Introduction to Temporal Logic

Mads Dam
Theoretical Computer Science
KTH, 2009

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?

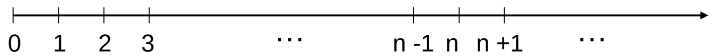
- Continuous real-valued functions?
- Discrete program traces?
- Execution trees?
- Automata?
- Markov chains?
- Java code?
- Distributed processes?
- Real time? Or implicit time?
- Histories or future?
- Finite or infinite?
- Linear or branching? Tree shaped? Graph shaped?

Default Choice – Traces/Paths/Runs

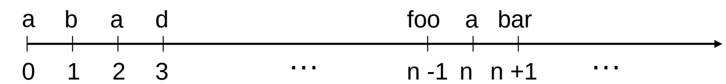
Time is discrete

Starts at 0

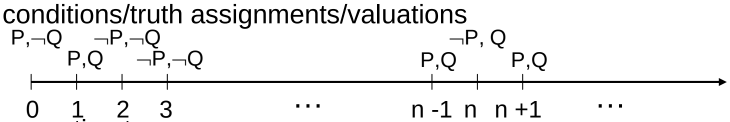
Goes on forever



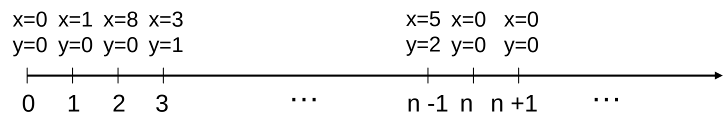
Time points decorated by events



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 $\sigma \in x \mapsto_{fin} v \in Val$

Configurations c ::= σ | $<Cmd, \sigma>$

Example II

Transitions:

- σ -> σ (… just to get looping …)
- $<skip,\sigma>$ -> σ
- $<x:=e,\sigma>$ -> $\sigma[x \mapsto ||e||\sigma]$
- $<Cmd_1;Cmd_2,\sigma>$ -> $<Cmd_1';Cmd_2,\sigma'>$
  if $<Cmd_1,\sigma>$ -> $<Cmd_1',\sigma'>$
- $<Cmd_1;Cmd_2,\sigma>$ -> $<Cmd_2,\sigma'>$
  if $<Cmd_1,\sigma>$ -> σ'
- (… remaining rules in class …)

Conditions: Boolean/FO expressions in dom(σ_ι)

Traces: c_0 -> c_1 -> c_2 -> … -> c_{n-1} -> c_n -> …

Linear Time Temporal Logic, LTL

Logic of temporal relations between events in a trace:

- Invariably (along this execution) $x \leq y + z$
- 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 ψ (*historical necessity*)

LTL

Syntax:

$\phi ::= P \mid \neg\phi \mid \phi\wedge\phi \mid F\phi \mid G\phi \mid \phi\ U\ \phi \mid O\phi$

Intuitive semantics:

- P: Propositional constant P holds now/at the current time instant
- $F\phi$: At *some future* time instant ϕ is true
- $G\phi$: *For all future* time instants ϕ is true
- $\phi\ U\ \psi$: ϕ is true *until* ψ becomes true
- $O\phi$: ϕ is true at the *next* time instant

Pictorially

$F\phi$:	…	…	…	…	ϕ	…	...
$G\phi$:	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
ϕ U ψ:	ϕ	ϕ	ϕ	ϕ	ψ	…	...
$O\phi$:	...	ϕ	...	...	...	…	...

Semantics

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

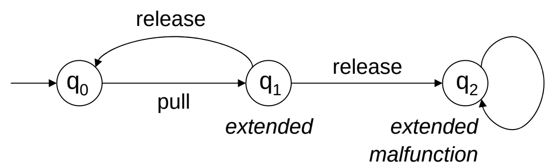
$w \models P$ iff $w(0) \in v(P)$
$w \models \neg\phi$ iff not $w \models \phi$
$w \models \phi \wedge \psi$ iff $w \models \phi$ and $w \models \psi$
$w \models F\phi$ iff exists $k \geq 0$. $w^k \models \phi$
$w \models G\phi$ iff for all $k \geq 0$. $w^k \models \phi$
$w \models \phi\ U\ \psi$ iff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

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

Some LTL Formulas

- $\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi)$
- $\phi \rightarrow \psi = \neg\phi \vee \psi$
- $F\phi$ = true U ϕ
- $G\phi = \neg F \neg\phi$
- $\phi \mathrm{V} \psi = []\psi \vee (\psi \mathrm{U} (\phi\wedge\psi))$
  - (sometimes called ”release”)
- $FG\phi$
  - ϕ holds from some point forever = ϕ holds *almost always*
- $GF\phi$
  - ϕ holds infinitely often (i.o.)
- $GF\phi \rightarrow GF\psi$
  - if ϕ holds infinitely often then so does ψ
  - Is this the same as $G(F\phi \rightarrow F\psi)$? As $GF(\phi \rightarrow \psi)$? As $FG\neg \phi \vee GF(\phi\wedge F\psi)$?

