Verification of liveness properties

Recall:

Safety: nothing bad can happen
Liveness: something good eventually happens

More formally:

- safety property: violations are finite executions

- liveness properties: violations are infinite executions

Approach:

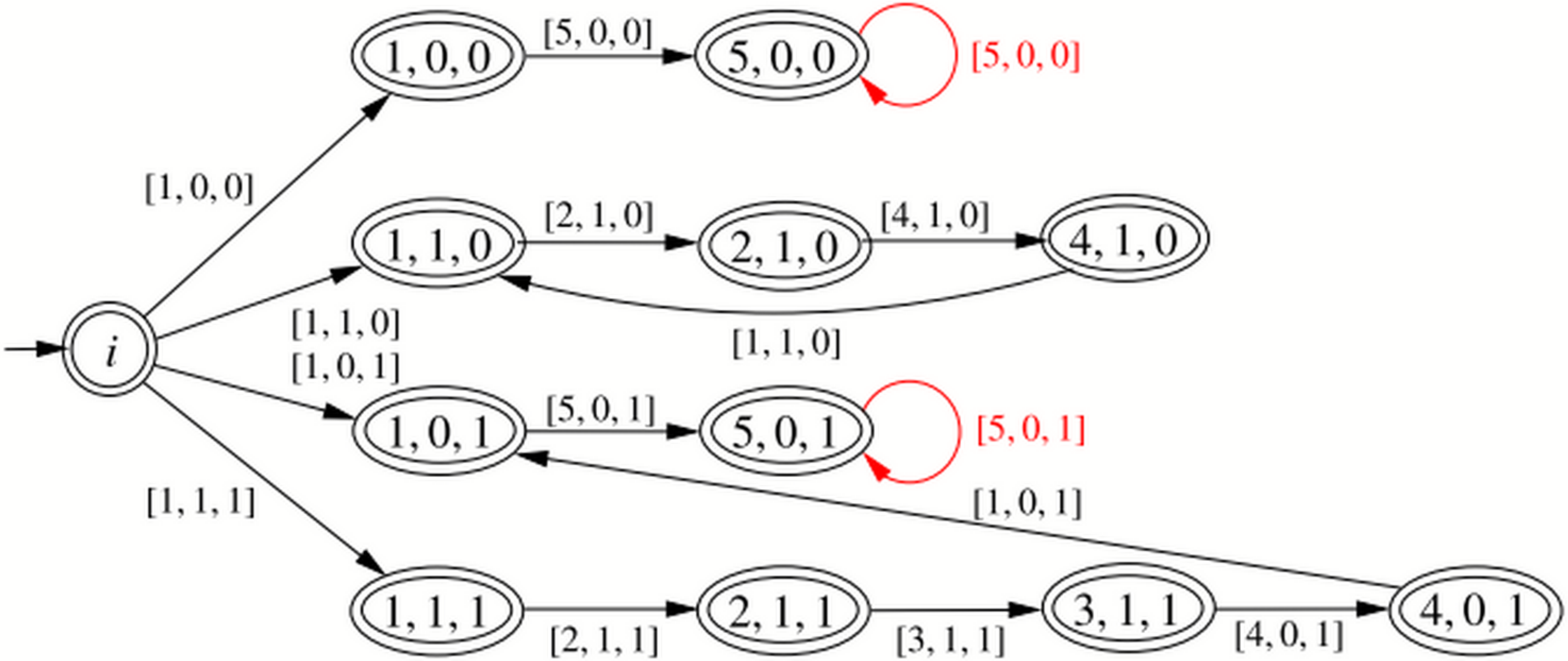
- represent the set of omega-executions as a NBA
  (the system NBA)

- represent the set of possible omega-executions violating
  the property as a NBA
  (the property NBA)

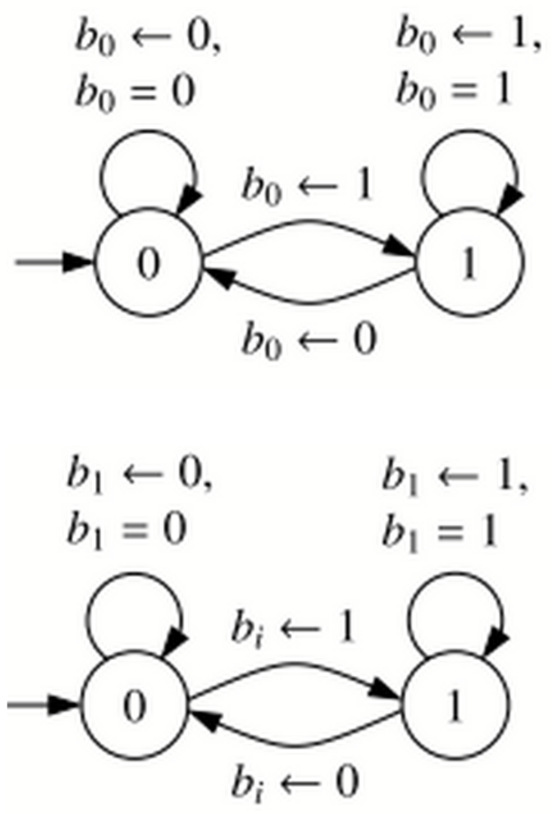
- check emptiness of the intersection of the two NBAs

