Test of Distributed Systems

Lecture 13: LTL Model checking

Århus School of Engineering

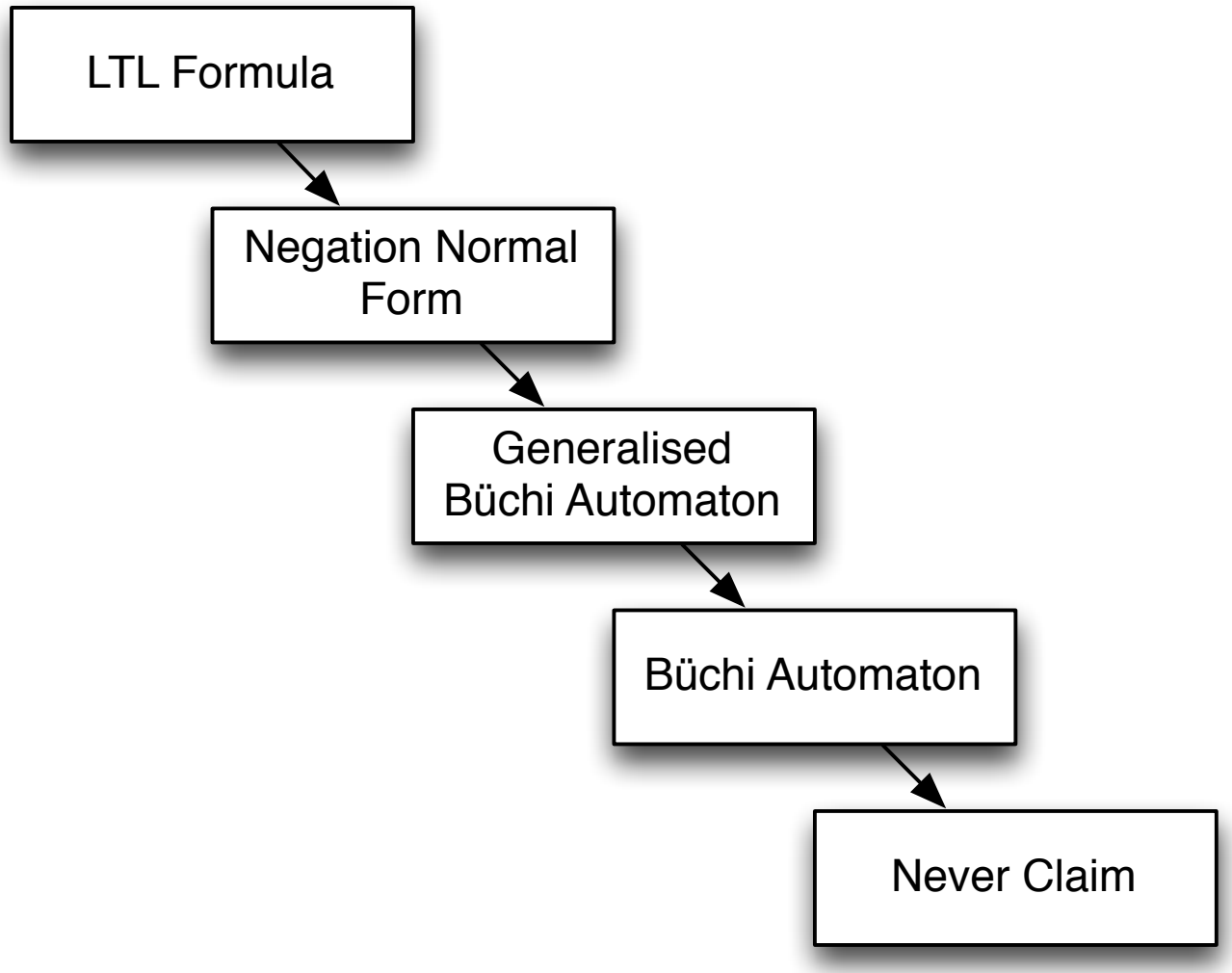
19 May 2014

Contents

What do we want to do?

