Reference Slides

COM2107 Logic in Computer Science

Brian Courtehoute
based on lecture notes by Georg Struth
and slides by Jonni Virtema

University of Sheffield

Week 11

Components of Logic Systems

	Syntax	**Deductive System**	**Semantics**	**Metalanguage**
purpose	constructs expressions	transforms expressions	interprets expressions	presents expressions
symbols	$\rightarrow$, $\leftrightarrow$	$\vdash$, $\dashv\vdash$	$\models$, $\equiv$	$\Rightarrow$, $\Leftrightarrow$
symbol meaning	'$\rightarrow$' is part of a logical expression	'$\vdash$' means entailment in a proof system	'$\models$' means entailment of truth values	'$\Rightarrow$' provides if-then-reasoning about the logic itself
related objects	expressions (i.e. formulas)	tabular proofs, proof trees, deduction rules	truth tables, valuations, structures	mathematical statements and proofs using metalanguage

Section 1

Propositional Logic

Propositional Logic: Syntax

The set $\mathbf{\Phi}$ of *formulas* is defined by the following grammar.

$$\mathbf{\Phi} ::= \bot \mid \top \mid p \mid (\neg \mathbf{\Phi}) \mid (\mathbf{\Phi} \wedge \mathbf{\Phi}) \mid (\mathbf{\Phi} \vee \mathbf{\Phi}) \mid (\mathbf{\Phi} \rightarrow \mathbf{\Phi})$$

where

- p is a *propositional variable* from a countably infinite set P,
- $\bot, \top, \neg, \vee, \wedge, \rightarrow$ are *propositional connectives*,
- (,) are *auxiliary symbols*.

$\bot$, $\top$, and propositional variables p are *atomic formulas*, and all other formulas are *composite*.

Propositional Logic: Deduction Rules

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi}\wedge I \qquad \frac{\varphi \wedge \psi}{\varphi}\wedge E_l \qquad \frac{\varphi \wedge \psi}{\psi}\wedge E_r$$

$$\frac{}{\top}\top I \qquad \frac{\bot}{\varphi}\bot E$$

$$\frac{\varphi}{\varphi \vee \psi}\vee I_l \qquad \frac{\psi}{\varphi \vee \psi}\vee I_r \qquad \frac{\varphi \vee \psi \qquad \begin{matrix}[\varphi]\\ \vdots \\ \chi\end{matrix} \qquad \begin{matrix}[\psi]\\ \vdots \\ \chi\end{matrix}}{\chi}\vee E$$

$$\frac{\begin{matrix}[\varphi]\\ \vdots \\ \psi\end{matrix}}{\varphi \to \psi}\to I \qquad \frac{\varphi \to \psi \qquad \varphi}{\psi}\to E$$

$$\frac{\begin{matrix}[\varphi]\\ \vdots \\ \bot\end{matrix}}{\neg\varphi}\neg I \qquad \frac{\varphi \qquad \neg\varphi}{\bot}\neg E$$

$$\frac{}{\varphi \vee \neg\varphi}\text{lem} \qquad \frac{\begin{matrix}[\neg\varphi]\\ \vdots \\ \bot\end{matrix}}{\varphi}\text{pbc} \qquad \frac{\neg\neg\varphi}{\varphi}\neg\neg E$$

Propositional Logic: Semantics

- Let $v\colon P \to \mathbb{B}$ be an assignment. A *valuation function* is defined as follows.

$$
\begin{aligned}
[\![\top]\!]_v &= 1 \\
[\![\bot]\!]_v &= 0 \\
[\![p]\!]_v &= v(p) \qquad \text{for } p \in P \\
[\![\neg\varphi]\!]_v &= 1 - [\![\varphi]\!]_v \\
[\![\varphi \wedge \psi]\!]_v &= \min([\![\varphi]\!]_v, [\![\psi]\!]_v) \\
[\![\varphi \vee \psi]\!]_v &= \max([\![\varphi]\!]_v, [\![\psi]\!]_v) \\
[\![\varphi \to \psi]\!]_v &= \max(1 - [\![\varphi]\!]_v, [\![\psi]\!]_v)
\end{aligned}
$$

- In the following definitions, φ and ψ are formulas and Γ is a set of formulas.

