Verification with ω-automata

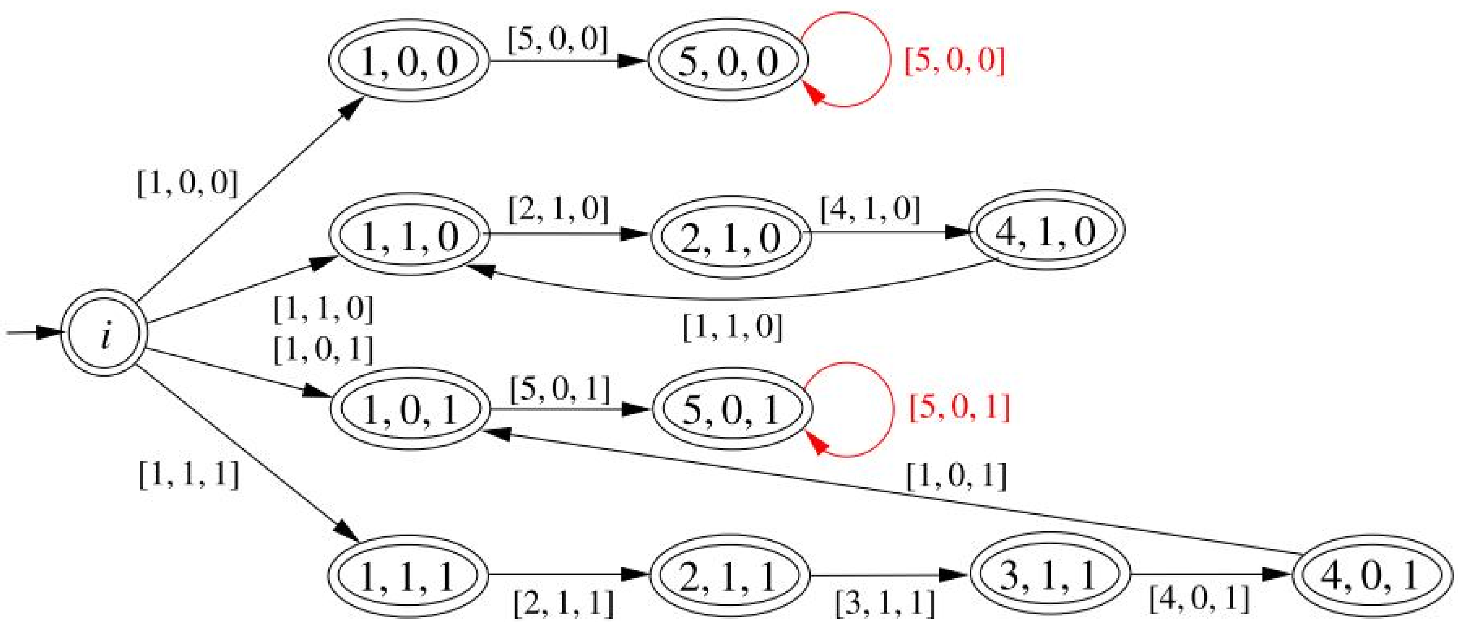
Programs and ω-executions

- Recall: a full execution of a program is an execution that cannot be extended (either infinite or ending at a configuration without successors).
- We consider programs that may have ω-executions.
- We assume w.l.o.g. that every full execution of the program is infinite (see next slide).
- Therefore: full executions = ω-executions

Handling finite full executions

```
while x = 1 do
    if y = 1 then
        x ← 0
    y ← 1 − x
end
```

We artificially ensure that every full execution is infinite by adding a self-loop to every state without successors.



Verifying a program

- Goal: automatically check if some ω-execution violates a property.
- Safety property: "nothing bad happens"
  - No configuration satisfies $x = 1$.
  - No configuration is a deadlock.
  - Along an execution the value of x cannot decrease.
- Liveness property: "something good eventually happens"
  - Eventually x has value 1.
  - Every message sent during the execution is eventually received.

Safety and liveness: more precisely

- A finite execution w is bad for a given property if every potential ω-execution of the form $w\,w'$ violates the property.
- A property is a safety property if every ω-execution that violates the property has a bad prefix.

  (Intuitively: after finite time we can already say that the property does not hold)
- A property is a liveness property if some ω-execution that violates the property has no bad prefix.

  (We can only tell that the property is a violation ``after seeing the complete ω-execution''.)

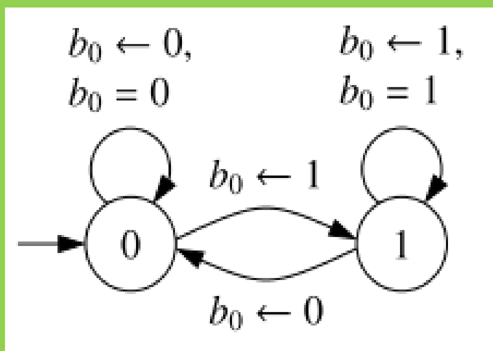
Approach to automatic verification

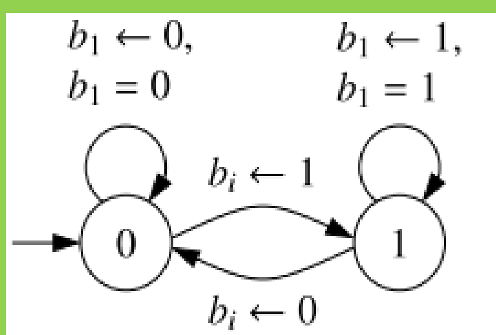
- Represent the set of ω-executions of the program as a NBA. (The system NBA).
- Represent the set of possible ω-executions that violate the property as a NBA (or an ω-regular expression). (The property NBA).
- Check emptiness of the intersection of the two NBAs.

