Informatics 1 - Computation & Logic: Tutorial 7

Propositional Logic: Inference and Resolution

Week 10: 20–24 November 2016

Please attempt the entire worksheet in advance of the tutorial, and bring all work with you. Tutorials cannot function properly unless you study the material in advance. Attendance at tutorials is **obligatory**; please let the ITO know if you cannot attend.

You may work with others, indeed you should do so; but you must develop your own understanding; you can't phone a friend during the exam. If you do not master the coursework you are unlikely to pass the exams.

This tutorial comes in two parts. Part A concerns the new topic, inference. Part B is additional material on resolution—this may be useful if you still need to develop your understanding of this topic.

This tutorial exercise sheet was written by Paolo Besana, and extended by Thomas French Areti Manataki, and Michael Fourman. Send comments to `Michael.Fourman@ed.ac.uk`

Part A

Rules

In informatics we often use rules to define sets of things inductively. This means that we start with some basic things and give rules that say how more complex things are produced from these. A rule of the form:

$$\frac{\beta_1 \quad \cdots \quad \beta_n}{\alpha}$$

allows us to derive the **conclusion** α from the **assumptions** $\beta_1, \ldots, \beta_n$.

For example, consider the language whose sentences are expressions of propositional logic. If we write $\varphi \in \mathcal{E}$ to mean that φ is a well-formed expression, then the following rules can be used to generate the set expressions generated from a set $\mathcal{A}$ of atoms (the propositional letters).

$$\frac{}{A \in \mathcal{E}} \text{ for each } A \in \mathcal{A} \qquad \frac{\varphi \in \mathcal{E}}{\neg\varphi \in \mathcal{E}} \qquad \frac{\varphi \in \mathcal{E} \quad \psi \in \mathcal{E}}{\varphi \vee \psi \in \mathcal{E}} \qquad \frac{\varphi \in \mathcal{E} \quad \psi \in \mathcal{E}}{\varphi \wedge \psi \in \mathcal{E}} \qquad \frac{\varphi \in \mathcal{E} \quad \psi \in \mathcal{E}}{\varphi \to \psi \in \mathcal{E}}$$

We start from atomic propositions and use simple rules to describe how the well-formed expressions, $\varphi \in \mathcal{E}$, are built from simpler ones using the connectives.

We can write this in pseudo Haskell

```
data Prop = Atom Name
          | ¬ Prop
          | Prop :∨:  Prop
          | Prop :∧:  Prop
          | Prop :→:  Prop
```

An Algebraic Data Type in Haskell is a Data Type whose elements can be "built" starting from some basic elements by means of some "data constructors". The constructors can use elements of the inductive Data Type itself as arguments (that's why we also call them "recursive" Data Types), but also elements of of other data types.

1. Write a data type in Haskell to represent regular expressions.

The rules above generate all well-formed expressions. What about more interesting sets? For example, is it possible to write rules that will only generate tautologies? The answer is actually, *Yes*.

However it is simpler to give rules that generate valid entailments, or valid sequents.

Entailment

Entailment $\Gamma \models \Delta$ is a relation between two finite sets of expressions, the *antecedents*, Γ and the *succedents*, Δ. This entailment is **valid** iff whenever a valuation $\mathbf{V}$ makes all of its antecedents (the formulæ before the turnstile) true, it also makes one of the succedents, the formulæ after the turnstile, true.

A **counterexample** to an entailment $\Gamma \models \Delta$ is a valuation that make all of the antecedents, to the left of the turnstile true, while making all of the succedents, to the right of the turnstile false.

An entailment is valid iff there is no counterexample.

We call this *semantic* entailment, since it is defined in terms of the semantics (meaning) of the constituent expressions.

In this tutorial, you will look at a variety of similar relations, $\Gamma \vdash \Delta$, between two finite sets of expressions, the *antecedents*, Γ and the *succedents*, Δ. These relations are generated by syntactic rules. This means that we look only at the form of the expressions. The assumptions and conclusions of our rules are entailments that match particular patterns.

Our goal is to give a set of rules such that $\Gamma \vdash \Delta$ (defined by syntactic rules) iff $\Gamma \models \Delta$ (defined by semantics).

A rule is **sound** iff whenever all of its assumptions are valid then so is its conclusion. Equivalently, rule is **sound** iff whenever there is a counterexample to its conclusion then there is a counterexample to at least one of its assumptions.

