Introduction to Temporal Logic

Mads Dam
Theoretical Computer Science
KTH, 2015
About the Course

• Lecturers
• Content
• Examination
• Lecture material
• Registration
What is TL About?

Formalised properties of time-varying systems

• What time-varying systems?
• What properties?
• Algorithms
• Proof systems

This is why we think formalisation pays off

Some form of tractability

Our tasks:
• Show we can do useful stuff with this
• Understand and compare set-ups for expressiveness and tractability
What Time-Varying Systems?

- Continuous real-valued functions?
- Discrete program traces?
- Execution trees?
- Automata?
- Markov chains?
- Java code?
- Distributed processes?
- Real time? Or implicit time?
- Histories or future?
- Finite or infinite?
- Linear or branching? Tree shaped? Graph shaped?
Default Choice – Traces/Paths/Runs

Time is discrete
Starts at 0
Goes on forever

Time points decorated by events

Or conditions/truth assignments/valuations

Or execution traces
How Are Traces Produced?

• Maximal runs through a transition system/automaton
  – \((Q,R,Q_0)\)
  – \(Q\) set of states
  – \(R \subseteq Q \times Q\) transition relation, total
  – \(Q_0 \subseteq Q\) initial states
  – Traces/runs \(w = q_0 R q_1 R \ldots R q_{n-1} R q_n R \ldots\)

In practice:
• Take your favourite programming/modeling language
• Equip it with discrete transition semantics
• Determine what should be observable events / conditions / execution states
• (Add looping at the end to get traces to be infinite)
• Off you go
Example - Concurrent While Language

Commands:
Cmd ::= skip | x := e | Cmd;Cmd | if e Cmd Cmd
    | while e Cmd | await e Cmd | spawn Cmd
    | Cmd || Cmd

Stores σ ∈ x ↦_{fin} v ∈ Val

Configurations c ::= σ | <Cmd, σ>
Example II

Transitions:

- $\sigma \rightarrow \sigma$ (... just to get looping ...)
- $\langle \text{skip}, \sigma \rangle \rightarrow \sigma$
- $\langle x := e, \sigma \rangle \rightarrow \sigma[x \mapsto ||e|| \sigma]$
- $\langle \text{Cmd}_1; \text{Cmd}_2, \sigma \rangle \rightarrow \langle \text{Cmd}_1', \text{Cmd}_2, \sigma' \rangle$
  if $\langle \text{Cmd}_1, \sigma \rangle \rightarrow \langle \text{Cmd}_1', \sigma' \rangle$
- $\langle \text{Cmd}_1; \text{Cmd}_2, \sigma \rangle \rightarrow \langle \text{Cmd}_2, \sigma' \rangle$
  if $\langle \text{Cmd}_1, \sigma \rangle \rightarrow \sigma'$
- (... remaining rules in class ...)

Conditions: Boolean/FO expressions in $\text{dom}(\sigma_i)$

Traces: $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{n-1} \rightarrow c_n \rightarrow \ldots$
Linear Time Temporal Logic, LTL

Logic of temporal relations between events in a trace:
- Invariably (along this execution) \( x \cdot y + z \)
- Sometime (along this execution) an acknowledgement packet is sent
- If thread T is infinitely often enabled (along this execution) then T is eventually executed

By no means the last word:
- Last packet received along channel a (along this execution) had the shape (b,c,d) (past)
- For all executions (from this state) there is an execution along which a reply is eventually sent (branching)
- No matter what choice B made in the past, it would necessarily come to pass that \( \psi \) (historical necessity)
LTL

Syntax:
\[ \phi ::= P \mid \phi \mid \phi \land \phi \mid F\phi \mid G\phi \mid \phi \mathbin{\text{U}} \phi \mid O\phi \]

Intuitive semantics:

- \( P \): Propositional constant \( P \) holds now/at the current time instant
- \( F\phi \): At *some future* time instant \( \phi \) is true
- \( G\phi \): *For all future* time instants \( \phi \) is true
- \( \phi \mathbin{\text{U}} \psi \): \( \phi \) is true *until* \( \psi \) becomes true
- \( O\phi \): \( \phi \) is true at the *next* time instant
Pictorially

