Chapter 1

Logics

Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Chapter 1

Learning Targets of Chapter “Logics”.

The chapter gives some basic information about “standard” logics, namely propositional logics and (classical) first-order logics.

Chapter 1

Outline of Chapter "Logics".

Section

Introduction

Chapter 1 “Logics”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Logics

What's logic?

General aspects of logics

- truth vs. provability
  - when does a formula *hold*, is *true*, is satisfied
  - valid
  - satisfiable
- syntax vs. semantics/models
- model theory vs. proof theory

Two separate worlds: model theory and proof theory?

proof theory
model theory
calculus

Section

Propositional logic

Chapter 1 “Logics”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Syntax

$$
\begin{array}{rcll}
\varphi & ::= & P \mid \top \mid \bot & \text{atomic formula} \\
& \mid & \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \to \varphi \mid \ldots & \text{formulas}
\end{array}
$$

Semantics

- truth values
- σ
- different "notations"
  - $\sigma \models \varphi$
  - evaluate φ, given σ $[\![\varphi]\!]^{\sigma}$

Proof theory

- decidable, so a "trivial problem" in that sense
- truth tables (brute force)
- one can try to do better, different derivation strategies (resolution, refutation, . . .)
- SAT is NP-complete

Section

Algebraic and first-order signatures

Chapter 1 “Logics”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Signature

- fixes the "syntactic playground"
- selection of
  - *functional* and
  - *relational*

  symbols, together with "arity" or sort-information

Sorts

- Sort
  - name of a domain (like `Nat`)
  - restricted form of type
- single-sorted vs. multi-sorted case
- single-sorted
  - one sort only
  - "degenerated"
  - *arity* = number of arguments (also for relations)

Terms

- given: signature Σ
- set of variables X (with typical elements $x, y', \ldots$)

$$
\begin{array}{rcll}
t & ::= & x & \text{variable} \\
 & | & f(t_1, \ldots, t_n) & f \text{ of arity } n
\end{array}
\tag{1}
$$

- $T_\Sigma(X)$
- terms without variables (from $T_\Sigma(\emptyset)$ or short T_Σ): *ground* terms

Substutition

- Substitution = *replacement*, namely of variables by terms
- notation $t[s/x]$

First-order signature (with relations)

- add relational symbols to Σ
- typical elements P, Q
- relation symbols with fixed arity n-ary predicates or relations)
- standard binary symbol: $\doteq$ (equality)

Section

First-order logic

Syntax
Semantics
Proof theory

Chapter 1 “Logics”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Syntax

- given: first order signature Σ

$$
\begin{array}{lcll}
\varphi & ::= & P(t,\ldots,t) \mid \top \mid \bot & \text{atomic formula} \\
& \mid & \varphi \land \varphi \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \ldots & \text{formulas} \\
& \mid & \forall x.\varphi \mid \exists x.\varphi &
\end{array}
$$

First-order structures and models

- given Σ
- assume single-sorted case

first-order model

model M

$$M = (A, I)$$

- A some domain/set
- interpretation I, respecting arity
  - $[\![f]\!]^I : A^n \to A$
  - $[\![P]\!]^I : A^n$

- cf. first-order structure

Giving meaning to variables

Variable assignment

- given Σ and model

$$\sigma : X \to A$$

- other names: *valuation*, *state*

(E)valuation of terms

- σ "straightforwardly extended/lifted to terms"
- how would one define that (or write it down, or implement)?

Free and bound occurrences of variables

- quantifiers *bind* variables
- *scope*
- other binding, scoping mechanisms
- variables can *occur* free or not (= *bound*) in a formula
- careful with substitution
- how could one define it?

Substitution

- basically:
  - generalize substitution from terms to formulas
  - careful about binders especially don't let substitution lead to variables being "captured" by binders

Example

$$\varphi = \exists x.x + 1 \doteq y \qquad \theta = [y/x]$$

Satisfaction

Definition ($\models$)

$M, \sigma \models \varphi$

- Σ fixed
- in model M and with variable assignment σ formula φ is true (holds
- M and σ satisfy φ
- minority terminology: M, σ model of φ

Exercises

- substitutions and variable assignments: similar/different?
- there are infinitely many primes
- there is a person with at least 2 neighbors (or exactly)
- every even number can be written as the sum of 2 primes

Proof theory

- how to infer, derive, deduce formulas (from others)
- mechanical process
- soundness and completeness
- *proof* = deduction (sequence or tree of steps)
- theorem
  - syntactic: derivable formula
  - semantical a formula which holds (in a given model)
- (fo)-theory: set of formulas which are
  - derivable
  - true (in a given model)
- soundness and completeness

Deductions and proof systems

A proof system for a given logic consists of

- axioms (or *axiom schemata*), which are formulae assumed to be true, and
- inference rules, of approx. the form

$$\frac{\varphi_1 \quad \dots \quad \varphi_n}{\psi}$$

- $\varphi_1, \dots, \varphi_n$ are premises and ψ conclusion.

A simple form of derivation

Derivation of φ

Sequence of formulae, where each formula is

- an axiom or
- can be obtained by applying an inference rule to formulae earlier in the sequence.

- $\vdash \varphi$
- more general: set of formulas Γ

$$\Gamma \vdash \varphi$$

- proof = derivation
- theorem: derivable formula (= last formula in a proof)

Proof systems and proofs: remarks

- "definitions" from the previous slides: not very formal

in general: a proof system: a "mechanical" (= formal and constructive) way of conclusions from axioms (= "given" formulas), and other already proven formulas

- Many different "representations" of how to draw conclusions exists, the one sketched on the previous slide
  - works with "sequences"
  - corresponds to the historically oldest "style" of proof systems ("Hilbert-style"), some would say outdated ...
  - otherwise, in that naive form: impractical (but sound & complete).
  - nowadays, better ways and more suitable for computer support of representation exists (especially using trees). For instance natural deduction style system

A proof system for prop. logic

Observation

We can axiomatize a subset of *propositional logic* as follows.

$$\varphi \to (\psi \to \varphi) \tag{Ax1}$$

$$(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \tag{Ax2}$$

$$((\varphi \to \bot) \to \bot) \to \varphi \tag{DN}$$

$$\frac{\varphi \quad \varphi \to \psi}{\psi} \tag{MP}$$

A proof system

Example

$p \to p$ is a theorem of PPL:

$(p \to ((p \to p) \to p)) \to$ $((p \to (p \to p)) \to (p \to p))$	Ax_2	(1)
$p \to ((p \to p) \to p)$	Ax_1	(2)
$(p \to (p \to p)) \to (p \to p)$	MP on (1) and (2)	(3)
$p \to (p \to p)$	Ax_1	(4)
$p \to p$	MP on (3) and (4)	(5)

Section

Modal logics

Volker Stolz, Martin Steffen
Autumn 2019

Introduction

- Modal logic: logic of *"necessity"* and *"possibility"*, in that originally the intended meaning of the *modal* operators $\Box$ and $\Diamond$ was
  - $\Box\varphi$: φ is necessarily true.
  - $\Diamond\varphi$: φ is possibly true.
- Depending on what we intend to capture: we can interpret $\Box\varphi$ differently.

temporal φ will always hold.

doxastic I believe φ.

epistemic I know φ.

intuitionistic φ is provable.

deontic It ought to be the case that φ.

We will restrict here the modal operators to $\Box$ and $\Diamond$ (and mostly work with a temporal "mind-set".

Kripke structures

Definition (Kripke frame and Kripke model)

- A Kripke frame is a structure (W, R) where
  - W is a non-empty set of *worlds*, and
  - $R \subseteq W \times W$ is called the *accessibility relation* between worlds.

