CS256/Spring 2008 — Lecture #11

Zohar Manna

Beyond Temporal Logics

Temporal logic expresses properties of infinite sequences of states, but there are interesting properties that cannot be expressed, e.g.,

"p is true only (at most) at even positions."

Questions (foundational/practical):

- What other languages can we use to express properties of sequences ($\Rightarrow$ properties of programs)?
- How do their expressive powers compare?
- How do their computational complexities (for the decision problems) compare?

ω-languages

Σ: nonempty finite set (alphabet) of characters

Σ^*: set of all finite strings of characters in Σ
finite word $w \in \Sigma^*$

Σ^ω: set of all infinite strings of characters in Σ
ω-word $w \in \Sigma^\omega$

(finitary) language: $\mathcal{L} \subseteq \Sigma^*$

ω-language: $\mathcal{L} \subseteq \Sigma^\omega$

States

Propositional LTL (PLTL) formulas are constructed from the following:

- propositions $p_1, p_2, \ldots, p_n$.
- boolean/temporal operators.
- a state $s \in \{f, t\}^n$
  i.e., every state s is a truth-value assignment to all n propositional variables.

`Example:`
If $n = 3$, then $$s: \ \langle p_1 : t,\ p_2 : f,\ p_3 : t \rangle$$ corresponds to state tft.
$p_1 \leftrightarrow p_2$ denotes the set of states $$\{fff, fft, ttf, ttt\}$$

- alphabet $\Sigma = \{f, t\}^n$
  i.e, 2^n strings, one string for every state.

Note: $\texttt{T}$, $\texttt{F}$ = formulas (syntax)
t, f = truth values (semantics)

Models of PLTL $\mapsto$ ω-languages

- A model of PLTL for the language with n propositions

$$\sigma : s_0, s_1, s_2, \ldots$$

can be viewed as an infinite string $s_0 s_1 s_2 \ldots$, i.e.,

$$\sigma \in (\{f, t\}^n)^\omega$$

- A PLTL formula φ denotes an ω-language

$$\mathcal{L} = \{\sigma \mid \sigma \vDash \varphi\} \subseteq (\{f, t\}^n)^\omega$$

`Example:`

If $n = 3$, then

$\varphi : \Box(p_1 \leftrightarrow p_2)$ denotes the ω-language

$$\mathcal{L}(\varphi) = \{f\,f\,f, f\,f\,t, t\,t\,f, t\,t\,t\}^\omega$$

Other Languages to Talk about Infinite Sequences

- $\boxed{\omega\text{-regular expressions}}$
- $\boxed{\omega\text{-automata}}$

Regular Expressions

Syntax:

$$r \ ::= \ \emptyset \ | \ \varepsilon \ | \ a \ | \ r_1 r_2 \ | \ r_1 + r_2 \ | \ r^*$$

$$(\varepsilon = \text{empty word}, \quad a \in \Sigma)$$

Semantics:

A regular expression r (on alphabet Σ) denotes a finitary language
$\mathcal{L}(r) \subseteq \Sigma^*$:

$$\begin{aligned}
\mathcal{L}(\emptyset) &= \emptyset \\
\mathcal{L}(\varepsilon) &= \{\varepsilon\} \\
\mathcal{L}(a) &= \{a\} \\
\mathcal{L}(r_1 r_2) &= \mathcal{L}(r_1) \cdot \mathcal{L}(r_2) = \\
&\quad \{xy \mid x \in \mathcal{L}(r_1), y \in \mathcal{L}(r_2)\} \\
\mathcal{L}(r_1 + r_2) &= \mathcal{L}(r_1) \cup \mathcal{L}(r_2) \\
\mathcal{L}(r^*) &= \mathcal{L}(r)^* = \{x_1 x_2 \cdots x_n \mid n \geq 0, \\
&\qquad\qquad x_1, x_2, \ldots, x_n \in \mathcal{L}(r)\}
\end{aligned}$$

ω-regular expressions

Syntax:

$$\omega r ::= r_1(s_1)^\omega \; + \; r_2(s_2)^\omega \; + \; \cdots \; + \; r_n(s_n)^\omega$$

$$n \geq 1, \;\; r_i, \, s_i = \text{regular expressions}$$

Semantics:

$$\begin{aligned} \mathcal{L}(rs^\omega) \;\; = \;\; \{ & xy_1y_2 \cdots \mid x \in \mathcal{L}(r), \\ & y_1, y_2, \ldots \in \mathcal{L}(s) \setminus \{\varepsilon\}\} \end{aligned}$$

rs^ω denotes all infinite strings with an initial prefix in $\mathcal{L}(r)$, followed by a concatenation of infinitely many nonempty words in $\mathcal{L}(s)$.

ω-regular expressions (cont.)

`Example:`

Take $A = \{a, b\}$. What languages do the following ω-r.e.'s denote?

