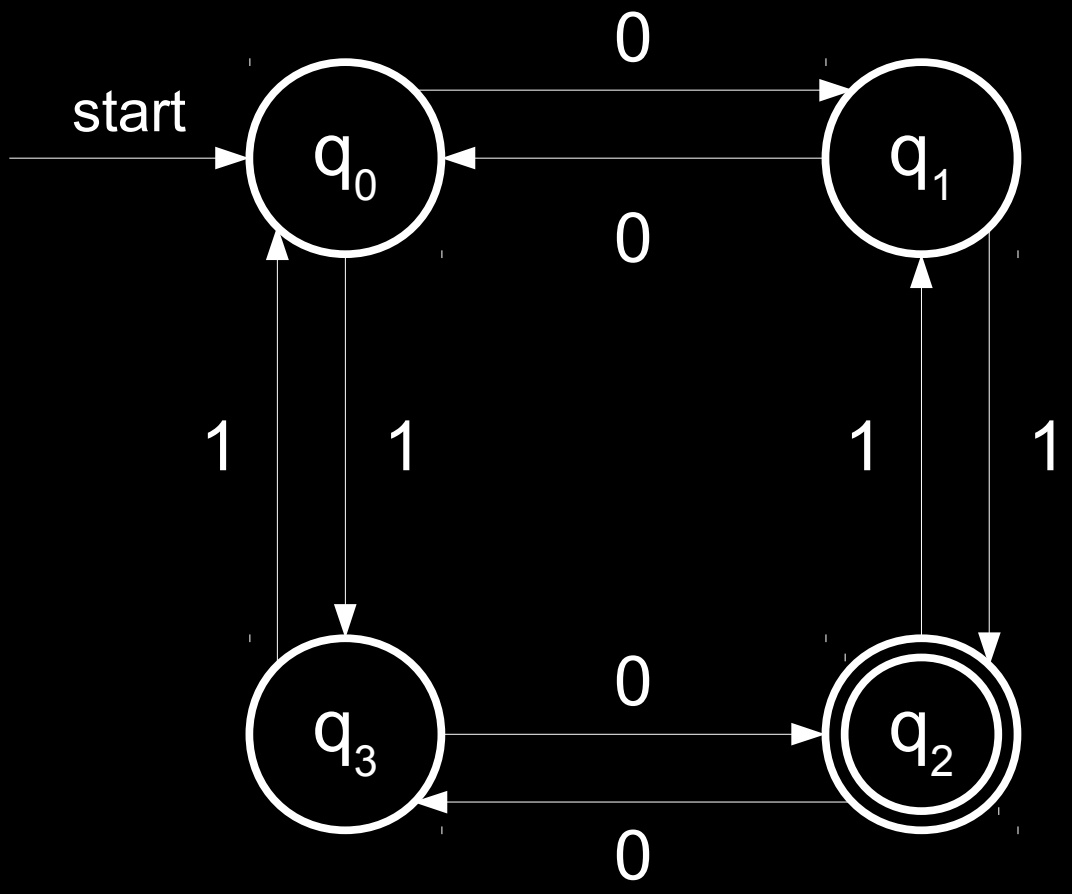
We also need to prove things about processes that proceed step-by-step.

So let's learn induction.

Okay! So now we're ready to go!

What problems are unsolvable?

Well, first we need a
definition of a computer!

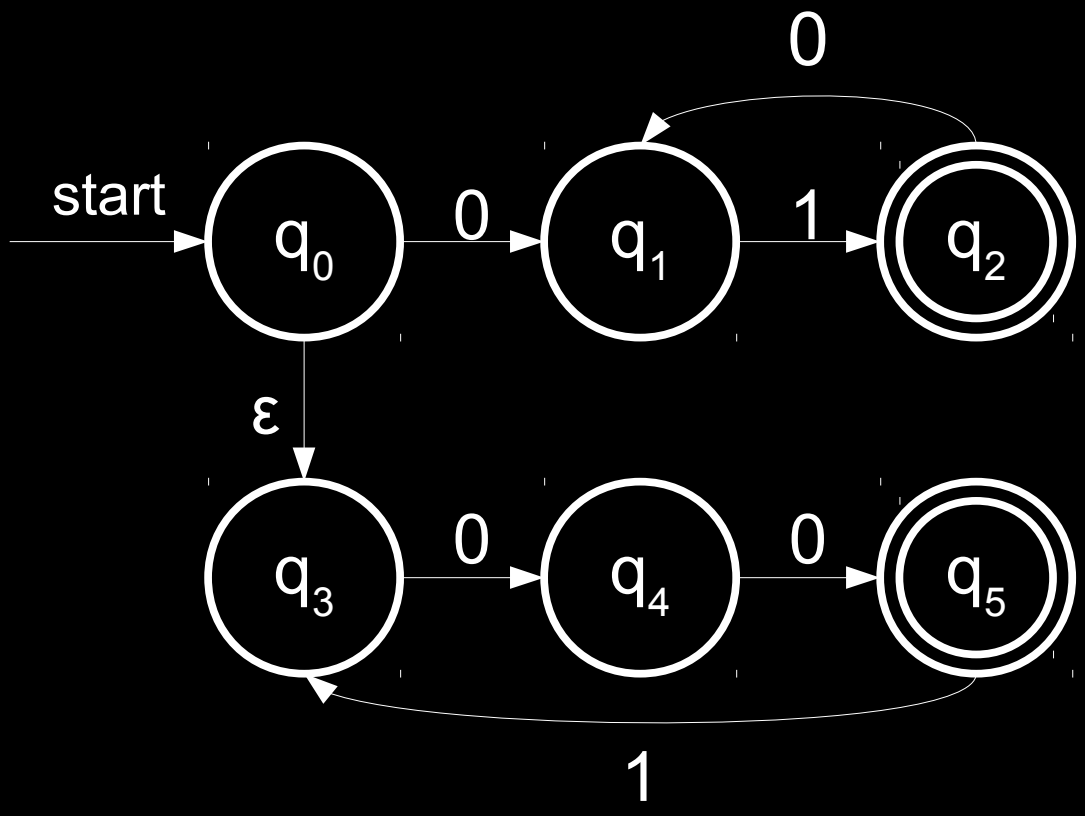

start
q_0
q_1
q_2
q_3
0
0
1
1
1
1
0
0

Cool! Now we have a model of a computer!

We're not quite sure what we can solve at this point, but that's okay for now.

