University of Malta
BSc(Hons.) IT Year I

Introduction to Logic and Logic Programming

Lecture 7

Proof Systems

- Truth Functional Completeness
- Formal Deductive Systems (Intro)

Department of Computer Science and AI
1988/9

Truth Functional Completeness

General issue is whether the set of connectives that we have introduced is **adequate**.

- Does a set of connectives allow us to express **all possible** truth functions?
- How many possible truth functions are there?
- Clearly depends on the truth function's arity. Let us proceed for truth functions of 1,2,3, ...n arguments.

How Many Functions for n=1

- How many possible truth tables are there for a one-argument truth function, i.e.
- How many **rows** does a one-argument truth function have? Answer: 2^1

P	f(P)
t	*
f	*

- How many distinct truth functions? Answer: 2^{2^1}

P	$f_0(P)$	$f_1(P)$	$f_2(P)$	$f_3(P)$
t	f	f	t	t
f	f	t	f	t

N.B All these can be defined in terms of $\wedge$, $\vee$ and -.

n=2: Binary Truth Functions

How many binary truth functions are possible?:

Answer: 2^{2^2}

P	Q	g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	$\cdots g_{15}$
t	t	f	f	f	f	f	f	f	f	t
t	f	f	f	f	f	t	t	t	t	t
f	t	f	f	t	t	f	f	t	t	t
f	f	f	t	f	t	f	t	f	t	t

For arity n there are 2^{2^n} truth functions

Building an Arbitrary Binary Truth Function

Problem: write down a truth function using only -, $\wedge$ and $\vee$ forgiven values of $v_1 - v_4$

P	Q	P*Q
T	T	v1
T	F	v2
F	T	v3
F	F	v4

- If all rows are false,

  $$P * Q = (P \wedge -P) \wedge (Q \wedge -Q)$$

- Otherwise create sentences that are true for exactly one row of the truth table.

  $$P \wedge Q,\ P \wedge -Q,\ -P \wedge Q,\ -P \wedge -Q$$

- Create a disjunction for the terms corresponding to each row where the value is true.

- The method can clearly be generalised to cover n-ary truth functions.

Example of Building an Arbitrary Truth Function

Problem: write down a truth function using only -, $\wedge$ and $\vee$ for

P	Q	P*Q	Sentence True Only in This Row
T	T	F	P $\wedge$ Q
T	F	T	P $\wedge$ -Q
F	T	T	-P $\wedge$ Q
F	F	F	-P $\wedge$ -Q

- Note that there is at least one row that is true.
- Create a disjunction over sentences true only in those rows that are truw

  (P $\wedge$ -Q) $\vee$ (-P $\wedge$ Q)

- Either $\vee$ or $\wedge$ (but not both) could be eliminated using DeMorgan's Laws.
- The method can clearly be generalised to cover n-ary truth functions.

Summary

- **Any n-ary truth function** can be constructed in terms of $\vee$, $\wedge$, and $-$.

- Clearly, one of $\vee$, $\wedge$ can be eliminated using DeMorgan's Laws.

- **Definition.**

  A set of operators is **complete** if it can be used to define all possible truth functions.

- Therefore the sets $\{\vee, -\}$, and $\{\wedge, -\}$ are complete sets of operators.

- To show that a given set of operators is complete it is sufficient to show that $\{\vee, -\}$ or $\{\wedge, -\}$ can be defined in terms of these operators since $\{\vee, -\}$ or $\{\wedge, -\}$ are complete sets.

The Sheffer Stroke

P	Q	xor($\oplus$)	Pierce($\downarrow$)	Sheffer(/)
T	T	F	F	F
T	F	T	F	T
F	T	T	F	T
F	F	F	T	T

Example Show that $\{/\}$ is a complete set.

- Define Negation: -P $=_{def}$ P/P

P	-P	P/P
T	F	F
F	T	T

- Define Disjunction: P v Q $=_{def}$ (P/P)/(Q/Q)

P	Q	(P v Q)	(P/P)/(Q/Q)
T	T	T	T
T	F	F	F
F	T	F	F
F	F	F	F

Formal Deductive Systems

Most of the methods discussed so far for establishing the validity of arguments ultimately depend on the evaluation of truth tables to discover tautologyhood.

$$\boxed{\begin{array}{c} P_1 \\ P_2 \\ \ldots \\ P_{n-1} \\ \hline P_n \end{array}} \text{ is valid}$$

if and only if

$$P_1, P_2, \ldots P_{n-1} \models P_n$$

if and only if

$$\models P_1 \& P_2, \& \ldots, \& P_{n-1} \rightarrow P_n$$

Such methods have certain disadvantages: when the number of variables becomes large, truth tables become intractable. For some arguments the number of variables is infinite.