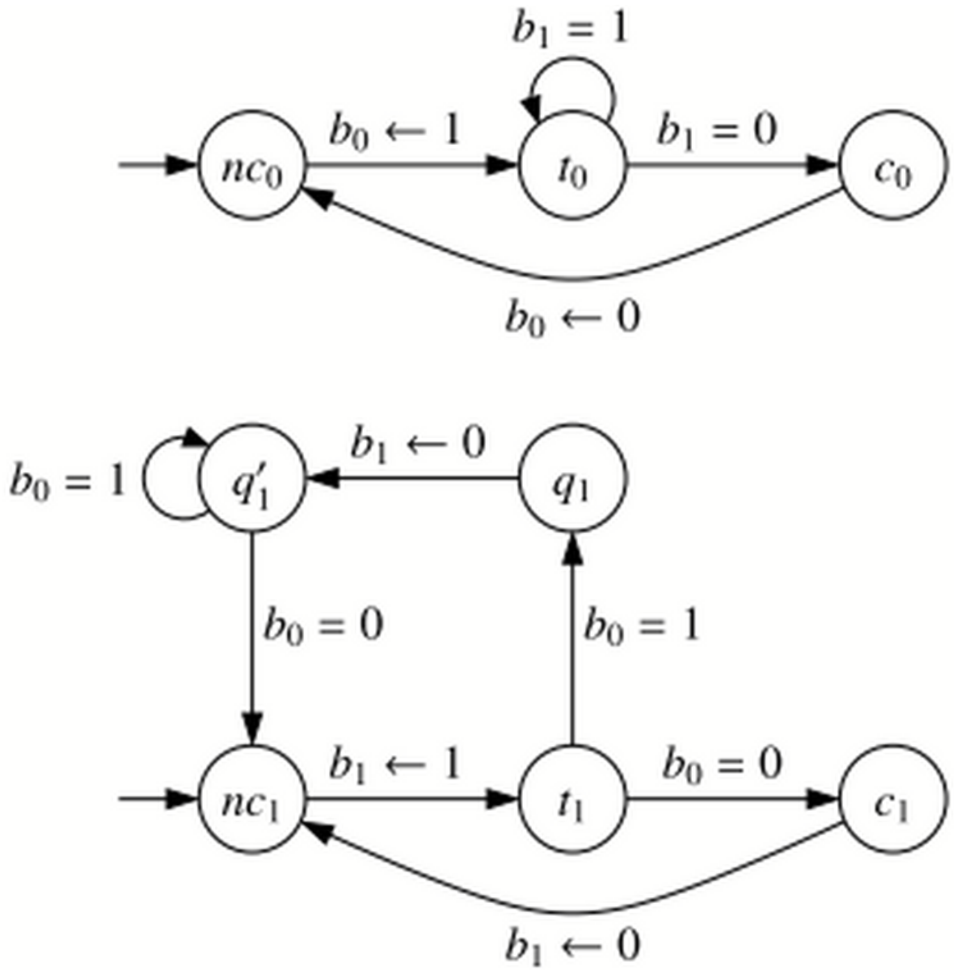
Some care needed when defining the omega-executions ...