If the relation $\Gamma \vdash \Delta$ is generated by sound rules, then $\Gamma \vdash \Delta$ implies $\Gamma \models \Delta$ — if we can derive $\Gamma \vdash \Delta$ using the rules, then $\Gamma \models \Delta$.

We say that a set of rules is **complete** if every valid entailment can be derived, that is if $\Gamma \models \Delta$ implies $\Gamma \vdash \Delta$.

History

We will present the sequent calculus, a sound and complete system of rules, discovered by Gentzen in 1933. Before that we briefly discuss some earlier systems.

Axioms and rules Early attempts to formalise deduction used systems of axioms and rules. These correspond to systems that only consider entailments of the form $\vdash \varphi$, with no antecedents, and a single succedent. This entailment is valid iff φ is a tautology.

An example, introduced by Hilbert, is a simple proof system based on a language with negation and implication as the only connectives, with three axioms (rules with no assumptions), together with the rule of *modus ponens* (MP).[1]

$$\frac{}{A \to (B \to A)}\ (A1) \qquad \frac{}{(A \to (B \to C)) \to ((A \to B) \to (A \to C))}\ (A2)$$

$$\frac{}{(\neg B \to \neg A) \to ((\neg B \to A) \to B)}\ (A3) \qquad \frac{A \quad A \to B}{B}\ (MP)$$

We can use these rules to prove various tautologies. For example

$$\dfrac{\dfrac{}{A \to (A \to A)}\ (A1) \quad \dfrac{\dfrac{}{A \to ((A \to A) \to A) \to A))}\ (A1) \quad \dfrac{}{(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))}\ (A2)}{(A \to (A \to A)) \to (A \to A)}\ (MP)}{A \to A}\ (MP)$$

2. Each line represents an application of one of the rules. For example, rule $(A1)$ is applied twice: once with $A = A,\ B = A$, once with $A = A,\ B = (A \to A) \to A$.

   For each of the remaining rule applications, how do the variables in the rule match with the variables in the problem?

3. Show that each of these rules is sound.

Hilbert's system is complete in the sense that $\vdash \varphi$ iff $\models \varphi$. A formula φ is provalble in the system iff it is a tautology. We can write any propositional connective in terms of $\neg, \to$, so this gives a sound and complete system of rules for propositional logic, but derivations in this system are often convoluted.

Hilbert systems have been developed for many logics. They typically have many axioms (rules with no assumptions), and few other rules.

[1] Modus Ponens was known to the ancient Greeks in the 3rd century BC, but complete systems of inference even for propositional logic, were not discovered until the late 19th century.

Natural Deduction Natural deduction is a proof calculus that uses entailments with a single conclusion. Here are some simple rules (all sound, but not a complete set as we give no rules for negation) for generating valid entailments with a single succedent.

$$\frac{}{\Gamma, X \vdash X}\ (I) \qquad \frac{\Gamma \vdash X \quad \Delta, X \vdash Y}{\Gamma, \Delta \vdash Y}\ Cut$$

$$\frac{\Gamma \vdash X \quad \Gamma \vdash Y}{\Gamma \vdash X \wedge Y}\ (\wedge) \qquad \frac{\Gamma, X \vdash Z \quad \Gamma, Y \vdash Z}{\Gamma, X \vee Y \vdash Z}\ (\vee) \qquad \frac{\Gamma, X \vdash Y}{\Gamma \vdash X \to Y}\ (\to)$$

Here, Γ and Δ are variables that range over sets of expressions of propositional logic, and X, Y and Z are variables that range over expressions themselves.
The interpretation of $\Gamma \vdash X$ is *It follows from* Γ *that* X. The rules are supposed to model "natural" forms of argument. For example, the $\to$ introduction rule corresponds to arguing that $X \to Y$ by showing that Y follows from X.

1. Show that these rules are sound, by showing that if a valuation is a counterexample to the conclusion then it is a counterexample to at least one of the assumptions. Why is this sufficient to show the rule is sound?

The *immediate* rule (I) has no assumptions. The double line used for the other three structural rules means that the rule can be used in either direction. The entailment below the double line is valid iff *all* of the entailments above the line are valid. Read from top to bottom, they are called *introduction rules* ($^+$), since they introduce a new connective into the argument. Read from bottom to top, they are *elimination rules* ($^-$) since a connective is eliminated.

