Session 8
Algorithms for Model Checking

Contents

1 Algorithms for Model Checking LTL using automata

Algorithms for Model Checking for LTL

Problem: Given a formula $\varphi \in LTL$, a model (transition system) $\mathcal{M} = (S, \longrightarrow, AP, L)$ and a state s, check if

$$\mathcal{M}, s \models \varphi$$

- This means, that for all $\pi \in Paths(s)$, $\pi \models \varphi$.
- and thus model checking is just a procedure to verify that the following inclusion holds:
$$Paths(s) \subseteq \{\pi \mid \pi \models \varphi\},$$
- that is
$$Paths(s) \cap \overline{\{\pi \mid \pi \models \varphi\}} = \emptyset.$$

Algorithms for Model Checking for LTL based on Automata

- Build an automata $\mathcal{A}_{\neg\varphi}$ that has an accepting run π if and only if $\pi \models \neg\varphi$, i.e. $\pi \not\models \varphi$:
$$\mathcal{L}(\mathcal{A}_{\neg\varphi}) = \{trace(\pi) \mid \pi \not\models \varphi\} = \overline{\{trace(\pi) \mid \pi \models \varphi\}}.$$
- Represent $(\mathcal{M}, s)$ by an automaton $\mathcal{A}_{\mathcal{M},s}$, that accepts exactly the traces (paths) of $\mathcal{M}$ that start on s:
$$\mathcal{L}(\mathcal{A}_{\mathcal{M},s}) = \{trace(\pi) \mid \pi \in Paths(s)\}$$

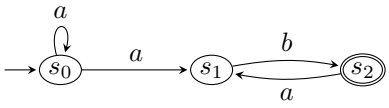
- Check that
$$\mathcal{L}(\mathcal{A}_{\neg\varphi}) \cap \mathcal{L}(\mathcal{A}_{\mathcal{M},s}) = \emptyset$$
If it is *true*, we have
$$\mathcal{M}, s \models \varphi,$$
otherwise any trace σ that belongs to the intersection of the two languages
$$\sigma \in \mathcal{L}(\mathcal{A}_{\neg\varphi}) \cap \mathcal{L}(\mathcal{A}_{\mathcal{M},s})$$
can be used as a *counterexample*.

Automata over finite words (NFA)

Σ alphabet, $L \subseteq \Sigma^\star$ language
$\mathcal{A} = (\Sigma, S, S^0, \delta, F)$ where $S^0, F \subseteq S$, $\delta : S \times \Sigma \to 2^S$

$$L(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(s_0, w) \cap F \neq \emptyset \wedge s_o \in S^0\}$$

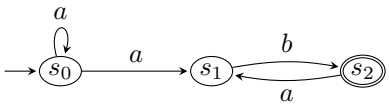
where $\delta(s, \varepsilon) = s$ and $\delta(s, ax) = \delta(\delta(s, a), x)$.



$$aaaababababab \in L(\mathcal{A})$$
$$L((a^*ab(ab)^*) = L(\mathcal{A})$$

Languages accepted by NFAs are closed under complementation, union, and intersection.

Automata over Infinite Words (Büchi Automata)



The word
$$w = aaababab\ldots$$
is *accepted* (only acceptance condition changes...)

1.1 Automata over Infinite Words – Büchi Automata

Infinite Words

Given an alphabet Σ, a infinite word over Σ is a infinite sequence

$$a_0 a_1 a_2 \cdots a_m a_{m+1} \cdots$$

of symbols $a_i \in \Sigma$.