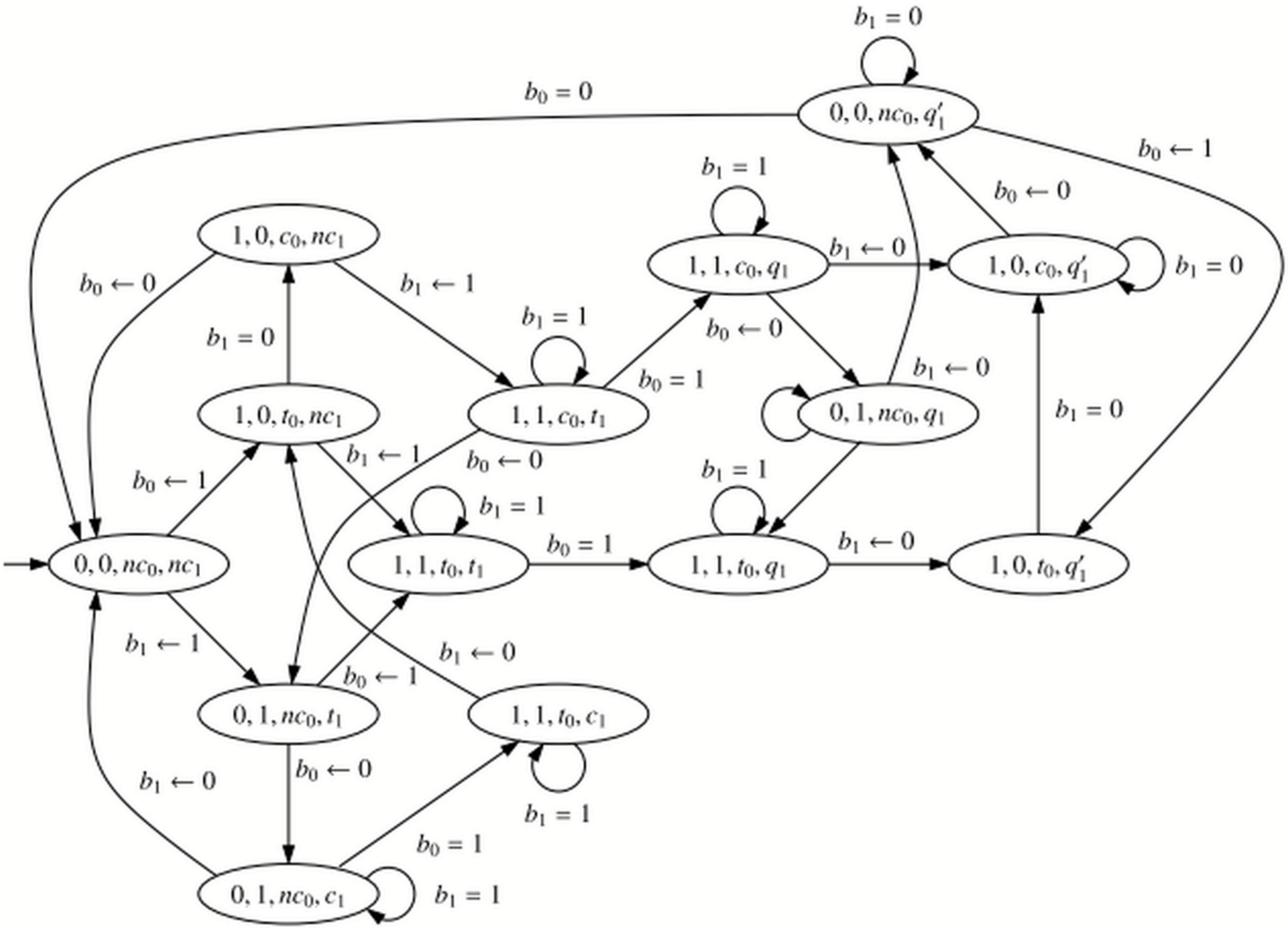
```
while x = 1 do
    if y = 1 then
        x ← 0
    y ← 1 − x
end
```



Lamport's algorithm







$b_1 = 0$
$b_0 = 0$
$0, 0, nc_0, q_1'$
$b_0 \leftarrow 1$
$b_1 = 1$
$b_0 \leftarrow 0$
$1, 0, c_0, nc_1$
$1, 1, c_0, q_1$
$b_1 \leftarrow 0$
$1, 0, c_0, q_1'$
$b_1 = 0$
$b_0 \leftarrow 0$
$b_1 \leftarrow 1$
$b_1 = 1$
$b_0 \leftarrow 0$
$b_1 = 0$
$b_1 \leftarrow 0$
$b_0 = 1$
$1, 0, t_0, nc_1$
$1, 1, c_0, t_1$
$0, 1, nc_0, q_1$
$b_1 = 0$
$b_1 \leftarrow 1$
$b_0 \leftarrow 0$
$b_0 \leftarrow 1$
$b_1 = 1$
$b_1 = 1$
$b_0 = 1$
$b_1 \leftarrow 0$
$0, 0, nc_0, nc_1$
$1, 1, t_0, t_1$
$1, 1, t_0, q_1$
$1, 0, t_0, q_1'$
$b_1 \leftarrow 1$
$b_1 \leftarrow 0$
$b_0 \leftarrow 1$
$0, 1, nc_0, t_1$
$1, 1, t_0, c_1$
$b_1 \leftarrow 0$
$b_0 \leftarrow 0$
$b_1 = 1$
$b_0 = 1$
$0, 1, nc_0, c_1$
$b_1 = 1$

"Finite waiting" property: if a process is trying to access the critical section, then it eventually will.

If NCi, Ti, Ci are the sets of configurations where process i is in the non-critical section, trying to access it, and in the critical section, respectively, then the possible violations are:

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

Observe: we can use the intersection algorithm for NFAs.

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

Is this a real problem of the algorithm ?
No! We have not properly specified the property!

Reformulate: in every omega-execution in which both processes execute actions infinitely often, if a process is trying to access the critical section, then it eventually will.

Fairness assumption: both processes execute infinitely many actions infinitely often

(Usually a weaker assumption is used: if a process can execute an action infinitely often, it will. In this case they are equivalent.)

The violations of the property are now the intersection of

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

and the omega-executions in which both processes "make a move" infinitely often.

