Organizatorics

- Alternative Lecture Dates:
  - Monday, 14-16 Monday 11-13 in HS001 ?
  - Thursday, 9-11 to
    - Wednesday, 14-16; [room available?] ?
    - Friday, 14-16, HS002 ?

- Tutorial dates:
  - No feedback on available rooms yet.

Embedded Systems: Definition

- Embedded systems are embedded in a physical environment and interact with it, *usually* for measuring or controlling purposes.
- Typical characteristics of embedded applications:
  - complex interaction with environment (non-deterministic, contineous behavior; signals arrive at enormeous rates; signal processing and information filtering become key issues)
  - high dependability requirements
  - often real-time processing required

Special Case: Real-Time Systems

- In a real-time system, the correctness not only depends on the logical results but also on the timing of the applications.

- Definition (Oxford Dictionary of Computing):
  Any system in which the time at which output is produced is significant.

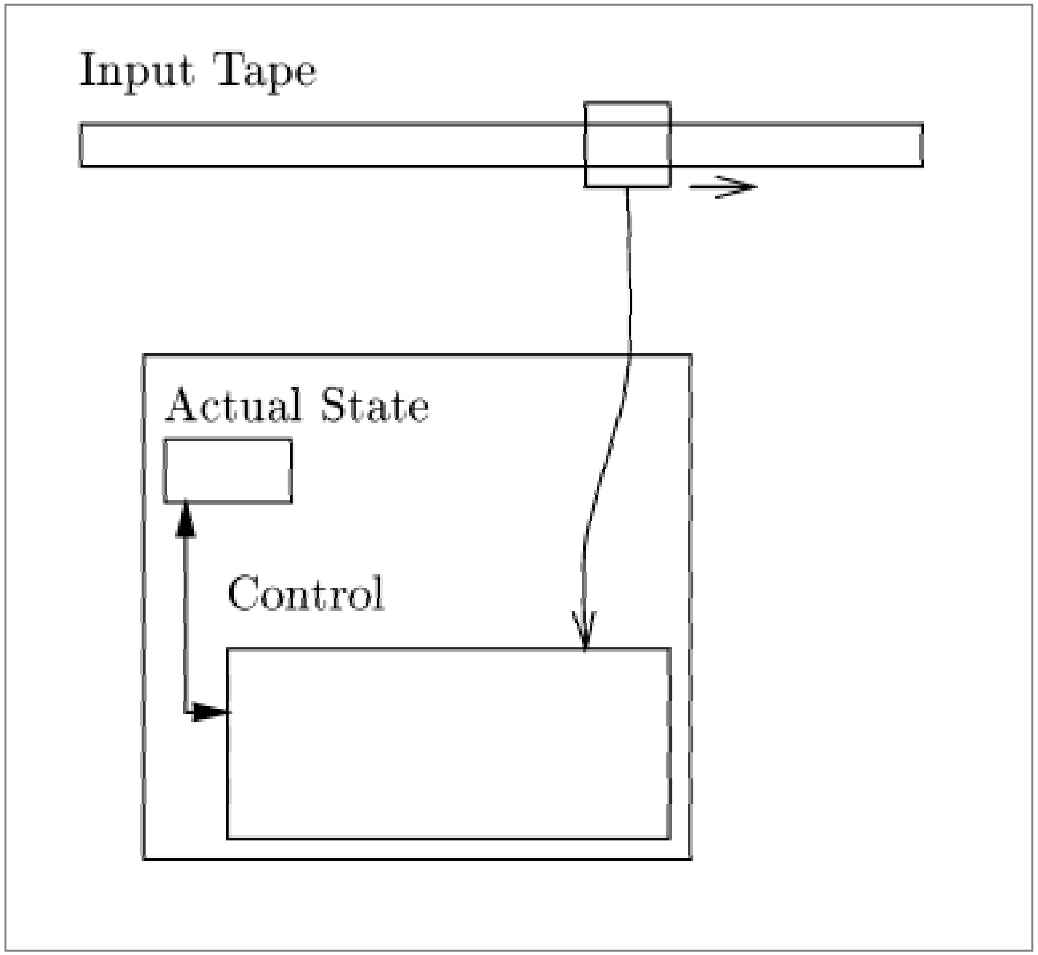
  This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.

