CS 3100, Models of Computation, Spring 20, Lec 4

Ganesh Gopalakrishnan
School of Computing
University of Utah
Salt Lake City, UT 84112

bit.ly/3100s20Syllabus

SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

Questions from Lec-3 Answered

- Define DFA mathematically via its language
  - The set of strings that take it from its initial state to ONE of its final states
- Does it matter if a DFA emits a decision so long as it hasn't fallen outside the language!
  - Or does it need to be "restarted"?
  - Answer: These are a matter of interpretation of our intended use
  - In general, a device that emits a decision so long as it hasn't fallen outside of the language is all we need!
- E.g. "Odd 1's" : possible to rescue it by sending it more 1's
- E.g. "Every block of 3 has two 1's" : once it violates this, there is no way to rescue – so crash into a black-hole

More Questions from Lec-3 Answered

- On vacuous statements
- Consider this:
  - Every person 6 feet tall or more must wear a red shirt
  - What does this mean for a 5-footer?

- Same story with “every block of length 3 must have two 1’s”

Observations from Lec-3

- The state naming trick gets many DFAs designed
  - E.g. “The third from the last (or “third-last”) bit is a 1”

- But it is not enough for many other languages
  - E.g.
    - A DFA for the set of strings that begin with 01 and end with 10 and don’t contain a 0010 anywhere ☺
    - It has a DFA but…. Think of a state naming trick → can’t do ☹
      - No worries – we can arrive at this machine via Boolean Operations

- And some languages don’t even have DFA !!

Lecture 3, covering Chapter 4.6-4.9

We will now study the Pumping Lemma

A regular language is one that has a DFA

- IF one can describe a language exactly via a DFA
  - THEN the language is regular
    - Regular(L) ➔ ExistsDFA(L)

- Contrapositive:
  - IF no DFA exists for a language
  - THEN the language is not regular
    - ! ExistsDFA(L) ➔ ! Regular(L)

We will introduce the “Pumping Lemma” (PL)

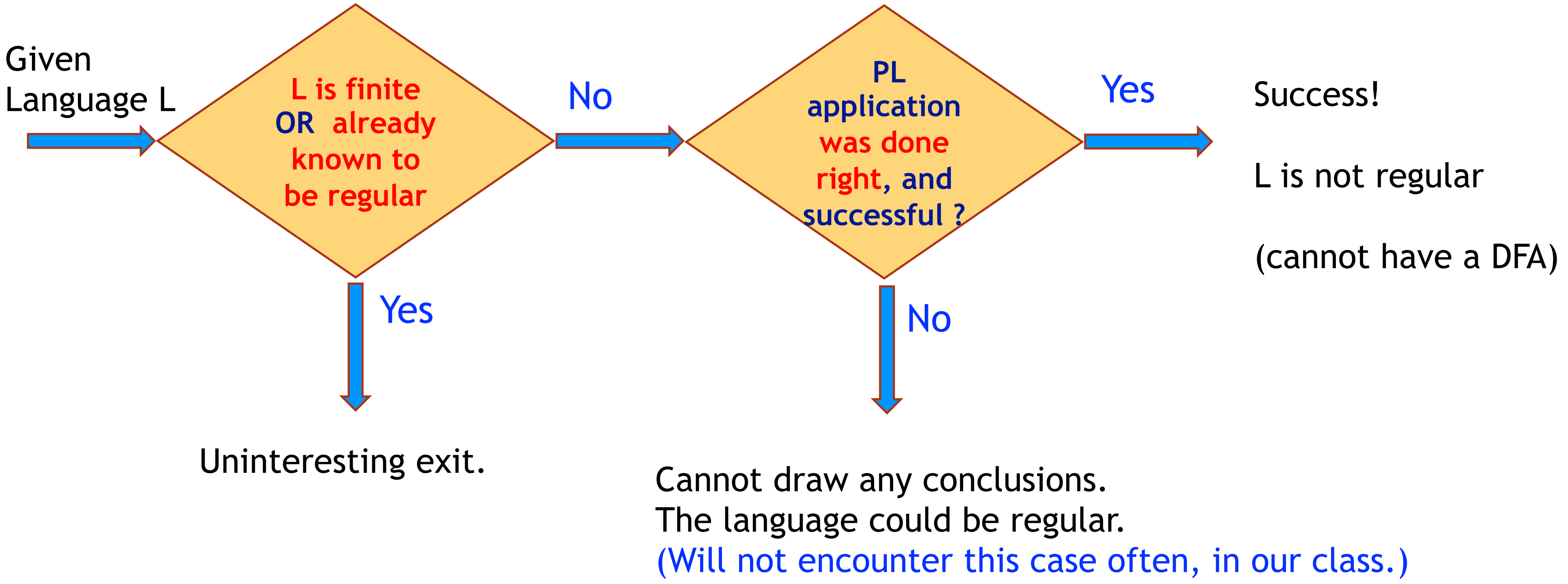
- The crux of the PL is this direction of argument