Notation	Name	Definition
$\models \varphi$	φ is a *tautology*	$[\![\varphi]\!]_v = 1$ for all assignments v
$\Gamma \models \varphi$	Γ *entails* φ	For every assignment v, if $[\![\psi]\!]_v = 1$ for each $\psi \in \Gamma$, then $[\![\varphi]\!]_v = 1$
$\varphi \equiv \psi$	φ and ψ are *logically equivalent*	$[\![\varphi]\!]_v = [\![\psi]\!]_v$ for all assignments v

Section 2

Predicate Logic

Predicate Logic: Syntax

The set $\mathbf{\Phi}_\Sigma$ of Σ-formulas is defined by the following grammar.

$$\begin{aligned}\mathbf{\Phi}_\Sigma ::= {} & \bot \mid \top \mid t_1 = t_2 \mid P(t_1, \dots, t_n) \mid (\forall x.\ \mathbf{\Phi}_\Sigma) \mid (\exists x.\ \mathbf{\Phi}_\Sigma) \\ & \mid (\neg \mathbf{\Phi}_\Sigma) \mid (\mathbf{\Phi}_\Sigma \wedge \mathbf{\Phi}_\Sigma) \mid (\mathbf{\Phi}_\Sigma \vee \mathbf{\Phi}_\Sigma) \mid (\mathbf{\Phi}_\Sigma \to \mathbf{\Phi}_\Sigma)\end{aligned}$$

where

- $\Sigma = \mathcal{F} \cup \mathcal{P}$ is a *signature*,
- $\mathcal{F} = \{f_1, f_2, \dots\}$ is a set of *function symbols*, each of fixed arity,
- $\mathcal{P} = \{P_1, P_2, \dots\}$ is a set of *predicate symbols*, each of fixed arity,
- x is a *variable* from a countably infinite set $\mathcal{V}$,
- $t_1, \dots, t_n$ are *terms* defined by the grammar

$$\mathcal{T}_\Sigma ::= x \mid f(\mathcal{T}_\Sigma, \dots, \mathcal{T}_\Sigma), \text{ where } x \in \mathcal{V} \text{ and } f \in \mathcal{F},$$

- $=$ is the special binary predicate symbol for *equality*,
- $\bot, \top, \neg, \vee, \wedge, \to, \forall, \exists$ are *connectives*,
- (,) are *auxiliary symbols*.

Predicate Logic: Deduction Rules

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi}\ \wedge I \qquad \frac{\varphi \wedge \psi}{\varphi}\ \wedge E_l \qquad \frac{\varphi \wedge \psi}{\psi}\ \wedge E_r$$

$$\frac{}{\top}\ \top I \qquad \frac{\bot}{\varphi}\ \bot E$$

$$\frac{\varphi}{\varphi \vee \psi}\ \vee I_l \qquad \frac{\psi}{\varphi \vee \psi}\ \vee I_r \qquad \frac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]\\ \vdots \\ \chi\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots \\ \chi\end{array}}{\chi}\ \vee E$$

$$\frac{\begin{array}{c}[\varphi]\\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\ {\to} I \qquad \frac{\varphi \to \psi \qquad \varphi}{\psi}\ {\to} E$$

$$\frac{\begin{array}{c}[\varphi]\\ \vdots \\ \bot\end{array}}{\neg\varphi}\ \neg I \qquad \frac{\varphi \qquad \neg\varphi}{\bot}\ \neg E$$

$$\frac{}{\varphi \vee \neg\varphi}\ \text{lem} \qquad \frac{\begin{array}{c}[\neg\varphi]\\ \vdots \\ \bot\end{array}}{\varphi}\ \text{pbc} \qquad \frac{\neg\neg\varphi}{\varphi}\ \neg\neg E$$