Today

- Finite Automata

- Timed Automata

Finite Automata



Finite Automata

- Non-deterministic finite automaton (NFA):
  M = (Σ, Q, Δ, q_0, F) where
  - Σ: finite alphabet
  - Q: finite set of states
  - $q_0 \in$ Q: initial state
  - F $\subseteq$ Q: final states

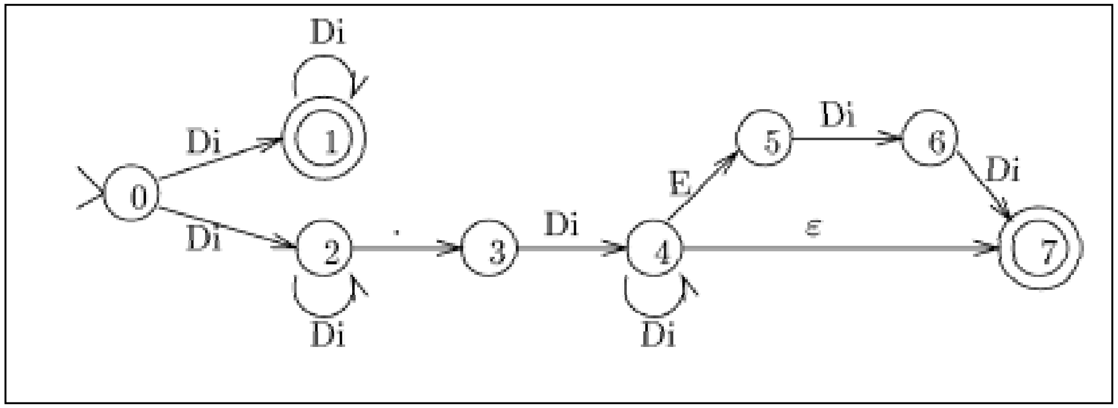
$$\bar{\Delta} \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$$

- M is called a deterministic finite automaton, if Δ is a partial function

$$\delta : Q \times \Sigma \rightarrow Q$$

Simple State Transition Diagram

- Used to represent a finite automaton
- Nodes: states
- q_0 has special entry mark
- Final states are doubly circled
- An edge from *p* to *q* is labelled by *a* if $(p,a,q) \in \Delta$
- Example: integer and real constants:



Language Accepted by an Automaton

- M = $(\Sigma, Q, \Delta, q_0, F)$
- For $q \in Q$, $w \in \Sigma^*$: (q,w) is a configuration.
- Step binary relation on configurations:
  $(q,aw) \vdash_M (p,w)$ iff $(q,a,p) \in \Delta : Q \times \Sigma \rightarrow Q$
- Reflexive transitive closure of $\vdash_M$ is denoted by $\vdash_M^*$
- Language accepted by M:

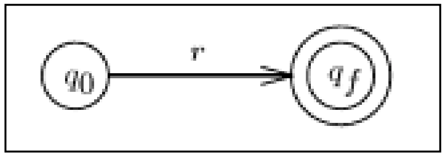
$$L(M) = \{w \mid w \in \Sigma^* \mid \exists q_f \in F : (q_0, w) \vdash_M^* (q_f, \varepsilon)\}$$