- Verify whether some program M satisfies some LTL formula f
- We can execute program M, inspect all its states
- How can we relate formula f to M?
- Idea: turn f into a program N that observes the states of M with respect to f
- The meaning of M is appropriately described by an automaton
- Can we produce an automaton for N to verify f
- Yes: we can use a Büchi automaton B
- The Büchi automaton B can be easily represented by a program N
- That program provides the never claim

From LTL to Never Claims



Contents

LTL Syntax

Syntax:

$$\begin{array}{lcl} \mathbf{G}\, f & : & \text{“always } f\text{”} \\ \mathbf{F}\, f & : & \text{“eventually } f\text{”} \\ \mathbf{X}\, f & : & \text{“next } f\text{”} \\ f\ \mathbf{U}\ g & : & \text{“}f\text{ until } g\text{”} \\ f\ \mathbf{R}\ g & : & \text{“}f\text{ release } g\text{”} \end{array}$$

Duality:

$$\begin{array}{rcl} \mathbf{F}\, f & \leftrightarrow & \neg\, \mathbf{G}\, \neg f \\ \mathbf{X}\, f & \leftrightarrow & \neg\, \mathbf{X}\, \neg f \\ f\ \mathbf{R}\ g & \leftrightarrow & \neg(\neg f\ \mathbf{U}\ \neg g) \end{array}$$

Contents

Definition of Büchi Automaton (BA)

The target of the translation

A Büchi automaton *accepts* infinite computation sequences, that is, infinite words over some alphabet Σ.

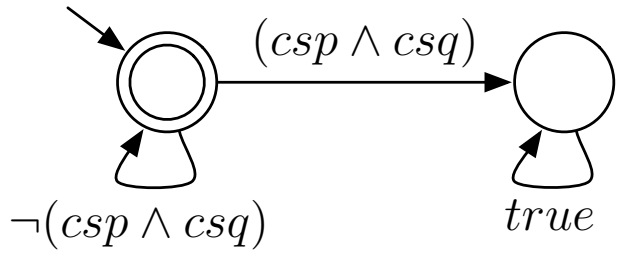
It is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ where:

- Q is a finite set of states
- Σ is a finite alphabet
- $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation
- $I \subset Q$ is a set of initial states
- $F \subset Q$ is a set of final states

A Büchi automaton accepts an infinite computation sequence if at least one infinitely often occurring state is in F.

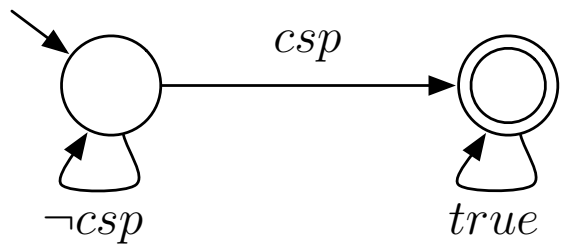
Examples

- Safety:



A Büchi automaton that *accepts* a safety property.

- Liveness:



A Büchi automaton that *accepts* a liveness property.

Definition of Generalised Büchi Automaton (GBA)

An intermediate representation

A Büchi automaton *accepts* infinite computation sequences, that is, infinite words over some alphabet Σ.

It is a tuple $\mathcal{B} = (Q, \Sigma, \Delta, I, \{G_1, \ldots, G_n\})$ where:

- Q is a finite set of states
- Σ is a finite alphabet
- $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation
- $I \subset Q$ is a set of initial states
- $G_i \subset Q$ for all i are sets of final states

A generalized Büchi automaton accepts an infinite computation sequence if the set of infinitely often occurring states contains at least one state from each G_i.

Tanslating a GBA into a BA

- Let $\mathcal{B} = (Q, \Sigma, \Delta, I, \{G_1, \ldots, G_n\})$ be a GBA
- Construct a BA

  $\mathcal{A} = (Q \times \{0, \ldots, n\}, \Sigma, \Delta', I \times \{0\}, Q \times \{n\})$,

  where

  $(\langle p, x\rangle, a, \langle q, y\rangle) \in \Delta'$ if
  - $(p, a, q) \in \Delta$
  - If $q \in G_i$ and $x = i - 1$ then $y = i$
  - If $x = n$ then $y = 0$
  - Otherwise $x = y$

So it is sufficient to construct a GBA.

