Chapter 11: Decidability

How to decide whether two regular expressions define the same language? (can we?)

How to decide whether two finite automata accept the same language? (can we?)

How to decide whether a language defined by a finite automaton have a finite number of words? Infinite? None? (can we?)

…

Three important questions:

1. Do two regular expressions define the same language?

2. Do two finite automata accept the same language?

1. Does the language defined by a finite automaton have a finite number of words? Infinite? None?

<u>Definition:</u> A problem is called a decision problem if its solution is either YES or NO.

<u>Definition:</u> a decision problem is decidable, or effectively solvable, if there is an algorithm that can find the answer for any input in a finite number of steps.

1. Do two regular expressions define the same language?
2. Do two finite automata accept the same language?

Problems 1 and 2 are equivalent:

It is enough to transfer the two regular expressions into equivalent finite automata (it could be done in a finite number of steps) and then resolve Problem 2.

Or we can transfer the two finite automata into regular expressions (it could be done in a finite number of steps) and then resolve Problem 2.

Problem 2: Finite algorithm

From two finite automata that accept languages L_1 and L_2, we construct a finite automaton that accepts

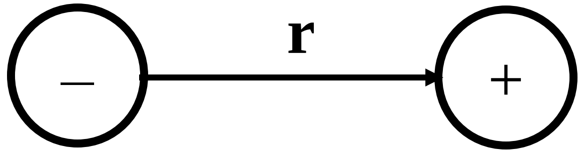
$$(L_1 \cap L_2') + (L_2 \cap L_1')$$

L_1 et L_2 are equivalent if and only if the language

$(L_1 \cap L_2') + (L_2 \cap L_1')$ is empty.

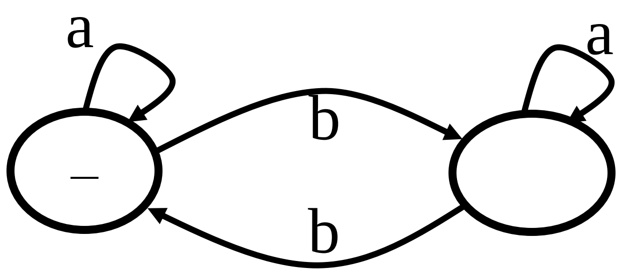
Method 1: Convert this finite automaton into a regular expression (using the finite algorithm defined in the proof of Kleene's theorem). If you end up with a regular expression then the language is not empty, otherwise it is empty.

If the algorithm leads to:

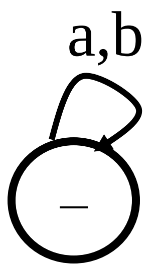


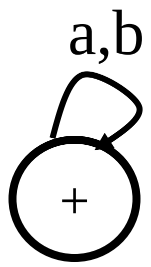
Then the finite automaton accepts at least one word

If the algorithm leads to:

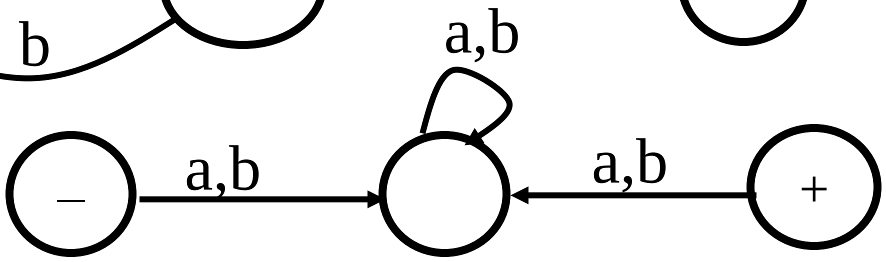


OR





OR



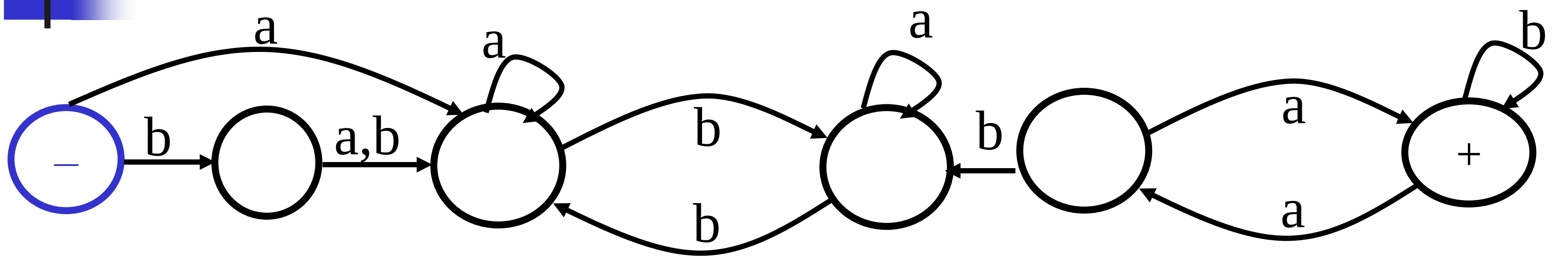
Then the finite automaton does not accept any word

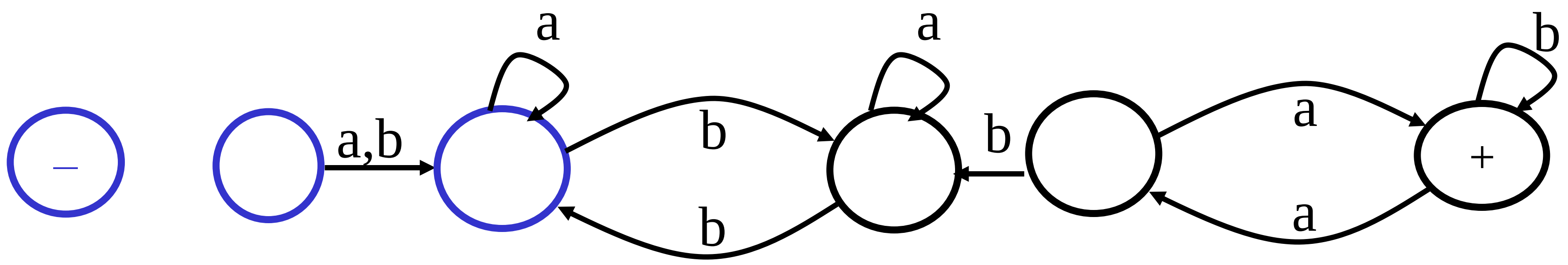
Method 2: Answer the question: is there any path from – to +?

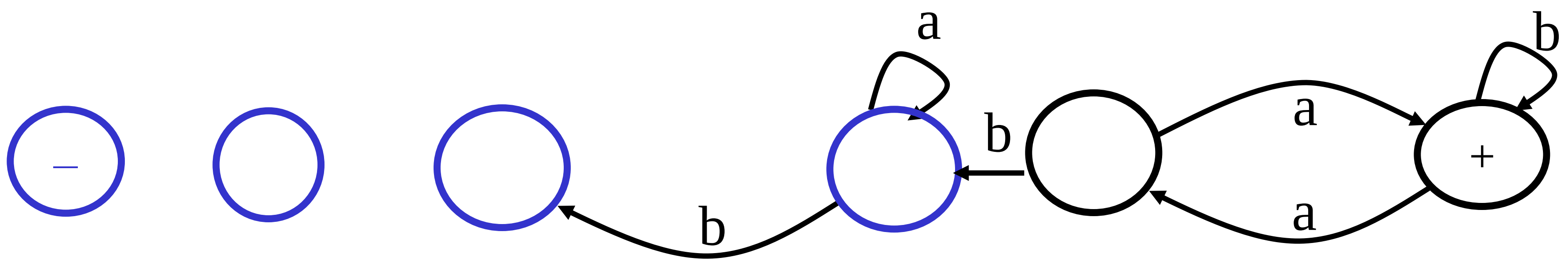
Algorithm:

1. Mark the start state. (Paint it blue.)
2. From every blue state, follow every arrow that goes out of the state, and paint the destination state blue. Delete each arrow that was followed.
3. Repeat step 2 until there are no new blue states.
4. If there is any final state that is blue, then there are words in the language accepted by the finite automaton. If not, the language is empty.