Regular Languages / Expressions

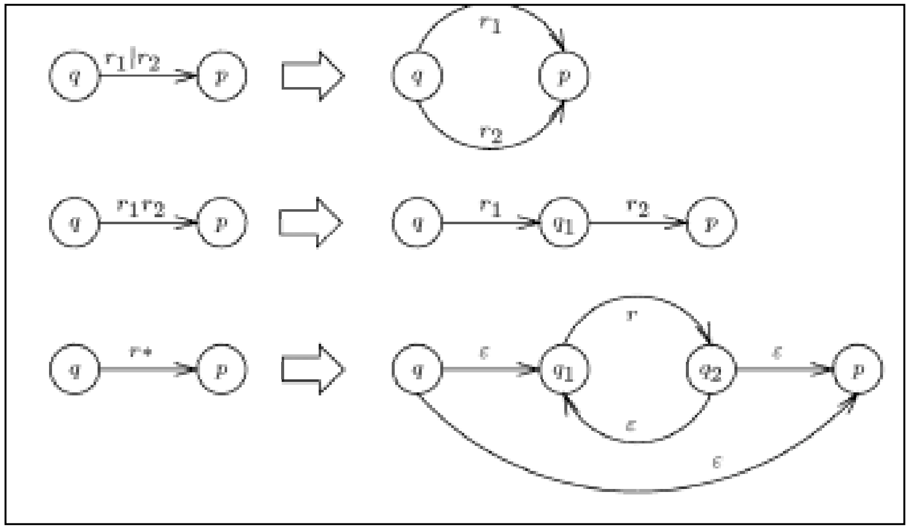
- Let Σ be an alphabet. The regular languages are defined inductively over Σ by:
  - $\emptyset$, $\{\varepsilon\}$ are regular languages over Σ
  - Forall $a \in \Sigma$, $\{a\}$ is a regular language
  - If R_1 and R_2 are regular languages over Σ, then so are $R_1 \cup R_2$, R_1R_2 and R_1^*.
- Regular expressions over Σ are defined by:
  - $\emptyset$ is a regular expression and describes the language $\emptyset$
  - ε is a regular expression and describes the language $\{\varepsilon\}$
  - a (for $a \in \Sigma$) is a regular expression and describes the regular language $\{a\}$
  - $(r_1|r_2)$ is a regular expression over Σ and describes the regular language $R_1 \cup R_2$
  - (r_1r_2) is a regular expression over Σ and describes the regular language R_1R_2
  - $(r_1)^*$ is a regular expression over Σ and describes the regular language R_1^*.

Regular Expressions and FA

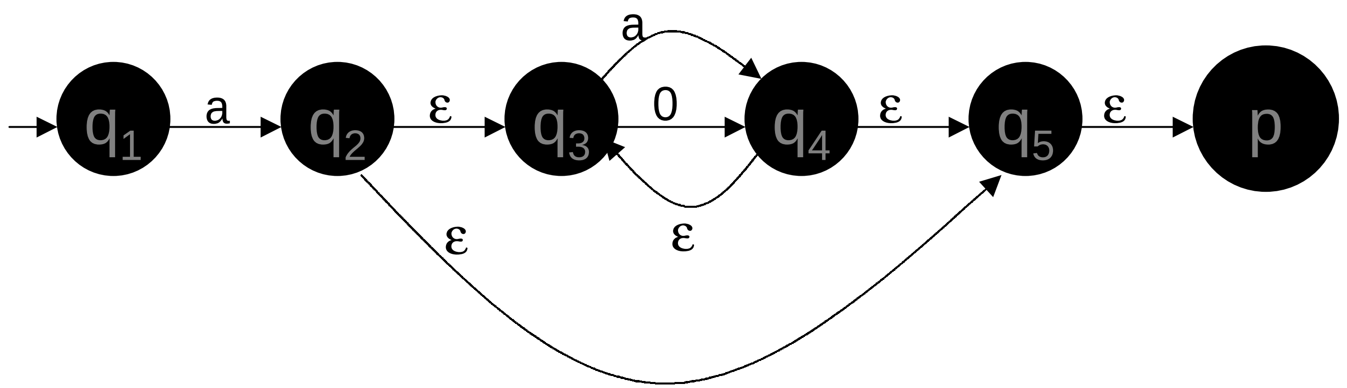
- For every regular language R, there exists an NFA M, such that L(M)=R.
- Constructive Proof (Subset Construction):
  - A regular language is defined by a regular expression r
  - Construct an NFA with one final state, q_f and the transition



  - Decompose r and develop the NFA according to the following rules -> until only transitions under single characters and ε remain.



