Artificial Intelligence Notes

The Foundations: Logic & Proofs

Rules of logic give precise meaning to mathematical statements. These rules are used to distinguish between valid & invalid mathematical arguments.

Propositions — Basic building block of logic. A proposition is a declarative sentence (a sentence that declares a fact) that is either true or false, but not both.

propositional var. ⇒ use letters, e.g. - p, q, r, s

- truth val. ⇒ T
- false val. ⇒ F

Propositional Calculus or logic — The area of logic that deals with propositions.

Compound propositions — New propositions that are formed from existing propositions using logical operators.

Examples of compound propositions

01. The negation of p ⇒ $\neg p$ / $\bar{p}$ (use negation operator)
02. Conjunction of p & q ⇒ $p \wedge q$ (use connective operator) (TT→T, F)
03. Disjunction of p & q ⇒ $p \vee q$ (FF→F, T) [inclusive or]
04. Exclusive or ⇒ $p \oplus q$ (TF/FT→T, F)
05. Conditional statement ⇒ $p \to q$ (TF→F, T) [implication]

here, p ⇒ hypothesis / antecedent / premise
q ⇒ conclusion / consequence

"if p, then q"	"p implies q"
"if p, q"	"p only if q"
"p is sufficient for q"	"a sufficient condition for q is p"
"q if p"	"q whenever p"
"q when p"	"q is necessary for p"
"a necessary condition for p is q"	"q follows from p"
"q unless $\neg p$"	

* Converse : $q \to p$
* Contrapositive : $\neg q \to \neg p$
* Inverse : $\neg p \to \neg q$

Conditional statement ≡ Contrapositive
Converse ≡ Inverse

Biconditional Statement — bi-implication: $p \leftrightarrow q$ (TT/FF, T)
* p if and only if q ; $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\to$	4
$\leftrightarrow$	5

Precedence of Logical operators

TABLE 9 Table for the Bit Operators *OR*, *AND*, and *XOR*.

x	y	$x \vee y$	$x \wedge y$	$x \oplus y$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

* bitwise operation

Propositional Equivalences

Tautology: A compound proposition that is always true. [$p \vee \neg p$]

Contradiction: A compound proposition that is always false. [$p \wedge \neg p$]

Contingency: A compound proposition that is neither a tautology nor a contradiction.

Logically equivalent: Compound propositions, $p \equiv q$ if $p \leftrightarrow q$ is a tautology
↪ $p \Longleftrightarrow q$

TABLE 6 Logical Equivalences.

Equivalence	*Name*
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

TABLE 7 Logical Equivalences Involving Conditional Statements.

$p \rightarrow q \equiv \neg p \vee q$
$p \rightarrow q \equiv \neg q \rightarrow \neg p$
$p \vee q \equiv \neg p \rightarrow q$
$p \wedge q \equiv \neg(p \rightarrow \neg q)$
$\neg(p \rightarrow q) \equiv p \wedge \neg q$
$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$
$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

TABLE 8 Logical Equivalences Involving Biconditional Statements.

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

* De Morgan's Laws extend to ⇒

$$\neg(p_1 \vee p_2 \vee \cdots \vee p_n) \equiv (\neg p_1 \wedge \neg p_2 \wedge \cdots \wedge \neg p_n)$$

$$\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \equiv (\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n)$$

* **Satisfiable:** a compound proposition is satisfiable if there is an assignment of truth val to its var. that makes it true.

Unsatisfiable: when compound proposition is false for all assignments of truth val to its var., the proposition is unsatisfiable.

Solution: solution is a particular assignment of truth val that makes compound proposition true.

Predicates & Quantifiers

Predicates: statement "x is greater than y" ⇒ denotes by $P(x)$

- subject of the statement
- Predicate - refers to a property that sub of statement can have
- propositional func^n P at x

* once a val has been assigned to var. x, statement $P(x)$ becomes a proposition and has a truth val.
* statement of the form $P(x_1, x_2, \dots, x_n)$ is the val of propositional function P at n-tuple $(x_1, x_2, \dots, x_n)$
  P is called an n-place predicates or n-ary predicate

if $x > 0$, then $x := x + 1$

Precondition — Post Condition

Quantifiers:

quantification: propositional function to proposition.
it expresses the extent to which a predicate is true over a range of elements. associated english word: all, some, many, none, few

universal quantification: it will tell that a predicate is true for every element under consideration.

existential quantification: it will tell that there is one or more element under consideration for which predicate is true.

predicate calculus: area of logic that deals ⇒ predicates + quantifiers.

* Universal quantifier: domain of discourse / universe of discourse / domain

$\forall x\, P(x)$ ⇒ $P(x)$ for all values of x in the domain

($\forall$: universal quantifier)

* an element for which $P(x)$ is false is called a counterexample of $\forall x\, P(x)$

* Existential quantifier:

$\exists x\, P(x)$ ⇒ there exists an element x in the domain such that $P(x)$
/ there is an x such that $P(x)$ / for some x $P(x)$
/ there is at least one x such that $P(x)$

($\exists$: existential quantifier)

* a domain must always be specified when a statement $\exists x\, P(x)$ used. The meaning of $\exists x\, P(x)$ changes when domain changes.
* $\exists x\, P(x)$ is false iff $P(x)$ is false for every elements of domain.

* elements of domain ⇒ $x_1, x_2, x_3, \ldots, x_n$

universal quantification $\forall x\, P(x) \equiv P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \ldots \wedge P(x_n)$ — conjunction

existential quantification $\exists x\, P(x) \equiv P(x_1) \vee P(x_2) \vee P(x_3) \vee \ldots \vee P(x_n)$ — disjunction

* Uniqueness quantifier:

$\exists! x\, P(x)$ or $\exists_1 x\, P(x)$ ⇒ there is one and only one x, $P(x)$

* Quantifier with restricted domain:

$\forall x<0\,(x^2>0) \equiv \forall x\,(x<0 \rightarrow x^2>0)$ → conditional statement

$\exists x>0\,(x^2=2) \equiv \exists x\,(x>0 \wedge x^2=2)$ → conjunction

* Precedence of quantifiers: ∀ & ∃ have higher precedence than all logical operators from propositional calculus.

* Binding var: $\exists x\,(x+y=1)$ ⇒ here x is bound, y is free

scope ⇒ $\exists x\,(P(x) \wedge Q(x)) \vee \forall x\, R(x)$

here, scope of $\exists x$ is the expression $P(x) \wedge Q(x)$, not to the rest.

* Logically Equivalence involving quantifiers: $S \equiv T$ → two statements with predicates and quantifiers and have same truth val

$\forall x\,(P(x) \wedge Q(x)) \equiv \forall x\, P(x) \wedge \forall x\, Q(x)$

TABLE 2 De Morgan's Laws for Quantifiers.

Negation	Equivalent Statement	When Is Negation True?	When False?
$\neg\exists x\, P(x)$	$\forall x \neg P(x)$	For every x, $P(x)$ is false.	There is an x for which $P(x)$ is true.
$\neg\forall x\, P(x)$	$\exists x \neg P(x)$	There is an x for which $P(x)$ is false.	$P(x)$ is true for every x.

e.g. $\neg\forall x\,(P(x) \rightarrow Q(x)) \equiv \exists x\,(P(x) \wedge \neg Q(x))$

here, $\neg\forall(x)\,(P(x) \rightarrow Q(x))$

$\equiv \exists x\, \neg(P(x) \rightarrow Q(x))$ [De Morgan's Laws for quantifiers]

$\equiv \exists x\, \neg(\neg P(x) \vee Q(x))$ [$p \rightarrow q \equiv \neg p \vee q$]

$\equiv \exists x\, \neg(\neg P(x)) \wedge \neg Q(x)$ [De Morgan's Laws for propositional logic]

$\equiv \exists x\,(P(x) \wedge \neg Q(x))$ [double negation law]

Nested Quantifiers

$\forall x \exists y\ (x+y=0)$ ⇒ Here, $\forall x\ Q(x)$

Nested quantifiers

$Q(x)$ is $\exists y\ P(x,y)$

$P(x,y)$ is $x+y=0$

Order ⇒ $\forall x \forall y\ P(x,y) \equiv \forall y \forall x\ P(x,y)$

matters $\exists y \forall x\ Q(x,y) \neq \forall x \exists y\ Q(x,y)$

TABLE 1 Quantifications of Two Variables.

Statement	*When True?*	*When False?*
$\forall x \forall y P(x, y)$ $\forall y \forall x P(x, y)$	$P(x, y)$ is true for every pair x, y.	There is a pair x, y for which $P(x, y)$ is false.
$\forall x \exists y P(x, y)$	For every x there is a y for which $P(x, y)$ is true.	There is an x such that $P(x, y)$ is false for every y.
$\exists x \forall y P(x, y)$	There is an x for which $P(x, y)$ is true for every y.	For every x there is a y for which $P(x, y)$ is false.
$\exists x \exists y P(x, y)$ $\exists y \exists x P(x, y)$	There is a pair x, y for which $P(x, y)$ is true.	$P(x, y)$ is false for every pair x, y.

negating nested quantifiers

$\neg \forall x \exists y\ (xy=1)$

$\equiv \exists x \neg \exists y\ (xy=1)$

[De Morgan's Law]

$\equiv \exists x \forall y \neg (xy=1)$

[De Morgan's Law]

$\equiv \exists x \forall y\ (xy \neq 1)$

Rules of Inference

Argument: seq. of proposition

p_1
p_2
$\vdots$
p_n
$\therefore q$ → Conclusion

Premises

Valid argument iff $(p_1 \wedge p_2 \wedge \dots \wedge p_n) \rightarrow q$ is tautology

* conclusion = true iff all premises = true

Example: premises $\neg p \wedge q$, $r \rightarrow p$, $\neg r \rightarrow s$, $s \rightarrow t$

conclusion t

Step	Reason
1. $\neg p \wedge q$	premise
2. $\neg p$	simplification using (1)
3. $r \rightarrow p$	premise
4. $\neg r$	Modus tollens using (2), (3)
5. $\neg r \rightarrow s$	premise
6. s	Modus Ponens using (4), (5)
7. $s \rightarrow t$	premise
8. t	Modus ponens using (6), (7)

TABLE 1 Rules of Inference.

Rule of Inference	*Tautology*	*Name*
p $p \to q$ $\therefore q$	$(p \wedge (p \to q)) \to q$	Modus ponens
$\neg q$ $p \to q$ $\therefore \neg p$	$(\neg q \wedge (p \to q)) \to \neg p$	Modus tollens
$p \to q$ $q \to r$ $\therefore p \to r$	$((p \to q) \wedge (q \to r)) \to (p \to r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$((p \vee q) \wedge \neg p) \to q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \to (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \to p$	Simplification
p q $\therefore p \wedge q$	$((p) \wedge (q)) \to (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \to (q \vee r)$	Resolution

* Resolution:

$$\frac{p \vee q \qquad \neg p \vee r}{q \vee r} \rightarrow \text{resolvent}$$

* Let $q = r$ in Resolution tautology,

$((p \vee q) \wedge (\neg p \vee q)) \to (q \vee q)$

$\equiv (p \vee q) \wedge (\neg p \vee q) \to q$ ✓

* let $r = F$ in Resolution tautology,

$((p \vee q) \wedge (\neg p \vee F)) \to (q \vee F)$

$\equiv (p \vee q) \wedge \neg p \to q$ [Disjunctive syllogism]

*** To construct proofs in propositional logic using resolution as the only rule of inference, the hypotheses and conclusion must be expressed as clauses, where a clause is a disjunction of var. or negations of these var.