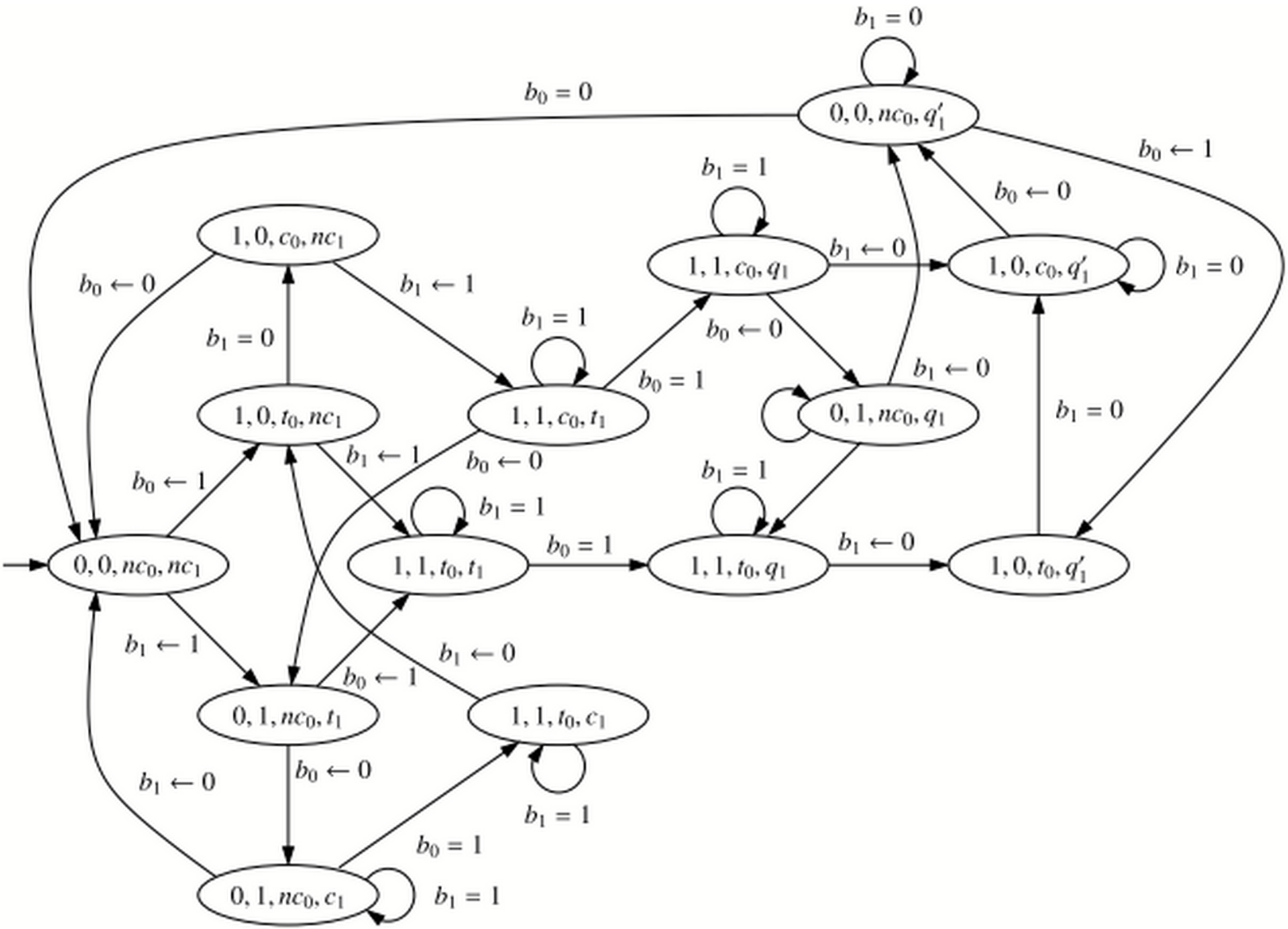
Problem: how to represent this as an regular omega-language?

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


$0, 0, nc_0, q'_1$
$b_1 = 0$
$b_0 = 0$
$b_0 \leftarrow 1$
$b_0 \leftarrow 0$
$1, 0, c_0, nc_1$
$b_0 \leftarrow 0$
$b_1 \leftarrow 1$
$b_1 = 1$
$1, 1, c_0, q_1$
$b_1 \leftarrow 0$
$1, 0, c_0, q'_1$
$b_1 = 0$
$b_1 = 0$
$b_1 = 1$
$b_0 \leftarrow 0$
$b_0 = 1$
$b_1 \leftarrow 0$
$1, 0, t_0, nc_1$
$1, 1, c_0, t_1$
$0, 1, nc_0, q_1$
$b_1 = 0$
$b_1 \leftarrow 1$
$b_0 \leftarrow 0$
$b_1 = 1$
$b_0 \leftarrow 1$
$b_1 = 1$
$0, 0, nc_0, nc_1$
$1, 1, t_0, t_1$
$b_0 = 1$
$1, 1, t_0, q_1$
$b_1 \leftarrow 0$
$1, 0, t_0, q'_1$
$b_1 \leftarrow 1$
$b_1 \leftarrow 0$
$b_0 \leftarrow 1$
$0, 1, nc_0, t_1$
$1, 1, t_0, c_1$
$b_1 \leftarrow 0$
$b_0 \leftarrow 0$
$b_1 = 1$
$b_0 = 1$
$0, 1, nc_0, c_1$
$b_1 = 1$