The set of infinite words over Σ is represented by Σ^ω. The word

$$aaabbbaaabbb \ldots$$

can be represented by the expression

$$(aaabbb)^\omega$$

Büchi Automaton

$\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$ where $S^0, F \subseteq S$, $\delta : S \times \Sigma \to 2^S$
Given $w = a_0 a_1 \ldots$, a run on w (r_w) is a sequence of states

$$s_0, s_1, \ldots,$$

with $s_0 \in S^0$ e $s_{i+1} \in \delta(s_i, a_i)$, $i \geq 0$. Let

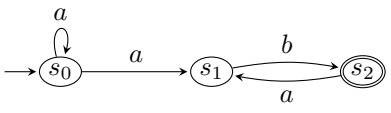
$$lim(r_w) = \{s \mid s = s_i \text{ for an infinite number of } i's\}$$

i.e., the set of states that occur on r_w *infinitely often.*
A run r_w is *accepting* if $lim(r_w) \cap F \neq \emptyset$
The *accepted language* of $\mathcal{A}_\omega$ is

$$\mathcal{L}(\mathcal{A}_\omega) = \{w \in \Sigma^\omega \mid \exists r_w \; lim(r_w) \cap F \neq \emptyset\}$$

i.e. words which runs have infinite many final states.

Büchi Automaton: automaton over infinite words



$w = aaababab \cdots$ is accepted and

$$\mathcal{L}(\mathcal{A}_\omega) = L_\omega((a^*(ab)^\omega))$$

Closures

Theorem 1. *The set of languages accepted by Büchi automata are closed for union, intersection, and complementation.*

Let $\mathcal{A}_i = (\Sigma, S_i, S_i^0, \delta_i, F_i)$, $i = 1, 2$ where S_1 and S_2 are disjoint.

- for union one can built $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 = (\Sigma, S_1 \cup S_2, S_1^0 \cup S_2^0, \delta_1 \cup \delta_2, F_1 \cup F_2)$
- for intersection the usual the product construction for automata over finite words cannot be used. The accepting states cannot be $F_1 \times F_2$ as each automaton can go through an accepting state in different moments. Let

  $$\mathcal{A}_\cap = (\Sigma, S_1 \times S_2 \times \{1, 2\}, S_1^0 \times S_2^0 \times \{1\}, \delta_\cap, F_1 \times F_2 \times \{1\})$$

  and $(s', t', j) \in \delta((s, t, i), a)$ if $s' \in \delta_1(s, a)$, $t' \in \delta_2(t, a)$, and $i = j$ but if $i = 1$ and $s \in F_1$, then $j = 2$ and if $i = 2$ and $t \in F_2$, then $j = 1$.
- for the complement, algorithms can be *doubly exponential*.

ω-Regular Languages

Theorem 2. *A ω-language L is accepted by a Büchi automaton $\mathcal{A}$ iff L is the finite union of languages VW^ω, where $V, W \subseteq \Sigma^\star$ are regular languages (over finite words) and $\varepsilon \notin W$.*

Proof. $\subseteq$:

- Büchi automata are closed for union
- If L is a regular language and $\varepsilon \notin L$, then L^ω is accepted by a Büchi automaton
- If L is a regular language and $\mathcal{A}$ a Büchi automaton then $L \cdot \mathcal{L}_\omega(\mathcal{A})$ is accepted by a Büchi automaton

$\supseteq$: $\mathcal{L}_\omega(\mathcal{A}) = \bigcup_{s_0 \in S_0, s \in F} \mathcal{L}_{s_0 s}(\mathcal{L}_{ss} \setminus \varepsilon)^\omega$ □

Determinism

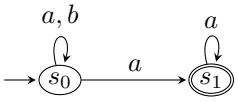
$\mathcal{A} = (\Sigma, S, S^0, \delta, F)$ is *deterministic* if

$$|S^0| \leq 1 \text{ and } |\delta(s, a)| \leq 1$$

for all $s \in S$ and $a \in \Sigma$.

In this case, determinism is weaker than nondeterminism.

