Computer aided verification

Lecture 2:

Expressing properties of systems in Linear Temporal Logic (LTL)

Sławomir Lasota
University of Warsaw
– Modeling systems as Kripke structures
– LTL
– Expressivity
– Model checking
Modeling systems as Kripke structures
Kripke structure (model)

**Def.:** Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, $P$ - atomic properties (propositional variables)
Kripke structure (model)

Def.: Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, $P$ - atomic properties (propositional variables)

Often we assume that $\rightarrow$ is total:

\[ \forall s \in S. \exists s' \in S. s \rightarrow s' \]

no deadlock!
Def.: Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, $P$ - atomic properties (propositional variables)

Often we assume that $\rightarrow$ is total:

\[ \forall s \in S. \ \exists s' \in S. \ s \rightarrow s' \]

no deadlock!
Proc$_1$

private:

\[
\begin{align*}
\text{wants}_1 & := \text{true}; \\
\text{turn} & := 2;
\end{align*}
\]

/* attempts */

\[
\begin{align*}
(\neg \text{wants}_2 \lor \text{turn} = 1);
\end{align*}
\]

/* critical section */

\[
\begin{align*}
\text{wants}_1 & := \text{false}; \\
\text{goto private}
\end{align*}
\]
Abstraction: program \(\mapsto\) Kripke structure

Proc₁

private:

\[
\begin{align*}
wants₁ & := \text{true}; \\
turn & := 2; \\
\end{align*}
\]

/* attempts */

\[
(!\text{wants₂} \lor turn = 1);
\]

/* critical section */

\[
wants₁ := \text{false}; \\
goto \text{private}
\]

---

Proc₁ : 

\[
\begin{array}{c}
\text{N₁} \\
\rightarrow T₁ \\
\rightarrow C₁ \\
\end{array}
\]

\[
\begin{align*}
wants₁ & := \text{true}; turn := 2; \\
\neg\text{wants₂} \lor turn = 1 \\
wants₁ & := \text{false}
\end{align*}
\]
Abstraction: program $\mapsto$ Kripke structure

Proc₁

```c
private:
    wants₁ := true;
    turn := 2; /* attempts */

    (!wants₂ ∨ turn = 1); /* critical section */
    wants₁ := false;
    goto private
```

Proc₂

```c
private:
    wants₂ := true;
    turn := 1; /* attempts */

    (!wants₁ ∨ turn = 2); /* critical section */
    wants₂ := false;
    goto private
```
Abstraction: program $\mapsto$ Kripke structure

- $N_i$ private section
- $T_i$ attempt to enter critical section
- $C_i$ critical section
Abstraction: program $\mapsto$ Kripke structure

Proc$_1$:

$N_1$ \(\xrightarrow{\text{wants}_1:=\text{true}; \text{turn}:=2} T_1 \xrightarrow{\neg \text{wants}_2 \lor \text{turn}=1} C_1 \)

$N_1$ \(\xleftarrow{\text{wants}_1:=\text{false}} T_1 \)

Proc$_2$:

$N_2$ \(\xrightarrow{\text{wants}_2:=\text{true}; \text{turn}:=1} T_2 \xrightarrow{\neg \text{wants}_1 \lor \text{turn}=2} C_2 \)

$N_2$ \(\xleftarrow{\text{wants}_2:=\text{false}} T_2 \)

Proc$_1$ | Proc$_2$:

Diagram of the Kripke structure showing the transitions and conditions for the states $N_1$, $N_2$, $T_1$, $T_2$, $C_1$, and $C_2$. The diagram includes arrows indicating the transitions between states and the conditions for reaching those states.
Abstraction: program $\mapsto$ Kripke structure
Abstraction: program $\mapsto$ Kripke structure

\[ P = \{\text{crit}_1, \text{crit}_2\} \]
Abstraction: program $\mapsto$ Kripke structure

$P = \{\text{crit}_1, \text{crit}_2\}$
Kripke structure $\rightarrow$ tree
\begin{align*}
v'_0 & := \neg v_0 \\
v'_1 & := v_0 \oplus v_1 \\
v'_2 & := (v_0 \land v_1) \oplus v_2
\end{align*}

**Question:** What is the induced Kripke structure?
And when the state space is huge, or even infinite?
And when the state space is huge, or even infinite?

- abstraction

- symbolic approaches
**Def.:** Path (run) is a maximal sequence

\[ \Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \]

**Notation:** \(|\Pi|\) – number of states in \(\Pi\)
**Def.:** Path (run) is a maximal sequence

\[ \Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \]

**Notation:** \(|\Pi|\) – number of states in \(\Pi\)

LTL says about paths.

