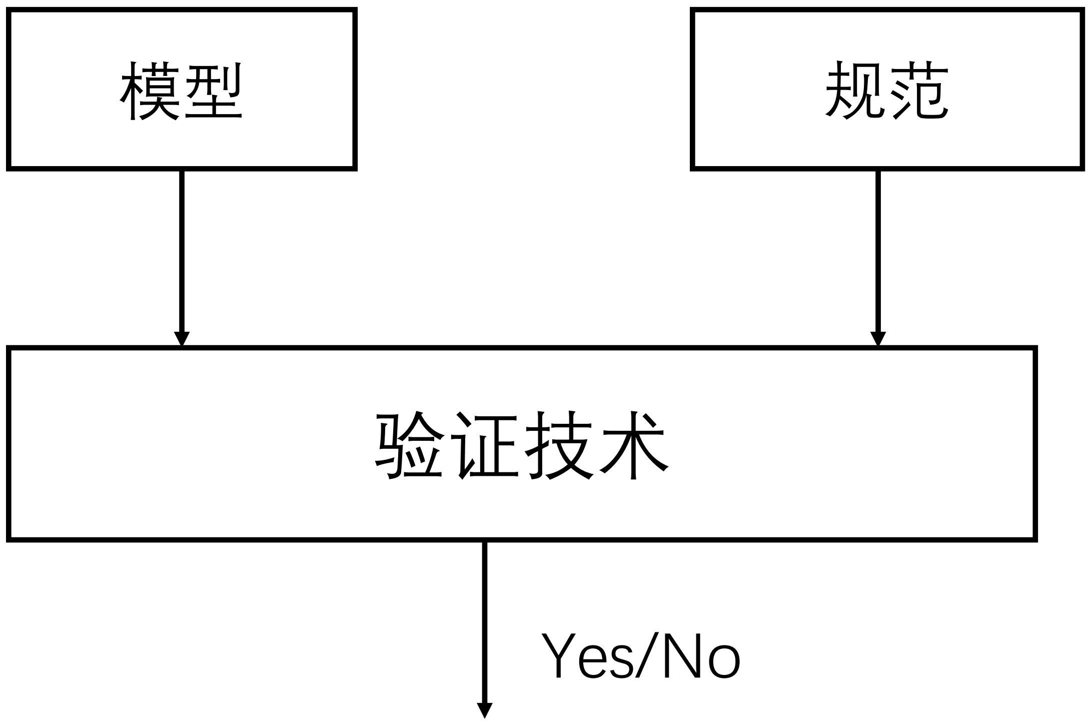
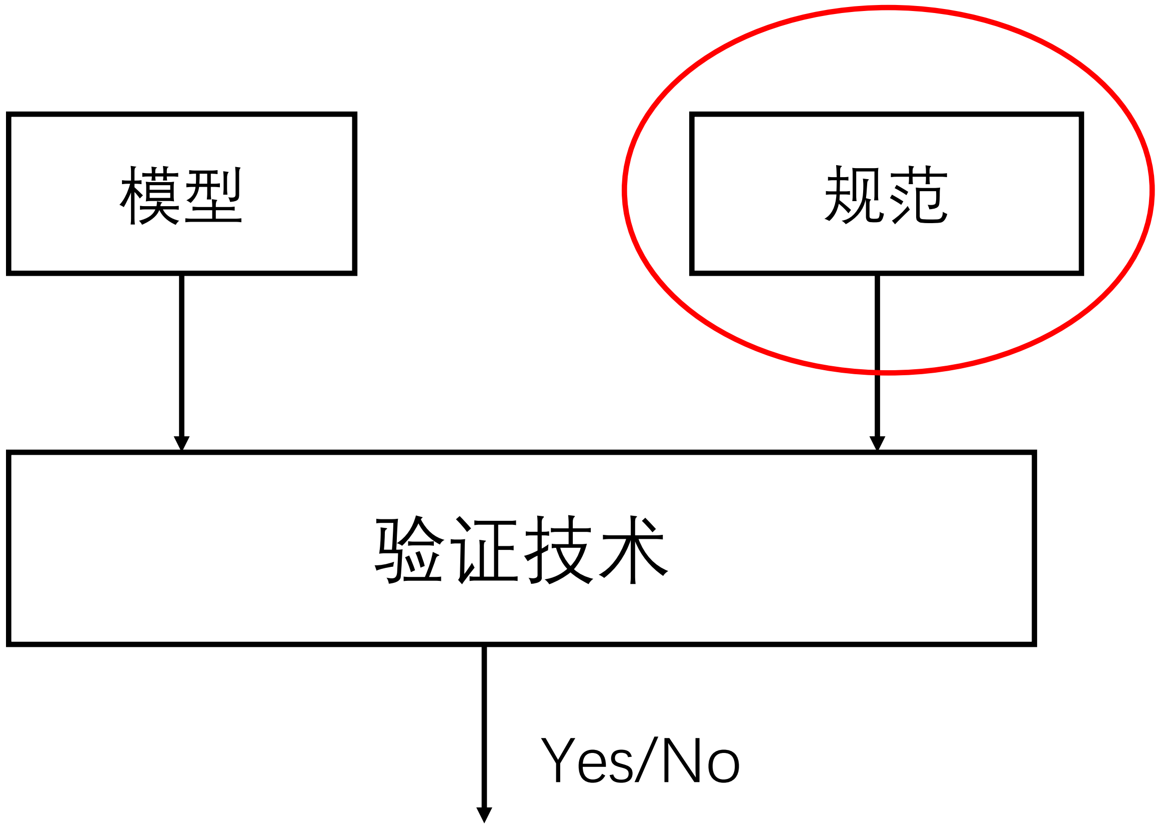
线性时态逻辑

李建文

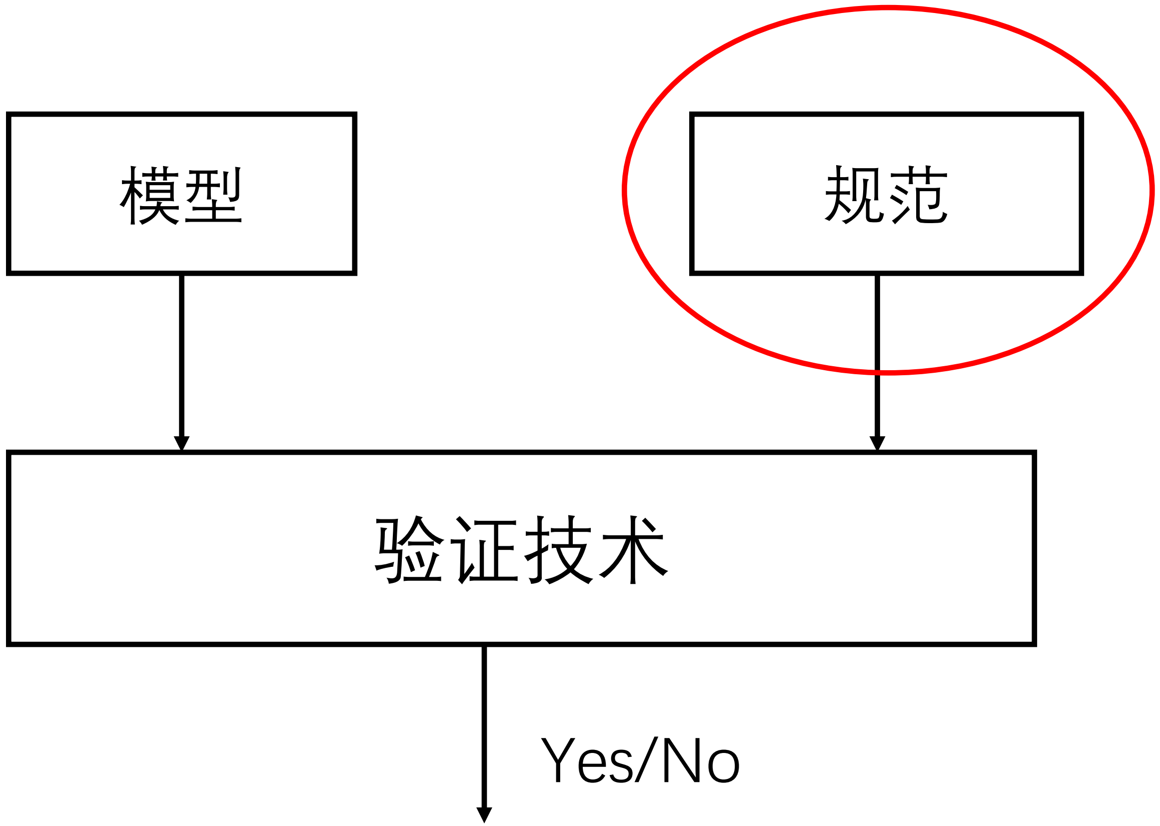
形式化验证



形式化验证



形式化验证



布尔逻辑
线性时态逻辑
分支时态逻辑
...

形式化验证

A design without specifications cannot be right or wrong, it can only be surprising!

-----Young, W., W. Boebert, and R. Kain, 1985: Proving a computer system secure. Scientific Honeyweller, 6(2), 1827.

为什么需要形式化的规范？

自然语言是不精确的

- I once shot an elephant in my pajamas. How he got in my pajamas I'll never know.
- Students hate annoying professors.
- The panda eats boots and leaves.

布尔逻辑

1. 一个布尔原子$p \in \{0,1\}$是一个布尔逻辑公式；

2. 如果φ是一个布尔逻辑公式，那么$(\varphi), \neg\varphi$，$\varphi\&\varphi$，$\varphi|\varphi$，$\varphi \Rightarrow \varphi$，$\varphi \Leftrightarrow \varphi$也是布尔逻辑公式。

线性时态逻辑 (LTL)

1. 一个布尔逻辑公式也是一个线性时态逻辑公式;

2. 如果φ是一个线性时态逻辑公式，那么
   $X\varphi$，$\varphi U\varphi$，$\varphi R\varphi$，$F\varphi$和$G\varphi$也是线性时态逻辑公式。

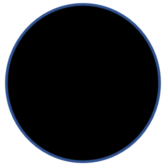
线性时态逻辑是布尔逻辑的一种扩展

线性时态逻辑 (LTL) -语法

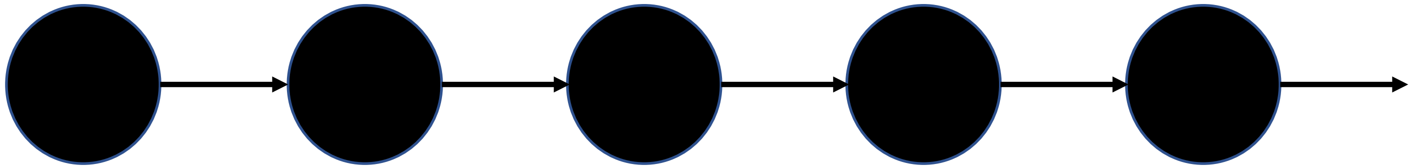
- $a \,\&\, b, a \,|\, b, !\, a$
- Xa & $X!\,b$ & cRd
- $X(aRd)$
- $a\ R\ (c\ U\ d)$
- FGa

