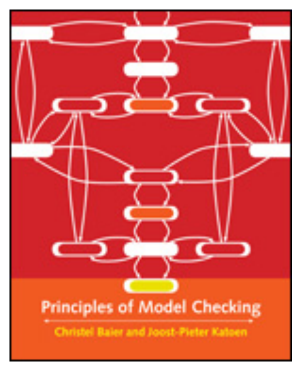
Lecture 3
Linear Temporal Logic (LTL)

Richard M. Murray
Nok Wongpiromsarn   Ufuk Topcu
California Institute of Technology

EECI, 18 March 2013

Outline

- Syntax and semantics of LTL
- Specifying properties in LTL
- Equivalence of LTL formulas
- Fairness in LTL
- Other temporal logics (if time)



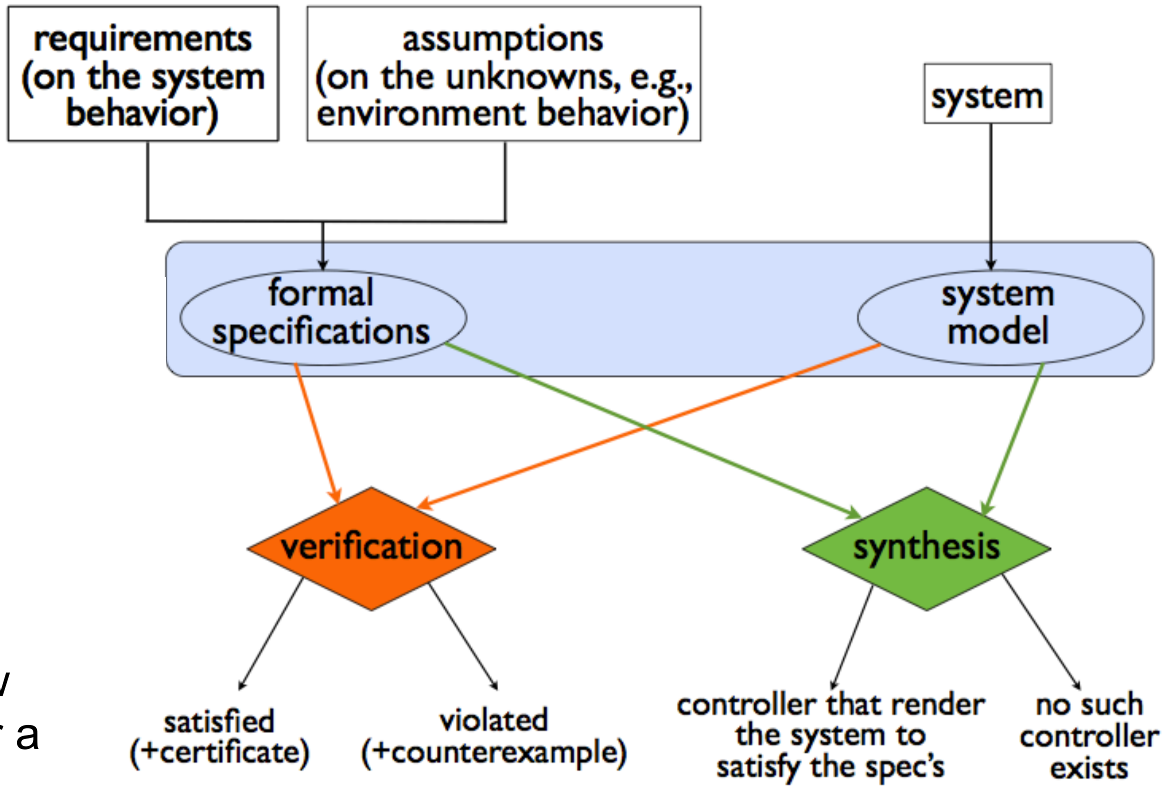
Principles of Model Checking,
Christel Baier and Joost-Pieter Katoen.
MIT Press, 2008.

Chapter 5

Formal Methods for System Verification

Specification using LTL

- Linear temporal logic (LTL) is a math'l language for describing linear-time prop's
- Provides a particularly useful set of operators for constructing LT properties without specifying sets



Methods for verifying an LTL specification

- *Theorem proving*: use formal logical manipulations to show that a property is satisfied for a given system model
- *Model checking*: explicitly check all possible executions of a system model and verify that each of them satisfies the formal specification
  - Roughly like trying to prove stability by simulating *every* initial condition
  - Works because discrete transition systems have finite number of states
  - Very good tools now exist for doing this efficiently (SPIN, nuSMV, etc)

Temporal Logic Operators

Two key operators in temporal logic

- ◊ "eventually" – a property is satisfied at some point in the future
- □ "always" – a property is satisfied now and forever into the future

"Temporal" refers underlying nature of time

- *Linear* temporal logic ⇒ each moment in time has a well-defined successor moment
- *Branching* temporal logic ⇒ reason about multiple possible time courses
- "Temporal" here refers to "ordered events"; no explicit notion of time

LTL = linear temporal logic

- Specific class of operators for specifying linear time properties
- Introduced by Pneuli in the 1970s (recently passed away)
- Large collection of tools for specification, design, analysis

