Software Verification and Testing

Lecture Notes: Temporal Logics

Motivation

traditional programs (whether terminating or non-terminating)

- can be modelled as relations
- are analysed wrt their input/output behaviour (pre/postconditions)
- invariants are also considered

correctness properties:

- partial correctness (of postcondition wrt precondition):
  "if the precondition holds then the postcondition will hold
  whenever the program terminates."
- total correctness (of postcondition wrt precondition):
  "if the precondition holds then the postcondition will hold
  and the program will terminate."

Motivation

reactive systems:

- yield no final output
- i/o-relations are not appropriate for their analysis
- partial/total correctness are not relevant
- invariants are much more important

correctness properties:

- are properties of the execution run/trace of the system
- have a dynamical/temporal flavour
- can be formalised/analysed using temporal logics

Time

question: what is time?

Augustinus: "If nobody asks me for it, I know it;
if I want to explain it to someone, I do not know it."

science/engineering:

- time is used and modelled without asking what it is
- selection of model is application-driven

question: then how to model time?

remark: still a difficult question. . .

Time

questions: is time

- discrete/continuous?
- instant/interval-based?
- linear/branching?
- ...

question: do we describe time by

- past, present, future?
- earlier, later?

question: is our model exogenous/endogenous?

- exogenous: compare different systems
- endogenous: focus on trace of one single system

Time in FOL

example: modelling time as relational structure $(T, <)$ in FOL

- possible properties of binary precedence relation $<$
  - irreflexivity: $\forall\, x. \neg x < x$
  - transitivity: $\forall\, x, y, z. (x < y \wedge y < z \rightarrow x < z)$
  - linearity: $\forall\, x, y. (x = y \vee x < y \vee y < x)$
  - (forward) discreteness:
    $\forall\, x\, \exists\, y. (x < y \rightarrow \exists\, z. (x < z \wedge \forall\, w. (x < w \rightarrow z < w)))$
  - density: $\forall\, x, y. (x < y \rightarrow \exists\, z. (x < z \wedge z < y))$
  - no end: $\forall\, x\, \exists\, y. x < y$

problems:

- many interesting properties cannot be expressed, e.g.,
  "every descending sequence of instants of time must be finite."
- formalism may be difficult to manipulate

Alternative Model

framework: we make the following assumptions

- instant-based model with initial state, infinite into future
- time is discrete (there are time-steps)
- we model temporal properties by propositional logics
- we add temporal operators (next-time, always, eventually,. . .) to model temporal system behaviour

remarks:

- temporal operators are examples of modal operators; temporal logics are examples of modal logics
- we will see logics for linear and branching time

Alternative Model

temporal logics and transition systems

- reactive system modelled by LTS
- formulae of propositional logics describe what holds in a state
- temporal operators describe what holds in the next state, in some future state, in all future states, etc.
- these descriptions can be understood as observer processes

consequence: satisfiability relation should be relativised to states or paths:

$$\mathcal{A}, s \models \phi \qquad \mathcal{A}, c \models \phi$$

where $\mathcal{A} = (S, T, \alpha, \beta, \lambda)$ is a LTS, $s \in S$, c is a path over T and ϕ is a formula of temporal logic

Temporal Logics and Transition Systems

example: consider a mutex algorithm (with two processes)

- we have seen how to model this as an LTS
- properties of interest might be
  - there never is a global state with both processes in the critical section (a safety property)
  - there never is a global state without any transition (deadlock-free)
  - traces from a global state are always infinite (deadlock-free)
  - every process that tries to enter the critical section from a global state will eventually enter it (a liveness property)
  - there is an infinite path where in each transition the two processes try to reach the critical section and never succeed (livelock)

Linear Temporal Logic

idea: linear temporal logic (LTL) expresses path properties of a LTS

syntax:

- the language of LTL is built from
  - a set P of propositional variables, the constant 0,
  - the logical connective $\rightarrow$,
  - the temporal operators X ("next") and U ("until")
- the formulae of LTL are defined (inductively) by the rules

