INF5140/INF9140 - Lecture 4
Hoare Logic and Temporal Logics

Lecturer: Cristian Prisacariu

Precise Modeling and Analysis group (PMA),
University of Oslo

17 Feb. 2011

Introduction

- First Order Logic is very expressive but undecidable. Good for mathematics but not good for computers.

!! FOL can talk about the state but NOT about change of state.

- Modal Logic gives us the power to talk about *changing of state*.

!! But ML does NOT talk about programs and change of program state.

- Dynamic Logic DOES talk about programs and change of state (regular programs that are more expressive than while programs).

!! DL is abstract and general. We will se this by encoding Hoare Logic.

- Hoare Logic is the first logical formalism defined for reasoning about programs in a compositional way.

  DL extends Hoare Logic...

Outline

1. Hoare Logic
2. Hoare Logic in Dynamic Logic
3. Temporal Logics

Reasoning about programs

- At the very basis programs can ofte be seen as implementing some functionality that is formally specified in terms of imputs and outputs.

!! If the program is given some imput, then it finishes with some output.

- More general: The program α, when started in a state that satisfies the specification ϕ, the state that it stops in is guaranteed to satisfy some output specification ψ.

$$\{\phi\}\alpha\{\psi\}$$

- We call a program *totally correct* iff it is guaranteed to terminate.
- The program is only *partially correct* when we do not care about its termination, but whenever it terminates the output specification is guaranteed.

Hoare Logic

Syntax of HL uses statements like (called Hoare triples):

$$\{\phi\}\alpha\{\psi\}$$

where α is the program under question and:

1. ϕ is called the pre-condition
   It specifies how the imput to the program should be (i.e., which kind of state the program can start in). Only imput that satisfies the pre-condition is relevant; all other imputs are not considered. ϕ is the least requirements (the assumptions) that the program makes on the environment.
2. ψ is called the post-condition
   It is guaranteed to be true at the end of the program. The state where the program terminates will satisfy the post-condition (it may satisfy more). This is the least guarantee from the program.

Hoare Logic

Rules of inference/deduction

Composition rule (for iteration of programs):

$$\frac{\{\phi_1\}\,\alpha\,\{\phi_2\}, \qquad \{\phi_2\}\,\beta\,\{\phi_3\}}{\{\phi_1\}\,\alpha;\beta\,\{\phi_3\}}$$

Conditional rule:

$$\frac{\{\phi_1 \wedge \phi\}\,\alpha\,\{\phi_2\}, \qquad \{\phi_1 \wedge \neg\phi\}\,\beta\,\{\phi_2\}}{\{\phi_1\}\,\textbf{if}\ \phi\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\,\{\phi_2\}}$$

While rule:

$$\frac{\{\psi \wedge \phi\}\,\alpha\,\{\psi\}}{\{\psi\}\,\textbf{while}\ \phi\ \textbf{do}\ \alpha\,\{\psi \wedge \neg\phi\}}$$

Hoare Logic

Rules of inference/deduction

Weakening rule:

$$\frac{\phi' \to \phi, \qquad \{\phi\}\,\alpha\,\{\psi\}, \qquad \psi \to \psi'}{\{\phi'\}\,\alpha\,\{\psi'\}}$$

Assignment axiom:

$$\{\phi[x/e]\}\; x := e \;\{\phi\}$$

- The assignment axiom is particular to the programming language.
- The assignment is the only basic/atomic program assumed here. Actually, there are infinitely many basic programs as there are infinitely many assignments.
- Another programming language may also have other basic programs...

Encoding Hoare logic into Dynamic logic

All that is to know is that the Hoare triple (partial correctness assertion)

$$\{\phi\}\,\alpha\,\{\psi\}$$

is encoded in Dynamic Logic as:

$$\phi \to [\alpha]\psi$$

The intuition is simple:

– The Dynamic logic formula says that in any state where ϕ holds it must be that $[\alpha]\psi$ holds. In all other states the formula is trivially true, hence those states are ignored.
– That $[\alpha]\psi$ holds means that in all the states where the program α ends (i.e., all those states related by the R_α) the ψ must hold.

Encoding Hoare logic into Dynamic logic

Deriving the Hoare rules

For each Hoare rule, assume the premise and derive the conclusion, using the derivation system of DL from end of Lecture 3.

