Unit-7: Linear Temporal Logic

B. Srivathsan

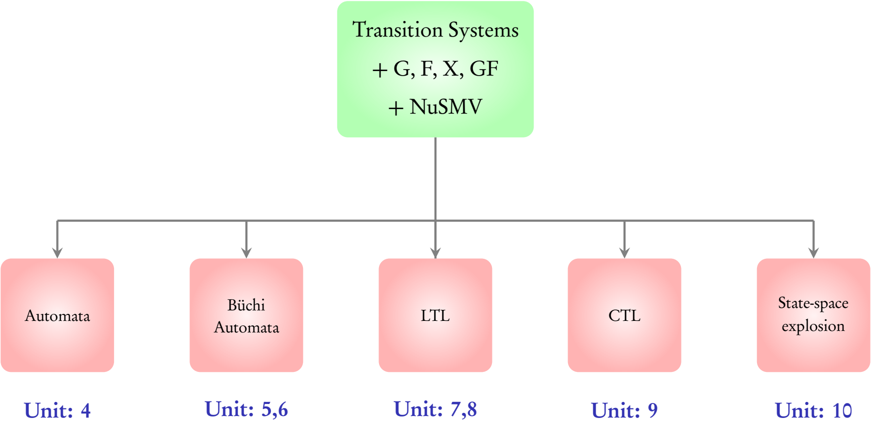
Chennai Mathematical Institute

NPTEL-course

July - November 2015

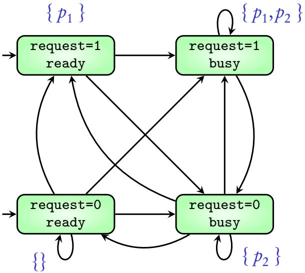
Module 1:

Introduction to LTL


Transition Systems
+ G, F, X, GF
+ NuSMV
Automata
Büchi Automata
LTL
CTL
State-space explosion
Unit: 4
Unit: 5,6
Unit: 7,8
Unit: 9
Unit: 10

$\mathrm{AP} = \{\ p_1,\ p_2\ \}$

Transition System



Property

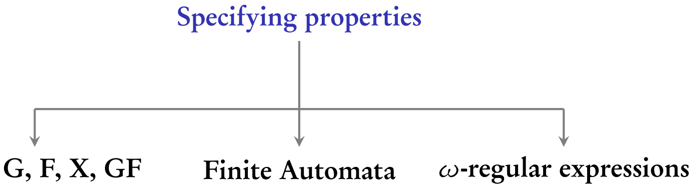
P

Transition system TS satisfies property P if

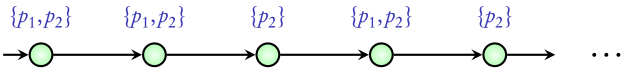
$$\textbf{Traces}(TS) \subseteq P$$

Specifying properties

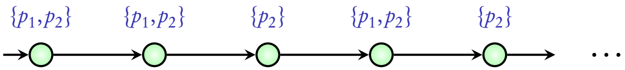
- G, F, X, GF
- Finite Automata
- ω-regular expressions



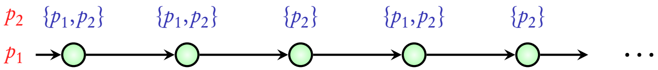
Here: Another formalism - **Linear Temporal Logic**



$\phi :=$

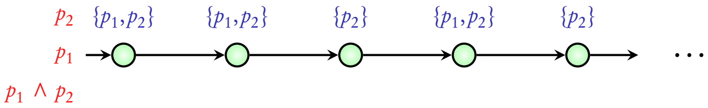


$\phi :=$ true |



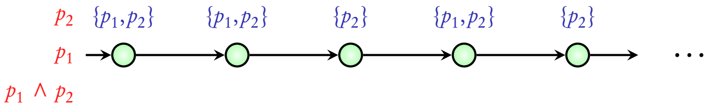
$$\phi := \text{ true} \mid p_i \mid$$

$p_i \in AP$



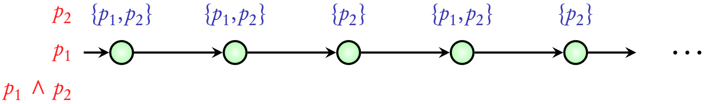
$$\phi := \text{ true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid$$