Replace propositional statement into clause ⇒ e.g. $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ → 2 clauses

$\neg(p \vee q) \equiv \neg p \wedge \neg q$

$p \to q \equiv \neg p \vee q$

* Fallacies:

Fallacy of affirming the conclusion ⇒ $((p \to q) \wedge q) \to p$ is not tautology [$p = F, q = T$]

Fallacy of denying the hypothesis ⇒ $((p \to q) \wedge \neg p) \to \neg q$ is not tautology [$p = F, q = T$]

* Rules of inference for quantified statements:

TABLE 2 Rules of Inference for Quantified Statements.

Rule of Inference	*Name*
$\forall x\, P(x)$ $\therefore\ P(c)$	Universal instantiation
$P(c)$ for an arbitrary c $\therefore\ \forall x\, P(x)$	Universal generalization
$\exists x\, P(x)$ $\therefore\ P(c)$ for some element c	Existential instantiation
$P(c)$ for some element c $\therefore\ \exists x\, P(x)$	Existential generalization

⇒ $P(c)$ ⇒ c is a particular member of the domain

⇒ element c that select must be arbitrary & not a specific

e.g. Show that the premises "A student in this class has not read the book," and "Everyone in this class passed the first exam" imply the conclusion "Someone who passed the first exam has not read the book."

Solution: Let $C(x)$ be "x is in this class," $B(x)$ be "x has read the book," and $P(x)$ be "x passed the first exam." The premises are $\exists x(C(x) \wedge \neg B(x))$ and $\forall x(C(x) \to P(x))$. The conclusion is $\exists x(P(x) \wedge \neg B(x))$. These steps can be used to establish the conclusion from the premises.

Step	Reason
1. $\exists x(C(x) \wedge \neg B(x))$	Premise
2. $C(a) \wedge \neg B(a)$	Existential instantiation from (1)
3. $C(a)$	Simplification from (2)
4. $\forall x(C(x) \to P(x))$	Premise
5. $C(a) \to P(a)$	Universal instantiation from (4)
6. $P(a)$	Modus ponens from (3) and (5)
7. $\neg B(a)$	Simplification from (2)
8. $P(a) \wedge \neg B(a)$	Conjunction from (6) and (7)
9. $\exists x(P(x) \wedge \neg B(x))$	Existential generalization from (8)

Universal modus ponens

$\forall x(P(x) \to Q(x))$

$P(a)$, where a is a particular element in the domain

$\therefore\ Q(a)$

Universal modus tollens

$\forall x(P(x) \to Q(x))$

$\neg Q(a)$, where a is a particular element in the domain

$\therefore\ \neg P(a)$

Introduction to Proofs

a proof is a valid argument that establishes the truth of a mathematical statement.

formal proofs → where all steps were supplied & rules for each step in arg. given

informal proofs → where more than one rule of inference may be used in each step, where steps may be skipped, where axioms being assumed, rules of inference used are not explicitly stated.

Theorem ⇒ it is a statement that can be shown to be true.

↳ less important theorems ⇒ propositions

Proof ⇒ a proof is a valid argument that establishes truth of a theorem

Axioms/Postulates ⇒ which are statements we assume to be true.
it may be stated using primitive terms that do not req. defn.

Lemma ⇒ a less important theorem that is helpful in the proof of other results

Corollary ⇒ it is a theorem that can be established directly from a theorem that has been proved individually.

Conjecture ⇒ it is a statement that is being proposed to be true statement, usually on the basis of partial evidance, a heuristic arg., or the intuition of an expert.

* Methods of proving theorems:

Direct proofs ⇒ assume hypothesis of the conditional statement is true.
direct proofs lead from the pemises of a theorem to conclusion

Proofs by Contra-position:
- Indirect proofs
  - ↳ Proof by Contraposition ⇒ use $p \to q \equiv \neg q \to \neg p$
- Vacuous proof ⇒ $p \to q$ true, if p false ⇒ we have proof
- Trivial proof ⇒ $p \to q$ true, if q true

Proof by Contradiction ⇒ assume $\neg p$ is true, and leads to a contradiction
↳ indirect proofs

Proofs of equivalence ⇒ $p \leftrightarrow q$

Chap-12

Propositional Logic & Reasoning

Syntax: formulae A in logic, e.g.: W, $W \to S$

Semantics: which formulae A is true under which assignment φ, written $\varphi \models A$

$\varphi := \{W \mapsto T, S \mapsto F\}$, then $\varphi \models W$ but $\varphi \not\models (W \to S)$

Entailment: $A \models B$, B are entailed by A, means for all φ with $\varphi \models A$, also $\varphi \models B$

Deduction: $A \vdash_C B$, statement B can be derived from A using a set of C inference rule.

* Soundness: when $A \vdash_C B$, we also have $A \models B$
* Completeness: when $A \models B$, we also have $A \vdash_C B$

Propositional Logic (PL^0)

propositional var, $\mathcal{V}_0 := \{P, Q, P', P^2, \ldots\}$

connectives, $\Sigma_0 := \{T, F, \neg, \vee, \wedge, \ldots\}$

$\neg A$, $A \wedge B$, $A \vee B$, $A \to B$, $A \leftrightarrow B$

atomy: propositional formulae without connectives

Complex: not atomy

* wff_0 - well-formed propositional formulae

Semantics of PL^0 (Models)

model $M := \langle D_0, I \rangle$

Universe $D_0 = \{T, F\}$ Fixed

Interpretation I

- $I(\neg) : D_0 \to D_0$; $T \to F$, $F \to T$
- $I(\wedge) : D_0 \times D_0 \to D_0$

var. assign. $\varphi : \mathcal{V}_0 \to D_0$

val. fun. $\mathcal{I}_\varphi : wff_0(\mathcal{V}_0) \to D_0$

$\mathcal{I}_\varphi(P) = \varphi(P)$

$\mathcal{I}_\varphi(\neg A) = I(\neg)(\mathcal{I}_\varphi(A))$

Truth table

Semantic Properties of Propositional Formulae

Definition 12.2.9. Let $\mathcal{M} := \langle \mathcal{U}, \mathcal{I} \rangle$ be our model, then we call $\mathbf{A}$

- ▷ true under φ (φ satisfies $\mathbf{A}$) in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = T$
- ▷ false under φ (φ falsifies $\mathbf{A}$) in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = F$
- ▷ satisfiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = T$ for some assignment φ
- ▷ valid in $\mathcal{M}$, iff $\mathcal{M} \models^\varphi \mathbf{A}$ for all assignments φ
- ▷ falsifiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = F$ for some assignments φ
- ▷ unsatisfiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = F$ for all assignments φ

* valid formulae is called tautology
* Entailment: $A \models B$
  A entails B
  iff $\mathcal{I}_\varphi(B) = T$ for all φ with $\mathcal{I}_\varphi(A) = T$

* First order signature:
  function constants: $\Sigma_K^f = \{f, g, h, \ldots\}$ - denoting function on individuals
  predicate constants: $\Sigma_K^P = \{P, Q, M, \ldots\}$ - denoting relationship among individuals.

The formulae of PL^{nq} are given by the following grammar

functions	f^k	$\in$	Σ_k^f	
predicates	p^k	$\in$	Σ_k^p	
terms	t	$::=$	X	variable
		$\mid$	f^0	constant
		$\mid$	$f^k(t_1,\ldots,t_k)$	application
formulae	$\mathbf{A}$	$::=$	$p^k(t_1,\ldots,t_k)$	atomic
		$\mid$	$\neg\mathbf{A}$	negation
		$\mid$	$\mathbf{A}_1 \wedge \mathbf{A}_2$	conjunction

The value function $\mathcal{I}$ assigns values to formulae

- $\mathcal{I}(f(\mathbf{A}^1,\ldots,\mathbf{A}^k)) := \mathcal{I}(f)(\mathcal{I}(\mathbf{A}^1),\ldots,\mathcal{I}(\mathbf{A}^k))$
- $\mathcal{I}(p(\mathbf{A}^1,\ldots,\mathbf{A}^k)) := \mathsf{T}$, iff $\langle \mathcal{I}(\mathbf{A}^1),\ldots,\mathcal{I}(\mathbf{A}^k)\rangle \in \mathcal{I}(p)$
- $\mathcal{I}(\neg\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}(\mathbf{A}))$ and $\mathcal{I}(\mathbf{A}\wedge\mathbf{B}) = \mathcal{I}(\wedge)(\mathcal{I}(\mathbf{A}),\mathcal{I}(\mathbf{G}))$

PL^{nq} semantics:

$D_i \neq \emptyset$ of individuals,

$\mathcal{I}: \Sigma_0^f \to D_i$ individual const. → individuals

$\mathcal{I}: \Sigma_k^f \to D_i^k \to D_i$ func^n as func^n

$\mathcal{I}: \Sigma_k^P \to \mathcal{P}(D_i^k)$ predicate as arbitrary relation

PL^{nq} & PL^{0} isomorphic !?

PL with Hilbert Calculus H^0 !?

Natural Deduction Calculus: ND_0

Introduction

$$\frac{A \quad B}{A\wedge B}\wedge I$$

$$\frac{\begin{array}{c}[A]^1\\ \vdots\\ B\end{array}}{A\to B}\to I^1$$

↳ Assumption (A), then derived B

A can not be use again in proof.

$$\frac{A}{A\vee B}\vee I_l \qquad \frac{B}{A\vee B}\vee I_r$$

$$\frac{\begin{array}{c}[A]^1\\ \vdots\\ F\end{array}}{\neg A}\neg I^1$$

$$\frac{\neg A \quad A}{F}FI$$

Elimination

$$\frac{A\wedge B}{A}\wedge E_l \qquad \frac{A\wedge B}{B}\wedge E_r$$

$$\frac{A\to B \quad A}{B}\to E$$

$$\frac{A\vee B \quad \begin{array}{cc}[A]^1 & [B]^1\\ \vdots & \vdots\\ C & C\end{array}}{C}\vee E^1$$

$$\frac{\neg\neg A}{A}\neg E$$

$$\frac{F}{A}FE$$

Axiom

$$\frac{}{A\vee\neg A}TND$$

ND_0 (sequent style version):

Deduction theorem for ND_0: $H, A \vdash_{ND_0} B$, iff $H \vdash_{ND_0} A \to B$

$$\frac{}{\Gamma,\mathbf{A}\vdash\mathbf{A}}Ax \qquad \frac{\Gamma\vdash\mathbf{B}}{\Gamma,\mathbf{A}\vdash\mathbf{B}}weaken \qquad \frac{}{\Gamma\vdash\mathbf{A}\vee\neg\mathbf{A}}TND$$

$$\frac{\Gamma\vdash\mathbf{A}\quad\Gamma\vdash\mathbf{B}}{\Gamma\vdash\mathbf{A}\wedge\mathbf{B}}\wedge I \qquad \frac{\Gamma\vdash\mathbf{A}\wedge\mathbf{B}}{\Gamma\vdash\mathbf{A}}\wedge E_l \qquad \frac{\Gamma\vdash\mathbf{A}\wedge\mathbf{B}}{\Gamma\vdash\mathbf{B}}\wedge E_r$$

$$\frac{\Gamma\vdash\mathbf{A}}{\Gamma\vdash\mathbf{A}\vee\mathbf{B}}\vee I_l \qquad \frac{\Gamma\vdash\mathbf{B}}{\Gamma\vdash\mathbf{A}\vee\mathbf{B}}\vee I_r \qquad \frac{\Gamma\vdash\mathbf{A}\vee\mathbf{B}\quad\Gamma,\mathbf{A}\vdash\mathbf{C}\quad\Gamma,\mathbf{B}\vdash\mathbf{C}}{\Gamma\vdash\mathbf{C}}\vee E$$

$$\frac{\Gamma,\mathbf{A}\vdash\mathbf{B}}{\Gamma\vdash\mathbf{A}\Rightarrow\mathbf{B}}\Rightarrow I \qquad \frac{\Gamma\vdash\mathbf{A}\Rightarrow\mathbf{B}\quad\Gamma\vdash\mathbf{A}}{\Gamma\vdash\mathbf{B}}\Rightarrow E$$

$$\frac{\Gamma,\mathbf{A}\vdash F}{\Gamma\vdash\neg\mathbf{A}}\neg I \qquad \frac{\Gamma\vdash\neg\neg\mathbf{A}}{\Gamma\vdash\mathbf{A}}\neg E$$

$$\frac{\Gamma\vdash\neg\mathbf{A}\quad\Gamma\vdash\mathbf{A}}{\Gamma\vdash F}FI \qquad \frac{\Gamma\vdash F}{\Gamma\vdash\mathbf{A}}FE$$