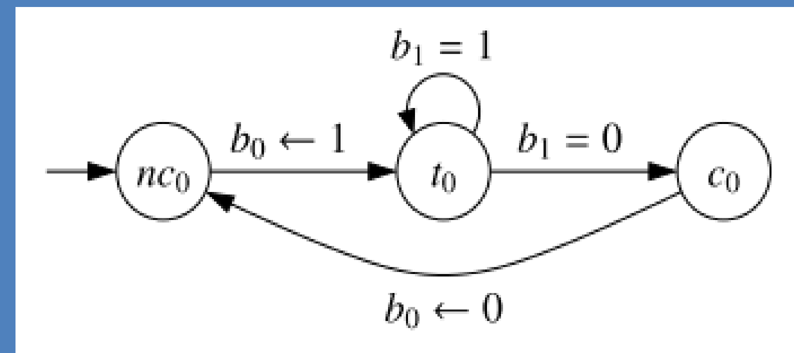
Problem: Fairness

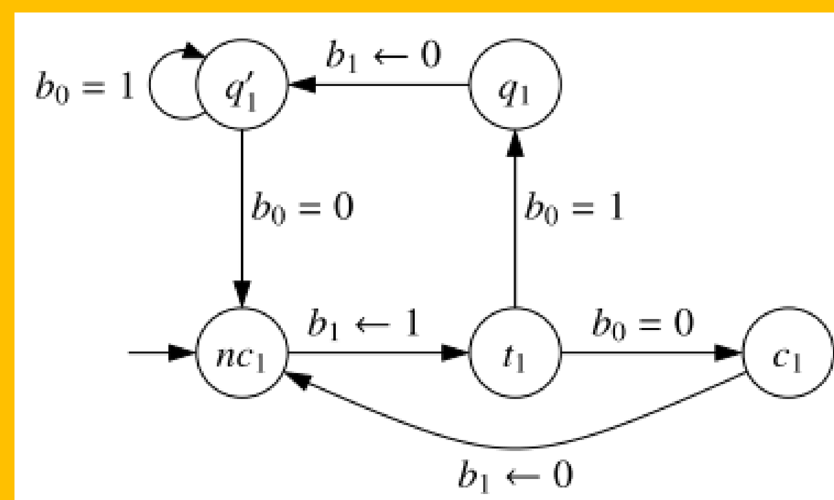
- We may want to exclude some ω-executions because they are “unfair”.
- Example: finite waiting property in Lamport‘s mutex algorithm.