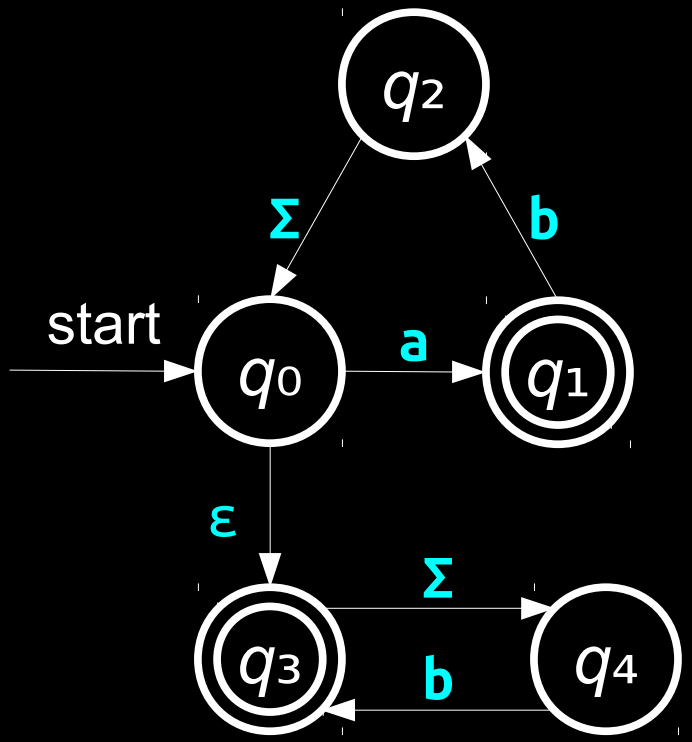
Let's call the languages we can capture this way the ***regular languages***.

I wonder what other
machines we can make?


0
start
q_0
0
q_1
1
q_2
ε
q_3
0
q_4
0
q_5
1

Wow! Those new machines are
way cooler than our old ones!

I wonder if they're more powerful?



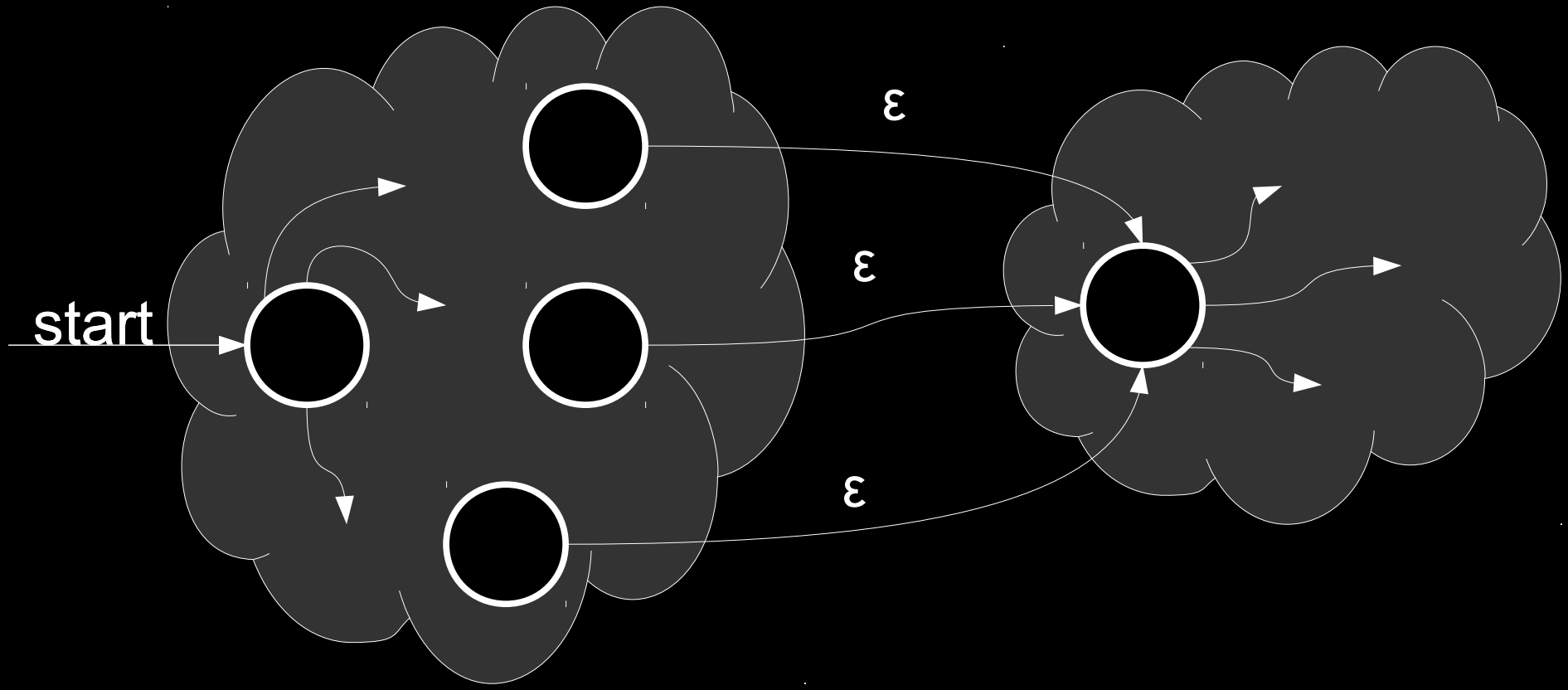
	a	b
*$\{q_0, q_3\}$	$\{q_1, q_4\}$	$\{q_4\}$
*$\{q_1, q_4\}$	$\emptyset$	$\{q_2, q_3\}$
$\{q_4\}$	$\emptyset$	$\{q_3\}$
*$\{q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$
*$\{q_3\}$	$\{q_4\}$	$\{q_4\}$
*$\{q_0, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_3, q_4\}$
*$\{q_3, q_4\}$	$\{q_4\}$	$\{q_3, q_4\}$
$\emptyset$	$\emptyset$	$\emptyset$

Wow! I guess not. That's surprising!

So now we have a new way of modeling computers with finite memory!

However, we have just seen that ***computability*** (what problems can you solve?) is not the same as ***complexity*** (how efficiently can you solve the problem?)