Other temporal logics

- CTL = computation tree logic (branching time; will see later, if time)
- TCTL = timed CTL - check to make sure certain events occur in a certain time
- TLA = temporal logic of actions (Lamport) [variant of LTL]
- μ calculus = for reactive systems; add "least fixed point" operator (more on Thu)

Syntax of LTL

LTL formulas:

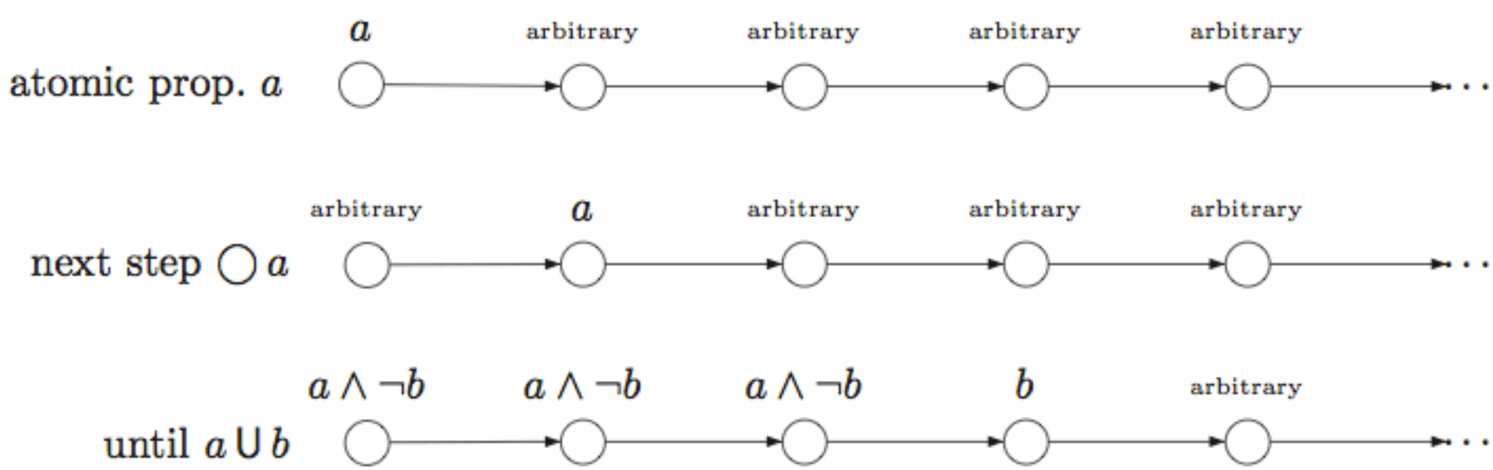
$$\varphi ::= \text{true} \;\big|\; a \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \bigcirc\,\varphi \;\big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

- a = atomic proposition
- ◯ = "next": φ is true at next step
- U = "until": φ_2 is true at some point, φ_1 is true until that time

Operator precedence

- Unary bind stronger than binary
- U takes precedence over ∧, ∨ and →

Formula evaluation: evaluate LTL propositions over a sequence of states (path):

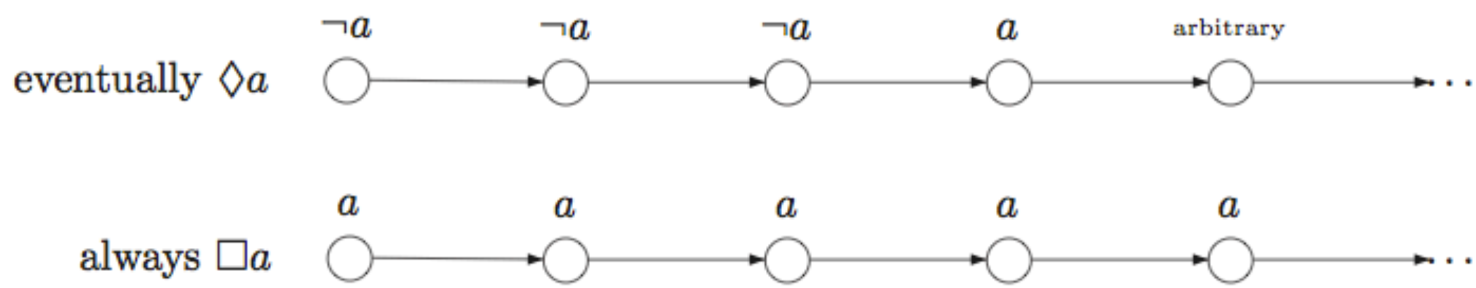


- Same notation as linear time properties: σ ⊨ φ (path "satisfies" specification)

Additional Operators and Formulas

"Primary" temporal logic operators

- Eventually $\Diamond\phi := \text{true}\ U\ \phi$ — ϕ will become true at some point in the future
- Always $\Box\phi := \neg\Diamond\neg\phi$ — ϕ is always true; "(never (eventually ($\neg\phi$)))"