$$\frac{\varphi\,[t/x]}{\exists x.\,\varphi}\ \exists I$$

where t is any term that is free for x in φ

$$\frac{\exists x.\,\varphi \qquad \begin{array}{c}[\varphi\,[a/x]]\\ \vdots \\ \psi\end{array}}{\psi}\ \exists E$$

where a is a fresh parameter

$$\frac{}{t = t}\ {=}I \qquad \frac{t_1 = t_2 \qquad \varphi\,[t_1/x]}{\varphi\,[t_2/x]}\ {=}E$$

$$\frac{\varphi\,[a/x]}{\forall x.\,\varphi}\ \forall I$$

where a is a fresh parameter

$$\frac{\forall x.\,\varphi}{\varphi\,[t/x]}\ \forall E$$

where t is any term that is free for x in φ

Predicate Logic: Semantics (Part 1)

Semantics of predicates and functions

A Σ-*structure* is a pair $\mathfrak{A} = (A, (-)^{\mathfrak{A}})$, where

- A is a nonempty set called the *carrier set*,
- $(-)^{\mathfrak{A}}$ is a function that maps
  - an n-ary predicate symbol $R \in \Sigma$ to an n-ary relation $R^{\mathfrak{A}} \subseteq A^n$,
  - an n-ary function symbol $f \in \Sigma$ to an n-ary function $f^{\mathfrak{A}} \colon A^n \to A$.

Semantics of terms

An *assignment* is a function $v \colon \mathcal{V} \to A$ that associates variables with constants $c^{\mathfrak{A}} \in A$.

An *interpretation* of terms is a function $[\![-]\!]^{\mathfrak{A}}_v \colon \mathcal{T}_\Sigma \to A$ defined by

$$
\begin{aligned}
[\![x]\!]^{\mathfrak{A}}_v &= v(x) && \text{if } x \in \mathcal{V},\\
[\![c]\!]^{\mathfrak{A}}_v &= c^{\mathfrak{A}} && \text{if } c \in \mathcal{F} \text{ is a constant},\\
[\![f(t_1, \dots, t_n)]\!]^{\mathfrak{A}}_v &= f^{\mathfrak{A}}([\![t_1]\!]^{\mathfrak{A}}_v, \dots, [\![t_n]\!]^{\mathfrak{A}}_v) && \text{if } n \geq 1.
\end{aligned}
$$

Predicate Logic: Semantics (Part 2)

Semantics of formulas

We extend $[\![-]\!]^{\mathfrak{A}}_v \colon \mathcal{T}_\Sigma \to A$ to an *interpretation* of Σ-formulas $[\![-]\!]^{\mathfrak{A}}_v \colon \mathbf{\Phi}_\Sigma \to \mathbb{B}$.