I wonder how we can combine
these machines together?


start
ε
ε
ε

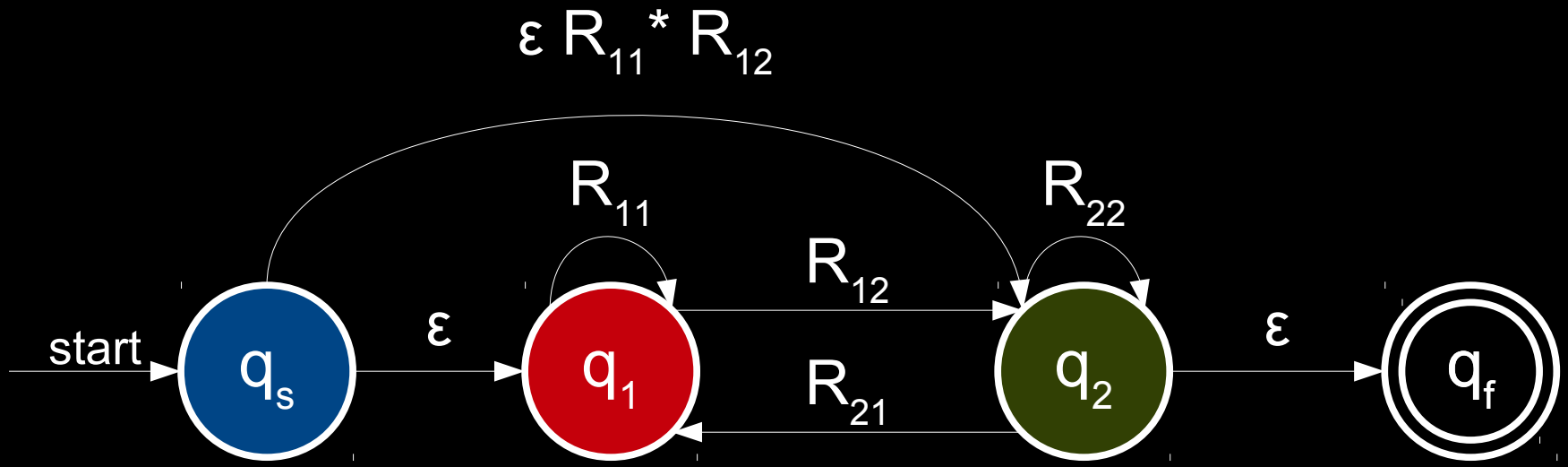
Cool! Since we can glue machines together, we can glue languages together as well.

How are we going to do that?

$$a^{+}(.a^{+})^{*}@a^{+}(.a^{+})^{+}$$

Wow! We've got a new way
of describing languages.

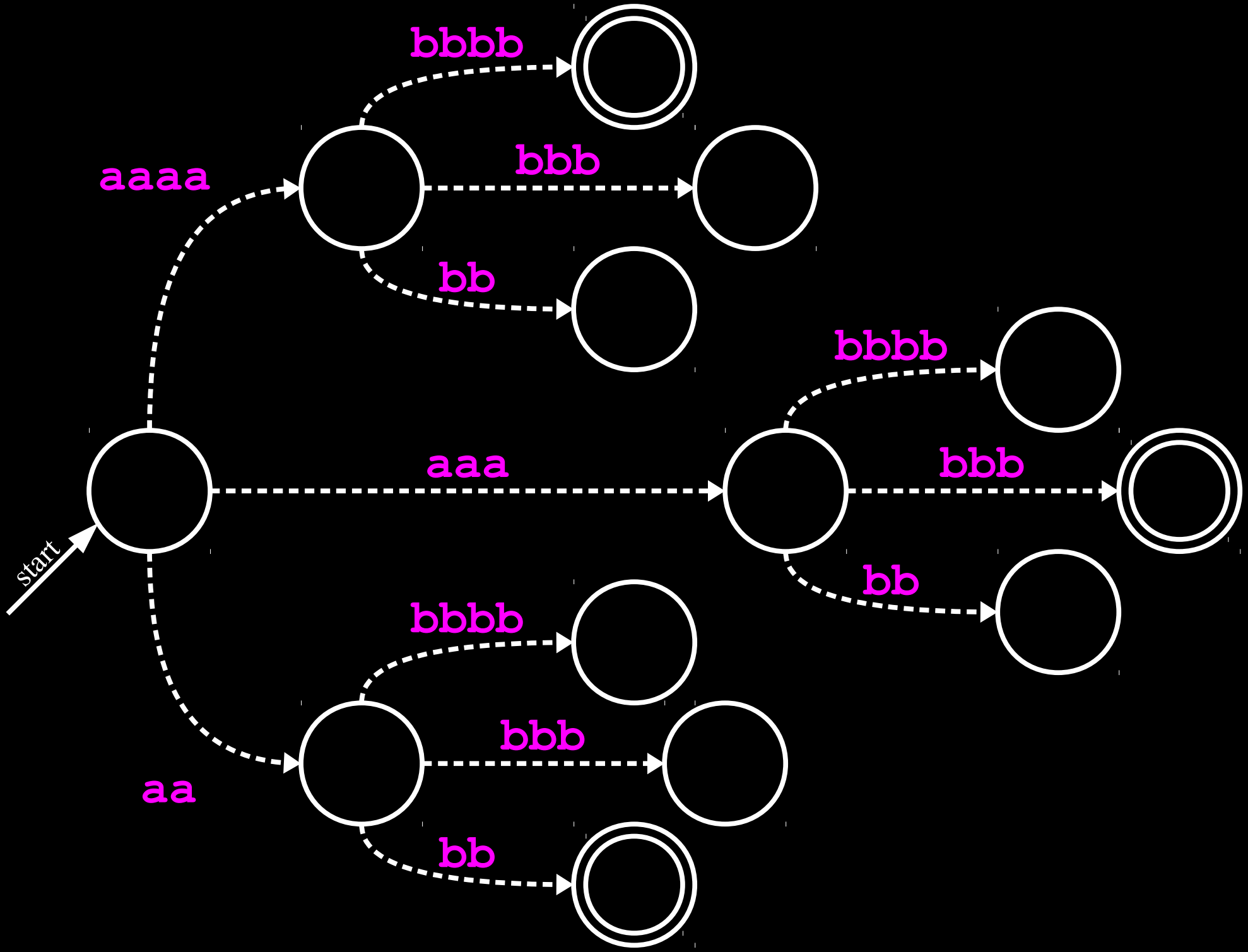
So what sorts of languages
can we describe this way?