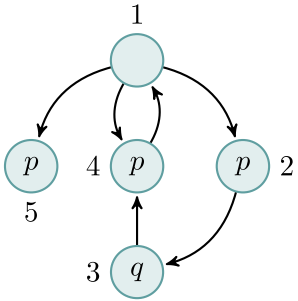
Kripke structures

Definition (Kripke frame and Kripke model)

- A Kripke frame is a structure (W, R) where
  - W is a non-empty set of *worlds*, and
  - $R \subseteq W \times W$ is called the *accessibility relation* between worlds.
- A Kripke model M is a structure (W, R, V) where
  - (W, R) is a frame, and
  - V a function of type $V : W \to (P \to \mathbb{B})$ (called valuation).

isomorphically: $V : W \to 2^P$

Illustration



Example (Kripke model)

Let $P = \{p, q\}$. Then let $M = (W, R, V)$ be the Kripke model such that

- $W = \{w_1, w_2, w_3, w_4, w_5\}$
- $R = \{(w_1, w_5), (w_1, w_4), (w_4, w_1), \ldots\}$
- $V = [w_1 \mapsto \emptyset, w_2 \mapsto \{p\}, w_3 \mapsto \{q\}, \ldots]$

Satisfaction

Definition (Satisfaction)

A modal formula φ is true in the world w of a model V, written $V, w \models \varphi$, if:

$$
\begin{array}{lll}
V, w \models p & \text{iff} & V(w)(p) = \top \\
\\
V, w \models \neg\varphi & \text{iff} & V, w \not\models \varphi \\
V, w \models \varphi_1 \vee \varphi_2 & \text{iff} & V, w \models \varphi_1 \text{ or } V, w \models \varphi_2 \\
\\
V, w \models \Box\varphi & \text{iff} & V, w' \models \varphi, \text{ for all } w' \text{ such that } wRw' \\
V, w \models \Diamond\varphi & \text{iff} & V, w' \models \varphi, \text{ for some } w' \text{ such that } wRw'
\end{array}
$$

“Box” and “diamond”

- modal operators $\Box$ and $\Diamond$
- often pronounced “nessecarily” and “possibly”
- mental picture: depends on “kind” of logic (temporal, epistemic, deontic . . .) and (related to that) the form of accessibility relation R
- formal definition: see previous slide

Different kinds of relations

R a *binary relation* on a set, say W, i.e., $R \subseteq W$

- reflexive
- transitive
- (right) Euclidian
- total
- order relation
- ....

Valid in frame/for a set of frames

If $(W, R, V), s \models \varphi$ for all s and V, we write

$$(W, R) \models \varphi$$

Example (Samples)

- $(W, R) \models \Box\varphi \to \varphi$ iff R is reflexive.
- $(W, R) \models \Box\varphi \to \Diamond\varphi$ iff R is total.
- $(W, R) \models \Box\varphi \to \Box\Box\varphi$ iff R is transitive.
- $(W, R) \models \neg\Box\varphi \to \Box\neg\Box\varphi$ iff R is Euclidean.

Some exercises

Prove the double implications from the slide before!

Base line axiomatic system (“K”)

$$\frac{\varphi \text{ is a propositional tautology}}{\varphi}\ \mathrm{PL}$$

$$\frac{}{\Box(\varphi_1 \to \varphi_2) \to (\Box\varphi_1 \to \Box\varphi_2)}\ \mathrm{K}$$

$$\frac{\varphi \to \psi \qquad \varphi}{\psi}\ \mathrm{MP}$$

$$\frac{\varphi}{\Box\varphi}\ \mathrm{G}$$

Sample axioms for different accessibility relations

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi) \tag{K}$$

$$\Box\varphi \to \Diamond\varphi \tag{D}$$

$$\Box\varphi \to \varphi \tag{T}$$

$$\Box\varphi \to \Box\Box\varphi \tag{4}$$

$$\neg\Box\varphi \to \Box\neg\Box\varphi \tag{5}$$

$$\Box(\Box\varphi \to \psi) \to \Box(\Box\psi \to \varphi) \tag{3}$$

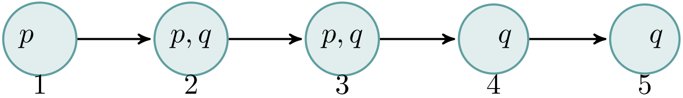
$$\Box(\Box(\varphi \to \Box\varphi) \to \varphi) \to (\Diamond\Box\varphi \to \varphi)) \tag{Dum}$$

Different "flavors" of modal logic

Logic	Axioms	Interpretation	Properties of R
D	K D	deontic	total
T	K T		reflexive
K45	K 4 5	doxastic	transitive/euclidean
S4	K T 4		reflexive/transitive
S5	K T 5	epistemic	reflexive/euclidean
			reflexive/symmetric/transitive
			equivalence relation

Some exercises

Consider the frame (W, R) with $W = \{1, 2, 3, 4, 5\}$ and $(i, i+1) \in R$

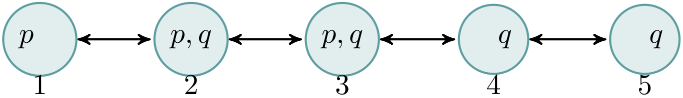


- $M, 1 \models \Diamond\Box p$
- $M, 1 \models \Diamond\Box p \rightarrow p$
- $M, 3 \models \Diamond(q \wedge \neg p) \wedge \Box(q \wedge \neg p)$
- $M, 1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$
- $M \models \Box q$

Exercises (2): bidirectional frames

Bidirectional frame

A frame (W, R) is bidirectional iff $R = R_F + R_P$ s.t. $\forall w, w'(wR_Fw' \leftrightarrow w'R_Pw)$.



Consider $M = (W, R, V)$ from before. Which of the following statements are correct in M and why?

1. $M, 1 \models \Diamond\Box p$
2. $M, 1 \models \Diamond\Box p \to p$
3. $M, 3 \models \Diamond(q \land \neg p) \land \Box(q \land \neg p)$
4. $M, 1 \models q \land \Diamond(q \land \Diamond(q \land \Diamond(q \land \Diamond q)))$
5. $M \models \Box q$
6. $M \models \Box q \to \Diamond\Diamond p$

Exercises (3): validities

Which of the following are *valid* in modal logic. For those that are not, argue why and find a class of frames on which they become valid.

1. $\Box\bot$
2. $\Diamond p \rightarrow \Box p$
3. $p \rightarrow \Box\Diamond p$
4. $\Diamond\Box p \rightarrow \Box\Diamond p$

Section

Dynamic logics

Multi-modal logic
Dynamic logics
Semantics of PDL

Chapter 1 “Logics”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Introduction

Problem

- FOL: "very" expressive but *undecidable*. Perhaps good for mathematics but not ideal for computers.

!! FOL can talk about the state of the system. But how to talk about *change of state* in a *natural* way?

- modal logic: gives us the power to talk about *changing of state*. Modal logics is natural when one is interested in systems that are essentially modeled as states and transitions between states.

Multi-modal logic

"Kripke frame" (W, R_a, R_b), where R_a and R_b are two relations over W.

Multi-modal logic

"Kripke frame" (W, R_a, R_b), where R_a and R_b are two relations over W.

Syntax (2 relations)

Multi-modal logic has one modality for each relation:

$$\varphi ::= p \mid \bot \mid \varphi \to \varphi \mid \Diamond_a \varphi \mid \Diamond_b \varphi \tag{6}$$

Semantics: "natural" generalization of the "mono"-case

$$M, w \models \Diamond_a \varphi \text{ iff } \exists w' : w R_a w' \text{ and } M, w' \models \varphi \tag{7}$$

- analogously for modality $\Diamond_b$ and relation R_b

Remarks

As *multi*-modal logic: *obvious generalization* of modal logic from before

1. The relations can overlap; i.e., their intersection need not be empty
2. of course: more than 2 relations possible, for each relation one modality.
3. There may be *infinitely* many relations and infinitely many modalities.