<table>
<thead>
<tr>
<th></th>
<th>F(\phi):</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>(\phi)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G(\phi):</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td></td>
</tr>
<tr>
<td>(\phi U \psi):</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\phi)</td>
<td>(\psi)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O(\phi):</td>
<td></td>
<td>(\phi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantics

Run $w$
Satisfaction relation $w^2 \phi$
Assume valuation $v$
$v(P)$: Set of states for which $P$ holds
$w^k$: $k$’th suffix of $w$

$w^2 P$ iff $w(0) \in v(P)$
$w^2 :\phi$ iff not $w^2 \phi$
$w^2 \phi \equiv \psi$ iff $w^2 \phi$ and $w^2 \psi$
$w^2 F\phi$ iff exists $k \geq 0$. $w^k \phi$
$w^2 G\phi$ iff for all $k \geq 0$. $w^k \phi$
$w^2 \phi U \psi$ iff exists $k \geq 0$. $w^k \psi$ and for all $i$: $0 \cdot i < k$. $w^i \phi$
$w^2 O\phi$ iff $w^1 \phi$

For transition system $T = (Q,R,Q_0)$ and all valuations $v$:
$T^2 \phi$ iff for all runs $w$ of $T$, $w^2 \phi$
Some LTL Formulas

- \( \phi \mathcal{C} \psi = (\lnot \phi \mathcal{A} \mathcal{E} \psi) \)
- \( \phi \mathcal{I} \psi = \lnot \phi \mathcal{C} \psi \)
- \( F\phi = \text{true} \cup \phi \)
- \( G\phi = \mathcal{F} \phi \)
- \( \phi \mathcal{V} \psi = [\psi \mathcal{C} (\psi \cup (\phi \mathcal{A} \psi))] \)
  - (sometimes called ”release”)
- \( FG\phi \)
  - \( \phi \) holds from some point forever = \( \phi \) holds *almost always*
- \( GF\phi \)
  - \( \phi \) holds infinitely often (i.o.)
- \( GF\phi \mathcal{I} GF\psi \)
  - if \( \phi \) holds infinitely often then so does \( \psi \)
  - Is this the same as \( G(F\phi \rightarrow F\psi) \)? As \( GF(\phi \rightarrow \psi) \)? As \( FG \lnot \phi \vee GF(\phi \land F\psi) \)?
Spring Example

Conditions: *extended, malfunction*

Sample paths:

- \( q_0 q_1 q_0 q_1 q_2 q_2 q_2 \ldots \)
- \( q_0 q_1 q_2 q_2 q_2 \ldots \)
- \( q_0 q_1 q_0 q_1 q_0 q_1 \ldots \)
Satisfaction by Single Path

For $r$:

- extended?
- $O_{extended}$?
- $OO_{extended}$?
- $F_{extended}$?
- $G_{extended}$?
- $FG_{extended}$?
- $FG_{malfunction}$?

$w = q_0 q_1 q_0 q_1 q_2 q_2 q_2 \ldots$
Satisfaction by Transition System

For T:

- extended?
- Oextended?
- OOextended?
- Fextended?
- Gextended?
- FGextended?
- FGmalfuction?
Example: Mutex

Assume there are 2 processes, \( P_l \) and \( P_r \).

State assertions:
- \( \text{tryCS}_i \): Process \( i \) is trying to enter critical section
  
  E.g. \( \text{tryCS}_i \): \( pc_i = l_4 \)

- \( \text{inCS}_i \): Process \( i \) is inside its critical section
  
  E.g. \( \text{inCS}_i \): \( pc_i = l_5 \land pc_i = l_6 \)

Mutual exclusion:
\[
G(\neg (\text{inCS}_i \land \text{inCS}_r))
\]

Responsiveness:
\[
G(\neg \text{tryCS}_i \land F \text{inCS}_i)
\]

Process keeps trying until access is granted:
\[
G(\neg \text{tryCS}_i \land (\neg (\text{tryCS}_i \lor \text{inCS}_i) \land G\neg \text{tryCS}_i))
\]
Example: Fairness

States: Pairs \((q, \alpha)\)

\(\alpha\) label of last transition taken, so

\[
\frac{q!^\alpha q'}{(q, \beta) !^\alpha (q', \alpha)}
\]