These rules are designed to allow us to produce *valid* entailments. We say that a valuation validates $\mathcal{A} \vdash X$ if it makes at least one of the assumptions $A \in \mathcal{A}$ *false* or it makes X *true*. The entailment is valid iffit is validated by **every** valuation. So it is valid iff any valuation that makes all the premisses in $\mathcal{A}$ true also makes X true.[2]

Using these rules we can prove validity. For example, the following proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \to B, C \vdash A \to B}\ (I)}{A \to B, C, A \vdash B}\ (\to^-) \qquad \dfrac{}{A \to B, C, A \vdash C}\ (I)}{A \to B, C, A \vdash B \wedge C}\ (\wedge^+)}{A \to B, C \vdash A \to (B \wedge C)}\ (\to^+)$$

shows that $A \to B, C \vdash A \to (B \wedge C)$ is valid.

We start with the goal of proving the bottom line — showing that it is valid. The fact that all of the rules are sound, and we can derive the goal starting from no assumptions shows that the goal is valid.

To find such a proof we start with the bottom line as our goal. Matching this goal with the conclusion of a rule allows us to replace the original goal with the assumptions of the rule. If we can derive these assumptions, then the rule we have just introduced allows us to derive the original goal.

[2]Note that the rule (I) is certainly sound, since X occurs on both sides of the turnstile.

2. Use the natural deduction rules given above to show that

$$A \to B,\ B \to C \vdash A \to C$$

This system is complete for the fragment of propositional logic without negation, and it can be expanded to include negation, but finding proofs is often tricky. When we mix introduction and elimination rules, and search for a proof, it is sometimes hard to tell whether we are making progress, or just going round in circles.

Sequent Calculus

As we saw in the case of DFA and NFA, it is sometimes helpful to place our objects of study in a wider context. Although every NFA is equivalent to a DFA, in many ways NFA are easier to construct, and to reason about.

Gentzen introduced the idea of sequents that include multiple premisses *and* multiple conclusions. Within this context, he gave an elegant set of rules that eliminate the searching from propositional proof.

Here is Gentzen's beautifully symmetric set of rules (Γ, Δ vary over finite sets of expressions; A, B vary over expressions):

$$\frac{}{\Gamma, A \vdash \Delta, A}\ (I)$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ (\wedge L) \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ (\vee R)$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ (\vee L) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ (\wedge R)$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\ (\to L) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\ (\to R)$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ (\neg L) \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ (\neg R)$$

These are all introduction rules. This means that a goal-directed proof will always produce simpler and simpler sequents (but maybe many many simpler sequents) as our trees grow upwards.

These rules all share a crucial property:

> For each of these rules,
> a valuation is a counterexample to the conclusion
> **iff** it is a counterexample to (at least) one of the assumptions.

From this, it follows that each rule is sound — if the assumptions are valid then so is the conclusion.

It also follows that this system of rules is complete. Consider a proof tree using these rules. First observe that if we have a counterexample that contradicts any one of the assumptions of the proof tree then it is a counterexample to the conclusion. Furthermore, any proof attempt either succeeds with every leaf of the tree being reached by the immediate rule, with no assumptions, or fails with sequents containing only atomic propositions, such that the set of sequents to the left of the turnstile is disjoint from the set to the right. A valuation making everything to the left true and everything to the right false, provides a counterexample to such a sequent.

3. Show that each of these rules has the crucial property.

4. The following rules are suggested for xor ($\oplus$).
   Do they have the crucial property?

$$\frac{\Gamma, A \vdash B, \Delta \quad \Gamma, B \vdash A, \Delta}{\Gamma, A \oplus B \vdash \Delta}\ (\oplus L) \qquad \frac{\Gamma, A, B \vdash \Delta \quad \Gamma \vdash A, B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ (\oplus R)$$

   Suggest corresponding rules for $\leftrightarrow$, and show that they have the crucial property.

5. Use the rules for $\leftrightarrow$ in your answer to Question 4 and the rules for $\oplus$ given there, to show that

$$(A \leftrightarrow B) \leftrightarrow C \vdash (A \oplus B) \oplus C$$

6. For each of the entailments listed below, construct a proof tree, by applying the Gentzen rules until the leaves of your tree contain no connectives. Then say whether the entailment is valid. How can a proof attempt fail? How can you can construct a falsifying valuation from a failed proof attempt?