- XRd
- $cFGa$

从命题逻辑到线性时态逻辑

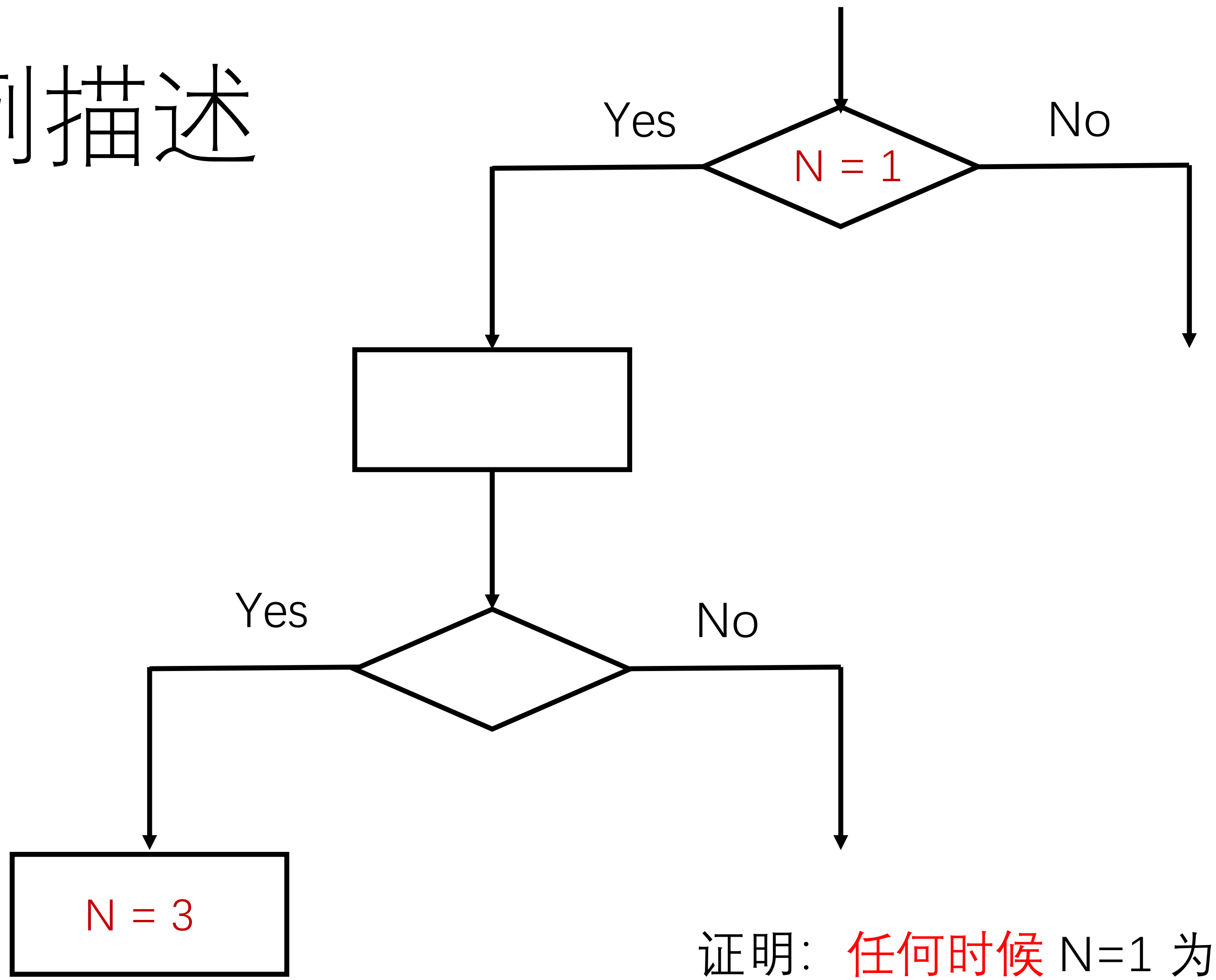


布尔逻辑：语义解释在一个时刻上



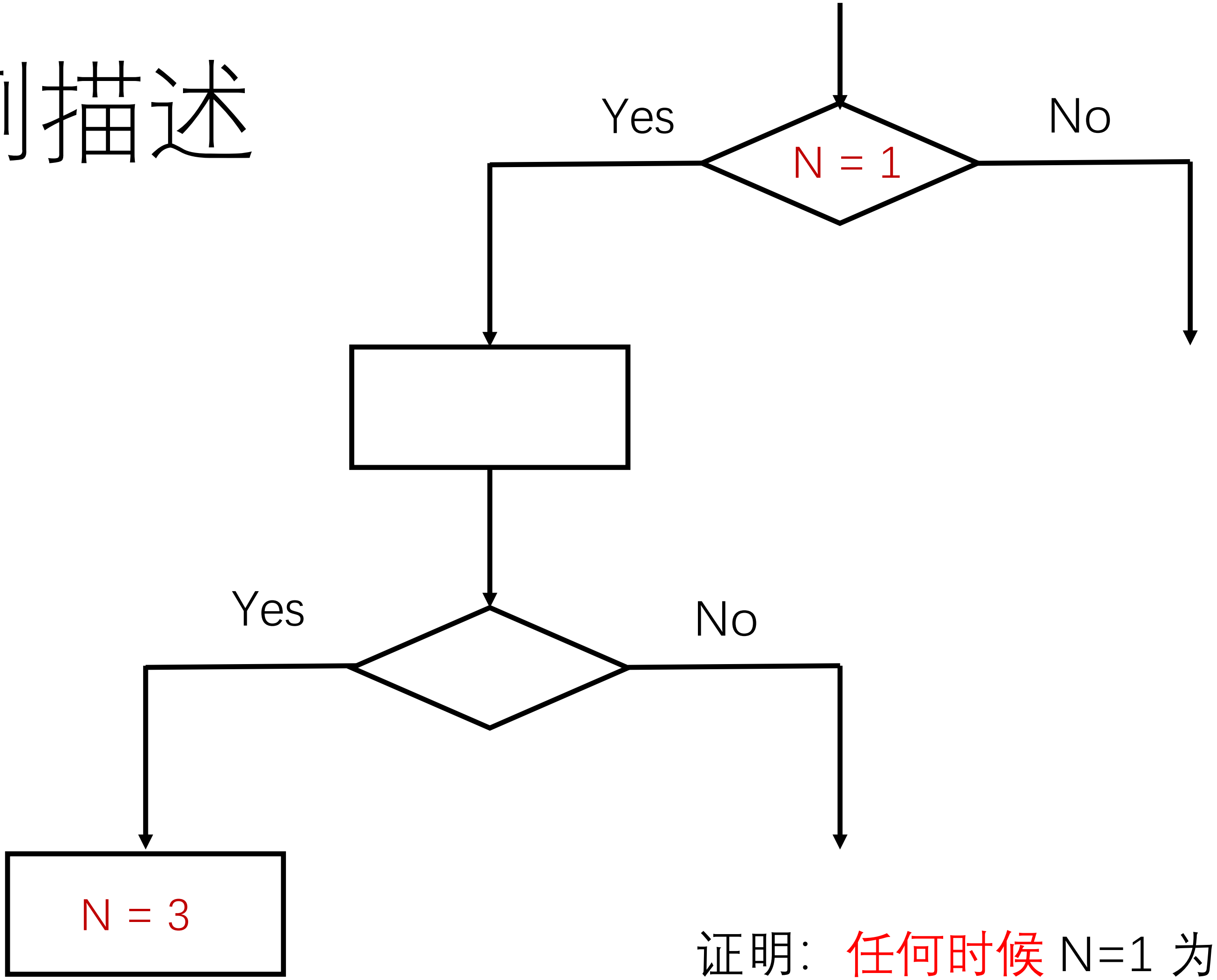
线性时态逻辑:
语义解释在一个无限长的时间序列上

案例描述



证明：任何时候 N=1 为真, N=3 在将来的某个时刻也会为真？

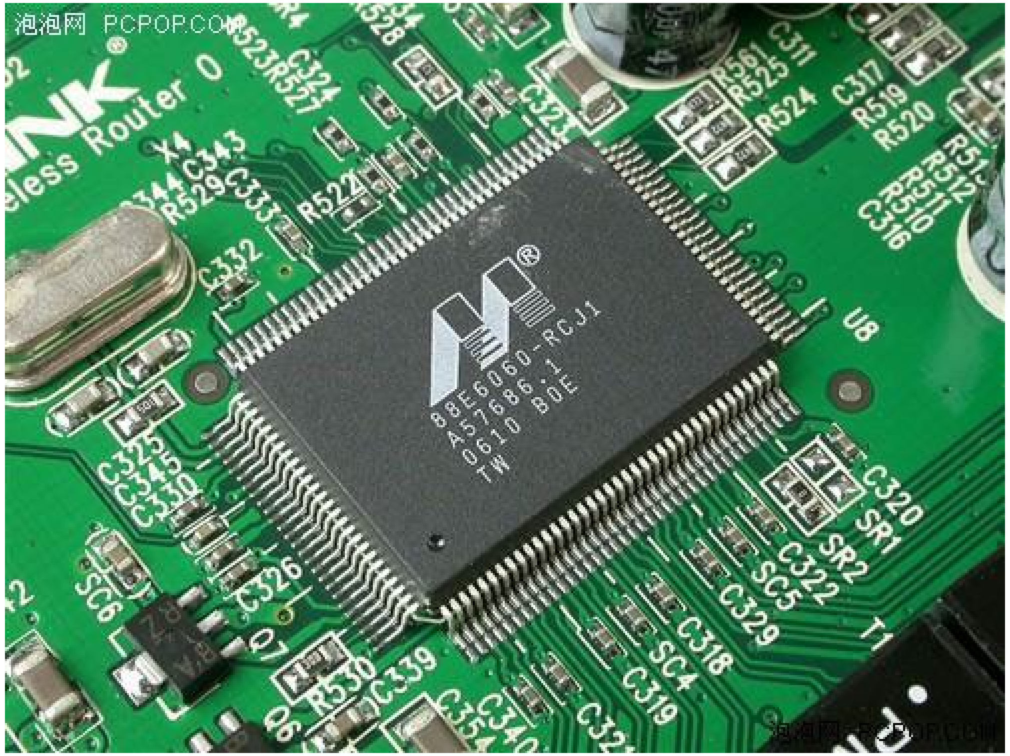
案例描述



证明：任何时候 N=1 为真, N=3 在将来的某个时刻也会为真?

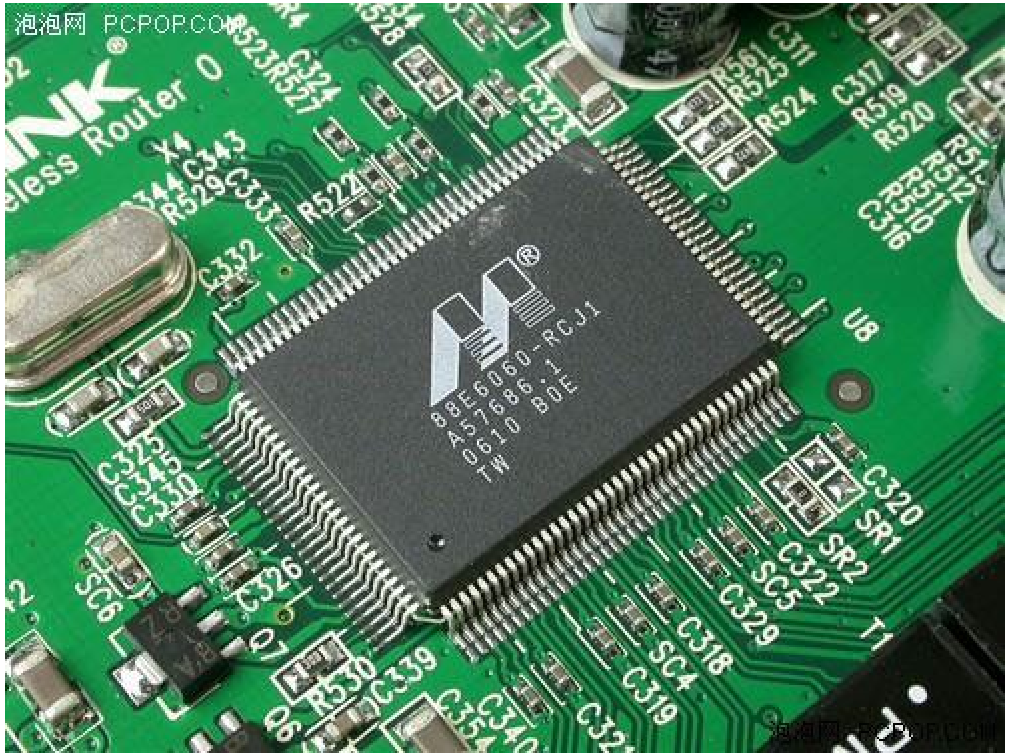
G (N=1 => F (N=3))

案例描述



证明：对任意的输入信号，输出信号都为1?

案例描述

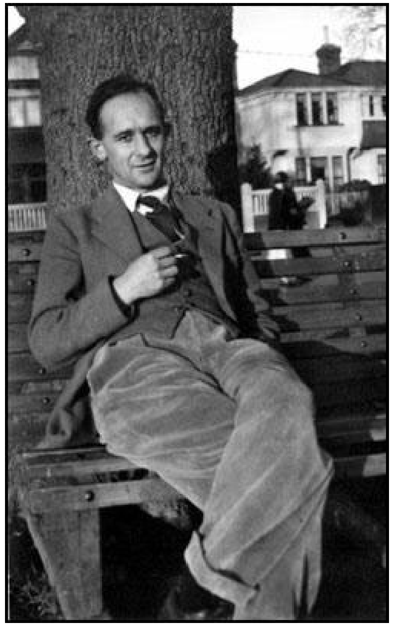


证明：对任意的输入信号，输出信号都为1?

G (output = 1)

线性时态逻辑LTL

From Church and Prior to PSL. Moshe Vardi at 25 years of Model Checking.



Arthur Prior
(1914–1969)

"I remember his waking me one night [in 1953], coming and sitting on my bed, . . ., and saying he thought one could make a formalised tense logic."

