Formal Methods and Functional Programming

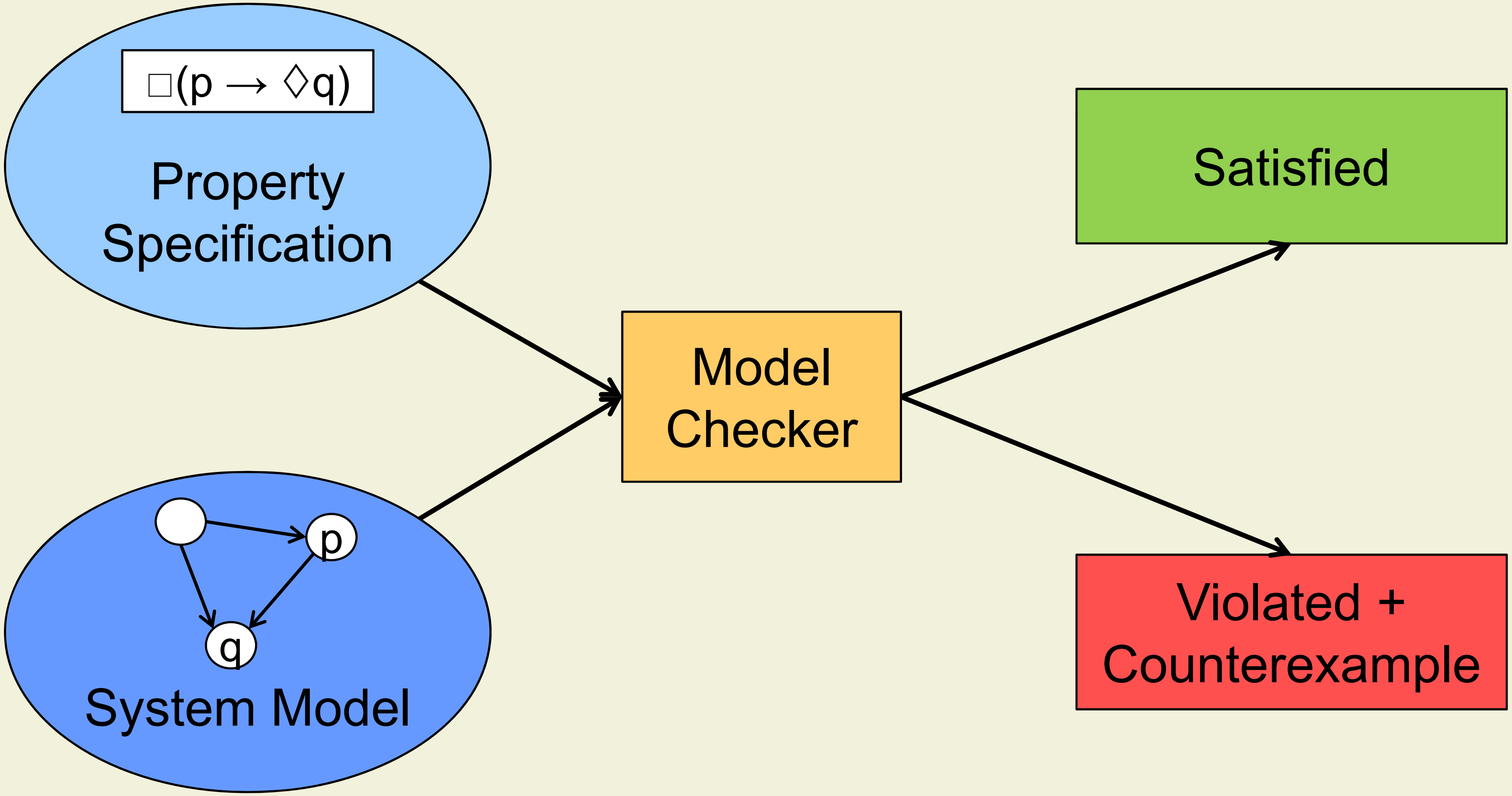
Linear Temporal Logic

Peter Müller

Programming Methodology Group
ETH Zurich

The slides in this section are partly based on the course *Automata-based System Analysis* by Felix Klaedtke

Model Checking



Motivation

- Many interesting properties relate several states
- Example: all opened files must be closed eventually

Motivation

- Many interesting properties relate several states
- Example: all opened files must be closed eventually
  - For a terminating program s

$$\langle \mathtt{s}, \sigma \rangle \rightarrow_1^* \sigma' \text{ and } \sigma(\mathtt{o}) = 0 \text{ then } \sigma'(\mathtt{o}) = 0$$