In a Kripke structure \(M\), formula \(\phi \in \text{LTL}\) is interpreted as follows:

for every path \(\Pi\) such that \(s_0 \in S_{\text{init}}\), \(\phi\) holds.
**Def.:** Path (run) is a maximal sequence

\[ \Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \]

**Notation:** \( |\Pi| \) – number of states in \( \Pi \)

---

**LTL says about paths.**

In a Kripke structure \( M \), formula \( \phi \in \mathsf{LTL} \) is interpreted as follows:

for every path \( \Pi \) such that \( s_0 \in S_{\text{init}} \), \( \phi \) holds.

**Notation:** \( M \vDash \phi \), \( \Pi \vDash \phi \)
LTL
Def.: LTL (Linear Temporal Logic)

\[ \phi \ := \ p \ | \ \neg \phi \ | \ \phi_1 \land \phi_2 \ | \ X \phi \ | \ \phi_1 \ U \phi_2 \]
**Def.:** LTL (Linear Temporal Logic)

\[
\phi := p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid X \phi \mid \phi_1 U \phi_2
\]

- **p:**
  - \(p\)
  - \(\Rightarrow\)
  - \(?\)
  - \(\Rightarrow\)
  - \(?\)
  - \(\Rightarrow\)
  - \(\ldots\)

- **Xp:**
  - \(?\)
  - \(\Rightarrow\)
  - \(p\)
  - \(\Rightarrow\)
  - \(?\)
  - \(\Rightarrow\)
  - \(\ldots\)

- **p U q:**
  - \(p\)
  - \(\Rightarrow\)
  - \(p\)
  - \(\Rightarrow\)
  - \(p\)
  - \(\Rightarrow\)
  - \(q\)
  - \(\Rightarrow\)
  - \(?\)
  - \(\Rightarrow\)
  - \(\ldots\)
**Def.:** LTL (Linear Temporal Logic)

\[ \phi := p \mid \neg \phi \mid \phi_1 \land \phi_2 \mid X \phi \mid \phi_1 \mathbf{U} \phi_2 \]

\[
\begin{align*}
\text{Example:} \quad 
\neg \text{starts} \mathbf{U} \text{key}, \quad 
\neg \text{starts} \mathbf{U} \neg \text{starts} \land \text{key}
\end{align*}
\]
Semantics:

\[ \Pi = s_0 \xrightarrow{} s_1 \xrightarrow{} s_2 \xrightarrow{} \ldots \]

\[ \Pi \models p \iff p \in L(s_0) \]

\[ \Pi \models \neg \phi \iff \Pi \not\models \phi \]

\[ \Pi \models \phi_1 \land \phi_2 \iff \Pi \models \phi_1 \text{ and } \Pi \models \phi_2 \]

\[ \Pi \models X \phi \iff \Pi^1 \models \phi, \text{ where } \Pi^i = s_i \rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow \ldots \]

\[ \Pi \models \phi_1 \mathbf{U} \phi_2 \iff \exists i < |\Pi|. \Pi^i \models \phi_2 \land \forall j < i. \Pi^j \models \phi_1 \]

\[ M \models \phi \iff \Pi \models \phi \text{ for all paths } \Pi \text{ starting in an initial state} \]
Alternative semantics:

\[ \Pi = s_0 \xrightarrow{} s_1 \xrightarrow{} s_2 \xrightarrow{} \cdots \]

\[ \Pi \models \phi \iff \Pi, 0 \models \phi \]

\[ \Pi, n \models p \iff p \in L(s_n) \]

\[ \Pi, n \models \neg \phi \iff \Pi, n \not\models \phi \]

\[ \Pi, n \models \phi_1 \land \phi_2 \iff \Pi, n \models \phi_1 \text{ and } \Pi, n \models \phi_2 \]

\[ \Pi, n \models X \phi \iff \Pi, n + 1 \models \phi \]

\[ \Pi, n \models \phi_1 U \phi_2 \iff \exists i \geq n. \Pi, i \models \phi_2 \land \forall j \in [n \ldots i - 1]. \Pi, j \models \phi_1 \]

\[ M \models \phi \iff \Pi \models \phi \text{ for all paths } \Pi \text{ starting in an initial state} \]
An LTL formula essentially specifies a subset of $\mathcal{P}(P)^\omega$.
**Question:** How to write

- **always** φ
  
  \[ \phi \rightarrow \phi \rightarrow \phi \rightarrow \phi \rightarrow \phi \rightarrow \ldots \]