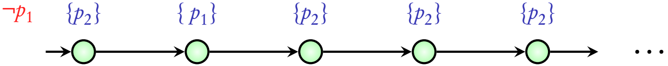
$p_i \in AP$ $\qquad$ ϕ_1, ϕ_2 : LTL formulas



$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid$$

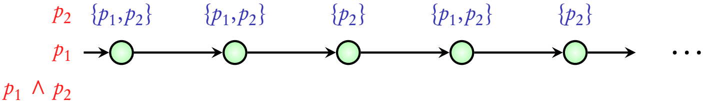
$p_i \in AP$ $\qquad$ ϕ_1, ϕ_2 : LTL formulas

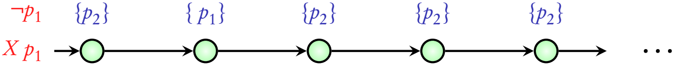




$$\phi := \text{ true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg \phi_1 \mid$$

$p_i \in AP$ $\quad$ ϕ_1, ϕ_2 : LTL formulas





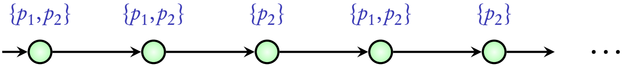
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid$$

$p_i \in AP$ $\qquad$ ϕ_1, ϕ_2 : LTL formulas

p_2

p_1

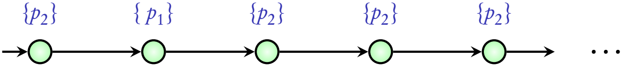
$p_1 \wedge p_2$



$\neg p_1$

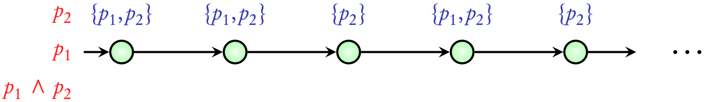
$X\ p_1$

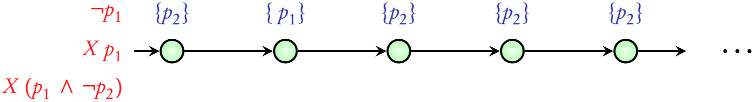
$X\ (p_1 \wedge \neg p_2)$

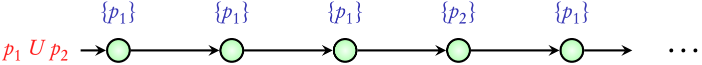


$$\phi := \text{ true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg \phi_1 \mid X\ \phi \mid$$

$p_i \in AP$ $\quad$ ϕ_1, ϕ_2 : LTL formulas





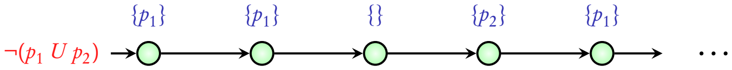


$$\phi := \text{ true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

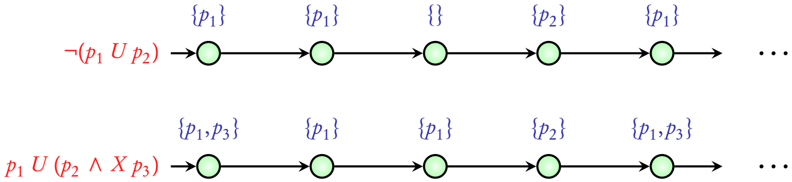
$p_i \in AP$ ϕ_1, ϕ_2 : LTL formulas

$$\phi := \text{ true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

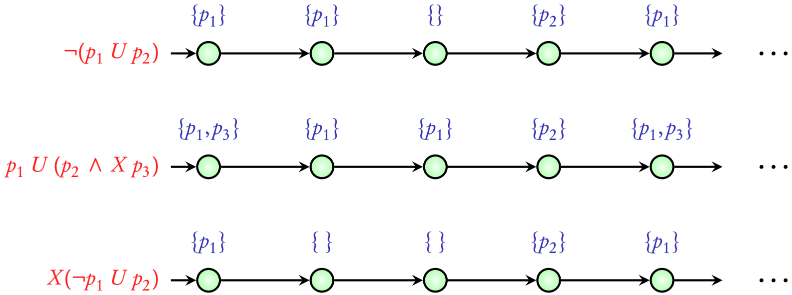
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