Example: a(a|0)*


$$\begin{tikzcd}
\arrow[r] & q_1 \arrow[r, "a"] & q_2 \arrow[r, "\varepsilon"] \arrow[rrr, bend right, "\varepsilon"'] & q_3 \arrow[r, "0"] \arrow[r, bend left, "a"] & q_4 \arrow[r, "\varepsilon"] \arrow[l, bend left, "\varepsilon"] & q_5 \arrow[r, "\varepsilon"] & p
\end{tikzcd}$$

Nondeterminism

- Sources of nondeterminism:
  - many transitions may be possible under the same character in a given state
  - ε-moves (next character is not read) may compete with non- ε-moves
- DFA:
  - No ε-transition
  - At most one transition from every state under a given character, ie for every $q \in Q$, $a \in \Sigma$:

$$|\{q' \mid (q,a,q') \in \Delta\}| \leq 1$$

NFA -> DFA

- Let M=(Σ, Q, Δ, q_0, F) be an NFA and let q $\in$ Q. The set of ε successor states of q, ε-SS, is

$$\varepsilon - SS(q) = \{ p \mid (q,\varepsilon) \vdash_M^* (p,\varepsilon) \}$$

or the set of all states p, including q, for which there exists an ε path from q to p in the transition diagram for M.
We extend ε-SS to sets of states S $\subseteq$ Q:

$$\varepsilon - SS(S) = \bigcup_{q \in S} \varepsilon - SS(q)$$

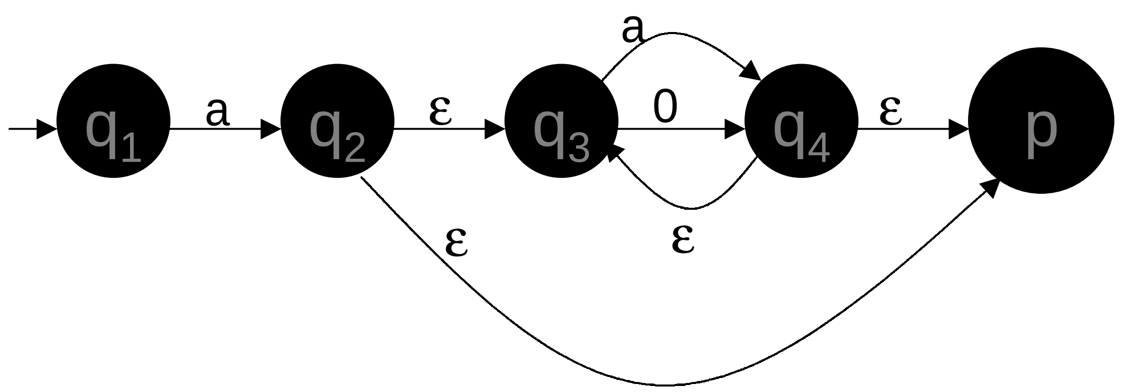
NFA -> DFA

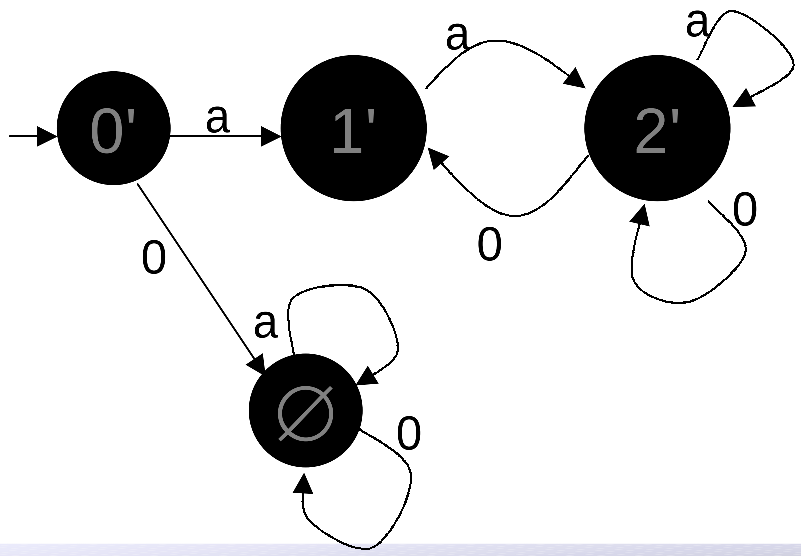
- If a language L is accepted by a NFA then there is also a DFA accepting L.
- Let M=(Σ, Q, Δ, q_0, F) be an NFA. The DFA associated with M, M'=(Σ, Q', δ, q_0', F') is defined by:
  - Q' $\subseteq$ P(Q)
  - q_0' = ε-SS(q_0)
  - $F' = \{S \subseteq Q \mid S \cap Q \neq \emptyset\}$
  - $\delta(S,a) = \varepsilon - SS(\{p \mid (q,a,p) \in \Delta \text{ for } q \in S\}) \text{ for } a \in \Sigma$
- Thus, the successor state S under a character *a* in *M'* is obtained by combining the sucessor states of all states *q* ∈ *S* under *a* and adding the ε sucessor states.