Linearized notation for sequent-style ND proofs

1. $\mathcal{H}_1 \vdash \mathbf{A}_1 \quad (\mathcal{J}_1)$
2. $\mathcal{H}_2 \vdash \mathbf{A}_2 \quad (\mathcal{J}_2)$
3. $\mathcal{H}_3 \vdash \mathbf{A}_3 \quad (\mathcal{J}_3 1, 2)$

correspond to,

$$\frac{\mathcal{H}_1\vdash\mathbf{A}_1\quad\mathcal{H}_2\vdash\mathbf{A}_2}{\mathcal{H}_3\vdash\mathbf{A}_3}\mathcal{R}$$

Chap 13

Machine Oriented Calculi for PL

* Unsatisfiability theorem: $H \models A$ iff $H \cup \{\neg A\}$ is unsatisfiable

Normal Forms:

▷ **Definition 13.1.6.**

A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions of literals: i.e. if it is of the form $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} l_{ij}$

▷ **Definition 13.1.7.**

A formula is in disjunctive normal form (DNF) if it is a disjunction of conjunctions of literals: i.e. if it is of the form $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} l_{ij}$

Test Calculi: Tableaux

Formulae < atomic / complex | labeled formulae A^α, if $\alpha \in \{T, F\}$

labeled atom A^α < pos if $\alpha = T$ / neg if $\alpha = F$

opposite or partner literal A^α, B^β if $\alpha \neq \beta$

inference rules of propositional tableau calculus T_0

$$\frac{(A \wedge B)^T}{\begin{array}{l} A^T \\ B^T \end{array}}\, T_0\wedge \qquad \frac{(A \wedge B)^F}{A^F \mid B^F}\, T_0\vee \qquad \frac{\neg A^T}{A^F}\, T_0\neg^T \qquad \frac{\neg A^F}{A^T}\, T_0\neg^F$$

$$\frac{\begin{array}{l} A^\alpha \\ A^\beta \quad \alpha \neq \beta \end{array}}{\bot}\, T_0\bot$$

(→ contradiction)

tableau < saturated: iff no rule add new materials & a branch closed, else open.

closed: iff all of its branches are closed

* T_0-theorem / Derivability: A is a T_0-theorem ($\vdash_{T_0} A$), iff there is a closed tableau with A^F at the root.

Derived rules of inference:

$$\frac{(A \Rightarrow B)^T}{A^F \mid B^T} \qquad \frac{(A \Rightarrow B)^F}{\begin{array}{l} A^T \\ B^F \end{array}} \qquad \frac{\begin{array}{l} A^T \\ (A \Rightarrow B)^T \end{array}}{B^T} \qquad \frac{(A \vee B)^T}{A^T \mid B^T} \qquad \frac{(A \vee B)^F}{\begin{array}{l} A^F \\ B^F \end{array}} \qquad \frac{A \Leftrightarrow B^T}{\begin{array}{l|l} A^T & A^F \\ B^T & B^F \end{array}} \qquad \frac{A \Leftrightarrow B^F}{\begin{array}{l|l} A^T & A^F \\ B^F & B^T \end{array}}$$

* Soundness: T_0 is sound, i.e. $\Phi \subseteq wff_0(\mathcal{V}_0)$ valid, if there is a closed tableau T for Φ^F
* Completeness: T_0 is complete, i.e. if $\Phi \subseteq wff_0(\mathcal{V}_0)$ is valid, then there is a closed tableau T for Φ^F
* Termination: T_0 terminates, i.e. every T_0 tableau becomes saturated after finitely many rule applications.

- tableau calculus basically computes disjunctive normal form (DNF), every branch is a disjunct that is conjunction of literals.

Resolution for PL:

inference rule for resol^n calculus R_0

$$\frac{P^T \vee A \quad P^F \vee B}{A \vee B}\,R$$

P^T, P^F } Cut literals

▷ **Definition 13.5.5 (Transformation into Clause Normal Form).** The CNF transformation calculus CNF_0 consists of the following four inference rules on sets of labeled formulae.

* clause C

$$\frac{\mathbf{C} \vee (\mathbf{A} \vee \mathbf{B})^T}{\mathbf{C} \vee \mathbf{A}^T \vee \mathbf{B}^T} \quad \frac{\mathbf{C} \vee (\mathbf{A} \vee \mathbf{B})^F}{\mathbf{C} \vee \mathbf{A}^F ; \mathbf{C} \vee \mathbf{B}^F} \qquad \frac{\mathbf{C} \vee \neg\mathbf{A}^T}{\mathbf{C} \vee \mathbf{A}^F} \quad \frac{\mathbf{C} \vee \neg\mathbf{A}^F}{\mathbf{C} \vee \mathbf{A}^T}$$

derived rules of inference

$$\frac{\mathbf{C} \vee (\mathbf{A} \Rightarrow \mathbf{B})^T}{\mathbf{C} \vee \mathbf{A}^F \vee \mathbf{B}^T} \quad \frac{\mathbf{C} \vee (\mathbf{A} \Rightarrow \mathbf{B})^F}{\mathbf{C} \vee \mathbf{A}^T ; \mathbf{C} \vee \mathbf{B}^F} \qquad \frac{\mathbf{C} \vee (\mathbf{A} \wedge \mathbf{B})^T}{\mathbf{C} \vee \mathbf{A}^T ; \mathbf{C} \vee \mathbf{B}^T} \quad \frac{\mathbf{C} \vee (\mathbf{A} \wedge \mathbf{B})^F}{\mathbf{C} \vee \mathbf{A}^F \vee \mathbf{B}^F}$$

Chap 14

Formal Systems: Syntax, Semantics, Entailment & Derivation in General

Let $\langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, $\mathcal{M} \in \mathcal{K}$ be a model and $\mathbf{A} \in \mathcal{L}$ a formula, then we say that $\mathbf{A}$ is

- ▷ satisfied by $\mathcal{M}$, iff $\mathcal{M} \models \mathbf{A}$.
- ▷ falsified by $\mathcal{M}$, iff $\mathcal{M} \not\models \mathbf{A}$.
- ▷ satisfiable in $\mathcal{K}$, iff $\mathcal{M} \models \mathbf{A}$ for some $\mathcal{M} \in \mathcal{K}$.
- ▷ valid in $\mathcal{K}$ (write $\models \mathcal{M}$), iff $\mathcal{M} \models \mathbf{A}$ for all $\mathcal{M} \in \mathcal{K}$.
- ▷ falsifiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for some $\mathcal{M} \in \mathcal{K}$.
- ▷ unsatisfiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for all $\mathcal{M} \in \mathcal{K}$.

▷ **Definition 14.0.12.**

An inference rule $\mathcal{I}$ is called admissible in a calculus $\mathcal{C}$, if the extension of $\mathcal{C}$ by $\mathcal{I}$ does not yield new theorems.

▷ **Definition 14.0.13.** An inference rule $\frac{\mathbf{A}_1 \quad \ldots \quad \mathbf{A}_n}{\mathbf{C}}$ is called derivable (or a derived rule) in a calculus $\mathcal{C}$, if there is a $\mathcal{C}$ derivation $\mathbf{A}_1, \ldots, \mathbf{A}_n \vdash_{\mathcal{C}} \mathbf{C}$.

▷ **Observation 14.0.14.** *Derivable inference rules are admissible, but not the other way around.*

Chap 15

Propositional Reasoning: SAT Solvers

▷ **Definition 15.1.1.** The SAT problem (SAT): Given a propositional formula $\mathbf{A}$, decide whether or not $\mathbf{A}$ is satisfiable. We denote the class of all SAT problems with SAT

▷ The SAT problem was the first problem proved to be **NP**-complete!

▷ $\mathbf{A}$ is commonly assumed to be in CNF. This is without loss of generality, because any $\mathbf{A}$ can be transformed into a satisfiability-equivalent CNF formula (cf. chapter 12) in polynomial time.

SAT vs. CSP

▷ **Observation 15.1.3 (SAT: A kind of CSP).** *SAT can be viewed as a CSP problem in which all variable domains are Boolean, and the constraints have unbounded arity.*

▷ **Theorem 15.1.4 (Encoding CSP as SAT).** *Given any constraint network $\mathcal{C}$, we can in low order polynomial time construct a CNF formula $\mathbf{A}(\mathcal{C})$ that is satisfiable iff $\mathcal{C}$ is solvable.*

* Davis-Putnam (Logemann-Loveland) Procedure (DPLL):
  - How to systematically test satisfiability
  - a restricted form of resolution

DPLL (Vanilla Example):

▷ **Example 15.2.2 (UP and Splitting).** Let $\Delta := (P^{\mathsf{T}} \vee Q^{\mathsf{T}} \vee R^{\mathsf{F}}; P^{\mathsf{F}} \vee Q^{\mathsf{F}}; R^{\mathsf{T}}; P^{\mathsf{T}} \vee Q^{\mathsf{F}})$

1. UP Rule: $R \mapsto \mathsf{T}$
   $P^{\mathsf{T}} \vee Q^{\mathsf{T}}\,;\ P^{\mathsf{F}} \vee Q^{\mathsf{F}}\,;\ P^{\mathsf{T}} \vee Q^{\mathsf{F}}$
2. Splitting Rule:
   - 2a. $P \mapsto \mathsf{F}$
     $Q^{\mathsf{T}}\,;\ Q^{\mathsf{F}}$
     - 3a. UP Rule: $Q \mapsto \mathsf{T}$
       $\square$
       returning "unsatisfiable"
     - 3b. UP Rule: $Q \mapsto \mathsf{F}$
       clause set empty
       returning "$R \mapsto \mathsf{T}, P \mapsto \mathsf{T}, Q \mapsto \mathsf{F}$
   - 2b. $P \mapsto \mathsf{T}$
     Q^{F}

* DPLL $\widehat{=}$ (restricted form of) resolution

* clause learning ??

▷ **Definition 15.3.1 (Unit Resolution).** Unit resolution (UR) is the test calculus consisting of the following inference rule:

$$\frac{C \vee P^{\alpha} \quad P^{\beta} \quad \alpha \neq \beta}{C}\,\mathsf{UR}$$

▷ Unit propagation $\widehat{=}$ resolution restricted to cases where one parent is unit clause.

▷ **Observation 15.3.2 (Soundness).** *UR is refutation sound.* *(since resolution is)*

▷ **Observation 15.3.3 (Completeness).** *UR is not refutation complete (alone).*

Chap 16

First Order Predicate Logic

predicate < T / F
function $x \xrightarrow{\text{maps}} y$

▷ **Definition 16.2.2.** A first-order signature consists of (all disjoint; $k \in \mathbb{N}$)

- ▷ connectives: $\Sigma^o = \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$ (functions on truth values)
- ▷ function constants: $\Sigma^f_k = \{f, g, h, \ldots\}$ (functions on individuals)
- ▷ predicate constants: $\Sigma^p_k = \{p, q, r, \ldots\}$ (relationships among individuals.)
- ▷ (Skolem constants: $\Sigma^{sk}_k = \{f^1_k, f^2_k, \ldots\}$) (witness constructors; countably ∞)

PL' syntax

▷ **Definition 16.2.4.** Terms: $\mathbf{A} \in wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ (denote individuals)

- ▷ $\mathcal{V}_\iota \subseteq wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$,
- ▷ if $f \in \Sigma^f_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$.