! ExistsDFA(L) → ! Regular(L)

- We will accept a candidate language L
- We will show that no DFA can exist for L
- We will then conclude that L is not regular

- We will never use the PL to show that a language is regular

Flow-chart of use of our PL



Given Language L

L is finite OR already known to be regular

Yes

Uninteresting exit.

No

PL application was done right, and successful ?

Yes

Success!

L is not regular

(cannot have a DFA)

No

Cannot draw any conclusions.
The language could be regular.
(Will not encounter this case often, in our class.)

How can we correctly apply the PL?

- We must rule out every possible DFA for L

How can we rule out every possible DFA?

- Find an important property any DFA must obey

- Rule out that this property does not hold

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - (Of course) w has to go from the start state to a state in F
  - w has to loop somewhere
  - This loop has to be among the first N states of the DFA
  - Nothing else is known about such a path (it purely depends on the DFA)

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA,

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - w = x y z
    - w consists of an initial part x that reaches the first loop and then yz that follows x, and reaches a final state f in F

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - w = x y z
    - w consists of an initial part x that reaches the first loop and then yz that follows x, and reaches a final state f in F
  - x could be empty
    - The loop could begin right at the first state

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - w = x y z
    - w consists of an initial part x that reaches the first loop and then yz that follows x, and reaches a final state f in F
  - x could be empty
    - The loop could begin right at the first state
  - y is non-empty
    - There is a loop

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - w = x y z
    - w consists of an initial part x that reaches the first loop and then yz that follows x, and reaches a final state f in F
  - x could be empty
    - The loop could begin right at the first state
  - y is non-empty
    - There is a loop
  - y could be taken any number of times and we can still reach a state f in F
    - The loop is along the path; one can skip it; one can take it more than once

What is one important property of any DFA?

- If that DFA accepts a string w that is at least as long as the number of states (N) of the DFA, then
  - w = x y z
    - w consists of an initial part x that reaches the first loop and then yz that follows x, and reaches a final state f in F
  - x could be empty
    - The loop could begin right at the first state
  - y is non-empty
    - There is a loop
  - y could be taken any number of times and we can still reach a state f in F
    - The loop is along the path; one can skip it; one can take it more than once
  - All such strings must be in L

    - xz must reach F (hence in L)
    - x y z must reach F (hence in L)
    - x yy z must reach F (hence in L)
    - x y^i z must reach F for i >= 0

When can a DFA not exist for L ?

- If we assume there is an N-state DFA (unknown internal details)
- And it accepts a string w of length >= N
  - That is, w = xyz goes from the initial to a final state
- And
  - For any xyz split of w, with xy being of length at-most N
  - we have x y^i z not being in the language for some i