Algorithm NFA->DFA

```
$q'_0$:= $\varepsilon$-SSQ($q_0$); Q':={$q_{0'}$};
marked($q'_0$):=false; $\delta$ :=$\emptyset$;
while $\exists$ S$\in$ Q' and marked(S)=false do
    marked(S):=true;
    foreach a$\in \Sigma$ do
        T:= $\varepsilon$-SSQ({p$\in$ Q| (q,a,p) $\in$ $\Delta$ and q $\in$ S});
        if  T$\notin$ Q
            Q':=Q' $\cup$ {T}; // new state
            marked(T):=false
        $\delta$ := $\delta$ $\cup$ {(S,a)->T}; // new transition
```

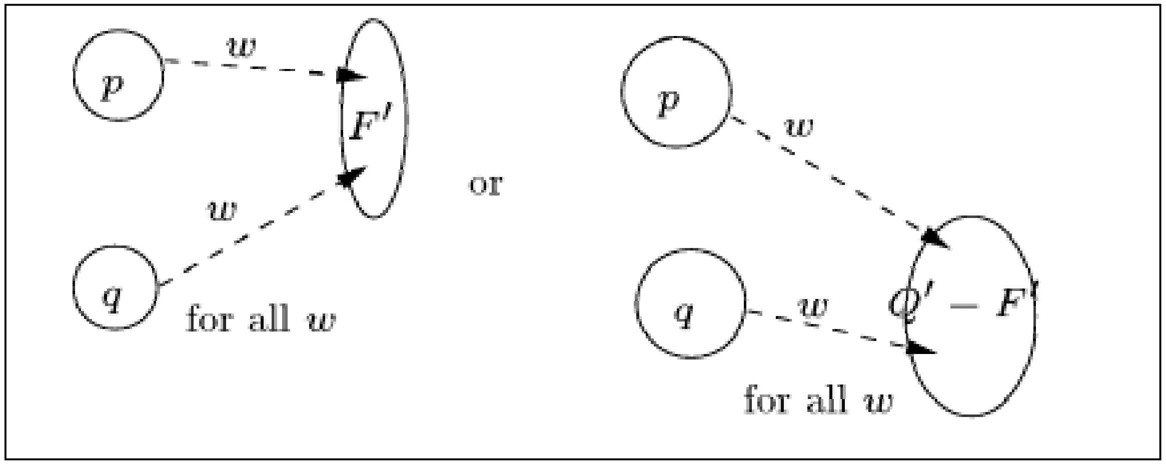
Example: a(a|0)*





DFA Minimization

- After NFA->DFA the DFA need not have minimal size, ie minimal number of states and transitions.
- p and q are undistinguishable, iff for all words w (q,w) and (p,w) lead by $\vdash_M^*$ into either F' or Q'-F'.
- Undistinguishable states can be merged.