Denote by M0 the set of letters (c, 0)
M1 the set of letters (c,1)

The set of possible execution where both processes move infinitely often is now given by

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

The new finite waiting property holds for process 0 but not for process 1 because of

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

Temporal logic

Writing property NBAs requires training in automata theory.

We search for a more intuitive (but still formal) description language: Temporal Logic.

Temporal logic: extension of propositional logic with temporal operators like "always" and "eventually"

Linear temporal logic: temporal logic interpreted over linear structures.

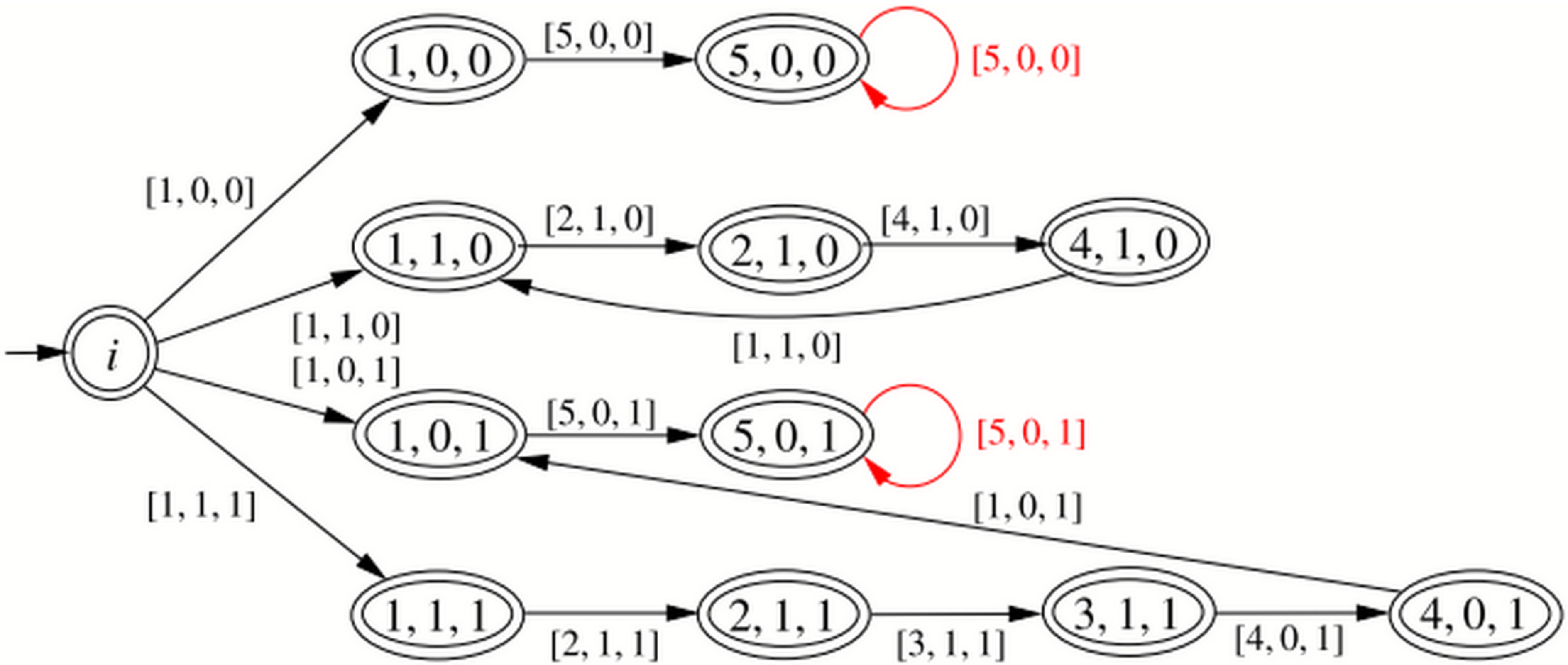
Linear Temporal Logic (LTL)

The setting is as follows. We are given:

- set AP of atomic propositions
  (names for basic properties)

- valuation assigning to each atomic proposition a set of configurations
  (configurations satisfying the property, assigns meaning to each name)

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



Example

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

$$AP = \{\texttt{at\_1}, \texttt{at\_2}, \ldots, \texttt{at\_5}, \texttt{x=0}, \texttt{x=1}, \texttt{y=0}, \texttt{y=1}\}$$

- $\mathcal{V}(\texttt{at\_i}) = \{[\ell, x, y] \in C \mid \ell = i\}$ for every $i \in \{1, \ldots, 5\}$.
- $\mathcal{V}(\texttt{x=0}) = \{[\ell, x, y] \in C \mid x = 0\}$, and similarly for $x = 1, y = 0, y = 1$.

