Biçimsel Diller ve Otomata Teorisi

Sunu X
Karar Verilebilirlik

İZZET FATİH ŞENTÜRK

Equivalence

- Three basic questions to consider
  - How can we tell whether two regular expressions define the same language
  - How can we tell whether two FAs accept the same language
  - How can we tell whether the language defined by an FA has finitely many or infinitely many words in it, or any words at all?
- In mathematical logic, we say that a problem is effectively solvable if there is an algorithm that provides the answer in a finite number of steps, no matter what the particular inputs are

Equivalence

- The maximum number of steps the algorithm will take must be predictable before we begin to execute the algorithm
- For example
  - What is the solution to a quadratic equation?
  - The quadratic formula provides an algorithm with a predetermined number of arithmetic operations: four multiplications, two subtractions, one square root, one division
  - The number of steps in the algorithm is never gerater no matter what the particular coeeficients of the polynomial are
  - Other suggestions for solving a quadratic equation: keep guessing until you find a number that satisfies the equation – does not guarantee to work in a fixed number of steps

Decidable

- Definition:
  - An effective solution to a problem that has a yes or no answer is called a **decision procedure**
  - A problem that has a decision procedure is called **decidable**
- The first thing we want to decide
  - Whether two REs determine the exact same language
  - We might, simply, use the expressions to generate many words from each language until we find one that is not in the language of the other. We may generate the words in size order
  - In practice, this method works fairly well but there is no mathematical guarantee that we find such a word at any time in the next six years

Example

- Consider two expressions
  - **a**(**a** + **b**)* and (**b** + **Λ**)(**baa** + **ba***)*
- The wordsfirst expression begin with the letter a and the second expression begin with the letter b.
- These expressions have no word in common. This fact is very clear

Example

- Consider the two expressions
  - (**aa** + **ab** + **ba** + **bb**)* and ((**ba** + **ab**)*(**aa** + **bb**)*)*
- Both define the language of all strings over Σ = {a b} with an even number of letters
- We could generate many examples of words from the languages each represents but we could not find a difference. Could we then conclude they are equivalent?
- Generating words and praying for inspiration is not an effective procedure and it does not decide the problem

Equivalence

- If we had a decision procedure to determine whether two REs were equivalent we could use it to determine whether two FAs were equivalent
  - We would convert the FAs into REs and then decide
- Similarly, if we had an effective procedure to determine whether two FAs were equivalent, we could use it to decide the problem for REs by converting them into FAs
- We already developed all the algorithms necessary to decide the "equivalency problem" for FAs and thereby REs

Equivalence

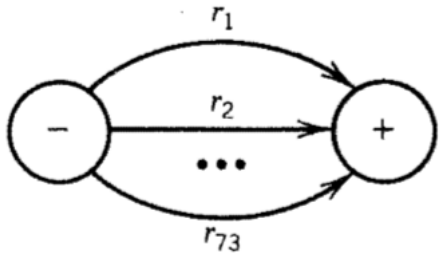
- Given two languages L1 and L2 defined by either REs or FAs, we have developed the procedures necessary to produce FA for the languages: L_1', L_2', $L_1 \cap L_2$', and $L_2 \cap L_1$'
- Therefore, we can produce an FA that accepts the language
  - $(L_1 \cap L_2') + (L_2 \cap L_1')$
  - If L_1 and L_2 are the same language, this machine cannot accept any words
  - If this machine accepts even one word, then L_1 is not equal to L_2 even if the one word is the null word

Equivalence

- How to determine whether an FA accepts any words
- Method 1
  - Conver the FA into RE. Every RE defines some words. First delete all stars. Then for each +, throw away the right half of the sum and the +. Remove the paranheses when we have no more *'s or +'s. We have a concatenation of a's, b's and the null
- Example: **(a + Λ)(ab* + ba*)*(Λ + b*)***
  - **(a + Λ)(ab + ba)(Λ + b)**
  - **(a)(ab)(Λ)**
  - *aab*