\(\Sigma\): Finite set of labels partitioned into subsets \(P\)

\(P\): "(finite) set of labels of some process"

State assertions:

- \(en_P\): Some transition labelled \(\alpha\) 2 \(P\) is enabled
  i.e. \((q, \beta) 2 v(en_\alpha)\) iff \(9 q'.q!^\alpha q'\)

- \(exec_P\): Label of last executed transition is in \(P\)
  i.e. \((q, \alpha) 2 v(exec_P)\) iff \(\alpha 2 P\)

Note: \(en_P \not\subset \cup \alpha 2 P\) and \(exec_P \not\subset \cup \alpha 2 P\exec\{\alpha\}\
Fairness Conditions

Weak transition fairness:
\[ \mathcal{A}_{\alpha_2 \Sigma} : \text{FG} \left( \text{en}_{\{\alpha\}} \mathcal{A} : \text{exec}_{\{\alpha\}} \right) \]
Or equivalently
\[ \mathcal{A}_{\alpha_2 \Sigma} ( \text{FGen}_{\{\alpha\}} \mid \text{GFexec}_{\{\alpha\}} ) \]

Strong transition fairness:
\[ \mathcal{A}_{\alpha_2 \Sigma} ( \text{GFen}_{\{\alpha\}} \mid \text{GFexec}_{\{\alpha\}} ) \]

Weak process fairness:
\[ \mathcal{A}_p : \text{FG} \left( \text{en}_p \mathcal{A} : \text{exec}_p \right) \]

Strong process fairness:
\[ \mathcal{A}_p ( \text{GFen}_p \mid \text{GFexec}_p ) \]

(Many other variants are possible)

Exercise: Figure out which implications hold between these four fairness conditions. Draw a picture
Branching Time Logic

Sets of paths?

Or computation tree?
Computation Tree Logic - CTL

Syntax:
\[ \phi ::= P | :\phi | \phi \Box \phi | AF\phi | AG\phi | A(\phi U \phi) | AX\phi \]

Formulas hold of states, not paths

A: Path quantifier, along all paths from this state

So:
- \( AF\phi \): Along all paths, at some future time instant \( \phi \) is true
- \( AG\phi \): Along all paths, for all future time instants \( \phi \) is true
- \( A(\phi U \psi) \): Along all paths, \( \phi \) is true until \( \psi \) becomes true
- \( AX\phi \): \( \phi \) is true for all next states

Note: CTL is closed under negation so also express dual modalities \( EF, EG, EU, EX \) (E is existential path quantifier). Check!
CTL, Semantics

Valuation \( v : P \rightarrow Q \) as before

\[ q^2 P \iff q \in v(P) \]
\[ q^2 :\phi \iff \neg q^2 \phi \]
\[ q^2 \phi \land q^2 \psi \iff q^2 \phi \land q^2 \psi \]
\[ q^2 AF\phi \iff \text{for all } w \text{ such that } w(0) = q \text{ exists } k \in 2N \text{ such that } w(k) \supseteq q^2 \phi \]
\[ q^2 AG\phi \iff \text{for all } w \text{ such that } w(0) = q, \text{ for all } k \in 2N, w(k) \supseteq q^2 \phi \]
\[ q^2 A(\phi U \psi) \iff \text{for all } w \text{ such that } w(0) = q, \text{ exists } k \in 2N \text{ such that } w(k) \supseteq q^2 \psi \]
\[ \text{and for all } i : 0 \leq i < k. w(i) \supseteq q^2 \phi \]
\[ q^2 AX\phi \iff \text{for all } w \text{ such that } w(0) = q, w(1) \supseteq q^2 \phi \]
\[ \quad \text{(iff for all } q' \text{ such that } q \models q', q'^2 \phi) \]

For transition system \( T = (Q,R,Q_0) \):
\[ T^2 \phi \iff \text{for all } q_0 \in Q_0, q_0^2 \phi \]
CTL – LTL: Brief Comparison

LTL in branching time framework:
- $\phi \mapsto A\phi$ (\(\phi\) to hold for all paths)