$aa\ b^\omega$	ω-word starting with two a's, followed by b's
$a^*\ b^\omega$	all ω-words starting with a finite string of a's, followed by b's
$(a+b)^*\ b^\omega$	all ω-words with only finitely many a's
$((a+b)^*b)^\omega$	all ω-words containing infinitely many b's

PLTL (future) $\mapsto$ ω-r.e.'s

`Example:`

p is an abbreviation for $tt + tf$
q is an abbreviation for $tt + ft$
T is an abbreviation for $tt + tf + ft + ff$

$$\Downarrow$$

$\Box\, p$:	p^ω
$\Diamond\, q$:	$\mathrm{T}^* q\ \mathrm{T}^\omega$
$p \mathcal{U} q$:	$p^* q\ \mathrm{T}^\omega$
$p \Rightarrow \Box\, q$:	$(\neg p)^*\ q^\omega\ +\ (\neg p)^\omega$
$\Box \Diamond\, p$:	$(\mathrm{T}^* p)^\omega$
$\Diamond \Box\, q$:	$\mathrm{T}^*\ q^\omega$

Expressive Power

- Every PLTL formula has an equivalent ω-r.e.

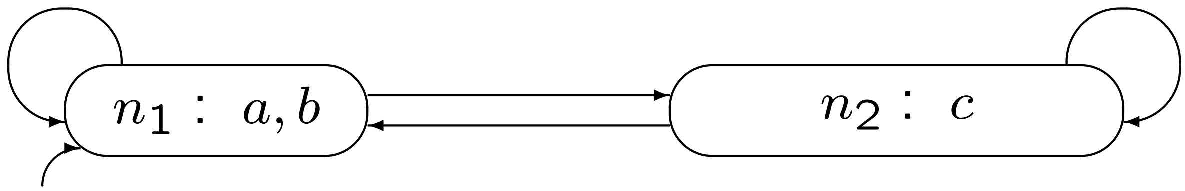
- PLTL is strictly weaker than ω-r.e.'s:

  "p is true only (at most) at even positions."

  - not expressible in PLTL (Pierre Wolper, 1983)
  - ω-r.e.: $(\top(\neg p))^\omega$

- ω-r.e.'s are equivalent to ω-automata.

Finite-State Automata



Finite alphabet Σ.

Automaton $\mathcal{A}$: $\langle N,\ N_0,\ E,\ \mu,\ F \rangle$, where

- N: nodes
- $N_0 \subseteq N$: initial nodes
- $E \subseteq N \times N$: edges
- $\mu : N \to 2^{\Sigma}$: node labeling function
- $F \subseteq N$: final nodes

Note: We label the nodes and not the edges.

Finite-State Automata (Cont'd)

Main question:
Given a string

$$\sigma\colon s_0 \ldots s_k$$

over Σ, is σ accepted by $\mathcal{A}$?

- path
  A sequence of nodes

  $$\pi\colon n_0, \ldots, n_k$$

  is a path of $\mathcal{A}$ if

  – $n_0 \in N_0$

  – for every $i\colon 0 \ldots k{-}1$, $\langle n_i, n_{i+1} \rangle \in E$.

Finite-State Automata (Cont'd)

- <u>trail</u>
  A path
  $$\pi\colon\ n_0, \ldots, n_k$$
  of $\mathcal{A}$ is a <u>trail of a string</u>
  $$\sigma\colon s_0, \ldots, s_k$$
  in $\mathcal{A}$ if for every $i\colon 0 \ldots k$,
  $$s_i \in \mu(n_i).$$

- <u>accepted</u>
  A string
  $$\sigma\colon s_0 \ldots s_k$$
  is <u>accepted</u> by $\mathcal{A}$ if it has a trail
  $$\pi\colon n_0, \ldots, n_k$$
  in $\mathcal{A}$ such that
  $$n_k \in F.$$

Finite-State Automata (Cont'd)

- $\underline{\mathcal{L}(\mathcal{A})}$
  The set of all strings ("languages") accepted by $\mathcal{A}$.

- <u>deterministic</u>
  An automaton $\mathcal{A}$ is called <u>deterministic</u> if every string has <u>exactly</u> one (not necessarily accepting) trail in $\mathcal{A}$.

- <u>total</u>
  An automaton $\mathcal{A}$ is called <u>total</u> if every string has <u>at least</u> one (not necessarily accepting) trail in $\mathcal{A}$.

Finite-State Automata: Decision Problems

- Emptiness:

  Is any string accepted?

$$\mathcal{L}(\mathcal{A}) \stackrel{?}{=} \emptyset$$

- Universality:

  Are all strings accepted?

$$\mathcal{L}(\mathcal{A}) \stackrel{?}{=} \Sigma^*$$

- Inclusion:

  Are all strings accepted by $\mathcal{A}_1$ accepted by $\mathcal{A}_2$?

$$\mathcal{L}(\mathcal{A}_1) \stackrel{?}{\subseteq} \mathcal{L}(\mathcal{A}_2)$$

Finite-State Automata: Operations

- Complementation: $\overline{\mathcal{A}}$

  $$\mathcal{L}(\overline{\mathcal{A}}) = \Sigma^* - \mathcal{L}(\mathcal{A})$$

- Product: $\mathcal{A}_1 \times \mathcal{A}_2$

  $$\mathcal{L}(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$$

- Union: $\mathcal{A}_1 + \mathcal{A}_2$

  $$\mathcal{L}(\mathcal{A}_1 + \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$$

Using complementation and product construction, we only need a decision procedure for emptiness to decide universality and inclusion:

- Universality:

  $$\mathcal{L}(\mathcal{A}) = \Sigma^* \iff \mathcal{L}(\overline{\mathcal{A}}) = \emptyset$$

- Inclusion:

  $$\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2) \iff \mathcal{L}(\mathcal{A}_1 \times \overline{\mathcal{A}_2}) = \emptyset$$

Finite-State Automata: Determinization

For every nondeterministic automaton $\mathcal{A}_N$, there exists a deterministic automaton $\mathcal{A}_D$ such that

$$\mathcal{L}(\mathcal{A}_N) = \mathcal{L}(\mathcal{A}_D).$$

(May cause exponential blowup in size.)