ε R11* R12
R11
R22
R12
start
ε
ε
R21
qs
q1
q2
qf

Awesome! We got back the
exact same class of languages.

It seems like all our models give us the same power! Did we get every language?

$$xw \in L$$
$$yw \notin L$$


bbbb
aaaa
bbb
bb
bbbb
aaa
bbb
start
bb
bbbb
bbb
aa
bb

Wow, I guess not.

But we did learn something cool:

We have just explored what problems can be solved with finite memory.

So what else is out there?

Can we describe languages another way?

$$S \rightarrow aX$$
$$X \rightarrow b \mid C$$
$$C \rightarrow Cc \mid \varepsilon$$

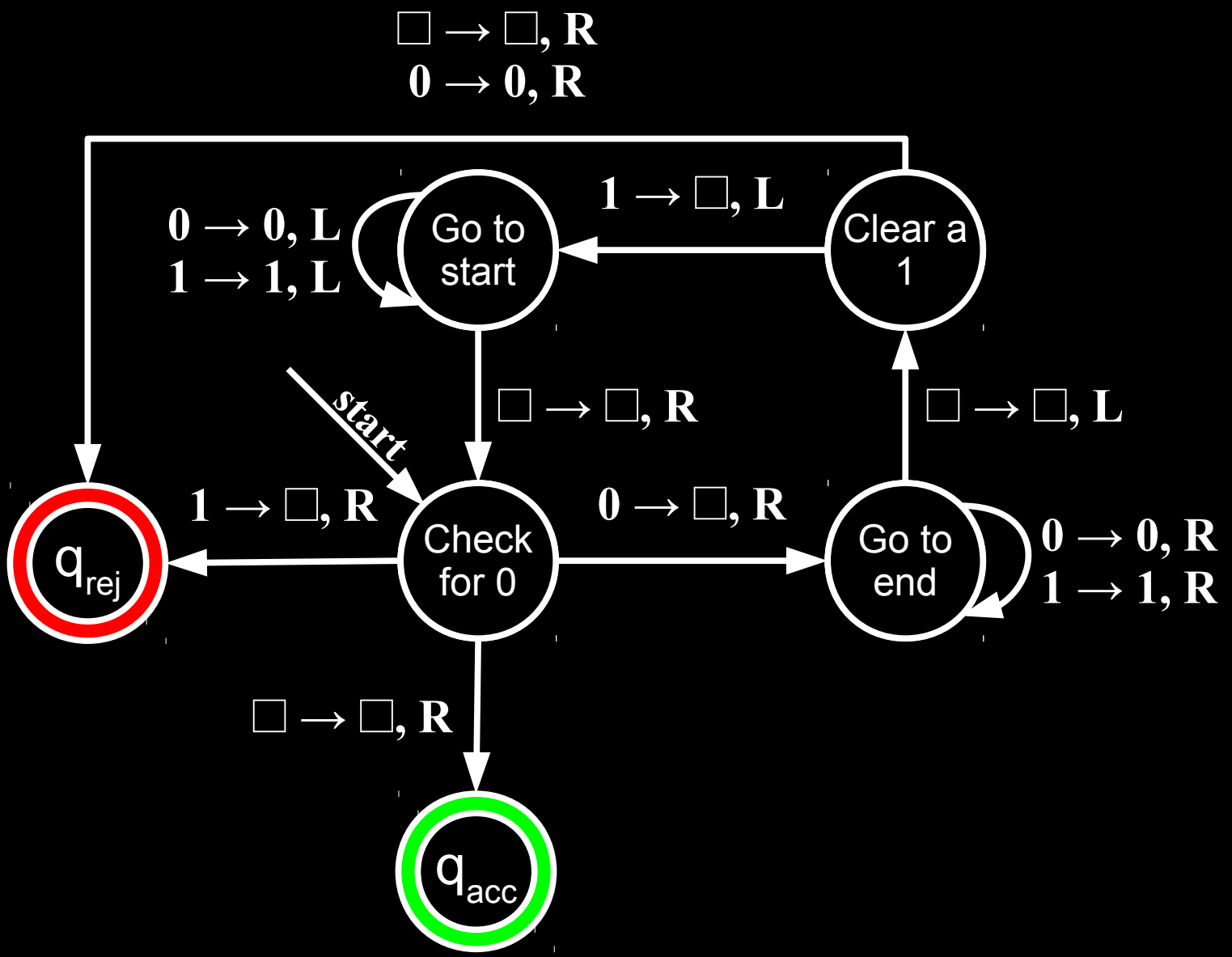
Awesome!

So, did we get every language yet?

$$|\Sigma^*| < |\wp(\Sigma^*)|$$

Hmmm... guess not.

So what if we make our memory a little better?

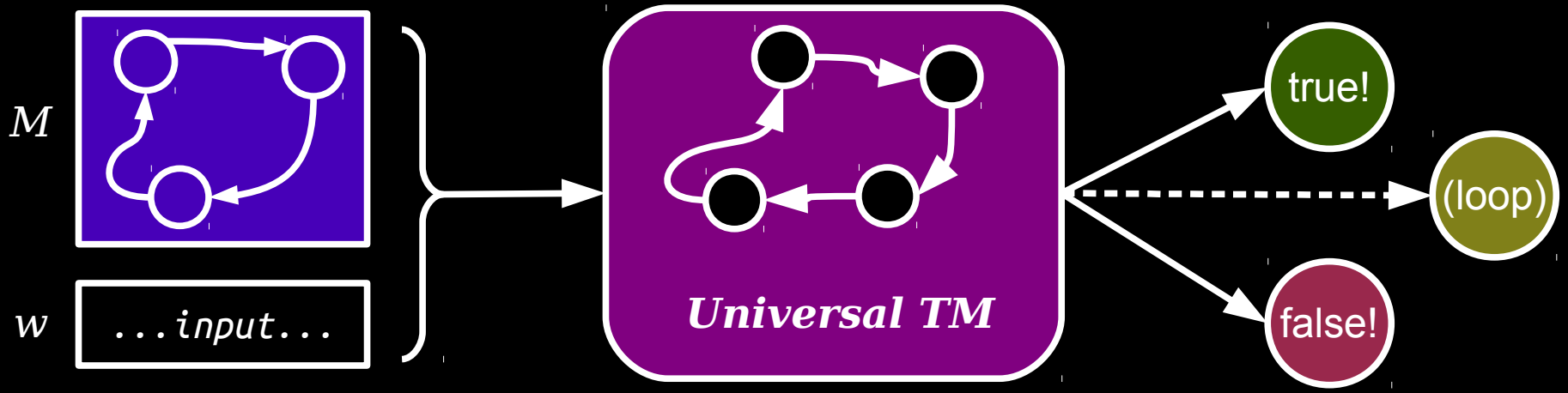

□ → □, R
0 → 0, R
0 → 0, L
1 → 1, L
Go to start
1 → □, L
Clear a 1
start
□ → □, R
□ → □, L
1 → □, R
0 → □, R
Check for 0
Go to end
0 → 0, R
1 → 1, R
q_rej
□ → □, R
q_acc

Cool! Can we make these more powerful?