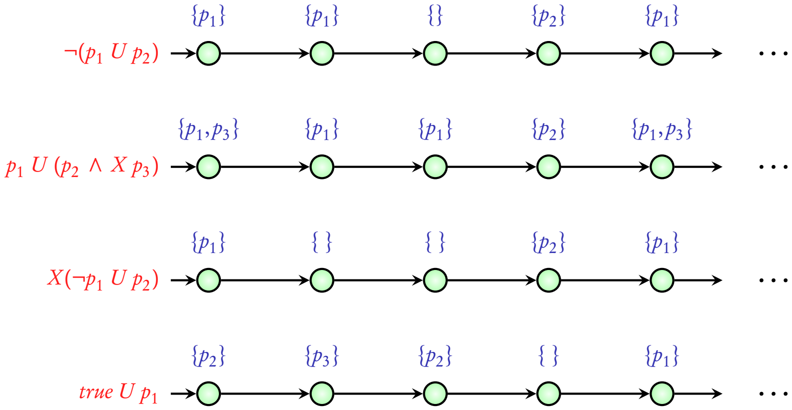
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



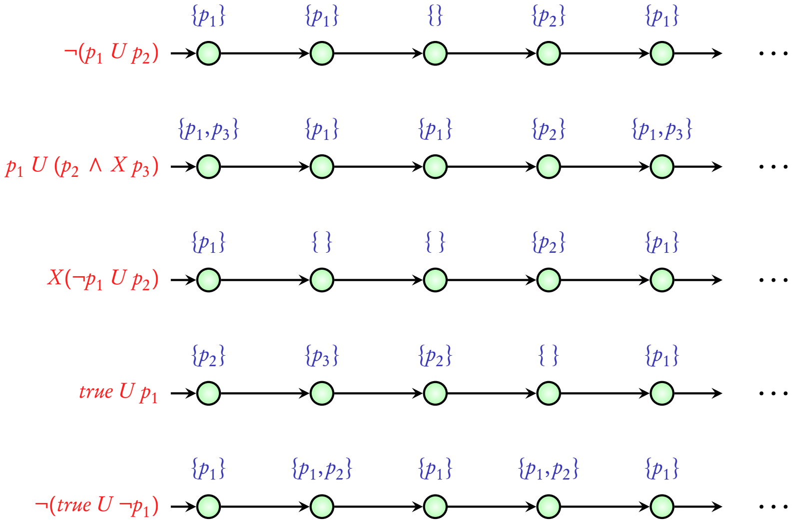
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



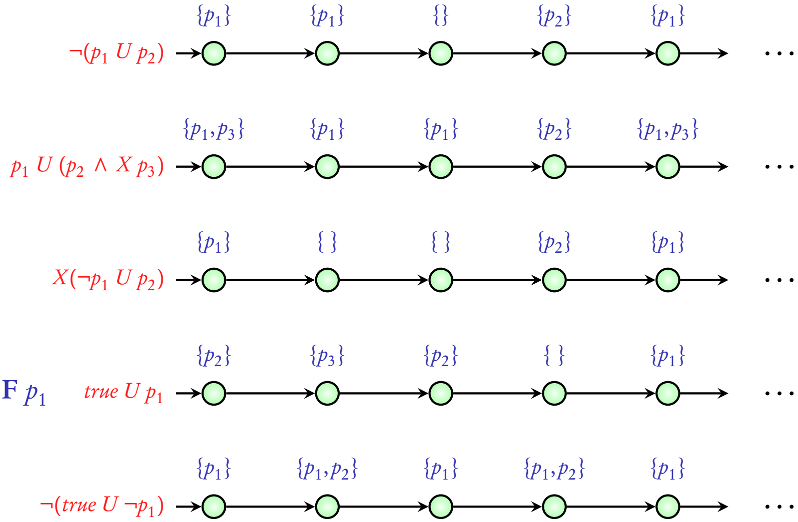
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