An infinite sequence of subsets of AP is called a computation

Example: AP={p,q}

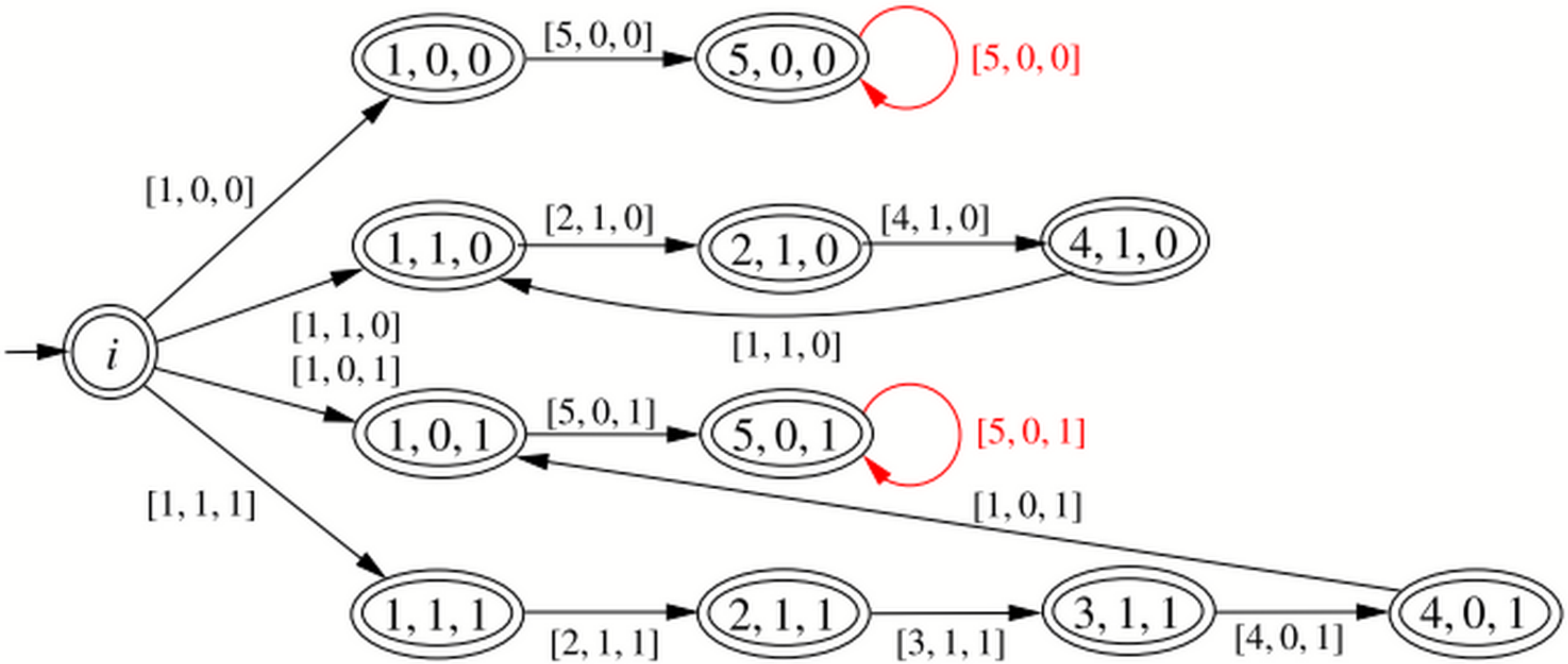
{p} {p,q} emp emp {p}^\omega

Every possible execution can be assigned a computation: just assign to every configuration the set of atomic propositions that are satisfied by it.

possible execution ====> computation
execution ====> executable computation

computation ====> in general set of possible executions

Example



$$
\begin{aligned}
e_1 &= [1,0,0]\ [5,0,0]^\omega \\
e_2 &= (\ [1,1,0]\ [2,1,0]\ [4,1,0]\ )^\omega \\
e_3 &= [1,0,1]\ [5,0,1]^\omega \\
e_4 &= [1,1,1]\ [2,1,1]\ [3,1,1]\ [4,0,1]\ [1,0,1]\ [5,0,1]^\omega
\end{aligned}
$$

$$
\begin{aligned}
e_1 &= [1,0,0]\ [5,0,0]^\omega \\
e_2 &= (\ [1,1,0]\ [2,1,0]\ [4,1,0]\ )^\omega \\
e_3 &= [1,0,1]\ [5,0,1]^\omega \\
e_4 &= [1,1,1]\ [2,1,1]\ [3,1,1]\ [4,0,1]\ [1,0,1]\ [5,0,1]^\omega
\end{aligned}
$$