- **eventually** φ
  
  \[ ? \rightarrow ? \rightarrow ? \rightarrow \phi \rightarrow ? \rightarrow \ldots \]
Question: How to write

always $\phi$

eventually $\phi$

Notation:

$$F \phi \equiv \text{true} U \phi$$
$$G \phi \equiv \neg F \neg \phi$$
$$\phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2)$$

$$X \phi \equiv \bigcirc \phi$$

$$F \phi \equiv \Diamond \phi$$
$$G \phi \equiv \Box \phi$$
Typical properties

**safety**

Bad will never happen

**liveness**

Good will eventually happen
Typical properties

safety

liveness

possibility
Typical properties

- **Safety**:  
  \[ G \phi \]
  \[ G \neg (cr_1 \land cr_2) \]

- **Liveness**:  
  \[ F \phi \]
  \[ F \text{ granted} \]

- **Possibility**:  
  \[ G \neg \phi \]
  \[ \neg G \neg \phi \]
  \[ G \neg \text{ooc} \]
Classification of properties

**Def.:** Property = subset of $\mathcal{P}(P)^\omega$

**Safety properties** $X$

negative decision *always* after finitely many steps
Classification of properties

**Def.**: Property = subset of $\mathcal{P}(P)^\omega$

**Safety properties $X$**

negative decision *always* after finitely many steps

if $\pi \notin X$ then there is a prefix $\rho < \pi$ such that

$\rho < \pi'$ implies $\pi' \notin X$
Def.: Property = subset of $\mathcal{P}(P)^\omega$

**Safety properties** $X$

negative decision *always* after finitely many steps

if $\pi \notin X$ then there is a prefix $\rho < \pi$ such that
$\rho < \pi'$ implies $\pi' \notin X$

**Liveness properties** $X$

negative decision *never* after finitely many steps

for every $\rho$ exists $\pi > \rho$ such that $\pi \in X$
Example properties

- infinitely often $\phi$
- almost always $\phi$
- "weak" $\phi_1 \cup \phi_2 : \phi_2$ not necessarily
- if req then granted later
Example properties

- infinitely often $\phi$
- almost always $\phi$
- "weak" $\phi_1 U \phi_2 : \phi_2$ not necessarily
- if req then granted later

G F $\phi$
F G $\phi$
G $\phi_1 \lor \phi_1 U \phi_2$

req $\implies$ X F granted

- fairness: if stubbornly req then granted
  "weak": stubbornly = almost always
  "strong": stubbornly = infinitely often

? ?
Example properties

- infinitely often $\phi$
- almost always $\phi$
- "weak" $\phi_1 U \phi_2 : \phi_2$ not necessarily
- if $\text{req}$ then granted later

- fairness: if stubbornly $\text{req}$ then granted
  
  "weak": stubbornly = almost always
  
  "strong": stubbornly = infinitely often
Example properties

- infinitely often $\phi$
  \[ G F \phi \]
- almost always $\phi$
  \[ F G \phi \]
- "weak" $\phi_1 U \phi_2 : \phi_2 \text{ not necessarily}
  \[ G \phi_1 \lor \phi_1 U \phi_2 \]
- if $req$ then granted later
  \[ req \implies X F \text{ granted} \]
- fairness: if stubbornly $req$ then granted
  "weak": stubbornly = almost always
  \[ F G req \implies F \text{ granted} \]
  "strong": stubbornly = infinitely often
  \[ G F req \implies F \text{ granted} \]
- fairness (variant):
  "weak":
  \[ F G req \implies G F \text{ granted} = G ( F G req \implies F \text{ granted}) \]
  "strong":
  \[ G F req \implies G F \text{ granted} = G ( G F req \implies F \text{ granted}) \]
De Morgan’s laws

$\phi_1 \lor \phi_2 \equiv \neg(\neg \phi_1 \land \neg \phi_2)$

$G \phi \equiv \neg F \neg \phi$

$? \equiv \neg X \neg \phi$
De Morgan’s laws

\[ \phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2) \]

\[ G \phi \equiv \neg F \neg \phi \]

\[ ? \equiv \neg (\neg \phi \lor \neg \psi) \]

\[ X \phi \equiv \neg X \neg \phi \]
De Morgan’s laws

\[ \phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2) \]

\[ \text{G } \phi \equiv \neg \text{F } \neg \phi \]

\[ \phi \text{ R } \psi \equiv \neg (\neg \phi \text{ U } \neg \psi) \]

\[ \neg \phi \text{ U } \neg \psi \]

\[ \Pi \models \phi \text{ R } \psi \text{ iff } ? \]
De Morgan’s laws

\[ \phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2) \]

