Recall From Last Time

Defining a Model of Computation:

- How is the input organized?
- What computational operations are allowed?
- Do we have internal memory, and how much?

Some Formal Models of Computation:

- **Boolean:** (AND, OR, NOT, SLP's)
- **Formal Language Theory:** (starting today)
- **First-Order Logic:** ($\exists$, $\forall$)
- **Recursive Function Theory:** (Bloop)
- **Abstract RAM:** (as in an algorithms course)

Relations Among The Models:

Let's look at a single problem. Given n input bits, we want to know whether *exactly two* of them are ones. This question can be posed in each of our models:

- **Boolean:** There are various ways to build an SLP or circuit, which we'll explore on HW#1.
- **Finite-State Machine:** Sweep the input string left-to-right, remembering whether we've seen zero, one, two, or more than two ones.
- **First-Order Logic:**

$$\exists x : \exists y : \neg(x = y) \wedge \forall z : I(z) \leftrightarrow (z = x \vee z = y)$$

- **Numerical Input:** Is the input the sum of two distinct powers of two? On HW#1 you'll write a Bloop program to decide this.
- **Abstract RAM:** The problem probably defaults to one of the others once we decide on our data representation.

For the next few lectures we'll be looking at computational problems defined in terms of *strings*:

Definition: An **alphabet** is a non-empty finite set, e.g., $\Sigma = \{0, 1\}$, $\Gamma = \{a, b, c\}$, etc.

Definition: A **string** over an alphabet Σ is a finite sequence of zero or more symbols from Σ. The unique string with zero symbols is called ϵ. The set of all strings over Σ is called Σ^*.

Definition: A **language** over Σ is any subset of Σ^*. The decision problem for a language L is to input a string w and determine whether $w \in L$.

(Compare to the Java `String` type and the `charAt` method. In some ways, though, strings in formal language theory are more like files with only *sequential access*.)

In formal language theory we look at various kinds of *machines* that take a string as input, look at one letter at a time, and decide whether the string is in some language.

We also look at various formal ways to *specify* a language, such as regular expressions and context-free grammars, that are used in the real world.

In the 1950's and 1960's it was discovered that each of the most natural machine models corresponded to a specification system: the languages that could be decided by the machines were exactly those that could be specified in a certain way. Here we'll see some examples of that phenomenon.

Finally, we will always be interested in when a language *cannot be decided* by any machine in some class, or *cannot be specified* within some system. Such a result is called a *lower bound*, because we show that some particular amount of resources is insufficient.

Recall Definitions: **alphabet**, **string**, **language**.

Definition: The set of **regular expressions** $\mathbf{R}(\Sigma)$ over alphabet Σ is the smallest set of strings such that:

1. if $a \in \Sigma$ then $a \in \mathbf{R}(\Sigma)$
2. $\epsilon \in \mathbf{R}(\Sigma)$
3. $\emptyset \in \mathbf{R}(\Sigma)$
4. if $e, f \in \mathbf{R}(\Sigma)$ then so are the following:
   (a) $(e \cup f)$
   (b) $(e \circ f)$
   (c) $(e^\star)$

So far this defines only the *syntax*, the set of strings over the larger alphabet (Σ, operators, and punctuation) that *denote* regular languages over Σ.

Conventions: Omit $\circ$, use hierarchy of operations with * before $\circ$ before $\cup$. Think of addition ($\cup$), multiplication ($\circ$), and exponentiation (*).

Examples:

- $e_1 = 0^\star \in R(\{0,1\})$
- $e_2 = ((a \cup b) \circ (a \cup b))^\star \in R(\{a,b\})$
- $e_3 = a^\star(ba^\star ba^\star)^\star \in R(\{a,b,c\})$

Meanings:

- $\mathcal{L}(0^\star) \;=\; \{\epsilon, 0, 00, 0^3, 0^4, \ldots\} \;=\; \{0^i \mid i \in \mathbf{N}\}$
- $\mathcal{L}((a \cup b)^{2\,\star}) \quad = \quad \{w \in \{a,b\}^\star \mid |w| \equiv 0 \pmod 2\}$
- $\mathcal{L}(a^\star(ba^\star ba^\star)^\star) \quad = \quad \{w \in \{a,b\}^\star \mid \#_b(w) \equiv 0 \pmod 2\}$

Recall the meaning of Kleene star: For any set A,

$$
\begin{aligned}
A^\star &\equiv \bigcup_{i=0}^{\infty} A^i \\
&\equiv A^0 \cup A^1 \cup A^2 \cup \cdots \\
&\equiv \{\epsilon\} \cup A \cup \{xy \mid x, y \in A\} \cup \cdots \\
&\equiv \{x_1 x_2 \ldots x_n \mid n \in \mathbf{N}; x_1, \ldots, x_n \in A\}
\end{aligned}
$$

Meaning of a Regular Expression:

(A *recursive* definition of the mapping $\mathcal{L}$ from expressions to languages.)