▷ **Definition 16.2.5.** if Propositions: $\mathbf{A} \in wff_o(\Sigma_\iota, \mathcal{V}_\iota)$: (denote truth values)

- ▷ if $p \in \Sigma^p_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_o(\Sigma_\iota, \mathcal{V}_\iota)$,
- ▷ if $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma_\iota, \mathcal{V}_\iota)$ and $X \in \mathcal{V}_\iota$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg\mathbf{A}, \forall X.\mathbf{A} \in wff_o(\Sigma_\iota, \mathcal{V}_\iota)$. $\forall$ is a binding operator called the universal quantifier.

▷ **Definition 16.2.6.** We define the connectives $F, \vee, \Rightarrow, \Leftrightarrow$ via the abbreviations $\mathbf{A} \vee \mathbf{B} := \neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, $\mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}$, $\mathbf{A} \Leftrightarrow \mathbf{B} := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$, and $F := \neg T$. We will use them like the primary connectives $\wedge$ and $\neg$

▷ **Definition 16.2.7.** We use $\exists X.\mathbf{A}$ as an abbreviation for $\neg(\forall X.\neg\mathbf{A})$. $\exists$ is a binding operator called the existential quantifier.

Bound & Free Var.

$\forall x\, P(x, y)$

- x bound
- y free

PL' Semantics:

- universe $\mathcal{D}_\iota \neq \emptyset$ of individuals
- interpretation $\mathcal{I}$: $\mathcal{I}: \Sigma^f_k \to \mathcal{D}_\iota^k \to \mathcal{D}_\iota$

$\mathcal{I}: \Sigma^p_k \to \mathcal{P}(\mathcal{D}_\iota^k)$

▷ **Definition 16.2.17.**

Given a model $\langle \mathcal{D}, \mathcal{I} \rangle$, the value function $\mathcal{I}_\varphi$ is recursively defined: terms & propositions)

- ▷ $\mathcal{I}_\varphi$: $wff_\iota(\Sigma_\iota, \mathcal{V}_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
  - ▷ $\mathcal{I}_\varphi(X) := \varphi(X)$ and
  - ▷ $\mathcal{I}_\varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_k))$
- ▷ $\mathcal{I}_\varphi$: $wff_o(\Sigma_\iota, \mathcal{V}_\iota) \to \mathcal{D}_o$ assigns values to formulae:
  - ▷ $\mathcal{I}_\varphi(T) = \mathcal{I}(T) = \mathsf{T}$,
  - ▷ $\mathcal{I}_\varphi(\neg \mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}_\varphi(\mathbf{A}))$
  - ▷ $\mathcal{I}_\varphi(\mathbf{A} \wedge \mathbf{B}) = \mathcal{I}(\wedge)(\mathcal{I}_\varphi(\mathbf{A}), \mathcal{I}_\varphi(\mathbf{B}))$
  - ▷ $\mathcal{I}_\varphi(p(\mathbf{A}_1, \ldots, \mathbf{A}_k)) := \mathsf{T}$, iff $\langle \mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_k) \rangle \in \mathcal{I}(p)$
  - ▷ $\mathcal{I}_\varphi(\forall X.\mathbf{A}) := \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$.

* Assignment Extension!
φ is var. assignment, then φ, [a/x] is called extension of φ with [a/x]

* First Order Substitutions:

Subs on terms $[z/x]$

Subs on proposition: var capture $[f(x)/Y](\forall x.P(x,Y)) \equiv \forall x.P(x, f(x))$ (x bound; Y free)

First Order ND':

▷ **Definition 16.3.1 (New Quantifier Rules).** The first-order natural deduction calculus $\mathcal{ND}^1$ extends $\mathcal{ND}_0$ by the following four rules:

$$\frac{\mathbf{A}}{\forall X.\mathbf{A}}\forall I^* \qquad \frac{\forall X.\mathbf{A}}{[\mathbf{B}/X](\mathbf{A})}\forall E$$

$$\frac{[\mathbf{B}/X](\mathbf{A})}{\exists X.\mathbf{A}}\exists I \qquad \frac{\exists X.\mathbf{A} \quad \begin{array}{c}[[c/X](\mathbf{A})]^1 \\ \vdots \\ \mathbf{C}\end{array} \quad c \in \Sigma^{sk}_0 \text{ new}}{\mathbf{C}}\exists E^1$$

First-order ND' in sequent form:

▷ **Definition 16.3.3 (New Quantifier Rules).** The inference rules of the first-order sequent calculus $\mathcal{ND}^1_\vdash$ consist of those from $\mathcal{ND}^0_\vdash$ plus the following quantifier rules:

$$\frac{\Gamma \vdash \mathbf{A} \quad X \notin \text{free}(\Gamma)}{\Gamma \vdash \forall X.\mathbf{A}}\forall I \qquad \frac{\Gamma \vdash \forall X.\mathbf{A}}{\Gamma \vdash [\mathbf{B}/X](\mathbf{A})}\forall E$$

$$\frac{\Gamma \vdash [\mathbf{B}/X](\mathbf{A})}{\Gamma \vdash \exists X.\mathbf{A}}\exists I \qquad \frac{\Gamma \vdash \exists X.\mathbf{A} \quad \Gamma, [c/X](\mathbf{A}) \vdash \mathbf{C} \quad c \in \Sigma^{sk}_0 \text{ new}}{\Gamma \vdash \mathbf{C}}\exists E$$

▷ **Definition 16.3.5.** For the calculus of natural deduction with equality ($\mathcal{ND}^1_=$) we add the following two rules to $\mathcal{ND}^1$ to deal with equality:

$$\frac{}{\mathbf{A} = \mathbf{A}}=I \qquad \frac{\mathbf{A} = \mathbf{B} \quad \mathbf{C}[\mathbf{A}]_p}{[\mathbf{B}/p]\mathbf{C}}=E$$

where $\mathbf{C}[\mathbf{A}]_p$ if the formula $\mathbf{C}$ has a subterm $\mathbf{A}$ at position p and $[\mathbf{B}/p]\mathbf{C}$ is the result of replacing that subterm with $\mathbf{B}$.

Chap 17

Automated Theorem Proving in FO Logic

FO Standard Tableaux ($\mathcal{T}_1$)

▷ **Definition 17.1.8.** The standard tableau calculus ($\mathcal{T}_1$) extends $\mathcal{T}_0$ (propositional tableau calculus) with the following quantifier rules:

$$\frac{(\forall X.\mathbf{A})^{\mathsf{T}} \quad \mathbf{C}\in cwff_\iota(\Sigma_\iota)}{([\mathbf{C}/X](\mathbf{A}))^{\mathsf{T}}}\mathcal{T}_1\forall \qquad \frac{(\forall X.\mathbf{A})^{\mathsf{F}} \quad c\in\Sigma_0^{sk} \text{ new}}{([c/X](\mathbf{A}))^{\mathsf{F}}}\mathcal{T}_1\exists$$

Free var. Tableaux (T_1^f)

▷ **Definition 17.1.9.** The free variable tableau calculus ($\mathcal{T}_1^f$) extends $\mathcal{T}_0$ (propositional tableau calculus) with the quantifier rules:

$$\frac{(\forall X.\mathbf{A})^{\mathsf{T}} \quad Y \text{ new}}{([Y/X](\mathbf{A}))^{\mathsf{T}}}\mathcal{T}_1^f\forall \qquad \frac{(\forall X.\mathbf{A})^{\mathsf{F}} \quad \mathsf{free}(\forall X.\mathbf{A})=\{X^1,\ldots,X^k\} \quad f\in\Sigma_k^{sk} \text{ new}}{([f(X^1,\ldots,X^k)/X](\mathbf{A}))^{\mathsf{F}}}\mathcal{T}_1^f\exists$$

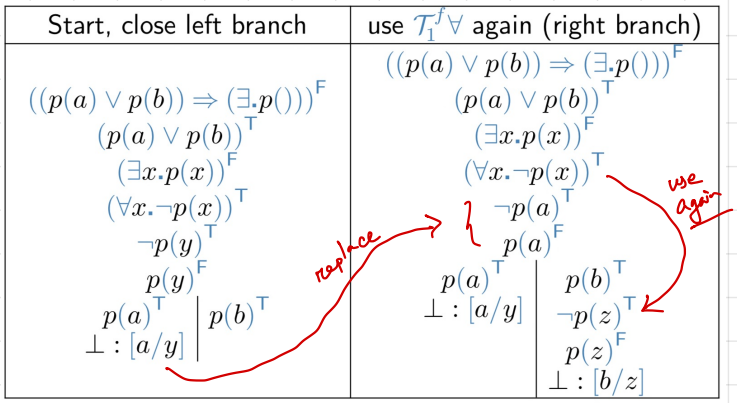
and generalizes its cut rule $\mathcal{T}_0\bot$ to:

$$\frac{\mathbf{A}^\alpha \quad \mathbf{B}^\beta \quad \alpha\neq\beta \quad \sigma(\mathbf{A})=\sigma(\mathbf{B})}{\bot:\sigma}\mathcal{T}_1^f\bot$$

$\mathcal{T}_1^f\bot$ instantiates the whole tableau by σ.

Example of T_1^f

Start, close left branch	use $\mathcal{T}_1^f\forall$ again (right branch)
$((p(a)\vee p(b))\Rightarrow(\exists.p()))^{\mathsf{F}}$ $(p(a)\vee p(b))^{\mathsf{T}}$ $(\exists x.p(x))^{\mathsf{F}}$ $(\forall x.\neg p(x))^{\mathsf{T}}$ $\neg p(y)^{\mathsf{T}}$ $p(y)^{\mathsf{F}}$ $p(a)^{\mathsf{T}} \mid p(b)^{\mathsf{T}}$ $\bot:[a/y]$	$((p(a)\vee p(b))\Rightarrow(\exists.p()))^{\mathsf{F}}$ $(p(a)\vee p(b))^{\mathsf{T}}$ $(\exists x.p(x))^{\mathsf{F}}$ $(\forall x.\neg p(x))^{\mathsf{T}}$ $\neg p(a)^{\mathsf{T}}$ $p(a)^{\mathsf{F}}$ $p(a)^{\mathsf{T}} \mid p(b)^{\mathsf{T}}$ $\bot:[a/y] \mid \neg p(z)^{\mathsf{T}}$ $p(z)^{\mathsf{F}}$ $\bot:[b/z]$



Derivable quantifier rules in T_1^f

$$\frac{(\exists X.\mathbf{A})^{\mathsf{T}} \quad \mathsf{free}(\forall X.\mathbf{A})=\{X^1,\ldots,X^k\} \quad f\in\Sigma_k^{sk} \text{ new}}{([f(X^1,\ldots,X^k)/X](\mathbf{A}))^{\mathsf{T}}}$$

$$\frac{(\exists X.\mathbf{A})^{\mathsf{F}} \quad Y \text{ new}}{([Y/X](\mathbf{A}))^{\mathsf{F}}}$$

First Order Unification:

▷ **Definition 17.1.12.** For given terms $\mathbf{A}$ and $\mathbf{B}$, unification is the problem of finding a substitution σ, such that $\sigma(\mathbf{A})=\sigma(\mathbf{B})$.