Spring Example

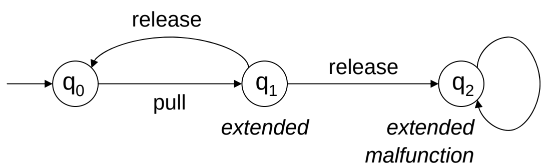


Conditions: *extended, malfunction*

Sample paths:

- $q_0\ q_1\ q_0\ q_1\ q_2\ q_2\ q_2$ ...
- $q_0\ q_1\ q_2\ q_2\ q_2$ ...
- $q_0\ q_1\ q_0\ q_1\ q_0\ q_1$ ...

Satisfaction by Single Path



$w = q_0q_1q_0q_1q_2q_2q_2 \ldots$

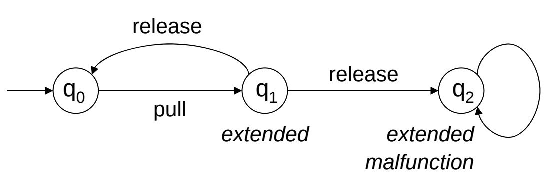
For r:

extended?
O*extended*?
OO*extended*?
F*extended*?
G*extended*?
FG*extended*?
FG*malfunction*?

GF*extended*?
extended U *malfunction*?
(¬*extended*) U *extended*?
(F*extended*) U *malfunction*?
(F¬*extended*) U *malfunction*?
G(¬*extended* → O*extended*)

Satisfaction by Transition System

T:



For T:

extended?
O*extended*?
OO*extended*?
F*extended*?
G*extended*?
FG*extended*?
FG*malfunction*?

GF*extended*?
extended U *malfunction*?
(¬*extended*) U *extended*?
(F*extended*) U *malfunction*?
(F¬*extended*) U *malfunction*?
G(¬*extended* → O*extended*)

Example: Mutex

Assume there are 2 processes, P_l and P_r

State assertions:

- $tryCS_i$: Process i is trying to enter critical section
  E.g. $tryCS_l$: $pc_l = l_4$
- $inCS_i$: Process i is inside its critical section
  E.g. $inCS_l$: $pc_l = l_5 \vee pc_l = l_6$

Mutual exclusion:

$$G(\neg(inCS_l \wedge inCS_r))$$

Responsiveness:

$$G(tryCS_i \rightarrow F\ inCS_i)$$

Process keeps trying until access is granted:

$$G(tryCS_i \rightarrow ((tryCS_i\ U\ inCS_i) \vee G tryCS_i))$$

Example: Fairness

States: Pairs (q,α)

α label of last transition taken, so

$$\frac{q \rightarrow^{\alpha} q'}{(q,\beta) \rightarrow^{\alpha} (q',\alpha)}$$

Σ: Finite set of labels partitioned into subsets P

P: ”(finite) set of labels of some process”

State assertions:

- en_P: Some transition labelled $\alpha \in P$ is enabled
  i.e. $(q,\beta) \in v(en_\alpha)$ iff $\exists\, q'.q \rightarrow^{\alpha} q'$
- $exec_P$: Label of last executed transition is in P
  i.e. $(q,\alpha) \in v(exec_P)$ iff $\alpha \in P$

Note: $en_P \leftrightarrow \vee_{\alpha \in P} en_{\{\alpha\}}$ and $exec_P \leftrightarrow \vee_{\alpha \in P} exec_{\{\alpha\}}$

Fairness Conditions

Weak transition fairness:

$$\wedge_{\alpha\in\Sigma}\neg FG(en_{\{\alpha\}} \wedge \neg\, exec_{\{\alpha\}})$$

Or equivalently

$$\wedge_{\alpha\in\Sigma}(FGen_{\{\alpha\}} \rightarrow GFexec_{\{\alpha\}})$$

Strong transition fairness:

$$\wedge_{\alpha\in\Sigma}(GFen_{\{\alpha\}} \rightarrow GFexec_{\{\alpha\}})$$

Weak process fairness:

$$\wedge_P\neg FG(en_P \wedge \neg\, exec_P)$$

Strong process fairness:

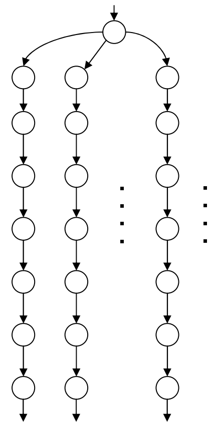
$$\wedge_P (GFen_P \rightarrow GFexec_P)$$

(Many other variants are possible)

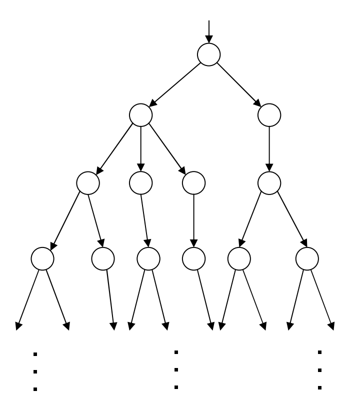
Exercise: Figure out which implications hold between these four fairness conditions. Draw a picture

Branching Time Logic

Sets of paths?