CTL * LTL: EF\(\phi\) not expressible in LTL

LTL * CTL: FGP not expressible in CTL

CTL*: Extension of CTL with free alternation A, F, G, U, X

Advantages and disadvantages:
- LTL often "more natural"
- Satisfiability: LTL: PSPACE complete, CTL: DEXPTIME complete
- Model checking: LTL: PSPACE complete, CTL: In P
**Adding Past**

Add to LTL pasttime versions of the LTL future time modalities

Previously, some time in the past, always in the past, since

**Theorem** (Gabbay’s separation theorem): Every formula in LTL + past is equivalent to a boolean combination of "pure pasttime" or "pure future time" formulas

Note: This applies regardless of whether time starts at 0 or at $-\infty$

**Theorem** (Elimination of past): Pasttime modalities do not add expressive power to LTL

But:

**Theorem** (Succinctness, LMS’02): LTL + past is exponentially more succinct than LTL
Expressive Completeness

LTL is easily embedded into FOL + linear order

FOL + linear order: First-order logic with 0 and <, unary predicate symbols, and interpreted over \( \omega \)

**Theorem** (Kamp’68, GPSS’80, Expressive completeness)
If L is definable in FOL + linear order then L is definable in LTL
So Are We Done?

What about "every even state"

Theorem: A"every even state"\(P\) is not expressible in LTL, CTL, CTL*

One solution:
- LTL formulas determine infinite words
- So: skip temporal logic (… temporarily …) and use automata on infinite words instead
Automata Over Finite Words

Finite state automaton $A = (Q, \Sigma, \Delta, I, F)$:
- $Q$: Finite set of states
- $\Sigma$: Finite alphabet
- $\Delta \subseteq Q \times \Sigma \times Q$: Transition relation
  Write $q \xrightarrow{a} q'$ for $\Delta(q, a, q')$ as before
- $I \subseteq Q$: Start states
- $F \subseteq Q$: Accepting states

Word $a_1a_2...a_n$ is accepted, if there is sequence

$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} ... \xrightarrow{a_n} q_n$

such that $q_0 \in I$ and $q_n \in F$
Automata Over Infinite Words

Letters $\alpha_2 \Sigma$ can represent events, conditions, states

Infinite word $w \in \Sigma^\omega$:
- Function $w: \omega \rightarrow \Sigma$
- Equivalently: Infinite sequence $w = a_0a_1a_2 ... a_n ...$
- Terminology: $\omega$-words
- $\omega$-words are traces / paths / runs

*Buchi automaton*: Finite state automaton, but on infinite words

$\omega$-word $w$ is accepted if accepting state visited infinitely often

$\omega$-language $L \subseteq \Sigma^\omega$ is *Buchi definable* if $L$ is the set of $\omega$-words accepted by some B. A.
Example

Which infinite words are accepted?

- ababab ... \((= (ab)^\omega)\) ?
- aaaaaa... \((= a^\omega)\) ?
- bbbbbbb... \((= b^\omega)\) ?
- aaabbbbb... \((= aaab^\omega)\) ?
- ababbabbbabbbba... ?
Nondeterminism

• What is the language accepted by this automaton?
• What is the corresponding LTL property if $b = \text{inCS}$ and $a = : b$?
Another Example

Letters represent propositions

Example: GFinCS, a=inCS, b=: inCS
Yet More Examples

• \( a = \text{inCS}_1 \land \text{inCS}_2 \)
• \( b = : a \)
• \( c = \text{true} \)
• Property: \( G: a \)

• Property: \( G(d ! Fe) \)
• Idea:
  – \( q_0; \) Have seen : \( d \land e \)
  – \( q_1; \) Saw \( d \), now wait for \( e \)
Even More...

Property: $G(a \rightarrow (b \lor c))$

Idea:
- $q_0$: Body of $G$ immediately ok
- $q_1$: Awaiting $c$

Property: $\neg G(a \rightarrow (b \lor c)) = F(a \rightarrow \neg (b \lor c))$

Idea:
- $\neg (b \lor c)$: $b$ becomes false some time without $c$ having become true first
- $q_0$: Waiting ...
- $q_1$: Have seen $a$ with $b$ and $\neg c$
- $q_2$: Committing ...
Generally

**Theorem:** If $L$ is LTL definable then $L$ is the set of words accepted by some B.A.

Why? The set of B.A. recognizable languages is closed under all LTL connectives

Hard case is complementation [Safra’88]