Dynamic logics

- different variants
- can be seen as special case of multi-modal logics
- variant of Hoare-logics
- here: PDL on regular programs
- "P" stands for "propositional"

Regular programs

DL

Dynamic logic is a multi-modal logic to talk about programs.

here: dynamic logic talks about regular programs

Regular programs

DL

Dynamic logic is a multi-modal logic to talk about programs.

here: dynamic logic talks about regular programs
Regular programs are formed syntactically from:

- *atomic* programs $\Pi_0 = \{a, b, ...\}$, which are indivisible, single-step, basic programming constructs
- *sequential* composition $\alpha \cdot \beta$, which means that program α is executed/done first and then β.
- *nondeterministic choice* $\alpha + \beta$, which nondeterministically chooses one of α and β and executes it.
- *iteration* α^*, which executes α some nondeterministically chosen finite number of times.
- the special `skip` and `fail` programs (denoted $\mathbf{1}$ resp. $\mathbf{0}$

Regular programs and tests

Definition (Regular programs)

The syntax of regular programs $\alpha, \beta \in \Pi$ is given according to the grammar:

$$\alpha ::= a \in \Pi_0 \mid \mathbf{1} \mid \mathbf{0} \mid \alpha \cdot \alpha \mid \alpha + \alpha \mid \alpha^* \mid \varphi? \,. \quad (8)$$

The clause $\varphi?$ is called *test*.

Tests can be seen as special atomic programs which may have logical structure, but their execution terminates in the same state iff the test succeeds (is true), otherwise fails if the test is deemed false in the current state.

Tests

- *simple* Boolean tests:
  $\varphi ::= \top \mid \bot \mid \varphi \rightarrow \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$
- *complex* tests: $\varphi?$ where φ is a logical formula in *dynamic logic*

Propositional Dynamic Logic: Syntax

Definition (DPL syntax)

The formulas φ of *propositional dynamic logic* (PDL) over regular programs α are given as follows.

$$\begin{array}{lcl} \alpha & ::= & a \in \Pi_0 \mid \mathbf{1} \mid \mathbf{0} \mid \alpha \cdot \alpha \mid \alpha + \alpha \mid \alpha^* \mid \varphi? \\ \varphi & ::= & p, q \in \Phi_0 \mid \top \mid \bot \mid \varphi \rightarrow \varphi \mid [\alpha]\varphi \end{array} \tag{9}$$

where Φ_0 is a set of atomic propositions.

1. programs, which we denote $\alpha ... \in \Pi$
2. formulas, which we denote $\varphi ... \in \Phi$

Propositional Dynamic Logic (PDL): because based on propositional logic, only

PDL: remarks

- Programs α interpreted as a relation R_α

$\Rightarrow$ multi-modal logic.

- $[\alpha]\varphi$ defines many modalities, one modality for each program, each interpreted over the relation defined by the program α.
- The relations of the basic programs are just given.
- Operations on/composition of programs are interpreted as operations on relations.
- ∞ many complex programs $\Rightarrow$ ∞ many relations/modalities
- But we think of a single modality $[..]\varphi$ with programs inside.
- $[..]\varphi$ is the universal one, with $\langle..\rangle\varphi$ defined as usual.

Intiutive meaning/semantics of $[\alpha]\varphi$

"If program α is started in the current state, then, *if* it terminates, then in its final state, φ holds."

Exercises: "programs"

Define the following programming constructs in PDL:

$$
\begin{aligned}
\textbf{skip} &\quad \triangleq \\
\textbf{fail} &\quad \triangleq \\
\textbf{if } \varphi \textbf{ then } \alpha \textbf{ else } \beta &\quad \triangleq \\
\textbf{if } \varphi \textbf{ then } \alpha &\quad \triangleq \\
\textbf{case } \varphi_1 \textbf{ then } \alpha_1;\ \ldots\ &\quad \triangleq \\
\textbf{case } \varphi_n \textbf{ then } \alpha_n & \\
\textbf{while } \varphi \textbf{ do } \alpha &\quad \triangleq \\
\textbf{repeat } \alpha \textbf{ until } \varphi &\quad \triangleq
\end{aligned}
$$

*(General **while** loop)*

$$\textbf{while } \varphi_1 \textbf{ then } \alpha_1 \mid \cdots \mid \varphi_n \textbf{ then } \alpha_n \textbf{ od} \quad \triangleq$$

Exercises: "programs"

Define the following programming constructs in PDL:

$$
\begin{array}{rcl}
\textbf{skip} & \triangleq & \top? \\
\textbf{fail} & \triangleq & \bot? \\
\textbf{if } \varphi \textbf{ then } \alpha \textbf{ else } \beta & \triangleq & (\varphi? \cdot \alpha) + (\neg\varphi? \cdot \beta) \\
\textbf{if } \varphi \textbf{ then } \alpha & \triangleq & (\varphi? \cdot \alpha) + (\neg\varphi? \cdot \textbf{skip}) \\
\textbf{case } \varphi_1 \textbf{ then } \alpha_1;\ \ldots & \triangleq & (\varphi_1? \cdot \alpha_1) + \ldots + (\varphi_n? \cdot \alpha_n) \\
\textbf{case } \varphi_n \textbf{ then } \alpha_n & & \\
\textbf{while } \varphi \textbf{ do } \alpha & \triangleq & (\varphi? \cdot \alpha)^* \cdot \neg\varphi? \\
\textbf{repeat } \alpha \textbf{ until } \varphi & \triangleq & \alpha \cdot (\neg\varphi? \cdot \alpha)^* \cdot \varphi?
\end{array}
$$

*(General **while** loop)*

$$
\begin{array}{rcl}
\textbf{while } \varphi_1 \textbf{ then } \alpha_1 \mid \cdots \mid \varphi_n \textbf{ then } \alpha_n \textbf{ od} & \triangleq & (\varphi_1? \cdot \alpha_1 + \ldots + \varphi_n? \cdot \alpha_n)^* \cdot \\
& & \cdot(\neg\varphi_1 \wedge \ldots \neg \wedge \varphi_n)?
\end{array}
$$

Making Kripke structures "multi-modal-prepared"

Definition (Labeled Kripke structures)

Assume a set of labels Σ. A *labeled Kripke structure* is a tuple (W, R, Σ) where

$$R = \bigcup_{l \in \Sigma} R_l$$

is the disjoint union of the relations indexed by the labels of Σ.

for us (at leat now): The labels of Σ can be thought as programs

- Σ: aka alphabet,
- alternative: $R \subseteq W \times \Sigma \times W$
- labels $l, l_1 \ldots$ but also $a, b, \ldots$ or others
- often: $\xrightarrow{a}$, like $w_1 \xrightarrow{a} w_2$ or $s_1 \xrightarrow{a} s_2$

Regular Kripke structures

- "labels" now have "strucuture"
- remember regular program syntax
- interpretation of certain programs/labels fixed,
  - $\mathbf{0}$: failing program
  - $\alpha_1 \cdot \alpha_2$: sequential composition
  - ...
- thus, relations like $\mathbf{0}$, $R_{\alpha_1 \cdot \alpha_2}$, ... must obey side-conditions

Basically

leaving open the interpretation of the "atoms" a, we fix the interpretation/semantics of the constructs of regular programs

Regular Kripke structures

Definition (Regular Kripke structures)

A regular Kripke structure is a Kripke structure labeled as follows. For all basic programs $a \in \Pi_0$, choose some relation R_a. For the remaining syntactic constructs (except tests), the corresponding relations are defined inductively as follows.

$$
\begin{array}{lcl}
R_{\mathbf{1}} & = & Id \\
R_{\mathbf{0}} & = & \emptyset \\
R_{\alpha_1 \cdot \alpha_2} & = & R_{\alpha_1} \circ R_{\alpha_2} \\
R_{\alpha_1 + \alpha_2} & = & R_{\alpha_1} \cup R_{\alpha_2} \\
R_{\alpha^*} & = & \bigcup_{n \geq 0} R_\alpha^n
\end{array}
$$

Kripke *models* and interpreting PDL formulas

Now: add *valutions* $\Rightarrow$ Kripke model

Definition (Semantics)

A PDL formula φ is true in the world w of a regular Kripke model M, i.e., we have attached a valuation V also, written $M, w \models \varphi$, if:

$M, w \models p_i$	iff	$p_i \in V(w)$ for all propositional constants
$M, w \not\models \bot$	and	$M, w \models \top$
$M, w \models \varphi_1 \to \varphi_2$	iff	whenever $M, w \models \varphi_1$ then also $M, w \models \varphi_2$
$M, w \models [\alpha]\varphi$	iff	$M, w' \models \varphi$ for all w' such that $wR_\alpha w'$
$M, w \models \langle\alpha\rangle\varphi$	iff	$M, w' \models \varphi$ for some w' such that $wR_\alpha w'$

Semantics (cont'd)

- programs and formulas: mutually dependent
- omitted so far: what relationship corresponds to

$$\varphi?$$

- remember the intuitive meaning (semantics) of tests

Test programs

Intuition: tests interpreted as subsets of the identity relation.

$$R_{\varphi?} = \{(w, w) \mid w \models \varphi\} \subseteq I \tag{10}$$

More precisely:

- for $\top?$ the relation becomes $R_{\top?} = Id$
  (testing $\top$ succeeds everywhere and is as the **skip** program)
- for $\bot?$ the relation becomes $R_{\bot?} = \emptyset$
  ($\bot$ is nowhere true and is as the **fail** program)
- $R_{(\varphi_1 \wedge \varphi_2)?} = \{(w, w) \mid w \models \varphi_1 \text{ and } w \models \varphi_2\}$
- Testing a complex formula involving $[\alpha]\varphi$ is like looking into the future of the program and then deciding on the action to take...