Or computation tree?



Computation Tree Logic - CTL

Syntax:

$\phi ::= P \mid \neg\phi \mid \phi\wedge\phi \mid AF\phi \mid AG\phi \mid A(\phi\ U\ \phi) \mid AX\phi$

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 ϕ is true
- $AG\phi$: Along all paths, for all future time instants ϕ is true
- $A(\phi\ U\ \psi)$: Along all paths, ϕ is true until ψ becomes true
- $AX\phi$: ϕ 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: $P \mapsto Q' \subseteq Q$ as before

$q \models P$ iff $q \in v(P)$

$q \models \neg\phi$ iff not $q \models \phi$

$q \models \phi \wedge \psi$ iff $q \models \phi$ and $q \models \psi$

$q \models AF\phi$ iff for all w such that $w(0)=q$ exists $k\in\mathbb{N}$ such that $w(k) \models \phi$

$q \models AG\phi$ iff for all w such that $w(0)=q$, for all $k\in\mathbb{N}$, $w(k) \models \phi$

$q \models A(\phi\ U\ \psi)$ iff for all w such that $w(0)=q$, exists $k\in\mathbb{N}$ such that $w(k) \models \psi$ and for all i: $0\leq i < k$. $w(i) \models \phi$

$q \models AX\phi$ iff for all w such that $w(0) = q$, $w(1) \models \phi$

(iff for all q' such that $q \rightarrow q'$, $q' \models \phi$)

For transition system $T = (Q,R,Q_0)$:

$T \models \phi$ iff for all $q_0 \in Q_0$, $q_0 \models \phi$

CTL – LTL: Brief Comparison

LTL in branching time framework:

- $\phi \mapsto A\phi$ (ϕ to hold for all paths)

CTL $\not\subseteq$ LTL: EFϕ not expressible in LTL

LTL $\not\subseteq$ 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 -∞

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

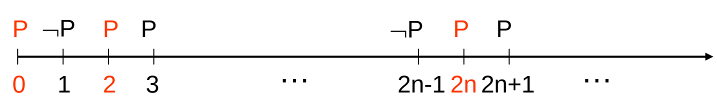
FOL + linear order: First-order logic with 0 and <, unary predicate symbols, and interpreted over ω

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*

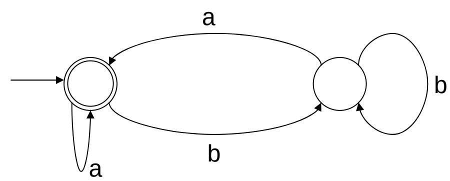
One solution:

- 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,Σ,Δ,I,F):

- Q: Finite set of states
- Σ: Finite alphabet
- $\Delta \subseteq Q\times \Sigma \times Q$: Transition relation
  Write $q\to^{a} q'$ 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

$$q_0 \to^{a_1} q_1 \to^{a_2} ... \to^{a_n} q_n$$

such that $q_0\in I$ and $q_n\in F$

Automata Over Infinite Words

Letters $a\in\Sigma$ can represent events, conditions, states

Infinite word $w \in \Sigma^\omega$:

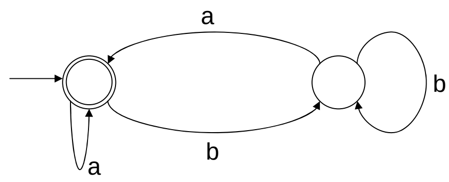
- Function w: $\omega \to \Sigma$
- Equivalently: Infinite sequence $w = a_0a_1a_2 ... a_n ...$
- Terminology: ω-words
- ω-words are traces / paths / runs

Buchi automaton: Finite state automaton, but on infinite words

ω-word w is *accepted* if accepting state visited infinitely often (!)

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

Example

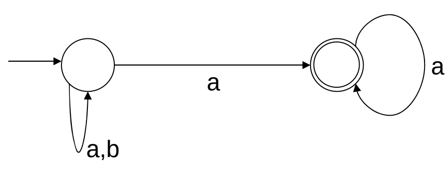


Which infinite words are accepted?

- ababab ... (= $(ab)^\omega$) ?
- aaaaaa... (= a^ω) ?
- bbbbbb... (= b^ω) ?
- aaabbbbb... (= $aaab^\omega$) ?
- ababbabbbabbbba... ?