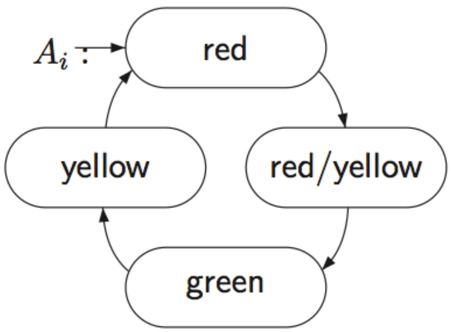
Some common composite operators

- $p \rightarrow \Diamond q$ — p implies eventually q (response)
- $p \rightarrow q\ U\ r$ — p implies q until r (precedence)
- $\Box\Diamond p$ — always eventually p (progress)
- $\Diamond\Box p$ — eventually always p (stability)
- $\Diamond p \rightarrow \Diamond q$ — eventually p implies eventually q (correlation)

Operator precedence

- Unary binds stronger than binary
- Bind from right to left:
  $\Box\Diamond p = (\Box\ (\Diamond p))$
  $p\ U\ q\ U\ r = p\ U\ (q\ U\ r)$
- U takes precedence over $\wedge$, $\vee$ and $\rightarrow$

Example: Traffic Light



System description

- Focus on lights in on particular direction
- Light can be any of three colors: green, yellow, read
- Atomic propositions = light color

Ordering specifications

- Liveness: “traffic light is green infinitely often”

$$\Box\Diamond \text{green}$$

- Chronological ordering: “once red, the light cannot become green immediately”

$$\Box\, (\text{red} \rightarrow \neg \bigcirc \text{green})$$

- More detailed: “once red, the light always becomes green eventually after being yellow for some time”

$$\Box(\text{red} \rightarrow (\Diamond\, \text{green} \wedge (\neg\, \text{green}\ U\ \text{yellow})))$$

$$\Box(\text{red} \rightarrow \bigcirc\, (\text{red}\ U\ (\text{yellow} \wedge \bigcirc\, (\text{yellow}\ U\ \text{green}))))$$

Progress property

- Every request will eventually lead to a response

$$\Box\, (\text{request} \rightarrow \Diamond\text{response})$$

Semantics: when does a path satisfy an LTL spec?

Definition 5.6. Semantics of LTL (Interpretation over Words)

Let φ be an LTL formula over AP. The LT property induced by φ is

$$Words(\varphi) = \left\{ \sigma \in (2^{AP})^{\omega} \mid \sigma \models \varphi \right\}$$

where the satisfaction relation $\models \subseteq (2^{AP})^{\omega} \times LTL$ is the smallest relation with the properties in Figure 5.2. ■

$$\begin{array}{lcllll}
\sigma & \models & \text{true} & & & \\
\sigma & \models & a & \text{iff} & a \in A_0 \quad (\text{i.e., } A_0 \models a) & \\
\sigma & \models & \varphi_1 \wedge \varphi_2 & \text{iff} & \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 & \\
\sigma & \models & \neg \varphi & \text{iff} & \sigma \not\models \varphi & \\
\sigma & \models & \bigcirc \varphi & \text{iff} & \sigma[1\ldots] = A_1 A_2 A_3 \ldots \models \varphi & \\
\sigma & \models & \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \exists j \geqslant 0.\ \sigma[j\ldots] \models \varphi_2 \text{ and } \sigma[i\ldots] \models \varphi_1, \text{ for all } 0 \leqslant i < j &
\end{array}$$

$$\begin{array}{lclll}
\sigma & \models & \Diamond \varphi & \text{iff} & \exists j \geqslant 0.\ \sigma[j\ldots] \models \varphi \\
\sigma & \models & \Box \varphi & \text{iff} & \forall j \geqslant 0.\ \sigma[j\ldots] \models \varphi.
\end{array}$$

Figure 5.2: LTL semantics (satisfaction relation $\models$) for infinite words over 2^{AP}.

Semantics of LTL

The semantics of the combinations of $\Box$ and $\Diamond$ can now be derived:

$$\sigma \models \Box\Diamond\varphi \quad \text{iff} \quad \overset{\infty}{\exists} j.\ \sigma[j\ldots] \models \varphi$$
$$\sigma \models \Diamond\Box\varphi \quad \text{iff} \quad \overset{\infty}{\forall} j.\ \sigma[j\ldots] \models \varphi.$$

Here, $\overset{\infty}{\exists} j$ means $\forall i \geqslant 0.\ \exists j \geqslant i$, "for infinitely many $j \in \mathbb{N}$", while $\overset{\infty}{\forall} j$ stands for $\exists i \geqslant 0.\ \forall j \geqslant i$, "for almost all $j \in \mathbb{N}$".

Definition 5.7. Semantics of LTL over Paths and States

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states, and let φ be an LTL-formula over AP.

- For infinite path fragment π of TS, the satisfaction relation is defined by

$$\pi \models \varphi \qquad \text{iff} \qquad trace(\pi) \models \varphi.$$

- For state $s \in S$, the satisfaction relation $\models$ is defined by

$$s \models \varphi \qquad \text{iff} \qquad (\forall \pi \in Paths(s).\ \pi \models \varphi).$$

- TS satisfies φ, denoted $TS \models \varphi$, if $Traces(TS) \subseteq Words(\varphi)$.