The language $(a + b)^* a^\omega$ is not accepted by a deterministic Büchi automaton. An NBA is



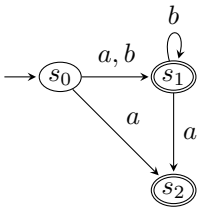
Note that in a DBA the words $a^{n_1}ba^{n_2}\cdots ba^{n_i}$ for $n_i \geq 1$ must reach different final states (one for each i) but there can be only a finite number of states thus there will be for distinct i and j words that reach a same final state. One concludes that $a^{n_1}b\cdots ba^{n_i}(ba^{n_{i+1}}b\cdots ba^{n_j})^\omega$ would belong to $\mathcal{L}((a+b)^*a^\omega)$.

Büchi automata

Consider the Büchi automaton

$$\mathcal{A}_\omega = (\Sigma = \{a, b\}, \{s_0, s_1, s_2\}, \{s_0\}, \delta, \{s_1, s_2\}),$$

where $\delta(s_0, a) = \{s_1, s_2\}$, $\delta(s_0, b) = \{s_1\}$, $\delta(s_1, a) = \{s_2\}$, $\delta(s_1, b) = \{s_1\}$, $\delta(s_2, a) = \delta(s_2, b) = \emptyset$.
Determine $L(\mathcal{A}_\omega) \subseteq \Sigma^\omega$.



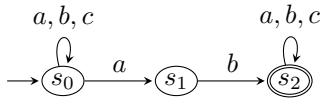
$\mathcal{A}_\omega$

$$L(\mathcal{A}_\omega) = L((a+b)b^\omega)$$

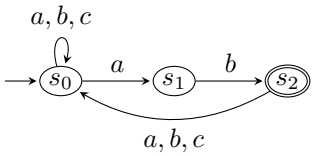
For each language, build a Büchi automaton that accepts it for $\Sigma = \{a, b, c\}$.

1. $L_1 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains at least one infix } ab\}$;
2. $L_2 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains the infix } ab \text{ infinitely often}\}$.

$L_1 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains at least one infix } ab\}$;



$L_2 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains the infix } ab \text{ infinitely often}\}$



$L_\omega(\mathcal{A}_\omega) = \emptyset$ **?** **e** $L_\omega(\mathcal{A}_\omega) = \Sigma^\omega$

Theorem 3. *The problem of deciding if the accepted language of a Büchi automaton $\mathcal{A}_\omega$ is empty can be decided in linear time.*

Proof. Given $\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$, $L_\omega(\mathcal{A}_\omega)$ is nonempty iff there exists $s_0 \in S^0$ and $t \in F$ such that s_0 is connected to t e t is connected to himself. □

Theorem 4. *The problem of deciding if the accepted language of a Büchi automaton $\mathcal{A}_\omega$ is Σ^ω can be decided in exponential time.*

Proof. The complementary automaton of a Büchi automaton $\mathcal{A}$ is exponentially larger than $\mathcal{A}$ and $L_\omega(\mathcal{A}) \neq \Sigma^\omega \leftrightarrow L_\omega(\overline{\mathcal{A}}) \neq \emptyset$. □

1.2 Transition Systems and Büchi Automata

Transition Systems and Büchi Automata

Let $\mathcal{M} = (S, \longrightarrow, AP, L)$ be a transition system with no terminal states (model) over the set AP of propositional variables. A trace

$$\sigma = L(s_0)L(s_1)\cdots$$

is an infinite sequence of subsets of AP. Given a model $\mathcal{M}$ and $s_0 \in S$ we associate the NBA

$$A_M = (2^{AP}, S, \{s_0\}, \delta, S),$$

such that $s' \in \delta(s, A)$ iff $s \longrightarrow s'$ and $A = L(s)$.