unification algo:

$$\frac{\mathcal{E}\wedge f(\mathbf{A}^1,\ldots,\mathbf{A}^n)=^?f(\mathbf{B}^1,\ldots,\mathbf{B}^n)}{\mathcal{E}\wedge\mathbf{A}^1=^?\mathbf{B}^1\wedge\ldots\wedge\mathbf{A}^n=^?\mathbf{B}^n}\mathcal{U}\text{dec} \qquad \frac{\mathcal{E}\wedge\mathbf{A}=^?\mathbf{A}}{\mathcal{E}}\mathcal{U}\text{triv}$$

$$\frac{\mathcal{E}\wedge X=^?\mathbf{A} \quad X\notin\mathsf{free}(\mathbf{A}) \quad X\in\mathsf{free}(\mathcal{E})}{[\mathbf{A}/X](\mathcal{E})\wedge X=^?\mathbf{A}}\mathcal{U}\text{elim}$$

unification e.g.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f(g(X,X),h(a))=^?f(g(a,Z),h(Z))}{g(X,X)=^?g(a,Z)\wedge h(a)=^?h(Z)}\mathcal{U}\text{dec}}{X=^?a\wedge X=^?Z\wedge h(a)=^?h(Z)}\mathcal{U}\text{dec}}{X=^?a\wedge X=^?Z\wedge a=^?Z}\mathcal{U}\text{elim}}{X=^?a\wedge a=^?Z\wedge a=^?Z}\mathcal{U}\text{elim}}{X=^?a\wedge Z=^?a\wedge a=^?a}\mathcal{U}\text{triv}}{X=^?a\wedge Z=^?a}$$

MGU: $[a/X],[a/Z]$

$$\dfrac{\dfrac{\dfrac{\dfrac{f(g(X,X),h(a))=^?f(g(b,Z),h(Z))}{g(X,X)=^?g(b,Z)\wedge h(a)=^?h(Z)}\mathcal{U}\text{dec}}{X=^?b\wedge X=^?Z\wedge h(a)=^?h(Z)}\mathcal{U}\text{dec}}{X=^?b\wedge X=^?Z\wedge a=^?Z}\mathcal{U}\text{elim}}{X=^?b\wedge b=^?Z\wedge a=^?Z}\mathcal{U}\text{elim}$$

$X=^?b\wedge Z=^?b\wedge a=^?b$

$a=^?b$ not unifiable

first order resolution (à CNF)

* Horn clause is a clause with at most one positive literal

▷ **Definition 17.2.1.** The first-order CNF calculus CNF_1 is given by the inference rules of CNF_0 extended by the following quantifier rules:

$$\frac{(\forall X.\mathbf{A})^{\mathsf{T}} \vee \mathbf{C} \quad Z \notin (\mathsf{free}(\mathbf{A}) \cup \mathsf{free}(\mathbf{C}))}{([Z/X](\mathbf{A}))^{\mathsf{T}} \vee \mathbf{C}}$$

$$\frac{(\forall X.\mathbf{A})^{\mathsf{F}} \vee \mathbf{C} \quad \{X_1, \ldots, X_k\} = \mathsf{free}(\forall X.\mathbf{A}) \quad f \in \Sigma_k^{sk} \text{ new}}{([f(X_1, \ldots, X_k)/X](\mathbf{A}))^{\mathsf{F}} \vee \mathbf{C}}$$

$CNF_1(\Phi)$ is the set of all clauses that can be derived from Φ.

▷ **Definition 17.2.2 (First-Order Resolution Calculus).** First-order resolution ($\mathcal{R}_1$) is a test calculus that manipulates formulae in conjunctive normal form. $\mathcal{R}_1$ has two inference rules:

$$\frac{\mathbf{A}^{\mathsf{T}} \vee \mathbf{C} \quad \mathbf{B}^{\mathsf{F}} \vee \mathbf{D} \quad \sigma = \mathbf{mgu}(\mathbf{A}, \mathbf{B})}{(\sigma(\mathbf{C})) \vee (\sigma(\mathbf{D}))} \qquad \frac{\mathbf{A}^{\alpha} \vee \mathbf{B}^{\alpha} \vee \mathbf{C} \quad \sigma = \mathbf{mgu}(\mathbf{A}, \mathbf{B})}{(\sigma(\mathbf{A})) \vee (\sigma(\mathbf{C}))}$$

Derived rules for FO CNF

$$\frac{(\exists X.\mathbf{A})^{\mathsf{T}} \vee \mathbf{C} \quad \{X_1, \ldots, X_k\} = \mathsf{free}(\forall X.\mathbf{A}) \quad f \in \Sigma_k^{sk} \text{ new}}{([f(X_1, \ldots, X_k)/X](\mathbf{A}))^{\mathsf{T}} \vee \mathbf{C}}$$

$$\frac{(\exists X.\mathbf{A})^{\mathsf{F}} \vee \mathbf{C} \quad Z \notin (\mathsf{free}(\mathbf{A}) \cup \mathsf{free}(\mathbf{C}))}{([Z/X](\mathbf{A}))^{\mathsf{F}} \vee \mathbf{C}}$$

three principle modes of inference

- **Definition 3.22.** Deduction $\widehat{=}$ knowledge extension
- **Example 3.23.** $\dfrac{rains \Rightarrow wet\_street \quad rains}{wet\_street} D$
- **Definition 3.24.** Abduction $\widehat{=}$ explanation
- **Example 3.25.** $\dfrac{rains \Rightarrow wet\_street \quad wet\_street}{rains} A$
- **Definition 3.26.** Induction $\widehat{=}$ learning general rules from examples
- **Example 3.27.** $\dfrac{wet\_street \quad rains}{rains \Rightarrow wet\_street} I$

chap 18

Knowledge Representation & the Semantic Web

Intro of KR: knowledge is the information necessary to support intelligent reasoning! knowledge appears in layers,

Character Set → Glyphs → Syntax → Data → Context → Information → Networking → Knowledge

* evaluation criteria for KR approaches:

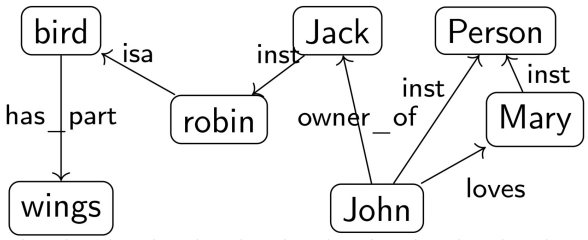
- Expressive adequacy
- Primitives
- Completeness
- Reasoning efficiency
- Meta representation

Semantic Networks: [example of KR]

semantic network is a directed graph for representing knowledge

- nodes: objects & concepts (classes of obj.)
- edges: relation between nodes

A semantic network for birds and persons:



links
- 'isa' an inclusion or isa link
- 'inst' instance or inst link

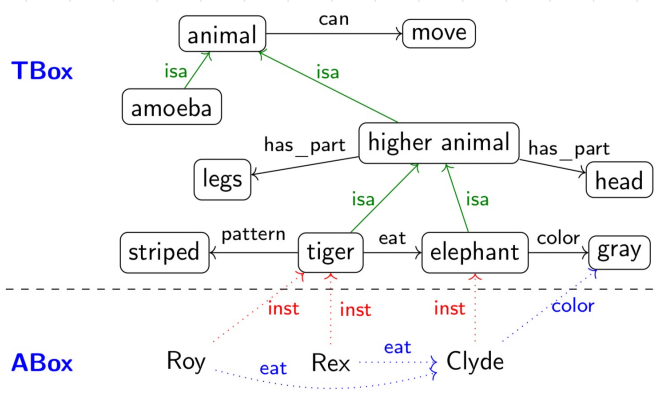
▷ **Definition 18.1.7 (Inference in Semantic Networks).** We call all link labels except "inst" and "isa" in a semantic network relations.

Let N be a semantic network and R a relation in N such that $A \xrightarrow{\text{isa}} B \xrightarrow{R} C$ or $A \xrightarrow{\text{inst}} B \xrightarrow{R} C$, then we can derive a relation $A \xrightarrow{R} C$ in N.

The process of deriving new concepts and relations from existing ones is called inference and concepts/relations that are only available via inference implicit (in a semantic network).

subgraph of semantic network N spanned by isa links & relations between concepts
→ terminology / TBox

subgraph spanned by inst link and relations between objects
— assertions / ABox



* Semantic Web

Propositional Logic as Set Description Language:

▷ **Definition 18.2.1.** Let $\mathsf{PL}^0_{\mathsf{DL}}$ be given by the following grammar for the $\mathsf{PL}^0_{\mathsf{DL}}$ concepts. (formulae)

$$\mathcal{L} ::= C \mid \top \mid \bot \mid \overline{\mathcal{L}} \mid \mathcal{L} \sqcap \mathcal{L} \mid \mathcal{L} \sqcup \mathcal{L} \mid \mathcal{L} \sqsubseteq \mathcal{L} \mid \mathcal{L} \equiv \mathcal{L}$$

i.e. $\mathsf{PL}^0_{\mathsf{DL}}$ formed from

- ▷ atomic ($\widehat{=}$ propositional variables)
- ▷ concept intersection ($\sqcap$) ($\widehat{=}$ conjunction $\wedge$)
- ▷ concept complement ($\bar{\cdot}$) ($\widehat{=}$ negation $\neg$)
- ▷ concept union ($\sqcup$), subsumption ($\sqsubseteq$), and equality ($\equiv$) defined from these. ($\widehat{=}$ $\vee$, $\Rightarrow$, and $\Leftrightarrow$)

Definition 2.2 (Formal Semantics).
Let $\mathcal{D}$ be a given set (called the domain) and $\varphi\colon \mathcal{V}_0 \to \mathcal{P}(\mathcal{D})$, then we define

- $[\![P]\!] := \varphi(P)$, (remember $\varphi(P) \subseteq \mathcal{D}$).
- $[\![\mathsf{A} \sqcap \mathsf{B}]\!] := [\![\mathsf{A}]\!] \cap [\![\mathsf{B}]\!]$ and $[\![\overline{\mathsf{A}}]\!] := \mathcal{D} \setminus [\![\mathsf{A}]\!]$ ...

$[\![\top]\!] = [\![p]\!] \cup [\![\bar{p}]\!] = [\![p]\!] \cup (\mathcal{D} \setminus [\![p]\!]) = \mathcal{D}$ Analogously: $[\![\bot]\!] = \emptyset$

Translation into PL': $-fo(x)$ the translation into FO logic w.r.t x

Definition	Comment
$\overline{p}^{fo(x)} := p(x)$	
$\overline{\overline{\mathsf{A}}}^{fo(x)} := \neg \overline{\mathsf{A}}^{fo(x)}$	
$\overline{\mathsf{A} \sqcap \mathsf{B}}^{fo(x)} := \overline{\mathsf{A}}^{fo(x)} \wedge \overline{\mathsf{B}}^{fo(x)}$	$\wedge$ vs. $\sqcap$
$\overline{\mathsf{A} \sqcup \mathsf{B}}^{fo(x)} := \overline{\mathsf{A}}^{fo(x)} \vee \overline{\mathsf{B}}^{fo(x)}$	$\vee$ vs. $\sqcup$
$\overline{\mathsf{A} \sqsubseteq \mathsf{B}}^{fo(x)} := \overline{\mathsf{A}}^{fo(x)} \Rightarrow \overline{\mathsf{B}}^{fo(x)}$	$\Rightarrow$ vs. $\sqsubseteq$
$\overline{\mathsf{A} = \mathsf{B}}^{fo(x)} := \overline{\mathsf{A}}^{fo(x)} \Leftrightarrow \overline{\mathsf{B}}^{fo(x)}$	$\Leftrightarrow$ vs. $=$
$\overline{\mathsf{A}}^{fo} := (\forall x. \overline{\mathsf{A}}^{fo(x)})$	for formulae

Ontologies & Description Logics:

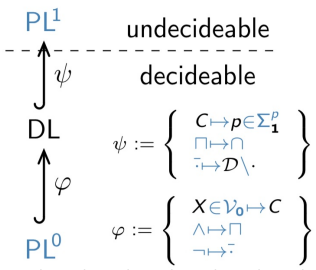
↳ world description

Semantic Networks

$\mathsf{PL}^0_{\mathsf{DL}}$, PL'

⟹ ontology format example

- **Definition 2.10.** An ontology consists of a logical system $\langle \mathcal{L}, \mathcal{K}, \models \rangle$ and concept axioms (expressed in $\mathcal{L}$) about
  - individuals: concrete instances of object in the domain,
  - concepts: classes of individuals that share properties and aspects, and
  - relations: ways in which concept and individuals can be related to one another.
- **Definition 2.14.** A description logic is a formal system for talking about collections of objects and their relations that is at least as expressive as PL^0 with set-theoretic semantics and offers individuals and relations.
  A description logic has the following four components:
  - a formal language $\mathcal{L}$ with logical constants $\sqcap$, $\bar{\cdot}$, $\sqcup$, $\sqsubseteq$, and $\equiv$,
  - a set-theoretic semantics $\langle \mathcal{D}, [\![\cdot]\!] \rangle$,
  - a translation into first-order logic that is compatible with $\langle \mathcal{D}, [\![\cdot]\!] \rangle$, and
  - a calculus for $\mathcal{L}$ that induces a decision procedure for $\mathcal{L}$-satisfiability.