1. if $a \in \Sigma$ then $a \in R(\Sigma)$; $\mathcal{L}(a) = \{a\}$
2. $\epsilon \in \mathbf{R}(\Sigma)$; $\mathcal{L}(\epsilon) = \{\epsilon\}$
3. $\emptyset \in \mathbf{R}(\Sigma)$; $\mathcal{L}(\emptyset) = \emptyset$
4. if $e, f \in R(\Sigma)$ then so are $(e \cup f)$, $(e \circ f)$, $(e^\star)$:

$$\begin{aligned}
\mathcal{L}(e \cup f) &= \mathcal{L}(e) \cup \mathcal{L}(f) \\
\mathcal{L}(e \circ f) &= \mathcal{L}(e)\mathcal{L}(f) = \{uv \mid u \in \mathcal{L}(e), v \in \mathcal{L}(f)\} \\
\mathcal{L}(e^\star) &= (\mathcal{L}(e))^\star
\end{aligned}$$

Definition 2.1 $A \subseteq \Sigma^\star$ is **regular** iff

$$(\exists e \in R(\Sigma))(A = \mathcal{L}(e))$$ ♠

In other words, a set, A, is regular iff there exists a regular expression that denotes it.

Definition: A **deterministic finite automaton (DFA)** is a tuple,

$$D = (Q, \Sigma, \delta, s, F)$$

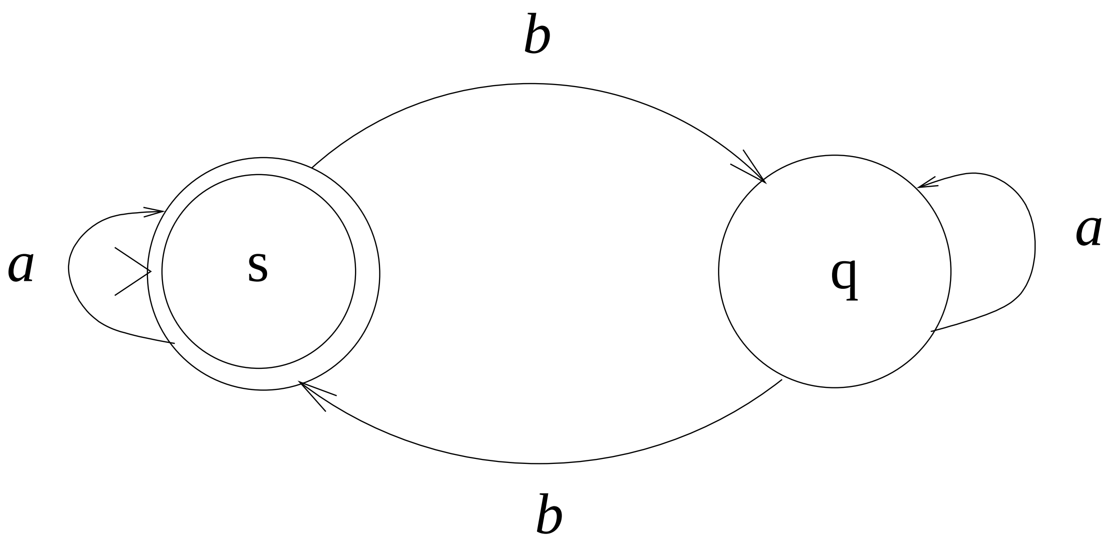
- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $s \in Q$ is the start state, and
- $F \subseteq Q$ is the set of final or accept states.