$$
\begin{aligned}
[\![\bot]\!]^{\mathfrak{A}}_v &= 0 \\
[\![\top]\!]^{\mathfrak{A}}_v &= 1 \\
[\![t_1 = t_2]\!]^{\mathfrak{A}}_v &= \begin{cases} 1 & \text{if } [\![t_1]\!]^{\mathfrak{A}}_v = [\![t_2]\!]^{\mathfrak{A}}_v \\ 0 & \text{otherwise} \end{cases} \\
[\![P(t_1, \ldots, t_n)]\!]^{\mathfrak{A}}_v &= \begin{cases} 1 & \text{if } ([\![t_1]\!]^{\mathfrak{A}}_v, \ldots, [\![t_n]\!]^{\mathfrak{A}}_v) \in P^{\mathfrak{A}} \\ 0 & \text{otherwise} \end{cases} \\
[\![\neg\varphi]\!]^{\mathfrak{A}}_v &= 1 - [\![\varphi]\!]^{\mathfrak{A}}_v \\
[\![\varphi \wedge \psi]\!]^{\mathfrak{A}}_v &= \min([\![\varphi]\!]^{\mathfrak{A}}_v, [\![\psi]\!]^{\mathfrak{A}}_v) \\
[\![\varphi \vee \psi]\!]^{\mathfrak{A}}_v &= \max([\![\varphi]\!]^{\mathfrak{A}}_v, [\![\psi]\!]^{\mathfrak{A}}_v) \\
[\![\varphi \to \psi]\!]^{\mathfrak{A}}_v &= \max([\![1 - \varphi]\!]^{\mathfrak{A}}_v, [\![\psi]\!]^{\mathfrak{A}}_v) \\
[\![\forall x.\, \varphi]\!]^{\mathfrak{A}}_v &= \min_{c^{\mathfrak{A}} \in A}([\![\varphi]\!]^{\mathfrak{A}}_{v[x \mapsto c^{\mathfrak{A}}]}) \\
[\![\exists x.\, \varphi]\!]^{\mathfrak{A}}_v &= \max_{c^{\mathfrak{A}} \in A}([\![\varphi]\!]^{\mathfrak{A}}_{v[x \mapsto c^{\mathfrak{A}}]}) \qquad \text{where } v\left[x \mapsto c^{\mathfrak{A}}\right](y) = \begin{cases} c^{\mathfrak{A}} & \text{if } x = y \\ v(y) & \text{otherwise} \end{cases}
\end{aligned}
$$

Section 3

Regular Expressions as Predicate Logic Formulas

Regular Expressions and Word Structures

Grammar generating regular expressions

$$R ::= \emptyset \mid \epsilon \mid a \mid R + R \mid R \cdot R \mid R^*$$

where $\emptyset$ is the empty expression, ϵ the empty string, $a \in \Sigma$ a character, $+$ alternation, $\cdot$ concatenation, and $*$ the Kleene star.

Grammar generating star-free regular expressions

$$R ::= \emptyset \mid \epsilon \mid a \mid R + R \mid R \cdot R \mid R \cap R \mid \overline{R}$$

where $\cap$ denotes intersection of languages and $\overline{R}$ the complement of R.

Word structures

A word $w \in \Sigma^*$ using characters in $\Sigma = \{a_1, \ldots, a_n\}$ corresponds to a Σ'-structure $\mathfrak{A}_w$, with $\Sigma' = \{\leq, P_{a_1}, \ldots, P_{a_n}\}$, where

- $\leq$ is a binary relation symbol, and $P_{a_1}, \ldots, P_{a_n}$ are unary relation symbols
- The carrier set of $\mathfrak{A}_w$ is $\{1, 2, \ldots, |w| + 1\}$
- $\leq^{\mathfrak{A}_w} = \{(i, j) \mid 1 \leq i \leq j \leq |w| + 1\}$
- $P_a^{\mathfrak{A}_w}$ is the set of positions character a takes in word w

Expanding on Word Structures

Additional word structure symbols

We extend Σ' with

- Constant symbol m interpreted as 1,
- Constant symbol M interpreted as $|w|+1$,
- Binary relation symbol $\prec$ interpreted as $\{(i, i+1) \mid 1 \leq i \leq |w|\}$.

Axioms on word structures

- Position M is not labelled with any character:

$$\bigwedge_{a \in \Sigma} \neg P_a(M)$$

- Every non-M position is labelled with exactly one character:

$$\forall x \Big(x \neq M \rightarrow \bigvee_{a \in \Sigma} \Big(P_a(x) \wedge \bigwedge_{\substack{b \in \Sigma \\ b \neq a}} \neg P_b(x) \Big) \Big)$$

From Regular Expressions to Predicate Logic

For a word w, $w(i)$ is the i^{th} character of w, and $w[i,j]$ is the subword $w(i)w(i+1)\dots w(j-1)$.