TBox & ABox

- **Definition 2.15.** Given a description logic $\mathcal{D}$, a $\mathcal{D}$ ontology consists of
  - a terminology (or TBox): concepts and roles and a set of concept axioms that describe them, and
  - sassertion (or ABox): a set of individuals and statements about concept membership and role relationships for them.

* TBox is a finite set of concept def^n and concept axioms. It is called acyclic, iff it doesnot contain recursive def^n.

▷ **Definition 18.2.20.** A formula $\mathbf{A}$ is called normalized wrt. an TBox $\mathcal{T}$, iff it does not contain concepts defined in $\mathcal{T}$. (convenient)

▷ **Definition 18.2.21 (Algorithm).** (for arbitrary DLs)
Input: A formula $\mathbf{A}$ and a TBox $\mathcal{T}$.

▷ While [$\mathbf{A}$ contains concept c and $\mathcal{T}$ a concept definition $c{=}\mathbf{C}$]

▷ substitute c by $\mathbf{C}$ in $\mathbf{A}$.

Description Logic & Inference:

▷ **Definition 18.2.23.** Ontology systems employ three main reasoning services:

▷ Consistency test: is a concept definition satisfiable?

▷ Subsumption test: does a concept subsume another?

▷ Instance test: is an individual an example of a concept?

Simple description logic : ALC

Syntax of ALC :

— Concepts : person, woman, man — etc

special concepts ← top concept $\top$ (for "true" or "all")
bottom concept $\bot$ (for "false" or "none")

— Roles : binary relations | has_child, loves, — etc.

▷ **Definition 18.3.7 (Grammar).** $F_{\mathcal{ALC}} ::= C \mid \top \mid \bot \mid \overline{F_{\mathcal{ALC}}} \mid F_{\mathcal{ALC}} \sqcap F_{\mathcal{ALC}} \mid F_{\mathcal{ALC}} \sqcup F_{\mathcal{ALC}} \mid \exists \mathrm{R}.F_{\mathcal{ALC}} \mid \forall \mathrm{R}.F_{\mathcal{ALC}}$

* all/only ⇒ use $\forall$
* at least ⇒ use $\exists$

class inclusion : Prof. $\sqsubseteq$ FacultyMember
Class equivalence : Prof. $\equiv$ FacultyMember

e.g. man = person $\sqcap$ $\exists$ has_chrom. Y_chrom
(man: definiendum; person: primitive concept)

TBox Normalization in ALC : → for only non-acyclic

- **Definition 3.15.** We call an $\mathcal{ALC}$ formula φ normalized wrt. a set of concept definitions, iff all concept names occurring in φ are primitive.
- **Definition 3.16.** Given a set $\mathcal{D}$ of concept definitions, normalization is the process of replacing in an $\mathcal{ALC}$ formula φ all occurrences of definienda in $\mathcal{D}$ with their definientia.

Semantics of ALC :

▷ **Definition 18.3.21.** A model for $\mathcal{ALC}$ is a pair $\langle \mathcal{D}, [[\cdot]] \rangle$, where $\mathcal{D}$ is a non-empty set called the domain and $[[\cdot]]$ a mapping called the interpretation, such that

Op.	formula semantics
	$[\![c]\!] \subseteq \mathcal{D} = [\![\top]\!] \quad [\![\bot]\!] = \emptyset \quad [\![r]\!] \subseteq \mathcal{D} \times \mathcal{D}$
$\overline{\cdot}$	$[\![\overline{\varphi}]\!] = \overline{[\![\varphi]\!]} = \mathcal{D} \setminus [\![\varphi]\!]$
$\sqcap$	$[\![\varphi \sqcap \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$
$\sqcup$	$[\![\varphi \sqcup \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$
$\exists \mathrm{R}.$	$[\![\exists \mathrm{R}.\varphi]\!] = \{x \in \mathcal{D} \mid \exists y. \langle x, y \rangle \in [\![\mathrm{R}]\!] \text{ and } y \in [\![\varphi]\!]\}$
$\forall \mathrm{R}.$	$[\![\forall \mathrm{R}.\varphi]\!] = \{x \in \mathcal{D} \mid \forall y. \text{if } \langle x, y \rangle \in [\![\mathrm{R}]\!] \text{ then } y \in [\![\varphi]\!]\}$

* ALC into PL¹:

▷ **Definition 18.3.22.** The translation of $\mathcal{ALC}$ into PL^1 extends the one from ?? by the following quantifier rules:

$$\overline{\forall \mathsf{R}.\varphi}^{fo(x)} := (\forall y.\mathsf{R}(x,y) \Rightarrow \overline{\varphi}^{fo(y)}) \quad \overline{\exists \mathsf{R}.\varphi}^{fo(x)} := (\exists y.\mathsf{R}(x,y) \wedge \overline{\varphi}^{fo(y)})$$

* ALC Identities:

▷

1	$\overline{\exists \mathsf{R}.\varphi} = \forall \mathsf{R}.\overline{\varphi}$	3	$\overline{\forall \mathsf{R}.\varphi} = \exists \mathsf{R}.\overline{\varphi}$
2	$\forall \mathsf{R}.(\varphi \sqcap \psi) = \forall \mathsf{R}.\varphi \sqcap \forall \mathsf{R}.\psi$	4	$\exists \mathsf{R}.(\varphi \sqcup \psi) = \exists \mathsf{R}.\varphi \sqcup \exists \mathsf{R}.\psi$

▷ Proof of 1

$$\begin{aligned}
[\![\overline{\exists \mathsf{R}.\varphi}]\!] = \mathcal{D} \setminus [\![\exists \mathsf{R}.\varphi]\!] &= \mathcal{D} \setminus \{x \in \mathcal{D} \mid \exists y.(\langle x,y\rangle \in [\![\mathsf{R}]\!]) \text{ and } (y \in [\![\varphi]\!])\} \\
&= \{x \in \mathcal{D} \mid \text{not } \exists y.(\langle x,y\rangle \in [\![\mathsf{R}]\!]) \text{ and } (y \in [\![\varphi]\!])\} \\
&= \{x \in \mathcal{D} \mid \forall y.\text{if } (\langle x,y\rangle \in [\![\mathsf{R}]\!]) \text{ then } (y \notin [\![\varphi]\!])\} \\
&= \{x \in \mathcal{D} \mid \forall y.\text{if } (\langle x,y\rangle \in [\![\mathsf{R}]\!]) \text{ then } (y \in (\mathcal{D} \setminus [\![\varphi]\!]))\} \\
&= \{x \in \mathcal{D} \mid \forall y.\text{if } (\langle x,y\rangle \in [\![\mathsf{R}]\!]) \text{ then } (y \in [\![\overline{\varphi}]\!])\} \\
&= [\![\forall \mathsf{R}.\overline{\varphi}]\!]
\end{aligned}$$

Negation Normal Form (NNF)

ALC formula is in NNF iff $\neg$ is only applied to concept names.

example	by rule
$\overline{\exists \mathsf{R}.(\forall \mathsf{S}.e \sqcap \overline{\forall \mathsf{S}.d})}$	
$\mapsto \forall \mathsf{R}.\overline{\forall \mathsf{S}.e \sqcap \overline{\forall \mathsf{S}.d}}$	$\overline{\exists \mathsf{R}.\varphi} \mapsto \forall \mathsf{R}.\overline{\varphi}$
$\mapsto \forall \mathsf{R}.(\overline{\forall \mathsf{S}.e} \sqcup \overline{\overline{\forall \mathsf{S}.d}})$	$\overline{\varphi \sqcap \psi} \mapsto \overline{\varphi} \sqcup \overline{\psi}$
$\mapsto \forall \mathsf{R}.(\exists \mathsf{S}.\overline{e} \sqcup \overline{\overline{\forall \mathsf{S}.d}})$	$\overline{\forall \mathsf{R}.\varphi} \mapsto \forall \mathsf{R}.\overline{\varphi}$
$\mapsto \forall \mathsf{R}.(\exists \mathsf{S}.\overline{e} \sqcup \forall \mathsf{S}.d)$	$\overline{\overline{\varphi}} \mapsto \varphi$

* ALC with Assertions about individuals (ABox)

individuals $a : \varphi$ → concept

Definition 3.25. We define the assertions for $\mathcal{ALC}$

- $a{:}\varphi$ (a is a φ)
- $a\ \mathsf{R}\ b$ (a stands in relation R to b)

assertions make up the ABox in $\mathcal{ALC}$.

Definition 3.26. Let $\langle \mathcal{D}, [\![\cdot]\!] \rangle$ be a model for $\mathcal{ALC}$, then we define

- $[\![a{:}\varphi]\!] = \mathsf{T}$, iff $[\![a]\!] \in [\![\varphi]\!]$, and
- $[\![a\ \mathsf{R}\ b]\!] = \mathsf{T}$, iff $([\![a]\!], [\![b]\!]) \in [\![\mathsf{R}]\!]$.

Definition 3.27. We extend the PL^1 translation of $\mathcal{ALC}$ to $\mathcal{ALC}$ assertions:

- $\overline{a{:}\varphi}^{fo} := \overline{\varphi}^{fo(a)}$, and
- $\overline{a\ \mathsf{R}\ b}^{fo} := \mathsf{R}(a,b)$.

Inference for ALC:

Definition 3.28. The $\mathcal{ALC}$ tableau calculus $\mathcal{T}_{\mathcal{ALC}}$ acts on assertions

- $x{:}\varphi$ (x inhabits concept φ)
- $x\ \mathsf{R}\ y$ (x and y are in relation R)

where φ is a normalized $\mathcal{ALC}$ concept in negation normal form with the following rules:

$$\frac{x{:}c \quad x{:}\overline{c}}{\bot}\,\mathcal{T}_\bot \qquad \frac{x{:}\varphi \sqcap \psi}{\begin{array}{l}x{:}\varphi \\ x{:}\psi\end{array}}\,\mathcal{T}_\sqcap \qquad \frac{x{:}\varphi \sqcup \psi}{x{:}\varphi \mid x{:}\psi}\,\mathcal{T}_\sqcup \qquad \frac{x{:}\forall \mathsf{R}.\varphi \quad x\ \mathsf{R}\ y}{y{:}\varphi}\,\mathcal{T}_\forall \qquad \frac{x{:}\exists \mathsf{R}.\varphi}{\begin{array}{l}x\ \mathsf{R}\ y \\ y{:}\varphi\end{array}}\,\mathcal{T}_\exists$$

We study the following properties of a tableau calculus $\mathcal{C}$:

- **Termination**: there are no infinite sequences of rule applications.
- **Correctness**: If φ is satisfiable, then $\mathcal{C}$ terminates with an open branch.
- **Completeness**: If φ is in unsatisfiable, then $\mathcal{C}$ terminates and all branches are closed.
- **Complexity** of the algorithm (time and space complexity).