A DFA executes the following pseudo-Java algorithm:

```
public boolean isAccepted (String w) {
   State s = startState;
   for (int i=0; i < w.length(); i++)
      s = delta(s, w.charAt(i));
   return isFinalState(s);}
```

$$D_1 \; = \; (\{s, q\}, \{a, b\}, \delta_1, s, \{s\})$$

$$\delta_1 \; = \; \{\langle\langle s, a\rangle, s\rangle, \langle\langle s, b\rangle, q\rangle, \langle\langle q, a\rangle, q\rangle, \langle\langle q, b\rangle, s\rangle\}$$



$a \quad a \quad b \quad b \quad a \quad b \quad a \quad a \quad b \quad a$

s

$$\mathcal{L}_1 \; = \; \mathcal{L}(D_1) \; = \; \{w \in \{a, b\}^\star \mid \#_b(w) \equiv 0 \pmod 2\}$$

$$\mathcal{L}_1 \quad = \quad \mathcal{L}(a^\star(ba^\star ba^\star)^\star)$$

A DFA is an abstraction of any algorithm that:

- inputs a string (or text file)
- reads one letter at a time
- reads the input left to right, only once
- has only $O(1)$ bits of internal memory

We will be interested in the following results about DFA's and regular languages.

- **Kleene's Theorem:** A language is decided by some DFA iff it is regular.
- **Myhill-Nerode Theorem:** There is a *minimal* DFA for any regular language, definable in terms of a purely language-theoretic property.
- **Non-Regularity Proofs:** If a language is not regular, we can usually prove that fact.

To prove Kleene's Theorem it is convenient to introduce a new, *artificial* model of computation:

Definition: A **nondeterministic finite automaton (NFA)** is a tuple,

$$N = (Q, \Sigma, \Delta, s, F)$$

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\Delta : (Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$ is the transition function,
- $s \in Q$ is the start state, and
- $F \subseteq Q$ is the set of final or accept states.

$$\mathcal{L}(N) \quad = \quad \{w \mid s \underset{w}{\overset{\star}{\rightarrow}} q \in F\}$$

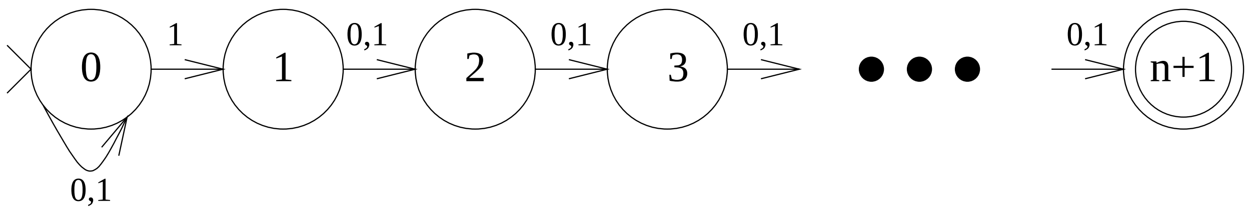
Recall that $\wp(S)$, the power set of S, is $\{A \mid A \subseteq S\}$.

So $\Delta(q, a)$ is the *set* of states to which N *might* go if it reads a when in state q. There might be zero, one, or more than one.

Example:

$$N_n = (\{q_0, \ldots, q_{n+1}\}, \{0,1\}, \Delta_n, q_0, \{q_{n+1}\})$$

$$\Delta_n = \{\langle\langle q_0, 0\rangle, \{q_0\}\rangle, \langle\langle q_0, 1\rangle, \{q_0, q_1\}\rangle, \ldots, \langle\langle q_n, 1\rangle, \{q_{n+1}\}\rangle\}$$



This NFA *might* accept a string w if it is in $\Sigma^*1\Sigma^n$. It *cannot* accept a string that is not in this language.

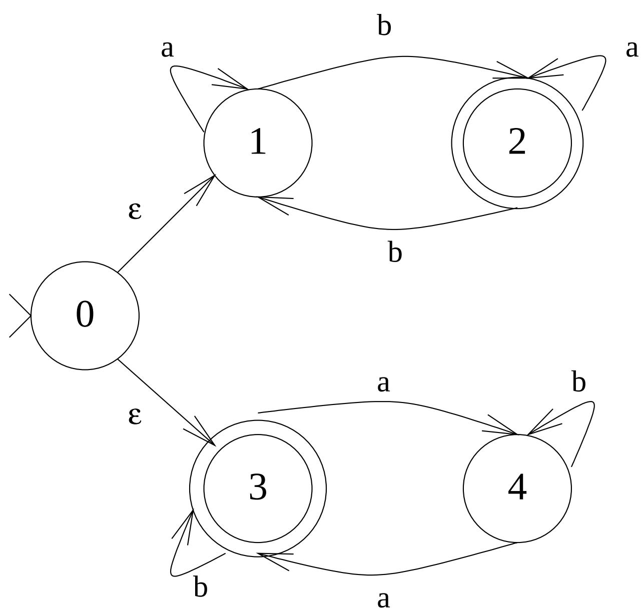
The natural DFA deciding $\mathcal{L}(N_n)$ has 2^{n+1} states. We'll see next time that no DFA with fewer than this many states can decide this language.

