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?

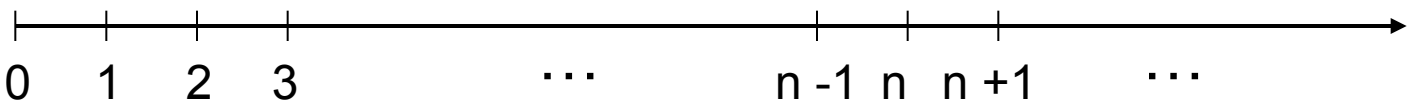
- Continuous real-valued functions?
- Discrete program traces?
- Execution trees?
- Automata?
- Markov chains?
- Java code?
- Distributed processes?
- Real time? Or implicit time?
- Histories or future?
- Finite or infinite?
- Linear or branching? Tree shaped? Graph shaped?

Default Choice – Traces/Paths/Runs

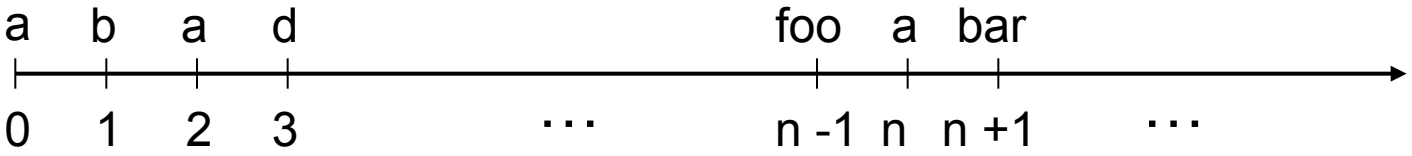
Time is discrete

Starts at 0

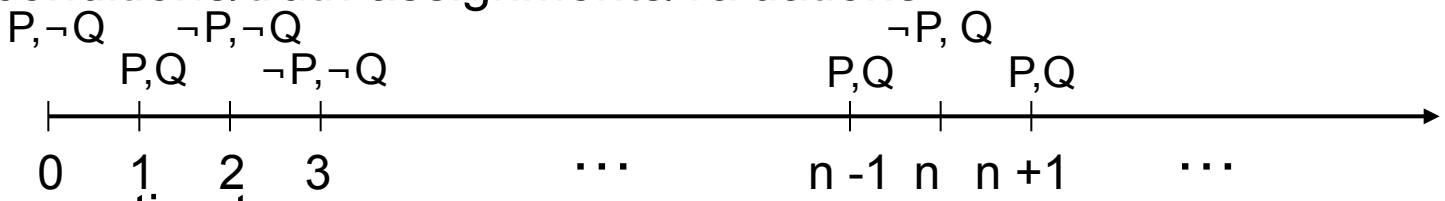
Goes on forever



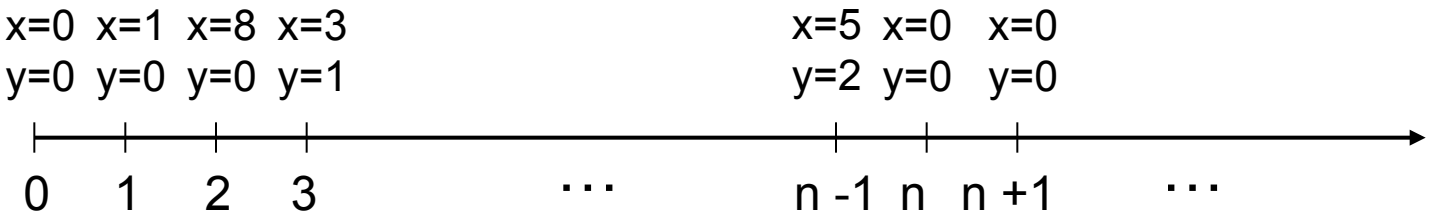
Time points decorated by events



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

Configurations c ::= σ | <Cmd, σ>

Example II

Transitions:

- σ -> σ (… just to get looping …)
- <skip,σ> -> σ
- <x:=e,σ> -> $\sigma[x \mapsto ||e||\sigma]$
- <Cmd_1;Cmd_2,σ> -> <Cmd_1';Cmd_2,σ'>
  if <Cmd_1,σ> -> <Cmd_1',σ'>
- <Cmd_1;Cmd_2,σ> -> <Cmd_2,σ'>
  if <Cmd_1,σ> -> 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) $x \cdot y + z$
- 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 ψ (*historical necessity*)