Negation Normal Form (NNF)

LTL Laws

First we have to prepare the formula for the translation:

$$
\begin{aligned}
\neg\,\mathbf{G}\,f &\leftrightarrow \mathbf{F}\,\neg f \\
\neg\,\mathbf{F}\,f &\leftrightarrow \mathbf{G}\,\neg f \\
\neg(p\,\mathbf{U}\,g) &\leftrightarrow \neg f\,\mathbf{R}\,\neg g \\
\neg(p\,\mathbf{R}\,g) &\leftrightarrow \neg f\,\mathbf{U}\,\neg g \\
\neg\,\mathbf{X}\,f &\leftrightarrow \mathbf{X}\,\neg f \\
\mathbf{F}\,f &\leftrightarrow true\,\mathbf{U}\,f \\
\mathbf{G}\,f &\leftrightarrow false\,\mathbf{R}\,f \\
f \rightarrow g &\leftrightarrow \neg f \vee g \\
f \leftrightarrow g &\leftrightarrow (f \rightarrow g) \wedge (g \rightarrow f)
\end{aligned}
$$

NNF: push negation to the inside
and remove all logical operators except for **X**, **U**, **R**, $\neg$, $\wedge$, $\vee$

NNF Example

Transform $(f \ \mathbf{U} \ g) \rightarrow \mathbf{F}\, h$ into negation normal form:

$$
\begin{aligned}
&(f \ \mathbf{U} \ g) \rightarrow \mathbf{F}\, h \\
&\quad \{\text{NNF}\} \\
&\neg(f \ \mathbf{U} \ g) \vee \mathbf{F}\, h \\
&\quad \{\text{NNF}\} \\
&(\neg f \ \mathbf{R} \ \neg g) \vee \mathbf{F}\, h \\
&\quad \{\text{NNF}\} \\
&(\neg f \ \mathbf{R} \ \neg g) \vee (\mathit{true} \ \mathbf{U} \ h)
\end{aligned}
$$

Sub-formula closure

The is the basis for expressing the states of the Büchi automaton

For a LTL formula f the sub-formula closure $\mathcal{C}(f)$ is the smallest set satisfying:

- $true \in \mathcal{C}(f)$
- $f \in \mathcal{C}(f)$
- if $g \in \mathcal{C}(f)$ then $neg(g) \in \mathcal{C}(f)$
  where neg is defined for g with $g \neq \neg h$, for some h, by:

$$neg(g) = \neg g, \quad neg(\neg g) = g,$$

  furthermore: $neg(true) = false$ and $neg(false) = true$.
- if $\mathbf{X}\, g \in \mathcal{C}(f)$ then $g \in \mathcal{C}(f)$
- if $g \odot h \in \mathcal{C}(f)$ then $g \in \mathcal{C}(f)$ and $h \in \mathcal{C}(f)$,
  where $\odot$ is one of: $\wedge$, $\vee$, **U**, **R**

Maximally consistent sets of formulas

- A set $S \in \mathcal{C}(f)$ is called maximally consistent if
  - $true \in S$
  - $g \in S \quad \leftrightarrow \quad neg(g) \notin S$
  - $g \wedge h \in S \quad \leftrightarrow \quad g \in S \quad \wedge \quad h \in S$
  - $g \vee h \in S \quad \leftrightarrow \quad g \in S \quad \vee \quad h \in S$

- The states of the GBA to be constructed are subsets of maximally consistent sets.

- The idea of the construction is that if a state S of the GBA is visited infinitely often, then the formulas in S are true.
- We denote the set of consistent sets of formulas by $\mathcal{M}(f)$.

The Transition Relation of the GBA

- If a state contains the next operator $\mathbf{X}\, f$, then in the successor state f must hold
- Dealing the operators **U** and **R** requires the use of the following two recurrence equations:

$$f\ \mathbf{U}\ g \quad \leftrightarrow \quad g \vee (f \wedge \mathbf{X}(f\ \mathbf{U}\ g)) \tag{1}$$

$$f\ \mathbf{R}\ g \quad \leftrightarrow \quad g \wedge (f \vee \mathbf{X}(f\ \mathbf{R}\ g)) \tag{2}$$

- We can turn the formula (2) into a disjunction:

$$f\ \mathbf{R}\ g \quad \leftrightarrow \quad (g \wedge f) \vee (g \wedge \mathbf{X}(f\ \mathbf{R}\ g)) \tag{3}$$

- These equations resemble loops!