DFA Minimization

- Input: DFA $M=(\Sigma, Q, \delta, q_0, F)$
- Output: DFA $M_{min}=(\Sigma, Q_{min}, \delta_{min}, q_{0min}, F_{min})$ with $L(M)=L(M_{min})$ and Q_{min} minimal.
- Iteratively refine a partition of the set of states where each set S in the partition consists of states so far undistinguishable
- Start with the partition $\Pi=\{F, Q-F\}$
- Refine the current Π by splitting sets $S\in\Pi$ into S_1, S_2 if there exist $q_1, q_2 \in S$ such that
  - $\delta(q_1,a) \in S_1$
  - $\delta(q_2,a) \in S_2$
  - $S_1 \neq S_2$
- Merge sets of undistinguishable states into a single state.

Algorithm minDFA

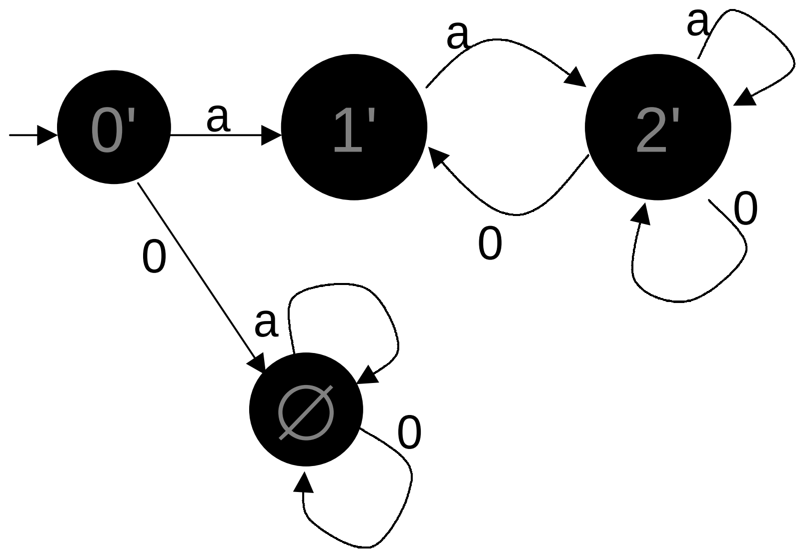
```
$\Pi$ := {F, Q-F}
do changed := false
    $\Pi' := \Pi$;
    foreach K in $\Pi$ do
        $\Pi' := (\Pi' - \{K\}) \cup \{\{K_i\}_{1 \le i \le n}\}$ with $K_i$ maximal such that
        $K = \bigcup_{1 \le i \le n} K_i$ and $\forall a \in \Sigma \, \forall q \in K_i \, \exists K_i' \in \Pi : \delta(q,a) \in K_i'$
    if $n > 1$ then changed := true fi
    $\Pi' := \Pi$;
until not changed;
$Q_{min} = \Pi$ - (Dead $\cup$ Unreachable);
```

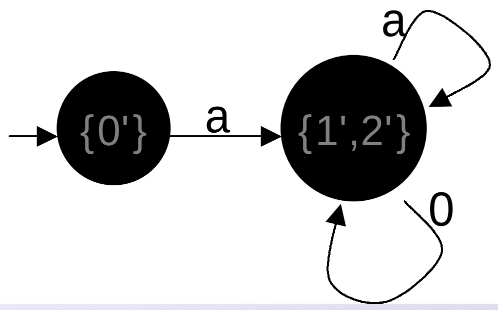
q_{0min}	Class of Π containing q_0
F_{min}	Classes containing an element of F

$\delta_{min}(K,a)=K'$ if $\delta(q,a)=p$ with $a \in \Sigma$ and $p \in K'$ for one (ie for all) $q \in K$

$K \in$ Dead	if K is not final state and contains only transitions to itself
K Unreachable	if there is no path from the initial state to K

Example: $a(a|0)^*$





Mealy Automata

- Mealy automata are finite-state machines that act as transducers, or translators, taking a string on an input alphabet and producing a string of equal length on an output alphabet.
- A machine in state q_j, after reading symbol sigmaσ_k writes symbol λ_k; the output symbol depends on the state just reached and the corresponding input symbol.
- A Mealy automaton is a six-tuple $M_E=(Q, \Sigma, \Gamma, \delta, \lambda, q_0)$ where
  - Q is a finite set of states
  - Σ is a finite input alphabet
  - Γ is a finite output alphabet
  - δ: $Q \times \Sigma$ -> Q is the transition function
  - λ: $Q \times \Sigma$ -> Γ is the output function
  - q_0 is the initial state