Lamport´s algorithm

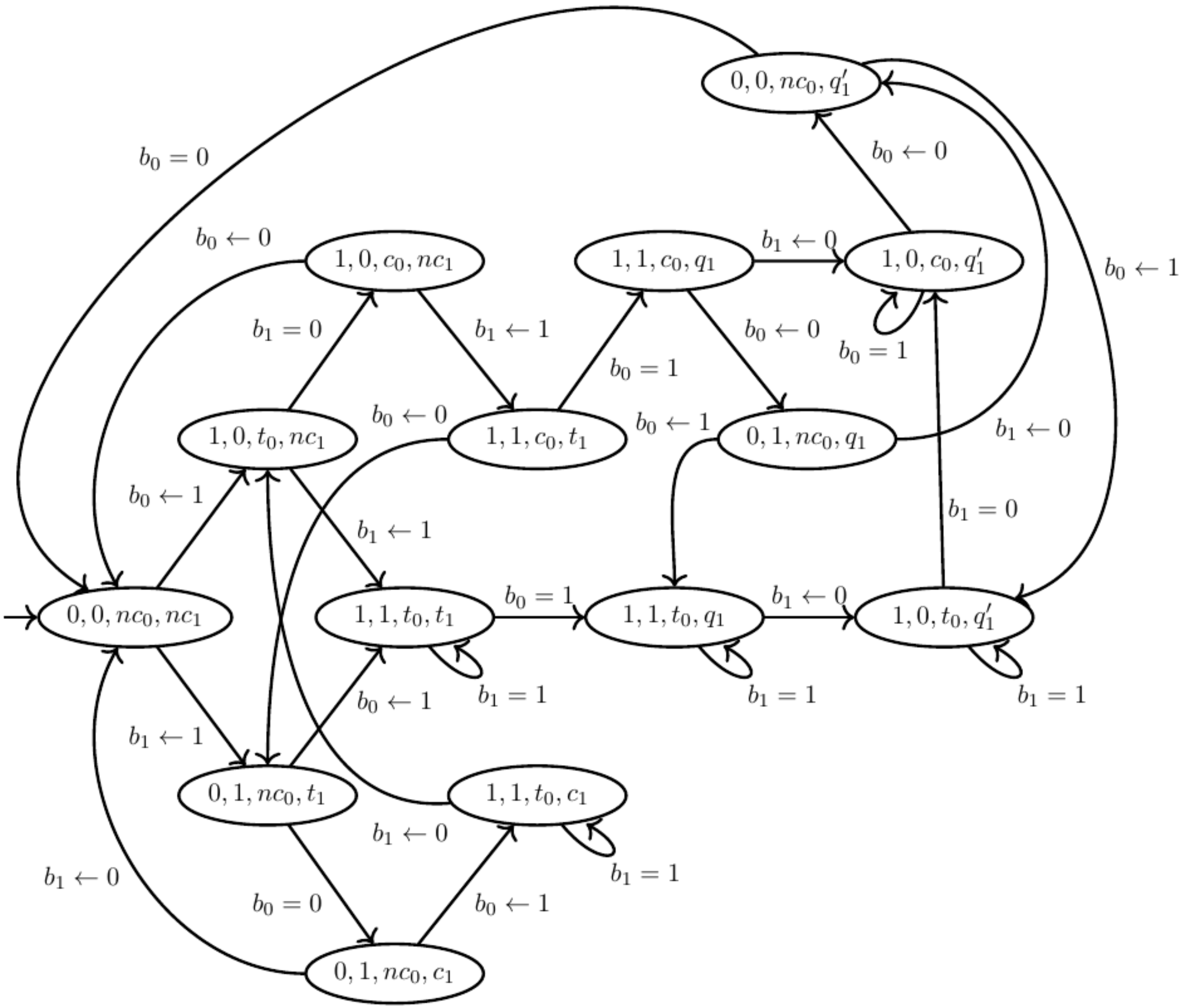








Asynchronous product



Finite waiting property

- Finite waiting: If a process is trying to access the critical section, it eventually will.
- Formalization: Let NC_i, T_i, C_i be atomic propositions mapped to the sets of configurations where process i is in the non-critical section, trying to access it, and in the critical section, respectively.
  The full executions that violate finite waiting for process i are

$$\Sigma^* T_i \, (\Sigma \setminus C_i)^\omega$$

- Observe: all states of the system NBA are final, and so we can intersect NBAs using the algorithm for NFAs

Finite waiting property

- The finite waiting property does not hold because of

$$[0{,}0, nc_0, nc_1]\ \ [1{,}0, t_0, nc_1]\ \ [1{,}1, t_0, t_1]^{\omega}$$

- Is this a real problem of the algorithm?
  No! We have not specified correctly.
- Fairness assumption: both processes execute infinitely many actions.
  (Usually a weaker assumption is used: if a process can execute actions infinitely often, it executes infinitely many actions.)
- Reformulation: in every fair ω-execution, if a process is trying to access the critical section, it will eventually access it.

Finite waiting property

- The violations of the property under fairness are the intersection of $\Sigma^* T_i (\Sigma \setminus C_i)^\omega$ and the ω-executions in which both processes make a move infinitely often.
- Problem: how do we represent this condition as an ω-regular language?
- Solution: enrich the alphabet of the NBA
  Letter: pair (c, i) where c is a configuration and i is the index of the process making the move.

Finite waiting property

- Denote by M_0 and M_1 the set of letters with index 0 and 1, respectively.
- The possible ω-executions where both processes move infinitely often is given by

$$\left((M_0 + M_1)^* M_0 M_1\right)^\omega$$

- Finite waiting holds under fairness for process 0 but not for process 1 because of

$$(\ [0,0,nc_0,nc_1][0,1,nc_0,t_1][1,1,t_0,t_1][1,1,t_0,q_1]$$
$$[1,0,t_0,q_1'][1,0,c_0,q_1'][0,0,nc_0,q_1'] \ )^\omega$$

Temporal logic

- Writing property NBAs or ω-regular expressions requires training in automata theory
- We search for a more intuitive (but still formal) description language: Temporal Logic.
- Temporal logic extends propositional logic with temporal operators like always and eventually.
- Linear Temporal Logic (LTL) is a temporal logic interpreted over linear structures.

Linear Temporal Logic (LTL)

- We are given:
  - A set AP of atomic propositions (names for basic properties)
  - A valuation assigning to each atomic proposition a set of configurations (intended meaning: the set of configurations that satisfy the property).

Example

```
1  while x = 1 do
2      if y = 1 then
3          x ← 0
4      y ← 1 − x
5  end
```

- AP : at_1, $\mathrm{at}_2, \ldots,$ at_5, x=0, x=1, y=0, y=1
- $V(\mathrm{at}_i) = \{[\ell, x, y] \in C \mid \ell = i\}$ for every $i \in \{1, \ldots, 5\}$
- $V(\mathrm{x{=}0}) = \{[\ell, x, y] \in C \mid \mathrm{x} = 0\}$

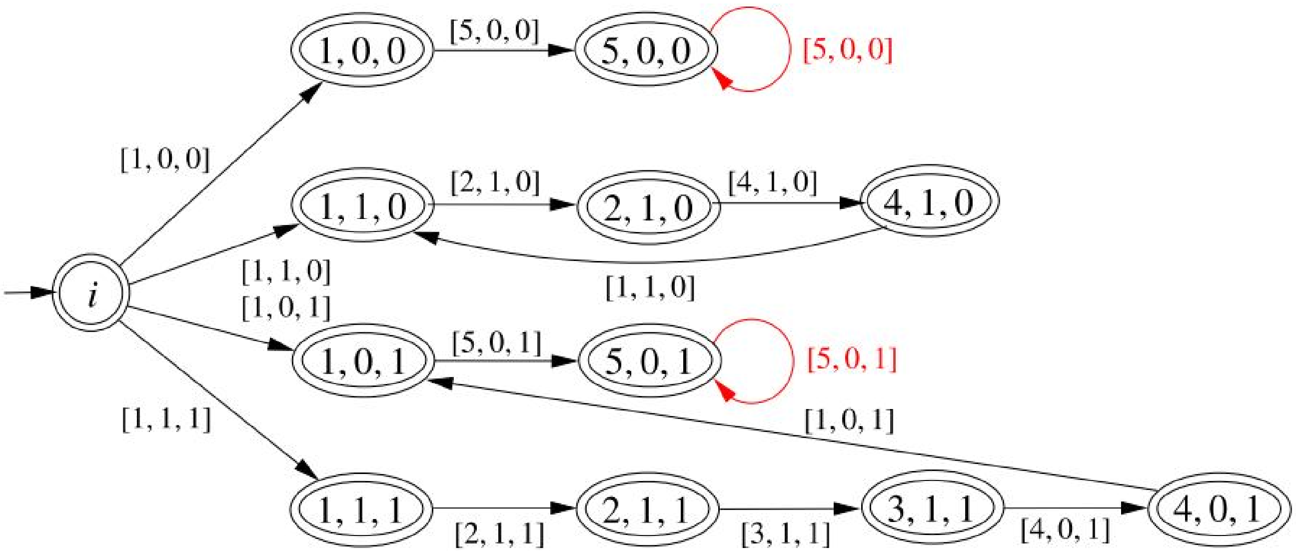
Computations

- A computation is an infinite sequence of subsets of AP.
- Examples for $AP = \{p, q\}$