$\boxed{\omega\text{-Automata}}$

Finite-state automata over <u>infinite strings</u>.

Main question:
Given an <u>infinite</u> sequence of <u>states</u>

$$\sigma : s_0, s_1, s_2, \ldots$$

is σ accepted by $\mathcal{A}$?

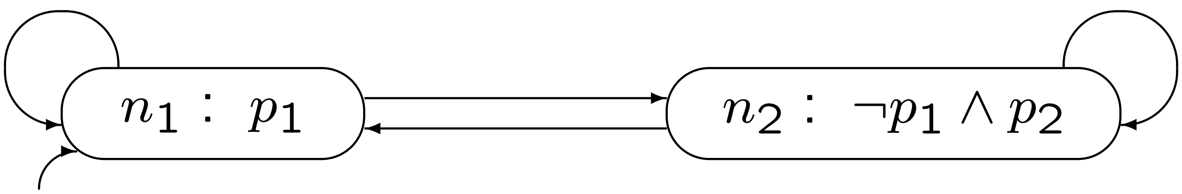
<u>Additional references:</u>

- Section 5 of Wolfgang Thomas: "Languages, Automata, and Logic". In G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, V. III. (Tech Report version available on the web), pp. 389–455, 1997.

- Part I of Wolfgang Thomas: "Automata on Infinite Objects". In Jan van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, vol. B, Elsevier, 1990, pp.133–165.

- Moshe Vardi and Pierre Wolper, "An Automata Theoretic Approach to Program Verification", *Symposium on Logic in Computer Science*, 1986, pp.322–331.

ω-Automata (Motivation)



n_1 represents all states in which p_1 is true;
i.e. $t f$ and $t\, t$.

$$\mu(n_1) = \{tf, \; t\, t\}$$

n_2 represents all states in which p_1 is false and p_2 is true.

$$\mu(n_2) = \{ft\}$$

ω-Automata (Definition)

Set of propositions: $p_1, \ldots, p_n$.
Alphabet $\Sigma = \{t,\ f\}^n$.

Automaton $\mathcal{A}$: $\langle N,\ N_0,\ E,\ \mu,\ F \rangle$, where

- N: finite set of nodes
- $N_0 \subseteq N$: initial nodes
- $E \subseteq N \times N$: edges
- $\mu : N \to 2^{\Sigma}$: node labeling function (assertions)
- F: acceptance condition

Note: Most of the literature on ω-automata uses edge labeling, similarly to automata on finite strings. However, we use node labeling to ease the transition to diagrams. The two approaches are equally expressive and can easily be translated into each other.

ω-Automata: Trails

Definition: A path

$$\pi : n_0, n_1, \ldots$$

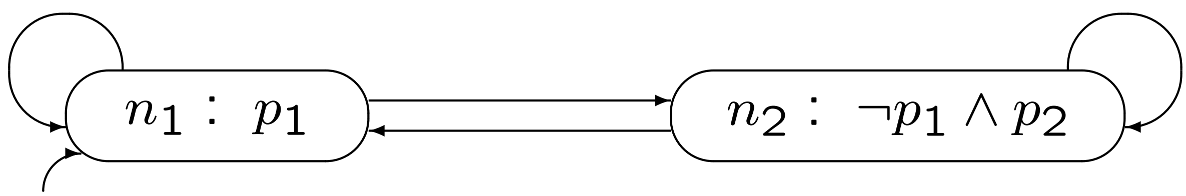
of $\mathcal{A}$ is a <u>trail</u> of an infinite sequence of states

$$\sigma : s_0, s_1, \ldots$$

if for every $i \geq 0$,

$$s_i \vDash \mu(n_i) \quad (\text{or } s_i \in \mu(n_i)).$$

Example:



The sequence of states

$$\sigma : \overset{\overset{p_1}{\downarrow}}{t}\,\overset{\overset{p_2}{\downarrow}}{t}, \quad tf, \quad ft, \quad tt, \quad tf, \quad ft, \quad \ldots$$

has trail

$$\pi : \quad n_1, \quad n_1, \quad n_2, \quad n_1, \quad n_1, \quad n_2, \quad \ldots$$

Note: no trail for $\sigma : \ldots, ff, \ldots$.

- In general, $\mathcal{A}$ is <u>nondeterministic</u> i.e., trail π is not necessarily unique for σ.

- $\mathcal{A}$ is <u>deterministic</u> if for every σ, there is exactly one trail π of σ.

$\mathsf{Inf}(\pi)$

$$\underline{\text{infinite sequence of states}}\ \sigma : s_0, s_1, s_2, \ldots$$
$$\downarrow$$
$$\underline{\text{infinite trail}}\ \pi : n_0, n_1, n_2, \ldots$$

<u>inf(π)</u>:	The set of nodes appearing infinitely often in π.

