Logic   LVA 703600 VU3

http://cl-informatik.uibk.ac.at/teaching/ws05/logic/

Georg Moser (VU)[1]   Christian Vogt (VU)[2]

[1]georg.moser@uibk.ac.at
office hours: Thursday 1pm–3pm

[2]christian.vogt@uibk.ac.at
office hours: Tuesday 9am–11am

Autumn 2005

The Replacement Theorem

Definition let $F(A_1, \ldots, A_n)$ be a formula; propositional letters are among $A_1, \ldots, A_n$

$$F(X_1, \ldots, X_n) :\Leftrightarrow F(A_1, \ldots, A_n)\frac{X_1,\ldots,X_n}{A_1,\ldots,A_n}$$

Example suppose $F(P)$ is $(P \to Q) \vee \neg P$; X is $(P \wedge R)$:

$$F(X) = ((P \wedge R) \to Q) \vee \neg(P \wedge R)$$

Theorem If $X \equiv Y$ is a tautology, so is $F(X) \equiv F(Y)$.

Theorem Let $F(P)$, X, Y be formulas, v a valuation. If $v(X) = v(Y)$ then $v(F(X)) = v(F(Y))$.

Proof

call $F(P)$ good if $v(X) = v(Y)$ implies $v(F(X)) = v(F(Y))$

by structural induction we show that every formula is good:

- Base: trivial
- Step: case $F(P) = G(P) \circ H(P)$; assume $v(X) = v(Y)$:

$$\begin{aligned} v(F(X)) &= v(G(X) \circ H(X)) && \text{definition of } F \\ &= v(G(X)) \circ v(H(X)) && \text{definition of } v \\ &= v(G(Y)) \circ v(H(Y)) && \text{IH} \\ &= v(G(Y) \circ H(Y)) = v(F(Y)) \end{aligned}$$

similar for case $F(P) = \neg G(P)$ □

Proof (of the Replacement Theorem)

$X \equiv Y$ is a tautology; v an arbitrary valuation:

- $X \equiv Y$ implies $v(X \equiv Y) = \mathbf{t}$ by Exercise 2.4.4: $v(X) = v(Y)$
- $v(F(X)) = v(F(Y))$ previous Theorem
- $v(F(X) \equiv F(Y)) = \mathbf{t}$ Exercise □

suppose $F(A) = ((\neg P \vee Q) \wedge (R \uparrow A))$; as $(\neg P \vee Q) \equiv (P \to Q)$:

$$((\neg P \vee Q) \wedge (R \uparrow (\neg P \vee Q))) \equiv ((\neg P \vee Q) \wedge (R \uparrow (P \to Q)))$$

Definition negation normal form (NNF)

- all negation symbols occur in front of propositional letters

NB: every formula can be put into NNF

Uniform Notation

- basic binary connectives: all Primary Connectives
- all Secondary Connectives are defined
- group $(X \circ Y)$ and $\neg(X \circ Y)$:

Conjunctive

α	α_1	α_2
$X \wedge Y$	X	Y
$\neg(X \vee Y)$	$\neg X$	$\neg Y$
$\neg(X \rightarrow Y)$	X	$\neg Y$
$\neg(X \leftarrow Y)$	$\neg X$	Y
$\neg(X \uparrow Y)$	X	Y
$X \downarrow Y$	$\neg X$	$\neg Y$
$X \not\rightarrow Y$	X	$\neg Y$
$X \not\leftarrow Y$	$\neg X$	Y

Disjunctive

β	β_1	β_2
$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$X \vee Y$	X	Y
$X \rightarrow Y$	$\neg X$	Y
$X \leftarrow Y$	X	$\neg Y$
$X \uparrow Y$	$\neg X$	$\neg Y$
$\neg(X \downarrow Y)$	X	Y
$\neg(X \not\rightarrow Y)$	$\neg X$	Y
$\neg(X \not\leftarrow Y)$	X	$\neg Y$

Theorem for every valuation v, for all α- and β-formulas

- $v(\alpha) = v(\alpha_1) \wedge v(\alpha_2)$
- $v(\beta) = v(\beta_1) \vee v(\beta_2)$

Theorem Principle of Structural Induction

Every propositional formula has property **Q**, if

- Basis: for every atomic formula A: A and $\neg A$ have property **Q**
- Step:
  - X has property **Q** $\Longrightarrow$ $\neg\neg X$ has property **Q**
  - α_1, α_2 have property **Q** $\Longrightarrow$ α has property **Q**
  - β_1, β_2 have property **Q** $\Longrightarrow$ β has property **Q**

Proof by (original) principle of structural induction □

Theorem Principle of Structural Recursion

There is exactly one function f defined on **P** such that

- Basis: the value of f is explicitly defined on A and $\neg A$, A atomic
- Step:
  - $f(\neg\neg X)$ is defined in terms of $f(X)$
  - $f(\alpha)$ is defined in terms of $f(\alpha_1), f(\alpha_2)$
  - $f(\beta)$ is defined in terms of $f(\beta_1), f(\beta_2)$