As the set of accepting states is S (all states are accepting), any path π is an accepting run and thus the accepted language of A_M are exactly the set of traces of $\mathcal{M}$,

$$L_\omega(A_M) = \{\sigma \mid \sigma \in Traces(\mathcal{M})\}.$$

Example

Consider the model $T = (S, \rightarrow, \{a, b\}, \{q_1\}, L)$ with

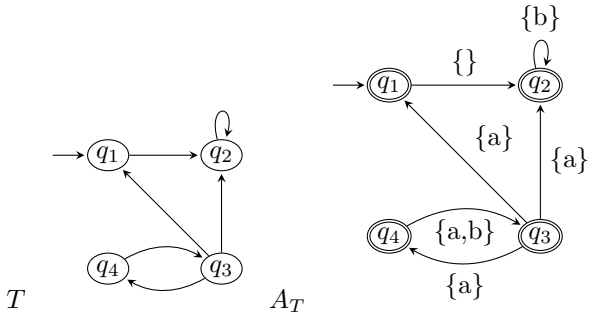
- $S = \{q_1, q_2, q_3, q_4\}$,
- $\rightarrow = \{q_1 \rightarrow q_2, q_2 \rightarrow q_2, q_3 \rightarrow q_1, q_3 \rightarrow q_2, q_3 \rightarrow q_4, q_4 \rightarrow q_3\}$,
- and $L(q_1) = \{\}, L(q_2) = \{b\}, L(q_3) = \{a\}, L(q_4) = \{a, b\})$.

The model T can by represented by the determinístic Büchi automaton

$$\mathcal{A}_T = (2^{\{a,b\}}, \{q_1, q_2, q_3, q_4\}, \delta, \{q_1\}, S)$$

where

$$\begin{aligned} \delta(q_1, \{\}) &= \{q_2\} \\ \delta(q_2, \{b\}) &= \{q_2\} \\ \delta(q_3, \{a\}) &= \{q_1, q_2, q_4\} \\ \delta(q_4, \{a, b\}) &= \{q_3\} \end{aligned}$$



1.3 Translating LTL formulae to Büchi Automata

LTL and Büchi Automata

- Each LTL formula φ can be associated with a Büchi automaton φ, $\mathcal{A}_\varphi$,
- The accepted language $L_\omega(\mathcal{A}_\varphi)$ is exactly the set of traces (paths) that satisfy the formula φ.
- Considering $\Sigma = 2^{AP}$ it is easy to associate LTL formulas to Büchi automata
- But, note that this method is computationally inefficient.
- $a \in \Sigma$ corresponds to a valuation of the propositional variables of AP.
- For a propositional formula φ (only with $\wedge$, $\vee$ and $\neg$) let

$$\Sigma_\varphi = \{a \in \Sigma \mid a \models \varphi\}$$

.

- If $AP = \{p, q\}$, $\Sigma = \{\{\}, \{p\}, \{q\}, \{p, q\}\}$, and $\Sigma_{p \vee q} = \{\{p\}, \{q\}, \{p, q\}\}$.

Propositional formula and Büchi Automata

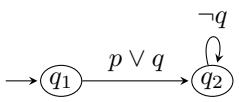
For example if $p \in AP$,

$$\begin{aligned} \Sigma_p &= \{a \in \Sigma \mid p \in a\} \\ \Sigma_{\neg p} &= \Sigma \setminus \Sigma_p \\ \Sigma_{p \wedge q} &= \Sigma_p \cap \Sigma_q \\ \Sigma_{p \vee q} &= \Sigma_p \cup \Sigma_q \\ \Sigma_{p \to q} &= \Sigma \setminus \Sigma_p \cup \Sigma_q \end{aligned}$$

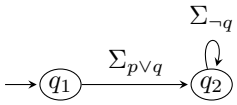
Given two states s and s' and φ a propositional formula, let

$$s \xrightarrow{\Sigma_\varphi} s' = \{s \xrightarrow{a} s' \mid a \in \Sigma_\varphi\}$$