$$
\begin{aligned}
\sigma_1 &= \{\mathtt{at\_1}, \mathtt{x{=}0}, \mathtt{y{=}0}\}\ \{\mathtt{at\_5}, \mathtt{x{=}0}, \mathtt{y{=}0}\}^\omega \\
\sigma_2 &= (\ \{\mathtt{at\_1}, \mathtt{x{=}1}, \mathtt{y{=}0}\}\ \{\mathtt{at\_2}, \mathtt{x{=}1}, \mathtt{y{=}0}\}\ \{\mathtt{at\_4}, \mathtt{x{=}1}, \mathtt{y{=}0}\}\ )^\omega \\
\sigma_3 &= \{\mathtt{at\_1}, \mathtt{x{=}0}, \mathtt{y{=}1}\}\ \{\mathtt{at\_5}, \mathtt{x{=}0}, \mathtt{y{=}1}\}^\omega \\
\sigma_4 &= \{\mathtt{at\_1}, \mathtt{x{=}1}, \mathtt{y{=}1}\}\ \{\mathtt{at\_2}, \mathtt{x{=}1}, \mathtt{y{=}1}\}\ \{\mathtt{at\_3}, \mathtt{x{=}1}, \mathtt{y{=}1}\}\ \{\mathtt{at\_4}, \mathtt{x{=}0}, \mathtt{y{=}1}\} \\
&\quad \{\mathtt{at\_1}, \mathtt{x{=}0}, \mathtt{y{=}1}\}\ \{\mathtt{at\_5}, \mathtt{x{=}0}, \mathtt{y{=}1}\}^\omega
\end{aligned}
$$

Syntax of LTL

Given: set AP of atomic propositions ("basic properties")
valuation assigning to each atomic proposition a set of configurations (configurations satisfying the property

Set of LTL formulas inductively defined as follows:

- **true** is a formula;
- p is a formula for every $p \in AP$; and
- if ϕ, ϕ_1 and ϕ_2 are formulas, then so are $\neg\phi$ and $\phi_1 \wedge \phi_2$; and
- if ϕ, ϕ_1 and ϕ_2 are formulas, then so are $X\,\phi$ and $\phi_1\,U\,\phi_2$.

Semantics of formulas

Formulas are interpreted on computations
The satisfaction relation is inductively defined as follows:

- $\sigma \models \mathbf{true}$.
- $\sigma \models p$ iff $p \in c(0)$.
- $\sigma \models \neg\phi$ iff $\sigma \not\models \phi$.
- $\sigma \models \phi_1 \wedge \phi_2$ iff $\sigma \models \phi_1$ and $\sigma \models \phi_2$.
- $\sigma \models X\phi$ iff $\sigma^1 \models \phi$.
- $\sigma \models \phi_1 \, U \, \phi_2$ iff there exists $k \geq 0$ such that $\sigma^k \models \phi_2$ and $\sigma^i \models \phi_1$ for all $0 \leq i < k$.

Abbreviations

- **false**, $\vee$, $\rightarrow$ and $\leftrightarrow$, interpreted in the usual way.
- $F\,\phi = \textbf{true}\,U\,\phi$ ("eventually ϕ"). According to the semantics above, $\sigma \models F\,\phi$ iff there exists $k \geq 0$ such that $\sigma^k \models \phi$,.
- $G\,\phi = \neg F \neg\phi$ ("always ϕ" or "globally ϕ"). According to the semantics above, $\sigma \models G\,\phi$ iff $\sigma^k \models \phi$ for every $k \geq 0$.

$AP = \{\texttt{at\_1}, \texttt{at\_2}, \ldots, \texttt{at\_5}, \texttt{x=0}, \texttt{x=1}, \texttt{y=0}, \texttt{y=1}\}$

- $\mathcal{V}(\texttt{at\_i}) = \{[\ell, x, y] \in C \mid \ell = i\}$ for every $i \in \{1, \ldots, 5\}$.
- $\mathcal{V}(\texttt{x=0}) = \{[\ell, x, y] \in C \mid x = 0\}$, and similarly for $x = 1, y = 0, y = 1$.

- $\phi_0 \;=\; \texttt{x=1} \wedge X\, \texttt{y=1} \wedge X\, X\, \texttt{at\_3}$.

  $\phi_1 \;=\; F\, \texttt{x=0}$.

  $\phi_2 \;=\; \texttt{x=0}\, U\, \texttt{at\_5}$.

  $\phi_3 \;=\; \texttt{y=1} \wedge F\,(\texttt{y=0} \wedge \texttt{at\_5}) \;\wedge\; \neg(F\,(\texttt{y=0} \wedge X\, \texttt{y=1}))$.

Lamport's algorithm

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

and valuation as expected.