<u>Observe</u>:

- inf(π) is <u>nonempty</u> since the set of nodes of the automaton is finite.
- The nodes in inf(π) form a <u>Strongly Connected Subgraph</u> (SCS) in $\mathcal{A}$.

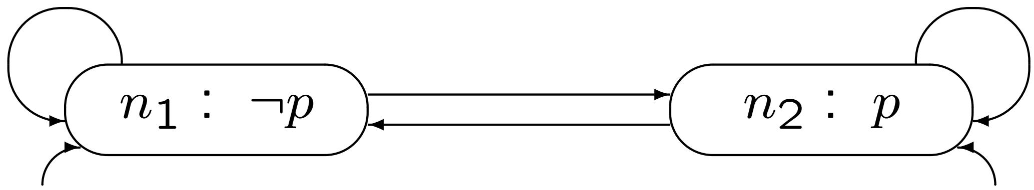
<u>SCS S</u>: Every node in S is reachable from every other node in S.

<u>MSCS S</u>: a <u>maximal</u> SCS;
i.e., S is not contained in any larger SCS.

Definition: An infinite sequence of states σ is <u>accepted</u> by $\mathcal{A}$ if it has a trail π such that inf(π) is accepted by the <u>acceptance condition</u>.

ω-Automata: Acceptance Conditions

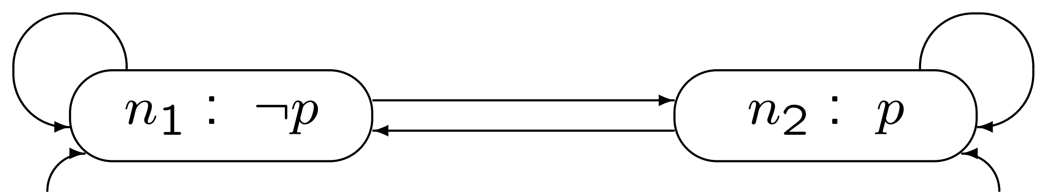
$\mathcal{A}$:



Name	Büchi	Muller
Type of acceptance condition	$F \subseteq N$ a set of nodes	$F \subseteq 2^N$ a set of subsets of nodes
Condition for acceptance	$\inf(\pi) \cap F \neq \emptyset$	$\inf(\pi) \in F$
To accept $\mathcal{L}(\Box \Diamond p)$ with $\mathcal{A}$	$F = \{n_2\}$	$F = \{\{n_1, n_2\}, \{n_2\}\}$
To accept $\mathcal{L}(\Diamond \Box p)$ with $\mathcal{A}$	no deterministic Büchi automaton accepts this language	$F = \{\{n_2\}\}$

ω-Automata: Acceptance Conditions (Cont'd)

$\mathcal{A}$:



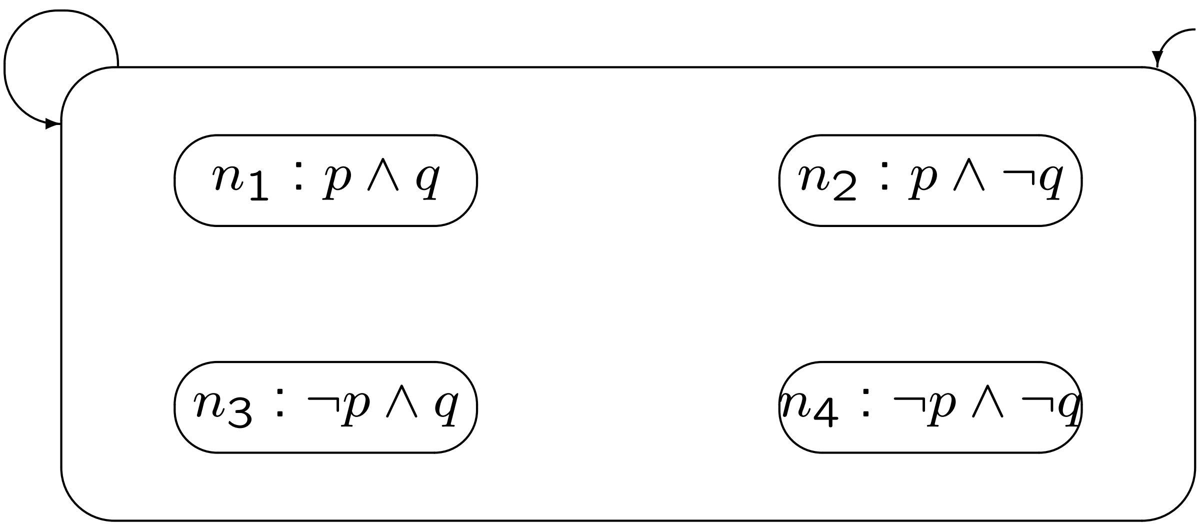
Name	<u>Streett</u>	<u>Rabin</u>
Type of acceptance condition	$F \subseteq 2^N \times 2^N$ a set of pairs $\{(P_1, R_1), \ldots, (P_n, R_n)\}$ where each P_i, R_i is a set of nodes	
Condition for acceptance	for <u>every</u> $i : [1..n]$ $\inf(\pi) \subseteq P_i$ <u>or</u> $\inf(\pi) \cap R_i \neq \emptyset$	for <u>some</u> $i : [1..n]$ $\inf(\pi) \subseteq P_i$ <u>and</u> $\inf(\pi) \cap R_i \neq \emptyset$
To accept $\mathcal{L}(\Box \Diamond p)$ with $\mathcal{A}$	$F = \{(\emptyset, \{n_2\})\}$	$F = \{(\{n_1, n_2\}, \{n_2\})\}$
To accept $\mathcal{L}(\Diamond \Box p)$ with $\mathcal{A}$	$F = \{(\{n_2\}, \emptyset)\}$	$F = \{(\{n_2\}, \{n_2\})\}$