Moore Automata

- Moore automata are finite-state machines that act as transducers, or translators, taking a string on an input alphabet and producing a string of equal length on an output alphabet.
- Symbols are output after the transition to a new state is completed; output symbol depends only on the state just reached.
- A Moore automaton is a six-tuple $M_O=(Q, \Sigma, \Gamma, \delta, \lambda, q_0)$ where
  - Q is a finite set of states
  - Σ is a finite input alphabet
  - Γ is a finite output alphabet
  - $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
  - $\lambda: Q \rightarrow \Gamma$ is the output function
  - q_0 is the initial state

Timed Automata

- Modelling goal: reasoning about systems that must interact with physical processes and that are subject to real-time constraints. Important issue: Verification.
- Modelling basics: executions are infinite sequences of events paired with times, where the ith element of the time sequence gives the time of occurrence of the ith event.
- Finite automata: effective constructions and decision procedures for automatically manipulating and analyzing system behavior.
- Question: Can timed sequences be represented by some formal language and can the behavior be described by some formal automaton?

Time Models

- Linear time model:
  - Execution is only modelled as a sequence of states or events, called an execution trace.
  - The behavior of the system is a set of execution traces.
  - Can be modeled by finite automata.
  - No reasoning about the timing of events.

- Discrete time model:
  - Time sequence is monotonically increasing sequence of integers.
  - Can be modelled by finite automata after inserting silent events such that the times increase by exactly one at each step and, hence, the time sequence can be discarded.
  - Appropriate for synchronous digital systems, but accuracy with which physical systems can be modeled is limited (continuous time is approximated by discrete time).

Time Models

- Dense-time model:
  - Times of events are real numbers increasing monotonically without bound.
  - Problem: How to transform set of dense-time traces into ordinary formal language?

Timed Automata...

- ...accept timed words, ie infinite sequences in which a real-valued time of occurrence is associated with each symbol.
- ...are finite automata with a finite set of real-valued clocks:
  - Clocks can be independently reset to 0 with the transitions of the automaton.
  - Clocks keep track of time elapsed since last reset.
  - Transitions can put constraints on clock values: a transition may be taken only if the current values of the clocks satisfy the associated constraints.
    Example: channel delivers each message within 3-5 time units of its receipt.

ω-Automata

- Problem: timed sequences can be inifinite.
- Regular language: finite words over some finite alphabet.

- An ω-regular language over a finite alphabet Σ is a subset of Σ^{ω}, ie the set of all infinite words over Σ.

- ω-automaton: corresponds to NFA, but with acceptance condition modified to handle infinite input words.
  - Büchi automata
  - Muller automata

ω-Automata

- A transition table A is a tuple (Σ, Q, Q_0, E) where Σ is an input alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of start states and $E \subseteq Q \times \Sigma \times Q$ is a set of edges. If $(s,a,s') \in E$ then the automaton can change its state from s to s' reading the input symbol a.
- For a word $\sigma=\sigma_1\sigma_2\ldots$ over Σ,

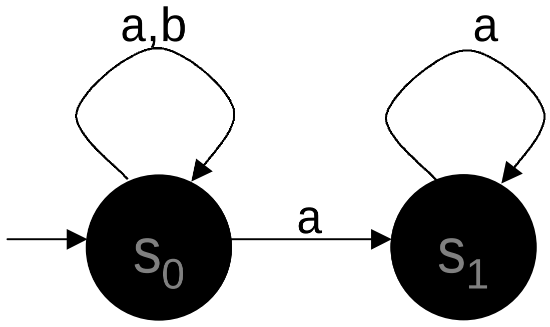
$$r : s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \xrightarrow{\sigma_3} \ldots$$

is a run of A over σ for $s_0 \in Q_0$ and $(s_{i-1}, \sigma_i, s_i) \in E$ for all $i \geq 1$.