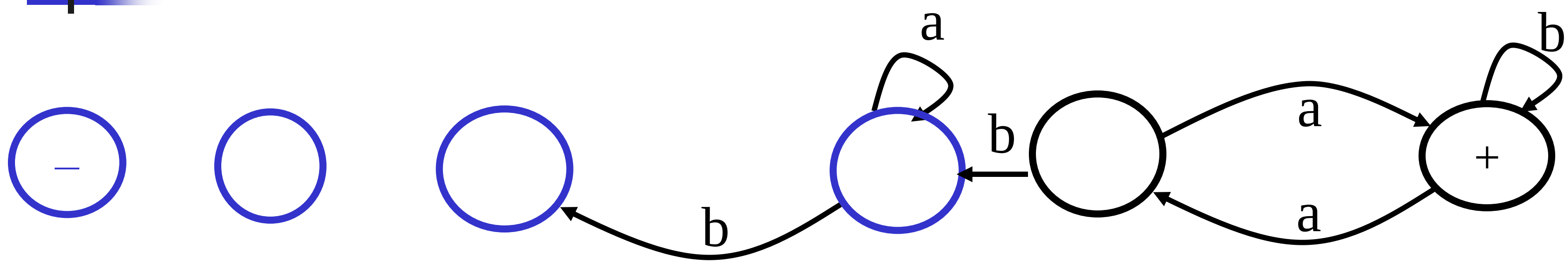
Chapter 11: Decidability

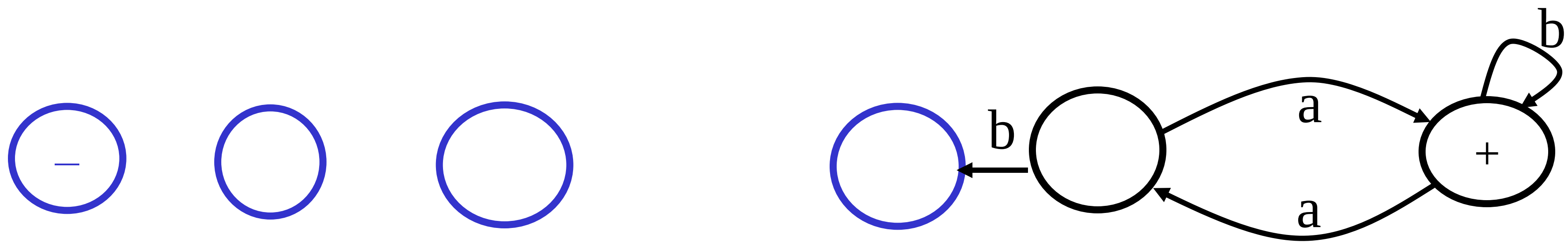






Chapter 11: Decidability

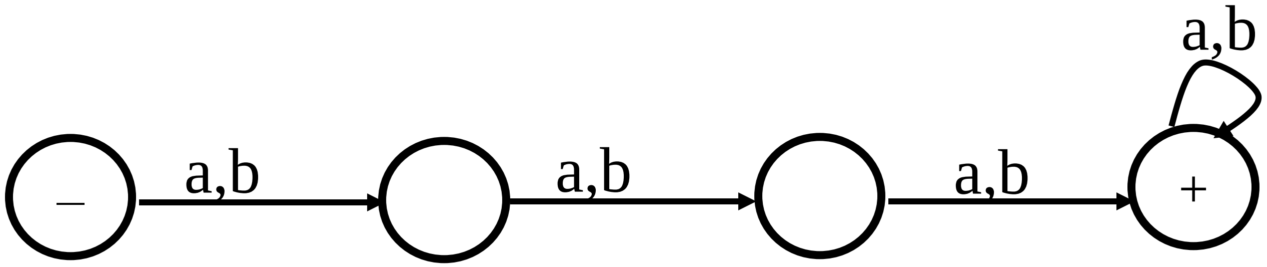




The language is empty.

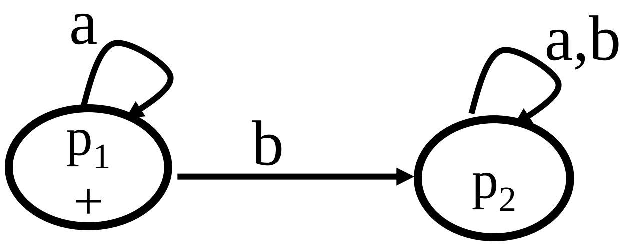
Theorem. Let F be a finite automaton with N states. If the language accepted by F is not empty, then it contains at least one word with N or fewer letters.