Proposition 2.2 *Every NFA N can be translated into an NFA without ϵ-transitions N' such that $\mathcal{L}(N) = \mathcal{L}(N')$.*

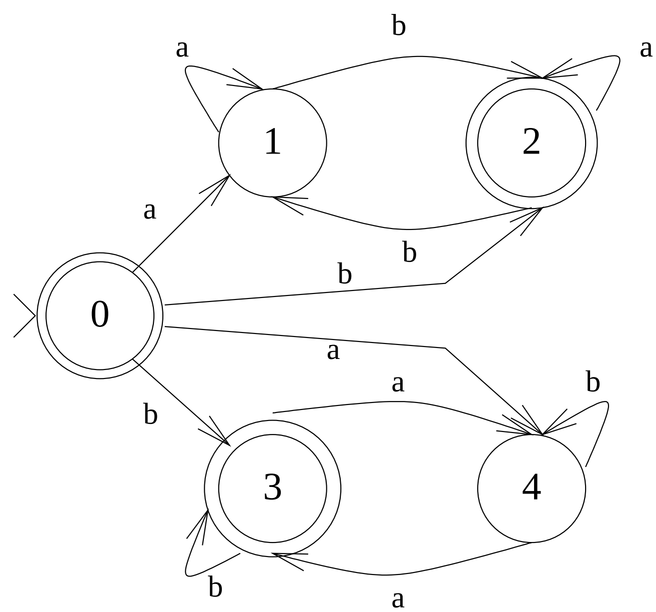
Proof: Given $N = (Q, \Sigma, \Delta, q_0, F)$, let $N' = (Q, \Sigma, \Delta', q_0, F')$ where

$$\Delta'(q,a) \;=\; \{r \mid (\exists s,t) q \overset{\epsilon^\star}{\rightarrow} s \overset{a}{\rightarrow} t \overset{\epsilon^\star}{\rightarrow} r\}$$

$$\mathbf{F}' \;=\; \{q \mid (\exists s \in F) q \overset{\epsilon^\star}{\rightarrow} s\}$$



N



N' ♠

Notation: For a DFA, $D = (Q, \Sigma, \delta, s, F)$, let $\delta^\star(q, w)$ be the state that D will be in after reading string w, when started in q,

$$\delta^\star(q, \epsilon) \equiv q$$
$$\delta^\star(q, wa) \equiv \delta(\delta^\star(q, w), a)$$

$$\mathcal{L}(D) \equiv \{w \mid \delta^\star(s, w) \in F\}$$

For an NFA without ϵ transitions, $N = (Q, \Sigma, \Delta, s, F)$, let $\Delta^\star(q, w)$ be the set of states that N can be in after reading string w, when started in q,

$$\begin{aligned} \Delta^\star(q, \epsilon) &:= \{q\} \\ \Delta^\star(q, wa) &:= \bigcup_{r \in \Delta^\star(q,w)} \Delta(r, a) \end{aligned}$$

$$\mathcal{L}(N) \equiv \{w \mid \Delta^\star(s, w) \cap F \neq \emptyset\}$$

Proposition 2.3 *For every NFA, N, with n states, there is a DFA, D, with at most 2^n states s.t. $\mathcal{L}(D) = \mathcal{L}(N)$.*

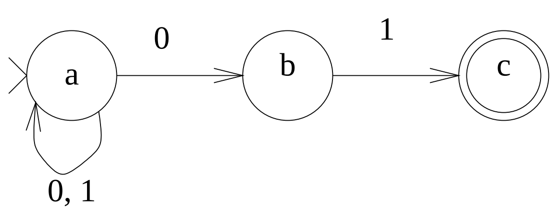
Proof: Let $N = (Q, \Sigma, \Delta, q_0, F)$. By Proposition 2.2 may assume that N has no ϵ transitions.