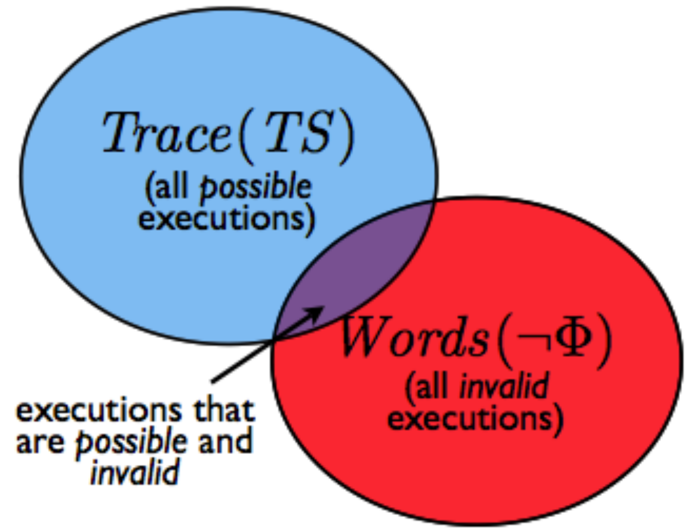
Semantics of LTL

From this definition, it immediately follows that

$$\begin{array}{ll} & TS \models \varphi \\ \text{iff} & \hfill \text{(* Definition 5.7 *)} \\ & Traces(TS) \subseteq Words(\varphi) \\ \text{iff} & \hfill \text{(* Definition of } \models \text{ for LT properties *)} \\ & TS \models Words(\varphi) \\ \text{iff} & \hfill \text{(* Definition of } Words(\varphi) \text{ *)} \\ & \pi \models \varphi \text{ for all } \pi \in Paths(TS) \\ \text{iff} & \hfill \text{(* Definition 5.7 of } \models \text{ for states *)} \\ & s_0 \models \varphi \text{ for all } s_0 \in I. \end{array}$$

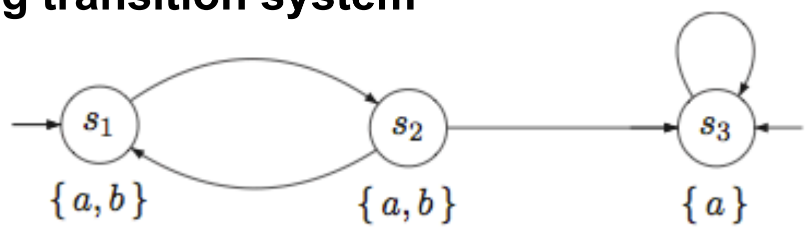
Remarks

- Which condition you use depends on type of problem under consideration
- For reasoning about correctness, look for (lack of) intersection between sets:



"Quiz"

Consider the following transition system



Consider the transition system *TS* depicted in Figure 5.3 with the set of propositions $AP = \{a, b\}$. For example, we have that $TS \models \Box a$, since all states are labeled with a, and hence, all traces of *TS* are words of the form $A_0 A_1 A_2 \ldots$ with $a \in A_i$ for all $i \geqslant 0$. Thus, $s_i \models \Box a$ for $i = 1, 2, 3$. Moreover:

$$s_1 \models \bigcirc(a \wedge b) \text{ since } s_2 \models a \wedge b \text{ and } s_2 \text{ is the only successor of } s_1$$

$$s_2 \not\models \bigcirc(a \wedge b) \text{ and } s_3 \not\models \bigcirc(a \wedge b) \text{ as } s_3 \in Post(s_2), s_3 \in Post(s_3) \text{ and } s_3 \not\models a \wedge b.$$

This yields $TS \not\models \bigcirc(a \wedge b)$ as s_3 is an initial state for which $s_3 \not\models \bigcirc(a \wedge b)$. As another example:

$$TS \models \Box(\neg b \rightarrow \Box(a \wedge \neg b)),$$

since s_3 is the only $\neg b$ state, s_3 cannot be left anymore, and $a \wedge \neg b$ in s_3 is true. However,

$$TS \not\models b \,\mathsf{U}\, (a \wedge \neg b),$$

since the initial path $(s_1 s_2)^\omega$ does not visit a state for which $a \wedge \neg b$ holds. Note that the initial path $(s_1 s_2)^* s_3^\omega$ satisfies $b \,\mathsf{U}\, (a \wedge \neg b)$. ■

Specifying Timed Properties for Synchronous Systems

For *synchronous* systems, LTL can be used as a formalism to specify "real-time" properties that refer to a discrete time scale. Recall that in synchronous systems, the involved processes proceed in a lock step fashion, i.e., at each discrete time instance each process performs a (sometimes idle) step. In this kind of system, the next-step operator $\bigcirc$ has a "timed" interpretation: $\bigcirc\varphi$ states that "at the next time instant φ holds". By putting applications of $\bigcirc$ in sequence, we obtain, e.g.:

$$\bigcirc^k \varphi \stackrel{\text{def}}{=} \underbrace{\bigcirc\bigcirc\ldots\bigcirc}_{k\text{-times}}\varphi \qquad \text{"}\varphi \text{ holds after (exactly) } k \text{ time instants".}$$

Assertions like "φ will hold within at most k time instants" are obtained by

$$\Diamond^{\leqslant k}\varphi = \bigvee_{0\leqslant i\leqslant k} \bigcirc^i \varphi.$$