Automata

Automaton for $\Box \Diamond p \rightarrow \Box \Diamond q$
(if p happens infinitely often, then q happens infinitely often)

$$\Diamond \Box \neg p \;\vee\; \Box \Diamond q$$

<u>Deterministic</u>:



<u>Muller</u> acceptance condition ($\mathcal{P}$ = powerset):

$$F \;=\; \mathcal{P}(\{n_1, n_2, n_3, n_4\}) - \{\{n_2\}, \{n_2, n_4\}\}$$

<u>Streett</u> acceptance condition:

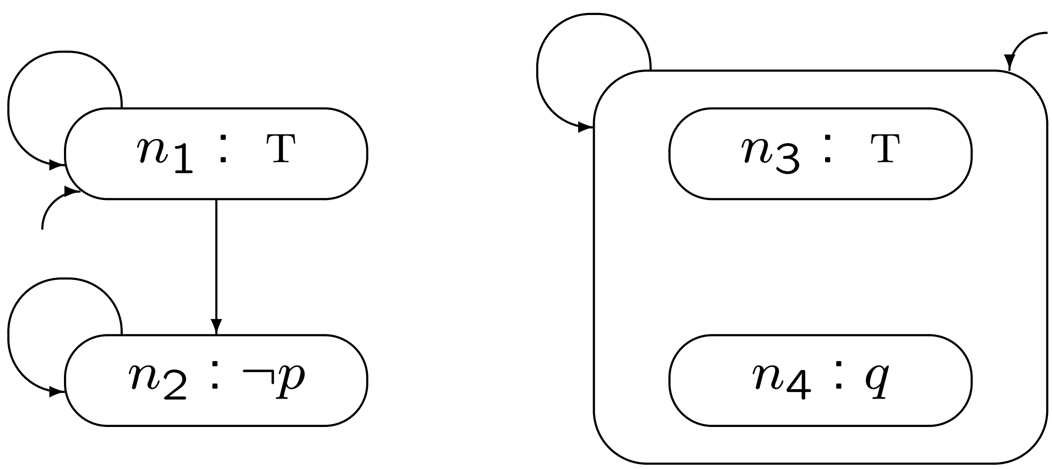
$$F \;=\; \{(\overbrace{\{n_3,\ n_4\}}^{\substack{\text{eventually}\\ \text{always } \neg p}} \quad \overset{\text{or}}{,} \quad \overbrace{\{n_1,\ n_3\}}^{\substack{\text{infinitely}\\ \text{often } q}})\}$$

Automata (Cont'd)

Automaton for $\Box \Diamond p \rightarrow \Box \Diamond q$

$$\Diamond \Box \neg p \ \vee \ \Box \Diamond q$$

<u>Nondeterministic</u>:



<u>Muller</u> acceptance condition:

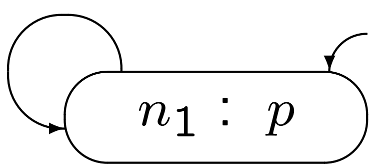
$$F = \{\{n_2\},\ \{n_4\},\ \{n_3,\ n_4\}\}$$

<u>Streett</u> acceptance condition:

$$F = \{(\{n_2\},\ \{n_4\})\}$$

More Examples: Muller/Streett

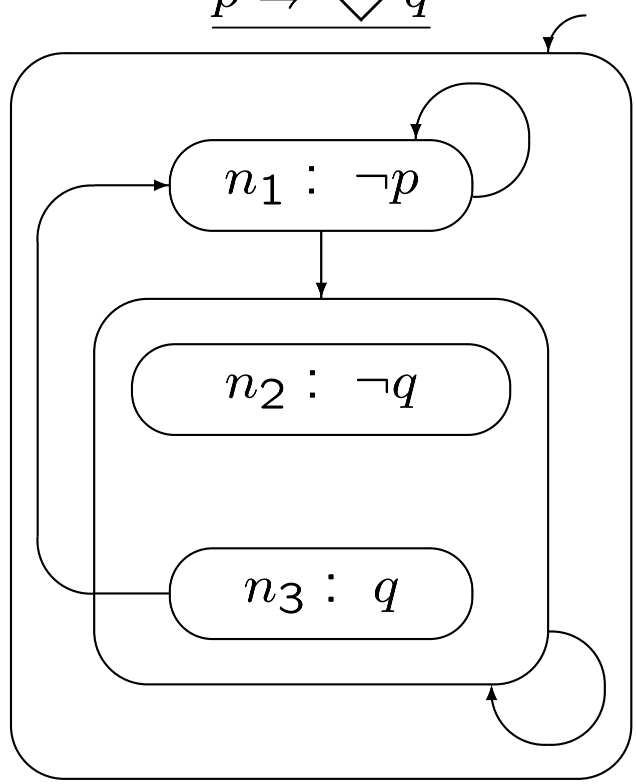
$\Box p$



$$F_M = \{\{n_1\}\}$$

$$F_S = \{(\{n_1\}, \varnothing)\}$$

$p \Rightarrow \Diamond q$

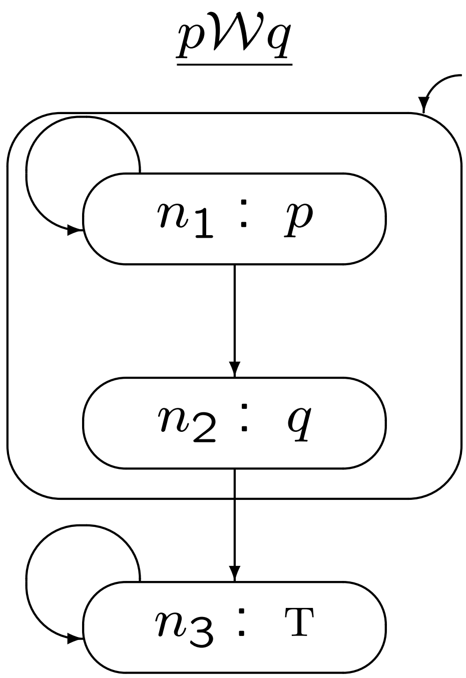


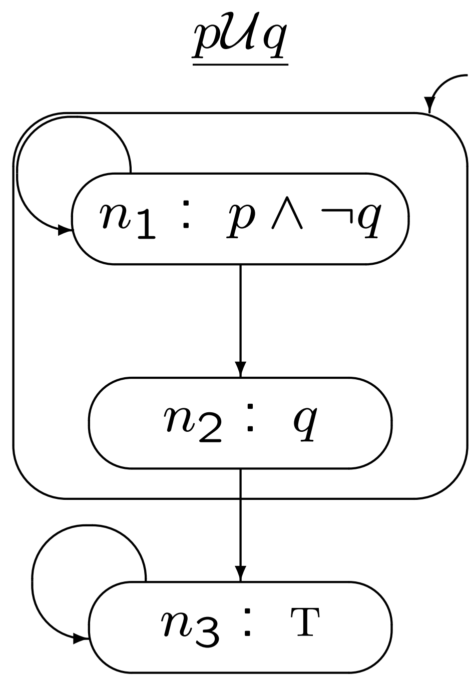
$$F_M = \{\{n_1\}, \{n_3\}, \{n_1, n_3\}, \{n_2, n_3\}, \{n_1, n_2, n_3\}\}$$

$$F_S = \{(\varnothing, \{n_1, n_3\})\}$$

Question: Why is $\{n_1, n_2\}$ not in F_M?

More Examples: Muller/Streett





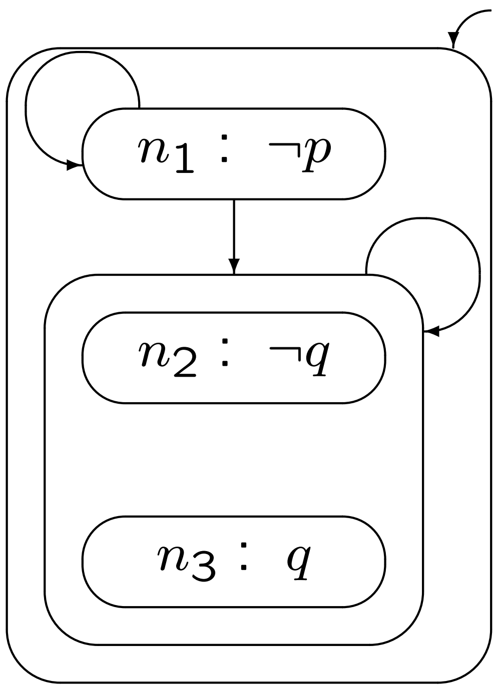
$$F_M = \{\{n_1\}, \{n_3\}\} \qquad F_M = \{\{n_3\}\}$$

$$F_S = \{(\{n_1, n_3\}, \emptyset)\} \qquad F_S = \{(\{n_3\}, \emptyset)\}$$

Question: Why $n_1 : p \wedge \neg q$ and not $n_1 : p$?