   (a) $\mathsf{B} \wedge \mathsf{C} \vdash (\mathsf{A} \to \mathsf{B}) \wedge (\mathsf{A} \to \mathsf{C})$

   (b) $\mathsf{A} \wedge (\mathsf{B} \wedge \mathsf{C}) \vdash (\mathsf{A} \wedge \mathsf{B}) \wedge \mathsf{C}$

   (c) $\mathsf{A} \to \mathsf{B}, \mathsf{A} \wedge \mathsf{C} \vdash \mathsf{B} \wedge C$

   (d) $\mathsf{A} \vee \mathsf{B} \to \mathsf{C}$, $\mathsf{C} \to \mathsf{A} \vdash \mathsf{C} \to \mathsf{B}$

   (e) $\mathsf{A} \to \mathsf{C} \vdash \mathsf{A} \to (\mathsf{B} \vee \mathsf{C})$

Part B

In this section we revisit the use of resolution to determine the validity of an entailment, and consider an alternative treatment in which the entailment relation is generated by inference rules. We use two sets of constraints, $\mathcal{A}$ and $\mathcal{B}$, as running examples:

$\mathcal{A}$

(i) $(C \vee \neg D) \rightarrow (A \vee B)$
(ii) $(A \vee \neg B) \rightarrow (D \rightarrow C)$
(iii) $(A \vee B) \rightarrow (C \vee D)$
(iv) $D \rightarrow (B \rightarrow A)$
(v) $B \rightarrow (C \rightarrow D)$
(vi) $C \rightarrow (A \rightarrow B)$

$\mathcal{B}$

(i) $(C \vee \neg D) \rightarrow (A \vee B)$
(ii) $(A \vee \neg B) \rightarrow (D \rightarrow C)$
(iii) $(A \vee B) \rightarrow (C \vee D)$
(iv) $(C \vee D) \rightarrow (A \vee \neg B)$
(v) $A \rightarrow \neg C$

Satisfaction

1. For each set of constraints use the Karnaugh map provided to show which states are *excluded* by each constraint.

   For example, the constraint $\mathcal{A}(i)$ is $(C \vee \neg D) \rightarrow (A \vee B)$; the states excluded are those that make $C \vee \neg D$ true and make $A \vee B$ false. A state makes $C \vee \neg D$ true iff it is a row where CD = 00, 11, or 10; it makes $A \vee B$ false if it is in the column where AB = 00. So there are three states excluded by this constraint, as shown on the Karnaugh map.

$\mathcal{A}$

CD \ AB	00	01	11	10
00	(i)			
01				
11	(i)			
10	(i)			

$\mathcal{B}$

CD \ AB	00	01	11	10
00				
01				
11				
10				

   Can you see from your maps whether each set of constraints is satisfiable?

In tutorial 4, we introduced conjunctive normal form (CNF), and showed that,

> any collection of constraints expressed in propositional logic is equivalent to a conjunction of clauses, where each clause is a disjunction of literals.

CNF

2. (a) For each set of constraints, use Boolean algebra to derive an equivalent conjunctive normal form.

$\mathcal{A}$

$\mathcal{B}$

(b) For each CNF show on the Karnaugh map which states *excluded* by each clause.

$\mathcal{A}$

CD \ AB	00	01	11	10
00				
01				
11				
10				

$\mathcal{B}$

CD \ AB	00	01	11	10
00				
01				
11				
10				

Resolution

In tutorial 4, we introduced resolution as a method for determining whether a given set of constraints, expressed in CNF, is consistent. We introduced examples of the resolution rule (Tutorial 4 p.3).

3. Formulate the resolution rule, and show that it is sound, in the sense that if a state satisfies the two constraints being resolved, then it satisfies the resolvent.

This means that if from some initial set of constraints (clauses), we can use resolution to derive the empty clause (which is the impossible constraint, not satisfied by any valuation), then every valuation must be refuted by at least one of the initial constraints.

> So, if we can derive the empty clause then the inital set of constraints is inconsistent: there is no valuation that satisfies all the constraints.

4. Use resolution to show that one of the two sets of clauses $\mathcal{A}, \mathcal{B}$ is inconsistent.

To show that resolution is **complete** we must show that,

> If the inital set of constraints is inconsistent,
> then we can derive the empty clause.

It suffices to show that if we cannot derive the empty clause then there is a valuation that satisfies the initial set of clauses — because the existence of such a valuation shows that the set of clauses is consistent.