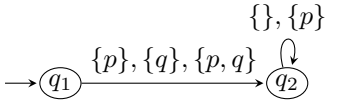
This means that $s \xrightarrow{\Sigma_\varphi} s'$ is an abbreviation of a set of transitions. We also can just write $s \xrightarrow{\varphi} s'$.
If $AP = \{p, q\}$, $\Sigma = \{\{\}, \{p\}, \{q\}, \{p, q\}\}$ and



or



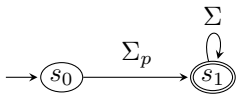
correspond to



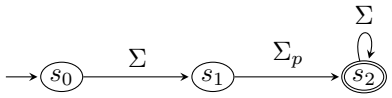
LTL formulae and Büchi Automata

Here are some examples of automata corresponding to simple LTL formulae (where p can be substituted by any propositional formula)

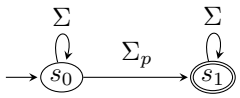
p



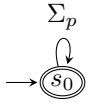
Xp



Fp

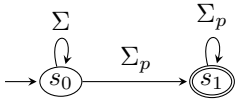


Gp

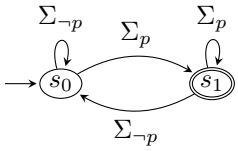


LTL formulae and Büchi Automata

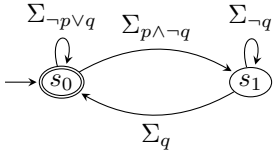
FGp



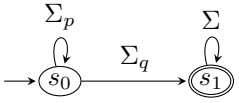
GFp



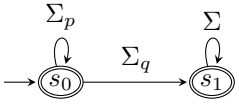
G($p \rightarrow$ Fq)



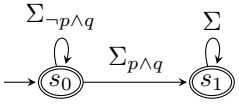
$p\mathrm{U}q$



$p\mathrm{W}q$



$p\mathrm{R}q$



- Because Büchi automata are closed under union, intersection and complementation, an automaton can be constructed for any LTL formula.
- However due to the fact that complementation yields an exponential blowup of the number of states of the resulting automaton this naïve method is not used in practical model checkers. In general, either generalised or alternating Büchi automata are used.

Algorithm of *Model Checking* for LTL

The problem of checking if a model satisties a formula φ, $M, s_o \models \varphi$ reduces to decide if

$$L_\omega(A_M) \subseteq L_\omega(A_\varphi)$$

Or equivalently,

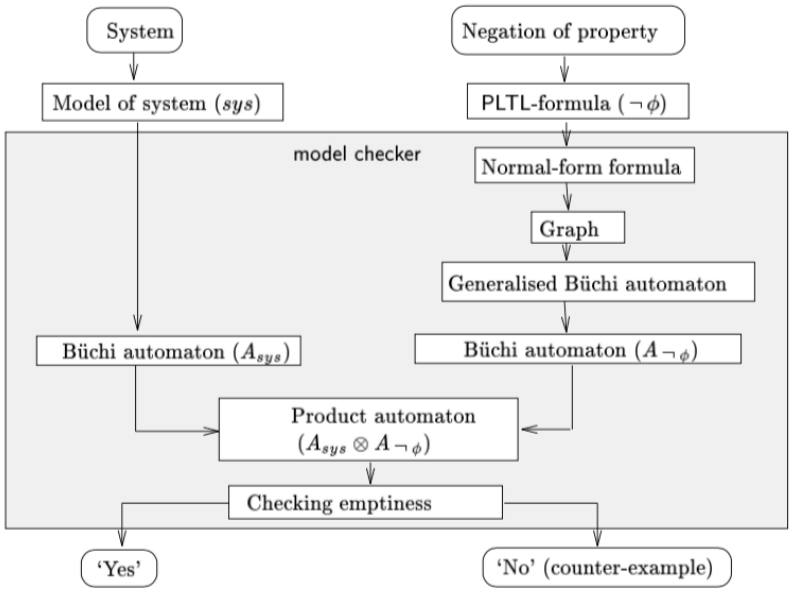
$$L_\omega(A_M) \cap L_\omega(\overline{A_\varphi}) = \emptyset$$

where

The automaton for intersection can have $|S|.2^{\mathcal{O}(|\varphi|)}$ states.

