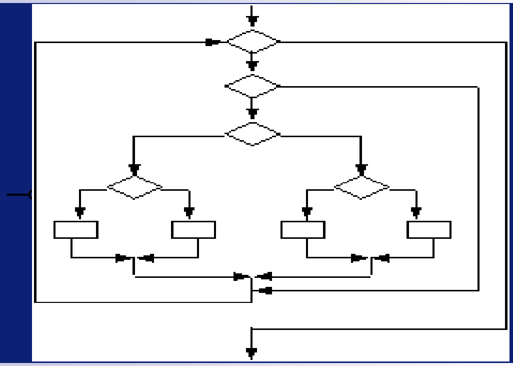
Spring 2007



Program correctness

Linear Time Temporal Logic

Marcello Bonsangue

Leiden Institute of Advanced Computer Science
Research & Education

Formal Verification

Verification techniques comprise

- a modelling framework $\mathcal{M}$

  to describe a system

- a specification language ϕ

  to describe the properties to be verified

- a verification method $\mathcal{M} \vDash \phi, \quad \Gamma \vdash \phi$

  to establish whether a model satisfies a property

Motivations

- For an elevator system, consider the requirements:
  - any request must ultimately be satisfied
  - the elevator never traverses a floor for which a request is pending without satisfying it

- Both concern the dynamic behavior of the system. They can be formalized using a time-dependent notation, like

$$z(t) = 1/2gt^2$$

  for the free-falling elevator

Example

- In first order logic, with
  - $E(t)$ = elevator position at time t
  - $P(n,t)$ = pending request at floor n at time t
  - $S(n,t)$ = servicing of floor n at time t

Any request must ultimately be satisfied

$$\forall t\ \forall n\ (\ P(n,t) \Rightarrow \exists t' > t : S(n,t')\ )$$

The elevator never traverse a floor for which a request is pending without satisfying it

$$\forall t \forall t' > t \forall n\ (P(n,t) \wedge E(t') \neq n \wedge \exists t < t'' < t' : E(t'') = n) \Rightarrow \exists t < t''' < t' : S(n,t''')$$

Temporal Logic

- First order logic is too cumbersome for these specifications
- Temporal logic is a logic tailored for describing properties involving time
  - the time parameter t disappears
  - temporal operators mimic linguistic constructs
    - always, until, eventually
  - the truth of a proposition depend on the state on which the system is

LTL: the language

- Atomic propositions $p_1, p_2, \ldots, q, \ldots$
  - to make statements about states of the system
  - elementary descriptions which in a given state of the system have a well-defined truth value:
    - the printer is busy
    - nice weather
    - open
    - $x+2=y$
  - Their choice depend on the system considered

LTL: the language

- Boolean combinators
  - true $\top$
  - false $\bot$
  - negation $\neg$
  - conjunction $\wedge$
  - disjunction $\vee$
  - implication $\Rightarrow$

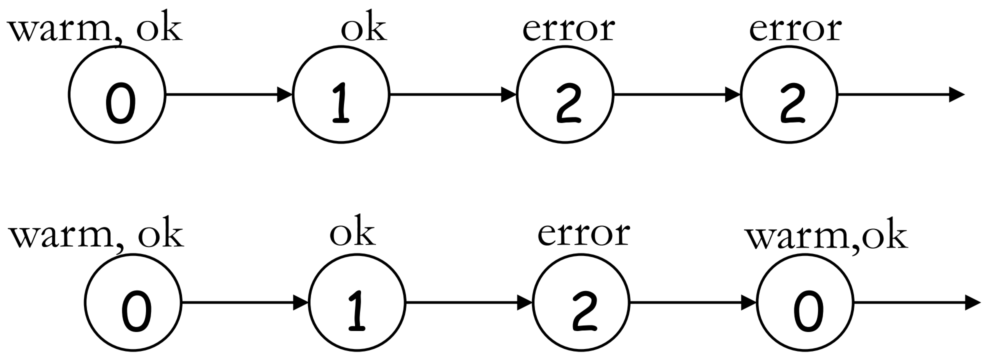
Note: read p $\Rightarrow$ q as "*if p then q*" rather than "*p implies q*".

Try (1 = 2) $\Rightarrow$ Sint_Klas_exists

LTL: the language

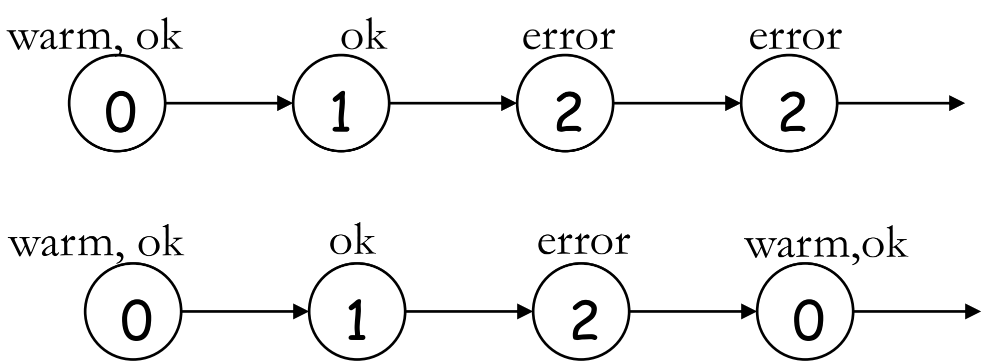
- Temporal combinators allows to speak about the sequencing of states along a computation (rather than about states individually)