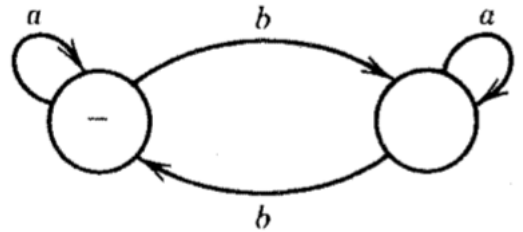
Equivalence

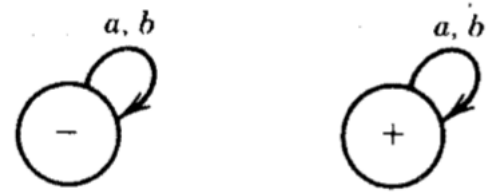
- If every RE defines at least one word, it seems at first glance that every FA must accept at least one word
  - How could we ever show that two languages are equal?
- The hole in this reasoning is that the process of converting this FA into a RE breaks down
- When we convert an FA to a RE, we come to this point at the last step. There must be edges running from – to +

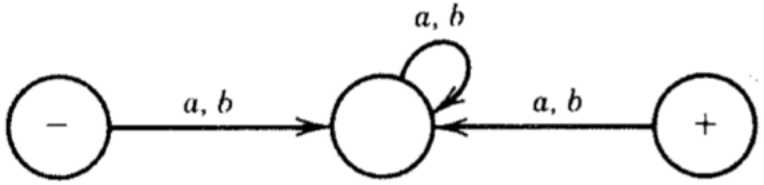


Equivalence

- However, at the last step, we can realize that <u>there are no paths</u> from – to + at all
- This could happen in three different ways:
  - The machine has no final states
  - The final state is disconnected from the start state
  - The final state is unreachable from the start state







Equivalence

- Method 2
  - Examine the FA to see whether or not there is any path from – to +
  - If there is any path, then the machine must accept some words
  - In a large FA with thousands of states and millions of directed edges, it may be impossible to decide whether there is a path from – to + without the guidance of an effective procedure

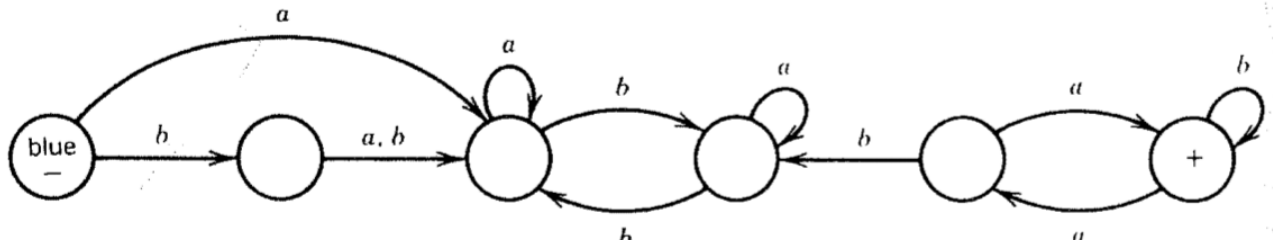
Equivalence

- One such procedure is this
  - Step 1: Paint the start state blue
  - Step 2: From every blue state, follow each edge that leads out of it and paint the destination state blue, then delete this edge from the machine
  - Step 3: Repeat step 2 until no new state is painted blue, then stop
  - Step 4: When the procedure has stopped, if any of the final states are painted blue, then the machine accepts some words and, if not, it does not

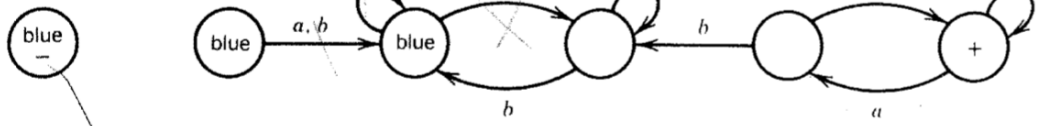
Example

- Apply the procedure on the machine

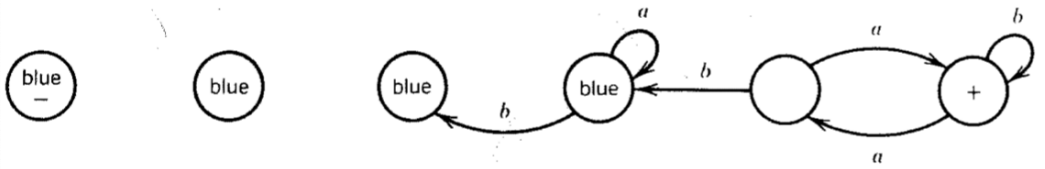
after step 1:



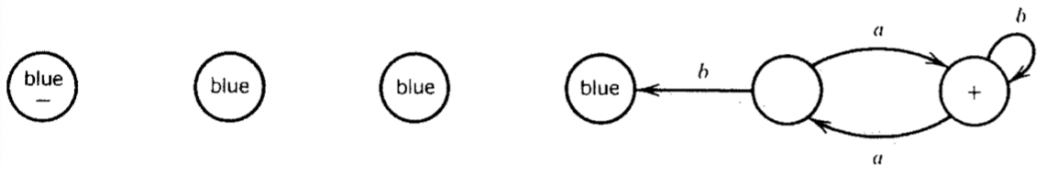
after step 2:



after step 2 again:



after step 2 again:



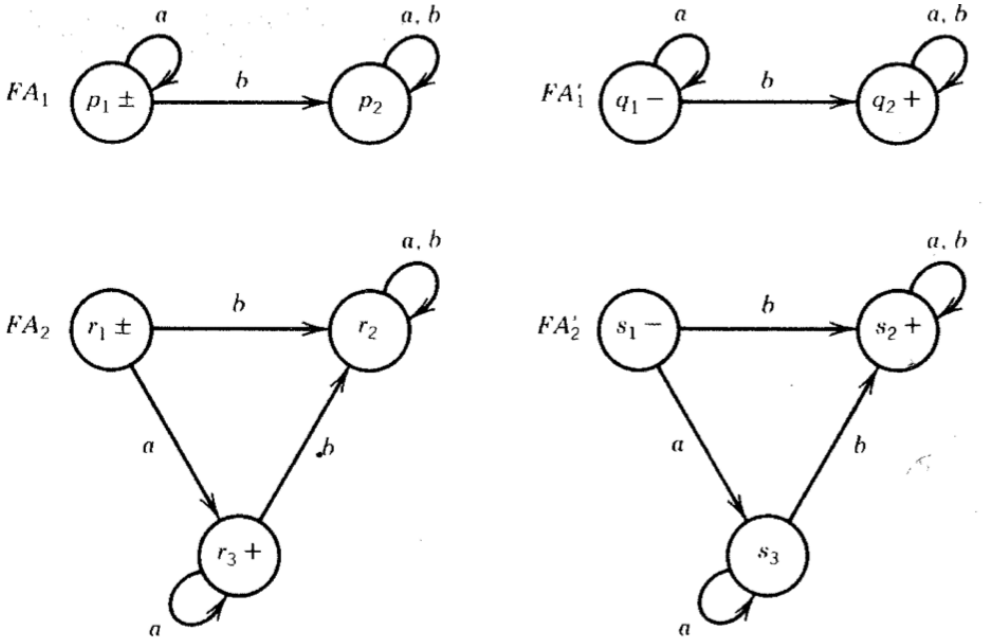
- No new states were painted blue this time, the procedure stops. The + is not blue. The machine accepts no words. Notice that step 2 cannot be repeated more times then there are total states in the machine

Equivalence

- Method 3
  - Test all words with fewer than N letters by running them on the FA. If the FA accepts none of them, then it accepts no words at all. (N = the number of states in the machine)
- These methods are all effective. The question of which is more efficient is a whole other issue. Our ultimate concern is the question, “what can be done and what cannot?”

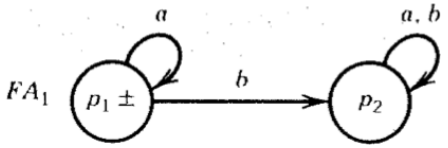
Example

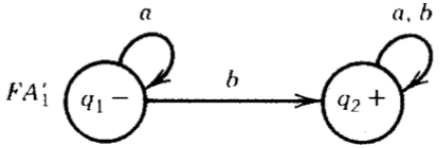
- Two REs are: $\mathbf{r_1} = \mathbf{a^*}$ and $\mathbf{r_2} = \mathbf{\Lambda} + \mathbf{aa^*}$
- We can understand that these two define the same language. Let us see the proof