BTW then we can do LTL model checking:

- Represent model as B.A. $A_1$
- Represent LTL spec as $A_2$
- Emptiness of $L(A) = \{w \mid A \text{ accepts } w\}$ is polynomially decidable
- $L(A_1) \subseteq L(A_2)$ iff $L(A_1) \cap \neg L(A_2)$ is empty
- Example: The SPIN model checker
Aside: Deterministic Buchi Automata

Consider $\phi = FGa$ where $\Sigma = \{a,b\}$

Suppose $A$ recognizes $\phi$
A deterministic
A reaches accepting state on some input $a^{n_1}$
And on $a^{n_1}ba^{n_2}$
And on $a^{n_1}ba^{n_2}ba^{n_3}$
And on $a^{n_1}ba^{n_2}ba^{n_3}b \ldots b \ldots b \ldots$

So: Nondeterministic Buchi automata strictly more expressive than deterministic ones
And: Deterministic B. A. not closed under complement
Temporal Equations

Idea: Extend LTL with solutions of equations

- \( F\phi = \phi \lor O F\phi \)
- \( G\phi = \phi \land OG\phi \)
- \( \phi U \psi = \psi \lor (\phi \land O(\phi U \psi)) \)
- \( \text{Even} \phi = \phi \land O \text{Even} \phi \)

Complication: Solutions are not unique

**Exercise:** How many solutions (as sets \( L \) of traces/words \( w \)) can you find to the above four equations?
The Linear Time $\mu$-calculus, $L_\mu$

Formula $\phi(X)$ in free formula variable $X$ determines function $\phi : L \mapsto \phi(L)$

If $\phi(X)$ is monotone in $X$ then $|| \phi ||$ is monotone as function on $\langle \{L \mid L \subseteq \Sigma^\omega \}, \subseteq \rangle$.

**Theorem** (Tarski’s fixed point theorem): A monotone function on a complete lattice has a complete lattice of fixed points.

So, for each monotone $\phi(X)$ can find a largest and a smallest solution of equation $X = \phi(X)$.
Notation:
• $\mu X. \phi(X)$: Least solution of $X = \phi(X)$
• $\nu X. \phi(X)$: Greatest solution of $X = \phi(X)$

Note:
• $F\phi = \mu X. \phi \vee OX$
• $G\phi = \nu X. \phi \land OX$
• $\phi U \psi = \mu X. \psi \vee (\phi \land OX)$
• Even $\phi = \nu X. \phi \land OOX$

**Exercise:** Exchange $\mu$ and $\nu$ in the 4 examples above. What property is defined?

**Hint:** Which is the largest, resp. smallest $L$ that solves the equation?
Expressiveness of $L_\mu$

**Theorem:** An $\omega$-language is definable in $L_\mu$ iff it is recognized by a B.A.

**Direct proof:**

$\Leftarrow$: Represent B.A. in $L_\mu$ (easy)

$\Rightarrow$: Show that B.A. definable languages are closed under all $L_\mu$ connectives. Hard part is $\mu$, cf. (Dam, 92)

But many alternative characterizations exist
Alternative Characterizations

S1S: Monadic second order logic of successor
\[ 9 \, X(0 \, X \, \forall \, 8y8z(\text{succ}(y,z) \, \! \, (y2X \, \$ : \, z2X)) \]
\[ \forall \, 8y(y2X \, \! \, a(y))) \]
(all even symbols are a’s)

QPLTL: LTL with propositional quantification
\[ 9 \, X((X \, \forall \, G(X \, \$ \, O:X) \, \forall \, G(x \, \! \, a)) \]

\(\omega\)-regular expressions
\[ a(((a \, [ \, b)a)^\omega) \]

**Theorem** (Buchi et al): An \(\omega\)-language is recognized by a B.A. iff it is definable in one of \(L_\mu\), S1S, QPLTL, or as an \(\omega\)-regular expression
What About Branching Time?

More difficult. Starting point are binary trees:

**Theorem** (Rabin): S2S (the monadic second-order theory of two successors) is decidable

For more general structures use e.g.
- Alternating tree automata
- Modal \( \mu \)-calculus
- Parity games

Much activity in the past 10 years

But this is outside the scope of this course