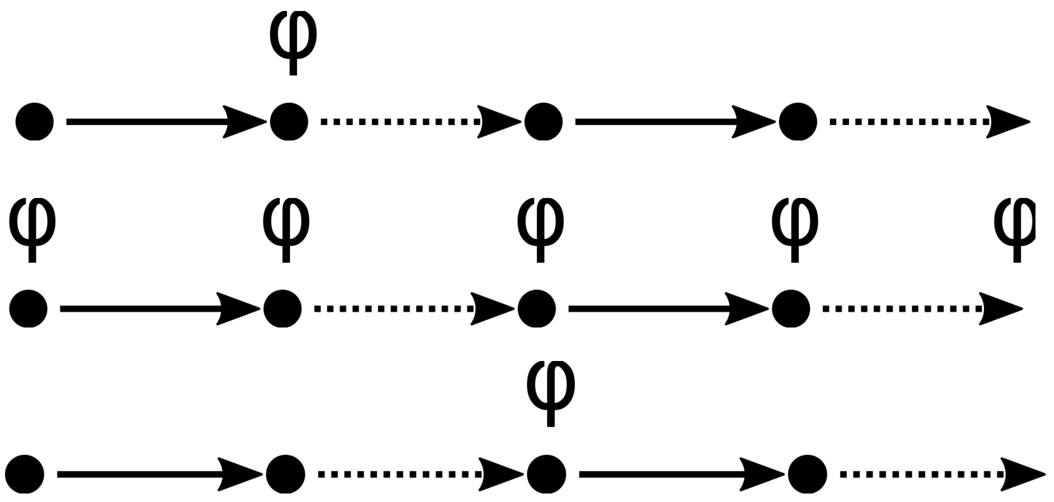
线性时态逻辑LTL

Amir Pnueli
(1941–2009)

First introduced into Computer Science in 1977. (1996 Turing Award)

线性时态逻辑LTL – 非形式化语义

$$\varphi \coloneqq p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi \cup \varphi \mid \varphi\ R\ \varphi \mid G\varphi \mid F\varphi$$



$X\varphi$ — φ在下一个时刻成立

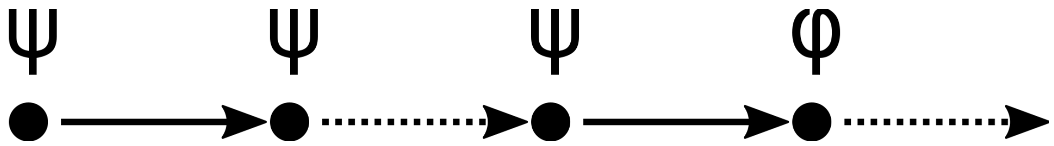
$G\varphi$ — φ在每一个时刻都成立

$F\varphi$ — φ在将来某个时刻成立

线性时态逻辑LTL – 非形式化语义

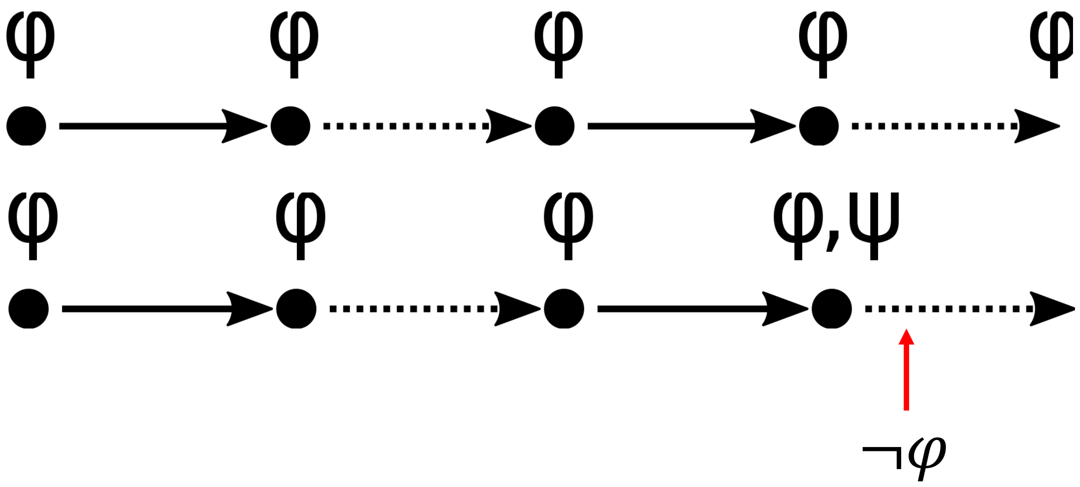
$$\varphi \coloneqq p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi \cup \varphi \mid \varphi\, R\, \varphi \mid G\varphi \mid F\varphi$$

$\psi \cup \varphi$



φ在将某个时刻成立，
并且ψ在那之前都成立

$\psi\,\mathrm{R}\,\varphi$



φ在每个时刻都成立，
否则ψ在那之前某个时刻成立

线性时态逻辑LTL-案例分析

Safety: Something bad (p) never happens.

线性时态逻辑LTL-案例分析

Safety: Something bad (p) never happens.

G !p

线性时态逻辑LTL-案例分析

Liveness: It is always the case that Something Good (p) eventually happens".

线性时态逻辑LTL-案例分析

Liveness: It is always the case that Something Good (p) eventually happens".

G F p

线性时态逻辑LTL-案例分析

Liveness/Fairness: p happens infinitely often"

线性时态逻辑LTL-案例分析

Liveness/Fairness: p happens infinitely often"

G F p

线性时态逻辑LTL-案例分析

Invariance: At some point, p will hold forever"

线性时态逻辑LTL-案例分析

Invariance: At some point, p will hold forever"

F G p

线性时态逻辑LTL-案例分析

Liveness: Every request is followed by a grant"

线性时态逻辑LTL-案例分析

Liveness: Every request is followed by a grant"

G (req -> F grant)

线性时态逻辑LTL-案例分析

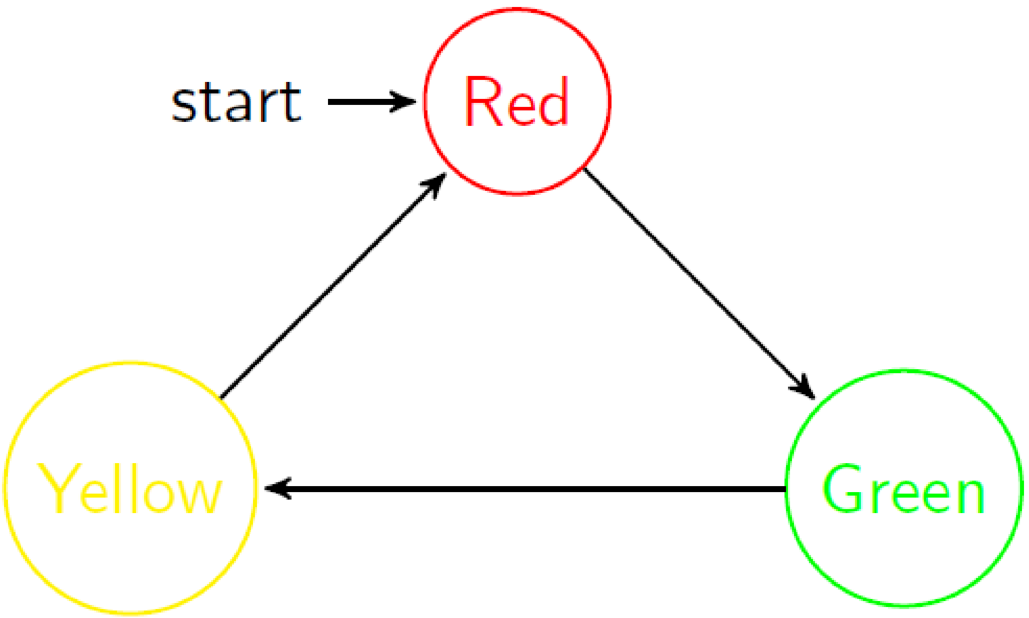
“p oscillates every time step"

线性时态逻辑LTL-案例分析

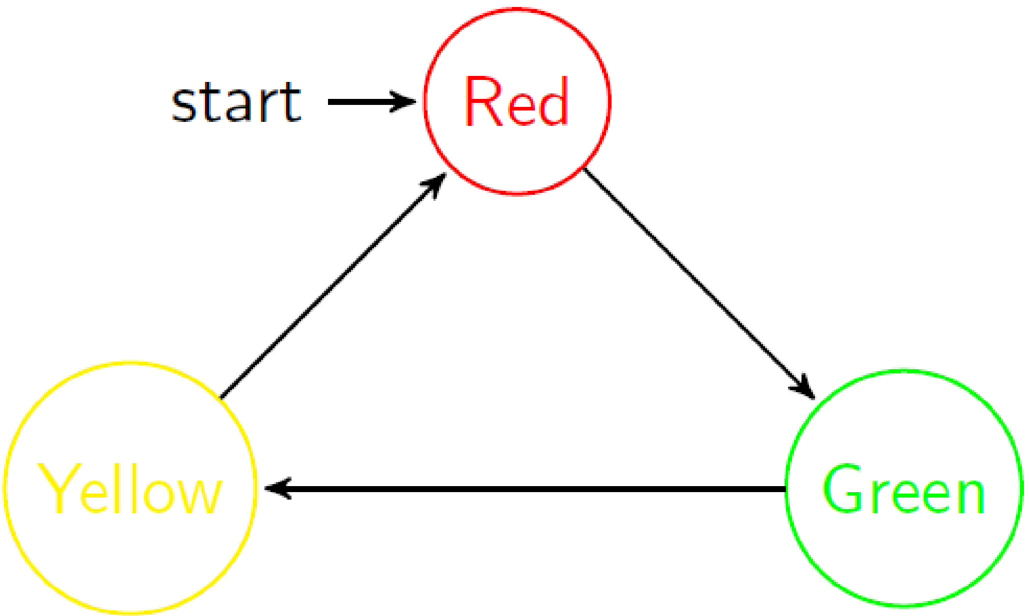
“p oscillates every time step"

G ((p -> X !p) & (!p -> X p))

线性时态逻辑LTL-课堂练习

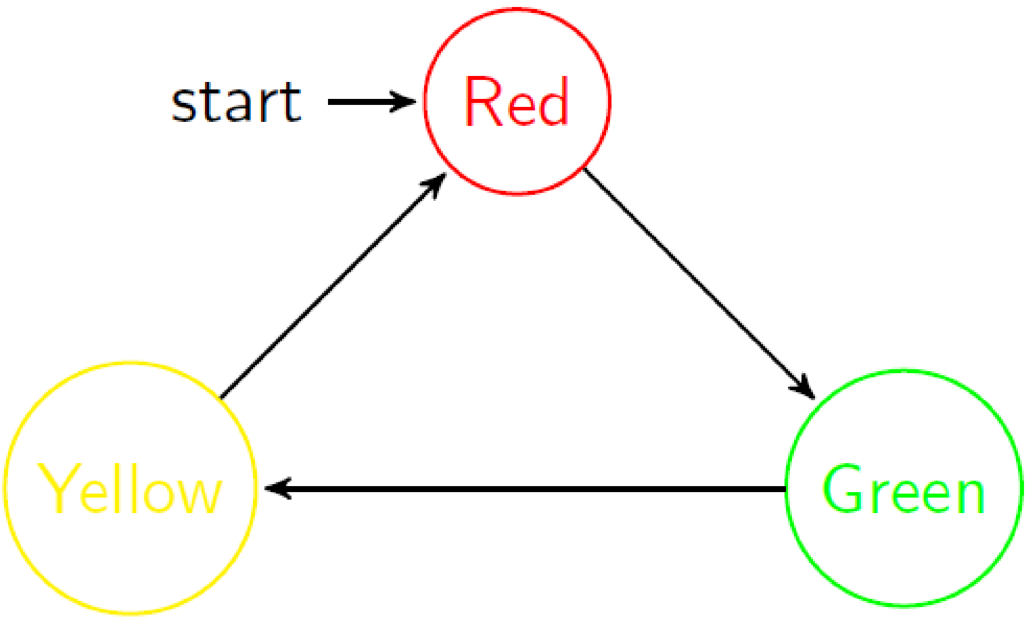


线性时态逻辑LTL-课堂练习



Traffic light is green infinitely often;

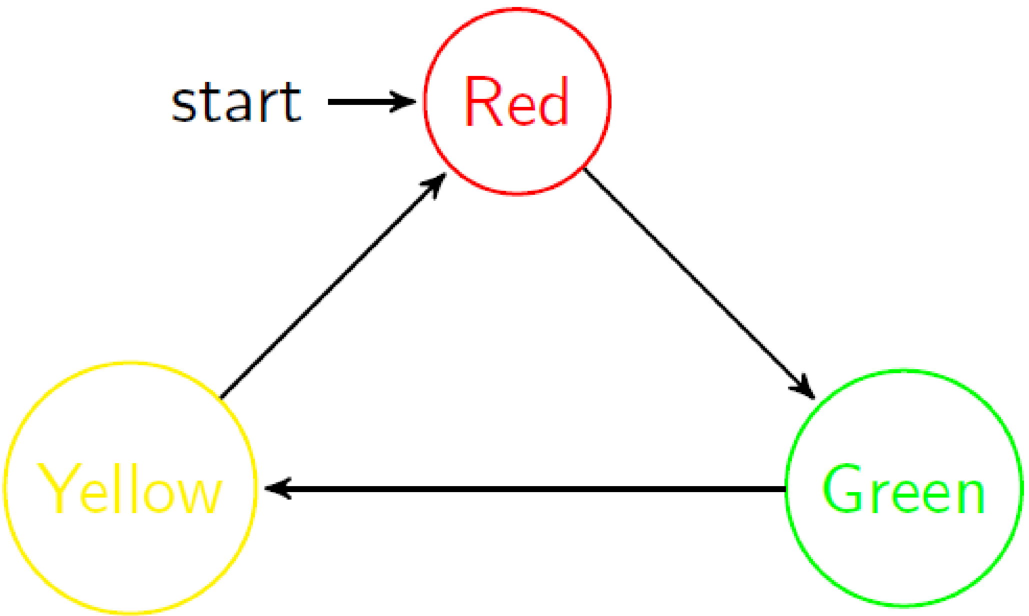
线性时态逻辑LTL-课堂练习



Traffic light is green infinitely often;