Axiomatic System of PDL

Take all tautologies of propositional logic (i.e., the axiom system of PL from Lecture 2) and add
Axioms:

$$[\alpha](\phi_1 \to \phi_2) \to ([\alpha]\phi_1 \to [\alpha]\phi_2) \tag{1}$$

$$[\alpha](\phi_1 \wedge \phi_2) \leftrightarrow [\alpha]\phi_1 \wedge [\alpha]\phi_2 \tag{2}$$

$$[\alpha + \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi \tag{3}$$

$$[\alpha \cdot \beta]\phi \leftrightarrow [\alpha][\beta]\phi \tag{4}$$

$$[\phi?]\psi \leftrightarrow \phi \to \psi \tag{5}$$

$$\phi \wedge [\alpha][\alpha^*]\phi \leftrightarrow [\alpha^*]\phi \tag{6}$$

$$\phi \wedge [\alpha^*](\phi \to [\alpha]\phi) \to [\alpha^*]\phi \tag{IND}$$

Rules: take the (MP) modus ponens and (G) generalization of Modal Logic.

Chapter 2

LTL model checking

Course "Model checking"

Volker Stolz, Martin Steffen

Autumn 2019

Chapter 2

Learning Targets of Chapter "LTL model checking".

The chapter covers LTL and how to do model checking for that logic, using Büchi-automata.

Chapter 2

Outline of Chapter "LTL model checking".

Section

Introduction

Chapter 2 “LTL model checking”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Temporal logic?

- Temporal logic: is the/a logic of "time"
- *modal* logic.
- different ways of modeling time.
  - linear vs. branching time
  - time instances vs. time intervals
  - discrete time vs. continuous time
  - past and future vs. future only
  - ...

LTL

- linear time temporal logic
- one central temporal logic in CS
- supported by Spinand other model checkers
- many variations

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.
  - $\langle x : 21, y : 49 \rangle \models x < y$

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.
  - $\langle x : 21, y : 49 \rangle \models x < y$
  - $\langle x : 21, y : 7 \rangle \not\models x < y$

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.
  - $\langle x : 21, y : 49 \rangle \models x < y$
  - $\langle x : 21, y : 7 \rangle \not\models x < y$
- A computation is a sequence of states.

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.
  - $\langle x : 21, y : 49 \rangle \models x < y$
  - $\langle x : 21, y : 7 \rangle \not\models x < y$
- A computation is a sequence of states.
- To express properties of computations, we need to extend FOL.

FOL (repetition)

First Order Logic

- We have used FOL to express properties of states.
  - $\langle x : 21, y : 49 \rangle \models x < y$
  - $\langle x : 21, y : 7 \rangle \not\models x < y$
- A computation is a sequence of states.
- To express properties of computations, we need to extend FOL.
- This we can do using temporal logic.

Section

LTL

LTL: speaking about "time"

In Linear Temporal Logic (LTL), also called linear-time temporal logic, we can describe such properties as, for instance, the following: assume time is a *sequence* of discrete points i in time, then: if i is *now*,

- p holds in i and every following point (the future)
- p holds in i and every preceding point (the past)

$$\ldots \longrightarrow \bullet^{p}_{i-2} \longrightarrow \bullet^{p}_{i-1} \longrightarrow \bullet^{p}_{i} \longrightarrow \bullet^{p}_{i+1} \longrightarrow \bullet^{p}_{i+2} \longrightarrow \ldots$$

Syntax

ψ			propositional/first-order formula
φ	$::=$	ψ	formulas of the "core" logics
	$\mid$	$\neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \ldots$	boolean combinations
	$\mid$	$\bigcirc\varphi$	next φ
	$\mid$	$\Box\varphi$	always φ
	$\mid$	$\Diamond\varphi$	eventually φ
	$\mid$	$\varphi\ U\ \varphi$	"until"
	$\mid$	$\varphi\ R\ \varphi$	"release"
	$\mid$	$\varphi\ W\ \varphi$	"waiting for", "weak until"

Paths and computations

Definition (Path)

- A path is an infinite sequence

$$\sigma = s_0, s_1, s_2, \ldots$$

of states.

Paths and computations

Definition (Path)

- A path is an infinite sequence

$$\sigma = s_0, s_1, s_2, \ldots$$

  of states.
- σ^k denotes the *path* $s_k, s_{k+1}, s_{k+2}, \ldots$

Paths and computations

Definition (Path)

- A path is an infinite sequence

$$\sigma = s_0, s_1, s_2, \ldots$$

  of states.
- σ^k denotes the *path* $s_k, s_{k+1}, s_{k+2}, \ldots$
- σ_k denotes the *state* s_k.

Satisfaction (semantics)

Definition

An LTL formula φ is true relative to a path σ, written $\sigma \models \varphi$, as follows.

$$
\begin{array}{lll}
\sigma \models \psi & \text{iff} & \sigma_0 \models_{\mathsf{ul}} \varphi \text{ where } \psi \text{ in underlying core language} \\
\sigma \models \neg\varphi & \text{iff} & \sigma \not\models \varphi \\
\sigma \models \varphi_1 \vee \varphi_2 & \text{iff} & \sigma \models \varphi_1 \text{ or } \sigma \models \varphi_2 \\
\\
\sigma \models \Box\varphi & \text{iff} & \sigma^k \models \varphi \text{ for all } k \geq 0 \\
\sigma \models \Diamond\varphi & \text{iff} & \sigma^k \models \varphi \text{ for some } k \geq 0 \\
\sigma \models \bigcirc\varphi & \text{iff} & \sigma^1 \models \varphi
\end{array}
$$

(cont.)

Satisfaction (semantics) (2)

Definition

(cont.)

$$\sigma \models \varphi_1 \; U \; \varphi_2 \quad \text{iff} \quad \sigma^k \models \varphi_2 \text{ for some } k \geq 0\text{, and}$$
$$\sigma^i \models \varphi_2 \text{ for every } i \text{ such that } 0 \leq i < k$$

$$\sigma \models \varphi_1 \; R \; \varphi_2 \quad \text{iff} \quad \text{for every } j \geq 0,$$
$$\text{if } \sigma^i \not\models \varphi_1 \text{ for every } i < j \text{ then } \sigma^j \models \varphi_2$$

$$\sigma \models \varphi_1 \; W \; \varphi_2 \quad \text{iff} \quad \sigma \models \varphi_1 \; U \; \varphi_2 \text{ or } \sigma \models \Box\varphi_1$$

Validity and semantic equivalence

Definition

- We say that φ is (temporally) valid, written $\models \varphi$, if

$$\sigma \models \varphi \text{ for all paths } \sigma.$$

Validity and semantic equivalence

Definition

- We say that φ is (temporally) valid, written $\models \varphi$, if
  $$\sigma \models \varphi \text{ for all paths } \sigma.$$
- We say that φ and ψ are equivalent, written $\varphi \sim \psi$, if
  $$\models \varphi \leftrightarrow \psi \text{ (i.e. } \sigma \models \varphi \text{ iff } \sigma \models \psi, \text{ for all } \sigma).$$