Mutual exclusion:

Naive finite waiting:

Finite waiting with fairness:

The bounded overtaking property for process 0 is expressed by

$$G\,(\,T_0 \rightarrow (\neg C_1\; U\,(C_1\; U\,(\neg C_1\; U\; C_0))))$$

The formula states that whenever T_0 holds, the computation continues with a (possibly empty!) interval at which we see $\neg C_1$ holds, followed by a (possibly empty!) interval at which C_1 holds, followed by a point at which C_0 holds. The

Getting used to LTL ...

Express in natural language: FG p GF p

Are these pairs of formulas equivalent?

FF p	Fp	GGp	Gp
FG p	GF p	FGF p	GF p
p U q	p U (p/\q)		
Fp	p \/ XF p	Fp	p /\ XF p
Gp	p \/ XG p	Gp	p \/ GF p
p U q	p \/ X(p U q)	p U q	p /\ X (p U q)
p U q	q \/ X(p U q)	p U q	q /\ X (p U q)
p U q	q \/ (p /\ X(p U q))	p U q	q /\ (p \/ X(p U q))

From formulas to NBAs

Given: set AP of atomic propositions

Language L(f) of a formula f: set of computations satisfying f

Examples for AP={p,q}

L(F p) =   s1 s2 s3 ... where p belongs to si for some i

L(G (p /\ q)) =   {p, q}^\omega

L(f) is an omega-language over the alphabet 2^AP

For AP = {p,q}, 2^AP ={ emp, {p}, {q}, {p,q} }

NBAs for some formulas

AP = {p, q}

F p

G p

p U q

GF p

From LTL formulas to NGAs

We present an algorithm that takes a formula f as input (over a fixed set AP) and returns a NGA A_f such that L(A_f) = L(f).

Closure cl(f) of a formula f: set containing all subformulas of f and their "negations".

Closure of p U ~q: { p, ~p, ~q, q, p U ~q, ~(p U q) }

Satisfaction sequence of a computation wrt a formula:

- add to each subset of the computation the formulas of the closure that hold

Take $\{p\}^\omega$ and $p \, U \, q$. The satisfaction sequence is

Take $(\{p\}\{q\})^\omega$ and $p \, U \, q$. The satisfaction sequence is

Definition 14.8 *A* pre-Hintikka sequence *for ϕ is an infinite sequence $\alpha = \alpha_0\alpha_1\alpha_2\ldots$ of atoms satisfying the following conditions for every $i \geq 0$:*

(l1) For every $X\phi \in cl(\phi)$: $X\phi \in \alpha_i$ if and only if $\phi \in \alpha_{i+1}$.

(l2) For every $\phi_1 U \phi_2 \in cl(\phi)$: $\phi_1 U \phi_2 \in \alpha_i$ if and only if $\phi_2 \in \alpha_i$ or $\phi_1 \in \alpha$ and $\phi_1 U \phi_2 \in \alpha_{i+1}$.

A pre-Hintikka sequence is a Hintikka sequence *if it also satisfies*

(g) For every $\phi_1 U \phi_2 \in \alpha_i$, there exists $j \geq i$ such that $\phi_2 \in \alpha_j$.

A pre-Hintikka or Hintikka sequence α matches *a computation σ if for every atomic proposition $p \in AP$, $p \in \alpha_i$ if and only if $p \in \sigma_i$.*[1]

Main theorem: the satisfaction sequence for a formula and a computation is the unique Hintikka sequence that can be obtained by adding formulas of the closure to the subsets of the computation.

Strategy for constructing the NGA

We have:

sigma satisfies f
iff
f belongs to the first subset of the sat. sequence
iff
f belongs to the first subset of the Hintikka sequence

Strategy: design the NGA so that
- the runs correspond to the pre-Hintikka sequences such that f belongs to the first subset, and
- a run is accepting iff its pre-Hintikka sequence is also a Hintikka sequence

- The alphabet of A_ϕ is 2^{AP}.
- The states of A_ϕ (apart from the distinguished initial state q_0) are atoms of ϕ.
- The output transitions of the initial state q_0 are the triples $q_0 \xrightarrow{\sigma_0} \alpha$ such that σ_0 matches α (i.e., for every atomic proposition $p \in AP$, $p \in \sigma_0$ if and only if $p \in \alpha_0$), and $\phi \in \alpha$.
- The output transitions of any other state α (where α is an atom) are the triples $\alpha \xrightarrow{\sigma} \beta$ such that σ matches β, and the pair α, β satisfies conditions (l1) and (l2) (where α and β play the roles of α_i resp. α_{i+1}).
- The accepting condition contains a set $F_{\phi_1 \, U \, \phi_2}$ of accepting states for each subformula $\phi_1 \, U \, \phi_2$ of ϕ. An atom belongs to $F_{\phi_1 \, U \, \phi_2}$ if it does not contain $\phi_1 \, U \, \phi_2$ or if it contains ϕ_2.

NGA (NBA) for the formula p U q