Motivation

- Many interesting properties relate several states
- Example: all opened files must be closed eventually
  - For a terminating program $\mathtt{s}$

$$\langle \mathtt{s}, \sigma \rangle \rightarrow_1^* \sigma' \text{ and } \sigma(\mathtt{o}) = 0 \text{ then } \sigma'(\mathtt{o}) = 0$$

  - For a deterministic, non-terminating program $\mathtt{s}$

$$\langle \mathtt{s}, \sigma \rangle \rightarrow_1^* \langle s', \sigma' \rangle \text{ and } \sigma(\mathtt{o}) = 0 \text{ and } \sigma'(\mathtt{o}) = 1 \text{ then there exist } s'', \sigma'' \text{ such that } \langle s', \sigma' \rangle \rightarrow_1^* \langle s'', \sigma'' \rangle \text{ and } \sigma''(\mathtt{o}) = 0$$

Motivation

- Many interesting properties relate several states
- Example: all opened files must be closed eventually
  - For a terminating program s

$$\langle \texttt{s}, \sigma\rangle \rightarrow_1^* \sigma' \text{ and } \sigma(\texttt{o}) = 0 \text{ then } \sigma'(\texttt{o}) = 0$$

  - For a deterministic, non-terminating program s

$$\langle \texttt{s}, \sigma\rangle \rightarrow_1^* \langle s', \sigma'\rangle \text{ and } \sigma(\texttt{o}) = 0 \text{ and } \sigma'(\texttt{o}) = 1 \text{ then there exist } s'', \sigma'' \text{ such that } \langle s', \sigma'\rangle \rightarrow_1^* \langle s'', \sigma''\rangle \text{ and } \sigma''(\texttt{o}) = 0$$

  - For a non-deterministic, non-terminating program s

$$wc : \text{Stm} \times \text{State} \times \mathbb{N} \rightarrow \text{Bool}$$

$$\begin{aligned} wc(s, \sigma, n) \Leftrightarrow \sigma(o) = 0 \vee \\ &(\text{for all } s', \sigma' : \text{if } \langle s, \sigma\rangle \rightarrow_1 \langle s', \sigma'\rangle \text{ then there exists} \\ &m \in \mathbb{N} \text{ such that } m < n \text{ and } wc(s', \sigma', m)) \end{aligned}$$

$$\langle \texttt{s}, \sigma\rangle \rightarrow_1^* \langle s', \sigma'\rangle \text{ and } \sigma(\texttt{o}) = 0 \text{ and } \sigma'(\texttt{o}) = 1 \text{ then there exists } n \in \mathbb{N} \text{ such that } wc(s', \sigma', n)$$

6. Linear Temporal Logic

Transition Systems Revisited

- We use a slightly different definition here (than earlier in the course)

- A finite transition system is a tuple $(\Gamma, \sigma_I, \rightarrow)$
  - Γ: a finite set of configurations
  - σ_I: an initial configuration, $\sigma_I \in \Gamma$
  - $\rightarrow$: a transition relation, $\rightarrow \subseteq \Gamma \times \Gamma$

- Difference: we have a fixed initial configuration
  - In this section, transition systems model only one program/system, not all programs of a programming language

- Difference: we omit terminal configurations from the definition
  - Simplifies theory
  - Termination can be modelled by transition to a special extra sink state (which allows further transitions only back to itself)

Transition System of a Promela Model

- Configurations: states (see previous section)
  - Global variables, global channels
  - Per active process: local variables, local channels, location counter

- Initial configuration: initial state (see previous section)

- Transition relation: defined by operational semantics of statements
  - We keep semantics informal

- A Promela model has a finite number of states
  - Finite number of active processes (limited to 255)
  - Finite number of variables and channels
  - Finite ranges of variables
  - Finite buffers of channels

Computations

- Infinite sequences
  - S^ω is the set of infinite sequences of elements of set S
  - $s_{[i]}$ denotes the i-th element of the sequence $s \in S^\omega$
- $\gamma \in \Gamma^\omega$ is a computation of a transition system if:
  - $\gamma_{[0]} = \sigma_I$
  - $\gamma_{[i]} \rightarrow \gamma_{[i+1]}$ (for all $i \geq 0$)
  - Note: we use σ to range over the states Γ of a transition system
  - Note (notation above): if $\gamma = \sigma_0\sigma_1\sigma_2\sigma_3\ldots$ then $\gamma_{[i]} = \sigma_i$
- $\mathcal{C}(TS)$ is the set of all computations of a transition system TS