Construction of the GBA

The GBA $\mathcal{B} = (\mathcal{M}(f) \cup \{\iota\}, \mathcal{P}(\mathcal{L}), \Delta, \{\iota\}, G)$ where

- $\mathcal{L}$ are the propositional literals
- $\Delta = \Delta' \cup \Delta''$
  - $(S, a, T) \in \Delta'$ if and only if $a = T \cap \mathcal{L}$ and
    - $\mathbf{X}\, g \in S \quad \leftrightarrow \quad g \in T$
    - $g \,\mathbf{U}\, h \in S \quad \leftrightarrow \quad h \in S \quad \vee \quad (g \in S \quad \wedge \quad g \,\mathbf{U}\, h \in T)$
    - $g \,\mathbf{R}\, h \in S \quad \leftrightarrow \quad g \wedge h \in S \quad \vee \quad (h \in S \quad \wedge \quad g \,\mathbf{R}\, h \in T)$
  - $(S, a, T) \in \Delta''$ if and only if $a = T \cap \mathcal{L}$ and $S = \iota$ and $f \in T$
- $g \,\mathbf{U}\, h \in \mathcal{C}(f)$ if and only if

  $\{S \in \mathcal{M}(f) \mid h \in S \ \vee \ \neg(g \,\mathbf{U}\, h) \in S\} \in G$

Contents

Example

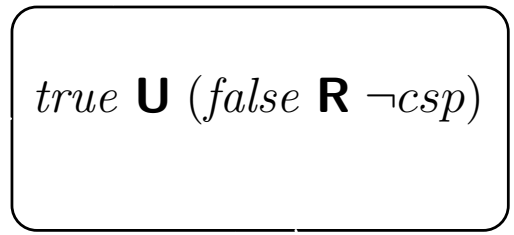
$$\neg \mathbf{G}\,\mathbf{F}\,csp$$

NNF of $\neg\,\mathbf{G}\,\mathbf{F}\,csp$

Transform $\neg\,\mathbf{G}\,\mathbf{F}\,cs$ into negation normal form:

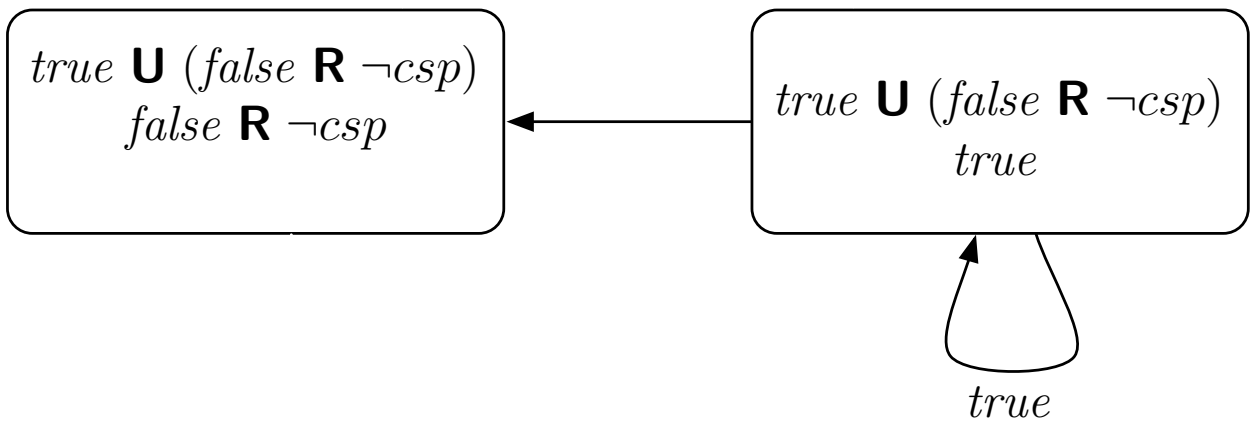
$$\neg\,\mathbf{G}\,\mathbf{F}\,cs$$
$$\{\text{NNF}^*\}$$
$$\mathbf{F}\,\mathbf{G}\,\neg csp$$
$$\{\text{NNF}^*\}$$
$$true\ \mathbf{U}\ (false\ \mathbf{R}\ \neg csp)$$