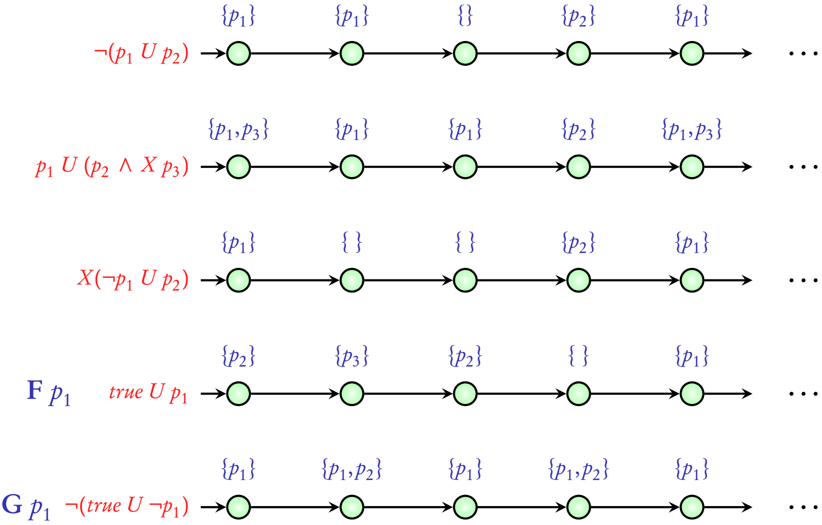
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$



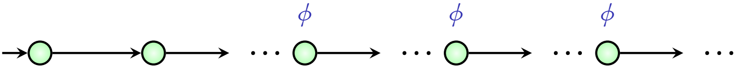
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\,U\,\phi_2$$



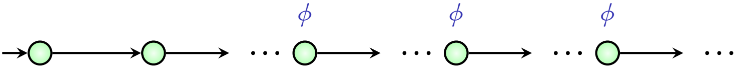
Derived operators

- $\phi_1 \vee \phi_2$: $\neg(\neg\phi_1 \wedge \neg\phi_2)$ **(Or)**
- $\phi_1 \rightarrow \phi_2$: $\neg\phi_1 \vee \phi_2$ **(Implies)**
- $F\ \phi$: $true\ U\ \phi$ **(Eventually)**
- $G\ \phi$: $\neg\ F\ \neg\phi$ **(Always)**

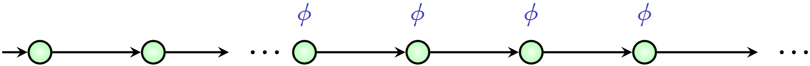
$\mathbf{G}\,\mathbf{F}\,\phi$ **(Infinitely often)**



G F ϕ (Infinitely often)



F G ϕ (Eventually forever)

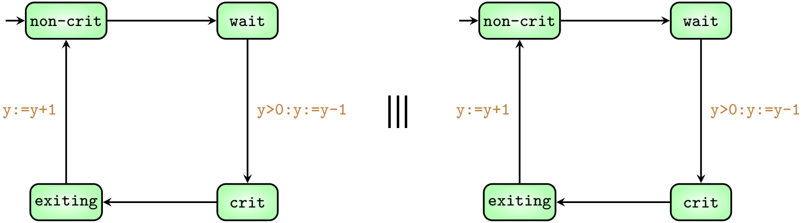


Coming next: More examples

Atomic propositions $\mathrm{AP} = \{ \, crit_1, wait_1, crit_2, wait_2 \, \}$

$crit_1$: `pr1.location=crit` $wait_1$: `pr1.location=wait`

$crit_2$: `pr2.location=crit` $wait_2$: `pr2.location=wait`



- **Safety:** both processes cannot be in critical section simultaneously

$$\mathbf{G}\,(\neg crit_1 \vee \neg crit_2)$$

- **Liveness:** each process visits critical section infinitely often

$$\mathbf{G}\,\mathbf{F}\,crit_1 \wedge \mathbf{G}\,\mathbf{F}\,crit_2$$

Summary

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

$F\,\phi$: $true\ U\ \phi$ **(Eventually)**

$G\,\phi$: $\neg\, F\, \neg\phi$ **(Always)**

Unit-7: Linear Temporal Logic

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 2:

Semantics of LTL

AP-INF $=$ set of **infinite words** over *PowerSet*(**AP**)

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

$$\{ A_0 A_1 A_2 \cdots \in \text{AP-INF} \mid \textbf{each } A_i \text{ contains } p_1 \}$$

$$\{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \dots$$

$$\{ p_1 \} \{ p_1, p_2 \} \{ p_1 \} \{ p_1, p_2 \} \{ p_1 \} \{ p_1, p_2 \} \dots$$

$$\vdots$$

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

$$\{ A_0 A_1 A_2 \cdots \in \text{AP-INF} \mid \textbf{ each } A_i \text{ contains } p_1 \}$$

$\{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \{ p_1 \} \ldots$

$\{ p_1 \} \{ p_1, p_2 \} \{ p_1 \} \{ p_1, p_2 \} \{ p_1 \} \{ p_1, p_2 \} \ldots$

$\vdots$

Property 2: p_1 is true at least once and p_2 is always true

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

Property 1: p_1 is always true

$$\{\, A_0A_1A_2\cdots \in \text{AP-INF} \mid \textbf{each } A_i \text{ contains } p_1 \,\}$$

$\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\{p_1\}\ldots$

$\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\{p_1\}\{p_1,p_2\}\ldots$

$\vdots$

Property 2: p_1 is true at least once and p_2 is always true

$$\{\, A_0A_1A_2\cdots \in \text{AP-INF} \mid \textbf{exists } A_i \text{ containing } p_1 \text{ and } \textbf{every } A_j \text{ contains } p_2 \,\}$$

$\{p_2\}\{p_1,p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_1,p_2\}\{p_2\}\ldots$

$\{p_1,p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\{p_2\}\ldots$

$\vdots$

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

A property over AP is a **subset** of AP-INF

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

A property over AP is a **subset** of AP-INF

LTL can be used to **specify properties**

AP-INF = set of **infinite words** over *PowerSet*(**AP**)

A property over AP is a **subset** of AP-INF

LTL can be used to **specify properties**

LTL can be used to **describe subsets** of AP-INF

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

LTL formula ϕ $\longrightarrow$ Words(ϕ)

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

LTL formula ϕ $\longrightarrow$ Words(ϕ) $\subseteq$ AP-INF

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

LTL formula ϕ $\longrightarrow$ Words(ϕ) $\subseteq$ AP-INF

Words(ϕ): set of words in AP-INF that **satisfy** ϕ

When does a **word satisfy LTL formula** ϕ?

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in$ AP-INF

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ \ldots \ \in$ AP-INF

Every word **satisfies** *true*

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in \text{AP-INF}$

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in$ AP-INF

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in$ AP-INF

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in$ AP-INF

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Word $\sigma : A_0 A_1 A_2 \ldots \in$ AP-INF

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ_2 and
for all $0 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** ϕ_1

$$\textbf{Words}(\phi) \;=\; \{\, \sigma \in \text{AP-}\textsc{inf} \mid \sigma \textbf{ satisfies } \phi \,\}$$

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1A_2A_3\ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_jA_{j+1}\ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_iA_{i+1}\ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

Words(*true*) = AP-INF

σ **satisfies** p_i if $p_i \in A_0$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1A_2A_3\ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_jA_{j+1}\ldots$ **satisfies** ϕ_2 and for all $1 \leq i < j$ $A_iA_{i+1}\ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

Words(*true*) = AP-INF

σ **satisfies** p_i if $p_i \in A_0$

$\text{Words}(p_i) = \{ A_0 A_1 A_2 \ldots \mid p_i \in A_0 \}$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

$\text{Words}(true) = \text{AP-INF}$

σ **satisfies** p_i if $p_i \in A_0$

$\text{Words}(p_i) = \{ A_0 A_1 A_2 \ldots \mid p_i \in A_0 \}$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

$\text{Words}(\phi_1 \wedge \phi_2) = \text{Words}(\phi_1) \cap \text{Words}(\phi_2)$