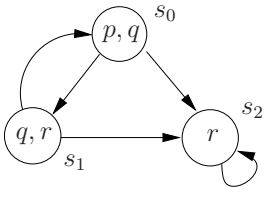
Thus model checking can be achieved in time $O(|S|.2^{\mathcal{O}(|\varphi|)})$. As the specification is is general short, the algorithms are relatively efficient.

Algorithm of *Model Checking* for LTL, using Generalised Büchi Automata



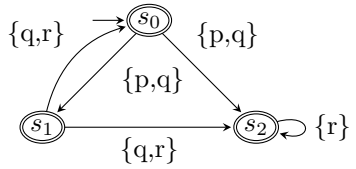
Exercício 8.1. *Consider the system* $\mathcal{M} = (S = \{s_0, s_1, s_2\}, \{s_0 \longrightarrow s_1, s_0 \longrightarrow s_2, s_1 \longrightarrow s_2, s_1 \longrightarrow s_0, s_2 \longrightarrow s_2\}, L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\})$*, with* $AP = \{p, q, r\}$ *Determine which relations are true*

1. $\mathcal{M}, s_0 \models p \wedge q$
2. $\mathcal{M}, s_0 \models Xr$
3. $\mathcal{M}, s_0 \models X(q \wedge r)$
4. $\mathcal{M}, s_0 \models G\neg(p \wedge r)$
5. $\mathcal{M}, s_0 \models GFp$
6. $\mathcal{M}, s_0 \models GFp \rightarrow GFr$



◇

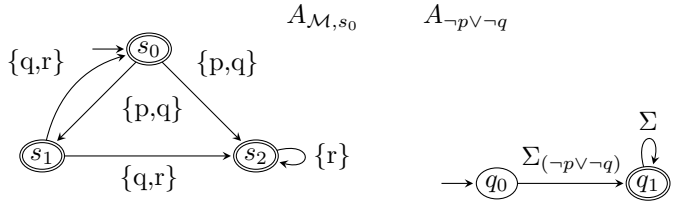
We have $\Sigma = \{\emptyset, \{p\}, \{r\}, \{q\}, \{p, q\}, \{p, r\}, \{q, r\}, \{p, q, r\}\}$.
The automaton for $\mathcal{M}, s_0$ is $A_{\mathcal{M},s_0}$



i. We have $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ and

$$\begin{aligned} \Sigma_{(\neg p \vee \neg q)} &= \{a \in \Sigma \mid a \models (\neg p \vee \neg q)\} \\ &= \{\emptyset, \{p\}, \{r\}, \{q\}, \{p, r\}, \{q, r\}\} \end{aligned}$$

ii. The corresponding Büchi automaton is $A_{(\neg p \vee \neg q)}$

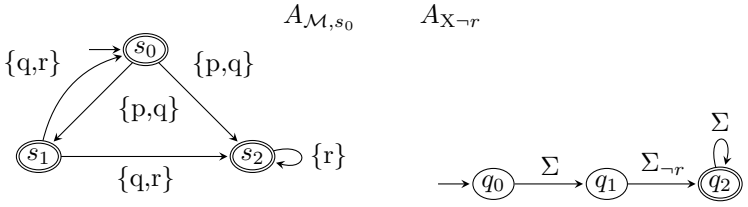


iii. It is easy to see that

$$L_\omega(A_M) \cap L_\omega(A_{(\neg p \vee \neg q)}) = \emptyset$$

as the first symbol of any accepting word (run) in A_M is $\{p, q\}$ and any accepting word in $A_{(\neg p \vee \neg q)}$ cannot start with $\{p, q\}$.

The automaton for $\mathcal{M}, s_0$ is $A_{\mathcal{M},s_0}$ (on the right) and on the left $A_{\mathrm{X}\neg r}$ is the automaton for $\neg \mathrm{X}r \equiv \mathrm{X}(\neg r)$.