Composition rule:

assume: $\phi_1 \to [\alpha]\phi_2$ and $\phi_2 \to [\beta]\phi_3$
derive: $\phi_1 \to [\alpha \cdot \beta]\phi_3$

Conditional rule:

assume: $\phi_1 \wedge \phi \to [\alpha]\phi_2$ and $\phi_1 \wedge \neg\phi \to [\beta]\phi_2$
derive: $\phi_1 \to [(\phi? \cdot \alpha) + ((\neg\phi)? \cdot \beta)]\phi_2$

While rule:

assume: $\psi \wedge \phi \to [\alpha]\psi$
derive: $\psi \to [(\phi? \cdot \alpha)^* \cdot \neg\phi?](\psi \wedge \neg\phi)$

Weakening rule:

assume: $\phi' \to \phi$ and $\phi \to [\alpha]\psi$ and $\psi \to \psi'$
derive: $\phi' \to [\alpha]\psi'$

Logics of Programs

Endogenous vs. Exogenous

Logics for programs can be divided into two:

1. Endogenous, which work over a single fixed program which is the model of the logic and the logic talks about the states of this program.

   Examples: Temporal logics

2. Exogenous, which have the programs explicit in the language. These logics work by decomposing the program into smaller programs.

   Examples: Hoare logic, Dynamic logics

What is Temporal Logic?

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

We will see a logic that talks about liear time, time instances, discrete time, and future only.

Linear Temporal Logic

Introduction

- Linear Temporal Logic talks about computations.
- A computation is a sequence of states.
- Therefore, temporal logic talks about sequences of states.
- Temporal logic extends Modal logic with some operators.
- Temporal logic restricts Modal logic in the sense that it works on restricted structures (i.e., the relations have strong restrictions)

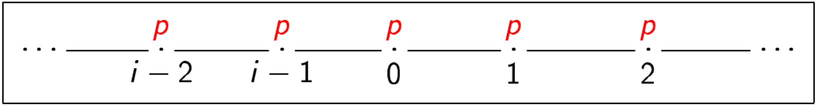
Linear Temporal Logic

Introduction

In Linear Temporal Logic (LTL) we can describe such properties as, if i is *now*,

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

We will only be concerned with the future.



Linear Temporal Logic

Introduction

We extend FOL to a temporal language by adding the temporal operators $\Box$, $\Diamond$, $\bigcirc$, U, R and W.

Interpretation

$\Box\varphi$	φ will *always* (in every state) hold
$\Diamond\varphi$	φ will *eventually* (in some state) hold
$\bigcirc\varphi$	φ will hold at the *next* point in time
$\varphi\, U\, \psi$	ψ will eventually hold, and *until* that point φ will hold
$\varphi\, R\, \psi$	ψ holds until (incl.) the point (if any) where φ holds (*release*)
$\varphi\, W\, \psi$	φ will hold until ψ holds (*weak until* or *waiting for*)

Linear Temporal Logic

Syntax

We define LTL formulae as follows.

Definition

- FOL formulae are also LTL formulae.
- If φ is an LTL formula, so are the following.

$$\Box\varphi \quad | \quad \Diamond\varphi \quad | \quad \bigcirc\varphi \quad | \quad \neg\varphi$$

- If φ and ψ are LTL formulae, so are

$$(\varphi\, U\, \psi) \quad | \quad (\varphi\, R\, \psi) \quad | \quad (\varphi\, W\, \psi)$$

$$(\varphi \vee \psi) \quad | \quad (\varphi \wedge \psi) \quad | \quad (\varphi \rightarrow \psi) \quad | \quad (\varphi \equiv \psi)$$

Linear Temporal Logic

Semantics

Definition

- 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.

Linear Temporal Logic

Semantics

Definition

We define the notion that an LTL formula φ is true (false) relative to a path σ, written $\sigma \models \varphi$ ($\sigma \not\models \varphi$) as follows.

$$
\begin{array}{ll}
\sigma \models \varphi & \text{iff } \sigma_0 \models \varphi \text{ when } \varphi \in FOL \\
\sigma \models \neg\varphi & \text{iff } \sigma \not\models \varphi \\
\sigma \models \varphi \vee \psi & \text{iff } \sigma \models \varphi \text{ or } \sigma \models \psi \\
\\
\sigma \models \Box\varphi & \text{iff } \sigma^k \models \varphi \text{ for all } k \geq 0 \\
\sigma \models \Diamond\varphi & \text{iff } \sigma^k \models \varphi \text{ for some } k \geq 0 \\
\sigma \models \bigcirc\varphi & \text{iff } \sigma^1 \models \varphi
\end{array}
$$

(cont.)

Linear Temporal Logic

Semantics

Definition

(cont.)

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

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

$$\sigma \models \varphi\, W\, \psi \text{ iff } \sigma \models \varphi\, U\, \psi \text{ or } \sigma \models \Box\varphi$$

Linear Temporal Logic

Semantics

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
$$\models \varphi \equiv \psi \text{ (i.e. } \sigma \models \varphi \text{ iff } \sigma \models \psi\text{, for all } \sigma).$$

Example

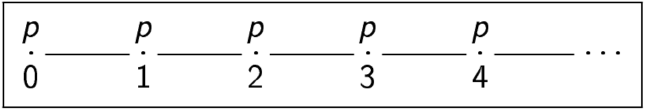
$\Box$ distributes over $\wedge$, while $\Diamond$ distributes over $\vee$.