LTL

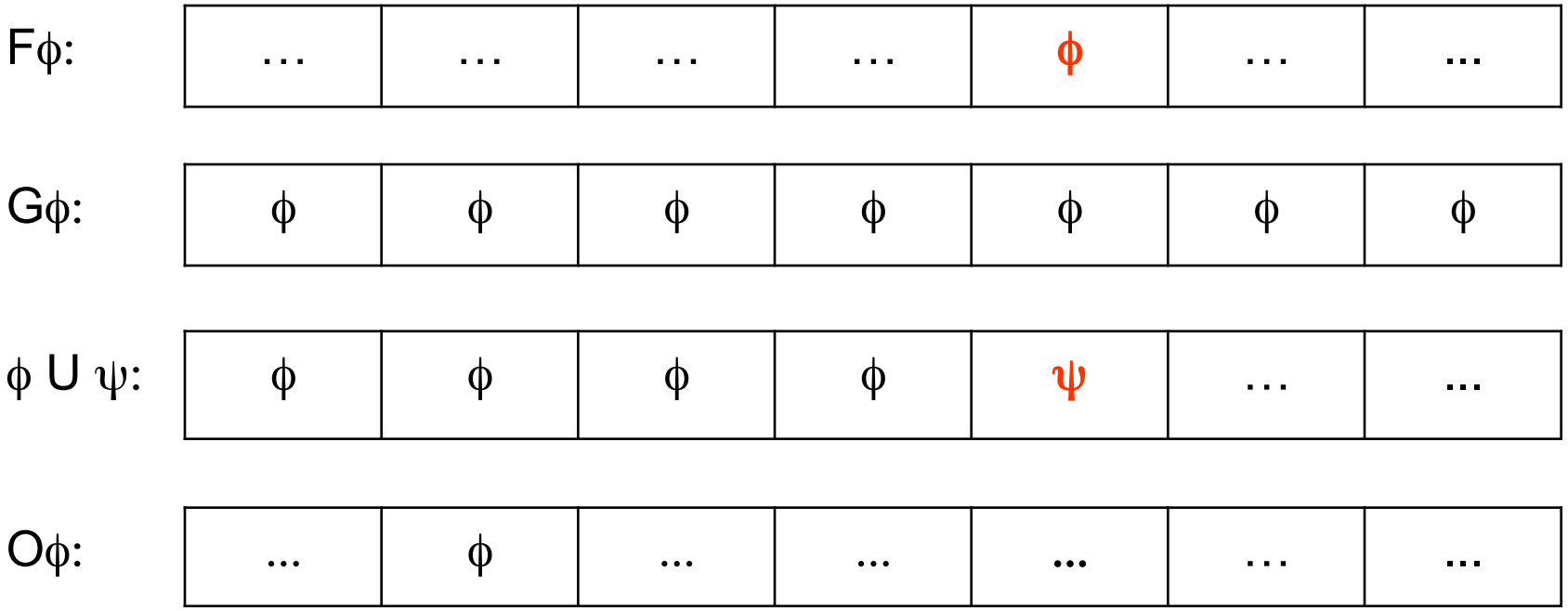
Syntax:

$\phi ::= P \mid :\phi \mid \phi Æ \phi \mid F\phi \mid G\phi \mid \phi \; U \; \phi \mid O\phi$

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



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 $w^1 \models \phi$

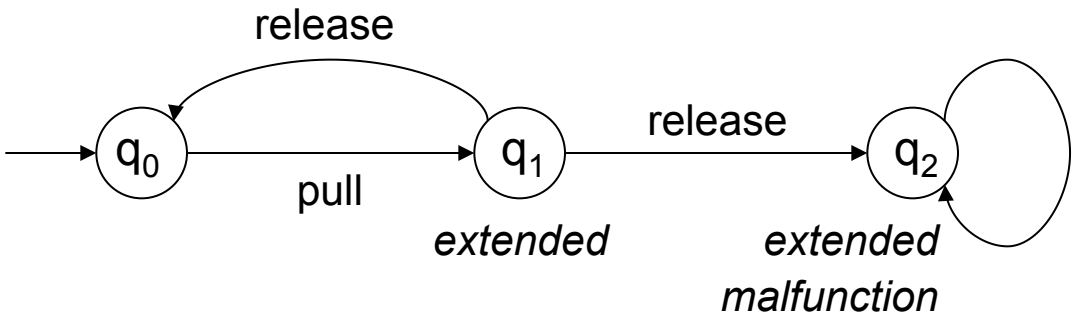
For transition system $T = (Q,R,Q_0)$ and all valuations v:

$T \models \phi$ iff for all runs w of T, $w \models \phi$

Some LTL Formulas

- ϕ Ç ψ = :(:ϕ Æ :ψ)
- ϕ ! ψ = :ϕ Ç ψ
- Fϕ = true U ϕ
- Gϕ = :F:ϕ
- ϕ V ψ = []ψ Ç (ψ U (ϕÆψ))
  - (sometimes called ”release”)
- FGϕ
  - ϕ holds from some point forever = ϕ holds *almost always*
- GFϕ
  - ϕ holds infinitely often (i.o.)
- GFϕ ! GFψ
  - if ϕ holds infinitely often then so does ψ
  - Is this the same as G(Fϕ → Fψ)? As GF(ϕ → ψ)? As FG¬ ϕ ∨ GF(ϕ∧ Fψ)?

Spring Example

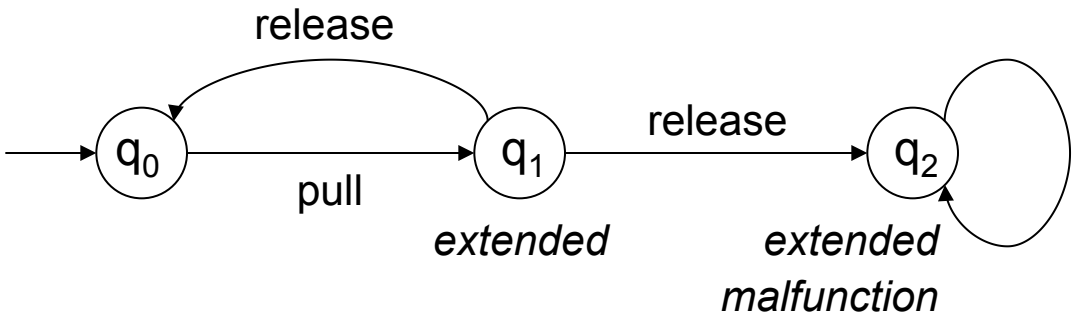


Conditions: *extended, malfunction*

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*?

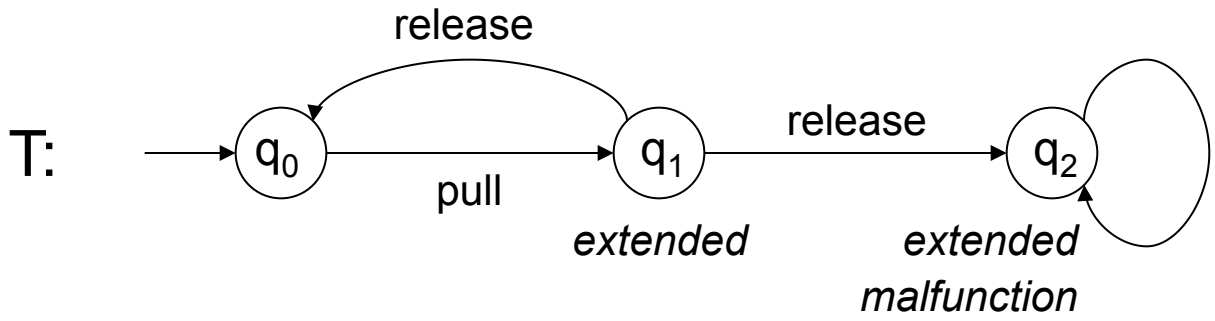
(:*extended*) U *extended*?

(F*extended*) U *malfunction*?

(F:*extended*) U *malfunction*?