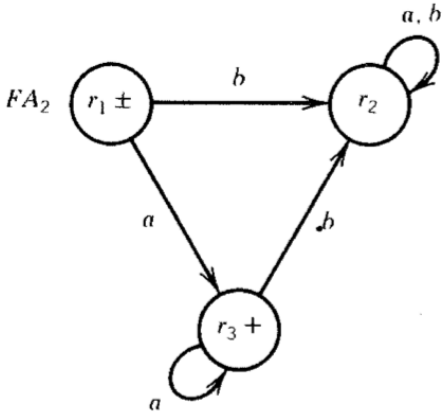


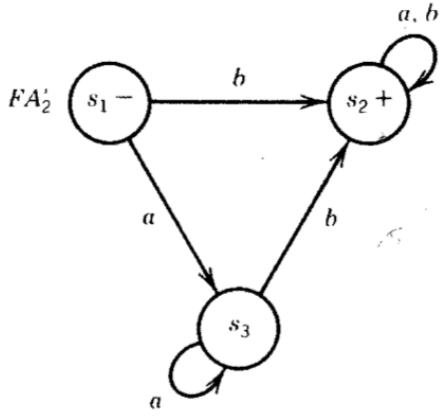
- Instead of using the logical formula $(L_1 \cap L_2') + (L_2 \cap L_1')$ we build our machine based on the equivalent set theory formula $(L_1' + L_2)' + (L_2' + L_1)'$

Example

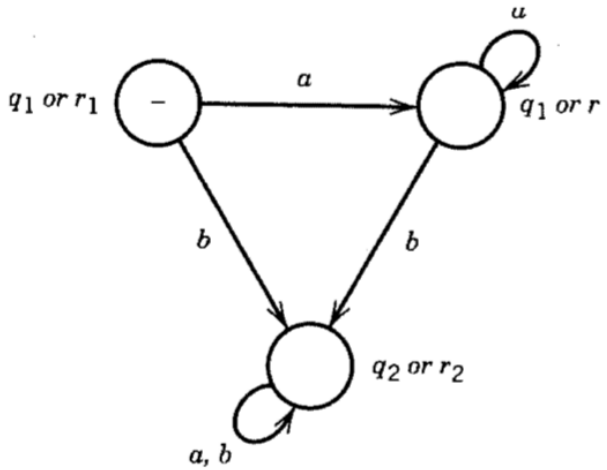




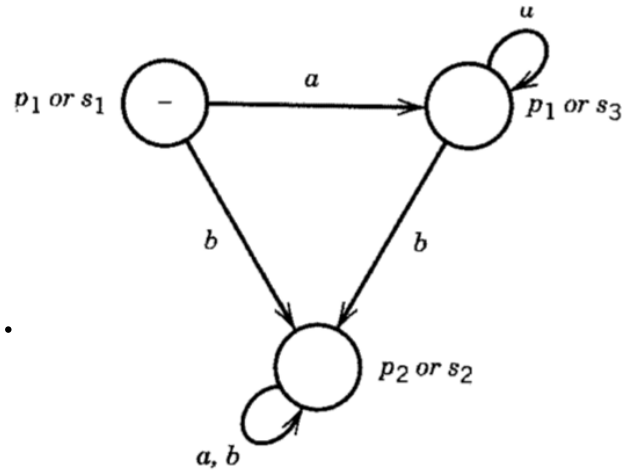




- $(FA_1' + FA_2)'$



- $(FA_1' + FA_2)'$



- Neither machine has any final states. If either accepts a word, $L_1 \neq L_2$

Theorem

- There is an effective procedure to decide whether
  - A given FA accepts any words
  - Two FAs are equivalent
  - Two REs are equivalent

Finiteness

- How can we tell whether an FA or RE defines a finite language or an infinite language?
  - With REs this is easy.The closure of any nonempty set, whether finite or infinite, is itself infinite
  - Even the closue of one letter is infinite
- If we have ever had to use the closure operator, the resulting language is infinite
  - This can be determined by scanning the expression to see whether it contains the symbol *. If the RE contains a *, then the language is infinite. The one exception is Λ* which is just Λ

Example

- The Λ exception can be tricky. Consider two REs
- (**Λ** + **aΛ***)(**Λ*** + **Λ**)* and (**Λ** + **aΛ**)*(**Λ*** + **Λ**)*
- Only the second defines an infinite language
- If the RE does not contain a *, the language is necessarily finite. Because the other rules of building REs cannot produce an infinite set from finite ones
- If we want to decide this question for an FA, we could first convert it to a RE