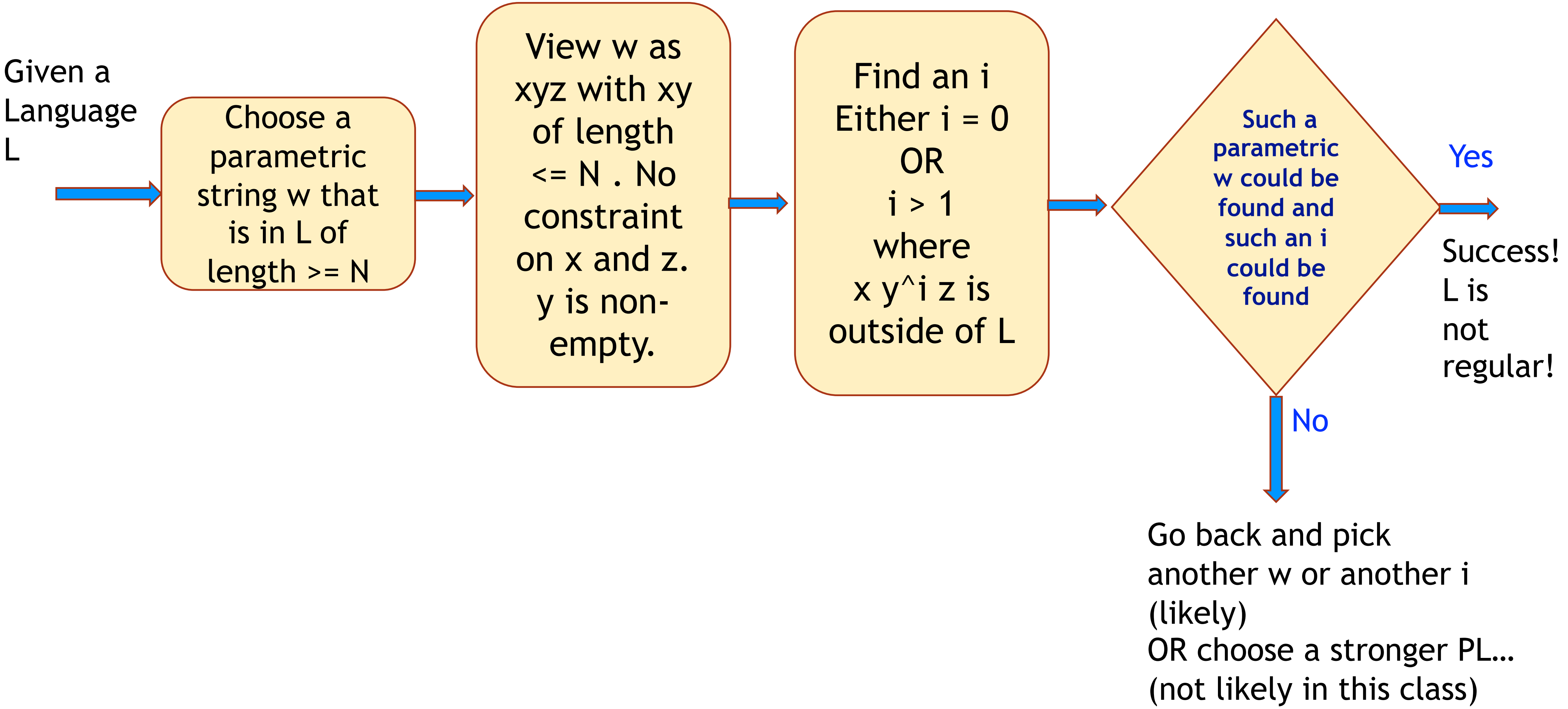
- Then there can’t be a DFA for L

So, here is how you rule out a DFA

- Pick w that is of length N (or much more…) in L
  - Not a specific w,
  - But a parametric string such as 0^N 1^N whose length is a function of N
- Show that if w = xyz
- And length(xy) <= N
- And there is an i such that x y^i z is not in L

- Then there is no DFA for L

Flow-chart for applying the PL



Given a Language L

Choose a parametric string w that is in L of length >= N

View w as xyz with xy of length <= N . No constraint on x and z. y is non-empty.

Find an i Either i = 0 OR i > 1 where x y^i z is outside of L

Such a parametric w could be found and such an i could be found

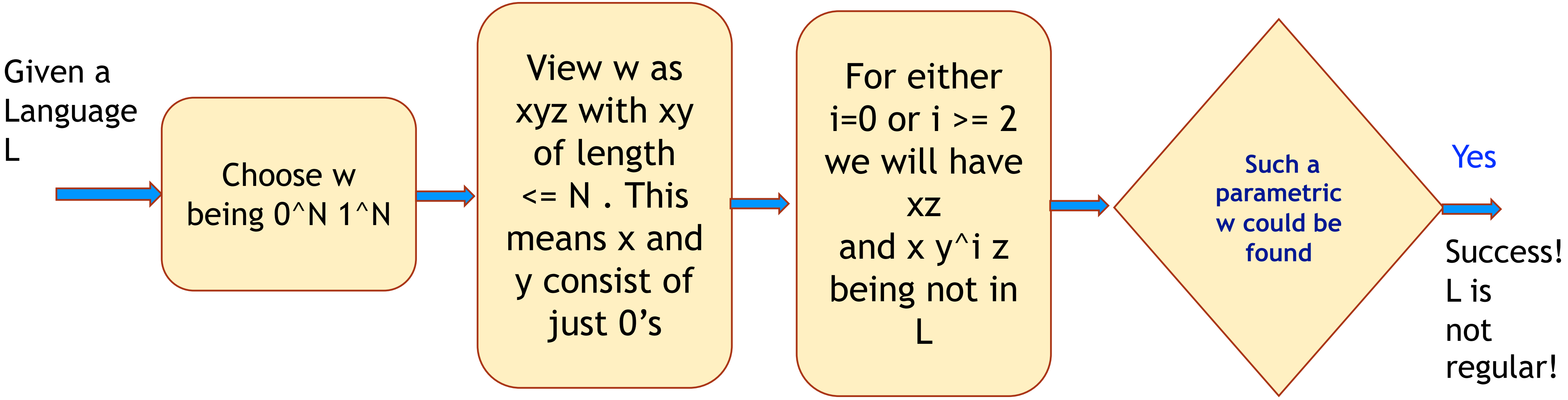
Yes

Success! L is not regular!