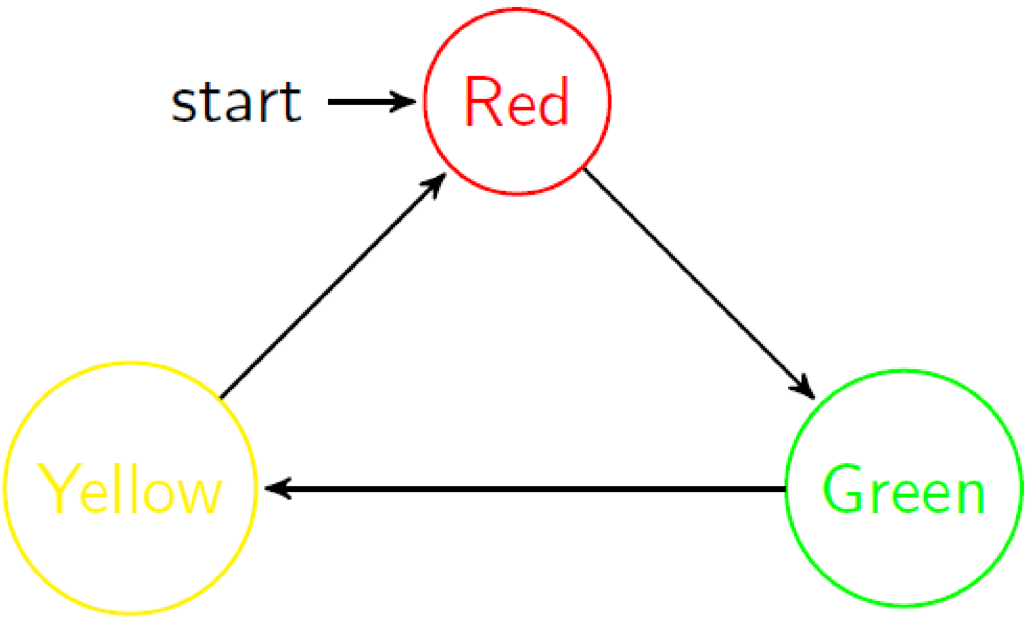
G F g

线性时态逻辑LTL-课堂练习



Traffic light cannot by green, red or yellow at the same time;

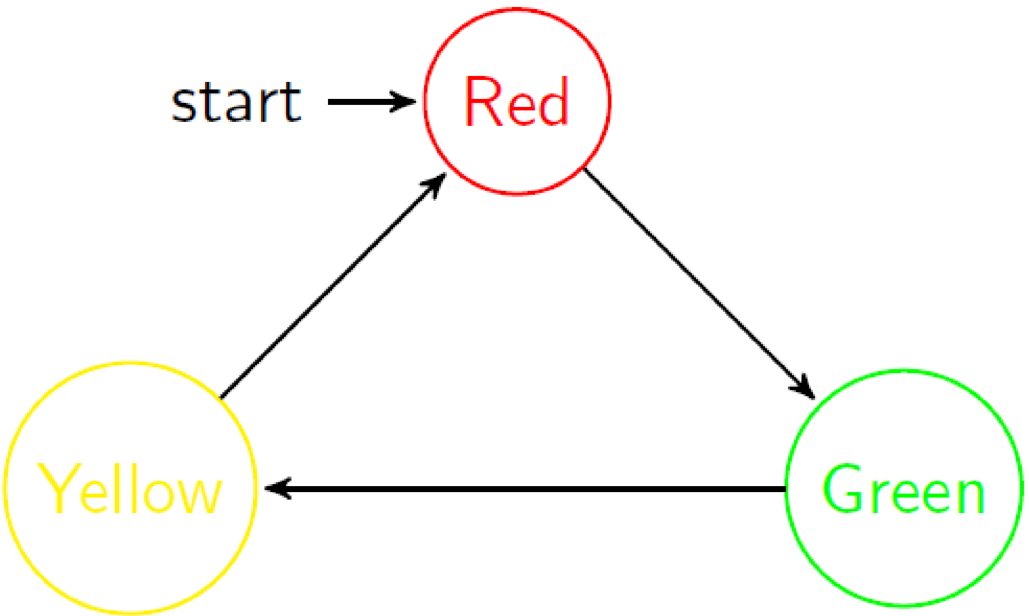
线性时态逻辑LTL-课堂练习



Traffic light cannot be green, red or yellow at the same time;

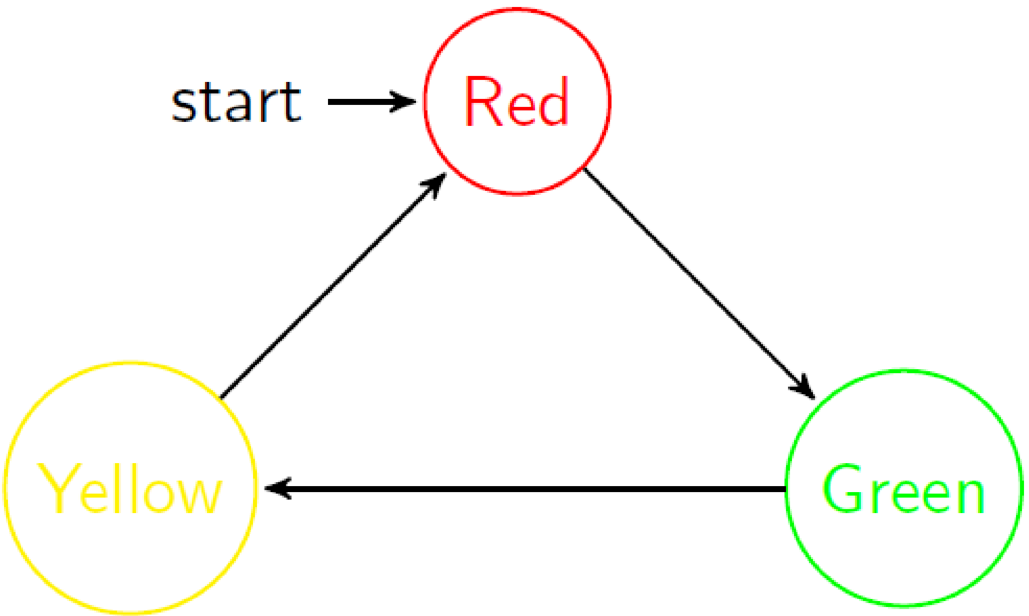
G ((g -> !r &!y) & (r -> !g&!y) & (y -> !g&!r))

线性时态逻辑LTL-课堂练习



There is no such case that the light color goes from green to red without being yellow at first;

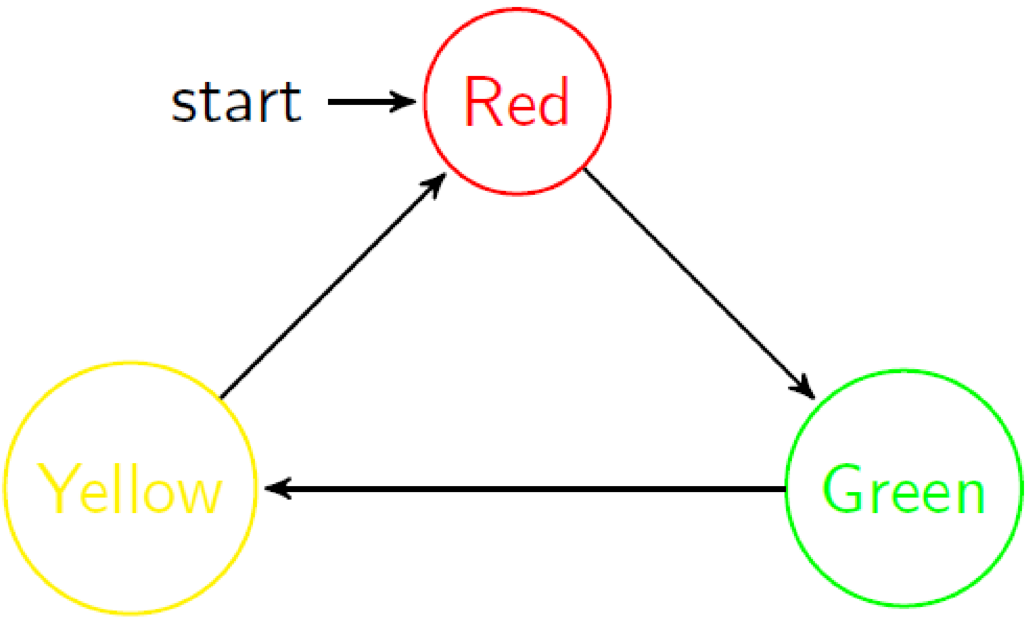
线性时态逻辑LTL-课堂练习



There is no such case that the light color goes from green to red without being yellow at first;

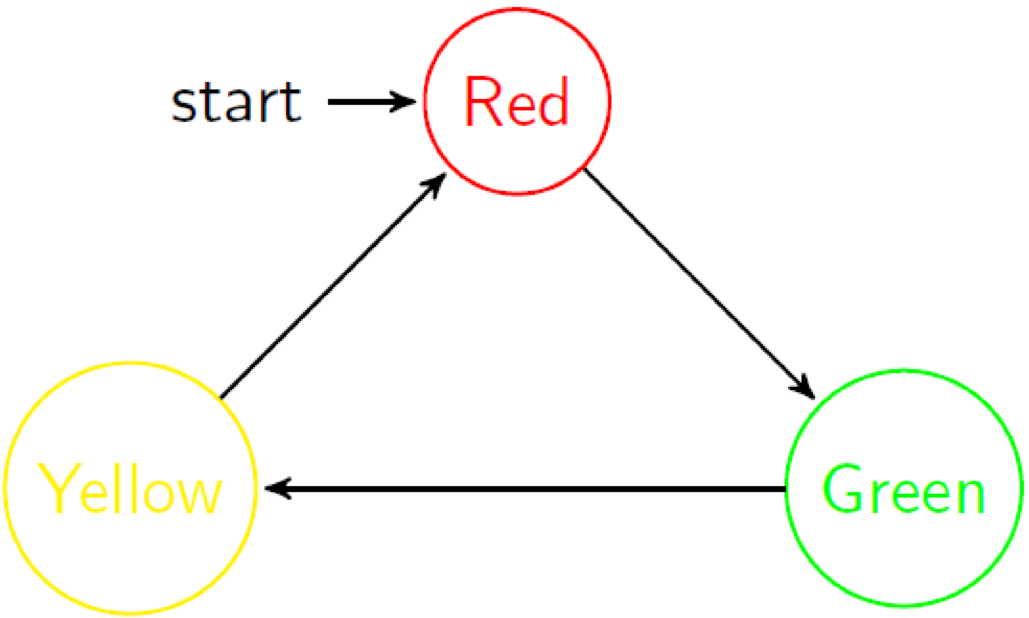
G !(g -> !y U r)

线性时态逻辑LTL-课堂练习



The light color turns into red, yellow and green in order;

线性时态逻辑LTL-课堂练习



The light color turns into red, yellow and green in order;

G !(g -> !y U r)
& !(y -> !r U g)
& !(r -> !g U y)

线性时态逻辑 – 形式化语义

给定一个无限序列$\sigma = \omega_0\omega_1\omega_2 \ldots$，令$\sigma(\mathrm{i}) = \omega_i$
为σ在时间i上的元素，$\sigma_i = \omega_i\ \omega_{i+1}\ \ldots$为$\sigma$从时间$i$开始的后缀。

- $\sigma \vDash true$且$\sigma \nvDash false$;
- $\sigma \vDash p$当且仅当$P \in \sigma(0)$;
- $\sigma \vDash \neg\varphi$当且仅当$\sigma \nvDash \varphi$;
- $\sigma \vDash \varphi_1 \wedge \varphi_2$当且仅当$\sigma \vDash \varphi_1$并且$\sigma \vDash \varphi_2$;
- $\sigma \vDash \varphi_1 \vee \varphi_2$当且仅当$\sigma \vDash \varphi_1$或者$\sigma \vDash \varphi_2$;

线性时态逻辑 – 形式化语义

给定一个无限序列$\sigma = \omega_0\omega_1\omega_2 \ldots$，令$\sigma(\mathrm{i}) = \omega_i$
为σ在时间i上的元素，$\sigma_i = \omega_i\ \omega_{i+1} \ldots$为$\sigma$从时间$i$开始的后缀。

- $\sigma \vDash X\varphi$当且仅当$\sigma_1 \vDash \varphi$;
- $\sigma \vDash \varphi_1 \mathrm{U} \varphi_2$当且仅当$\exists i \geq 0.(\sigma_i \vDash \varphi_2$成立，并且$\forall 0 \leq j < i. \sigma_j \vDash \varphi_1$成立);
- $\sigma \vDash \varphi_1 \mathrm{R} \varphi_2$当且仅当$\forall i \geq 0.$(如果$\sigma_i \vDash \varphi_2$不成立，则$\exists 0 \leq j < i. \sigma_j \vDash \varphi_1$成立);
- $\sigma \vDash F\varphi$当且仅当$\exists i \geq 0. \sigma_i \vDash \varphi$成立;
- $\sigma \vDash G\varphi$当且仅当$\forall i \geq 0. \sigma_i \vDash \varphi$成立;

形式化语义-课堂练习

证明$\varphi_1 \mathrm{U} \varphi_2 \equiv \neg(\neg\varphi_1 R \neg\varphi_2)$

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

1. Fa ?
2. F a & G ! a ?
3. a U b & !a R !b?

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

1. Fa ?
2. F a & G ! a ?
3. a U b & !a R !b?

1. Satisfiable;
2. Unsatisfiable;
3. Satisfiable

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

How?