GBA for $\neg \mathbf{G}\,\mathbf{F}\,csp$



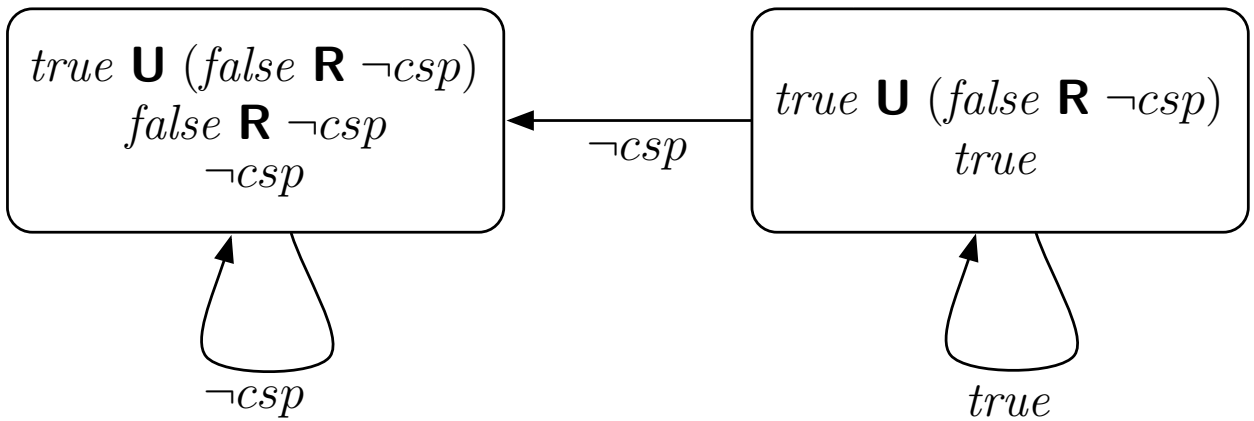
A Generalised Büchi automaton that *accepts* $\neg \mathbf{G}\,\mathbf{F}\,csp$.

GBA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$



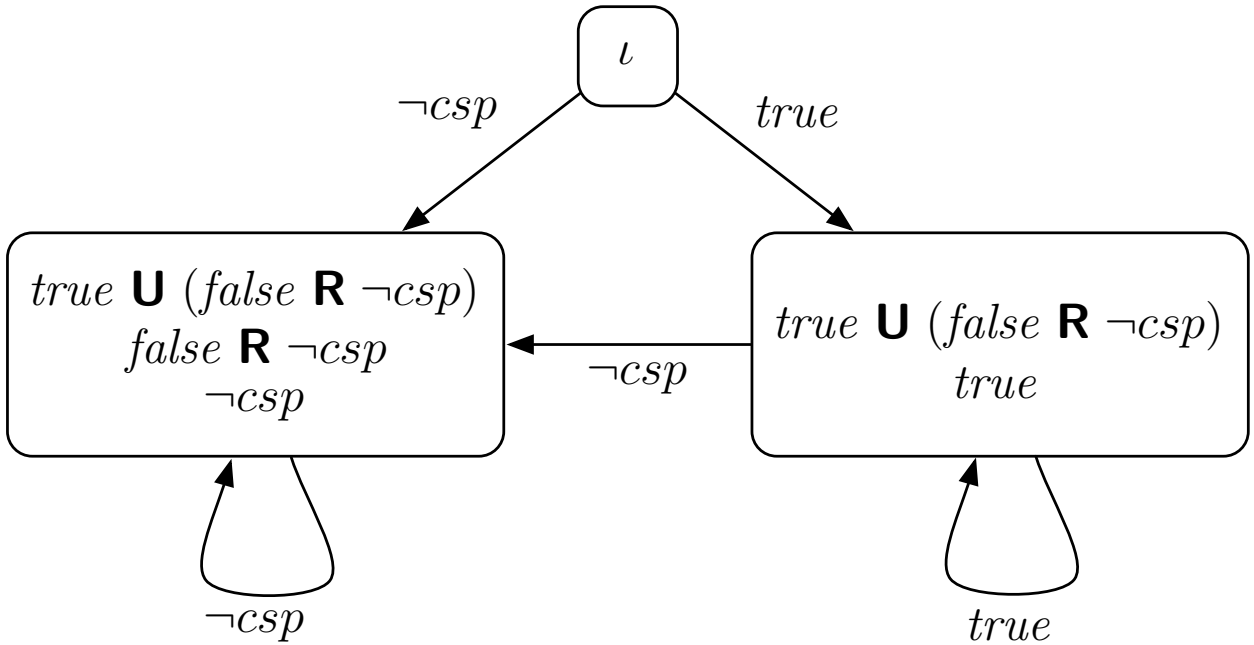
A Generalised Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.