Nondeterminism

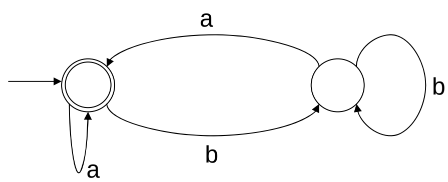
- What is the language accepted by this automaton?
- What is the corresponding LTL property if b = inCS and a = ¬ b?



Another Example

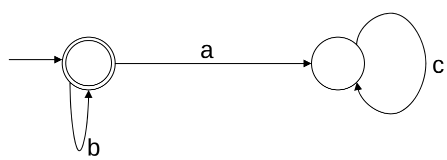
Letters represent propositions

Example: GFinCS, a=inCS, b=¬ inCS

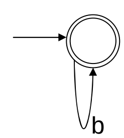


Yet More Examples

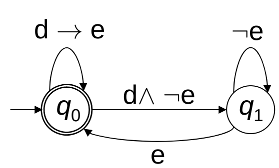
- a = $inCS_1 \wedge inCS_2$
- b = ¬ a
- c = true
- Property: G¬ a



Or just:



- Property: G(d → Fe)
- Idea:
  - q_0; Have seen ¬ d ∨ e
  - q_1: Saw d, now wait for e

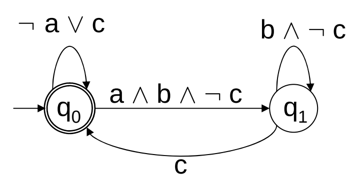


Even More...

Property: G(a → (bUc))

Idea:

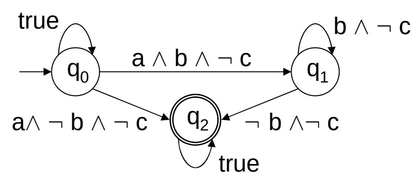
- q_0: Body of G immediately ok
- q_1: Awaiting c



Property: ¬G(a → (bUc)) = F(a ∧ ¬(bUc))

Idea:

- ¬(bUc): b becomes false some time without c having become true first
- q_0: Waiting ...
- q_1: Have seen a with b and ¬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

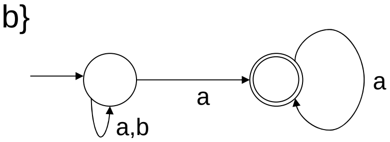
Hard case is complementation [Safra'88]

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 | A 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 ϕ = FGa where $\Sigma = \{a,b\}$



Suppose A recognizes ϕ

A deterministic

A reaches accepting state on some input a^{n1}

And on $a^{n1}ba^{n2}$

And on $a^{n1}ba^{n2}ba^{n3}$

And on $a^{n1}ba^{n2}ba^{n3}b$ ... b ... b ...

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

- $\underline{F\phi} = \phi \lor O\underline{F\phi}$
- $\underline{G\phi} = \phi \land O\underline{G\phi}$
- $\underline{\phi\ U\ \psi} = \psi \lor (\phi \land O(\underline{\phi\ U\ \psi}))$
- $\underline{Even\ \phi} = \phi \land OO\underline{Even\ \phi}$

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 μ-calculus, L_μ

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 $(\{L \mid L \subseteq \Sigma^\omega\},\subseteq)$

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 $X = \phi(X)$

L_μ

Notation:

- $\mu X.\phi(X)$: Least solution of $X = \phi(X)$
- $\nu X.\phi(X)$: Greatest solution of $X = \phi(X)$

Note:

- $F\phi = \mu X. \phi \vee OX$
- $G\phi = \nu X.\phi \wedge OX$
- $\phi \, U \, \psi = \mu X. \psi \vee (\phi \wedge OX)$
- Even $\phi = \nu X.\phi \wedge OOX$

Exercise: Exchange μ and ν in the 4 examples above. What property is defined?

Hint: Which is the largest, resp. smallest L that solves the equation?

Expressiveness of L_μ

Theorem: An ω-language is definable in L_μ iff it is recognized by a B.A.

Direct proof:

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

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

But many alternative characterizations exist

Alternative Characterizations

S1S: Monadic second order logic of successor

$$\exists X(0 \in X \wedge \forall y \forall z(succ(y,z) \rightarrow (y \in X \leftrightarrow \neg z \in X)) \wedge \forall y(y \in X \rightarrow a(y)))$$

(all even symbols are a's)

QPLTL: LTL with propositional quantification

$$\exists X((X \wedge G(X \leftrightarrow O \neg X) \wedge G(x \rightarrow a))$$

ω-regular expressions

$$a((a \cup b)a)^\omega$$

Theorem (Buchi et al): An ω-language is recognized by a B.A. iff it is definable in one of L_μ, S1S, QPLTL, or as an ω-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