Statements like "φ holds now and will hold during the next k instants" can be represented as follows:

$$\Box^{\leqslant k}\varphi = \neg\Diamond^{\leqslant k}\neg\varphi = \neg\bigvee_{0\leqslant i\leqslant k} \bigcirc^i \neg\varphi.$$

Remark

- Idea can be extended to non-synchronous case (eg, Timed CTL [later])

Equivalence of LTL Formulas

Definition 5.17. Equivalence of LTL Formulae

LTL formulae φ_1, φ_2 are *equivalent*, denoted $\varphi_1 \equiv \varphi_2$, if $Words(\varphi_1) = Words(\varphi_2)$. ∎

duality law	*idempotency law*
$\neg \bigcirc \varphi \equiv \bigcirc \neg \varphi$	$\Diamond\Diamond\varphi \equiv \Diamond\varphi$
$\neg\Diamond\varphi \equiv \Box\neg\varphi$	$\Box\Box\varphi \equiv \Box\varphi$
$\neg\Box\varphi \equiv \Diamond\neg\varphi$	$\varphi \,\mathsf{U}\, (\varphi \,\mathsf{U}\, \psi) \equiv \varphi \,\mathsf{U}\, \psi$
	$(\varphi \,\mathsf{U}\, \psi) \,\mathsf{U}\, \psi \equiv \varphi \,\mathsf{U}\, \psi$
absorption law	*expansion law*
$\Diamond\Box\Diamond\varphi \equiv \Box\Diamond\varphi$	$\varphi \,\mathsf{U}\, \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathsf{U}\, \psi))$
$\Box\Diamond\Box\varphi \equiv \Diamond\Box\varphi$	$\Diamond\psi \equiv \psi \vee \bigcirc\Diamond\psi$
	$\Box\psi \equiv \psi \wedge \bigcirc\Box\psi$
distributive law	
$\bigcirc(\varphi \,\mathsf{U}\, \psi) \equiv (\bigcirc\varphi) \,\mathsf{U}\, (\bigcirc\psi)$	
$\Diamond(\varphi \vee \psi) \equiv \Diamond\varphi \vee \Diamond\psi$	
$\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$	

Non-identities

- $\Diamond(a \wedge b) \not\equiv \Diamond a \wedge \Diamond b$
- $\Box(a \vee b) \not\equiv \Box a \vee \Box b$

Klavins
CDC, 03

LTL Specs for Control Protocols: RoboFlag Drill

Task description

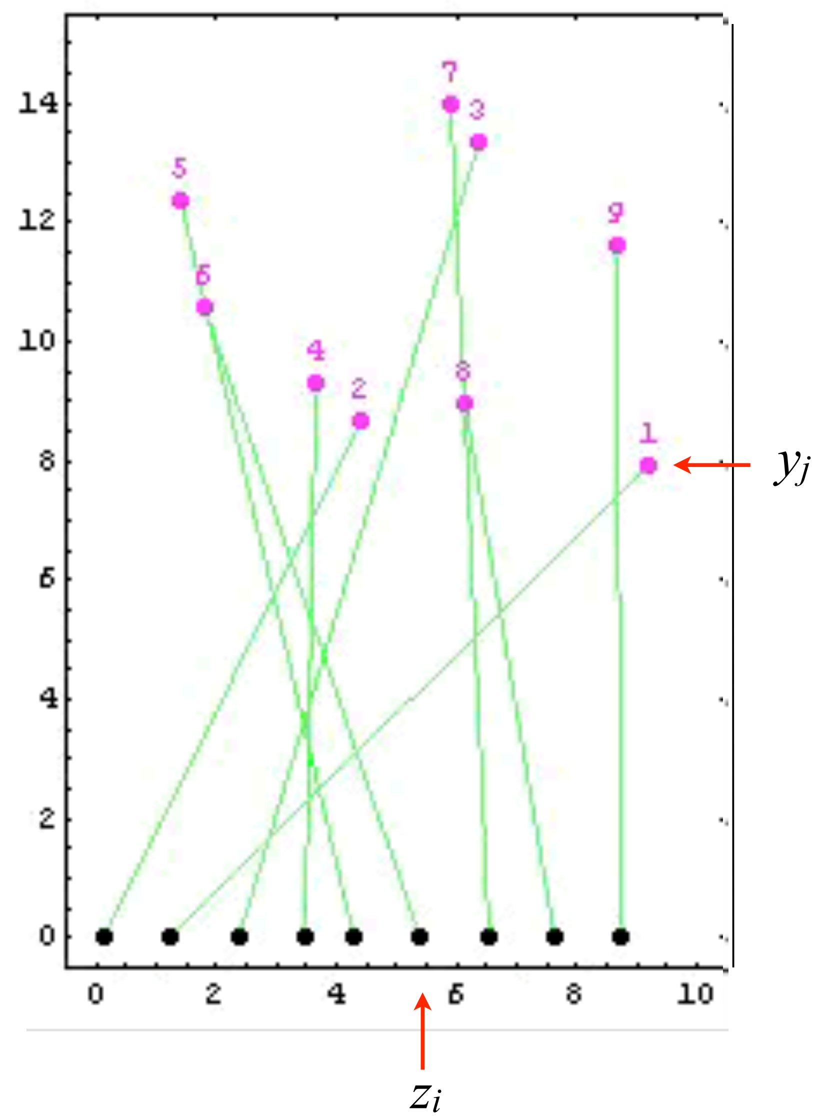
- Incoming robots should be blocked by defending robots
- Incoming robots are assigned randomly to whoever is free
- Defending robots must move to block, but cannot run into or cross over others
- Allow robots to communicate with left and right neighbors and switch assignments

Goals

- Would like a provably correct, distributed protocol for solving this problem
- Should (eventually) allow for lost data, incomplete information

Questions

- How do we describe task in terms of LTL?
- Given a protocol, how do we prove specs?
- How do we design the protocol given specs?