Wow! Looks like we can't
get any more powerful.

(The ***Church-Turing thesis*** says
that this is not a coincidence!)

So why is that?


M
w
...input...
Universal TM
true!
(loop)
false!

Wow! Our machines can simulate one another!

This is a theoretical justification for why all these models are equivalent to one another.

So... can we solve everything yet?

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

const vector<string> kToPrint = {
    ...
};

string mySource() {
  string result;
  for (string line: kToPrint) {
    if (line == "@") {
      for (string inLine: kToPrint) {
        result += "  R\")" + inLine + ")\",\n";
      }
    } else {
      result += line + '\n';
    }
  }
  return result;
}

int main() {
  cout << mySource() << endl;
}
```

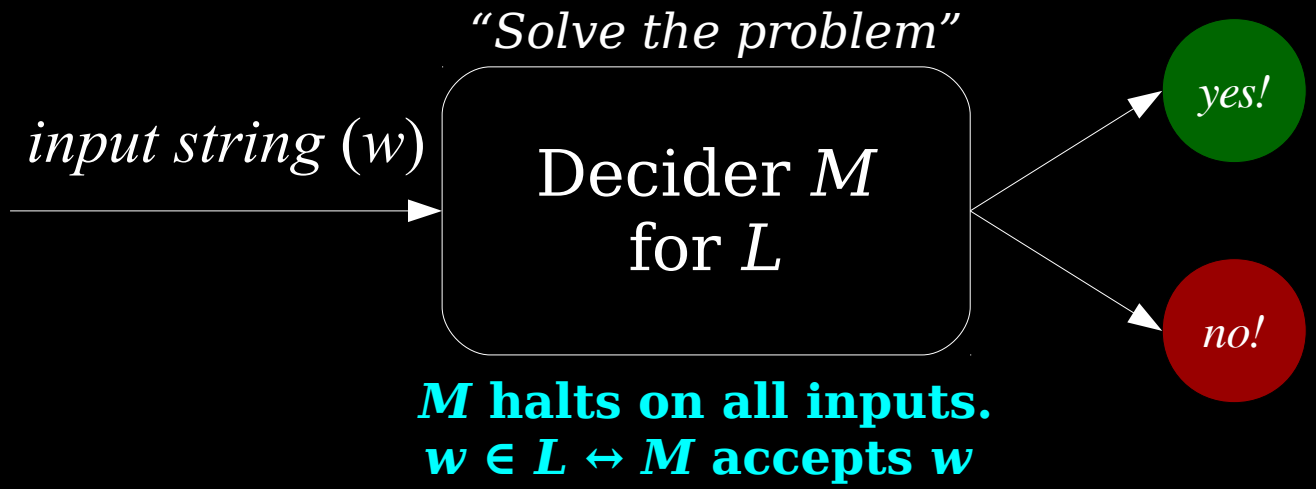
Weird! Programs can gain access
to their own source code!

Why does that matter?

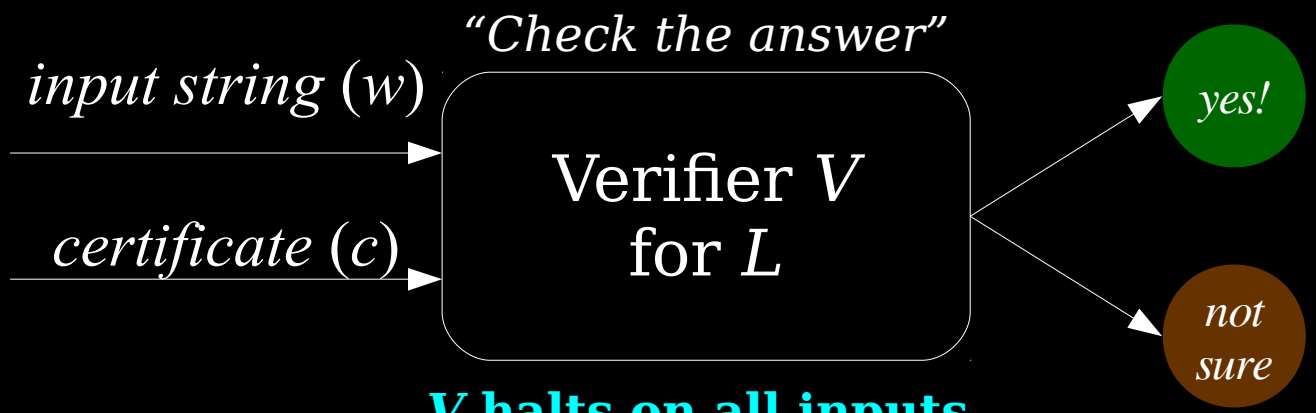
```
int main() {
   string me = mySource();
   string input = getInput();