Recursive definition of $\varphi_R(x,y)$ for regular expression R such that

$$\mathfrak{A}_w \models \varphi_R(i/x,\, j/y) \iff w[i,j] \in R$$

- Characters a: $\varphi_a(x,y) := x \prec y \ \wedge \ P_a(x)$
- Empty word: $\varphi_\epsilon(x,y) := x = y$
- Empty language: $\varphi_\emptyset(x,y) := \bot$
- Concatenation: $\varphi_{RS}(x,y) := \exists z \left(x \leq z \ \wedge \ z \leq y \ \wedge \ \varphi_R(x,z) \ \wedge \ \varphi_S(z,y)\right)$
- Alternation: $\varphi_{R+S}(x,y) := \varphi_R(x,y) \vee \varphi_S(x,y)$
- Intersection: $\varphi_{R\cap S}(x,y) := \varphi_R(x,y) \wedge \varphi_S(x,y)$
- Complement: $\varphi_{\overline{R}}(x,y) := \neg\varphi_R(x,y)$

Section 4

Automated Proof Search

Conjunctive Normal Form (Propositional Formulas)

Step 1: Eliminate $\leftrightarrow$ and $\rightarrow$ from the input formula.

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \lor \psi$$

Step 2: Push negation towards literals.

$$\neg\neg\varphi \equiv \varphi \qquad \neg(\varphi \land \psi) \equiv \neg\varphi \lor \neg\psi \qquad \neg\bot \equiv \top$$

$$\neg(\varphi \lor \psi) \equiv \neg\varphi \land \neg\psi \qquad \neg\top \equiv \bot$$

Step 3: Distribute disjunctions into conjunctions.

$$\varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi)$$

$$(\varphi \lor \psi) \land \chi \equiv (\varphi \lor \chi) \land (\psi \lor \chi)$$

Step 4: (optional) Simplify the resulting CNF formula:

- Rewrite clauses with complementary literals (e.g. p and $\neg p$) to $\top$.
- Rewrite $\varphi \land \top \equiv \varphi$, $\quad \varphi \land \bot \equiv \bot$, $\quad \varphi \lor \top \equiv \top$, $\quad \varphi \lor \bot \equiv \varphi$.
- Delete clause C if all literals of another clause C' occur in C (subsumption).

The DPLL Algorithm (Propositional Formulas)

Step 1: Delete all tautological clauses $\{p, \neg p, \dots\}$

Step 2: For each unit clause $\{l\}$:

- Delete all clauses containing l.
- Delete $\neg l$ from all clauses.

Step 3: Delete all clauses that contain pure literals l (i.e. no clause in the clause set contains the negation of l).

Step 4:

- If the empty clause $\square$ is generated, return "No".
- If the clause set $\emptyset$ is generated, return "Yes".

Step 5: Otherwise, pick a literal l and do a **case split** $\Gamma[\top/l]$ and $\Gamma[\bot/l]$ on the clause set Γ, simplifying as follows:

- **Case $\top$:** Delete all clauses containing l, delete $\neg l$ from all clauses.
- **Case $\bot$:** Delete all clauses containing $\neg l$, delete l from all clauses.

Run DPLL recursively on $\Gamma[\top/l]$ and $\Gamma[\bot/l]$.
Return "Yes" if one of the case splits returns "Yes", return "No" otherwise.

Prenex Normal Form (Predicate Formulas)

Step 1: Rename variables such that:

- No variable occurs both bound and free.
- Different occurrences of quantifiers bind different variables.