$$\phi ::= p \mid 0 \mid \phi \rightarrow \psi \mid X\phi \mid \phi U \psi$$

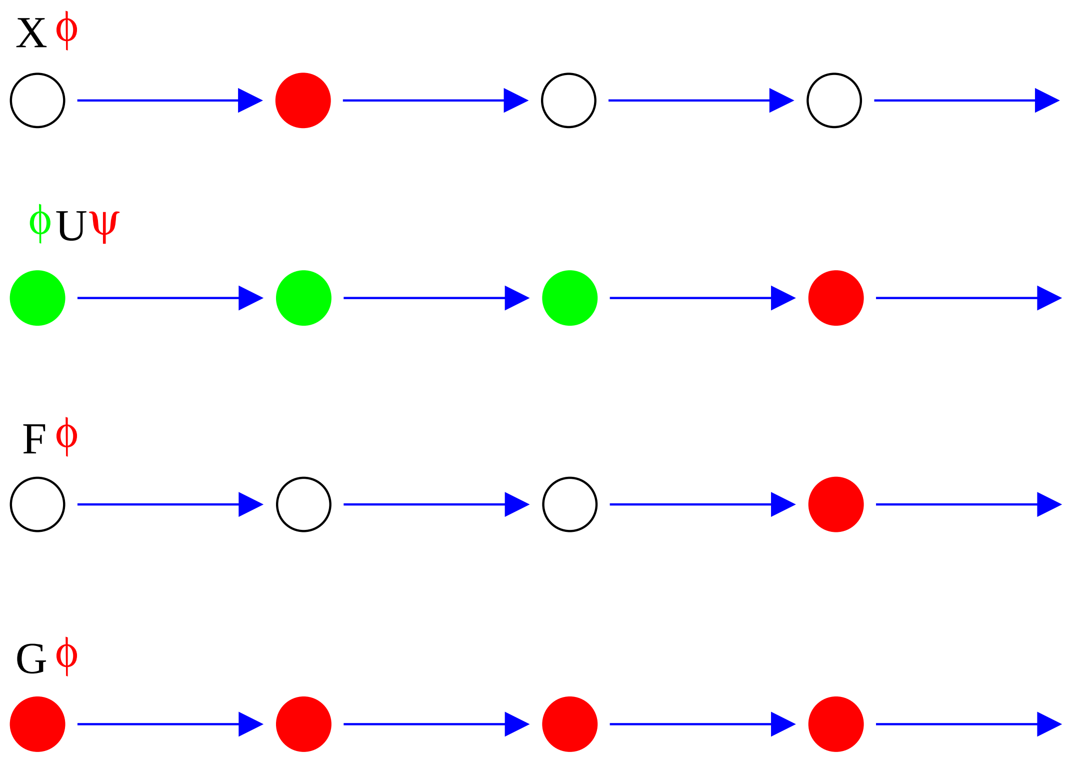
Linear Temporal Logic

syntax:

- other logical connectives can be defined as usual (question: how to define $\neg p$?)
- $1 = \neg 0$
- further temporal operators:
  - $F\phi = 1U\phi$ ("eventually", "finally")
  - $G\phi = \neg F \neg \phi$ ("always", "globally")

Linear Temporal Logic

intuitive semantics:



Linear Temporal Logic

semantics: Let $\mathcal{A} = (S, T\alpha, \beta, \lambda_\tau)$ be a LTS and c a path

- label states with the propositional variables that hold there

$$\lambda_\sigma : S \to 2^P$$

- we then inductively define $\mathcal{A}, c \models \phi$ for every LTL formula ϕ:
  - $\mathcal{A}, c \models 1$ and not $\mathcal{A}, c \models 0$
  - $\mathcal{A}, c \models p$ iff $p \in P$ and $p \in \lambda_\sigma(\alpha(c))$
  - $\mathcal{A}, c \models \phi \to \psi$ iff $\mathcal{A}, c \models \phi$ implies $\mathcal{A}, c \models \psi$
  - $\mathcal{A}, c \models X\phi$ iff $c = t \cdot c'$ and $\mathcal{A}, c' \models \phi$
  - $\mathcal{A}, c \models \phi U \psi$ iff $\mathcal{A}, c \models \psi$,
    or $c = t_1 \ldots t_n \cdot c'$ with $\mathcal{A}, c' \models \psi$ and $\forall\, 1 \leq i \leq n.\ \mathcal{A}, t_i \ldots t_n \cdot c' \models \phi$

Properties

global semantics:

- $\mathcal{A} \models \phi$ iff $\mathcal{A}, c \models \phi$ for all paths c
- $\models \phi$ iff $\mathcal{A} \models \phi$ for all LTS $\mathcal{A}$

semantics of always and eventually:

- $\mathcal{A}, c \models F\phi$ iff $c = c' \cdot c''$ (for some c') and $\mathcal{A}, c'' \models \phi$
- $\mathcal{A}, c \models G\phi$ iff $\mathcal{A}, c'' \models \phi$ for all c' with $c = c' \cdot c''$

further interesting operators:

- $F^\infty\phi = GF\phi$ "infinitely often"
- $G^\infty\phi = FG\phi$ "almost everywhere"

Properties

remark: many properties of temporal operators can be derived from their semantics

- $\models X\neg\phi \leftrightarrow \neg X\phi$
- $\models F(\phi \vee \psi) \leftrightarrow F\phi \vee F\psi$
- $\models (\phi U \psi) \leftrightarrow \psi \vee (\phi \wedge X(\phi U \psi))$
- $\models GG\phi \leftrightarrow G\phi$
- ...