G(:*extended* ! O*extended*)

Satisfaction by Transition System



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. $inCS_l$: $pc_l = l_5$ Ç $pc_l = l_6$

Mutual exclusion:

$$G(:(inCS_l \text{ Æ } inCS_r))$$

Responsiveness:

$$G(tryCS_i \text{ ! } F\ inCS_i)$$

Process keeps trying until access is granted:

$$G(tryCS_i \text{ ! } ((tryCS_i \text{ U } inCS_i) \text{ Ç } GtryCS_i))$$

Example: Fairness

States: Pairs (q,α)

α label of last transition taken, so

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

Σ: Finite set of labels partitioned into subsets P

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

State assertions:

- en_P: Some transition labelled α 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 $ Ç$_{\alpha 2\ P} en_{\{\alpha\}}$ and $exec_P$ $ Ç$_{\alpha 2\ P} exec_{\{\alpha\}}$

Fairness Conditions

Weak transition fairness:

$$Æ_{\alpha 2\Sigma}:FG(en_{\{\alpha\}} Æ : exec_{\{\alpha\}})$$

Or equivalently

$$Æ_{\alpha 2\Sigma}(FGen_{\{\alpha\}} ! GFexec_{\{\alpha\}})$$

Strong transition fairness:

$$Æ_{\alpha 2\Sigma}(GFen_{\{\alpha\}} ! GFexec_{\{\alpha\}})$$

Weak process fairness:

$$Æ_{P}:FG(en_{P} Æ : exec_{P})$$

Strong process fairness:

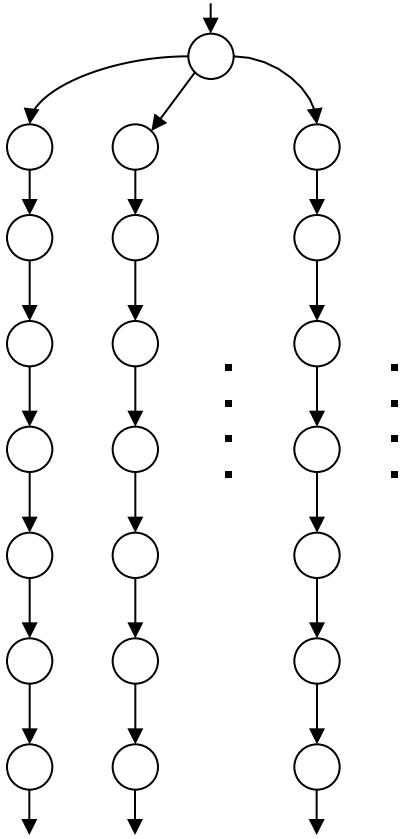
$$Æ_{P} (GFen_{P} ! GFexec_{P})$$

(Many other variants are possible)

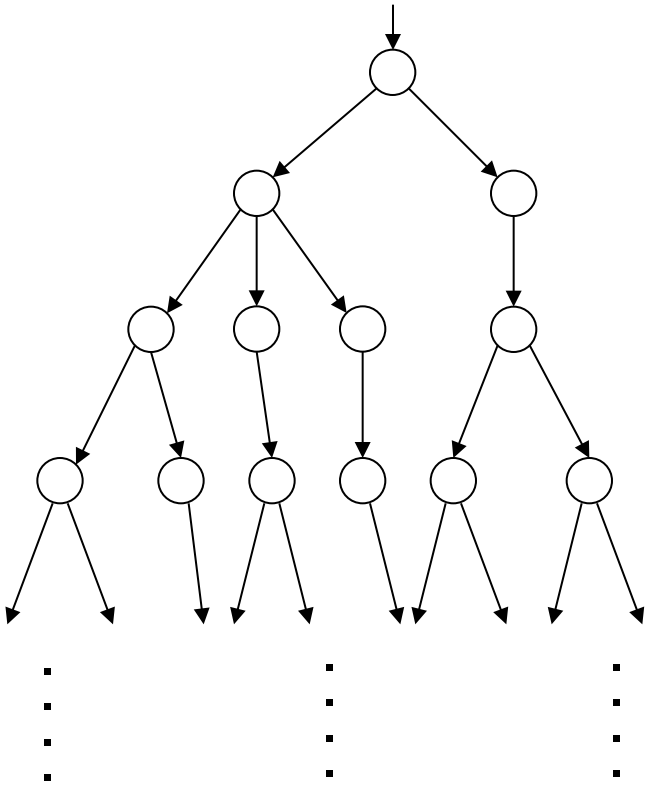
Exercise: Figure out which implications hold between these four fairness conditions. Draw a picture

Branching Time Logic

Sets of paths?