We say that a literal whose negation does not occur in any clause is **pure**. We can easily satisfy all clauses that contain a pure literal: if it is of the form $\neg A$ we let $\mathbf{V}(A) = \bot$; if it is of the form A we let $\mathbf{V}(A) = \top$.

In fact, if any valuation, $\mathbf{W}$, satisfies all of our constraints, then so does the valuation we obtain from $\mathbf{W}$ by making all pure literals true. So if we are only concerned with satisfiability, we can start by making all pure literals true, eliminate all clauses that containg any of them, and focus on finding a valuation of the remaining variables that satisfies the remaining clauses.

5. For each set of clauses, $\mathcal{A}, \mathcal{B}$, say how many resolution pairs there are for each variable.

	A	B	C	D
$\mathcal{A}$				
$\mathcal{B}$				

How many pairs would you find for a pure literal? ☐

The Davis-Putnam resolution procedure is based on a step that simplifies such a set of clauses, $\mathcal{X}$, by using resolution to eliminate one variable (for example, A), by resolving all available pairs for resolution using that variable. We take away all the clauses that mention A and add the results of resolving each $A, \neg A$ pair—except for any trivial results, clauses that include both some literal and its negation are trivial constraints. This produces a set of clauses, $\mathcal{X}_{\setminus A}$ that don't mention A.

6. For each example, $\mathcal{A}, \mathcal{B}$, what clauses are in the set after resolution on A?

$\mathcal{A}_{\setminus A} =$	$\mathcal{B}_{\setminus A} =$

This set, $\mathcal{X}_{\setminus A}$, has the property that any valuation of the remaining variables that satifies this set of constraints, $\mathcal{X}_{\setminus A}$, can be extended, by providing a suitable value for A, to a valuation that satisfies all the constraints in $\mathcal{X}$.

If a valuation, $\mathbf{V}$, satisfies $\mathcal{X}_{\setminus A}$ then either $\mathbf{V}$ makes all the clauses in $\mathcal{X}$ that include the literal $\neg A$ true, or $\mathbf{V}$ makes all the clauses in $\mathcal{X}$ that include the literal $\neg A$ true (or both). In either case, deleting all clauses satisfied by $\mathbf{V}$, if A appears in the remaining clauses, it will be as a pure literal, either A or $\neg A$. We can extend $\mathbf{V}$ with the value for A required makes this literal true.

This is the crucial property that allows us to construct a satisfying valuation if resolution fails to produce the empty clause. Unless we can produce the empty clause, resolution will end with every literal pure.

7. For each example, $\mathcal{X} = \mathcal{A}, \mathcal{B}$, explain how, *if* you were given a valuation for the remaining variables, B, C, D, satisfying every clause in $\mathcal{X}_{\setminus A}$, you could choose a valuation for A that would satisfy every clause in $\mathcal{X}$.

$\mathcal{A}$	$\mathcal{B}$

8. Suppose resolution fails to produce the empty clause,

   (a) How can you construct a counterexample to the remaining constraints?

   (b) It is possible that no clauses non-trivial remain. When does this happen? In this case, how do you construct a satisfying valuation?

9. For each example, $\mathcal{X} = \mathcal{A}, \mathcal{B}$ complete the procedure by resolving successively on all available pairs for each remaining variable B, C, D in turn.

   In each case, stop if at any stage you produce the empty clause.

$(\mathcal{A}_{\setminus A})_{\setminus B} =$	$(\mathcal{B}_{\setminus A})_{\setminus B} =$
$((\mathcal{A}_{\setminus A})_{\setminus B})_{\setminus C} =$	$((\mathcal{B}_{\setminus A})_{\setminus B})_{\setminus C} =$
$(((\mathcal{A}_{\setminus A})_{\setminus B})_{\setminus C})_{\setminus D} =$	$(((\mathcal{B}_{\setminus A})_{\setminus B})_{\setminus C})_{\setminus D} =$
Is there a satisfying valuation for $(((\mathcal{A}_{\setminus A})_{\setminus B})_{\setminus C})_{\setminus D}$? If so, give a satisfying valuation for $\mathcal{A}$.	Is there a satisfying valuation for $(((\mathcal{B}_{\setminus A})_{\setminus B})_{\setminus C})_{\setminus D}$? If so, give a satisfying valuation for $\mathcal{B}$.