Step 2: Transform into negation normal form:

- Neither $\leftrightarrow$ nor $\rightarrow$ occur.
- Every negation symbol immediately precedes an atomic formula.

$$\neg\forall x.\ \varphi \equiv \exists x.\ \neg\varphi$$
$$\neg\exists x.\ \varphi \equiv \forall x.\ \neg\varphi$$

Step 3: Pull quantifiers to the top of the syntax tree using the following equivalences:

$$(\forall x.\ \varphi) \wedge \psi \equiv \forall x.\ (\varphi \wedge \psi)$$
$$(\forall x.\ \varphi) \vee \psi \equiv \forall x.\ (\varphi \vee \psi)$$
$$(\exists x.\ \varphi) \wedge \psi \equiv \exists x.\ (\varphi \wedge \psi)$$
$$(\exists x.\ \varphi) \vee \psi \equiv \exists x.\ (\varphi \vee \psi)$$

$$(\forall x.\ \varphi) \wedge (\forall y.\ \psi) \equiv \forall x.\ (\varphi \wedge \psi[x/y])$$
$$(\exists x.\ \varphi) \vee (\exists y.\ \psi) \equiv \exists x.\ (\varphi \vee \psi[x/y])$$

Clause Normal Form (Predicate Formulas)

Step 1: Bring formula into prenex normal form.

Step 2: Skolemise (get rid of existential quantifiers):
For each existentially quantified variable $\exists y$ occuring in the formula

$$\forall x_1 \ldots \forall x_n \exists y.\ \varphi$$

expand the signature with an n-ary function symbol f and transform the formula into

$$\forall x_1 \ldots \forall x_n.\ \varphi[f(x_1, \ldots, x_n)/y].$$

Step 3: Transform into CNF and delete universal quantifiers.

Refutational Theorem Proving

To show that φ is valid:

- Transform $\neg\varphi$ into a CNF-formula ψ.
- Use resolution to derive $\square$ from ψ:

$$\frac{C \vee l \qquad C' \vee \neg l}{C \vee C'}$$

More general resolution (for terms L and L'):
if σ is a substitution such that $L\sigma = L'\sigma$ then:

$$\frac{C \vee L \qquad C' \vee \neg L'}{(C \vee C')\sigma}$$

Unification

Substitution σ is a *unifier* of terms s and t if $s\sigma = t\sigma$.

Unifier σ is the *most general unifier (mgu)* of s and t if for all unifiers ρ of s and t there exists a substitution τ such that $\rho = \tau \circ \sigma$.

By $s \approx t$, we denote that s and t are to be unified.

Unification Algorithm

The unification algorithm finds a most general unifier for a set
$E = \{s_1 \approx t_1, \dots, s_n \approx t_n\}$ of pairs of terms.
We abbreviate $E \cup \{s \approx t\}$ as $E,\ s \approx t$.

$$
\begin{array}{rcll}
E,\ s \approx s & \rightsquigarrow & E & \\
E,\ f(s_1, \dots, s_n) \approx f(t_1, \dots, t_n) & \rightsquigarrow & E,\ s_1 \approx t_1,\ \dots,\ s_n \approx t_n & \\
E,\ f(\dots) \approx g(\dots) & \rightsquigarrow & \bot & \\
E,\ t \approx x & \rightsquigarrow & E,\ x \approx t & \text{if } t \notin \mathcal{V} \\
E,\ x \approx t & \rightsquigarrow & \bot & \text{if } x \neq t,\ x \in V(t) \\
E,\ x \approx t & \rightsquigarrow & E[t/x],\ x \approx t & \text{if } x \in V(E),\ x \notin V(t)
\end{array}
$$

Resolution

We can show validity via refutational theorem proving by using the resolution rule

$$\frac{C \lor L \qquad C' \lor \neg L'}{(C \lor C')\sigma} \qquad \sigma = \mathrm{mgu}(L, L')$$

and the factoring rule

$$\frac{C \lor L \lor L'}{(C \lor L)\sigma} \qquad \sigma = \mathrm{mgu}(L, L')$$

Section 5

Linear Temporal Logic

Labelled Transition Systems

Labelled transition system (LTS): a structure $(S, \to, \lambda)$ over a set P of propositional variables where

- S is a finite set of states,
- $\to \subseteq S \times S$ is a binary *transition relation*,
- $\lambda : S \to \mathcal{P}(P)$ is a *labelling function* that maps states to sets of propositional variables.