Linear-Time Properties

- Linear-time properties (LT-properties) can be used to specify the permitted computations of a transition system

- A linear-time property P over Γ is a subset of Γ^ω
  - P specifies a particular set of infinite sequences of configurations

- TS satisfies LT-property P (over Γ)

$$TS \vDash P \text{ if and only if } \mathcal{C}(TS) \subseteq P$$

  - All computations of TS belong to the set P

- By contrast: branching-time properties (not in this course) can also express the existence of a computation
  - Example: "It is always possible to return to the initial state"

LT-Properties: Example

- All opened files must be closed eventually

$$P = \{\gamma \in \Gamma^\omega \mid \forall i \geq 0 : \gamma_{[i]}(o) = 1 \Rightarrow \exists n > 0 : \gamma_{[i+n]}(o) = 0\}$$

- LT-properties precisely express properties of computations
  - Non-termination is handled by infinite sequences
  - Non-determinism is handled by considering each computation separately

- However, the explicit representation above (defining the set of sequences) is not convenient

- Logical formalism needed to simplify specification of LT-properties

From Configurations to (Sets of) Propositions

- For a transition system TS, we additionally specify a set AP of atomic propositions (of our choice)
  - An atomic proposition is a proposition containing no logical connectives
  - Example: $AP = \{open, closed\}$ (for files)
  - Example: $AP = \{x > 0, y \leq x\}$
- We must provide a labeling function that maps configurations to sets of atomic propositions from AP
  - $L : \Gamma \rightarrow \mathcal{P}(AP)$
  - Example: $L(\sigma) = \begin{cases} \{open\} & \text{if } \sigma(o) = 1 \\ \{closed\} & \text{if } \sigma(o) = 0 \\ \{\} & \text{otherwise} \end{cases}$
- We call $L(\sigma)$ an abstract state
- From now on, we consider AP and L to be part of the transition system

Traces

- A trace is an abstraction of a computation
  - Observe only the propositions of each state, not the concrete state itself
  - Infinite sequence of abstract states ($\mathcal{P}(AP)^\omega$)
- $t \in \mathcal{P}(AP)^\omega$ is a trace of a transition system TS if $t = L(\gamma_{[0]})L(\gamma_{[1]})L(\gamma_{[2]}), \ldots$ and γ is a computation of TS
- $\mathcal{T}(TS)$ is the set of all traces of a transition system TS
- LT-properties are typically specified over infinite sequences of abstract states, rather than over sequences of configurations:

$$P = \{t \in \mathcal{P}(AP)^\omega \mid \forall i \geq 0 : open \in t_{[i]} \Rightarrow \exists n > 0 : closed \in t_{[i+n]}\}$$

Safety Properties

- Intuition
  - "Something bad is never allowed to happen (and can't be fixed)"

- An LT-property P is a safety property if for all infinite sequences $t \in \mathcal{P}(AP)^\omega$:
  if $t \notin P$ then there is a finite prefix $\hat{t}$ of t such that for every infinite sequence t' with prefix $\hat{t}$, $t' \notin P$
  - $\hat{t}$ is called a bad prefix; essentially, this finite sequence of steps already violates the property (whatever happens afterwards)

- Safety properties are violated in finite time and cannot be repaired

- Examples
  - State properties, for instance, invariants

$$P = \{t \in \mathcal{P}(AP)^\omega \mid \forall i \geq 0 : open \in t_{[i]} \lor closed \in t_{[i]}\}$$

  - "Money can be withdrawn only after correct PIN has been entered"

Liveness Properties

- Intuition
  - "Something good will happen eventually"
  - "If the good thing has not happened yet, it could happen in the future"

- An LT-property P is a liveness property if every finite sequence $\hat{t} \in \mathcal{P}(AP)^*$ is a prefix of an infinite sequence $t \in P$
  - A liveness property does not rule out any prefix
  - Every finite prefix can be extended to an infinite sequence that is in P

- Liveness properties are violated in infinite time

- Examples
  - All opened files must be closed eventually

$$P = \{t \in \mathcal{P}(AP)^\omega \mid \forall i \geq 0 : open \in t_{[i]} \Rightarrow \exists n > 0 : closed \in t_{[i+n]}\}$$

  - "The program terminates eventually"

6. Linear Temporal Logic

Linear Temporal Logic

- Linear Temporal Logic (LTL) allows us to formalize LT-properties of traces in a convenient and succinct way

- We will see syntax and semantics for LTL (no inference rules, etc.)