$$\emptyset^\omega \quad (\{p\}\{p,q\})^\omega \quad \{p\}\,\{p,q\}\,\emptyset\,\emptyset\,\{p\}^\omega$$

- We map every possible execution to a computation by mapping each configuration to the set of atomic propositions it satisfies.
- A computation is executable if some ω-execution maps to it.

Example



ω-executions:

e_1 = [1,0,0] [5,0,0]$^\omega$

e_2 = ([1,1,0] [2,1,0] [4,1,0])$^\omega$

e_3 = [1,0,1][5,0,1]$^\omega$

e_4 = [1,1,1][2,1,1][3,1,1][4,0,1][1,0,1][5,0,1]$^\omega$

From executions to computations

e_1 = [1,0,0] [5,0,0]$^\omega$

e_2 = ([1,1,0] [2,1,0] [4,1,0])$^\omega$

σ_1 = {at1, x=0, y=0} {at5, x=0, y=0}$^\omega$

σ_2 = ({at1, x=0, y=0} {at2, x=1, y=0} {at4, x=1, y=0})$^\omega$

Syntax of LTL

- Given: set AP of atomic propositions, valuation assigning to each atomic proposition a set configurations.
- The formulas of LTL are given by the syntax:

$$\varphi ::= \mathbf{true} \mid p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \mathrm{X}\varphi_1 \mid \varphi_1 \mathrm{U} \varphi_2$$

where $p \in AP$

Semantics of LTL

- Formulas are interpreted on computations (executable or not).
- The satisfaction relation $\sigma \vDash \varphi$ is given by:

$$\sigma \vDash \mathbf{true}$$

$$\sigma \vDash p \text{ iff } p \in \sigma(0)$$

$$\sigma \vDash \neg\varphi \text{ iff not } \sigma \vDash \varphi$$

$$\sigma \vDash \varphi_1 \wedge \varphi_2 \text{ iff } \sigma \vDash \varphi_1 \text{ and } \sigma \vDash \varphi_2$$

$$\sigma \vDash \mathrm{X}\varphi \text{ iff } \sigma^1 \vDash \varphi$$

$$\sigma \vDash \varphi_1 \mathrm{U} \varphi_2 \text{ iff there is } k \geq 0 \text{ s.t.}: \sigma^k \vDash \varphi_2 \text{ and } \sigma^i \vDash \varphi_1 \text{ for all } 0 \leq i < k$$

Abbreviations

- The boolean abbreviations **false**, $\vee$, $\rightarrow$, $\leftrightarrow$ etc. are defined as usual.
- $\mathrm{F}\varphi :=$ **true** $\mathrm{U}\ \varphi$ (eventually φ).

  According to the semantics:

  $\sigma \vDash \mathrm{F}\varphi$ iff there is $k \geq 0$ s. t. $\sigma^k \vDash \varphi$
- $\mathrm{G}\varphi := \neg\ \mathrm{F}\neg\varphi$ (always φ or globally φ).

  According to the semantics:

  $\sigma \vDash \mathrm{G}\varphi$ iff $\sigma^k \vDash \varphi$ for every $k \geq 0$

Getting used to LTL

- Express in natural language $\mathsf{FG}p$, $\mathsf{GF}p$
- Are these pairs of formulas equivalent?

$\mathsf{FF}p$	$\mathsf{F}p$	$\mathrm{GG}p$	$\mathrm{G}p$
$\mathsf{FG}p$	$\mathsf{GF}p$	$\mathrm{FGF}p$	$\mathrm{GF}p$
$p \,\mathsf{U}\, q$	$p \,\mathsf{U}\, (p \wedge q)$		
$\mathrm{F}p$	$p \vee \mathrm{XF}p$	$\mathrm{F}p$	$p \wedge \mathrm{XF}p$
$\mathrm{G}p$	$p \vee \mathrm{XG}p$	$\mathrm{G}p$	$p \wedge \mathrm{XG}p$
$p \,\mathsf{U}\, q$	$p \vee \mathsf{X}\,(p \,\mathsf{U}\, q)$	$p \,\mathsf{U}\, q$	$p \wedge \mathsf{X}\,(p \,\mathsf{U}\, q)$
$p \,\mathsf{U}\, q$	$q \vee \mathsf{X}\,(p \,\mathsf{U}\, q)$	$p \,\mathsf{U}\, q$	$q \wedge \mathsf{X}\,(p \,\mathsf{U}\, q)$
$p \,\mathsf{U}\, q$	$q \vee (p \wedge \mathsf{X}\,(p \,\mathsf{U}\, q))$	$p \,\mathsf{U}\, q$	$q \wedge (p \vee \mathsf{X}\,(p \,\mathsf{U}\, q))$