Method 3. Try all words with N and fewer letters. If the finite automaton accepts none of them, then the language accepted by this automaton is empty.

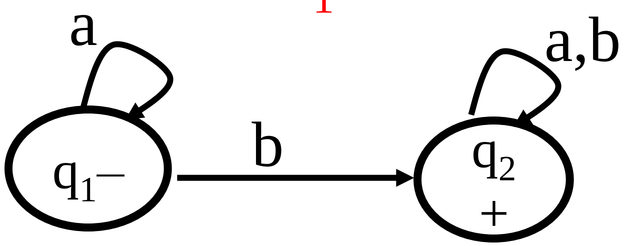


Chapter 11: Decidability

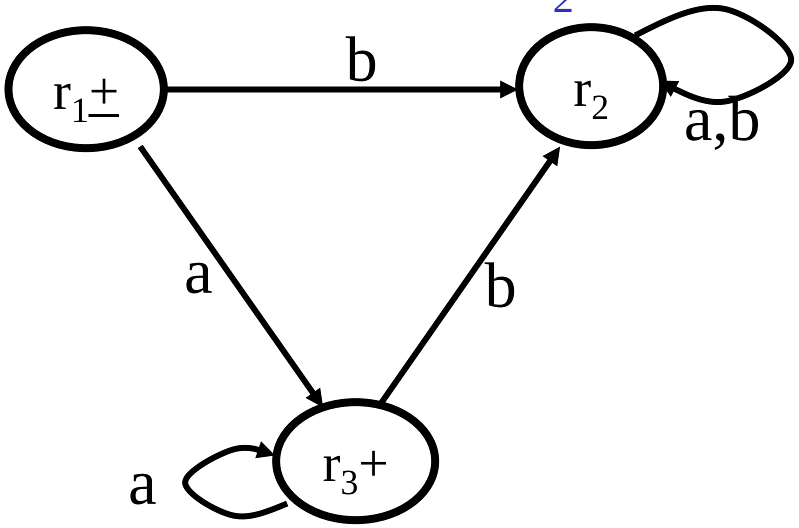
a* FA_1



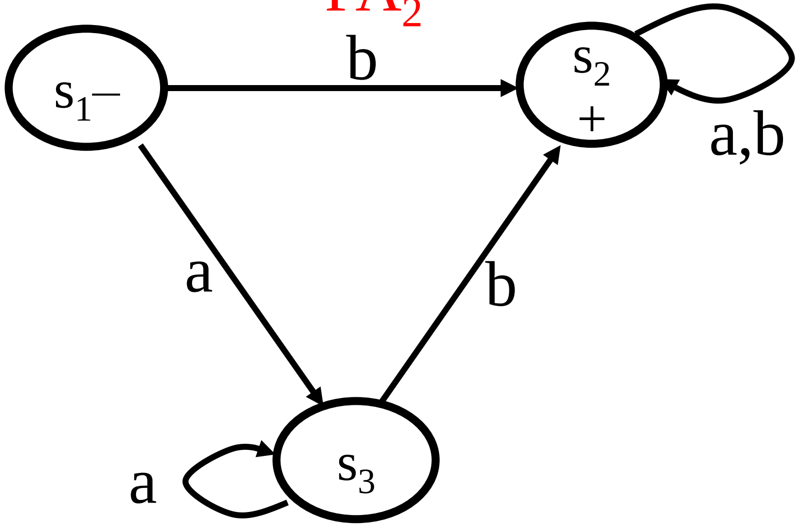
FA_1'