Validity and semantic equivalence

Definition

- We say that φ is (temporally) valid, written $\models \varphi$, if
  $$\sigma \models \varphi \text{ for all paths } \sigma.$$
- We say that φ and ψ are equivalent, written $\varphi \sim \psi$, if
  $$\models \varphi \leftrightarrow \psi \text{ (i.e. } \sigma \models \varphi \text{ iff } \sigma \models \psi\text{, for all } \sigma\text{)}.$$

Example

$\Box$ distributes over $\wedge$, while $\Diamond$ distributes over $\vee$.

$$\Box(\varphi \wedge \psi) \sim (\Box\varphi \wedge \Box\psi)$$
$$\Diamond(\varphi \vee \psi) \sim (\Diamond\varphi \vee \Diamond\psi)$$

Semantics

$$\sigma \models \Box p$$

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^p \longrightarrow \bullet_4^p \longrightarrow \cdots$$

$$\sigma \models \Diamond p$$

$$\bullet_0 \longrightarrow \bullet_1 \longrightarrow \bullet_2 \longrightarrow \bullet_3^p \longrightarrow \bullet_4 \longrightarrow \cdots$$

$$\sigma \models \bigcirc p$$

$$\bullet_0 \longrightarrow \bullet_1^p \longrightarrow \bullet_2 \longrightarrow \bullet_3 \longrightarrow \bullet_4 \longrightarrow \cdots$$

$\sigma \models p \ U \ q$ (sequence of p's is finite)

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^q \longrightarrow \bullet_4 \longrightarrow \ldots$$

$\sigma \models p \ R \ q$ (The sequence of qs may be infinite)

$$\bullet_0^q \longrightarrow \bullet_1^q \longrightarrow \bullet_2^q \longrightarrow \bullet_3^{p,q} \longrightarrow \bullet_4 \longrightarrow \ldots$$

$\sigma \models p \ W \ q$. The sequence of ps may be infinite.
$(p \ W \ q \sim p \ U \ q \vee \Box p)$.

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^p \longrightarrow \bullet_4^p \longrightarrow \ldots$$

The past

Observation

- [1] uses pairs (σ, j) of paths and positions instead of just the path σ because they have past-formulae: formulae without future operators (the ones we use) but possibly with past operators, like $\Box^{-1}$ and $\Diamond^{-1}$.

$$
\begin{array}{lll}
(\sigma, j) \models \Box^{-1}\varphi & \text{iff} & (\sigma, k) \models \varphi \text{ for all } k,\ 0 \leq k \leq j \\
(\sigma, j) \models \Diamond^{-1}\varphi & \text{iff} & (\sigma, k) \models \varphi \text{ for some } k,\ 0 \leq k \leq j
\end{array}
$$

The past

Observation

- [1] uses pairs (σ, j) of paths and positions instead of just the path σ because they have past-formulae: formulae without future operators (the ones we use) but possibly with past operators, like $\Box^{-1}$ and $\Diamond^{-1}$.

$$(\sigma, j) \models \Box^{-1}\varphi \quad \text{iff} \quad (\sigma, k) \models \varphi \text{ for all } k,\ 0 \leq k \leq j$$

$$(\sigma, j) \models \Diamond^{-1}\varphi \quad \text{iff} \quad (\sigma, k) \models \varphi \text{ for some } k,\ 0 \leq k \leq j$$

- However, it can be shown that for any formula φ, there is a future-formula (formulae without past operators) ψ such that

$$(\sigma, 0) \models \varphi \quad \text{iff} \quad (\sigma, 0) \models \psi$$

The past: examples

Example

What is a future version of $\Box(p \to \Diamond^{-1} q)$?

$(\sigma, 0) \models \Box(p \to \Diamond^{-1} q)$

$$\bullet^{p\to\Diamond^{-1}q} \longrightarrow \bullet^{p\to\Diamond^{-1}q} \longrightarrow \bullet^{p\to\Diamond^{-1}q} \longrightarrow \bullet^{p\to\Diamond^{-1}q} \longrightarrow \bullet \longrightarrow \ldots$$

$(\sigma, 0) \models q \; R \; (p \to q)$

$$\bullet^{p\to q} \longrightarrow \bullet^{p\to q} \longrightarrow \bullet^{p\to q, q} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \ldots$$

Examples

Example

$\varphi \rightarrow \Diamond\psi$: If φ holds initially, then ψ holds eventually.

$$\bullet^{\varphi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet^{\psi} \longrightarrow \bullet \longrightarrow \dots$$

This formula will also hold in every path where φ does not hold initially.

$$\bullet^{\neg\varphi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \dots$$

Example: Response

Example (Response)

$\Box(\varphi \rightarrow \Diamond\psi)$

Every φ-position coincides with or is followed by a ψ-position.

$$\bullet \longrightarrow \bullet^{\varphi} \longrightarrow \bullet \longrightarrow \bullet^{\psi} \longrightarrow \bullet \longrightarrow \bullet^{\varphi,\psi} \longrightarrow \dots$$

This formula will also hold in every path where φ never holds.

$$\bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \dots$$

Examples

Example

$\Box\Diamond\psi$

There are infinitely many ψ-positions.

$$\bullet^{\psi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet^{\psi} \longrightarrow \bullet \longrightarrow \bullet^{\psi} \longrightarrow \bullet \longrightarrow \dots$$

This formula can be obtained from the previous one, $\Box(\varphi \to \Diamond\psi)$, by letting $\varphi = \top$: $\Box(\top \to \Diamond\psi)$.

Example: permanence

Example

$\Diamond\Box\varphi$

Eventually φ will hold permanently.

$$\bullet \longrightarrow \bullet^{\varphi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet^{\varphi} \longrightarrow \bullet^{\varphi} \longrightarrow \bullet^{\varphi} \longrightarrow \dots$$

Equivalently: there are finitely many $\neg\varphi$-positions.

LTL example

Example

$(\neg\varphi)\ W\ \psi$

[WRONG SENTENCE] The first φ-position must coincide or be preceded by a ψ-position.

$$\bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\varphi} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$$

φ may never hold

$$\bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \bullet^{\neg\varphi} \longrightarrow \cdots$$

LTL Example

Example

$\Box(\varphi \to \psi \ W \ \chi)$

Every φ-position initiates a sequence of ψ-positions, and if terminated, by a χ-position.

$$\bullet \longrightarrow \bullet^{\varphi,\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\chi} \longrightarrow \bullet \longrightarrow \bullet^{\varphi,\psi} \longrightarrow \cdots$$

The sequence of ψ-positions need not terminate.

$$\bullet \longrightarrow \bullet^{\varphi,\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\psi} \longrightarrow \bullet^{\psi} \longrightarrow \cdots$$

Nested waiting-for

A nested waiting-for formula is of the form

$$\Box(\varphi \to (\psi_m \; W \; (\psi_{m-1} \; W \cdots (\psi_1 \; W \; \psi_0) \cdots))),$$

where $\varphi, \psi_0, \ldots, \psi_m$ in the underlying logic. For convenience, we write

$$\Box(\varphi \to \psi_m \, W \, \psi_{m-1} \, W \; \cdots \; W \, \psi_1 \, W \, \psi_0).$$

Every φ-position initiates a succession of intervals, beginning with a ψ_m-interval, ending with a ψ_1-interval and possibly terminated by a ψ_0-position. Each interval may be empty or extend to infinity.

$$\ldots \longrightarrow \bullet^{\varphi,\psi_m} \longrightarrow \bullet^{\psi_m} \dashrightarrow \bullet^{\psi_m} \longrightarrow \bullet^{\psi_{m-1}} \cdots\cdots \bullet^{\psi_{m-1}} \longrightarrow \ldots$$

$$\ldots \longrightarrow \bullet^{\psi_2} \cdots\cdots \bullet^{\psi_2} \longrightarrow \bullet^{\psi_1} \cdots\cdots \bullet^{\psi_1} \longrightarrow \bullet^{\psi_0} \longrightarrow \ldots$$

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$?

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.
- $\Box(\varphi \rightarrow \psi)$?