\[ G \phi \equiv \neg F \neg \phi \]

\[ \phi \mathcal{R} \psi \equiv \neg (\neg \phi \mathcal{U} \neg \psi) \]

\[ \neg \phi \mathcal{U} \neg \psi \]

\[ \Pi \models \phi \mathcal{R} \psi \text{ iff } \forall i < |\Pi|. \ (\forall j < i. \Pi^j \models \neg \phi) \implies \Pi^i \models \psi \]
De Morgan’s laws

\[
\phi_1 \lor \phi_2 \equiv \neg(\neg \phi_1 \land \neg \phi_2)
\]

\[
\mathbf{G} \phi \equiv \neg \mathbf{F} \neg \phi
\]

\[
\phi \mathbf{R} \psi \equiv \neg(\neg \phi \mathbf{U} \neg \psi)
\]

\[
\neg \phi \mathbf{U} \neg \psi
\]

\[
\Pi \models \phi \mathbf{R} \psi \iff \forall i < |\Pi|. (\forall j < i. \Pi^j \models \neg \phi) \implies \Pi^i \models \psi
\]

\[
\phi \mathbf{R} \psi \equiv \neg(\neg \phi \mathbf{U} \neg \psi) \equiv \psi \mathbf{U} (\psi \land \phi) \lor \mathbf{G} \psi \equiv \psi \mathbf{W} (\psi \land \phi)
\]
$\phi \text{U} \psi \equiv \psi \lor (\phi \land X(\phi \text{U} \psi))$
\[ \phi U \psi \equiv \psi \lor (\phi \land X(\phi U \psi)) \]

\[ \phi R \psi \equiv \psi \land (\phi \lor X(\phi R \psi)) \]
\neg(\phi_1 \land \phi_2) \equiv \neg\phi_1 \lor \neg\phi_2

\neg F \phi \equiv G \neg \phi

\neg G \phi \equiv F \neg \phi

\neg X \phi \equiv X \neg \phi
Pushing negation down

\[ \neg (\phi_1 \land \phi_2) \equiv \neg \phi_1 \lor \neg \phi_2 \]

\[ \neg F \phi \equiv G \neg \phi \]

\[ \neg G \phi \equiv F \neg \phi \]

\[ \neg X \phi \equiv X \neg \phi \]

\[ \neg (\phi U \psi) \equiv (\neg \psi) W (\neg \phi \land \neg \psi) \]

\[ \neg (\phi U \psi) \equiv \neg \phi R \neg \psi \]

why not in this way?
Expressivity
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

(1') . . . strictly beforehand

(2) every \( b \) is proceeded by \( a \) that appears after last \( b \), if any before

(3) alternating blocks of \( a \) and \( b \) ("relay")
\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ F b \iff (\neg b \cup a) \]
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ F b \implies (\neg b \cup a) \]

\[ \equiv \neg b W a \equiv Pr(a, b) \]
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ F b \implies (\neg b U a) \]

\[ \equiv \neg b W a \equiv Pr(a, b) \]

(1’)...strictly beforehand
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ F b \implies (\neg b \cup a) \]

\[ \equiv \neg b \mathcal{W} a \equiv Pr(a, b) \]

(1') . . . strictly beforehand

\[ F b \implies (\neg b \cup (a \land \neg b)) \]
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ Fb \iff (\neg b \cup a) \]
\[ \equiv \neg b \mathtt{W} a \equiv Pr(a, b) \]

(1’) . . . strictly beforehand

\[ Fb \iff (\neg b \cup (a \land \neg b)) \]
\[ \equiv \neg b \mathtt{W} (a \land \neg b) \equiv a \mathtt{R} \neg b \equiv SPr(a, b) \]
\( P = \{a, b, \ldots\} \)

(1) if \( b \) then some \( a \) was beforehand

\[
F b \iff (\neg b \cup a)
\]

\[
\equiv \neg b W a \equiv Pr(a, b)
\]

(1') ... strictly beforehand

\[
F b \iff (\neg b \cup (a \land \neg b))
\]

\[
\equiv \neg b W (a \land \neg b) \equiv a R \neg b \equiv SPr(a, b)
\]

(2) every \( b \) is proceeded by \( a \) that appears after last \( b \), if any before
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ F b \implies (\neg b U a) \]
\[ \equiv \neg b W a \equiv Pr(a, b) \]

(1’) . . . strictly beforehand

\[ F b \implies (\neg b U (a \land \neg b)) \]
\[ \equiv \neg b W (a \land \neg b) \equiv a R \neg b \equiv SPr(a, b) \]