Expressing properties of a program

- AP : at_1, at_2, ... , at_5, x=0, x=1, y=0, y=1

  $V(\text{at}_\text{i}) = \{[\ell, x, y] \in C \mid \ell = i\}$ for every $i \in \{1, \ldots, 5\}$

  $V(\text{x=0}) = \{[\ell, x, y] \in C \mid \text{x=0}\}$
- $\varphi_0 = \text{x=1} \wedge \text{X y=1} \wedge \text{X X at3}$
- $\varphi_1 = \text{F x=0}$
- $\varphi_2 = \text{x=0 U at5}$
- $\varphi_3 = \text{y=1} \wedge \text{F(x=0} \wedge \text{at5)} \wedge \neg\text{(F (y=0} \wedge \text{X y=1))}$

Expressing properties of Lamport´s algorithm

- $AP = \{ NC_0, T_0, C_0, NC_1, T_1, C_1, M_0, M_1\}$

  Valuation as expected.

- Mutual exclusion: $\mathsf{G}\ (\neg C_0 \lor \neg C_1)$
- Finite waiting: $\mathsf{G}\ (T_0 \rightarrow \mathsf{F} C_0) \land \mathsf{G}\ (T_1 \rightarrow \mathsf{F} C_1)$
- Fair finite waiting:

  $(\mathsf{GF}\ M_0 \land \mathsf{GF}\ M_1) \rightarrow \left(\mathsf{G}(T_0 \rightarrow \mathsf{F} C_0) \land \mathsf{G}(T_1 \rightarrow \mathsf{F} C_1)\right)$

Expressing properties of Lamport´s algorithm

- Bounded overtaking:

$$\mathsf{G}\left(T_0 \rightarrow \left(\neg C_1 \,\mathsf{U}\, \left(C_1 \,\mathsf{U}\, \left(\neg C_1 \,\mathsf{U}\, C_0\right)\right)\right)\right)$$

Whenever T_0 holds, the computation continues with

a (possibly empty) interval at which $\neg C_1$ holds,

followed by

a (possibly empty) interval at which C_1 holds,

followed by

a point at which C_0 holds.

From formulas to NBAs

- Given: set AP of atomic propositions
- Language $L(\varphi)$ of a formula φ : set of computations satisfying φ.
- Examples for $AP = \{p, q\}$
  - $L(\mathrm{F}p) =$ computations $s_1 s_2 s_3 \ldots$ such that $p \in s_i$ for some $i \geq 1$
  - $L\big(\mathsf{G}(p \wedge q)\big) = \{\, \{p, q\}^\omega \,\}$
- $L(\varphi)$ is an ω-language over the alphabet 2^{AP}
- For $AP = \{p, q\}$ we get $2^{AP} = \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$

NBAs for some formulas

$$AP = \{p, q\}$$

- $\mathrm{F}p$
- $\mathrm{G}p$
- $p \; \mathrm{U} \; q$
- $\mathrm{GF}p$

From LTL formulas to NGAs

We present an algorithm that takes a formula φ over a fixed set AP of atomic propositions as input and returns a NGA A_φ such that $L(A_\varphi) = L(\varphi)$.

Closure of a formula

- Define $\text{neg}(\varphi) = \begin{cases} \psi & \text{if } \varphi = \neg\psi \\ \neg\varphi & \text{otherwise} \end{cases}$
- The closure $cl(\varphi)$ of φ is the set containing ψ and $\text{neg}(\psi)$ for every subformula ψ of φ
- Example:

$$cl(p \, \mathrm{U} \neg q) = \{p, \neg p, \neg q, q, p \mathrm{U} \neg q, \neg(p \mathrm{U} \neg q)\}$$

Satisfaction sequence

- The satisfaction sequence of a computation $s_0 s_1 s_2 \ldots$ with respect to φ is the sequence $\alpha_0 \alpha_1 \alpha_2 \ldots$ where α_i contains the formulas of $cl(\varphi)$ satisfied by $s_i s_{i+1} s_{i+2} \cdots$

Satisfaction sequence

- The satisfaction sequence of a computation $s_0 s_1 s_2 \ldots$ with respect to φ is the sequence $\alpha_0 \alpha_1 \alpha_2 \ldots$ where α_i contains the formulas of $cl(\varphi)$ satisfied by $s_i s_{i+1} s_{i+2} \cdots$
- The satisfaction sequence of $\{p\}^\omega$ w.r.t. p U q is:

Satisfaction sequence

- The satisfaction sequence of a computation $s_0 s_1 s_2 \ldots$ with respect to φ is the sequence $\alpha_0 \alpha_1 \alpha_2 \ldots$ where α_i contains the formulas of $cl(\varphi)$ satisfied by $s_i s_{i+1} s_{i+2} \cdots$
- The satisfaction sequence of $\{p\}^\omega$ w.r.t. $p \cup q$ is:

$$\{p, \neg q, \neg(p \cup q)\}^\omega$$

Satisfaction sequence

- The satisfaction sequence of a computation $s_0 s_1 s_2 \ldots$ with respect to φ is the sequence $\alpha_0 \alpha_1 \alpha_2 \ldots$ where α_i contains the formulas of $cl(\varphi)$ satisfied by $s_i s_{i+1} s_{i+2} \cdots$
- The satisfaction sequence of $\{p\}^\omega$ w.r.t. $p \cup q$ is:

$$\{p, \neg q, \neg(p \cup q)\}^\omega$$

- The satisfaction sequence of $(\{p\}\{q\})^\omega$ w.r.t. $p \cup q$ is:

Satisfaction sequence

- The satisfaction sequence of a computation $s_0 s_1 s_2 \ldots$ with respect to φ is the sequence $\alpha_0 \alpha_1 \alpha_2 \ldots$ where α_i contains the formulas of $cl(\varphi)$ satisfied by $s_i s_{i+1} s_{i+2} \cdots$
- The satisfaction sequence of $\{p\}^\omega$ w.r.t. $p \cup q$ is:

$$\{p, \neg q, \neg(p \cup q)\}^\omega$$

- The satisfaction sequence of $(\{p\}\{q\})^\omega$ w.r.t. $p \cup q$ is:

$$(\,\{p, \neg q, p \cup q\}\,\{\neg p, q, p \cup q\}\,)^\omega$$

- Goal for the next slides: give a syntactic characterization of the satisfaction sequence

Atoms

- Intuition: an atom is a “maximal set of formulas of $cl(\varphi)$ that can be simultaneously true if one only knows the meaning of $\neg$ and $\land$”

Atoms

- Intuition: an atom is a “maximal set of formulas of $cl(\varphi)$ that can be simultaneously true if one only knows the meaning of ¬ and ∧”
- A set $\alpha \subseteq cl(\varphi)$ is an atom if it satisfies the following conditions:
  - If $\mathbf{true} \in cl(\varphi)$, then $\mathbf{true} \in \boldsymbol{\alpha}$
  - For every $\psi \in cl(\varphi)$, exactly one of ψ and neg(ψ) belong to α
  - For every $\psi_1 \wedge \psi_2 \in cl(\varphi)$, $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1 \in \alpha$ and $\psi_2 \in \alpha$

Atoms

- Intuition: an atom is a "maximal set of formulas of $cl(\varphi)$ that can be simultaneously true if one only knows the meaning of ¬ and ∧"
- A set $\alpha \subseteq cl(\varphi)$ is an atom if it satisfies the following conditions:
  - If $\mathbf{true} \in cl(\varphi)$, then $\mathbf{true} \in \boldsymbol{\alpha}$
  - For every $\psi \in cl(\varphi)$, exactly one of ψ and $\mathrm{neg}(\psi)$ belong to α
  - For every $\psi_1 \wedge \psi_2 \in cl(\varphi)$, $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1 \in \alpha$ and $\psi_2 \in \alpha$
- Examples of atoms for $\varphi = \neg(p \wedge q)\, \mathsf{U}\, \mathsf{F}p$:

$$\{\neg p, \neg q, \neg(p \wedge q), \mathsf{F}p, \varphi\} \quad \{p, q, (p \wedge q), \neg \mathsf{F}p, \neg\varphi\}$$

Atoms

- Intuition: an atom is a "maximal set of formulas of $cl(\varphi)$ that can be simultaneously true if one only knows the meaning of ¬ and ∧"
- A set $\alpha \subseteq cl(\varphi)$ is an atom if it satisfies the following conditions:
  - If $\mathbf{true} \in cl(\varphi)$, then $\mathbf{true} \in \boldsymbol{\alpha}$
  - For every $\psi \in cl(\varphi)$, exactly one of ψ and $\mathsf{neg}(\psi)$ belong to α
  - For every $\psi_1 \wedge \psi_2 \in cl(\varphi)$, $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1 \in \alpha$ and $\psi_2 \in \alpha$
- Examples of atoms for $\varphi = \neg(p \wedge q)\, \mathsf{U}\, \mathsf{F}p$:

$$\{\neg p, \neg q, \neg(p \wedge q), \mathsf{F}p, \varphi\} \quad \{p, q, (p \wedge q), \neg \mathsf{F}p, \neg\varphi\}$$

- Examples of non-atoms for $\varphi = \neg(p \wedge q)\, \mathsf{U}\, \mathsf{F}p$:

$$\{p, q, p \wedge q, \mathsf{F}p\} \quad \{\neg p, q, p \wedge q, \mathsf{F}p, \varphi\}$$

Atoms

- Intuition: an atom is a "maximal set of formulas of $cl(\varphi)$ that can be simultaneously true if one only knows the meaning of $\neg$ and $\wedge$"
- A set $\alpha \subseteq cl(\varphi)$ is an atom if it satisfies the following conditions:
  - If $\mathbf{true} \in cl(\varphi)$, then $\mathbf{true} \in \boldsymbol{\alpha}$
  - For every $\psi \in cl(\varphi)$, exactly one of ψ and $\mathrm{neg}(\psi)$ belong to α
  - For every $\psi_1 \wedge \psi_2 \in cl(\varphi)$, $\psi_1 \wedge \psi_2 \in \alpha$ iff $\psi_1 \in \alpha$ and $\psi_2 \in \alpha$
- Examples of atoms for $\varphi = \neg(p \wedge q) \, \mathrm{U} \, \mathrm{F}p$:

$$\{\neg p, \neg q, \neg(p \wedge q), \mathrm{F}p, \varphi\} \quad \{p, q, (p \wedge q), \neg \mathrm{F}p, \neg\varphi\}$$

- Examples of non-atoms for $\varphi = \neg(p \wedge q) \, \mathrm{U} \, \mathrm{F}p$:

$$\{p, q, p \wedge q, \mathrm{F}p\} \quad \{\neg p, q, p \wedge q, \mathrm{F}p, \varphi\}$$

- We have: all elements of a satisfaction sequence are atoms

Pre-Hintikka sequences

- A pre-Hinttika sequence for φ is a sequence $\alpha_0\alpha_1\alpha_2 \ldots$ of atoms satisfying the following conditions for every $i \geq 0$:
  - For every $\mathrm{X}\psi \in cl(\varphi)$:
    $\mathrm{X}\psi \in \alpha_i$ iff $\psi \in \alpha_{i+1}$
  - For every $\psi_1 \mathrm{U}\, \psi_2 \in cl(\varphi)$:
    $\psi_1 \mathrm{U}\, \psi_2 \in \alpha_i$ iff $\psi_2 \in \alpha_i$ or $\psi_1 \in \alpha_i$ and $\psi_1 \,\mathrm{U}\, \psi_2 \in \alpha_{i+1}$