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \to \psi$? $\varphi \to \psi$ holds in the initial state.
- $\square(\varphi \to \psi)$? $\varphi \to \psi$ holds in every state.

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.
- $\Box(\varphi \rightarrow \psi)$? $\varphi \rightarrow \psi$ holds in every state.
- $\varphi \rightarrow \Diamond\psi$?

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.
- $\Box(\varphi \rightarrow \psi)$? $\varphi \rightarrow \psi$ holds in every state.
- $\varphi \rightarrow \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \to \psi$? $\varphi \to \psi$ holds in the initial state.
- $\Box(\varphi \to \psi)$? $\varphi \to \psi$ holds in every state.
- $\varphi \to \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.
- $\Box(\varphi \to \Diamond\psi)$?

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.
- $\Box(\varphi \rightarrow \psi)$? $\varphi \rightarrow \psi$ holds in every state.
- $\varphi \rightarrow \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.
- $\Box(\varphi \rightarrow \Diamond\psi)$? We saw this earlier.

Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement "φ implies ψ"?

- $\varphi \to \psi$? $\varphi \to \psi$ holds in the initial state.
- $\Box(\varphi \to \psi)$? $\varphi \to \psi$ holds in every state.
- $\varphi \to \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.
- $\Box(\varphi \to \Diamond\psi)$? We saw this earlier.
- None of these is necessarily what we intended

Duals

Definition (Duals)

For binary boolean connectives[1] $\circ$ and $\bullet$, we say that $\bullet$ is the dual of $\circ$ if

$$\neg(\varphi \circ \psi) \sim (\neg\varphi \bullet \neg\psi).$$

Similarly for unary connectives: $\bullet$ is the dual of $\circ$ if $\neg \circ \varphi \sim \bullet\neg\varphi$.

Duality is symmetric:

- If $\bullet$ is the dual of $\circ$ then
- $\circ$ is the dual of $\bullet$, thus
- we may refer to two connectives as dual (of each other).

[1] Those are not concrete connectives or operators, they are meant as "placeholders"

Dual connectives

Which connectives are duals?

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

- What is the dual of $\rightarrow$?

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

- What is the dual of $\rightarrow$? It's $\not\leftarrow$:

$$\neg(\varphi \not\leftarrow \psi)$$

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

- What is the dual of $\rightarrow$? It's $\not\leftarrow$:

$$\neg(\varphi \not\leftarrow \psi) \sim \varphi \leftarrow \psi$$

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

- What is the dual of $\rightarrow$? It's $\not\leftarrow$:

$$\begin{aligned}\neg(\varphi \not\leftarrow \psi) &\sim \varphi \leftarrow \psi \\ &\sim \psi \rightarrow \varphi\end{aligned}$$

Dual connectives

Which connectives are duals?

- $\wedge$ and $\vee$ are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- $\neg$ is its own dual:

$$\neg\neg\varphi \sim \neg\neg\varphi.$$

- What is the dual of $\rightarrow$? It's $\not\leftarrow$:

$$\begin{aligned}\neg(\varphi \not\leftarrow \psi) &\sim \varphi \leftarrow \psi \\ &\sim \psi \rightarrow \varphi \\ &\sim \neg\varphi \rightarrow \neg\psi\end{aligned}$$

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \leftrightarrow \psi$ is equivalent to $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \leftrightarrow \psi$ is equivalent to $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.
- $\top$ is equivalent to $p \vee \neg p$

Complete sets of connectives

- A set of connectives is complete (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \leftrightarrow \psi$ is equivalent to $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.
- $\top$ is equivalent to $p \vee \neg p$
- $\bot$ is equivalent to $p \wedge \neg p$

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of $\Box$?

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of $\Box$? And of $\Diamond$?

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of $\Box$? And of $\Diamond$?
- $\Box$ and $\Diamond$ are duals.

$$\neg\Box\varphi \sim \Diamond\neg\varphi$$
$$\neg\Diamond\varphi \sim \Box\neg\varphi$$

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of $\Box$? And of $\Diamond$?
- $\Box$ and $\Diamond$ are duals.

$$\neg\Box\varphi \sim \Diamond\neg\varphi$$
$$\neg\Diamond\varphi \sim \Box\neg\varphi$$

- Any other?

Duals in LTL

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of $\Box$? And of $\Diamond$?
- $\Box$ and $\Diamond$ are duals.

$$\neg\Box\varphi \sim \Diamond\neg\varphi$$
$$\neg\Diamond\varphi \sim \Box\neg\varphi$$

- Any other?
- U and R are duals.

$$\neg(\varphi\ U\ \psi) \sim (\neg\varphi)\ R\ (\neg\psi)$$
$$\neg(\varphi\ R\ \psi) \sim (\neg\varphi)\ U\ (\neg\psi)$$

Complete set of LTL operators

We don't need all our temporal operators either.

Complete set of LTL operators

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ *is complete for LTL.*

Complete set of LTL operators

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ *is complete for LTL.*

Proof.

- $\Diamond\varphi \sim \top\ U\ \varphi$

Complete set of LTL operators

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ *is complete for LTL.*

Proof.

- $\Diamond\varphi \sim \top\ U\ \varphi$
- $\Box\varphi \sim \bot\ R\ \varphi$

Complete set of LTL operators

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ *is complete for LTL.*

Proof.

- $\Diamond\varphi \sim \top\ U\ \varphi$
- $\Box\varphi \sim \bot\ R\ \varphi$
- $\varphi\ R\ \psi \sim \neg(\neg\varphi\ U\ \neg\psi)$

Complete set of LTL operators

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ *is complete for LTL.*

Proof.

- $\Diamond\varphi \sim \top \; U \; \varphi$
- $\Box\varphi \sim \bot \; R \; \varphi$
- $\varphi \; R \; \psi \sim \neg(\neg\varphi \; U \; \neg\psi)$
- $\varphi \; W \; \psi \sim \Box\varphi \vee (\varphi \; U \; \psi)$ $\square$

Classification of properties

We can classify properties expressible in LTL.

Classification

safety	$\Box\varphi$
liveness	$\Diamond\varphi$
obligation	$\Box\varphi \lor \Diamond\psi$
recurrence	$\Box\Diamond\varphi$
persistence	$\Diamond\Box\varphi$
reactivity	$\Box\Diamond\varphi \lor \Diamond\Box\psi$

Safety

- important basic class of properties
- relation to testing and run-time verification
- "nothing bad ever happens"

Definition (Safety)

- A safety formula is of the form

$$\Box\varphi$$

for some first-order/prop. formula φ.

Safety

- important basic class of properties
- relation to testing and run-time verification
- "nothing bad ever happens"

Definition (Safety)

- A safety formula is of the form

$$\Box\varphi$$

for some first-order/prop. formula φ.

- A conditional safety formula is of the form

$$\varphi \rightarrow \Box\psi$$

for (first-order) formulae φ and ψ.

Safety

- important basic class of properties
- relation to testing and run-time verification
- "nothing bad ever happens"

Definition (Safety)

- A safety formula is of the form

$$\Box\varphi$$

  for some first-order/prop. formula φ.
- A conditional safety formula is of the form

$$\varphi \rightarrow \Box\psi$$

  for (first-order) formulae φ and ψ.
- Safety formulae express *invariance* of some state property φ: that φ holds in every state of the computation.

Safety property example

Example

- *Mutual exclusion* is a safety property. Let C_i denote that process P_i is executing in the critical section. Then

$$\Box\neg(C_1 \wedge C_2)$$

expresses that it should always be the case that not both P_1 and P_2 are executing in the critical section.

Safety property example

Example

- *Mutual exclusion* is a safety property. Let C_i denote that process P_i is executing in the critical section. Then

$$\Box\neg(C_1 \wedge C_2)$$

expresses that it should always be the case that not both P_1 and P_2 are executing in the critical section.

- Observe that the negation of a safety formula is a liveness formula; the negation of the formula above is the liveness formula

$$\Diamond(C_1 \wedge C_2)$$

which expresses that eventually it *is* the case that both P_1 and P_2 are executing in the critical section.