Connectives

Lemma 3.35. *If φ satisfiable, then $\mathcal{T}_{\mathcal{ALC}}$ terminates on $x{:}\varphi$ with open branch.*

Proof: Let $\mathcal{M}{:=}\langle\mathcal{D}, [\![\cdot]\!]\rangle$ be a model for φ and $w{\in}[\![\varphi]\!]$.

1. We define $[\![x]\!]{:=}w$ and
$$\begin{array}{lll} \mathcal{I}{\models}(x{:}\psi) & \text{iff} & [\![x]\!]{\in}[\![\psi]\!] \\ \mathcal{I}{\models}x\ \mathsf{R}\ y & \text{iff} & \langle x, y\rangle{\in}[\![\mathsf{R}]\!] \\ \mathcal{I}{\models}S & \text{iff} & \mathcal{I}{\models}c \text{ for all } c{\in}S \end{array}$$
2. This gives us $\mathcal{M}{\models}(x{:}\varphi)$ (base case)
3. If the branch is satisfiable, then either
   - no rule applicable to leaf, (open branch)
   - or rule applicable and one new branch satisfiable. (inductive case)
4. There must be an open branch. (by termination)

Completeness

Lemma 3.36. *Open saturated tableau branches for φ induce models for φ.*

Proof: construct a model for the branch and verify for φ

1. Let $\mathcal{B}$ be an open saturated branch
   - we define
$$\begin{array}{rcl} \mathcal{D} & := & \{x | x{:}\psi{\in}\mathcal{B} \text{ or } z\ \mathsf{R}\ x{\in}\mathcal{B}\} \\ [\![c]\!] & := & \{x | x{:}c{\in}\mathcal{B}\} \\ [\![\mathsf{R}]\!] & := & \{\langle x, y\rangle | x\ \mathsf{R}\ y{\in}\mathcal{B}\} \end{array}$$
   - well-defined since never $x{:}c, x{:}\overline{c}{\in}\mathcal{B}$ (otherwise $\mathcal{T}_\perp$ applies)
   - $\mathcal{M}$ satisfies all constraints $x{:}c$, $x{:}\overline{c}$ and $x\ \mathsf{R}\ y$, (by construction)
2. $\mathcal{M}{\models}(y{:}\psi)$, for all $y{:}\psi{\in}\mathcal{B}$ (on $k = \mathit{size}(\psi)$ next slide)
3. $\mathcal{M}{\models}(x{:}\varphi)$.

ABoxes, Instance testing:

Definition 3.44. An instance test computes whether given an $\mathcal{ALC}$ ontology an individual is a member of a given class.

Example 3.45 (An Ontology).

TBox (terminological Box)			ABox (assertional Box, data base)	
woman	=	person $\sqcap$ $\overline{\text{has\_Y}}$	tony:person	Tony is a person
man	=	person $\sqcap$ has_Y	tony:has_Y	Tony has a y-chrom

This entails: tony:man (Tony is a man).

- **Definition 3.46.** Realization is the computation of all instance relations between ABox objects and TBox concepts.

chap 19

Planning I : Framework

Planning Language is a logical language for the components of a search problem.

- possible states (vs. blackbox : data structure)
- initial state I (vs. data structure)
- goal test G (vs. goal test func^n)
- set A of actions ⇒ preconditions & effects
  (vs. func^n returning applicable action & successor states)

⇒ logical description of all of these is called a planning task

⇒ process of finding a plan given a planning task is called planning

* soln (plan) ≙ seq. of actions from A, transforming I into state that satisfies G.

* Satisficing planning : a plan for π, or "unsolvable" if no plan for π exists
  optimal planning : an optimal plan for π, or "unsolvable" if no plan for π exists

* Prog. solving these problems are called (optimal) planners, planning sys. / tool.

STRIPS Planning [Stanford Research Institute Problem Solver]

↳ Logic based planning language

STRIPS has only propositional var. as atomic formulae

whether an action applies
- preconditions, goals : conjunction of atoms → (positive atoms)
- Effects : Conjunction of literals → (pos or negated atoms)

STRIPS Syntax :

▷ **Definition 19.4.2.** A STRIPS task is a quadruple $\langle P, A, I, G\rangle$ where:

- ▷ P is a finite set of facts (aka proposition).
- ▷ A is a finite set of actions; each $a \in A$ is a triple $a = \langle \text{pre}_a, \text{add}_a, \text{del}_a\rangle$ of subsets of P referred to as the action's precondition, add list, and delete list respectively; we require that $\text{add}_a \cap \text{del}_a = \emptyset$.
- ▷ $I \subseteq P$ is the initial state.
- ▷ $G \subseteq P$ is the goal.

We will often give each action $a \in A$ a name (a string), and identify a with that name.

* assume every action has cost 1

π = <P, A, I, G>

pre_a → pre condition for applying an action

add_a / del_a → add/del set of facts to current world.

STRIPS Planning Semantics:

▷ **Definition 19.4.5.** Let $\Pi := \langle P, A, I, G\rangle$ be a STRIPS task. The search problem induced by Π is $\Theta_\Pi = \langle S_P, A, T_A, I, S_G\rangle$ where:

- ▷ The states (also world state) $S_P := \mathcal{P}(P)$ are the subsets of P.
- ▷ A is just Π's action set. (so we can define plans easily)
- ▷ The transition model T_A is $\{s \xrightarrow{a} \text{apply}(s,a) | \text{pre}_a \subseteq s\}$.
  If $\text{pre}_a \subseteq s$, then $a \in A$ is applicable in s and $\text{apply}(s,a) := s \cup \text{add}_a \setminus \text{del}_a$. If $\text{pre}_a \not\subseteq s$, then $\text{apply}(s,a)$ is undefined.
- ▷ I is Π's initial state.
- ▷ The goal states $S_G = \{s \in S_G | G \subseteq s\}$ are those that satisfy Π's goal.

An (optimal) plan for Π is an (optimal) solution Θ_Π, i.e., a path from s to some $s' \in S_G$. A solution for I is called a plan for Π. Π is solvable if a plan for Π exists.

For $a = \langle a_1, \ldots, a_n\rangle$, $\text{apply}(s,a) := \text{apply}(s, \text{apply}(s, a_2 \ldots \text{apply}(s, a_n)))$ if each a_i is applicable in the respective state; else, $\text{apply}(s,a)$ is undefined.

STRIPS

pre conditions

Action

Effects / post condition

Partial Order Planning:

Definition 5.7. Let $\Pi := \langle P, A, I, G\rangle$ be a STRIPS task, then a partially ordered plan $\mathcal{P} = \langle V, E\rangle$ is a labeled DAG, where the nodes in V (called steps) are labeled with actions from A, or are a

- start step, which has label effect I, or a
- finish step, which has label precondition G.

Every edge $(S,T) \in E$ is either labeled by:

- A non-empty set $p \subseteq P$ of facts that are effects of the action of S and the preconditions of that of T. We call such a labeled edge a causal link and write it $S \xrightarrow{p} T$.
- $\prec$, then call it a temporal constraint and write it as $S \prec T$.

An open condition is a precondition of a step not yet causally linked.

- **Definition 5.8.** Let be a partially ordered plan Π, then we call a step U possibly intervening in a causal link $S \xrightarrow{p} T$, iff $\Pi \cup \{S \prec U, U \prec T\}$ is acyclic.
- **Definition 5.9.** A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it.
- **Definition 5.10.** A partially ordered plan Π is called complete iff every precondition is achieved.
- **Definition 5.11.** Partial order planning is the process of computing complete and acyclic partially ordered plans for a given planning task.

* Planning Process:

Definition 5.13. Partial order planning is search in the space of partial plans via the following operations:

- add link from an existing action to an open precondition,
- add step (an action with links to other steps) to fulfil an open condition,
- order one step wrt. another to remove possible conflicts.

* Clobbering and Promotion/Demotion:

Definition 5.17. In a partially ordered plan, a step C clobbers a causal link $L := S \xrightarrow{p} T$, iff it destroys the condition p achieved by L.

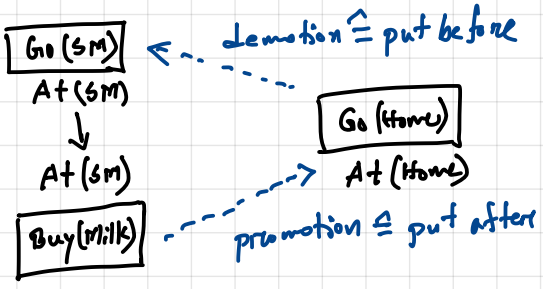
Definition 5.18. If C clobbers $S \xrightarrow{p} T$ in a partially ordered plan Π, then we can solve the induced conflict by

- demotion: add a temporal constraint $C \prec S$ to Π, or
- promotion: add $T \prec C$ to Π.

?!

* PoP is sound, complete, systematic - i.e. no repetition

e.g. Go(Home) clobbers At(Supermarket)



PDDL: Planning Domain Description Language

Definition 6.1. The Planning Domain Description Language (PDDL) is a standardized representation language for planning benchmarks in various extensions of the STRIPS formalism.

Definition 6.2. PDDL is not a propositional language

- Representation is lifted, using object variables to be instantiated from a finite set of objects. (Similar to predicate logic)
- Action schemas parameterized by objects.
- Predicates to be instantiated with objects.

Definition 6.3. A PDDL planning task comes in two pieces

- The problem file gives the objects, the initial state, and the goal state.
- The domain file gives the predicates and the actions.

chap 10

Planning II : Algorithms

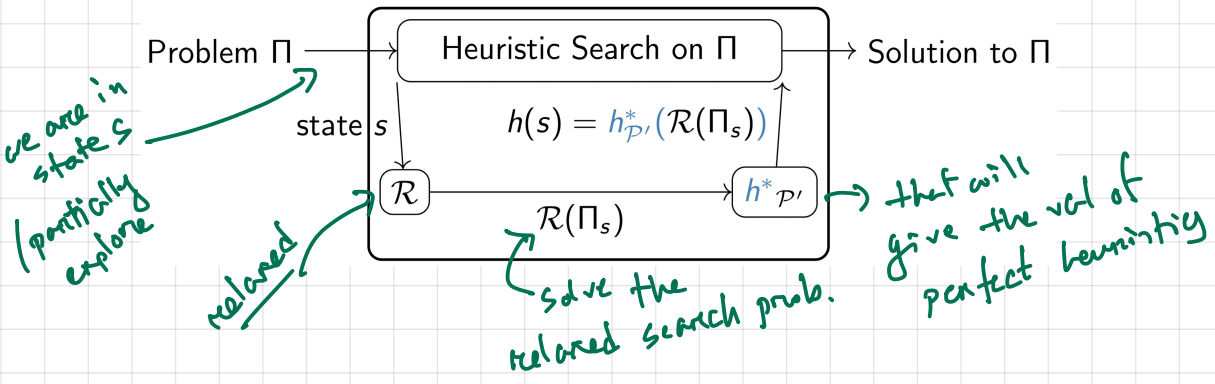
Heuristic Functions:

- **Definition 1.1.** Let Π be a STRIPS task with states S. A heuristic function, short heuristic, for Π is a function $h: S \to \mathbb{N} \cup \{\infty\}$ so that $h(s) = 0$ whenever s is a goal state.
- Exactly like our definition from 8 (Problem Solving and Search) in the AI lecture notes. Except, because we assume unit costs here, we use $\mathbb{N}$ instead of $\mathbb{R}^+$.