Pre-Hintikka sequences

- A pre-Hinttika sequence for φ is a sequence $\alpha_0\alpha_1\alpha_2 \ldots$ of atoms satisfying the following conditions for every $i \geq 0$:
  - For every $\mathrm{X}\psi \in cl(\varphi)$:
    $\mathrm{X}\psi \in \alpha_i$ iff $\psi \in \alpha_{i+1}$
  - For every $\psi_1 \mathrm{U}\, \psi_2 \in cl(\varphi)$:
    $\psi_1 \mathrm{U}\, \psi_2 \in \alpha_i$ iff $\psi_2 \in \alpha_i$ or $\psi_1 \in \alpha_i$ and $\psi_1 \,\mathrm{U}\, \psi_2 \in \alpha_{i+1}$
- We have: every satisfaction sequence is a pre-Hintikka sequence.

Hintikka sequences

- A pre-Hinttika sequence $\alpha_0\alpha_1\alpha_2$ ...is a Hinttika sequence if it satisfies for every $i \geq 0$:
  - For every $\psi_1 \mathrm{U}\, \psi_2 \in cl(\varphi)$: if $\psi_1 \mathrm{U}\, \psi_2 \in \alpha_i$ then there exists $j \geq i$ such that $\psi_2 \in \alpha_j$
- We have: every satisfaction sequence is a Hintikka sequence.

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \cup (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

Hintikka sequences: An example

- Let $\varphi = \neg(p \land q) \cup (r \land s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

    1. $\{p, \neg q, r, s, \varphi\}^{\omega}$

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \cup (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

  1. $\{p, \neg q, r, s, \varphi\}^\omega$
  2. $\{\neg p, r, \neg\varphi\}^\omega$

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \cup (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

1. $\{p, \neg q, r, s, \varphi\}^{\omega}$
2. $\{\neg p, r, \neg\varphi\}^{\omega}$
3. $\{\neg p, q, \neg r, (r \wedge s), \neg\varphi\}^{\omega}$

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \, \mathrm{U} \, (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

1. $\{p, \neg q, r, s, \varphi\}^{\omega}$
2. $\{\neg p, r, \neg \varphi\}^{\omega}$
3. $\{\neg p, q, \neg r, (r \wedge s), \neg \varphi\}^{\omega}$
4. $\{p, q, (p \wedge q), r, s, (r \wedge s), \neg \varphi\}^{\omega}$

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \,\mathrm{U}\, (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?

1. $\{p, \neg q, r, s, \varphi\}^\omega$
2. $\{\neg p, r, \neg\varphi\}^\omega$
3. $\{\neg p, q, \neg r, (r \wedge s), \neg\varphi\}^\omega$
4. $\{p, q, (p \wedge q), r, s, (r \wedge s), \neg\varphi\}^\omega$
5. $\{p, \neg q, \neg(p \wedge q), \neg r, s, \neg(r \wedge s), \varphi\}^\omega$

Hintikka sequences: An example

- Let $\varphi = \neg(p \wedge q) \cup (r \wedge s)$. Which of the following are pre-Hintikka and Hintikka sequences ?
  1. $\{p, \neg q, r, s, \varphi\}^{\omega}$
  2. $\{\neg p, r, \neg\varphi\}^{\omega}$
  3. $\{\neg p, q, \neg r, (r \wedge s), \neg\varphi\}^{\omega}$
  4. $\{p, q, (p \wedge q), r, s, (r \wedge s), \neg\varphi\}^{\omega}$
  5. $\{p, \neg q, \neg(p \wedge q), \neg r, s, \neg(r \wedge s), \varphi\}^{\omega}$
  6. $\{p, q, (p \wedge q), r, s, (r \wedge s), \varphi\}^{\omega}$

Main theorem

- Definition: A Hintikka sequence $\alpha_0\alpha_1\alpha_2 \ldots$ extends a computation $s_0 s_1 s_2 \ldots$ if $s_i \cap cl(\varphi) = \alpha_i \cap AP$ for every $i \geq 0$.
- Theorem: Every computation $s_0 s_1 s_2 \ldots$ can be extended to a unique Hintikka sequence, and this extension is the satisfaction sequence.

Strategy for the NGA of a formula

- Let σ be a computation over AP.

Strategy for the NGA of a formula

- Let σ be a computation over AP.
- We have: $\sigma \models \varphi$

iff φ belongs to the first set of the satisfaction sequence for σ

iff φ belongs to the first set of the Hintikka sequence for σ

Strategy for the NGA of a formula

- Let σ be a computation over AP.
- We have: $\sigma \models \varphi$

  iff φ belongs to the first set of the satisfaction sequence for σ

  iff φ belongs to the first set of the Hintikka sequence for σ

- Strategy: design the NGA so that for every σ
  - The runs on σ correspond to the pre-Hintikka sequences $\alpha_0\alpha_1\alpha_2 \ldots$ that extend σ and satisfy $\varphi \in \alpha_0$
  - A run is accepting iff its corresponding pre-Hintikka sequence is also a Hintikka sequence.

The NGA A_φ

The NGA A_φ

- Alphabet: 2^{AP}

The NGA A_φ

- Alphabet: 2^{AP}
- States: atoms of φ.

The NGA A_φ

- Alphabet: 2^{AP}
- States: atoms of φ.
- Initial states: atoms containing φ.