Or computation tree?



Computation Tree Logic - CTL

Syntax:

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

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' μ Q as before

q ² P iff q 2 v(P)

q ² :ϕ iff not q ² ϕ

q ² ϕ Æ ψ iff q ² ϕ and q ² ψ

q ² AFϕ iff for all w such that w(0)=q exists k2N such that w(k) ² ϕ

q ² AGϕ iff for all w such that w(0)=q, for all k2N, w(k) ² ϕ

q ² A(ϕ U ψ) iff for all w such that w(0)=q, exists k2N such that w(k) ² ψ and for all i: 0· i < k. w(i) ² ϕ

q ² AXϕ iff for all w such that w(0) = q, w(1) ² ϕ

(iff for all q' such that q ! q', q'² ϕ)

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

T ² ϕ iff for all $q_0$2 Q_0, q_0 ² ϕ

CTL – LTL: Brief Comparison

LTL in branching time framework:

- $\phi \mapsto A\phi$ (ϕ 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

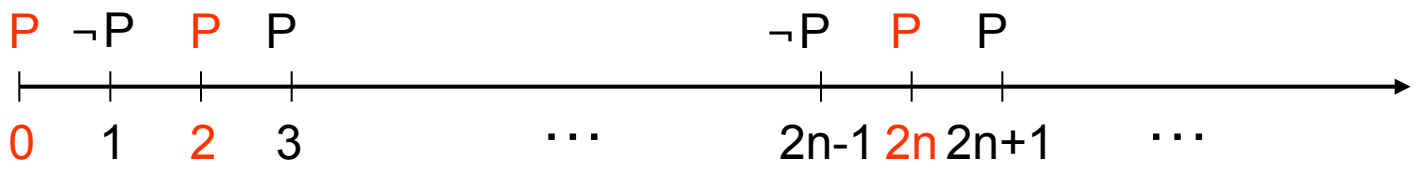
FOL + linear order: First-order logic with 0 and <, unary predicate symbols, and interpreted over ω

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*

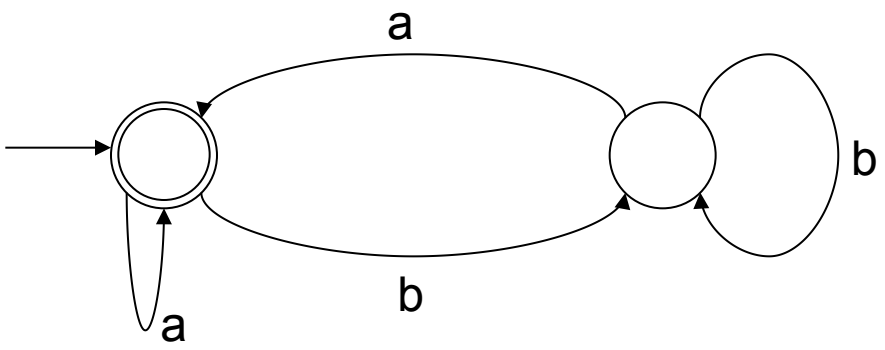
One solution:

- 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
- Δ μ Q£ Σ £ Q: Transition relation
  Write q![a] q' for Δ(q,a,q') as before
- I μ Q: Start states
- F μ Q: Accepting states