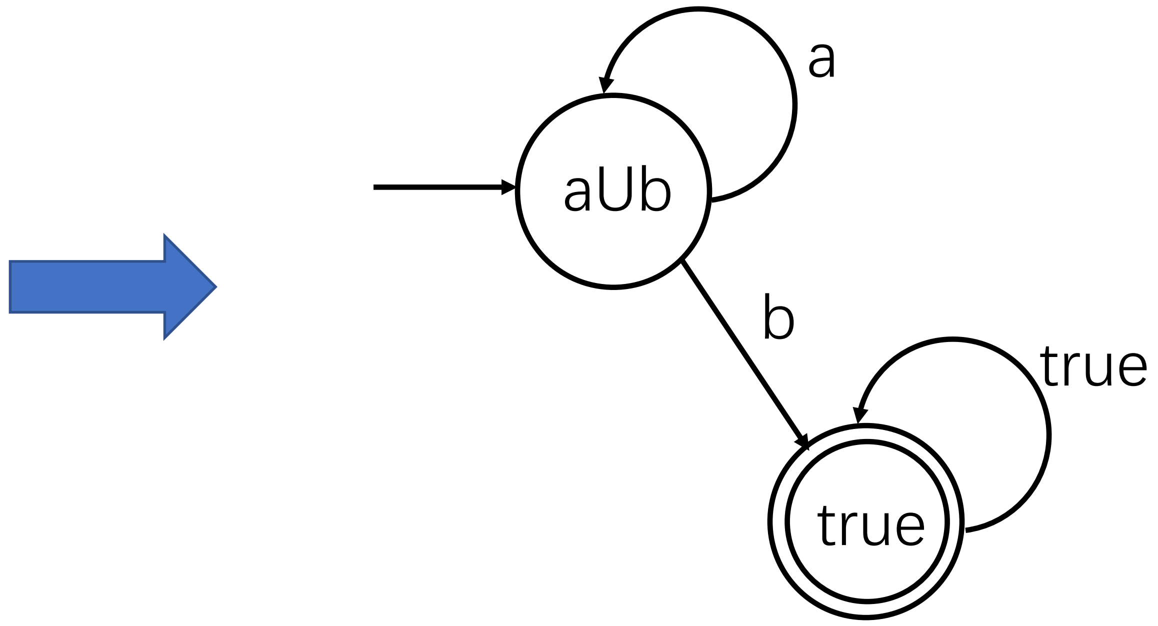
线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

How?

将LTL公式转成对应的B$\ddot{u}$chi自动机！

a U b



$B\ddot{u}$ ch i自动机

一个$B\ddot{u}$ ch i自动机可以表示成五元组$A = (\Sigma, S, T, I, F)$，其中

- Σ表示字母表的集合
- S表示状态的集合
- $T: S \times \Sigma \times S$表示边的集合
- $I \subseteq S$表示初始状态的集合
- $F \subseteq S$表示终止（接收）状态的集合

Büchi自动机的语义

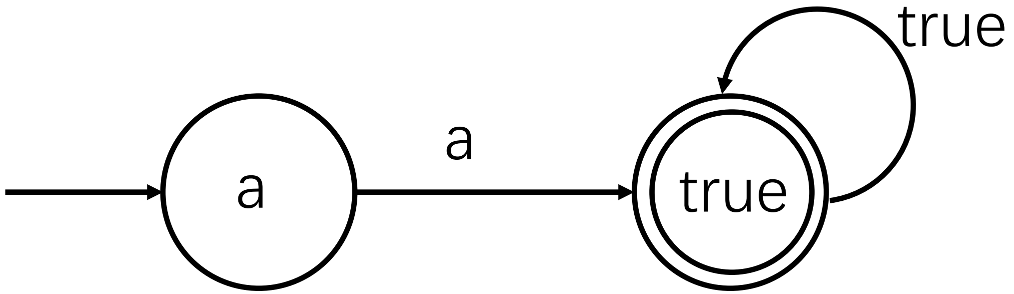
- 有限自动机$A=(\Sigma,S,T,I,F)$接收一组**无限长度**的字符串
- 给定一个字符串$\eta=a_0a_1\cdots$，η在A上的运行轨迹是一条无限长度的状态序列$s_0s_1\cdots$，使得s_0是一个初始状态并且(s_i,a_i,s_{i+1})是A上的一条边。
- 一个字符串$\eta=a_0a_1\cdots$可以被A接收当且仅当存在η在A上的一条运行轨迹，并且该轨迹**无限次的经过**F中的某个接收状态。
- $L(A)$用来表示A可以接收的所有字符串的集合。

LTL-to-BA

LTL公式为“a”

LTL-to-BA

LTL公式为“a”

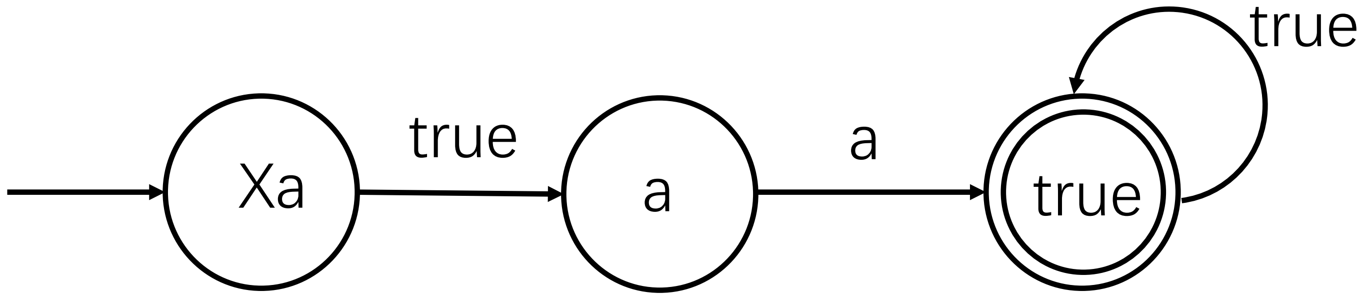


LTL-to-BA

LTL公式为“Xa”

LTL-to-BA

LTL公式为“Xa”

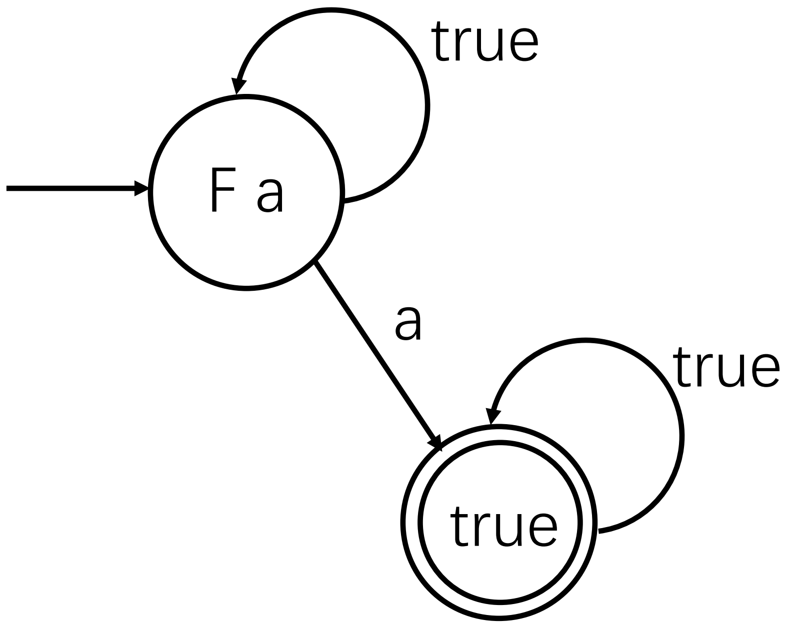


LTL-to-BA

LTL公式为“F a”

LTL-to-BA

LTL公式为“F a”

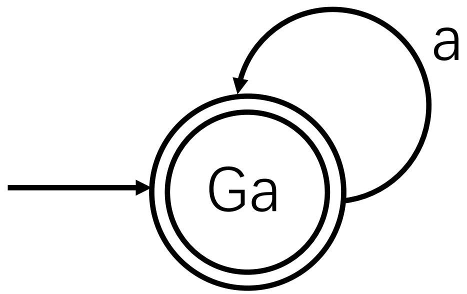


LTL-to-BA

LTL公式为“G a”

LTL-to-BA

LTL公式为“G a”



LTL-to-BA

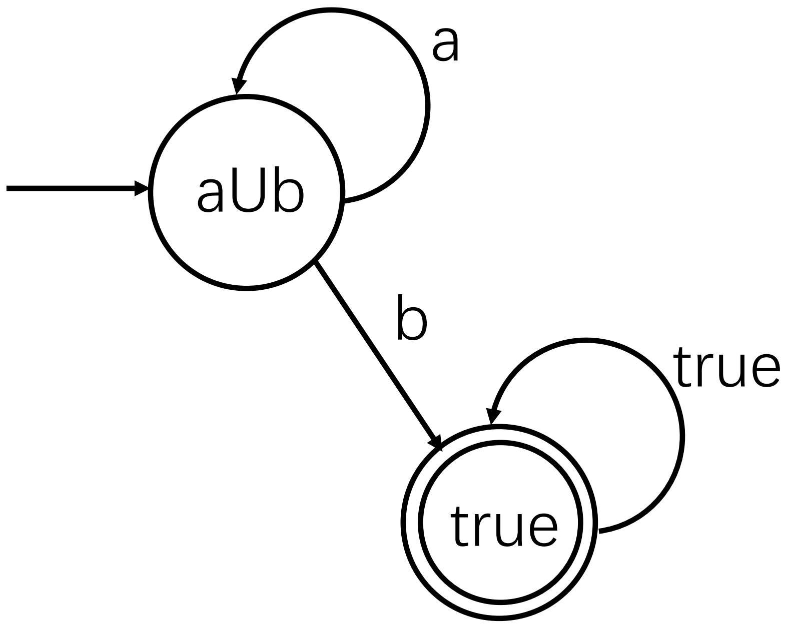
LTL公式为“aUb”

aUb = b | (a & X (aUb))

LTL-to-BA

LTL公式为“aUb”

aUb = b | (a & X (aUb))



LTL-to-BA

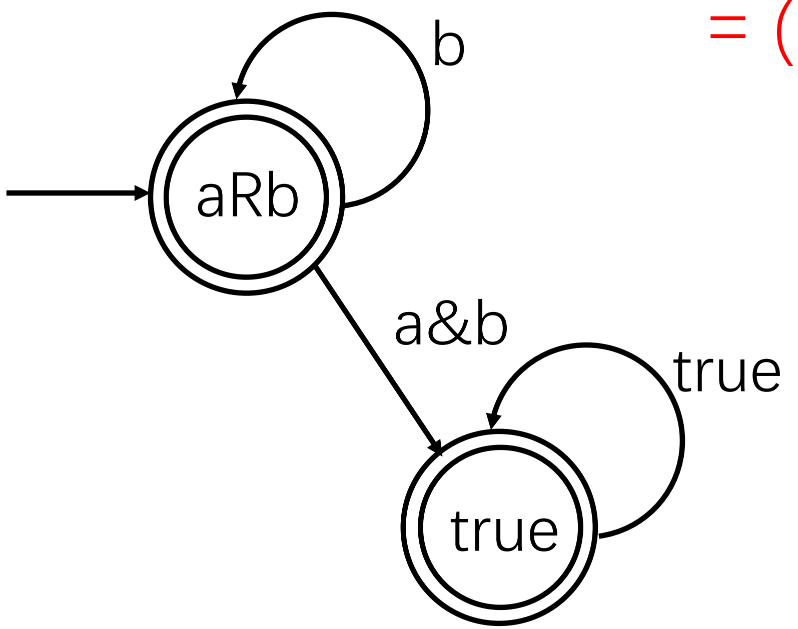
LTL公式为“aRb”

aRb = b & (a | X (aRb))
= (a&b) | (b & X (aRb))

LTL-to-BA

LTL公式为“aRb”

aRb = b & (a | X (aRb))
= (a&b) | (b & X (aRb))



停下来思考一下…

- 状态是什么?
- 边是什么?
- 初始状态是什么?
- 接收状态是什么?