Properties for RoboFlag program

CCL formulas (will cover in more detail later)

- q' $\quad \circ$ q $\quad$ evaluate q at the next action in path
- p ↝ q $\quad \Box(p \rightarrow \Diamond q)$ $\quad$ "p leads to q": if p is true, q will eventually be true
- p **co** q $\quad$ "$\Box(p \rightarrow \circ q)$" $\quad$ if p is true, then next time state changes, q will be true

Safety (Defenders do not collide)

$$z_i < z_{i+1} \textbf{ co } z_i < z_{i+1}$$

True if robots i and i +1 have targets that cause crossed paths

Stability (switch predicate stays false)

$$\forall i \;.\; y_i > 2\delta \wedge z_i + 2\delta < z_{i+1} \wedge \neg switch_{i,i+1} \textbf{ co } \neg switch_{i,i+1}$$

Robots are "far enough" apart.

"Lyapunov" stability

- Remains to show that we actually approach the goal (robots line up with targets)
- Will see later we can do this using a Lyapunov function

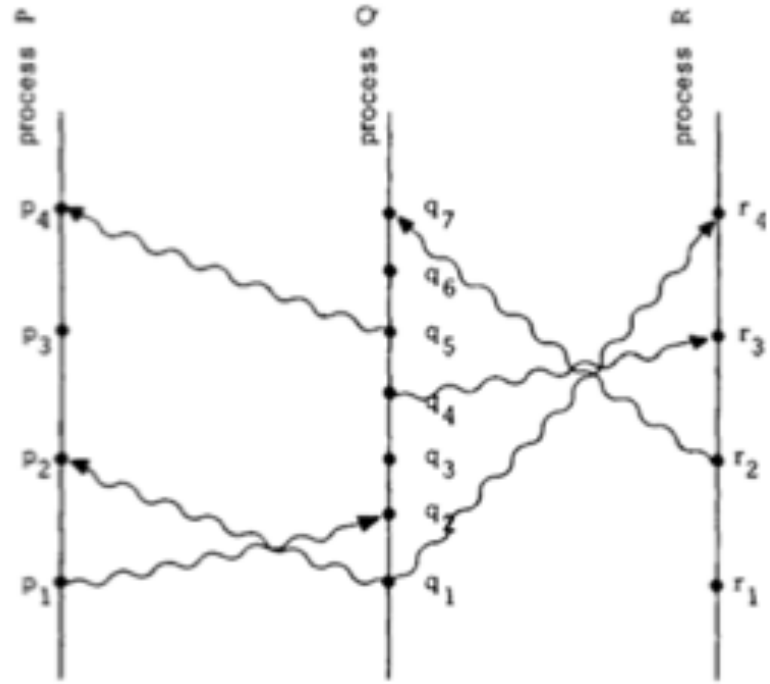
Fairness

Mainly an issue with concurrent processes

- To make sure that the proper interaction occurs, often need to know that each process gets executed reasonably often
- Multi-threaded version: each thread should receive some fraction of processes time

Two issues: implementation and specification

- Q1: How do we implement our algorithms to insure that we get "fairness" in execution
- Q2: how do we model fairness in a formal way to reason about program correctness



Example: Fairness in RoboFlag Drill

- To show that algorithm behaves properly, need to know that each agent communicates with neighbors regularly (infinitely often), in each direction

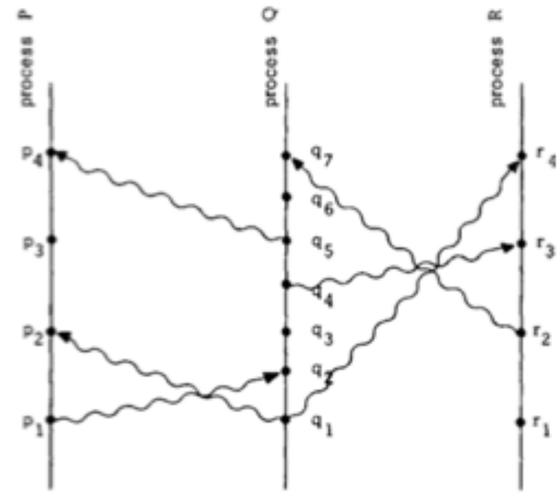
Difficulty in describing fairness depends on the logical formalism

- Turns out to be pretty easy to describe fairness in linear temporal logic
- Much more difficult to describe fairness for other temporal logics (eg, CTL & variants)

Fairness Properties in LTL

Definition 5.25 LTL Fairness Constraints and Assumptions

Let Φ and Ψ be propositional logical formulas over a set of atomic propositions



1. An *unconditional LTL fairness constraint* is an LTL formula of the form $ufair = \Box\Diamond\Psi.$

2. A *strong LTL fairness condition* is an LTL formula of the form $sfair = \Box\Diamond\Phi \longrightarrow \Box\Diamond\Psi.$

3. A *weak LTL fairness constraint* is an LTL formula of the form $wfair = \Diamond\Box\Phi \longrightarrow \Box\Diamond\Psi.$

An *LTL fairness assumption* is a conjunction of LTL fairness constraints (of any arbitrary type).

$$fair = ufair \wedge sfair \wedge wfair.$$

Rules of thumb

- strong (or unconditional) fairness: useful for solving contentions
- weak fairness: sufficient for resolving the non-determinism due to interleaving.