GBA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$



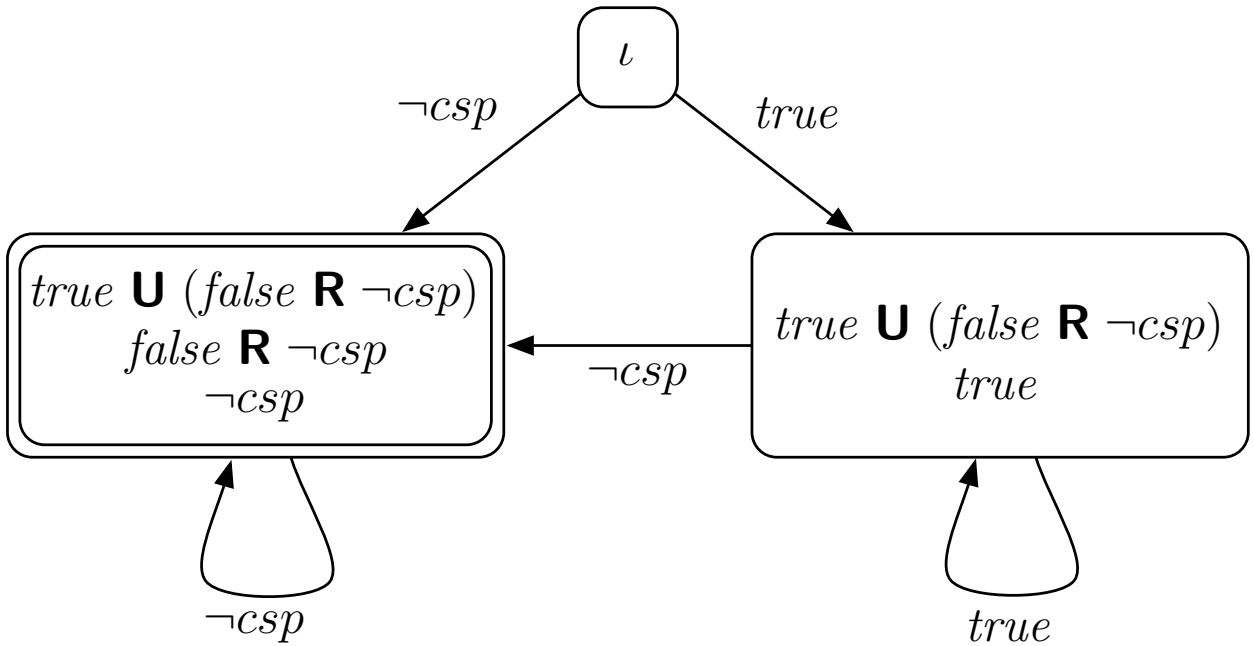
A Generalised Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.

GBA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$



A Generalised Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.