We require that for all $s \in S$ there exists $s' \in S$ such that $s \to s'$.

Path: a sequence $\pi \colon \mathbb{N} \to S$ such that $\pi(i) \to \pi(i+1)$ for all $i \in \mathbb{N}$.

We write

- $\pi = t_0\, t_1\, t_2 \ldots$ for a path,
- $\pi^i = t_i\, t_{i+1}\, t_{i+2} \ldots$ for the i^{th} suffix of π.

Linear Temporal Logic: Syntax

Linear temporal logic: Formulas are defined by the syntax

$$\Phi ::= \bot \mid \top \mid p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \to \Phi \mid X\Phi \mid F\Phi \mid G\Phi \mid \Phi U \Phi \mid \Phi W \Phi$$

Connective	Pronunciation	Meaning on a path π
$X\varphi$	next φ	φ holds on π^1
$F\varphi$	eventually φ	φ holds on π^k, for some k
$G\varphi$	globally φ	φ holds on π^k, for all k
$\varphi U \psi$	φ until ψ	ψ holds on π^k, for some k, and φ holds on π^i, for all $i < k$
$\varphi W \psi$	φ weak until ψ	Either $\varphi U \psi$ or $G\varphi$.

Linear Temporal Logic: Semantics

Satisfiability relation: $\pi \models \varphi$ defined by

- $\pi \not\models \bot$ and $\pi \models \top$
- $\pi \models p \;\Leftrightarrow\; p \in \lambda(s_0)$
- $\pi \models \neg\varphi \;\Leftrightarrow\; \pi \not\models \varphi$
- $\pi \models \varphi \wedge \psi \;\Leftrightarrow\; \pi \models \varphi$ and $\pi \models \psi$
- $\pi \models \varphi \vee \psi \;\Leftrightarrow\; \pi \models \varphi$ or $\pi \models \psi$
- $\pi \models \varphi \rightarrow \psi \;\Leftrightarrow\; \pi \models \psi$ whenever $\pi \models \varphi$

- $\pi \models X\varphi \;\Leftrightarrow\; \pi^1 \models \varphi$
- $\pi \models F\varphi \;\Leftrightarrow\; \pi^k \models \varphi$ for some $k \in \mathbb{N}$
- $\pi \models G\varphi \;\Leftrightarrow\; \pi^k \models \varphi$ for all $k \in \mathbb{N}$
- $\pi \models \varphi U \psi \;\Leftrightarrow\; \pi^k \models \psi$ for some $k \in \mathbb{N}$ and $\pi^i \models \varphi$ for all $0 \leq i < k$
- $\pi \models \varphi W \psi \;\Leftrightarrow\; \pi^k \models \psi$ for some $k \in \mathbb{N}$ and $\pi^i \models \varphi$ for all $0 \leq i < k$,
  or $\pi^j \models \varphi$ for all $j \in \mathbb{N}$

Equivalence and Model Checking

Equivalence: φ and ψ are *equivalent*, $\varphi \equiv \psi$
if $\pi \models \varphi \Leftrightarrow \pi \models \psi$ holds for all paths π of all LTS's $\mathfrak{G}$.

Some equivalences:

- $\neg X\varphi \equiv X\neg\varphi$
- $\neg G\varphi \equiv F\neg\varphi$
- $\neg F\varphi \equiv G\neg\varphi$
- $F(\varphi \vee \psi) \equiv F\varphi \vee F\psi$
- $G(\varphi \wedge \psi) \equiv G\varphi \wedge G\psi$
- $F\varphi \equiv \top U\varphi$
- $G\varphi \equiv \neg(\top U\neg\varphi)$
- $\varphi U\psi \equiv \psi \vee (\varphi \wedge X(\varphi U\psi))$

Model Checking Problem: $\mathfrak{G}, s \models \varphi$
is the problem of deciding whether $\pi \models \varphi$ for all paths π of $\mathfrak{G}$ with $s = s_0$.