- The set *inf(r)* for a run r consists of the states $s \in Q$ such that $s=s_i$ for infinitely many $i \geq 0$.

Büchi Automata

- A Büchi automaton A is a transition table (Σ, Q, Q_0, E) with an additional set $F \subseteq Q$ of accepting states. A run r of A over a word $\sigma \in \Sigma^\omega$ is an accepting run iff $\inf(r) \cap F \neq \emptyset$.
- The language L(A) accepted by A consists of the words $\sigma \in \Sigma^\omega$ such that A has an accepting run over σ.
- Example:



The automaton accepts all words with only a finite number of b's: $L_0=(a|b)^*a^\omega$

ω-Regular Languages

- An ω-language is called ω-regular, iff it is accepted by some Büchi automaton.

- Properties of ω-regular languages:
  - Closed under all Boolean operations
  - Intersection, complement decidable
  - Inclusion problem is decidable:
    - Testing for emptiness simple.
    - Complementing involves exponential blow-up in the states of the Büchi automaton: PSPACE-complete.
    - For deterministic automata inclusion problem can be solved in polynomial time.

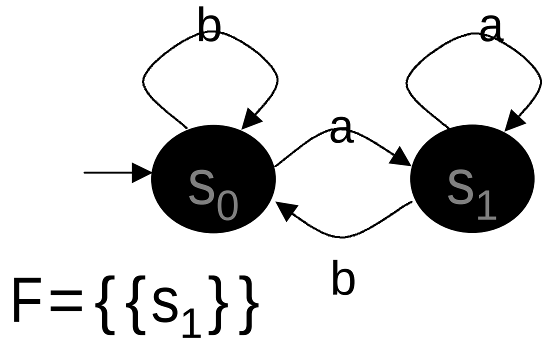
Deterministic Büchi Automata

- A transition table A is deterministic, iff
  - there is a single initial state
  - the number of a-labeled edges starting at s is at most one for all $s \in S$ and for all $a \in \Sigma$.

- The class of languages accepted by deterministic Büchi automata is strictly smaller than the class of ω-regular languages

Muller Automata

- A Muller automaton A is a transition table (Σ, Q, Q_0, E) with an acceptance family $F \subseteq 2^Q$. A run r of A over a word $\sigma \in \Sigma^\omega$ is an accepting run iff $\inf(r) \in F$.
- The class of languages accepted by Muller automata is the same as that accepted by Büchi automata, and also equals that accepted by deterministic Muller automata.
- The complement can be computed in polynomial time.
- Example:



The automaton accepts all words with only a finite number of b's:
$L_0=(a|b)^*a^\omega$

Timed Languages

- A time sequence $\tau=\tau_1\tau_2$... is an infinite sequence of time values $\tau_i \in R$ with $\tau_i>0$, satisfying the following constraints:
  - Monotonicity: τ increases strictly monotonically so that $\tau_i < \tau_{i+1}$ for all $i \geq 1$.
  - Progress: For every $t \in R$, there is some $i \geq 1$ such that $\tau_i > t$.
- A timed word over an alphabet Σ is a pair (σ,τ) where $\sigma=\sigma_1\sigma_2$... is an infinite word over Σ and τ is a time sequence. A timed language over Σ is a set of timed words over Σ.
- Viewed as an input to an automaton a timed word (σ,τ) presents the symbol σ at time τ.

Examples of Timed Languages

$$L_1 = \{((a \mid b)^\omega, \tau) \mid \forall i.((\tau_i > 5.6) \rightarrow (\sigma_i = a))\}$$

- Language consists of all timed words (σ,τ) such that there is no *b* after time 5.6.

$$L_2 = \{((ab)^\omega, \tau) \mid \forall i.((\tau_{2i} - \tau_{2i-1)}) < (\tau_{2i+2} - \tau_{2i+1}))\}$$

- Language consists of all timed words (σ,τ) in which *a* and *b* alternate and for successive pairs of *a* and *b* the time difference between *a* and *b* keeps increasing.