GBA/BA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$

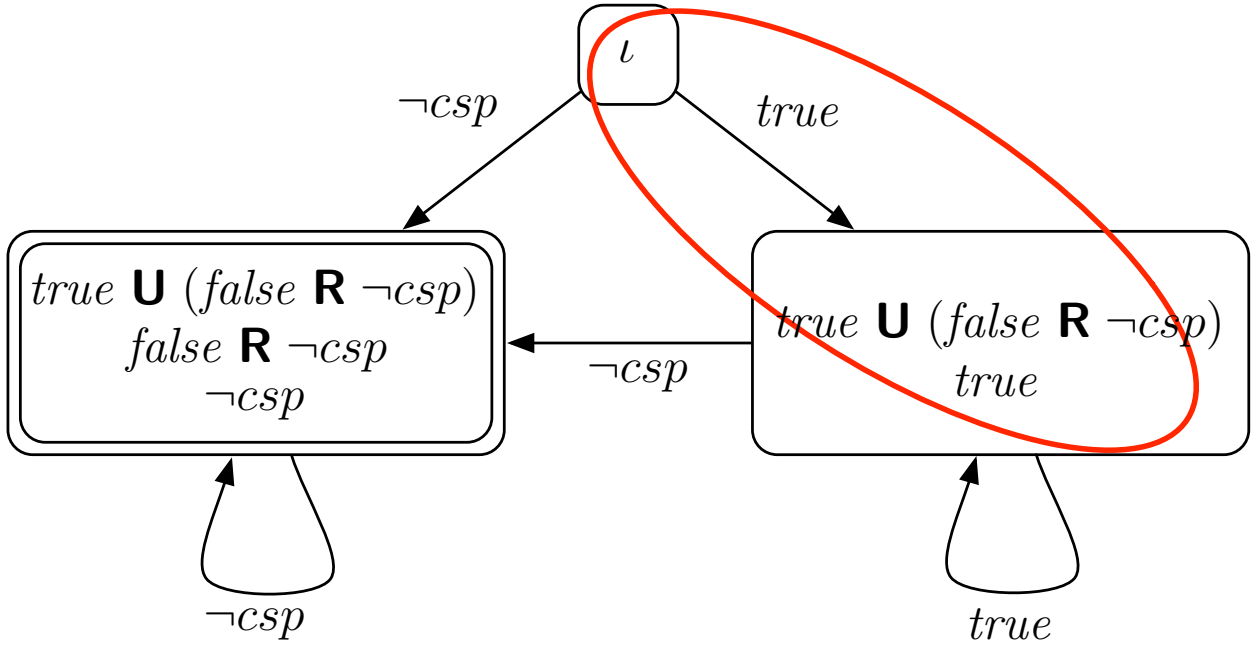


A Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.

Never Claim for $\neg \mathbf{G}\,\mathbf{F}\,csp$

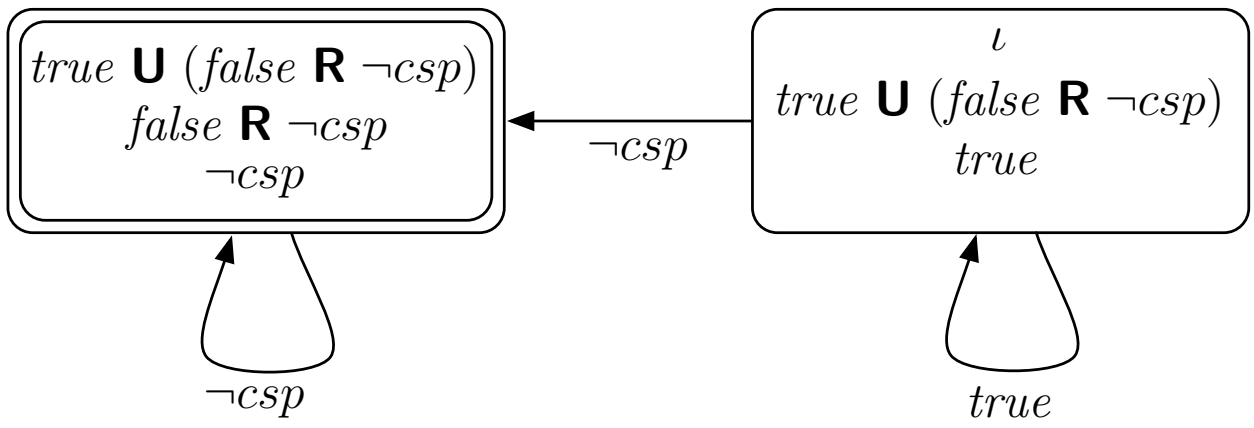
```
never { /* !([]<>csp) */
   T0_init:
   if
      :: (!((csp))) -> goto accept_S4
      :: (true) -> goto T0_init
   fi;
   accept_S4:
   if
      :: (!((csp))) -> goto accept_S4
   fi;
}
```

GBA/BA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$



A Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.

GBA/BA for $\neg\,\mathbf{G}\,\mathbf{F}\,csp$



A Büchi automaton that *accepts* $\neg\,\mathbf{G}\,\mathbf{F}\,csp$.