More Examples: Muller/Streett

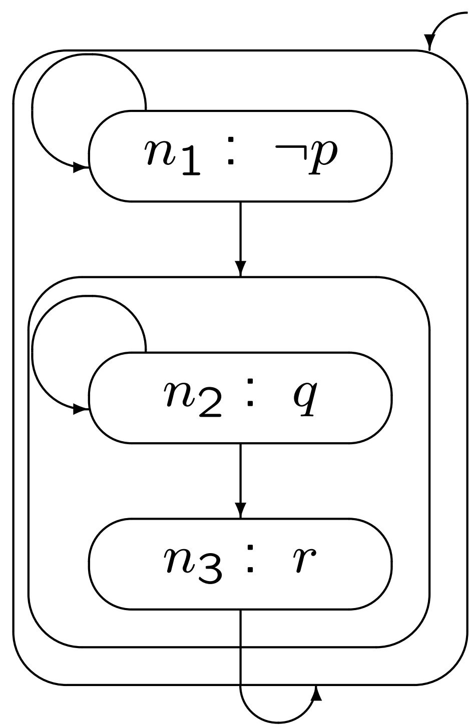
$\underline{p \Rightarrow \Diamond \Box q}$



$$F_M = \{\{n_1\}, \{n_3\}\}$$

$$F_S = \{(\{n_1, n_3\}, \emptyset)\}$$

$\underline{p \Rightarrow q \mathcal{W} r}$



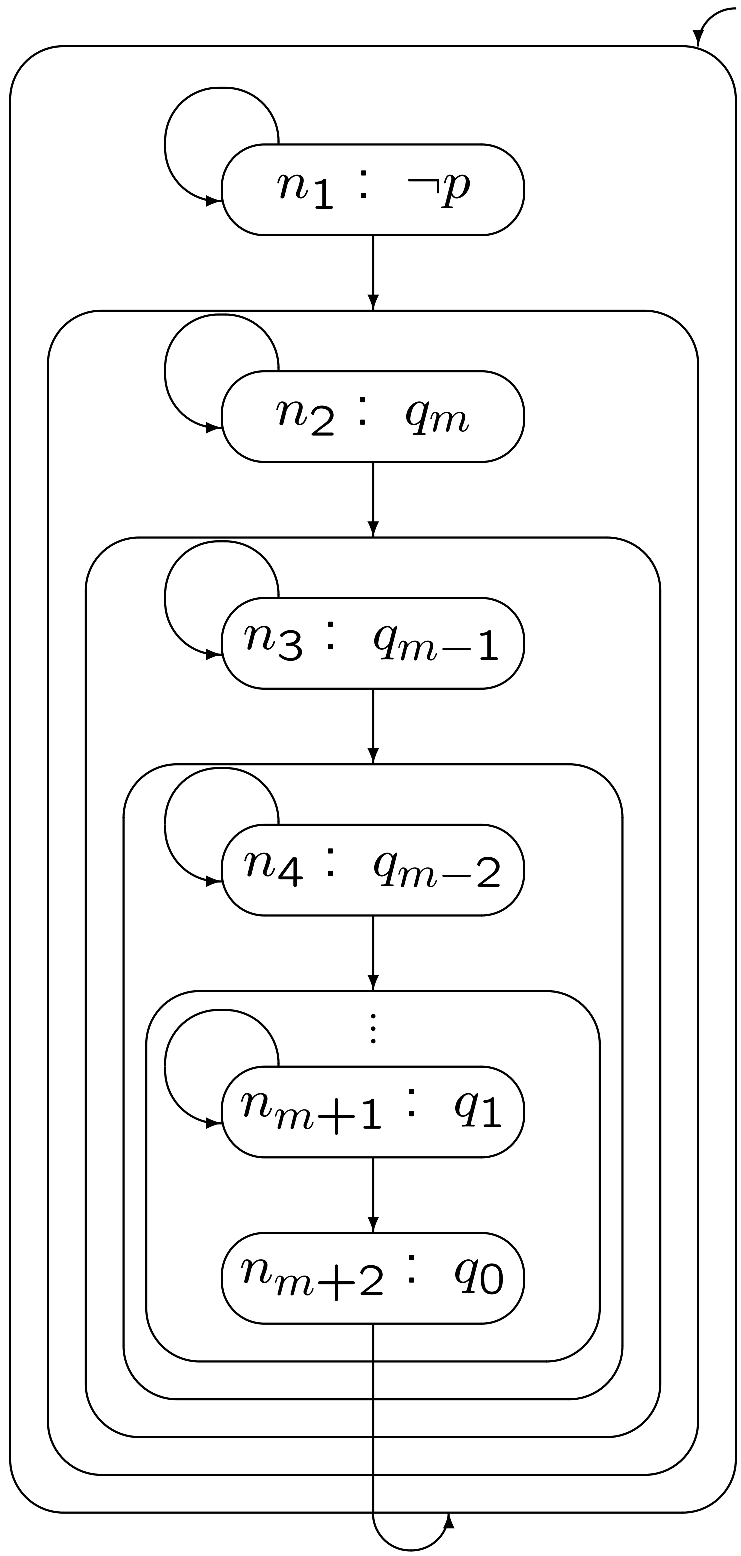
$$F_M = \mathcal{P}(\{n_1, n_2, n_3\}) - \{n_1, n_2\}$$

$$F_S = \{(\emptyset, \{n_1, n_2, n_3\})\}$$

More Examples: Muller/Streett

$$\underline{p \Rightarrow q_m \mathcal{W} q_{m-1} \ldots q_1 \mathcal{W} q_0}$$

$$\begin{aligned} F_M &= \mathcal{P}(\{n_1, \ldots, n_{m+2}\}) \\ F_S &= \{(\emptyset, \{n_1, \ldots, n_{m+2}\})\} \end{aligned}$$