(2) every \( b \) is proceeded by \( a \) that appears after last \( b \), if any before

\[ Pr(a, b) \land G (b \implies X Pr(a, b)) \]
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[ Fb \iff (\neg b \cup a) \]
\[ \equiv \neg b W a \equiv Pr(a, b) \]

(1') . . . strictly beforehand

\[ Fb \iff (\neg b \cup (a \land \neg b)) \]
\[ \equiv \neg b W (a \land \neg b) \equiv a R \neg b \equiv SPr(a, b) \]

(2) every \( b \) is proceeded by \( a \) that appears after last \( b \), if any before

\[ Pr(a, b) \land G(b \implies X Pr(a, b)) \]

(3) alternating blocks of \( a \) and \( b \) ("relay")
Write a formula . . .

\[ P = \{a, b, \ldots\} \]

(1) if \( b \) then some \( a \) was beforehand

\[
F b \implies (\neg b U a)
\]

\[
\equiv \neg b W a \equiv Pr(a, b)
\]

(1') . . . strictly beforehand

\[
F b \implies (\neg b U (a \land \neg b))
\]

\[
\equiv \neg b W (a \land \neg b) \equiv a R \neg b \equiv SPr(a, b)
\]

(2) every \( b \) is proceeded by \( a \) that appears after last \( b \), if any before

\[
Pr(a, b) \land G (b \implies X Pr(a, b))
\]

(3) alternating blocks of \( a \) and \( b \) (”relay”)

\[
G ( (a \implies a W (\neg a \land b)) \land (b \implies \ldots) )
\]
What is inexpressible?

(1) on every path a state appears such that
in every successor state
(on every path) \( a \) holds

(1’) on some path . . .

(2) on every path a state appears such that
in every following state \( a \) holds

– p. 35/43
What is inexpressible?

(1) on every path a state appears such that
    in every successor state
    (on every path) \( a \) holds

(1') on some path …

(2) on every path a state appears such that
    in every following state \( a \) holds
    too weak!

\[ F \ X \ a \ ? \]
\[ F \ G \ a \ ? \]

\[ a \rightarrow \neg a \rightarrow a \in\] \( F \ G \ a \)
What is inexpressible? (cont.)

(3) even(a): on every even position $a$
(3) even(a): on every even position $a$ 

(3') oddeven(a): on every even position $a$

and on every odd position $\neg a$
(3) even(a): on every even position $a$

(3') oddeven(a): on every even position $a$

and on every odd position $\neg a$

\[ G \left( (a \implies X \neg a) \land (\neg a \implies X a) \right) \]
(3) even(a): on every even position \( a \)  

(3’) oddeven(a): on every even position \( a \) 
and on every odd position \( \neg a \)

\[
G \left( (a \implies X \neg a) \land (\neg a \implies X a) \right)
\]

(4) from every reachable state some initial state is reachable
Expressivity

**Thm.**  LTL = LTL(X, U) is more expressive than LTL(X, F)

**Thm.**  LTL = FO(≤)

**Thm.**  Past temporal connectives:

\[ X^{-1}, U^{-1}, F^{-1}, G^{-1} \]

- do not increase expressive power.

**Thm.**  LTL(F, G, F^{-1}, G^{-1}) = ?
Model checking
Decision problems

Model checking
- input: $M, \phi$
- question: $M \models \phi$?

Satisfiability
- input: $\phi$
- question: $\exists M. M \models \phi$? ($\exists \Pi. \Pi \models \phi$?)

PSPACE-complete
Complexity of model checking

\[ |M| \cdot 2^{O(|\phi|)} \]

\[ 2^{O(|\phi|)} \quad \text{OK} \]

\[ |M| \quad \text{too much!} \]
Algorithm

(1) $M \mapsto A_M$

(2) $\neg \phi \mapsto A_{\neg \phi}$

(3) $L(A_M \times A_{\neg \phi}) = \emptyset$?

- yes $\rightarrow M \models \phi$

- no $\rightarrow \neg (M \models \phi)$, counterexample = a path in $M$

LTL $\rightarrow \omega$-automata
(1) \( M \mapsto A_M \)

(2) \( \neg \phi \mapsto A_{\neg \phi} \)

(3) \( L(A_M \times A_{\neg \phi}) = \emptyset? \)

yes \( \rightarrow M \models \phi \)

no \( \rightarrow \neg (M \models \phi), \) counterexample = a path in \( M \)

\[ \phi = G(p \implies X F q) \]

\[ A_{\neg \phi} = \]

---

LTL \( \rightarrow \omega \)-automata

\[ \frac{p}{s_0} \hspace{1cm} \frac{\neg q}{s_1} \]