Formal Deductive Systems
Rationale

We will now examine formal systems which employ **purely algebraic means** to determine that a particular sentence can be derived as a theorem.

The basic idea is then to show that there is a 1:1 correspondance between the theorems that can be derived and the tautologies of the underlying logic (in this case propositional calculus).

Formal Deductive System: Ingredients

1. **Formal Language:** an alphabet together with a set of syntactic rules for building well-formed formulas.

2. **Axioms:** a set of well-formed formulas from which all theorems will be derived. The set can be finite or infinite (specified using schemata). Example:

$$\mathcal{P} \rightarrow (\mathcal{Q} \rightarrow \mathcal{P})$$

3. **Inference Rules:** these enable new sentences to be derived from existing ones, and thus provide the means to define the notion of derivability that is used when constructing proofs. Example, *Modus Ponens (MP):*

   If $\mathcal{P}$ and $\mathcal{P} \rightarrow \mathcal{Q}$ are both derivable in the system, then $\mathcal{Q}$ is also derivable in the system.

Formal Deductive System

FDS

AXIOMS	$\Rightarrow$	$\boxed{\text{Language}}$
ASSUMPTIONS	$\Rightarrow$	$\boxed{\text{Inference Rules}}$
		$\Downarrow$
		SENTENCES

- System requires either axioms alone or assumptions alone or both.

- Output sentences are either
  - instances of axioms, or assumptions, or
  - derived as a result of a applying inference rules.

Different Formal Deductive Systems

System L: Minimal (one rule of inference, three axiom schemes, two operators). Hard to construct proofs. Easy to prove things about.

Fitch Style Systems (F. Fitch):

- 0 axioms
- 5 operators `=>`, $\sim$, `&`, `|`
- 13 rules of inference
- Natural deduction, ease of constructing proofs, readability of proofs.

Resolution Logic (J.A. Robinson 1965):

- 0 axioms
- 3 operators ($\sim$, `&`, `|`)
- 1 rule of inference (Resolution)
- Rationale: mechanisability (this is the system that underlies logic programming)

Proof and Theorem

Definition. A *proof* in a system is a sequence of formulas of that system such that each formula is either:

- An axiom or an instance of an axiom schema
- Derivable from earlier formulas in the sequence using one of the inference rules.

Definition. A proof whose last element is the formula $\mathcal{P}$ is called *a proof of $\mathcal{P}$ in the system.*

Definition. A formula $\mathcal{P}$ is a *theorem of the system* if there is a proof of $\mathcal{P}$ in the system. We write

$$\vdash_\sigma \mathcal{P}$$

To indicate that $\mathcal{P}$ is a theorem within a certain system σ. The subscript will be omitted when the context permits it.

Deduction

A notion closely related to proof is that of *deduction*. A proof is derived only from the axioms. A deduction allows the possibility of *additional assumptions*.

Definition. A deduction in a system is a sequence of formulas such that each formula is either

- an axiom or instance of an axiom schema
- one of the additional hypotheses
- derivable from earlier formulas in the sequence using an inference rule.

To indicate that $\mathcal{P}_n$ is derived from assumptions $\mathcal{P}_1\mathcal{P}_2, \ldots, \mathcal{P}_{n-1}$ we write

$$\mathcal{P}_1\mathcal{P}_2, \ldots, \mathcal{P}_{n-1} \vdash \mathcal{P}_n$$

Notation

Where appropriate, we employ the notation

$$\Gamma \vdash \mathcal{P}$$

to mean that $\mathcal{P}$ is deducible from the assumptions Γ. Note that Γ is a set, so that if we wanted to indicate $\mathcal{P}$ was derivable from a set of assumptions including a particular formula f, we might write

$$\Gamma \cup \{f\} \vdash \mathcal{P}$$

If Γ is the empty set, we write

$$\vdash \mathcal{P}$$

to indicate that p is a theorem. Strictly, we should write $\vdash_\sigma$ to show that the corresponding proof is carried out within System σ.

Checkpoint: What is the difference between $\vdash \mathcal{P}$ and $\models \mathcal{P}$?