停下来思考一下…

- 状态是什么?
- 边是什么?
- 初始状态是什么?
- 接收状态是什么?

一个LTL公式

一个布尔公式

最原始的LTL公式

LTL-to-BA

LTL公式为“aUb & cUd”

LTL-to-BA

LTL公式为“aUb & cUd”

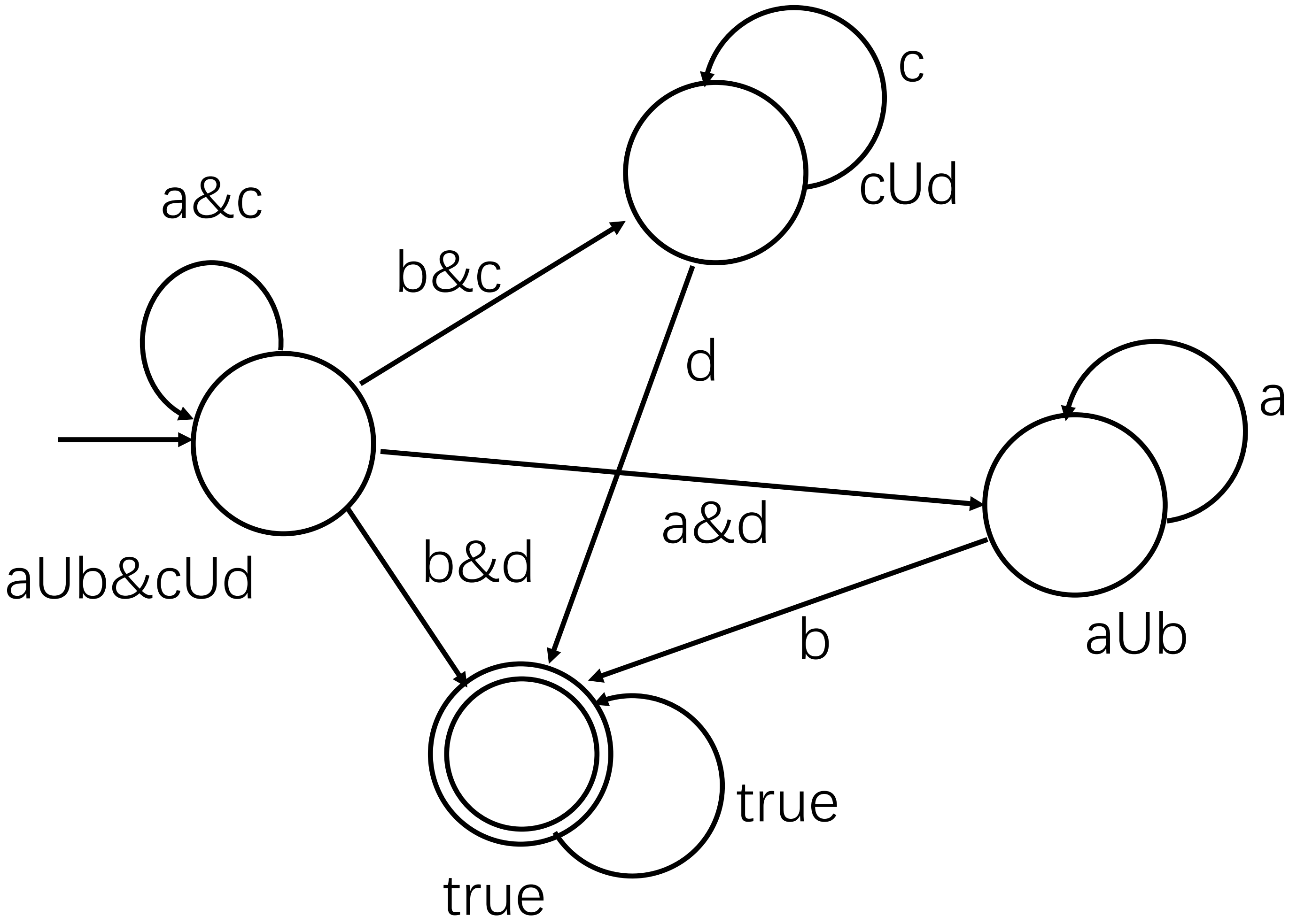
aUb & cUd = (b | a & X (a U b)) & (d | c & X(c U d))
= (b&d |
b&c&X(cUd) |
a&d&X (a U b) |
a&c&X (aUb & cUd))

aUb = b | a&X(aUb)

cUd = d | c&X(cUd)

true = true & Xtrue

BA for (aUb & cUd)



LTL 析取范式

$$\phi \equiv \bigvee (\alpha \wedge X\psi)$$

Every LTL formula has an equivalent normal form with an exponential translation cost.

LTL 析取范式

$$\varphi_1 U \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge X(\varphi_1 U \varphi_2))$$

$$\varphi_1 R \varphi_2 \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge X(\varphi_1 R \varphi_2))$$

LTL 析取范式

$\mathrm{NF}(\mathrm{p}) = \mathrm{p} \wedge \ Xtrue$

$\mathrm{NF}(\neg\mathrm{p}) = \neg\mathrm{p} \wedge \ Xtrue$

LTL 析取范式

$$NF\ (\varphi_1 \vee \varphi_2) = NF(\varphi_1) \cup NF(\varphi_2)$$

$$NF\ (\varphi_1 \wedge \varphi_2) = \{\alpha_1 \wedge \alpha_2 \wedge X(\psi_1 \wedge \psi_2) | \alpha_1 \wedge X\psi_1 \in NF(\varphi_1) \\ and\ \alpha_2 \wedge X\psi_2 \in NF(\varphi_2)\}$$

LTL 析取范式

$$NF\ (X\varphi) = true\ \ \wedge X\varphi$$

$$NF\ (\varphi_1 U \varphi_2) = NF(\varphi_2) \cup NF(\varphi_1 \wedge X(\varphi_1 U \varphi_2))$$

$$NF\ (\varphi_1 R \varphi_2) = NF(\varphi_1 \wedge \varphi_2) \cup NF(\varphi_2 \wedge X(\varphi_1 R \varphi_2))$$

思考题

如何写一个算法，求任意一个LTL公式的析取范式？

LTL-to-BA

LTL公式为“aUb & cRd”

LTL-to-BA

LTL公式为“aUb & cRd”

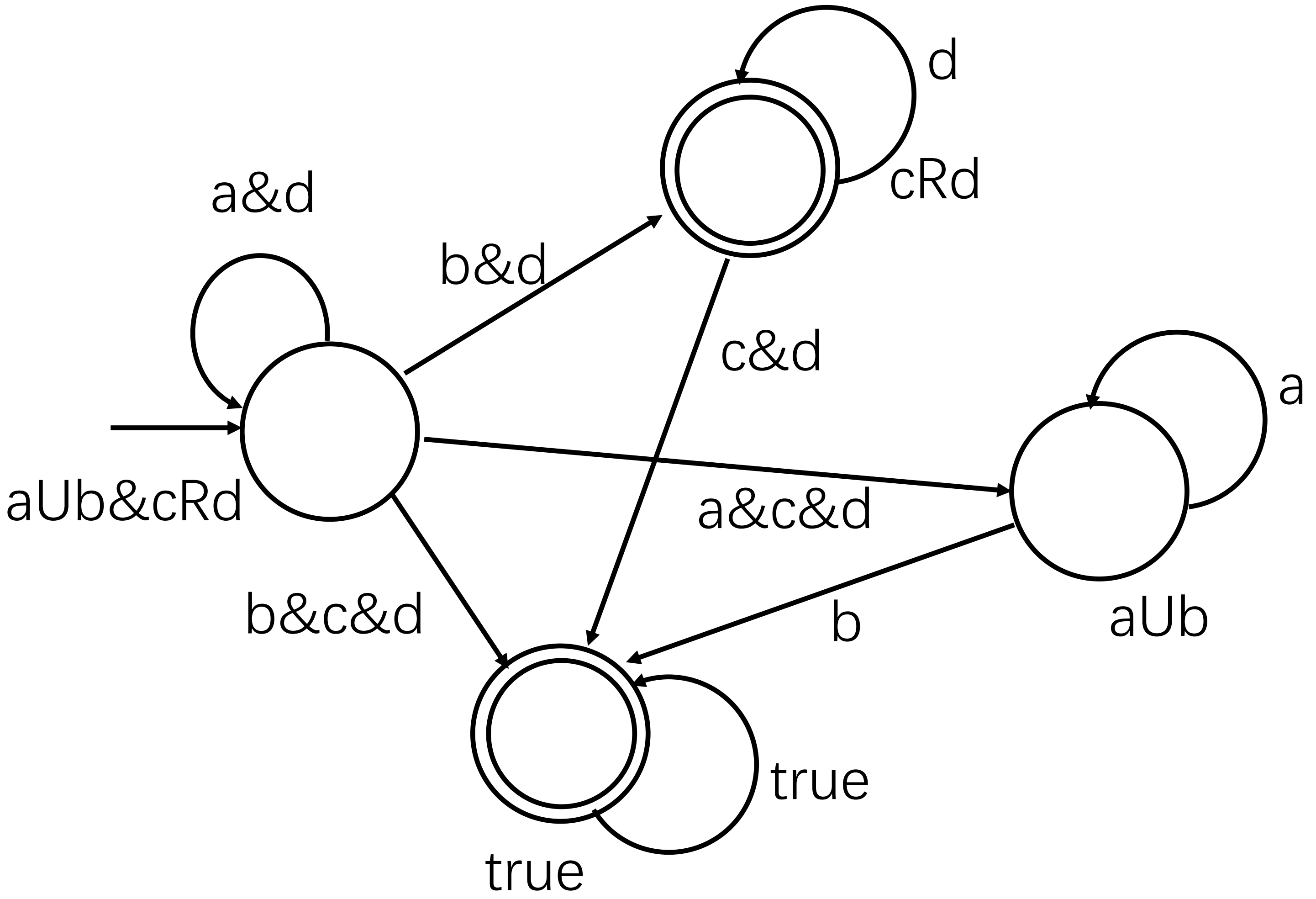
aUb & cRd = (b | a & X (a U b)) & (c&d | d & X(c R d))
= (b&c&d |
b&d&X(cRd) |
a&c&d&X (a U b) |
a&d&X (aUb & cRd))

aUb = b | a&X(aUb)

cRd = (c&d) | d&X(cRd)

true = true & Xtrue

BA for (aUb & cRd)



LTL-to-BA

LTL公式为“G (F a & F ! a)”

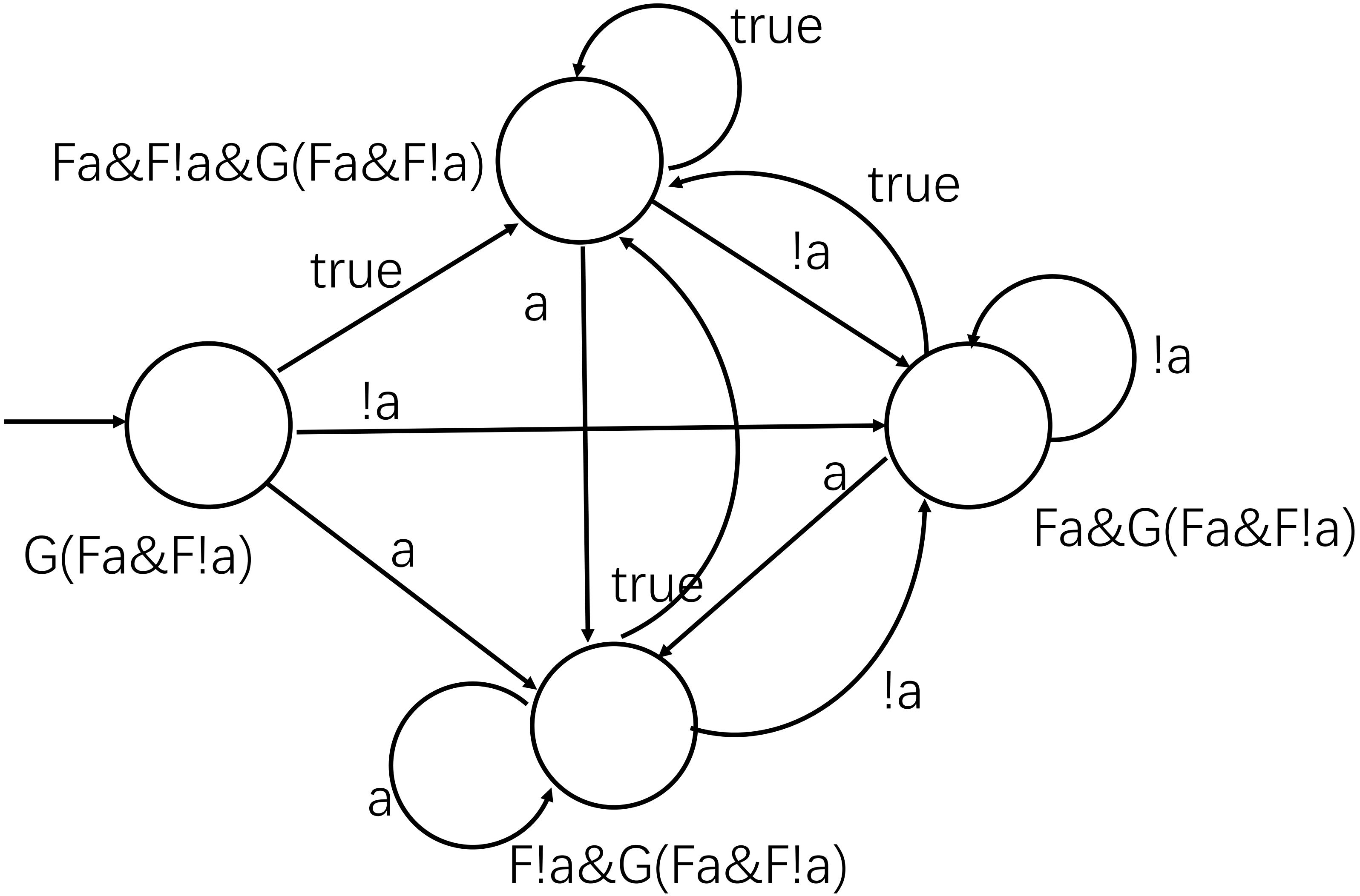
LTL-to-BA

LTL公式为“G (F a & F ! a)”