- neXt X
  - $X\phi$ = *in the next state ϕ holds*
  - Examples: XXerror and XXXok



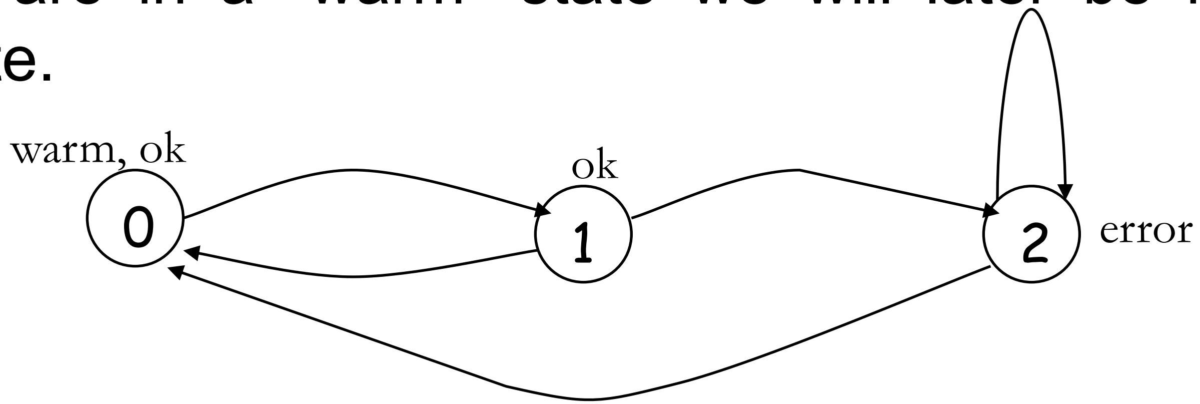
LTL: Temporal combinators

- Future F
  - Fϕ = *in some future state ϕ holds* (at least once and without saying in which state)
  - For example, warm ⇒ Fok holds if we are in a “warm” state then we will be in an “ok” state.



LTL: Temporal combinators

- Globally G
  - $G\phi$ = in all future states ϕ always holds
  - It is the dual of F: $G\phi = \neg F \neg \phi$
  - For example $G(\text{warm} \Rightarrow F\text{ok})$ holds if at any time when we are in a "warm" state we will later be in an "ok" state.



  - $G(\text{warm} \Rightarrow X\neg\text{warm})$? $G(\text{ok} \Rightarrow X\text{warm})$?

LTL: Temporal combinators

- Until U
  - $\phi_1 U \phi_2$ = ϕ_2 will hold in some future state, and in all intermediate states ϕ_1 will hold.
- Weak until W
  - $\phi_1 W \phi_2$ = ϕ_1 holds in all future states until ϕ_2 holds
  - it may be the case ϕ_2 will never hold

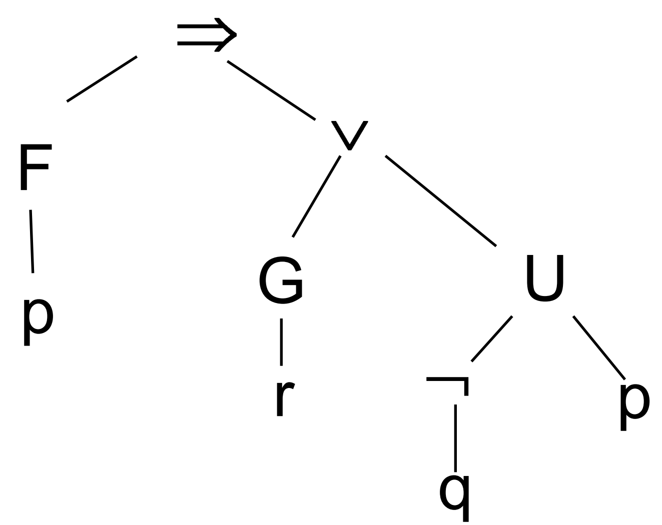
LTL: Temporal combinators

- Release R
  - $\phi_1 R \phi_2$ = ϕ_2 holds in all future state up to (and including) a state when ϕ_1 holds (if ever).
  - It is the dual of U: $\phi_1 R \phi_2 = \neg(\neg\phi_1 U \neg\phi_2)$

LTL - Priorities

- Unary connectives bind most tightly
  - $\neg$, X,F,G
- Next come U, R and W
- Finally come $\wedge$, $\vee$ and $\Rightarrow$