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

$\text{Words}(\mathit{true}) = \text{AP-INF}$

σ **satisfies** p_i if $p_i \in A_0$

$\text{Words}(p_i) = \{ A_0 A_1 A_2 \ldots \mid p_i \in A_0 \}$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

$\text{Words}(\phi_1 \wedge \phi_2) = \text{Words}(\phi_1) \cap \text{Words}(\phi_2)$

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

$\text{Words}(\neg\phi) = (\text{Words}(\phi))^c$

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

$\text{Words}(\mathit{true}) = \text{AP-INF}$

σ **satisfies** p_i if $p_i \in A_0$

$\text{Words}(p_i) = \{\, A_0A_1A_2\ldots \mid p_i \in A_0 \}$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

$\text{Words}(\phi_1 \wedge \phi_2) = \text{Words}(\phi_1) \cap \text{Words}(\phi_2)$

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

$\text{Words}(\neg\phi) = (\text{Words}(\phi))^c$

σ **satisfies** $X\,\phi$ if $A_1A_2A_3\ldots$ **satisfies** ϕ

$\text{Words}(X\,\phi) = \{\, A_0A_1A_2\ldots \mid A_1A_2\cdots \in \text{Words}(\phi) \,\}$

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_jA_{j+1}\ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_iA_{i+1}\ldots$ **satisfies** ϕ_1

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\,\phi \mid \phi_1\, U\, \phi_2$$

Every word **satisfies** *true*

$\text{Words}(\mathit{true}) = \text{AP-INF}$

σ **satisfies** p_i if $p_i \in A_0$

$\text{Words}(p_i) = \{ A_0 A_1 A_2 \ldots \mid p_i \in A_0 \}$

σ **satisfies** $\phi_1 \wedge \phi_2$ if σ **satisfies** ϕ_1 **and** σ **satisfies** ϕ_2

$\text{Words}(\phi_1 \wedge \phi_2) = \text{Words}(\phi_1) \cap \text{Words}(\phi_2)$

σ **satisfies** $\neg\phi$ if σ **does not satisfy** ϕ

$\text{Words}(\neg\phi) = (\text{Words}(\phi))^c$

σ **satisfies** $X\,\phi$ if $A_1 A_2 A_3 \ldots$ **satisfies** ϕ

$\text{Words}(X\,\phi) = \{ A_0 A_1 A_2 \ldots \mid A_1 A_2 \cdots \in \text{Words}(\phi) \}$

σ **satisfies** $\phi_1\, U\, \phi_2$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ_2 and
for all $1 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** ϕ_1

$$\text{Words}(\phi_1 U \phi_2) = \{ A_0 A_1 A_2 \ldots \mid \exists j. A_j A_{j+1} \cdots \in \text{Words}(\phi_2) \text{ and } \forall\, 0 \leq i < j.\ A_i A_{i+1} \cdots \in \text{Words}(\phi_1) \}$$

F ϕ: $\quad$ $true\ U\ \phi$

F ϕ: *true* U ϕ

σ **satisfies** *true* U ϕ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ
and for all $0 \leq i < j$ $A_i A_{i+1} \ldots$ **satisfies** *true*

F ϕ: $\quad$ $true\ U\ \phi$

σ **satisfies** $true\ U\ \phi$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ

F ϕ: $\quad$ $true\ U\ \phi$

σ **satisfies** $true\ U\ \phi$ $\quad$ if $\quad$ there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ

G ϕ: $\quad$ $\neg F \neg \phi$

F ϕ: $\mathit{true}\ U\ \phi$

σ **satisfies** $\mathit{true}\ U\ \phi$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ

G ϕ: $\neg F \neg \phi$

σ **satisfies** $F \neg \phi$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** $\neg \phi$

F ϕ: $\quad$ *true* $U\ \phi$

σ **satisfies** *true* $U\ \phi$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ

G ϕ: $\quad \neg F \neg \phi$

σ **satisfies** $F \neg \phi$ if there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** $\neg \phi$

σ **satisfies** $\neg F \neg \phi$ if σ **does not satisfy** $F \neg \phi$

F ϕ: $\quad$ *true* $U\ \phi$

σ **satisfies** *true* $U\ \phi$ $\quad$ if $\quad$ there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** ϕ