G(F a & F ! a) = Fa & F !a & X(G (F a & F !a))
= (a | XFa) & (!a | XF!a) & X (G (F a & F !a))
= (a & X(F!a & G (Fa & F!a))) |
(!a & X(Fa & G (Fa & F!a))) |
X(Fa & F!a & G (Fa & F!a))

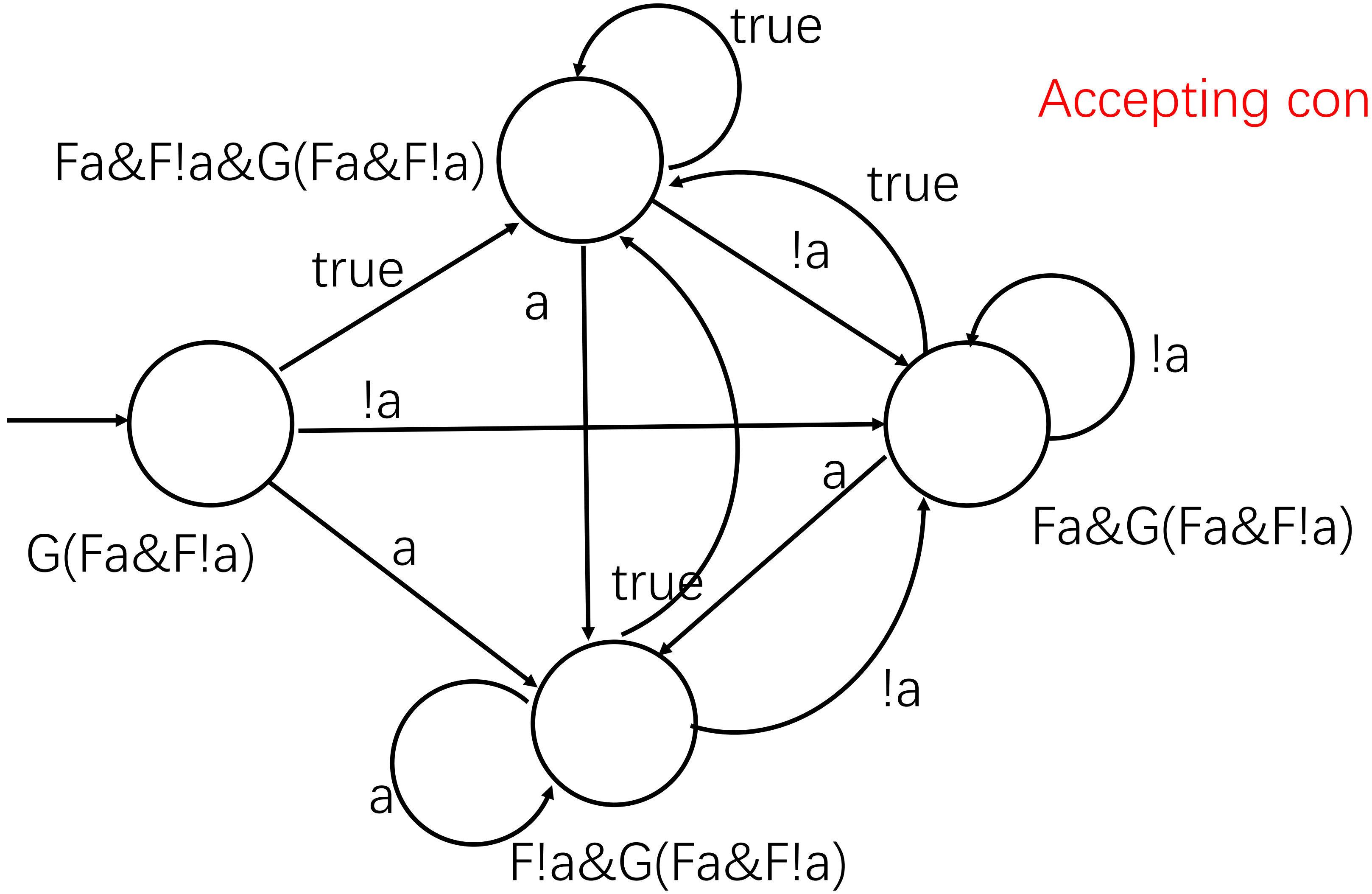
G(F a & F ! a) = F!a & G (Fa & F!a))) = Fa & G (Fa & F!a))) = Fa & F!a & G (Fa & F!a)))

BA for G(Fa & F!a)



true
Fa&F!a&G(Fa&F!a)
true
!a
true
a
!a
!a
G(Fa&F!a)
a
a
true
Fa&G(Fa&F!a)
!a
a
F!a&G(Fa&F!a)

BA for G(Fa & F!a)



Accepting condition?

不包含R和G算子的LTL-to-BA转换

- 接收条件：true状态

课堂练习：LTL-to-BA

LTL公式为“aUb & a U (c U d)

课堂练习：LTL-to-BA

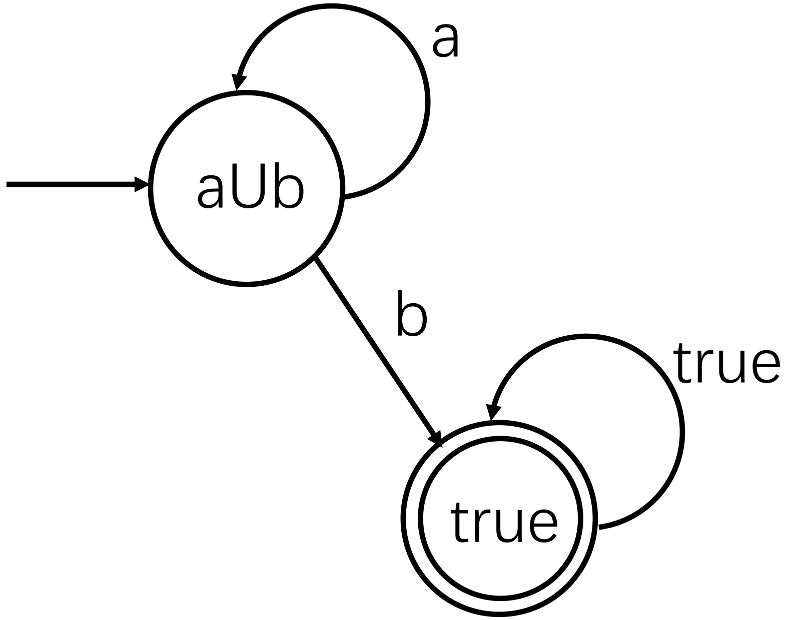
LTL公式为“(aUb)Uc & !c U d

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

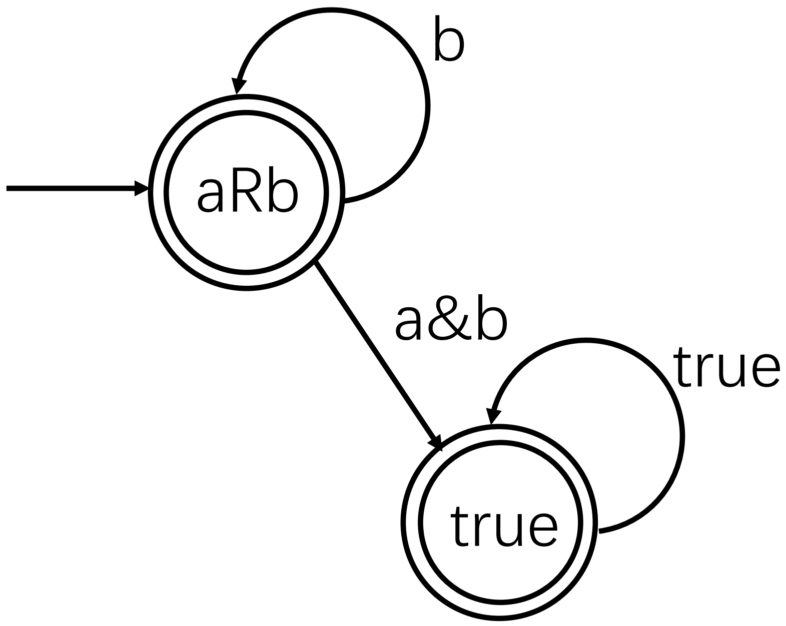
LTL Satisfiability Checking on Automata

LTL公式为“aUb”



LTL Satisfiability Checking on Automata

LTL公式为“aRb”



本章小节

- 介绍线性时态逻辑的语法和语义
- 完成关于线性时态逻辑的一些案例
- 介绍从线性时态逻辑公式到自动机的转化

Stop here

确定接收状态

$$\phi_1 U \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge X(\phi_1 U \phi_2))$$

$$\phi_1 U \phi \equiv \phi_2 \vee (p_{\phi_1 U \phi_2} \wedge \phi_1 \wedge X(\phi_1 U \phi_2))$$

确定接收状态

$$\phi_1 U \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge X(\phi_1 U \phi_2))$$

$$\phi_1 U \phi \equiv \phi_2 \vee (p_{\phi_1 U \phi_2} \wedge \phi_1 \wedge X(\phi_1 U \phi_2))$$

$p_{\phi_1 U \phi_2}$ 为真表示 $\phi_1 U \phi$：不成立

LTL-to-BA

The formula is “G (F a & F ! a)”

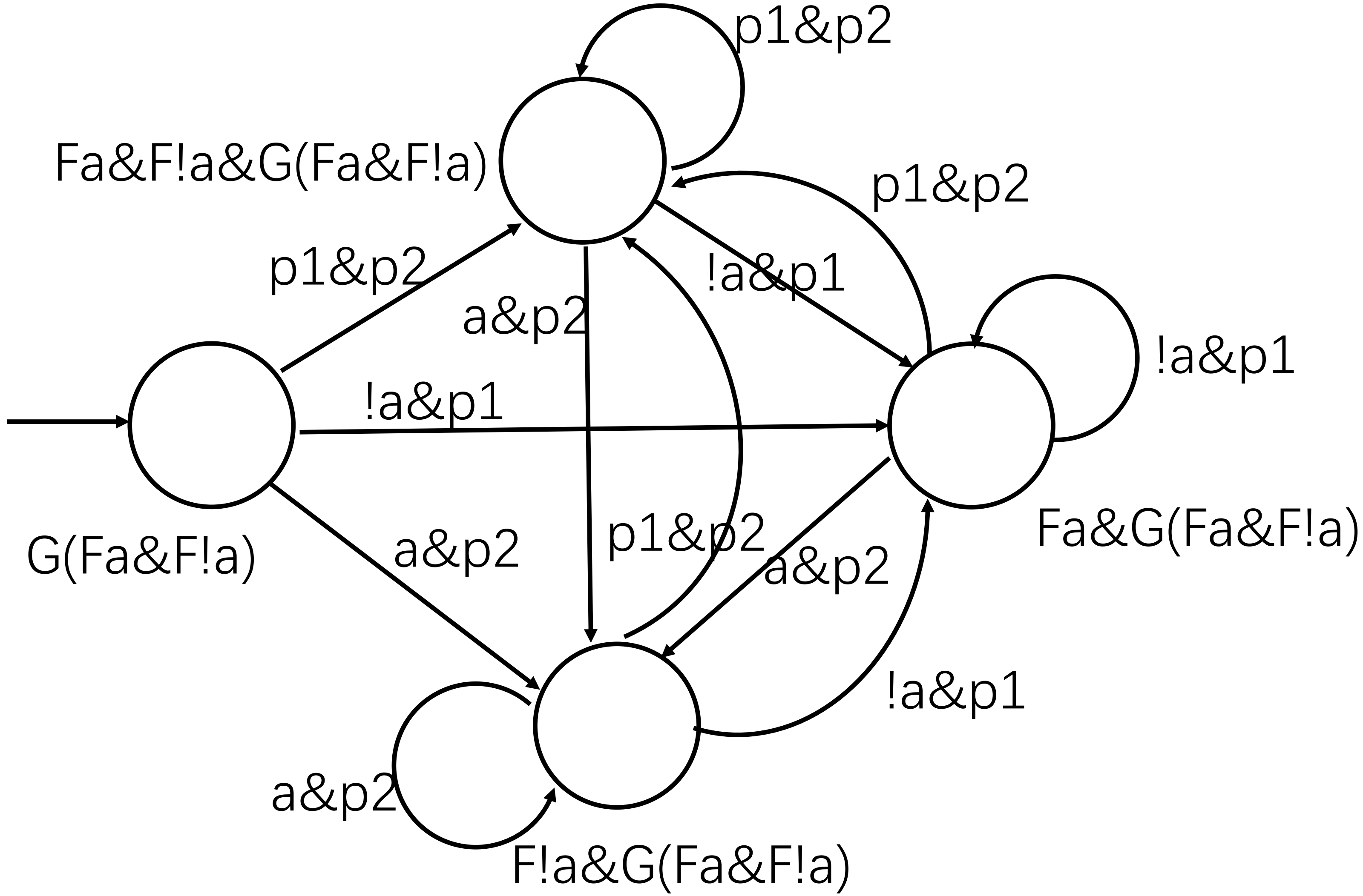
LTL-to-BA

LTL公式为“G (F a & F ! a)”