Let $D = (\wp(Q), \Sigma, \delta, \{q_0\}, F')$

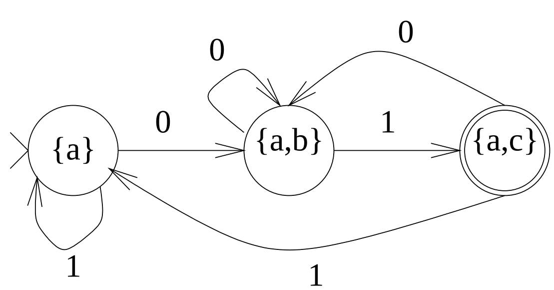
$$\delta(S, a) = \bigcup_{r \in S} \Delta(r, a)$$

$$F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$

N



D



Claim: For all $w \in \Sigma^\star$,

$$\delta^\star(\{q_0\}, w) \quad = \quad \Delta^\star(q_0, w)$$

By induction on $|w|$:

$|w| = 0$: $\delta^\star(\{q_0\}, \epsilon) \ = \ \{q_0\} \ = \ \Delta^\star(q_0, \epsilon)$

$|w| = k+1$: $w = ua$.

Inductively, $\delta^\star(\{q_0\}, u) \ = \ \Delta^\star(q_0, u)$

$$\begin{aligned}
\delta^\star(\{q_0\}, ua) &= \delta(\delta^\star(\{q_0\}, u), a) \\
&= \bigcup_{r \in \delta^\star(\{q_0\}, u)} \Delta(r, a) \\
&= \bigcup_{r \in \Delta^\star(q_0, u)} \Delta(r, a) \\
&= \Delta^\star(q, ua)
\end{aligned}$$

Therefore, $\mathcal{L}(D) = \mathcal{L}(N)$. ♠

Example: N_n had $n+2$ states but its equivalent DFA has 2^{n+1}. This is because every *reachable* state of the DFA is a set containing the start state of the N_n, so only half of the possible sets are reachable.

Theorem 2.4 (Kleene's Theorem) *Let* $A \subseteq \Sigma^\star$ *be any language. Then the following are equivalent:*

1. $A = \mathcal{L}(D)$, *for some DFA* D.
2. $A = \mathcal{L}(N)$, *for some NFA* N *with no* ϵ*-transitions*
3. $A = \mathcal{L}(N)$, *for some NFA* N.
4. $A = \mathcal{L}(e)$, *for some regular expression* e.
5. A *is regular.*

Proof: Obvious that $1 \to 2 \to 3$.

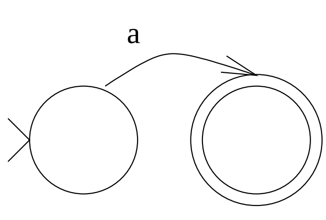
$3 \to 2$ by Prop. 1.2 (ϵ-elimination).

$2 \to 1$ by Prop. 1.3 (subset construction).

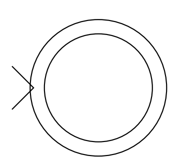
$4 \leftrightarrow 5$ by definition

$4 \to 3$: We show by induction on all regular expressions e that there is an NFA N with $\mathcal{L}(e) = \mathcal{L}(N)$:

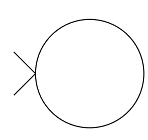
e* = *a



***e* =** ε

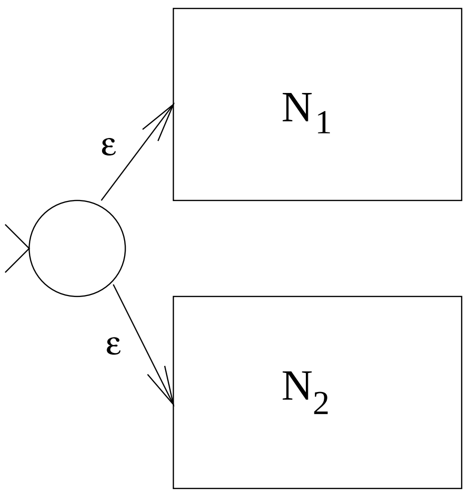


***e* =** ∅



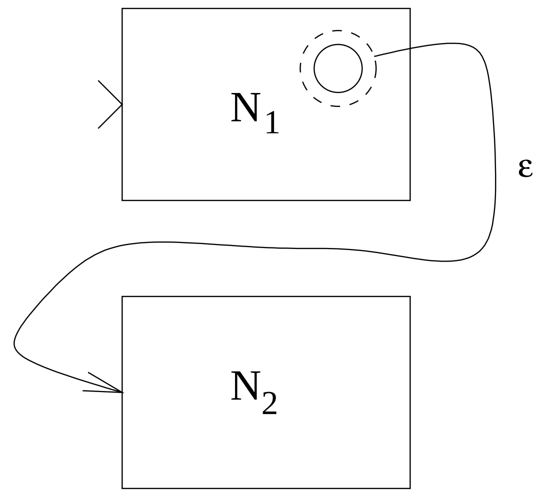
Union

$L(N) = L(N_1) + L(N_2)$



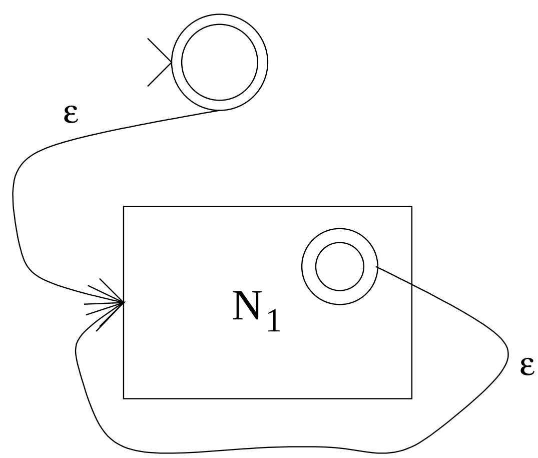
Concatenation

$L(N) = L(N_1)\ L(N_2)$



Kleene Star

$L(N) = (L(N_1))^*$



$3 \to 4$: (Cf. *state elimination* proof [S, Lemma 1.32])

Let $N = (\{1, \ldots, n\}, \Sigma, \Delta, 1, F)$, $F = \{f_1, \ldots, f_r\}$

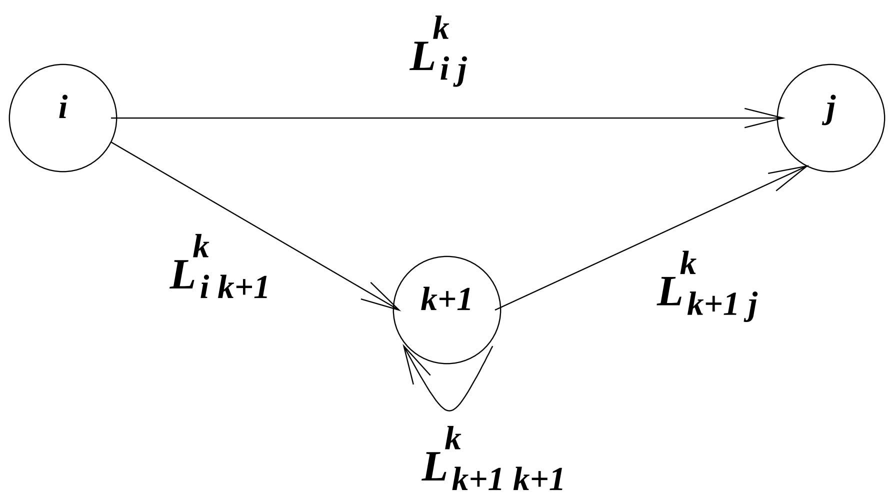
$$L^k_{ij} \;\equiv\; \{w \mid j \in \Delta^\star(i,w);\ \text{no intermediate state} \# > k\}$$

$$L^0_{ij} \;=\; \{a \mid j \in \Delta(i,a)\} \;\cup\; \{\epsilon \mid i = j\}$$

$$L^{k+1}_{ij} \;=\; L^k_{ij} \;\cup\; L^k_{i\,k+1}(L^k_{k+1\,k+1})^\star L^k_{k+1,j}$$

$$e \;=\; L^n_{1\,f_1} \cup \cdots \cup L^n_{1\,f_r}$$

$$\mathcal{L}(e) \;=\; \mathcal{L}(N)$$



♠