Fairness Properties in LTL

Fair paths and traces

$$
\begin{aligned}
FairPaths(s) &= \{ \pi \in Paths(s) \mid \pi \models fair \}, \\
FairTraces(s) &= \{ trace(\pi) \mid \pi \in FairPaths(s) \}.
\end{aligned}
$$

Definition 5.26. Satisfaction Relation for LTL with Fairness

For state s in transition system TS (over AP) without terminal states, LTL formula φ, and LTL fairness assumption *fair* let

$$
\begin{aligned}
s \models_{fair} \varphi \quad &\text{iff} \quad \forall \pi \in FairPaths(s).\, \pi \models \varphi \quad \text{and} \\
TS \models_{fair} \varphi \quad &\text{iff} \quad \forall s_0 \in I.\, s_0 \models_{fair} \varphi.
\end{aligned}
$$

■

Theorem 5.30. Reduction of $\models_{fair}$ to $\models$

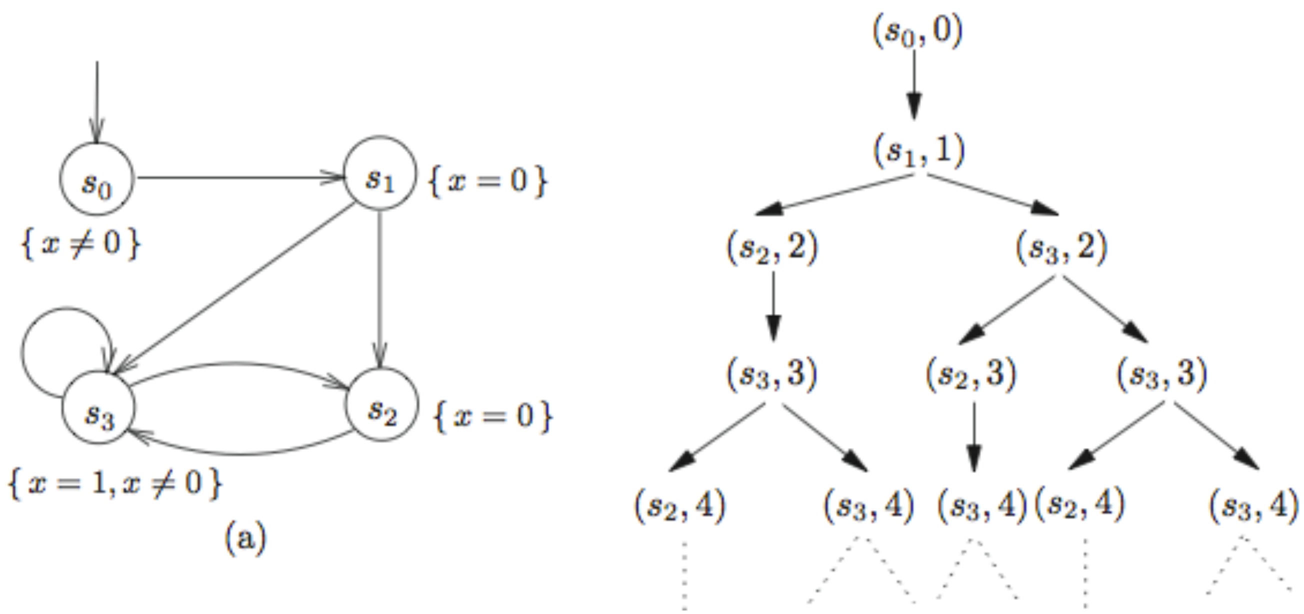
For transition system TS without terminal states, LTL formula φ, and LTL fairness assumption fair:

$$TS \models_{fair} \varphi \qquad \textit{if and only if} \qquad TS \models (fair \rightarrow \varphi).$$

Branching Time and Computational Tree Logic

Consider transition systems with multiple branches

- Eg, nondeterministic finite automata (NFA), nondeterministic Bucchi automata (NBA)
- In this case, there might be *multiple* paths from a given state
- Q: in evaluating a temporal logic property, which execution branch to we check?