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

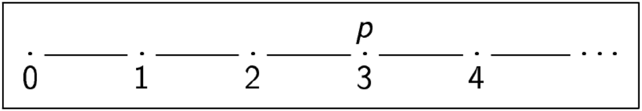
Linear Temporal Logic

Semantics

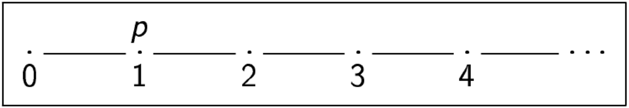
$\sigma \models \Box p$ (p will always hold)



$\sigma \models \Diamond p$ (p will eventually hold)



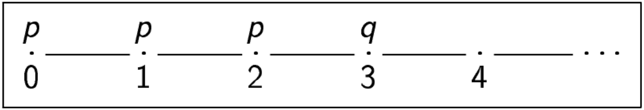
$\sigma \models \bigcirc p$ (p will hold at the next point)



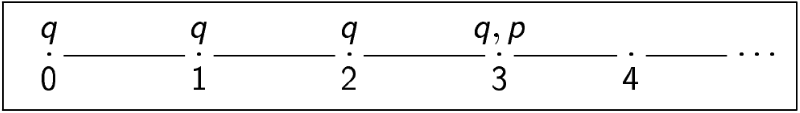
Linear Temporal Logic

Semantics

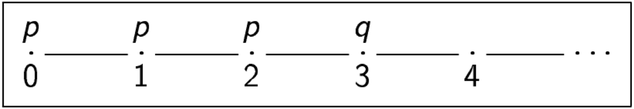
$\sigma \models p\, U\, q$. The sequence of ps is finite.



$\sigma \models p\, R\, q$. The sequence of qs may be infinite.



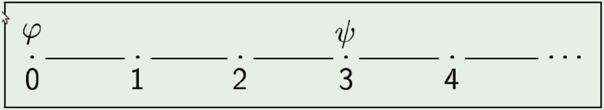
$\sigma \models p\, W\, q$. The sequence of ps may be infinite ($p\, W\, q \sim p\, U\, q \vee \Box p$).



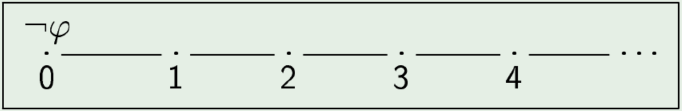
Example

$\varphi \rightarrow \Diamond\psi$

If φ holds initially, ψ holds eventually.



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



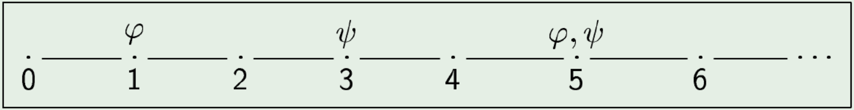
Linear Temporal Logic

Examples

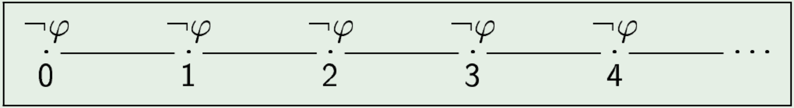
Example

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

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



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



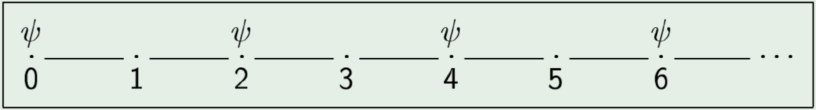
Linear Temporal Logic

Examples

Example

$\Box\Diamond\psi$

There are infinitely many ψ-positions.



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

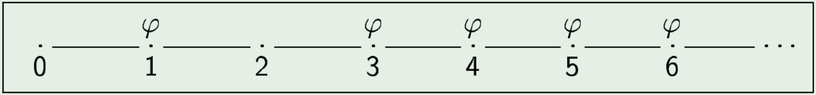
Linear Temporal Logic

Examples

Example

$\Diamond\Box\varphi$

Eventually φ will hold permanently.



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

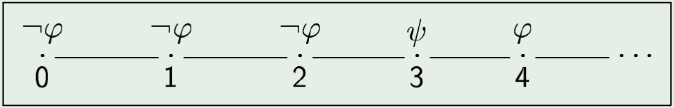
Linear Temporal Logic

Examples

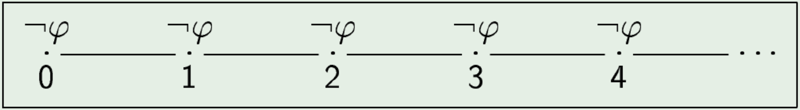
Example

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

The first φ-position must coincide or be preceded by a ψ-position.