   if (willAccept(me, input)) {
      reject();
   } else {
      accept();
   }
}
```

Crazy! The power of self-reference immediately limits what TMs can do!

What if we think about solving problems in a different way?



M halts on all inputs.
$w \in L \leftrightarrow M$ accepts w



V halts on all inputs.
$w \in L \leftrightarrow \exists c \in \Sigma^*.\ V$ accepts $\langle w, c \rangle$

Crazy!

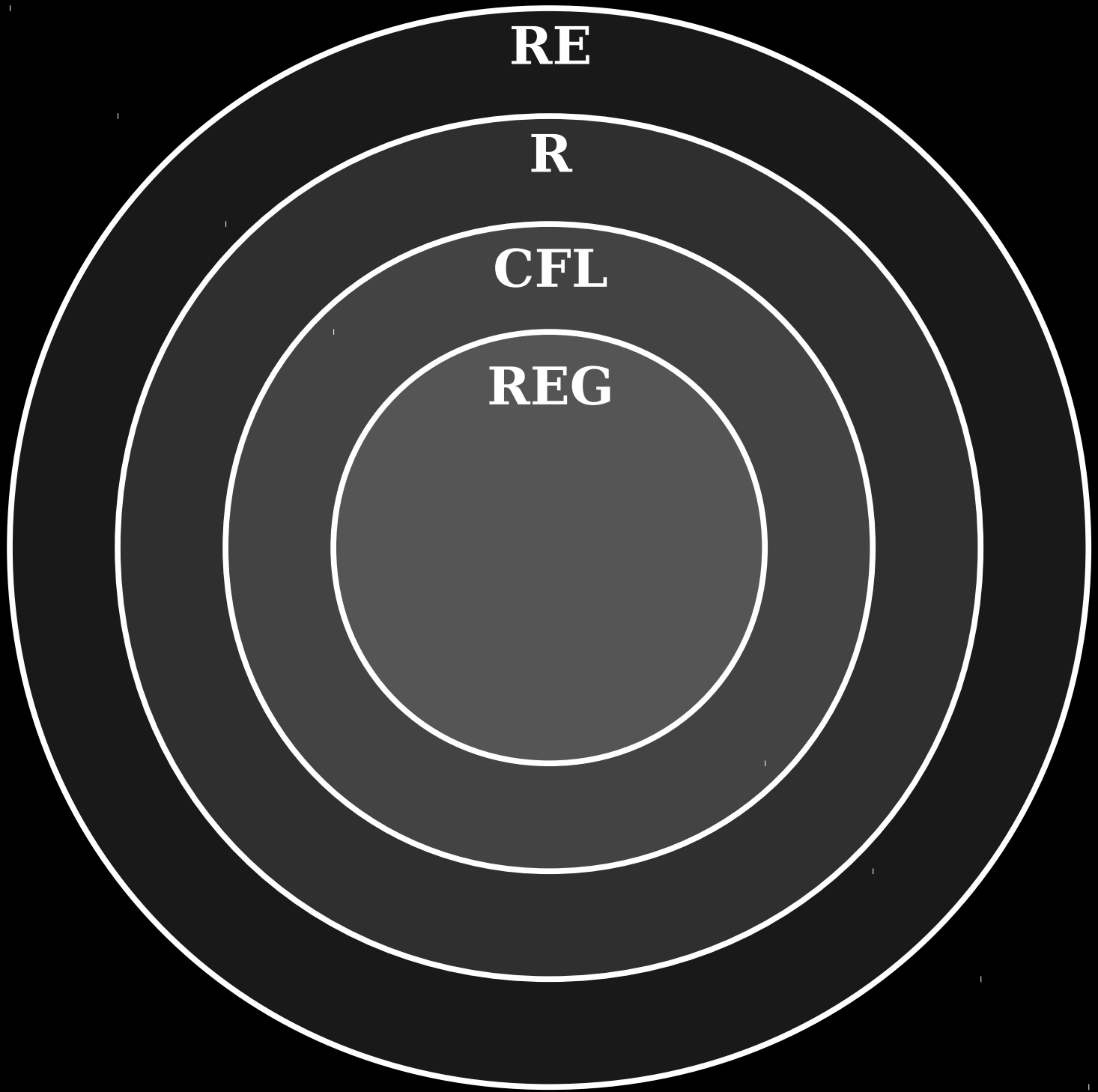
Can we at least verify everything?

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

Oh great. Some problems
are impossible to solve.

But look what we learned along the way!


RE
R
CFL
REG

Wow. That's pretty deep.

So... what can we do efficiently?

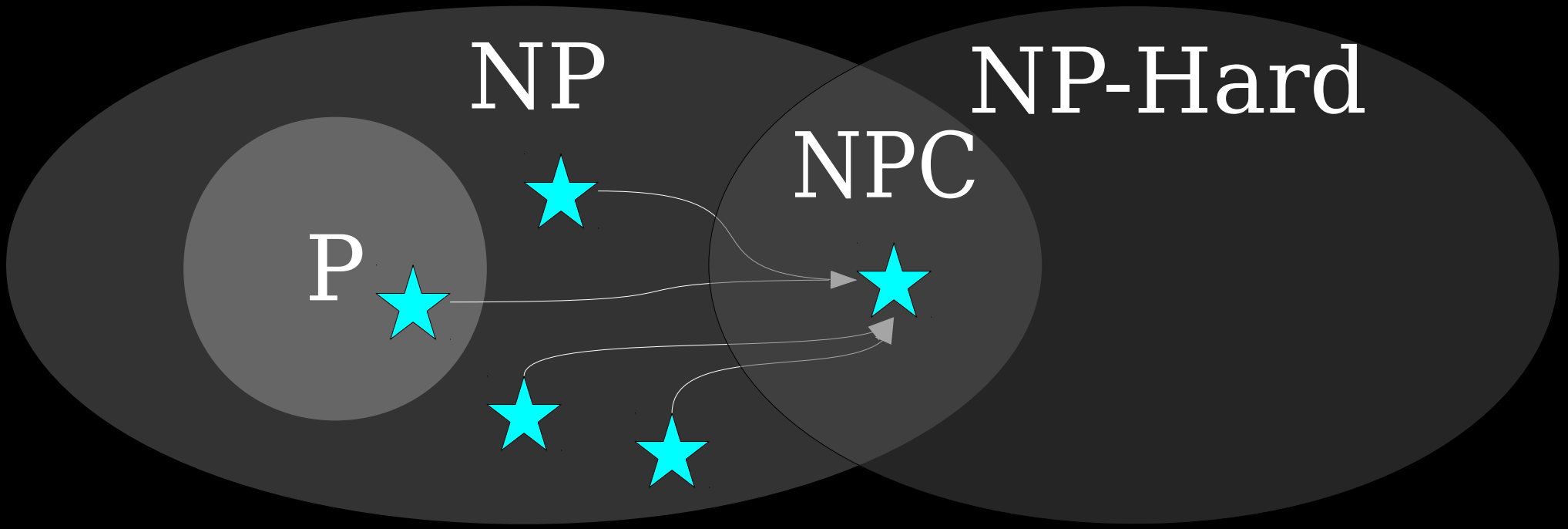
P


NP

So... how are you two related again?

No clue.

But what do we know about them?

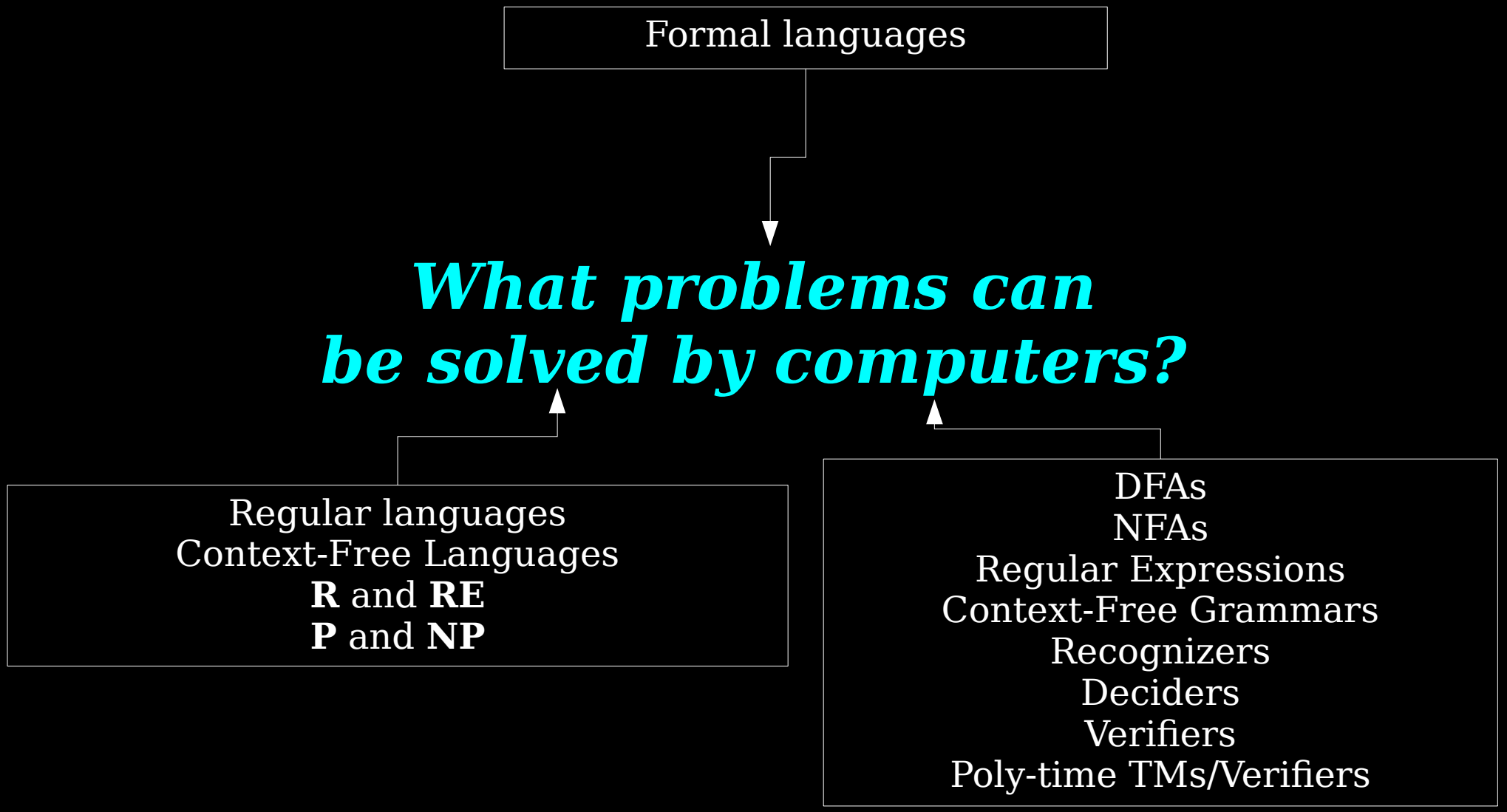

NP
NP-Hard
NPC
P

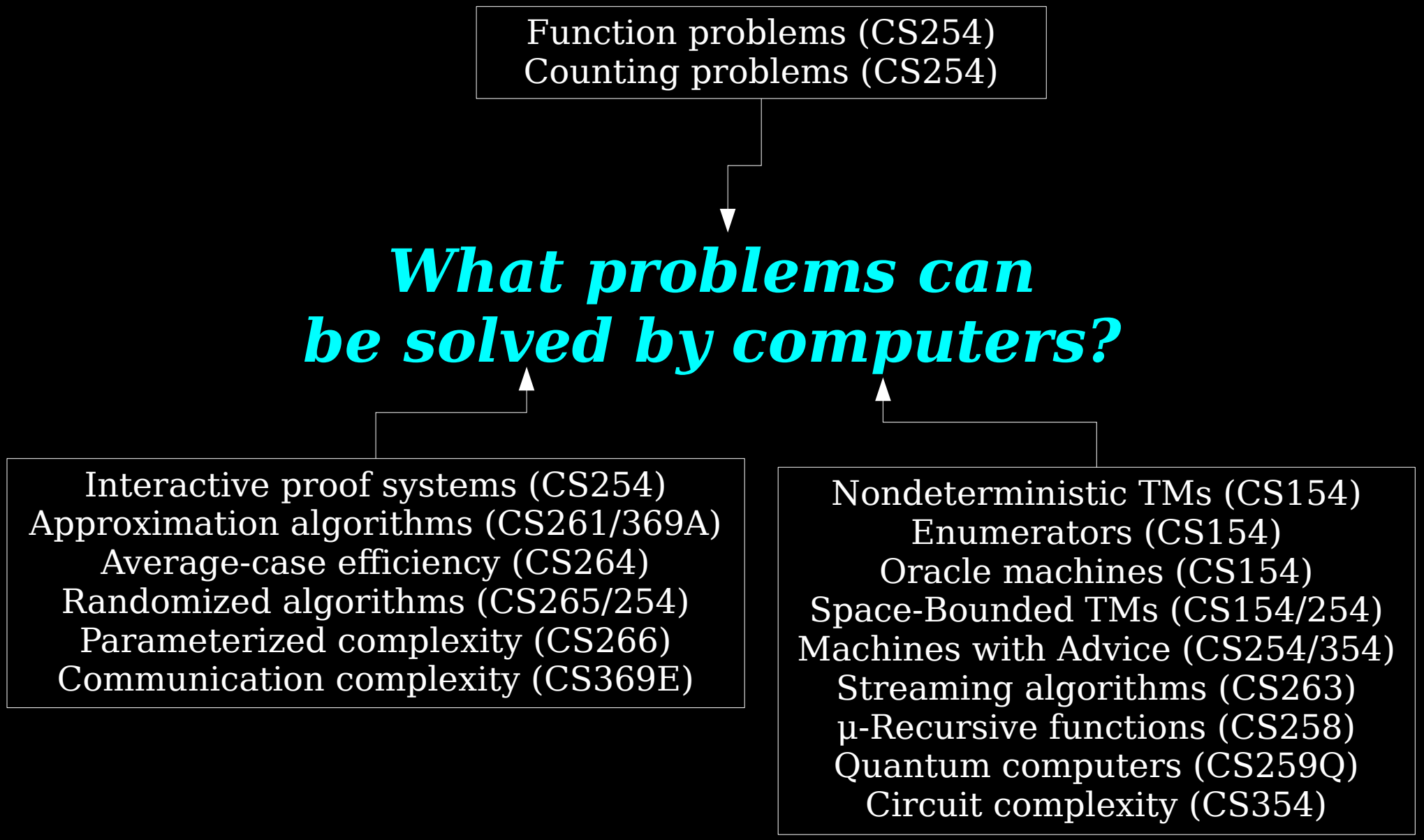
We've gone to the absolute limits of computing.

We've probed the limits of efficient computation.

Congratulations on making it this far!

What's next in CS theory?


Formal languages
What problems can be solved by computers?
Regular languages
Context-Free Languages
R and RE
P and NP
DFAs
NFAs
Regular Expressions
Context-Free Grammars
Recognizers
Deciders
Verifiers
Poly-time TMs/Verifiers


Function problems (CS254)
Counting problems (CS254)
What problems can be solved by computers?
Interactive proof systems (CS254)
Approximation algorithms (CS261/369A)
Average-case efficiency (CS264)
Randomized algorithms (CS265/254)
Parameterized complexity (CS266)
Communication complexity (CS369E)
Nondeterministic TMs (CS154)
Enumerators (CS154)
Oracle machines (CS154)
Space-Bounded TMs (CS154/254)
Machines with Advice (CS254/354)
Streaming algorithms (CS263)
μ-Recursive functions (CS258)
Quantum computers (CS259Q)
Circuit complexity (CS354)

How do we actually get the computer to effectively solve problems?

DFA design intuitions
Guess-and-check