The NGA A_φ

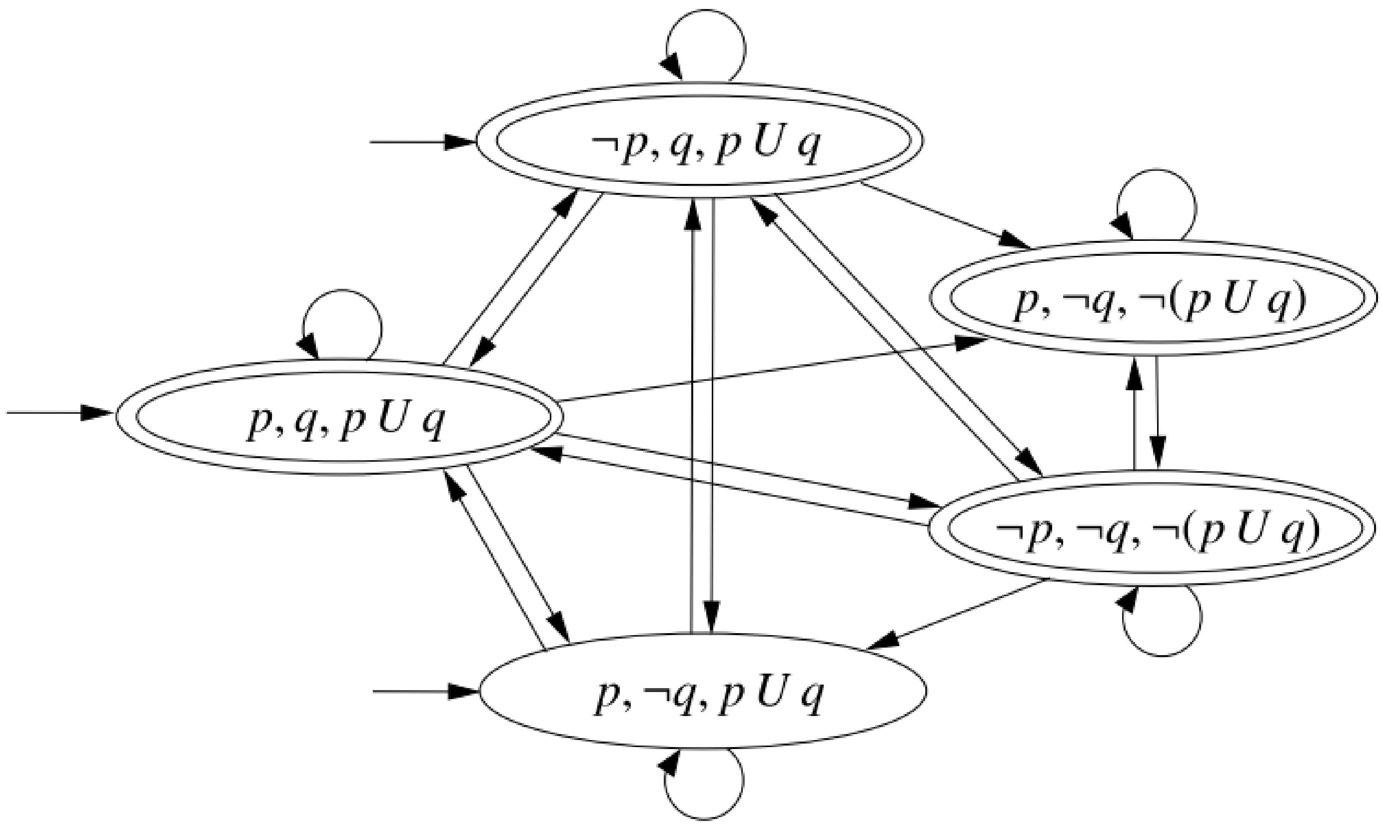
- Alphabet: 2^{AP}
- States: atoms of φ.
- Initial states: atoms containing φ.
- Transitions: triples $\alpha \xrightarrow{s} \beta$ such that $\alpha \cap AP = s$ and $\alpha\ \beta$ satisfies the conditions of a pre-Hintikka sequence.

The NGA A_φ

- Alphabet: 2^{AP}
- States: atoms of φ.
- Initial states: atoms containing φ.
- Transitions: triples $\alpha \xrightarrow{s} \beta$ such that $\alpha \cap AP = s$ and $\alpha\ \beta$ satisfies the conditions of a pre-Hintikka sequence.
- Sets of accepting states: A set $F_{\psi_1 U \psi_2}$ for every until-subformula $\psi_1 \mathrm{U} \psi_2$ of φ.

$F_{\psi_1 U \psi_2}$ contains the atoms α such that $\psi_1 \mathrm{U} \psi_2 \notin \alpha$ or $\psi_2 \in \alpha$.

Example: The NGA $A_{p \text{ U } q}$



(Labels of transitions omitted. The label of a transition from atom α is the set $\{p \in AP \mid p \in \alpha\}$. There is only one set of accepting states.)

Some observations

- All transitions leaving a state carry the same label.
- For every computation $s_0 s_1 s_2 \ldots$ satisfying φ there is a unique accepting run $\alpha_0 \xrightarrow{s_0} \alpha_1 \xrightarrow{s_1} \alpha_2 \xrightarrow{s_2} \cdots$, namely the one such that $\alpha_0 \alpha_1 \alpha_2 \ldots$ is the satisfaction sequence for $s_0 s_1 s_2 \ldots$.
- The sets of computations accepted from each initial state are pairwise disjoint.
- The number of states is bounded by $2^{|\varphi|}$.