Liveness properties

Definition (Liveness)

- A liveness formula is of the form

$$\Diamond\varphi$$

for some first-order formula φ.

Liveness properties

Definition (Liveness)

- A liveness formula is of the form

$$\Diamond \varphi$$

for some first-order formula φ.

- A conditional liveness formula is of the form

$$\varphi \rightarrow \Diamond \psi$$

for first-order formulae φ and ψ.

Liveness properties

Definition (Liveness)

- A liveness formula is of the form

$$\Diamond\varphi$$

  for some first-order formula φ.
- A conditional liveness formula is of the form

$$\varphi \rightarrow \Diamond\psi$$

  for first-order formulae φ and ψ.
- Liveness formulae *guarantee* that some event φ eventually happens: that φ holds in at least one state of the computation.

Connection to Hoare logic

Observation

- Partial correctness is a safety property. Let P be a program and ψ the post condition.

$$\Box(terminated(P) \rightarrow \psi)$$

Connection to Hoare logic

Observation

- Partial correctness is a safety property. Let P be a program and ψ the post condition.

$$\Box(\mathit{terminated}(P) \to \psi)$$

- In the case of full partial correctness, where there is a precondition φ, we get a *conditional safety* formula,

$$\varphi \to \Box(\mathit{terminated}(P) \to \psi),$$

which we can express as $\{\ \varphi\ \}\ P\ \{\ \psi\ \}$ in Hoare Logic.

Total correctness and liveness

Observation

- Total correctness is a liveness property. Let P be a program and ψ the post condition.

$$\Diamond(terminated(P) \land \psi)$$

Total correctness and liveness

Observation

- Total correctness is a liveness property. Let P be a program and ψ the post condition.

$$\Diamond(terminated(P) \land \psi)$$

- In the case of full total correctness, where there is a precondition φ, we get a *conditional liveness* formula,

$$\varphi \rightarrow \Diamond(terminated(P) \land \psi).$$

Duality of partial and total correctness

Observation

Partial and total correctness are dual.

Let

$$PC(\psi) \triangleq \Box(terminated \to \psi)$$
$$TC(\psi) \triangleq \Diamond(terminated \wedge \psi)$$

Then

$$\neg PC(\psi) \sim PC(\neg\psi)$$
$$\neg TC(\psi) \sim TC(\neg\psi)$$

Obligation

Definition (Obligation)

- A simple obligation formula is of the form

$$\Box\varphi \vee \Diamond\psi$$

for first-order formula φ and ψ.

Obligation

Definition (Obligation)

- A simple obligation formula is of the form

$$\Box\varphi \vee \Diamond\psi$$

  for first-order formula φ and ψ.

- An equivalent form is

$$\Diamond\chi \rightarrow \Diamond\psi$$

  which states that some state satisfies χ only if some state satisfies ψ.

Obligation (2)

Proposition

Every safety and liveness formula is also an obligation formula.

Obligation (2)

Proposition

Every safety and liveness formula is also an obligation formula.

Proof.

This is because of the following equivalences.

$$\Box\varphi \sim \Box\varphi \lor \Diamond\bot$$
$$\Diamond\varphi \sim \Box\bot \lor \Diamond\varphi$$

and the facts that $\models \neg\Box\bot$ and $\models \neg\Diamond\bot$. □

Recurrence

Definition (Recurrence)

- A recurrence formula is of the form

$$\Box\Diamond\varphi$$

for some first-order formula φ.

Recurrence

Definition (Recurrence)

- A recurrence formula is of the form

$$\Box\Diamond\varphi$$

for some first-order formula φ.
- It states that infinitely many positions in the computation satisfies φ.

Recurrence

Definition (Recurrence)

- A recurrence formula is of the form

$$\Box\Diamond\varphi$$

for some first-order formula φ.

- It states that infinitely many positions in the computation satisfies φ.

Observation

A response formula, of the form $\Box(\varphi \rightarrow \Diamond\psi)$, is equivalent to a recurrence formula, of the form $\Box\Diamond\chi$, if we allow χ to be a past-formula.

$$\Box(\varphi \rightarrow \Diamond\psi) \sim \Box\Diamond(\neg\varphi)\ W^{-1}\ \psi$$

Recurrence

Proposition

Weak fairness[2] can be specified as the following recurrence formula.

$$\Box\Diamond(enabled(\tau) \rightarrow taken(\tau))$$

[2] weak and strong fairness will be "recurrent" (sorry for the pun) themes. For instance they will show up again in the TLA presentation.

Recurrence

Proposition

Weak fairness[2] *can be specified as the following recurrence formula.*

$$\Box\Diamond(enabled(\tau) \rightarrow taken(\tau))$$

Observation

An equivalent form is

$$\Box(\Box enabled(\tau) \rightarrow \Diamond taken(\tau)),$$

which looks more like the first-order formula we saw last time.

[2] weak and strong fairness will be "recurrent" (sorry for the pun) themes. For instance they will show up again in the TLA presentation.

Persistence

Definition (Persistence)

- A persistence formula is of the form

$$\Diamond\Box\varphi$$

for some first-order formula φ.

[3] In other words: only finitely ("but") many position satisfy $\neg\varphi$. So at some point onwards, it's always φ.

Persistence

Definition (Persistence)

- A persistence formula is of the form

$$\Diamond\Box\varphi$$

for some first-order formula φ.
- It states that all but finitely many positions satisfy φ[3]

[3]In other words: only finitely ("but") many position satisfy $\neg\varphi$. So at some point onwards, it's always φ.

Persistence

Definition (Persistence)

- A persistence formula is of the form

$$\Diamond\Box\varphi$$

for some first-order formula φ.
- It states that all but finitely many positions satisfy φ[3]
- Persistence formulae are used to describe the eventual stabilization of some state property.

[3] In other words: only finitely ("but") many position satisfy $\neg\varphi$. So at some point onwards, it's always φ.

Recurrence and Persistence

Observation

Recurrence and persistence are duals.

$$\neg(\Box\Diamond\varphi) \sim (\Diamond\Box\neg\varphi)$$
$$\neg(\Diamond\Box\varphi) \sim (\Box\Diamond\neg\varphi)$$

Reactivity

Definition (Reactivity)

- A simple reactivity formula is of the form

$$\Box\Diamond\varphi \lor \Diamond\Box\psi$$

for first-order formula φ and ψ.

Reactivity

Definition (Reactivity)

- A simple reactivity formula is of the form

$$\Box\Diamond\varphi \vee \Diamond\Box\psi$$

  for first-order formula φ and ψ.
- A very general class of formulae are conjunctions of reactivity formulae.

Reactivity

Definition (Reactivity)

- A simple reactivity formula is of the form

$$\Box\Diamond\varphi \lor \Diamond\Box\psi$$

  for first-order formula φ and ψ.
- A very general class of formulae are conjunctions of reactivity formulae.
- An equivalent form is

$$\Box\Diamond\chi \to \Box\Diamond\psi,$$

  which states that if the computation contains infinitely many χ-positions, it must also contain infinitely many ψ-positions.

Reactivity

Proposition

Strong fairness can be specified as the following reactivity formula.

$$\Box\Diamond enabled(\tau) \to \Box\Diamond taken(\tau)$$

GCD Example

Below is a computation σ of our recurring GCD program.

P-computation

States are of the form $\langle \pi, x, y, g \rangle$.

$$
\begin{aligned}
\sigma: \quad & \langle l_1, 21, 49, 0\rangle \to \langle l_2^b, 21, 49, 0\rangle \to \langle l_6, 21, 49, 0\rangle \to \\
& \langle l_1, 21, 28, 0\rangle \to \langle l_2^b, 21, 28, 0\rangle \to \langle l_6, 21, 28, 0\rangle \to \\
& \langle l_1, 21, 7, 0\rangle \to \langle l_2^a, 21, 7, 0\rangle \to \langle l_4, 21, 7, 0\rangle \to \\
& \langle l_1, 14, 7, 0\rangle \to \langle l_2^a, 14, 7, 0\rangle \to \langle l_4, 14, 7, 0\rangle \to \\
& \langle l_1, 7, 7, 0\rangle \to \langle l_7, 7, 7, 0\rangle \to \langle l_8, 7, 7, 7\rangle \to \cdots
\end{aligned}
$$

GCD Example

Below is a computation σ of our recurring GCD program.