Massive parallelism
Myhill-Nerode lower bounds
Verification
Polynomial-time reductions

How do we actually get the computer to effectively solve problems?

Algorithm design (CS161)
Efficient data structures (CS166)
Modern algorithmic techniques (CS168)
Approximation algorithms (CS261/CS369A)
Average-case efficient algorithms (CS264)
Randomized algorithms (CS265)
Parameterized algorithms (CS266)
Geometric algorithms (CS268)
Game-theoretic algorithms (CS364A/B)

What mathematical structures arise in computer science?

Sets
Propositional and First-Order Logic
Equivalence Relations
Strict Orders
Functions
Injections, Surjections, Bijections
Graphs
Planar and Bipartite Graphs
Polynomial-Time Reductions

What mathematical structures arise in computer science?

Groups, Rings, and Fields (Math 120, CS255)
Trees (Math 108, CS161)
Graphs (Math 107, Math 108)
Hash Functions (CS109, CS161, CS255)
Permutations (Math 120, CS255)
Monoids (CS149)
Lattices and Semilattices (CS143)
Control-Flow Graphs (CS143)
Vectors and Matrices (Math 113, EE103, CS205A)
Modal Logic (Phil 154, CS224M)
Mapping Reductions (CS154)

Where does CS theory meet CS practice?

Finite state machines
Regular expressions
CFGs and programming languages
Password-checking
Secure voting
Polynomial-time reducibility
NP-hardness and **NP**-completeness

Where does CS theory meet CS practice?

Compilers (CS143)
Computational logic (CS157)
Program optimization (CS243)
Data mining (CS246)
Cryptography (CS255)
Programming languages (CS258)
Network protocol analysis (CS259)
Techniques in big data (CS263)
Graph algorithms (CS267)
Computational geometry (CS268)
Algorithmic game theory (CS364)

A Whole World of Theory Awaits!

What's being done here at Stanford?

Algorithms ∩ Game theory
(Tim Roughgarden)

Learning patterns in randomness
(Greg Valiant)

Moving from secrecy to privacy
(Omer Reingold)

Approximating NP-hard problems
(Moses Charikar)

Optimizing programs... randomly
(Alex Aiken)

Computing on encrypted data