Perfect Heuristic :

- **Definition 1.2.** Let Π be a STRIPS task with states S. The perfect heuristic h^* assigns every $s \in S$ the length of a shortest path from s to a goal state, or ∞ if no such path exists. A heuristic function h for Π is admissible if, for all $s \in S$, we have $h(s) \leq h^*(s)$.
- Exactly like our definition from 8 (Problem Solving and Search) in the AI lecture notes, except for path *length* instead of path *cost* (cf. above).
- In all cases, we attempt to approximate $h^*(s)$, the length of an optimal plan for s. Some algorithms guarantee to lower bound $h^*(s)$.

* Relax in Planning:



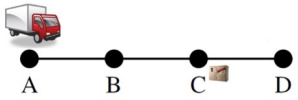
* Delete Relaxation:

Definition 3.1 (Delete Relaxation). Let $\Pi := \langle P, A, I, G \rangle$ be a STRIPS task. The delete relaxation of Π is the task $\Pi^+ = \langle P, A^+, I, G \rangle$ where $A^+ := \{a^+ | a \in A\}$ with $\text{pre}_{a^+} := \text{pre}_a$, $\text{add}_{a^+} := \text{add}_a$, and $\text{del}_{a^+} := \emptyset$.

In other words, the class of simpler problems $\mathcal{P}'$ is the set of all STRIPS tasks with empty delete lists, and the relaxation mapping $\mathcal{R}$ drops the delete lists.

- **Definition 3.2 (Relaxed Plan).** Let $\Pi := \langle P, A, I, G \rangle$ be a STRIPS task, and let s be a state. A relaxed plan for s is a plan for $\langle P, A, s, G \rangle^+$. A relaxed plan for I is called a relaxed plan for Π.
- A relaxed plan for s is an action sequence that solves s when pretending that all delete lists are empty.
- Also called "delete-relaxed plan": "relaxation" is often used to mean "delete-relaxation" by default.

A Relaxed Plan for "Logistics"



- Facts P: $\{\text{truck}(x) | x \in \{A, B, C, D\}\} \cup \{\text{pack}(x) | x \in \{A, B, C, D, T\}\}$.
- Initial state I: $\{\text{truck}(A), \text{pack}(C)\}$.
- Goal G: $\{\text{truck}(A), \text{pack}(D)\}$.
- Relaxed actions A^+: (Notated as "precondition ⇒ adds")
  - $\text{drive}(x, y)^+$: "$\text{truck}(x) \Rightarrow \text{truck}(y)$".
  - $\text{load}(x)^+$: "$\text{truck}(x), \text{pack}(x) \Rightarrow \text{pack}(T)$".
  - $\text{unload}(x)^+$: "$\text{truck}(x), \text{pack}(T) \Rightarrow \text{pack}(x)$".

  Relaxed plan:

  $$\langle \text{drive}(A, B)^+, \text{drive}(B, C)^+, \text{load}(C)^+, \text{drive}(C, D)^+, \text{unload}(D)^+ \rangle$$
- We don't need to drive the truck back, because "it is still at A".

* PlanEx$^+$

Definition 3.3 (Relaxed Plan Existence Problem). By PlanEx$^+$, we denote the problem of deciding, given a STRIPS task $\Pi := \langle P, A, I, G \rangle$, whether or not there exists a relaxed plan for Π.

* h^+ Heuristic:

Definition 4.1 (Optimal Relaxed Plan). Let $\langle P, A, I, G \rangle := \langle P, A, I, G \rangle$ be a STRIPS task, and let s be a state. A optimal relaxed plan for s is an optimal plan for $\langle P, A, s, G \rangle^+$.

Definition 4.2. Let $\langle P, A, I, G \rangle := \langle P, A, I, G \rangle$ be a STRIPS task with states S. The ideal delete relaxation heuristic h^+ for $\langle P, A, I, G \rangle$ is the function $h^+ : S \to \mathbb{N} \cup \{\infty\}$ where $h^+(s)$ is the length of an optimal relaxed plan for s if a relaxed plan for s exists, and $h^+(s) = \infty$ otherwise.

In other words, $h^+ = h^* \circ \mathcal{R}$, cf. previous slide.

h^+ admissible:

let $\Pi = \langle P, A, I, G \rangle$ planning task, s state,

if $\langle a_1, a_2, \ldots a_n \rangle$ is a plan for $\Pi_s := \langle P, A, \{s\}, G \rangle$,

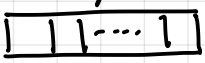
then $\langle a_1^+, a_2^+, \ldots, a_n^+ \rangle$ is plan for $\Pi^+ := \langle P, A, \{s\}, G \rangle^+$

$\text{apply}(s, \langle a_1, \ldots a_n \rangle) \subseteq \text{apply}(s, \langle a_1^+ \ldots, a_n^+ \rangle)$; $[0 \le i \le n]$

Search

Optimal Sound Complete

BFS : use queue for visiting node.



DFS : use stack for visiting node 

Iterative Deepening Search : (level wise DFS)
it will visit node level by level and
for each level it will use DFS.

Uniform - Cost Search :
fringe ⇒ list of nodes to be visited
cost ⇒ calculate cost from root node of nodes those are at fringe
visited ⇒ visit node with least cost nodes at fringe

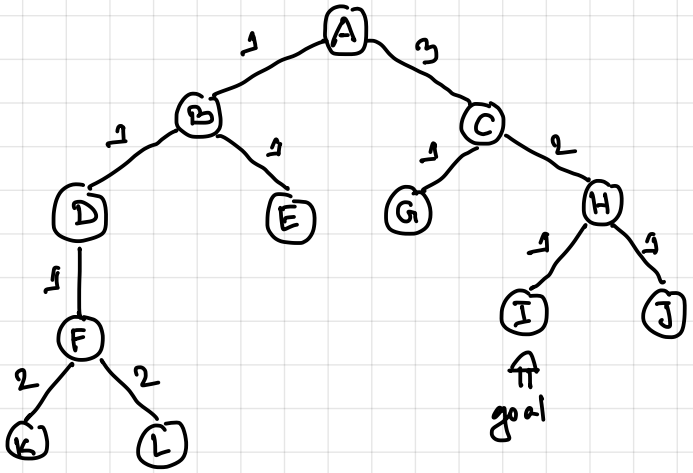
Greedy Search : $h(n)$
expand the node n that minimize h(n) function

A* Search : $g(n) + h(n)$
expand the node n that minimize $g(n)+h(n)$ function
$g(n)$: sum of cost from state I
$h(n)$: heuristic function

- $h^*(n)$: cost of cheapest path from n to goal state or ∞ : if no path exists
- admissible : $h(n) \leq h^*(n)$

Property	BFS	DFS	Uniform Cost	Iterative Deepening
Optimal	N	N	Y	N
Complete	Y	N	Y	Y

greedy	A*
N	Y
N	Y



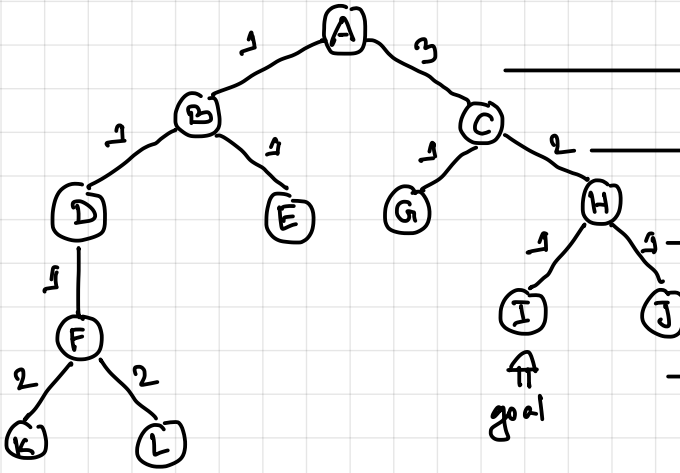
BFS:

visited node: A B C D E G H F (I)

DFS:

visited node: A B D F K L E C G H (I)

Iterative Deepening Search:



visited node

I_1 (d=0): DFS = A

I_2 (d=1): DFS = A B C

I_3 (d=2): DFS = A B D E C G H

I_4 (d=3): DFS = A B D F E C G H (I)

Uniform Cost Search:

fringe:	B	C	D	E	F	G	H	K	L	I
from root → cost:	1	3	2	2	3	4	5	5	5	6

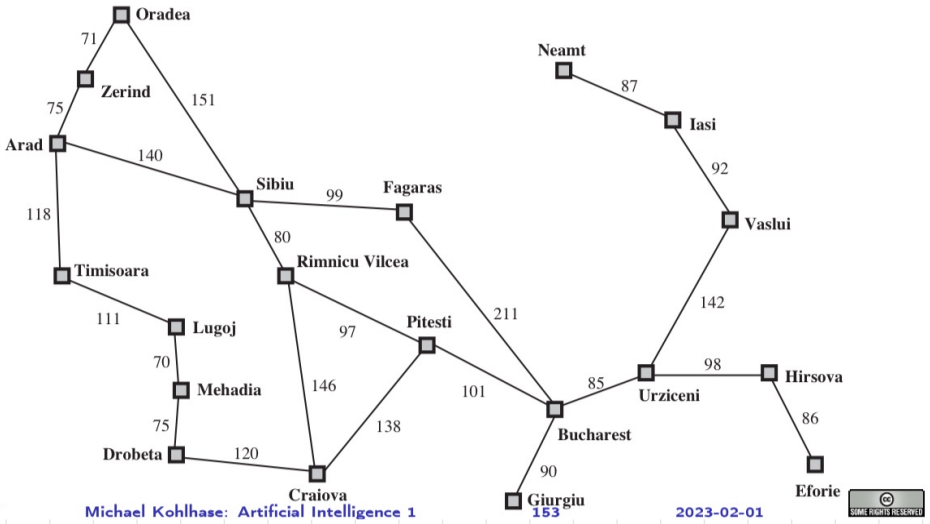
visited node: A B D E C F G H K L (I)

cost of root 0

Romania with Straight-Line Distances

- **Example 5.6 (Informed Travel).** $h_{SLD}(n)$ = straight-line distance to Bucharest

Arad	366	Mehadia	241	Bucharest	0	Neamt	234
Craiova	160	Oradea	380	Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193	Fragaras	176	Sibiu	253
Giurgiu	77	Timisoara	329	Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199	Lugoj	244	Zerind	274



SOME RIGHTS RESERVED

Lugoj to Bucharest

Greedy Search

h(n)

expanded nodes

Lugoj (244)
↓
Mehadia (241)
↓
Drobeta (242)
↓
Craiova (160)
↓
Pitesti (100)
↓
Bucharest (0)

fringe	h(n)	
Timisoara	329	
Mehadia	241	X
Drobeta	242	X
Craiova	160	X
Rimnicu vilcea	193	
Pitesti	100	X
Bucharest	0	X

A* Search

g(n)+h(n)

expanded nodes

Lugoj (244)
↓
Mehadia (311)
↓
Drobeta (387)
↓
Craiova (425)
↓
Timisoara (440)
↓
Pitesti (503)
↓
Bucharest (504)

fringe	g(n)+h(n)	total	
Lugoj	0+244	= 244	X
Timisoara	111+329	= 440	X
Mehadia	70+241	= 311	X
Drobeta	(70+75) +242	= 387	X
Craiova	(70+75+120) + 160	= 425	X
Rimnicu vilcea	(70+75+120 +146) + 193	= 604	
Pitesti	(70+75+120+ 138) + 100	= 503	X
Arad	(111+118) + 366	= 595	
Bucharest	(70+75+120 +138+100)+0	= 504	X