- a and b are fixed: $\sigma \models \Box(a \doteq 21 \wedge b \doteq 49)$.

P-computation

States are of the form $\langle \pi, x, y, g \rangle$.

$$\begin{array}{llll} \sigma: & \langle l_1, 21, 49, 0\rangle \rightarrow \langle l_2^b, 21, 49, 0\rangle \rightarrow \langle l_6, 21, 49, 0\rangle \rightarrow \\ & \langle l_1, 21, 28, 0\rangle \rightarrow \langle l_2^b, 21, 28, 0\rangle \rightarrow \langle l_6, 21, 28, 0\rangle \rightarrow \\ & \langle l_1, 21, 7, 0\rangle \rightarrow \langle l_2^a, 21, 7, 0\rangle \rightarrow \langle l_4, 21, 7, 0\rangle \rightarrow \\ & \langle l_1, 14, 7, 0\rangle \rightarrow \langle l_2^a, 14, 7, 0\rangle \rightarrow \langle l_4, 14, 7, 0\rangle \rightarrow \\ & \langle l_1, 7, 7, 0\rangle \rightarrow \langle l_7, 7, 7, 0\rangle \rightarrow \langle l_8, 7, 7, 7\rangle \rightarrow \cdots \end{array}$$

GCD Example

Below is a computation σ of our recurring GCD program.

- a and b are fixed: $\sigma \models \Box(a \doteq 21 \wedge b \doteq 49)$.
- $at(l)$ denotes the formulae $(\pi \doteq \{l\})$.

P-computation

States are of the form $\langle \pi, x, y, g \rangle$.

$$
\begin{aligned}
\sigma : \quad & \langle l_1, 21, 49, 0 \rangle \to \langle l_2^b, 21, 49, 0 \rangle \to \langle l_6, 21, 49, 0 \rangle \to \\
& \langle l_1, 21, 28, 0 \rangle \to \langle l_2^b, 21, 28, 0 \rangle \to \langle l_6, 21, 28, 0 \rangle \to \\
& \langle l_1, 21, 7, 0 \rangle \to \langle l_2^a, 21, 7, 0 \rangle \to \langle l_4, 21, 7, 0 \rangle \to \\
& \langle l_1, 14, 7, 0 \rangle \to \langle l_2^a, 14, 7, 0 \rangle \to \langle l_4, 14, 7, 0 \rangle \to \\
& \langle l_1, 7, 7, 0 \rangle \to \langle l_7, 7, 7, 0 \rangle \to \langle l_8, 7, 7, 7 \rangle \to \cdots
\end{aligned}
$$

GCD Example

Below is a computation σ of our recurring GCD program.

- a and b are fixed: $\sigma \models \Box(a \doteq 21 \wedge b \doteq 49)$.
- $at(l)$ denotes the formulae $(\pi \doteq \{l\})$.
- $terminated$ denotes the formula $at(l_8)$.

P-computation

States are of the form $\langle \pi, x, y, g \rangle$.

$$
\begin{aligned}
\sigma : \quad & \langle l_1, 21, 49, 0 \rangle \to \langle l_2^b, 21, 49, 0 \rangle \to \langle l_6, 21, 49, 0 \rangle \to \\
& \langle l_1, 21, 28, 0 \rangle \to \langle l_2^b, 21, 28, 0 \rangle \to \langle l_6, 21, 28, 0 \rangle \to \\
& \langle l_1, 21, 7, 0 \rangle \to \langle l_2^a, 21, 7, 0 \rangle \to \langle l_4, 21, 7, 0 \rangle \to \\
& \langle l_1, 14, 7, 0 \rangle \to \langle l_2^a, 14, 7, 0 \rangle \to \langle l_4, 14, 7, 0 \rangle \to \\
& \langle l_1, 7, 7, 0 \rangle \to \langle l_7, 7, 7, 0 \rangle \to \langle l_8, 7, 7, 7 \rangle \to \cdots
\end{aligned}
$$

GCD Example

Does the following properties hold for σ? And why?

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$
3. $at(l_8) \rightarrow terminated$

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$
3. $at(l_8) \rightarrow terminated$
4. $at(l_7) \rightarrow \Diamond terminated$ (conditional liveness)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$
3. $at(l_8) \rightarrow terminated$
4. $at(l_7) \rightarrow \Diamond terminated$ (conditional liveness)
5. $\Diamond at(l_7) \rightarrow \Diamond terminated$ (obligation)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box \mathit{terminated}$ (safety)
2. $at(l_1) \to \mathit{terminated}$
3. $at(l_8) \to \mathit{terminated}$
4. $at(l_7) \to \Diamond \mathit{terminated}$ (conditional liveness)
5. $\Diamond at(l_7) \to \Diamond \mathit{terminated}$ (obligation)
6. $\Box(\mathrm{gcd}(x, y) \doteq \mathrm{gcd}(a, b))$ (safety)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$
3. $at(l_8) \rightarrow terminated$
4. $at(l_7) \rightarrow \Diamond terminated$ (conditional liveness)
5. $\Diamond at(l_7) \rightarrow \Diamond terminated$ (obligation)
6. $\Box(\mathrm{gcd}(x, y) \doteq \mathrm{gcd}(a, b))$ (safety)
7. $\Diamond terminated$ (liveness)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box terminated$ (safety)
2. $at(l_1) \rightarrow terminated$
3. $at(l_8) \rightarrow terminated$
4. $at(l_7) \rightarrow \Diamond terminated$ (conditional liveness)
5. $\Diamond at(l_7) \rightarrow \Diamond terminated$ (obligation)
6. $\Box(\mathrm{gcd}(x, y) \doteq \mathrm{gcd}(a, b))$ (safety)
7. $\Diamond terminated$ (liveness)
8. $\Diamond\Box(y \doteq \mathrm{gcd}(a, b))$ (persistence)

GCD Example

Does the following properties hold for σ? And why?

1. $\Box \mathit{terminated}$ (safety)
2. $at(l_1) \rightarrow \mathit{terminated}$
3. $at(l_8) \rightarrow \mathit{terminated}$
4. $at(l_7) \rightarrow \Diamond \mathit{terminated}$ (conditional liveness)
5. $\Diamond at(l_7) \rightarrow \Diamond \mathit{terminated}$ (obligation)
6. $\Box(\mathrm{gcd}(x, y) \doteq \mathrm{gcd}(a, b))$ (safety)
7. $\Diamond \mathit{terminated}$ (liveness)
8. $\Diamond\Box(y \doteq \mathrm{gcd}(a, b))$ (persistence)
9. $\Box\Diamond \mathit{terminated}$ (recurrence)

Exercises

Exercises

1. Show that the following formulae are (not) LTL-valid.
   1.1 $\Box\varphi \leftrightarrow \Box\Box\varphi$
   1.2 $\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$
   1.3 $\neg\Box\varphi \rightarrow \Box\neg\Box\varphi$
   1.4 $\Box(\Box\varphi \rightarrow \psi) \rightarrow \Box(\Box\psi \rightarrow \varphi)$
   1.5 $\Box(\Box\varphi \rightarrow \psi) \vee \Box(\Box\psi \rightarrow \varphi)$
   1.6 $\Box\Diamond\Box\varphi \rightarrow \Diamond\Box\varphi$
   1.7 $\Box\Diamond\varphi \leftrightarrow \Box\Diamond\Box\Diamond\varphi$
2. A *modality* is a sequence of $\neg$, $\Box$ and $\Diamond$, including the empty sequence ϵ. Two modalities σ and τ are *equivalent* if $\sigma\varphi \leftrightarrow \tau\varphi$ is valid.
   2.1 Which are the non-equivalent modalities in LTL, and
   2.2 what are their relationship (ie. implication-wise)?

References I

Bibliography

[1] Manna, Z. and Pnueli, A. (1992). *The temporal logic of reactive and concurrent systems—Specification*. Springer Verlag, New York.