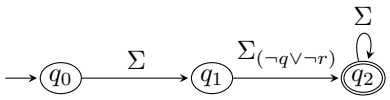
It is easy to see that

$$L_\omega(A_{M,s_0}) \cap L_\omega(A_{\mathrm{X}\neg r}) = \emptyset$$

as the second symbol of any word (run) in A_{M,s_0} contains r. Alternatively one could build the automaton for intersection and test for emptiness.

i. We have $\neg \mathrm{X}(q \wedge r) \equiv \mathrm{X}(\neg q \vee \neg r)$

ii. As before, $A_{\mathrm{X}(\neg q \vee \neg r)}$ is



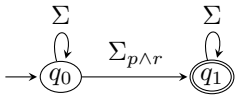
iii. It is easy to see that

$$L_\omega(A_M) \cap L_\omega(A_{\mathrm{X}(\neg q \vee \neg r)}) \neq \emptyset$$

as the word $\{p, q\}(\{r\})^\omega$ belongs to the intersection and is a counterexample.

i. We have $\neg\mathrm{G}\neg(p \wedge r) \equiv \mathrm{F}(p \wedge r)$

ii. We have $\Sigma_{p \wedge r} = \Sigma_p \cap \Sigma_r = \{\{p, r\}, \{p, q, r\}\}$. The automaton $A_{\mathrm{F}(p \wedge r)}$ can be



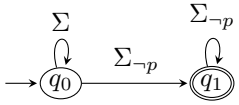
iii. We have

$$L_\omega(A_M) \cap L_\omega(A_{\mathrm{F}p \wedge r}) = \emptyset$$

as no (infinite) run in A_M as either the symbol $\{p, r\}$ or $\{p, q, r\}$.

i. We have $\neg\mathrm{GF}p \equiv \mathrm{FG}\neg p$

ii. We have $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p = \{\{\}, \{q\}, \{r\}, \{q, r\}\}$. The automaton $A_{\mathrm{FG}\neg p}$ can be



iii. It is easy to see that

$$L_\omega(A_M) \cap L_\omega(A_{\mathrm{FG}(\neg p)}) \neq \emptyset$$

as the word $\{p, q\}(\{r\})^\omega$ belongs to the intersection and is a counterexample.

2 Never Claims in SPIN

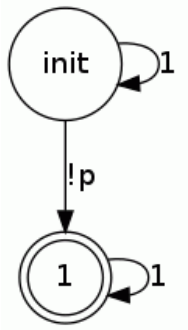
Never Claims

- Spin translates each negation of LTL formula into a *never claim* that is a Promela program (Büchi automaton) that states something that never should happen.

- A *never claim* runs in parallel with the model and the first to terminate wins. If the *never claim* wins the formula is not satisfied.
- The execution of the verifier can be seen as a game with two players: the *never claim* and the model.
- the model wins if it is never true the negation of the claim of the LTL formula. This means exactly that there are no trace of the model that satisfy the negation of the LTL formula.
- In this case, the verifier terminates when all state-space is searched.
- the *never claim* wins if it can be found an execution where the negation of the formula is true.
- in this case the never claim terminates

A *never claim* for a Safety property

```
$spin -f "![] p"
never { /* ! [] p */
T0_init :    /* init */
    if
    :: (1) -> goto T0_init
    :: (!p) -> goto accept_all
    fi;
accept_all :    /* 1 */
    skip }
```



New syntax

For example, for `ltl P1 [] (critical < 2)` we have

```
never P1 {  /* !([] ((critic <2))) */
T0_init:
```

```
  do
  :: atomic {(!(((critic<2)))) ->
                    assert(!(!(((critic<2)))))}
  :: (1) -> goto T0_init
  od;
accept_all:
  skip }
```

Running the verifier *Pan*

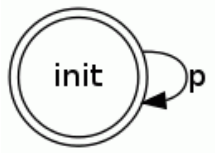
If you run

`$ spin -search -ltl P0 mesltl.pml`

(as with `-a` and compiled to `pan`)