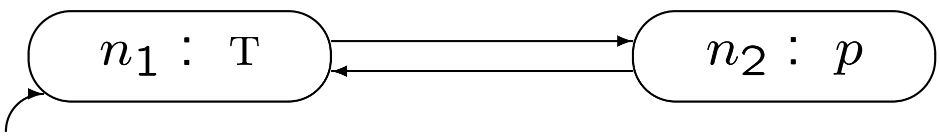

$n_1 : \neg p$
$n_2 : q_m$
$n_3 : q_{m-1}$
$n_4 : q_{m-2}$
$\vdots$
$n_{m+1} : q_1$
$n_{m+2} : q_0$

Existence of ω-Automaton

Theorem: For every PLTL formula φ, there exists an ω-automaton $\mathcal{A}_\varphi$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A}_\varphi)$.

Question: Does the converse also hold?

- Consider $\mathcal{A}$:



$$F_M = \{\{n_1, n_2\}\}$$

$\mathcal{L}(\mathcal{A}) =$ all sequences of form

$$\begin{array}{ccccccccc} p & & p & & p & & p & & \\ \neg p & p & \neg p & p & \neg p & p & \neg p & p & \dots \end{array}$$

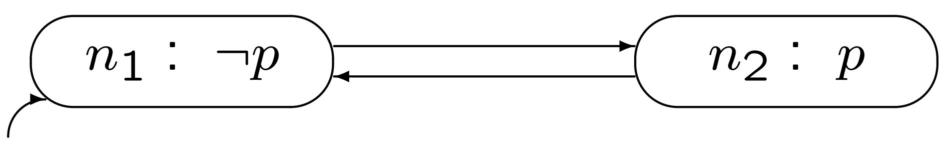
Is there a PLTL formula φ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$?

Existence of ω-Automaton (Cont'd)

- **First attempt**: $\bigcirc p \land \Box(p \leftrightarrow \bigcirc \neg p)$
  - Not good because it only accepts

$$\underline{\neg p \quad p \quad \neg p \quad p \quad \neg p \ldots}$$

  - That is, it accepts $\mathcal{L}(\mathcal{A}_1)$, with $\mathcal{A}_1$:



$$F_M = \{\{n_1, n_2\}\}$$

Existence of ω-Automaton (Cont'd)

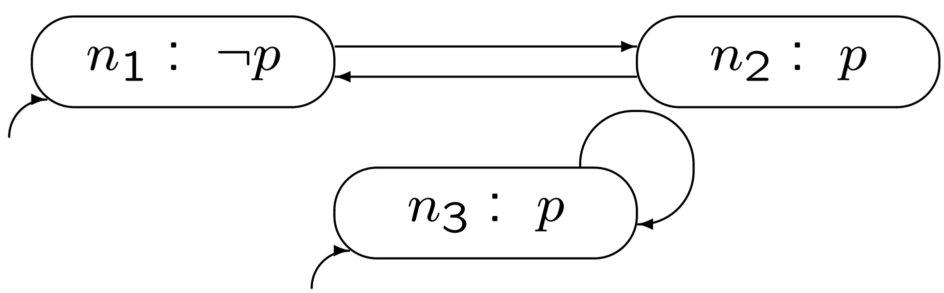
- **Second attempt**: $\bigcirc\, p \wedge \Box(p \equiv \bigcirc \bigcirc p)$
  - Not good because it accepts only

    $$\underline{\neg p \quad p \quad \neg p \quad p \quad \neg p \quad \ldots}$$

    and

    $$\underline{p \quad p \quad p \quad p \quad p \quad \ldots}$$

  - That is, it accepts $\mathcal{L}(\mathcal{A}_2)$, with $\mathcal{A}_2$:



$$F_M = \{\{n_1, n_2\}, \{n_3\}\}$$

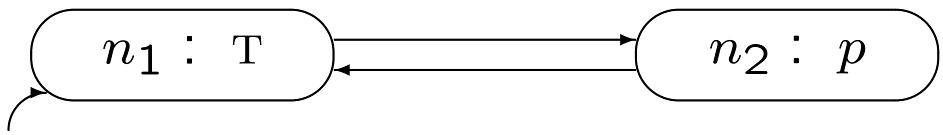
ω-Automaton Expressiblity

It was shown by Wolper (1982) that there does not exist a PLTL formula φ such that
$\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$ for the automaton $\mathcal{A}$ shown above.

Theorem: ω-automata are strictly more expressive than PLTL.

Theorem: For every ω-automaton $\mathcal{A}$ there exists an existentially quantified formula φ such that
$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$.

`Example:`



$$F = \{\{n_1, n_2\}\}$$

$$\exists k.(\ \underbrace{\bigcirc k}_{\substack{k \text{ holds in} \\ \text{the second} \\ \text{position}}} \ \wedge\ \underbrace{\Box(k \leftrightarrow \bigcirc \neg k)}_{\substack{k\text{-positions} \\ \text{alternate}}} \ \wedge\ \underbrace{\Box(k \rightarrow p)}_{\substack{\text{whenever } k \\ \text{holds, } p \\ \text{also holds}}})$$

k is a flexible, auxiliary boolean variable:
its value may be different in different positions.

Note: $\neg k$ at position 0. Why?