φ need never hold.



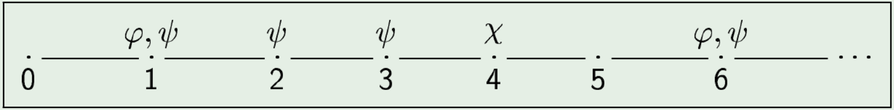
Linear Temporal Logic

Examples

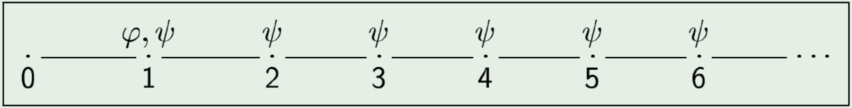
Example

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

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



The sequence of ψ-positions need not terminate.



Linear Temporal Logic

Formalization

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.
- None of these is necessarily what is meant.

Linear Temporal Logic

Duals

Definition (Duals)

For binary boolean connectives $\circ$ 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 symmetrical:

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

Linear Temporal Logic

Duals

Which connectives are duals?

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

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

- $\neg$ is its own dual:

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

Linear Temporal Logic

Duals

- 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 (eg. $\not\to$ 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 $\vee$.
- $\varphi \to \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \equiv \psi$ is equivalent to $(\varphi \to \psi) \wedge (\psi \to \varphi)$.
- $\top$ is equivalent to $p \vee \neg p$
- $\perp$ is equivalent to $p \wedge \neg p$

Linear Temporal Logic

Duals

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.

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

Linear Temporal Logic

Duals

We don't need all our temporal operators either.

Theorem

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

Proof.

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

$\Box$

Classification

Properties

We can classify a number of properties expressible in LTL.

Classification

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

Classification

Safety

Definition (Safety)

- A safety formula is of the form

$$\Box\psi$$

  for some first-order formula ψ.
- A conditional safety formula is of the form

$$\phi \rightarrow \Box\psi$$

  for first-order formulae ϕ and ψ.
- Safety formulae express *invariance* of some state property ψ: that ψ holds in every state of the computation.
  - Always in good states.
  - Never something bad happens.

Classification

Safety

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. (Never bad.)
- 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.

Classification

Liveness

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.
  - Eventually output something.
  - Still alive.

Observation

- Partial correctness is a safety property. Let P state that the program has terminated and ψ the post condition.

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

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

$$\varphi \rightarrow \Box(P \rightarrow \psi),$$

  which we can express as $\{\varphi\}\, P\, \{\psi\}$ in Hoare Logic.
- Never BAD can happen; where BAD is when program terminates and ψ does not hold.

Classification

Safety and Liveness

Observation

- Total correctness is a liveness property. Let P state that the program has terminated and ψ the post condition.

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

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

$$\varphi \to \Diamond(P \land \psi).$$

- Eventually the program terminates, and terminates well.

Classification

Safety and Liveness

Observation

Partial and total correctness are dual.

$$\neg(\Box(P \rightarrow \phi)) = \Diamond(P \wedge \neg\phi)$$

Classification

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

$$\Box(\varphi \to \Diamond\psi) \sim \Box\Diamond(\neg\varphi)\, W\, \psi$$

Definition (Persistence)

- A persistence formula is of the form

$$\Diamond\Box\varphi$$

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

Classification

Recurrence and Persistence

Observation

Recurrence and persistence are duals.

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

Classification

Fairness

In systems with parallel process that use a same resource, like processor, we talk about fair executions.

- unconditional fairness says that a process is executed infinitely often (in an infinite run)

$$\Box\Diamond ExectP_1$$

- strong fairness says that if a process is enabled infinitely often then it will be executed infinitely often

$$\Box\Diamond EnabP_1 \rightarrow \Box\Diamond ExectP_1$$

- weak fairness says that if a process is enabled infinitely long then it is executed infinitely often

$$\Diamond\Box EnabP_1 \rightarrow \Box\Diamond ExectP_1$$

"Infinitely long" should be understood as from some point on is always enabled

Exercises

Exercises

1. Show that the following formulae are (not) LTL-valid.
   1. $\Box\varphi \equiv \Box\Box\varphi$
   2. $\Diamond\varphi \equiv \Diamond\Diamond\varphi$
   3. $\neg\Box\varphi \to \Box\neg\Box\varphi$
   4. $\Box(\Box\varphi \to \psi) \to \Box(\Box\psi \to \varphi)$
   5. $\Box(\Box\varphi \to \psi) \vee \Box(\Box\psi \to \varphi)$
   6. $\Box\Diamond\Box\varphi \to \Diamond\Box\varphi$
   7. $\Box\Diamond\varphi \equiv \Box\Diamond\Box\Diamond\varphi$

Thank you!