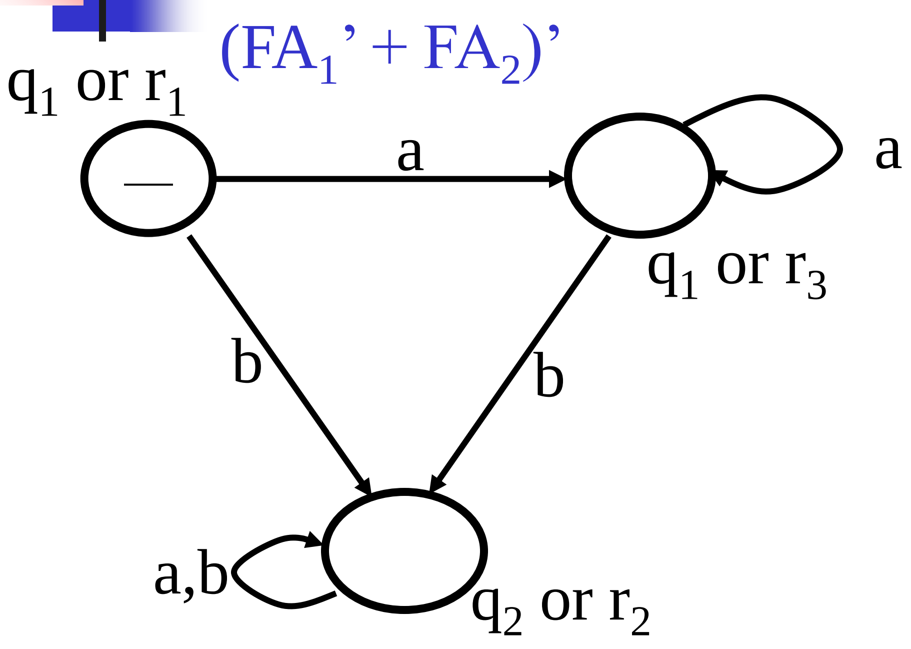
Λ+aa* FA_2

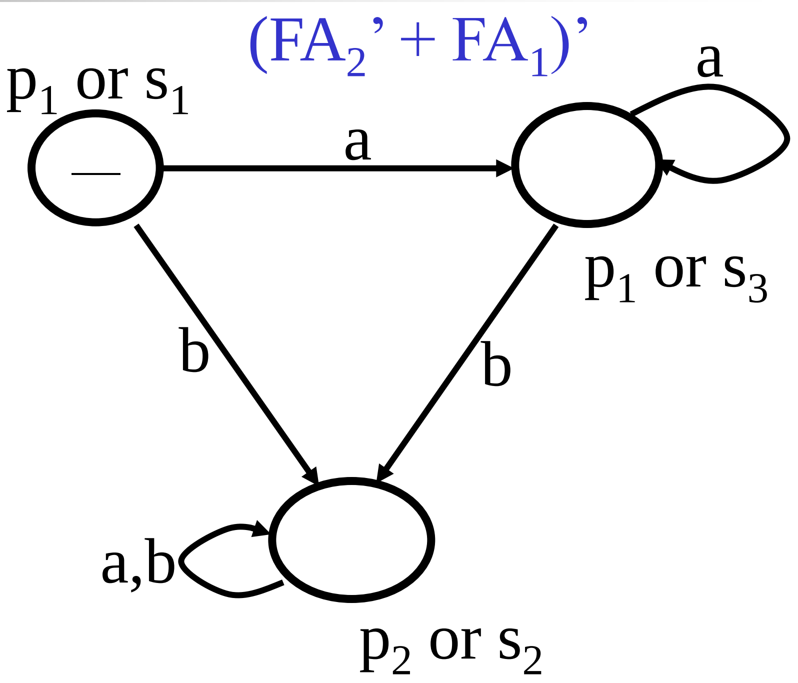


FA_2'



Chapter 11: Decidability





$(FA_1' + FA_2)' + (FA_2' + FA_1)'$?

FA_1 et FA_2 represente the same language

What if one of these two machines has a final state?

Problem 3. Does the language defined by a finite automaton have a finite number of words? Infinite? None?

none? algorithm from method 2.

finite or infinite?

Theorem. Let F be a finite automaton with N states.

1. If F accepts a word w such that $N \leq \text{length}(w) < 2N$, then the language accepted by F is infinite.
2. If the language accepted by F is infinite, then F accepts a word w such that $N \leq \text{length}(w) < 2N$.

Theorem. There is an algorithm that can decide whether a finite automaton accepts a finite or infinite language.

Algorithm:

It is enough to check all words of length between N and 2N. There is a finite number of tests to be done, and every check needs a finite number of steps.