G ϕ: $\quad$ $\neg\, F \neg\, \phi$

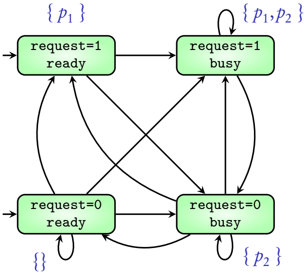
σ **satisfies** $F \neg\, \phi$ $\quad$ if $\quad$ there exists j s.t. $A_j A_{j+1} \ldots$ **satisfies** $\neg\, \phi$

σ **satisfies** $\neg\, F \neg\, \phi$ $\quad$ if $\quad$ σ **does not satisfy** $F \neg\, \phi$

σ **satisfies** $\neg\, F \neg\, \phi$ $\quad$ if $\quad$ for all j $A_j A_{j+1} \ldots$ **satisfies** ϕ

$$AP = \{ p_1, p_2 \}$$

Transition System

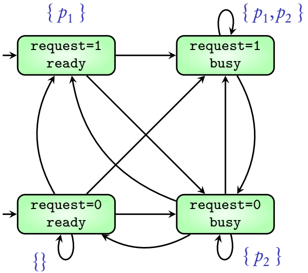


Property

LTL formula ϕ

$$\mathrm{AP} = \{\ p_1,\ p_2\ \}$$

Transition System

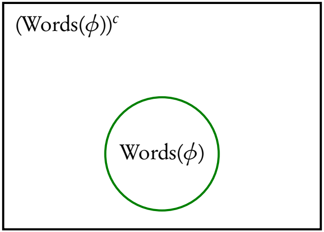


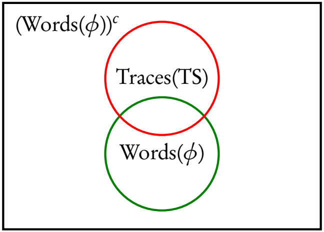
Property

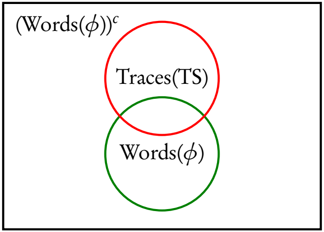
LTL formula ϕ

Transition system *TS* satisfies formula ϕ if

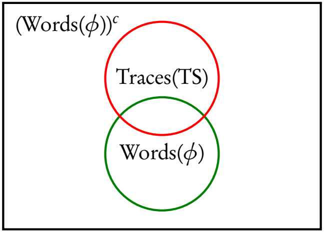
Traces(*TS*) ⊆ Words(ϕ)


(Words(ϕ))^c
Words(ϕ)

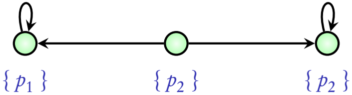

(Words(ϕ))^c
Traces(TS)
Words(ϕ)

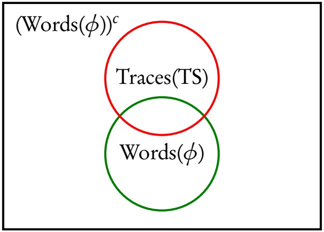


TS **does not satisfy** ϕ TS **does not satisfy** $\neg\phi$

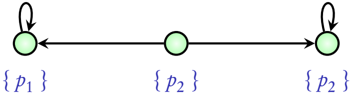


TS **does not satisfy** ϕ TS **does not satisfy** $\neg\phi$





TS **does not satisfy** ϕ TS **does not satisfy** $\neg\phi$



Above TS **does not satisfy** $F\ p_1$ Above TS **does not satisfy** $\neg F\ p_1$

Semantics of LTL

Unit-7: Linear Temporal Logic

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 3:

A Puzzle


MAN
GOAT
WOLF
CABBAGE
RIVER

- There is a **boat** that can be driven by the man

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there
- **Wolf and goat** cannot be left in the same bank if man is not there

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there
- **Wolf and goat** cannot be left in the same bank if man is not there

How can the man shift everyone to the right bank?