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

$$q_0 \,!^{a_1}\, q_1 \,!^{a_2}\, \ldots \,!^{a_n}\, q_n$$

such that $q_0 2$ I and $q_n 2$ F

Automata Over Infinite Words

Letters a2Σ can represent events, conditions, states

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

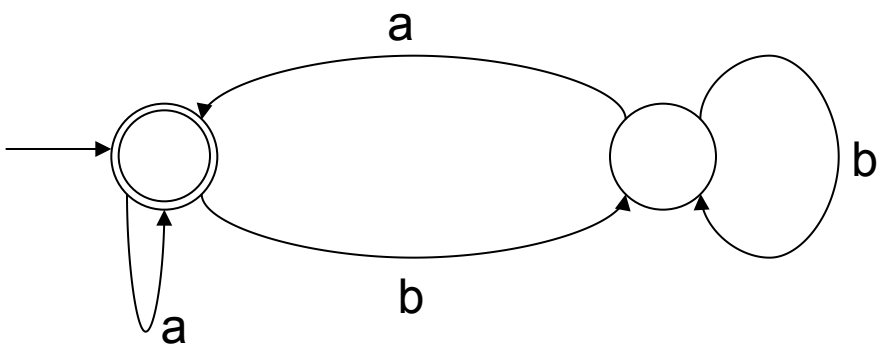
- Function w: ω ! Σ
- Equivalently: Infinite sequence w = $a_0 a_1 a_2 \ldots a_n \ldots$
- Terminology: ω-words
- ω-words are traces / paths / runs

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

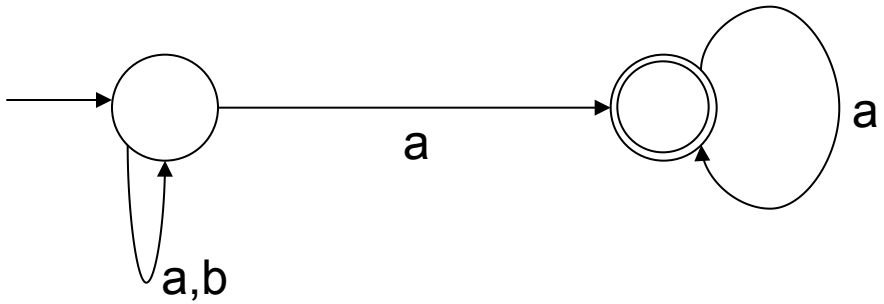


Which infinite words are accepted?

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

Nondeterminism

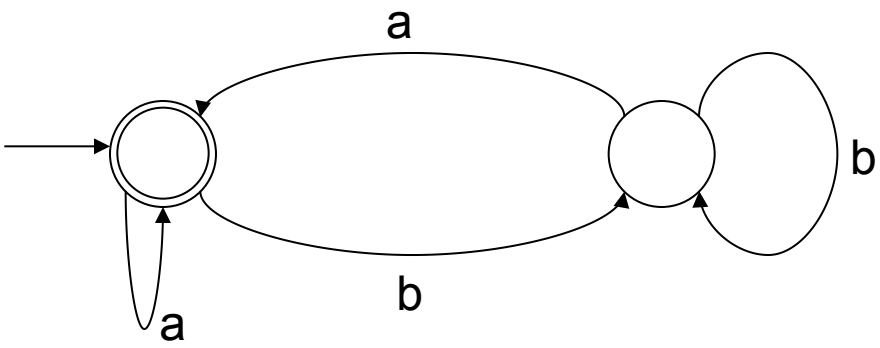
- What is the language accepted by this automaton?
- What is the corresponding LTL property if b = inCS and a = : b?