No

Go back and pick another w or another i (likely)
OR choose a stronger PL…
(not likely in this class)

Example: show L = {0^i 1^i : i >= 0} not reg.



Given a Language L

Choose w being 0^N 1^N

View w as xyz with xy of length <= N . This means x and y consist of just 0's

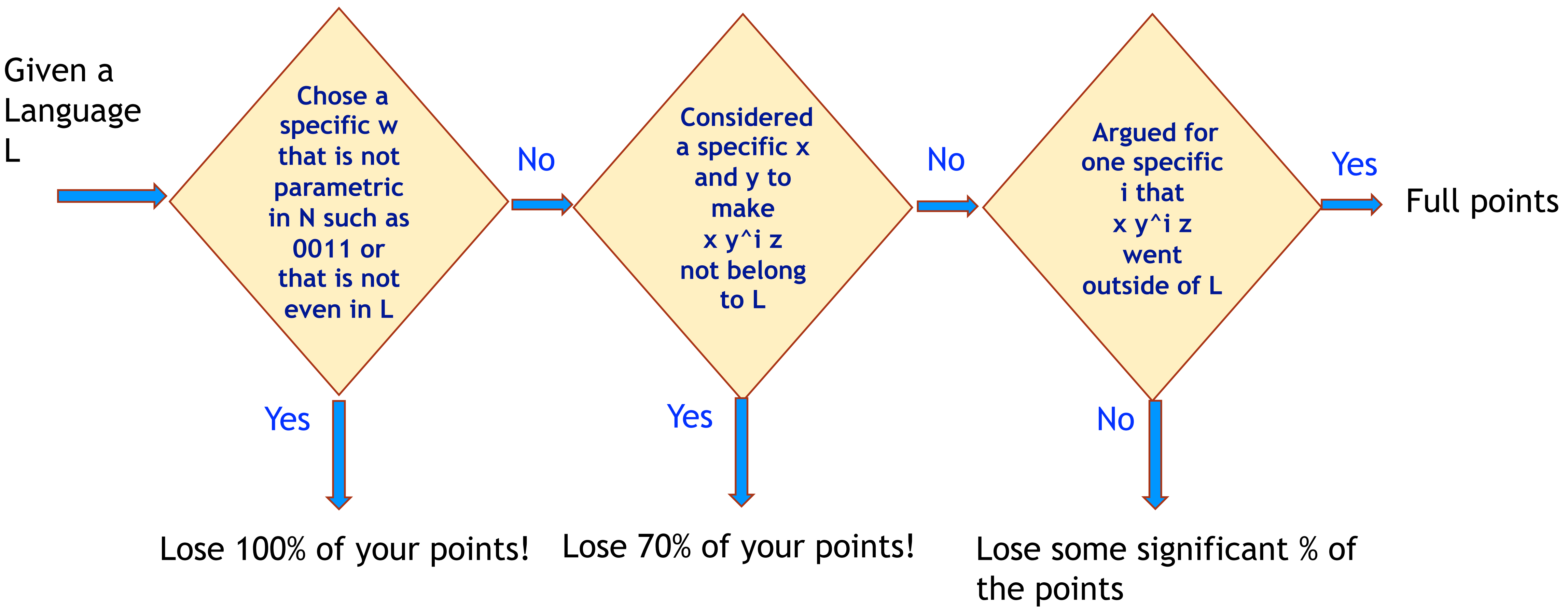
For either i=0 or i >= 2 we will have xz and x y^i z being not in L

Such a parametric w could be found

Yes

Success! L is not regular!

How to avoid losing points in a PL proof



Given a Language L

Chose a specific w that is not parametric in N such as 0011 or that is not even in L

Yes

Lose 100% of your points!

No

Considered a specific x and y to make x y^i z not belong to L

Yes

Lose 70% of your points!

No

Argued for one specific i that x y^i z went outside of L

No

Lose some significant % of the points

Yes

Full points

Show these languages not to be regular

- Equal # of 0’s and 1’s
- L_{ww}
- Language of palindromes
- Balanced parentheses language