Coming next: Solution using LTL model-checking

`man = 0`		`man = 1`
`goat = 0`		`goat = 1`
`wolf = 0`		`wolf = 1`
`cabbage = 0`		`cabbage = 1`

man = 0 | goat = 0 | wolf = 0 | cabbage = 0

man = 1 | goat = 1 | wolf = 1 | cabbage = 1

carry = {g,w,c,0}

`man = 0`		`man = 1`
`goat = 0`		`goat = 1`
	RIVER	
`wolf = 0`		`wolf = 1`
`cabbage = 0`		`cabbage = 1`

```
carry = {g,w,c,0}
```

`man` can carry a passenger which has same value as him

`man = 0`		`man = 1`
`goat = 0`		`goat = 1`
`wolf = 0`	RIVER	`wolf = 1`
`cabbage = 0`		`cabbage = 1`

`carry = {g,w,c,0}`

`man` can carry a passenger which has same value as him

NuSMV demo

Need a path in this transition system which satisfies:

ϕ: `((goat = cabbage | wolf = goat) -> man = goat)`
`U (man & cabbage & goat & wolf)`

Need a path in this transition system which satisfies:

ϕ: `((goat = cabbage | wolf = goat) -> man = goat) U (man & cabbage & goat & wolf)`

NuSMV checks property on all paths

Need a path in this transition system which satisfies:

ϕ: `((goat = cabbage | wolf = goat) -> man = goat)`
`U (man & cabbage & goat & wolf)`

NuSMV checks property on all paths

Check $!\phi$ and look at the **counter-example!**

Summary

LTL model-checking

Use in **planning problem**

Reference

Section 3.3.2

M. Huth and M. Ryan. Logic in Computer Science
(Second Edition, Cambridge University Press)

Unit-7: Linear Temporal Logic

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 3:

A Puzzle


MAN
GOAT
WOLF
CABBAGE
RIVER

- There is a **boat** that can be driven by the man

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time

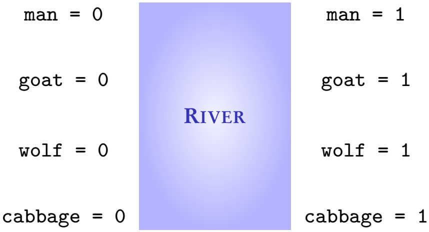
- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there
- **Wolf and goat** cannot be left in the same bank if man is not there

- There is a **boat** that can be driven by the man
- Man can take only **one passenger** in the boat with him at a time
- **Goat and cabbage** cannot be left in the same bank if man is not there
- **Wolf and goat** cannot be left in the same bank if man is not there

How can the man shift everyone to the right bank?

Coming next: Solution using LTL model-checking


man = 0
goat = 0
wolf = 0
cabbage = 0
RIVER
man = 1
goat = 1
wolf = 1
cabbage = 1

man = 0

goat = 0

wolf = 0

cabbage = 0

man = 1

goat = 1

wolf = 1

cabbage = 1

carry = {g,w,c,0}

`man = 0` | RIVER | `man = 1`

`goat = 0` | | `goat = 1`

`wolf = 0` | | `wolf = 1`

`cabbage = 0` | | `cabbage = 1`

`carry = {g,w,c,0}`

`man` can carry a passenger which has same value as him `man = 0` `man = 1`

`goat = 0` `goat = 1`

`wolf = 0` `wolf = 1`

`cabbage = 0` `cabbage = 1`

`carry = {g,w,c,0}`

`man` can carry a passenger which has same value as him

NuSMV demo

Need a path in this transition system which satisfies:

ϕ: `((goat = cabbage | wolf = goat) -> man = goat)`
`U (man & cabbage & goat & wolf)`

Need a path in this transition system which satisfies:

ϕ:
```
((goat = cabbage | wolf = goat) -> man = goat)
        U (man & cabbage & goat & wolf)
```

NuSMV checks property on all paths

Need a path in this transition system which satisfies:

ϕ: `((goat = cabbage | wolf = goat) -> man = goat)`
`U (man & cabbage & goat & wolf)`

NuSMV checks property on all paths

Check !ϕ and look at the **counter-example!**

Summary

LTL model-checking

Use in **planning problem**

Reference

Section 3.3.2

M. Huth and M. Ryan. Logic in Computer Science
(Second Edition, Cambridge University Press)