Another Example

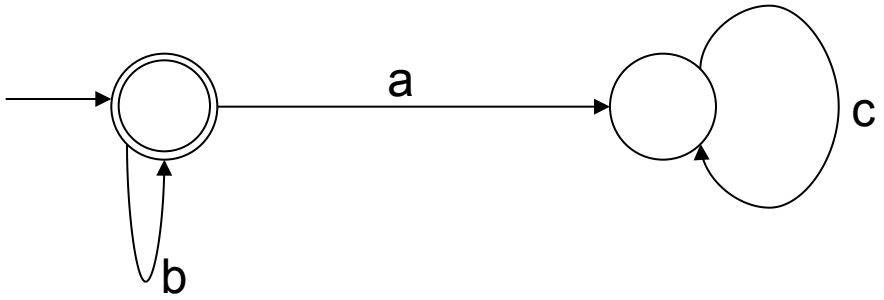
Letters represent propositions

Example: GFinCS, a=inCS, b=: inCS

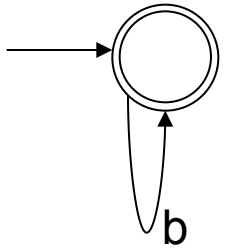


Yet More Examples

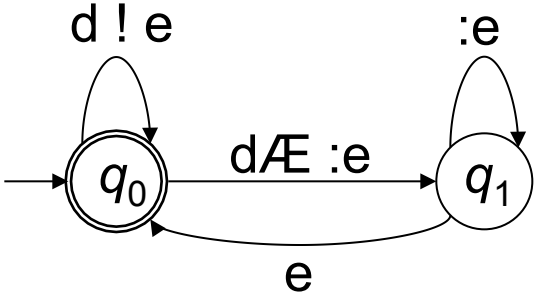
- a = $inCS_1$ Æ $inCS_2$
- b = : a
- c = true
- Property: G: a



Or just:



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

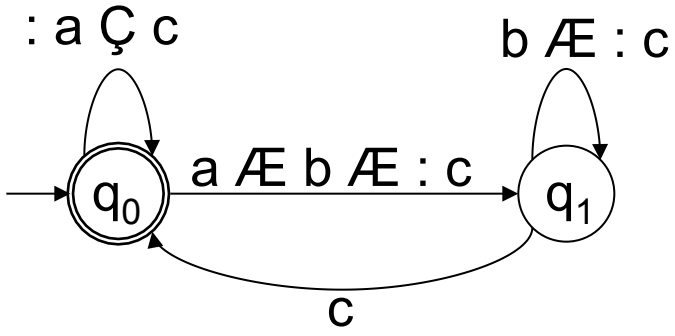


Even More...

Property: G(a ! (bUc))

Idea:

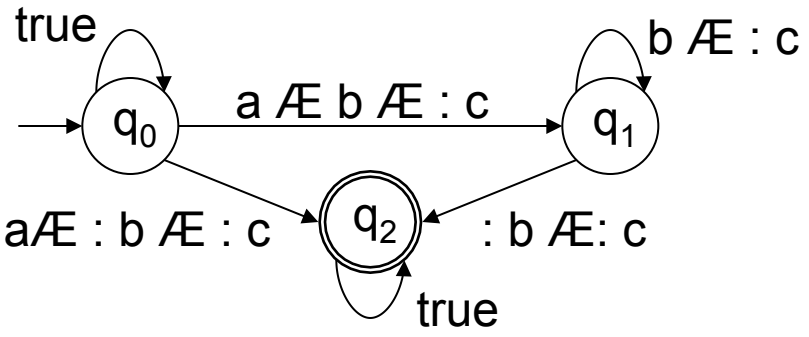
- q_0: Body of G immediately ok
- q_1: Awaiting c



Property: ¬G(a ! (bUc)) = F(a Æ ¬(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

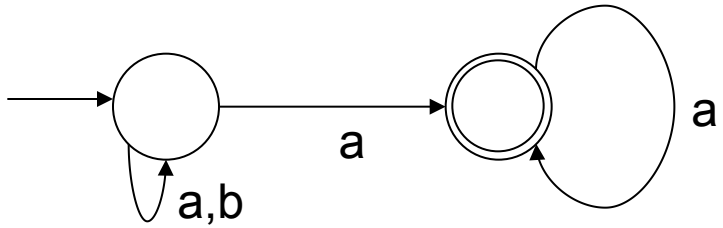
Hard case is complementation [Safra'88]

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 Σ = {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

- $\underline{F\phi} = \phi \vee O\underline{F\phi}$
- $\underline{G\phi} = \phi \wedge O\underline{G\phi}$
- $\underline{\phi \, U \, \psi} = \psi \vee (\phi \wedge O(\underline{\phi \, U \, \psi}))$
- $\underline{\text{Even } \phi} = \phi \wedge OO\underline{\text{Even } \phi}$

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 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

$$9\ X(02\ X\ \text{Æ}\ 8y8z(succ(y,z)\ !\ (y2X\ \$\ :\ z2X))\ \text{Æ}\ 8y(y2X\ !\ a(y)))$$

(all even symbols are a's)

QPLTL: LTL with propositional quantification

$$9\ X((X\ \text{Æ}\ G(X\ \$\ O{:}X)\ \text{Æ}\ G(x\ !\ a))$$

ω-regular expressions

$$a((a\ [\ b)a)^{\omega}$$

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