G(F a & F ! a) = Fa & F !a & X(G (F a & F !a))
= (a | p1&XFa) & (!a | p2& XF!a) & X (G (F a & F !a))
= (a & p2& X(F!a & G (Fa & F!a))) |
(!a & p1& X(Fa & G (Fa & F!a))) |
p1&p2& X(Fa & F!a & G (Fa & F!a))

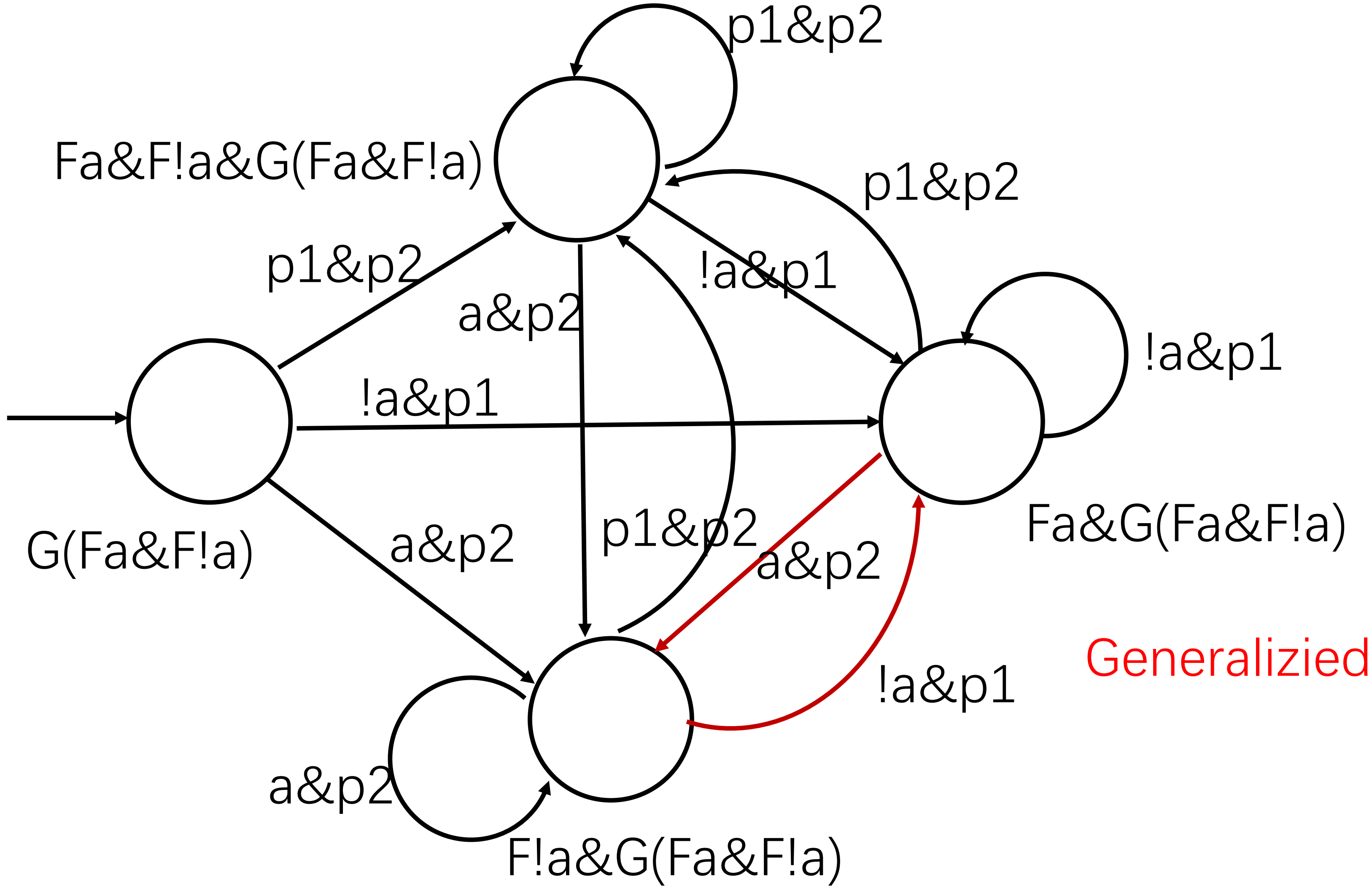
G(F a & F ! a) = F!a & G (Fa & F!a))) = Fa & G (Fa & F!a))) = Fa & F!a & G (Fa & F!a)))

BA for G(Fa & F!a)



p1&p2
Fa&F!a&G(Fa&F!a)
p1&p2
p1&p2
!a&p1
a&p2
!a&p1
!a&p1
G(Fa&F!a)
a&p2
p1&p2
a&p2
Fa&G(Fa&F!a)
!a&p1
a&p2
F!a&G(Fa&F!a)

BA for G(Fa & F!a)



Generalizied BA

一般$B\ddot{u}$ ch i自动机

一个一般$B\ddot{u}$ ch i自动机可以表示成五元组$A = (\Sigma, S, T, I, F)$，其中

- Σ表示字母表的集合
- S表示状态的集合
- $T: S \times \Sigma \times S$表示边的集合
- $I \subseteq S$表示初始状态的集合
- $F \subseteq 2^{\mathrm{T}}$表示接收条件的集合，每个接收条件$F_i \subseteq \mathrm{T}$是一组边的集合

一般Büchi自动机的语义

- 有限自动机$A = (\Sigma, S, T, I, F)$接收一组**无限长度**的字符串
- 给定一个字符串$\eta = a_0 a_1 \cdots$，η在A上的运行轨迹是一条无限长度的状态序列$s_0 s_1 \cdots$，使得s_0是一个初始状态并且(s_i, a_i, s_{i+1})是A上的一条边。
- 一个字符串$\eta = a_0 a_1 \cdots$可以被A接收当且仅当存在η在A上的一条运行轨迹，并且该轨迹**无限次的经过**F中的每个F_i里的某条边。
- $L(A)$用来表示A可以接收的所有字符串的集合。

一般Büchi自动机到Büchi自动机的转化

一般Büchi自动机和Büchi自动机可以相互转化。

一般Büchi自动机到Büchi自动机的转化

一般Büchi自动机和Büchi自动机可以相互转化。

From generalized Büchi automata (GBA) [edit]

Multiple sets of states in acceptance condition can be translated into one set of states by an automata construction, which is known as "counting construction". Let's say $A = (Q,\Sigma,\Delta,q_0,\{F_1,\dots,F_n\})$ is a GBA, where $F_1,\dots,F_n$ are sets of accepting states then the equivalent Büchi automaton is $A' = (Q', \Sigma, \Delta', q'_0, F')$, where

- $Q' = Q \times \{1,\dots,n\}$
- $q'_0 = (q_0,1)$
- $\Delta' = \{ ((q,i), a, (q',j)) \mid (q,a,q') \in \Delta$ and if $q \in F_i$ then $j=((i+1) \bmod n)$ else $j=i \}$
- $F'=F_1\times \{1\}$

Spot – LTL-to-BA translator

Spot: a platform for LTL and ω-automata manipulation

Spot is a C++14 library for LTL, ω-automata manipulation and model checking. It has the following notable features:

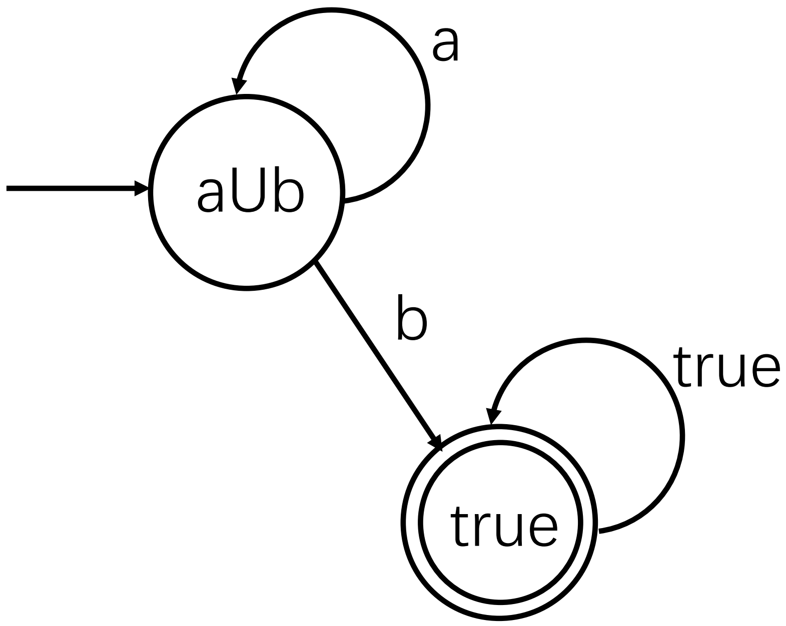
- Support for LTL (several syntaxes supported) and a subset of the linear fragment of PSL.
- Support for ω-automata with arbitrary acceptance condition.
- Support for transition-based acceptance (state-based acceptance is supported by a reduction to transition-based acceptance).
- The automaton parser can read a stream of automata written in any of four syntaxes (HOA, never claims, LBTT, DSTAR).
- Several algorithms for formula manipulation including: simplifying formulas, testing implication or equivalence, testing stutter-invar removing some operators by rewriting, translation to automata, testing membership to the temporal hierarchy of Manna & Pnueli..
- Several algorithms for automata manipulation including: product, emptiness checks, simulation-based reductions, minimization of w DBA, removal of useless SCCs, acceptance-condition transformations, determinization, SAT-based minimization of deterministic automata, etc.

线性时态逻辑LTL可满足性 (Satisfiability)

给定一个LTL公式φ，是否存在一个无限序列使得$\sigma \vDash \varphi$成立？

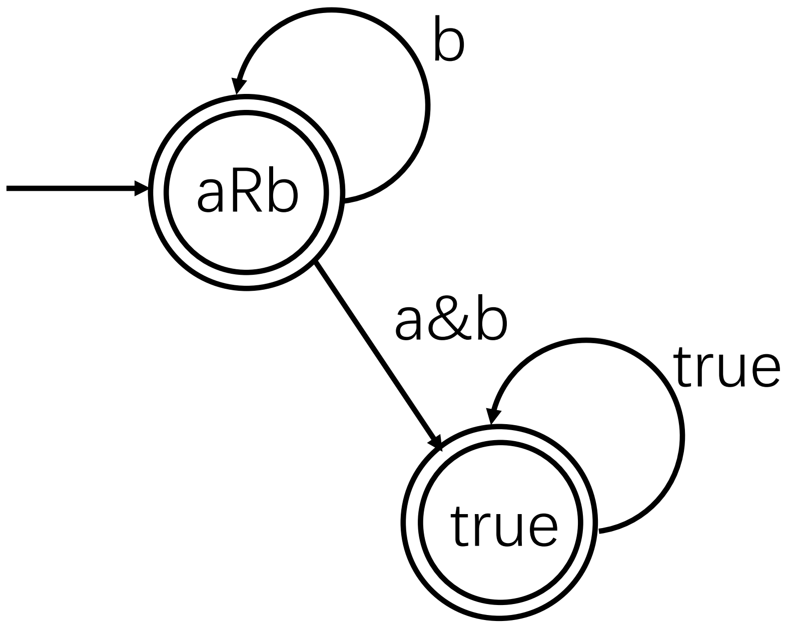
LTL Satisfiability Checking on Automata

LTL公式为“aUb”



LTL Satisfiability Checking on Automata

LTL公式为“aRb”



本章小节

- 介绍线性时态逻辑的语法和语义
- 完成关于线性时态逻辑的一些案例
- 介绍从线性时态逻辑公式到自动机的转化