Computational tree logic: allow evaluation over some or all paths

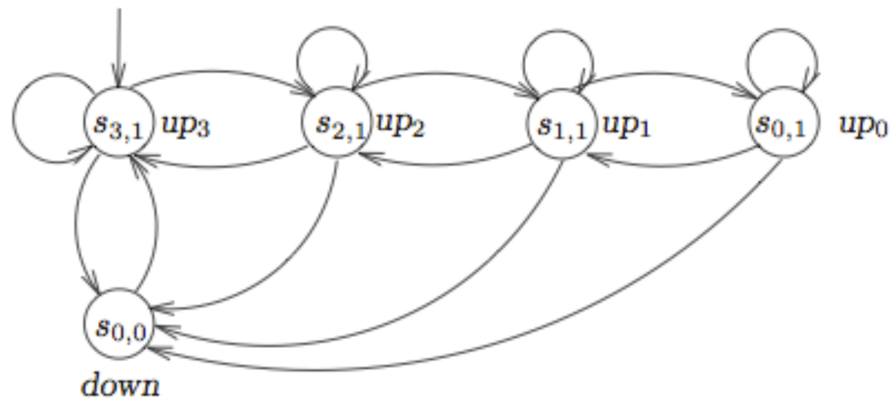
$$s \models \exists \varphi \quad \text{iff} \quad \pi \models \varphi \text{ for some } \pi \in Paths(s)$$

$$s \models \forall \varphi \quad \text{iff} \quad \pi \models \varphi \text{ for all } \pi \in Paths(s)$$

Example: Triply Redundant Control Systems

Systems consists of three processors and a single voter

- si,j = i processors up, j voters up
- Assume processors fail one at a time; voter can fail at any time
- If voter fails, reset to fully functioning state (all three processors up)
- System is operation if at least 2 processors remain operational



Properties we might like to prove

Property	*Formalization in CTL*	
Possibly the system never goes down	$\exists\Box\,\neg\,down$	Holds
Invariantly the system never goes down	$\forall\Box\,\neg\,down$	Doesn't hold
It is always possible to start as new	$\forall\Box\,\exists\Diamond\,up_3$	Holds
The system always eventually goes down and is operational until going down	$\forall\,((up_3 \,\vee\, up_2)\,\mathsf{U}\,down)$	Doesn't hold

Other Types of Temporal Logic

CTL ≠ LTL

- Can show that LTL and CTL are not proper subsets of each other
- LTL reasons over a complete path; CTL from a given state

Aspect	*Linear time*	*Branching time*
"behavior" in a state s	path-based: $trace(s)$	state-based: computation tree of s
temporal logic	LTL: path formulae φ $s \models \varphi$ iff $\forall \pi \in Paths(s).\, \pi \models \varphi$	CTL: state formulae existential path quantification $\exists\varphi$ universal path quantification: $\forall\varphi$

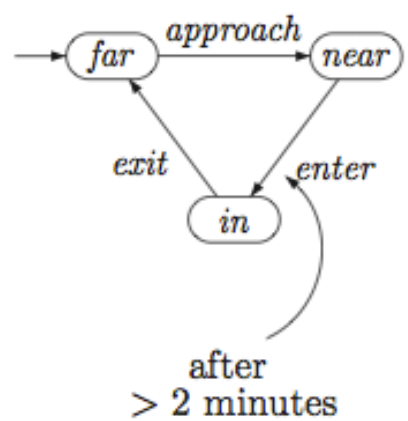
CTL* captures both

$$\Phi ::= \text{true} \;\big|\; a \;\big|\; \Phi_1 \wedge \Phi_2 \;\big|\; \neg\Phi \;\big|\; \exists\varphi \qquad \varphi ::= \Phi \;\big|\; \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \bigcirc\,\varphi \;\big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

Timed Computational Tree Logic

- Extend notions of transition systems and CTL to include "clocks" (multiple clocks OK)
- Transitions can depend on the value of clocks
- Can require that certain properties happen within a given time window

$$\forall\Box(far \longrightarrow \forall\Diamond^{\leqslant 1}\, \forall\Box^{\leqslant 1}\, up)$$



Summary: Specifying Behavior with LTL

Description

- State of the system is a snapshot of values of all variables
- Reason about *paths* σ: sequence of states of the system
- No strict notion of time, just ordering of events
- *Actions* are relations between states: state *s* is related to state *t* by action *a* if *a* takes *s* to *t* (via prime notation: x' = x + 1)
- *Formulas* (specifications) describe the set of allowable behaviors
- Safety specification: what actions are allowed
- Fairness specification: when can a component take an action (eg, infinitely often)

Example

- Action: $a \equiv x' = x + 1$
- Behavior: $\sigma \equiv x := 1, x := 2, x := 3, \ldots$
- Safety: $\Box x > 0$ (true for this behavior)
- Fairness: $\Box(x' = x + 1 \vee x' = x) \wedge \Box\Diamond (x' \neq x)$

- $\Box p$ ≡ **always** *p* (invariance)
- $\Diamond p$ ≡ **eventually** *p* (guarantee)
- $p \rightarrow \Diamond q$ ≡ *p* **implies eventually** *q* (response)
- $p \rightarrow q\, \mathcal{U}\, r$ ≡ *p* **implies** *q* **until** *r* (precedence)
- $\Box\Diamond p$ ≡ **always eventually** *p* (progress)
- $\Diamond\Box p$ ≡ **eventually always** *p* (stability)
- $\Diamond p \rightarrow \Diamond q$ ≡ **eventually** *p* **implies eventually** *q* (correlation)

Properties

- Can reason about time by adding "time variables" (t' = t + 1)
- Specifications and proofs can be difficult to interpret by hand, but computer tools existing (eg, TLC, Isabelle, PVS, SPIN, etc)