(Dan Boneh)

Interpreting structure from shape
(Leonidas Guibas)

Correcting errors automatically
(Mary Wooters)

So many options – what to do next?

Really enjoyed this class?

Give CS154 a try!

Interested in trying out CS?

Continue on to CS109!

Want to see this material come to life?

Check out CS143!

Want to tame infinity?

Dive into Math 161!

Like discrete structures?

Try Math 107 or Math 108!

Want to just go write code?

Take CS107!

Keep on exploring! There's so much more to learn!

A Final “Your Questions”

“What do you consider is your contribution to larger societal issues like poverty, climate change, hunger, etc.? Do you ever feel like you are not doing enough?”

“What would it take for you not to teach at Stanford?”

The answers to these questions are related. I'll take them at the same time.

“Can we model the human brain with a Turing machine?”

Time for another installment of “Keith Talks About Different Forms of Truth!”

“What is your opinion on the rescinded acceptance letters for Harvard pre-Frosh over dank memes?”

"What is your opinion on the rescinded acceptance letters for Harvard pre-Frosh over ~~dank~~ memes?"

racist, sexist,
antisemitic, and vile

I'll take this one in class only, given that there are a lot of strong emotions in this one.

“What role can ethics play in CS curriculum and what problems arise when engineers ignore ethics?”

I have a somewhat unusual take on this one. I’m curious to hear what you think!

Anything else?

Final Thoughts

There are more problems to solve than there are programs capable of solving them.

There is so much more to explore and so many big questions to ask – ***many of which haven't been asked yet!***


Theory
Practice

You now know what problems we can solve, what problems we can't solve, and what problems we believe we can't solve efficiently.

My questions to you:

What problems will you ***choose*** to solve?
Why do those problems matter to you?
And how are you going to solve them?