Properties

properties of paths expressed in LTL:

- $\mathcal{A}, c \models X1$: c is non-empty
- $\mathcal{A}, c \models \neg FX0$: c is infinite

correctness properties:

- safety properties ("bad things never happen"): $G\neg\phi$
- liveness properties ("good things eventually happen"): $F\psi$ or $G(\phi \to F\psi)$
- fairness properties ("all processes are treated fairly be the scheduler") (examples later)

Properties

examples:

- partial correctness: (a safety property)

$$\phi \rightarrow G(\text{terminates} \rightarrow \psi)$$

- total correctness: (a liveness property)

$$\phi \rightarrow F(\text{terminates} \wedge \psi)$$

Properties

examples:

- "it will never be the case that two cars are at the same time at the crossing" (a safety property)

$$G\neg(c_1 \neq c_2 \wedge \mathsf{atCrossing}(c_1) \wedge \mathsf{atCrossing}(c_2)$$

- "I'll be back" (a liveness property)

$$\mathsf{Terminator}(x) \rightarrow F\ \mathsf{isback}(x)$$

- "all plagiators will eventually be caught" (another liveness property)

$$G(\mathsf{isPlagiator}(x) \rightarrow F\ \mathsf{iscaught}(x))$$

Properties

fairness: (there are several other fairness properties)

- impartiality: every process is executed infinitely often

$$\forall\, i. F^{\infty}\text{executed}_i$$

- weak fairness: every process enabled almost everywhere is executed infinitely often

$$\forall\, i. (G^{\infty}\text{enabled}_i \rightarrow F^{\infty}\text{executed}_i)$$

- strong fairness: every process enabled infinitely often is executed infinitely often

$$\forall\, i. (F^{\infty}\text{enabled}_i \rightarrow F^{\infty}\text{executed}_i)$$

Computational Tree Logic

idea: for computational tree logic (CTL)

- instead of paths, consider properties of states of LTS unfolded to tree
- quantify existentially/universally over transitions from a state

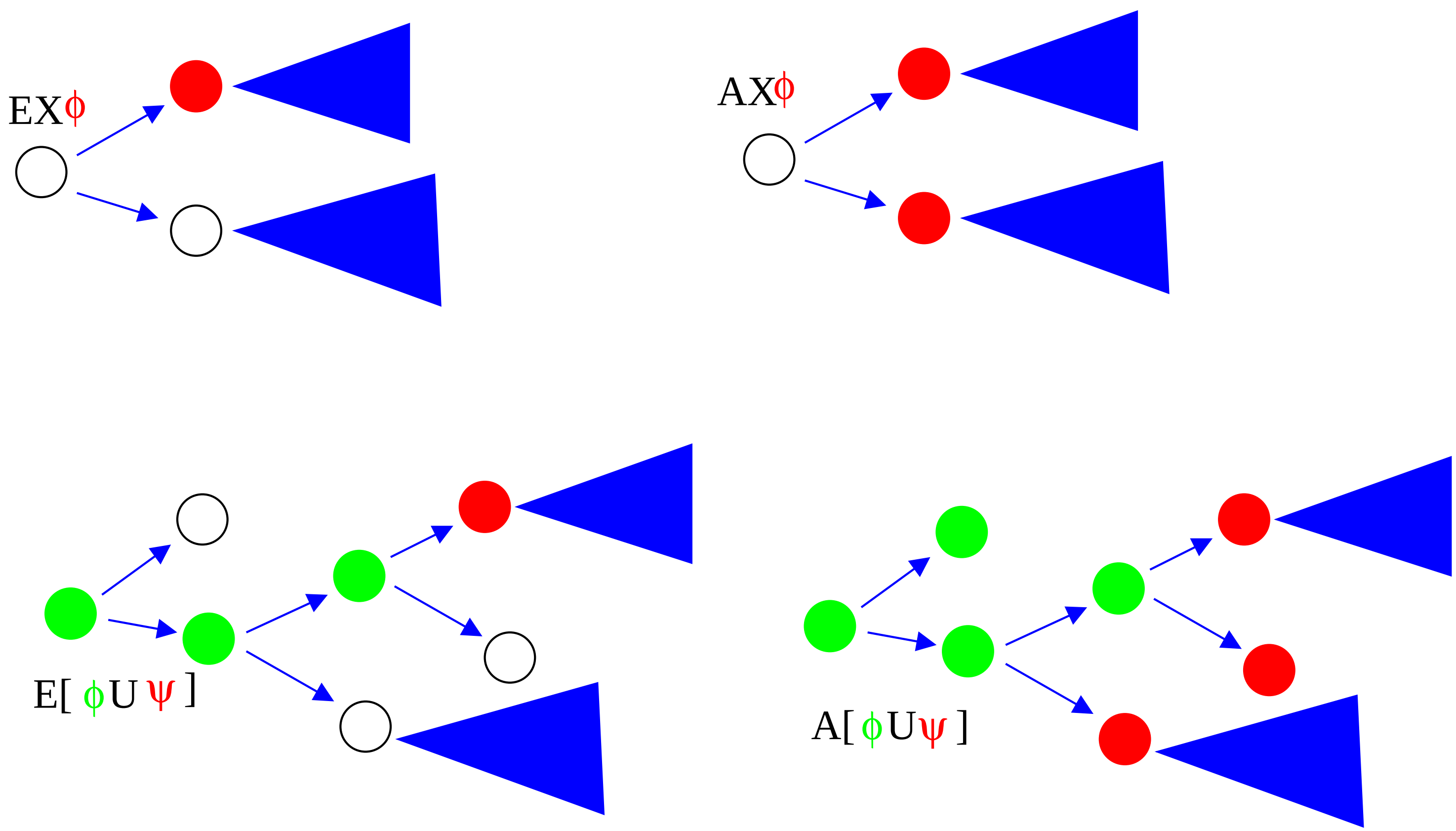
language:

- propositional part as for LTL
- next-step operators AX and EX
- until operators $E[\ \_U\_\ ]$ and $A[\ \_U\_\ ]$

formulae: $\phi ::= p \mid 0 \mid \phi \to \psi \mid EX\phi \mid AX\phi \mid E[\phi U \psi] \mid A[\phi U \psi]$

Computational Tree Logic

intuitive semantics:



Computational Tree Logic

definition: a maximal path is a path that is either infinite or ending in a state that is not the source of another transition

semantics: inductively define $\mathcal{A}, s \models \phi$ for state s and CTL-formula ϕ

- propositional cases similar to LTL ($\mathcal{A}, s \models p$ iff $p \in \lambda_\sigma(s)$)
- $\mathcal{A}, s \models EX\phi$ iff $\mathcal{A}, s' \models \phi$ for some transition with source s and target s'
- $\mathcal{A}, s \models AX\phi$ iff $\mathcal{A}, s' \models \phi$ for all transitions with source s and target s'
- $\mathcal{A}, s \models E[\phi U \psi]$ iff there is a maximal path $c = t_1 \ldots t_n \ldots$ with source s such that either $\mathcal{A}, s \models \psi$,
  or there is a $k \in \mathbb{N}$ such that $\mathcal{A}, \beta(t_k) \models \psi$ and $\forall 1 \leq i \leq k . \mathcal{A}, \alpha(t_i) \models \phi$
- $\mathcal{A}, s \models A[\phi U \psi]$ iff for all maximal paths $c = t_1 \ldots t_n \ldots$ with source s either $\mathcal{A}, s \models \psi$,
  or there is a $k \in \mathbb{N}$ such that $\mathcal{A}, \beta(t_k) \models \psi$ and $\forall 1 \leq i \leq k . \mathcal{A}, \alpha(t_i) \models \phi$,
  or c is a finite path and for every i, $\mathcal{A}, \alpha(t_i) \models \phi$ and $\mathcal{A}, \beta(t_i) \models \phi$

Computational Tree Logic

remarks:

- operators for "eventually" and "globally" can again be defined
- $\mathcal{A}, s \models AX\phi$ holds in particular if s is not the source of any transition
- again, a rich calculus for CTL follows from the semantics

LTL vs CTL: both logics have particular advantages

- CTL can distinguish some LTS that LTL cannot
- LTL cannot express "possibility" properties
- CTL cannot express fairness conditions
  (there are no path formulae to express F^{∞})

remark: the logic CTL^* overcomes these restrictions;
it subsumes both LTL and CTL and has state and path formulae

Model Checking

model checking problem: Given a LTS $\mathcal{A}$ and a formula ϕ (in some temporal logic), does the following hold?

$$\mathcal{A}, s \models \phi \qquad \mathcal{A}, c \models \phi \qquad \mathcal{A} \models \phi$$

model checking and verification: the LTS encodes a reactive system, the correctness properties is described in the temporal logic

remark: for finite LTS model checking problems are decidable

- validity of the temporal logic formulae can be checked by global search on the LTS
- different logics lead to different search complexities
- if the formula does not hold, the failure path/run provides information for bug fixing

Model Checking

algorithmic aspects:

- intuitively, the temporal logics formula is compiled into an observer process that runs in parallel with the LTS
- global model checking: recurse on formulae; but evaluate on global LTS
- local model checking: explore LTS locally; but evaluate entire formulae
- worst-case complexity is the same, but average behaviour can differ
- LTL is usually treated locally, CTL globally
- the performance critically depends on storage space, on efficient data structures (e.g. binary decision diagrams) and on heuristics