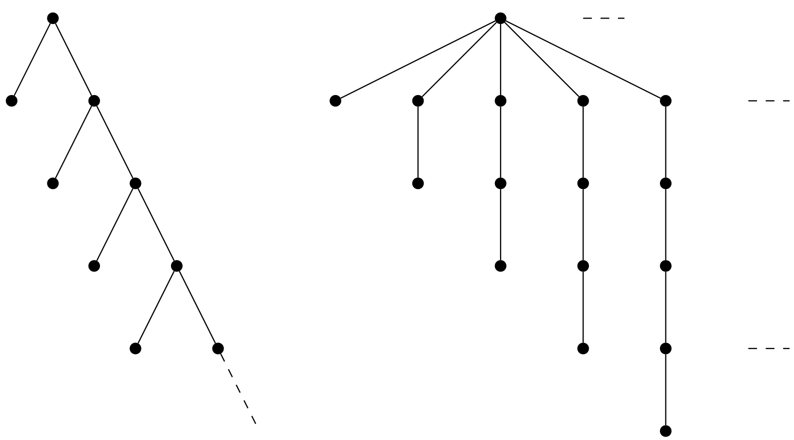
Example rank

- Base: $r(A) = r(\neg A) = 0$; $r(\top) = r(\bot) = 1$; $r(\neg\top) = r(\neg\bot) = 0$
- Step: $r(\neg\neg Z) = r(Z) + 1$; $r(\alpha) = r(\alpha_1) + r(\alpha_2) + 1$; $r(\beta) = r(\beta_1) + r(\beta_2) + 1$

König's Lemma

A tree that is finitely branching (i.e., each node has a finite number of children) must have an infinite branch.

Proof



□

- $[X_1, \dots, X_n]$ is the generalised disjunction of $X_1, \dots, X_n$
- for valuation v, we require $v([X_1, \dots, X_n]) = \mathbf{t}$ iff there exists i with $v(X_i) = \mathbf{t}$
- $\langle X_1, \dots, X_n \rangle$ is the generalised conjunction of $X_1, \dots, X_n$
- for valuation v, we require $v(\langle X_1, \dots, X_n \rangle) = \mathbf{t}$ iff for all i: $v(X_i) = \mathbf{t}$

note: $v([]) = \mathbf{f}$; $v(\langle\rangle) = \mathbf{t}$

Definition clause set

- a literal is $P_i, \neg P_i, \top, \bot$
- a clause is $[X_1, \dots, X_n]$ such that each X_i is a literal
- a clause set (or formula in CNF, or in clause form) is $\langle C_1, \dots, C_n \rangle$ such that each C_i is a clause

Definition Normal Form Algorithm

$S := \langle [X] \rangle$
loop
suppose $S = \langle D_1, \dots, D_k \rangle$; select D_i not a clause

- select N in D_i; N non-literal
- $N = \neg\top \Rightarrow$ replace N by $\bot$
- $N = \neg\bot \Rightarrow$ replace N by $\top$
- $N = \neg\neg Z \Rightarrow$ replace N by Z
- N is β-formula replace N by β_1, β_2
- N is α-formula replace D_i by $D_i(\alpha_1)$ and $D_i(\alpha_2)$

  $D_i(\alpha_i)$ is D_i with N replaced by α_i

Definition clause set reduction rules

$$\frac{\neg\neg Z}{Z} \qquad \frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top} \qquad \frac{\beta}{\begin{array}{c}\beta_1\\ \beta_2\end{array}} \qquad \frac{\alpha}{\alpha_1 | \alpha_2}$$

Lemma If S is a conjunction of disjunctions, and one of the clause set reduction rules is applied to S, producing S^*, then $S \equiv S^*$ is a tautology.

Proof

- Exercise 2.8.1
- Replacement Theorem (extended for generalised conjunctions and disjunctions) $\square$

Theorem

- The Normal Form Algorithm is correct.
- The Normal Form Algorithm is strongly normalising, i.e., no matter what choice is made, it terminates.

Proof (of strong normalisation)

- define the rank function r for generalised disjunctions
- fix a choice-sequence, associate with produced disjunctions their rank
- order the set of ranks in a (finitely branching) tree
- use König's Lemma $\square$

Definition dual clause set

- a dual clause is $\langle X_1, \dots, X_n \rangle$ such that each X_i is a literal
- a dual clause set (or formula in DNF, or in dual clause form) is $[D_1, \dots, D_n]$ such that each D_i is a dual clause

Definition dual clause set reduction rules

$$\frac{\neg\neg Z}{Z} \qquad \frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top} \qquad \frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 | \beta_2}$$

Summary

- the Replacement Theorem
- uniform notation
- König's Lemma
- normal forms: clause form & dual clause form
- normal form implementations