- Whether or not the traces of a finite transition system satisfy an LTL formula is decidable (see next section)

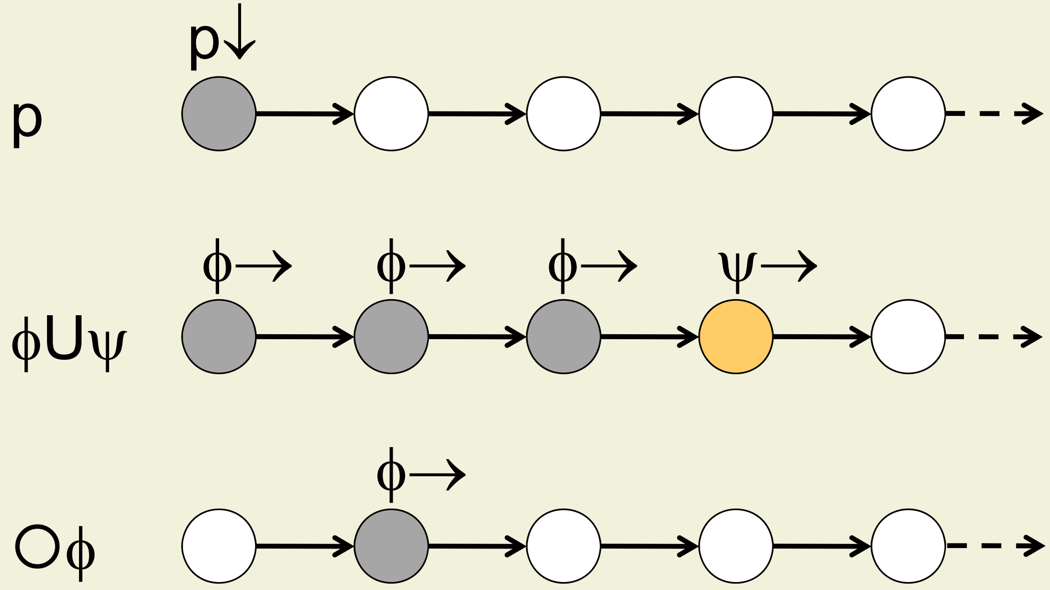
LTL: Basic Operators

- Syntax

$$\phi = p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \, \mathsf{U} \, \phi \mid \bigcirc\phi$$

  - where p is a proposition from a chosen set of atomic propositions $AP \neq \emptyset$

- Intuitive meaning of temporal logic formulas



p true "now"

ϕ "until" ψ

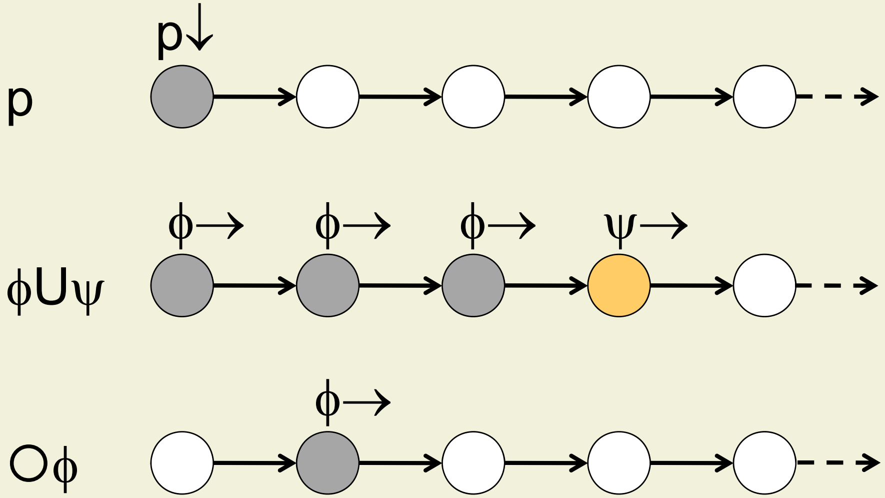
"next" ϕ

LTL: Semantics

- $t \models \phi$ expresses that trace $t \in \mathcal{P}(AP)^{\omega}$ satisfies LTL formula ϕ

$t \models p$	iff	$p \in t_{[0]}$
$t \models \neg\phi$	iff	not $t \models \phi$
$t \models \phi \wedge \psi$	iff	$t \models \phi$ and $t \models \psi$
$t \models \phi \,\mathsf{U}\, \psi$	iff	there is a $k \geq 0$ with $t_{(\geq k)} \models \psi$ and $t_{(\geq j)} \models \phi$ for all j such that $0 \leq j < k$
$t \models \bigcirc\phi$	iff	$t_{(\geq 1)} \models \phi$