- `$./pan` : runs the verifier
- `$./pan -d`: list of states and transitions
- `$./pan -D`: draws automata in dot format
- which can be visualized with dot:
- `dot -Tpng dot.out -o dot.png`
- for other options see `http://spinroot.com/spin/Man/Pan.html`.

```
$spin -f "[]p"
never  {    /* []p */
accept_init:
T0_init:
    do
    :: ((p)) -> goto T0_init
    od;
}
```

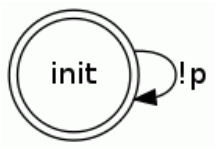


Labels starting with `accept` indicate accepting states.

A *never claim* for a liveness property

```
$ spin -f '!<>p'
never  {    /* !<>p */
accept_init:
```

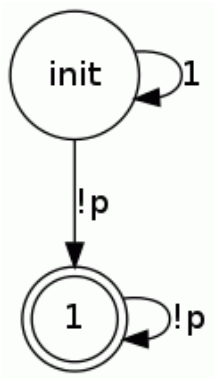
```
T0_init:
   do
   :: (! ((p))) -> goto T0_init
   od;
}
```



If p is true the *never claim* blocks (and "loses" as it is not possible to show that `[]!p`). If p is never true, the *never claim* passes infinitely often by the `init` state which is accepting (acceptance condition of Büchi automata). Thus `<>` `p` is not true and the *never claim* wins (the property does not hold).

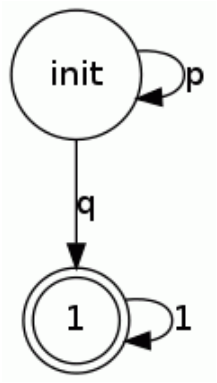
A *never claim* for GFp

```
$ spin -f '![]<>p'
never { /* !GF p */
T0_init :    /* init */
    if
    :: (1) -> goto T0_init
    :: (!p) -> goto accept_S2
    fi;
accept_S2 :    /* 1 */
    if
    :: (!p) -> goto accept_S2
    fi;
}
```



Until

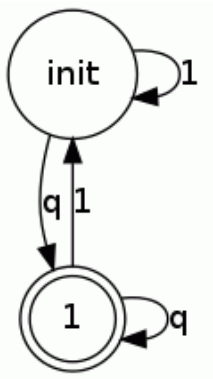
```
ltl P {pUq}
never { /* p U q */
T0_init :    /* init */
    if
    :: (p) -> goto T0_init
    :: (q) -> goto accept_all
    fi;
accept_all :    /* 1 */
   skip
}
```



Labels starting with `accept` indicate accepting states.

A more complicated example...with simplifications

```
$spin -f '[]<>(p U q)'
never  {    /* []<>(p U q) */
T0_init:
    do
    :: ((q)) -> goto accept_S9
    :: (1) -> goto T0_init
    od;
accept_S9:
    do
    :: (1) -> goto T0_init
    od;
}
```



Note that $\mathrm{GF}(p\mathrm{U}q) \equiv \mathrm{GF}q$

Converters from LTL to automata and *never claims*

- ltl2ba, `http://www.lsv.fr/~gastin/ltl2ba/`
- Spot, `https://spot.lrde.epita.fr/app/`
- Buchi store, `http://buchi.im.ntu.edu.tw/index.php/help/index/`

References

[BA08] Mordechai Ben-Ari. *Principles of the Spin Model Checker*. Springer, 2008.

[BKL08] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of Model Checking*. MIT Press, 2008.

[Var94] M. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff'94*, 1994.

[Var06] Moshe Vardi. Automata-theoretic techniques for temporal reasoning. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*. Elsevier, 2006.

[VW07] M. Vardi and T. Wilke. Automata: From logics to algorithms. In *WAL 07*, pages 645–753, 2007.