$\mathrm{F}p \Rightarrow \mathrm{G}r \vee \neg q\mathrm{U}p$

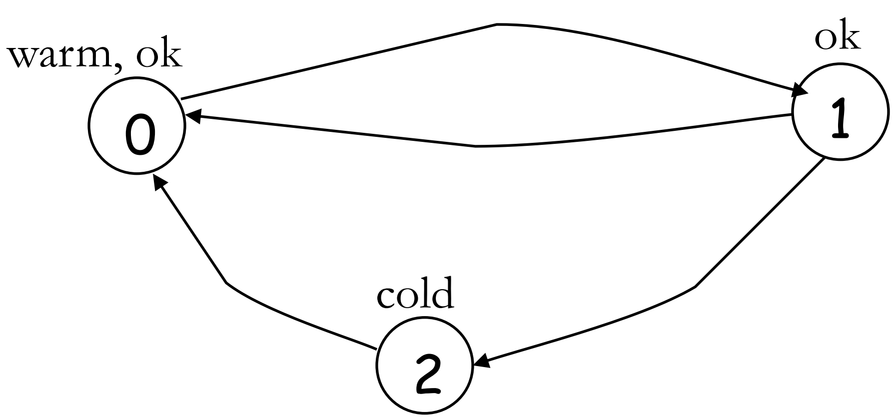


LTL models: Transition Systems

- Transition system: <S,$\rightarrow$,L>
  - S — set of states
  - L:S $\rightarrow$ $\mathcal{P}$(Atoms) — labelling function
  - $\rightarrow\ \subseteq$ SxS — transition relation
  - Every state s has some successor state s' with s $\rightarrow$ s'

- A system evolves from one state to another under the action of a transition

- We label a state with propositions that hold in that state



Computation paths

- Path: an infinite sequence π of states such that each consecutive pair is connected by a transition

$$0 \to 1 \to 2 \to 0 \to \ldots$$

- For $i \geq 1$, we write π^i for the suffix of a path π starting at i.

Semantics (I)

- Let M = <S,$\rightarrow$,L> be a transition system, and $\pi = s_1 \rightarrow s_2 \rightarrow \ldots$ a path of M.

- $\pi \vDash \top$ always
- $\pi \vDash p$ iff $p \in l(s_1)$
- $\pi \vDash \neg\phi$ iff $\pi \nvDash \phi$
- $\pi \vDash \phi_1 \wedge \phi_2$ iff $\pi \vDash \phi_1$ and $\pi \vDash \phi_2$

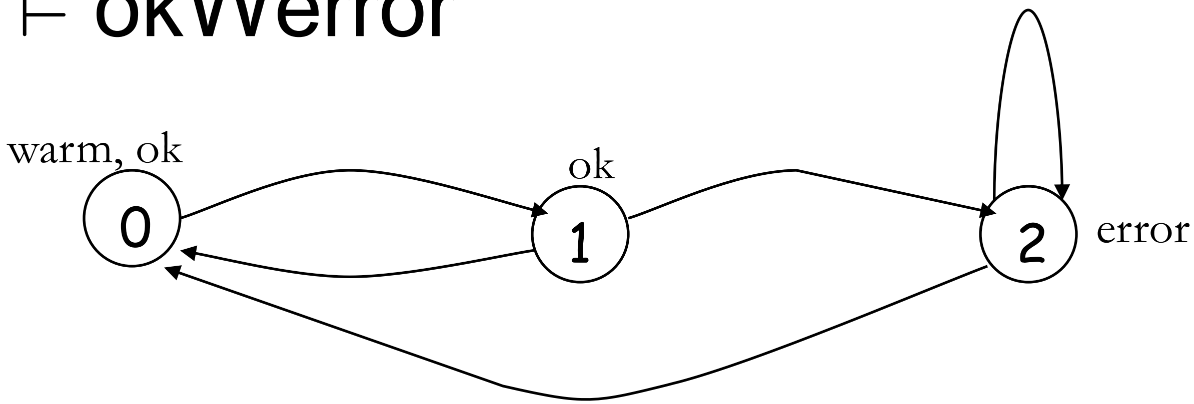
Semantics (II)

- $\pi \vDash X\phi$ iff $\pi^2 \vDash \phi$
- $\pi \vDash F\phi$ iff there is $1 \leq i$ such that $\pi^i \vDash \phi$
- $\pi \vDash G\phi$ iff for all $1 \leq i$, $\pi^i \vDash \phi$
- $\pi \vDash \phi_1 U \phi_2$ iff there is $1 \leq i$ such that $\pi^i \vDash \phi_2$ and for all $j<i$, $\pi^j \vDash \phi_1$
- $\pi \vDash \phi_1 W \phi_2$ iff either $\pi \vDash \phi_1 U \phi_2$ or for all $1 \leq i$, $\pi^i \vDash \phi_2$
- $\pi \vDash \phi_1 R \phi_2$ iff either there is $1 \leq i$ such that $\pi^i \vDash \phi_1$ and for all $j \leq i$, $\pi^j \vDash \phi_2$ or for all $1 \leq k$, $\pi^k \vDash \phi_2$

System properties

- M,s $\vDash \phi$ iff $\pi \vDash \phi$ for every path π of M starting from the state s

- M,0 $\vDash$ okWerror



- M,0 $\nvDash$ okUerror (Why?)