  - where $t_{(\geq i)}$ is the suffix of t starting at t_i



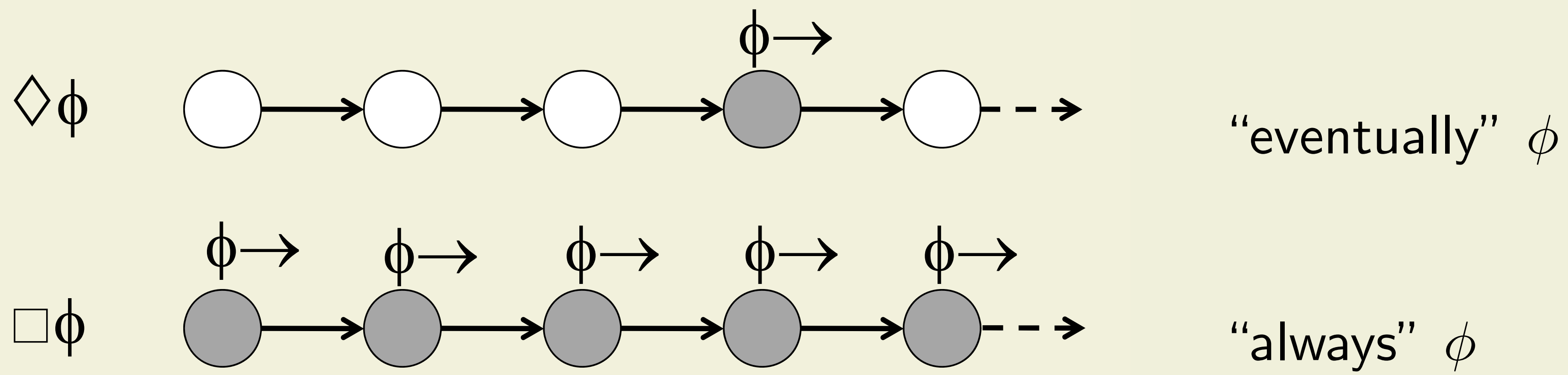
p true "now"

ϕ "until" ψ

"next" ϕ

Derived Operators

- *true*, *false*, $\vee$, $\Rightarrow$, $\Leftrightarrow$ defined as usual
- Eventually: $\Diamond\phi \equiv (\mathit{true}\ \mathsf{U}\ \phi)$
- Always (from now): $\Box\phi \equiv \neg\Diamond\neg\phi$



- Precedence: unary operators always have highest precedence. So, $\Diamond\phi \Rightarrow \psi$ means $(\Diamond\phi) \Rightarrow \psi$. We will usually use parentheses to explicitly clarify other ambiguities.

Useful Specification Patterns

- Strong invariant: $\Box\psi$
  - ψ always holds
  - A file is always open or closed: $\Box(\mathit{open} \vee \mathit{closed})$
  - Safety property

- Monotone invariant: $\Box(\psi \Rightarrow \Box\psi)$
  - Once ψ is true, then ψ is always true
  - For example, once information is public, it can never become secret again (but it may always stay secret): $\Box(\mathit{public} \Rightarrow \Box\mathit{public})$
  - Safety property

- Establishing an invariant: $\Diamond \Box \psi$
  - Eventually ψ will always hold
  - For example, system initialization starts server: $\Diamond \Box\, \mathit{serverRunning}$
  - Liveness property

Useful Specification Patterns (cont'd)

- Responsiveness: $\Box(\psi \Rightarrow \Diamond\phi)$
  - Every time that ψ holds, ϕ will eventually hold
  - For example, all opened files must be closed eventually: $\Box(\mathit{open} \Rightarrow \Diamond\mathit{closed})$
  - Liveness property
- Fairness: $\Box\Diamond\psi$
  - ψ holds infinitely often
  - For example, producer does not wait infinitely long before entering the critical section: $\Box\Diamond\,\mathit{critical}$
  - Liveness property

Needham-Schroeder Protocol

- If Alice and Bob have completed their protocol runs then Alice should believe her partner to be Bob if and only if Bob believes to talk to Alice

$$\Box(statusA = 1 \land statusB = 1 \Rightarrow (partnerA = agentB \Leftrightarrow partnerB = agentA))$$

- If Alice completed the protocol *talking with Bob*, the intruder will not know Alice's nonce (and dually, swapping the A's and B's):

$$\Box(statusA = 1 \land partnerA = agentB \Rightarrow knows\_nonceA = 0)$$