PHIL2002

Daniel Paparo

1 Propositional Logic

1.1 Validity

Logic: The study of what follows from what.

Proposition: A claim that can either be true or false

Arguments: The sequence of propositions that form arguments

Validity: An argument is valid when it is impossible for the premises to be true and the conclusion to be false.

Potentially the most important part of the course is the idea of validity - An argument cannot be valid if it is possible for the conclusion to be false with the arguments all being true. Validity is made up two components:

- Necessary truth preservation: The truth of the premises is preserved into the conclusion. This is to say that we can change the objects in the argument to be something else, and the truth still holds.

Validity = Necessary truth preservation in virtue of form

1.2 Propositional joints

Soundness: An argument is sound when it is *valid* and all of its premises are true.

Negation: Saying of the world that something is not the case.

Conjuction: Saying of the world that x and y are the case

Disjunction: Saying of the world that x or y are the caes

Conditional: Saying of the world that if something is the case then another thing must be the case too

Biconditional: Says of the world that two things stand or fall together.

1.3 The language of propositional logic

In propositional logic all basic propositions are declared using a single capital letter A, B, C, D etc. and can take on any meaning we chose. We also use declare joints in the following:

Symbol	Joint	Meaning
$\vee$	Disjunction	Or
$\wedge$	Conjuction	And
$\neg$	Negation	Not
$\rightarrow$	Conditional	If/then
$\Longleftrightarrow$	Biconditional	If and only if

Well Formed Formulae: Grammatical type rules used to define a good argument versus a bad one, and are in the following form:

Well formed formulae
$\neg\alpha$
$(\alpha \vee \beta)$
$(\alpha \wedge \beta)$
$(\alpha \rightarrow \beta)$
$(\alpha \iff \beta)$

Note: When doing translations for any proposition, the main connective always has the fewest parentheses around it.

1.4 Translation

We do not need to translate everything in the english sentence to logic - only the logical structure of the sentence. For a basic proposition we are focused on whether each component is *true* or *false*. For a compound proposition we are seeing whether the basic propositions are true or false.

1.5 Truth Tables

Truth tables are useful for visualising the outcomes for non-bassic claims. The truth table rules are as follows:

A	B	$\neg A$
T	T	F
T	F	F
F	T	T
F	F	T

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

A	B	$A \iff B$
T	T	T
T	F	F
F	T	F
F	F	T

With truth tables of compound propositions we are working out way from the inner most joint to the outermost. Truth tables represent the sementics of the language - they tell us when each compound proposition is true or false for all possible values of each proposition.

There are 2^n rows for each truth table, where n is the number of of basic propositions.

1.6 Steps to solve PL tables

1. Place propositions in a truth table with the left columns of the basic propositions (with appropriate T and F values)
2. Find the inner most connective.
3. Work outwards determining what each connective and bracket will be (T or F) based on the basic or compound propositions within.
4. Cross off connectives as they have been completed - Once the table is completely finished a single T or F will remain

Tautology: A T in every row - also refered to as a logical truth
Contradiction: A F in every row - also called a logical falsehood
Satisfiable: A T in some or all rows - not a contradiction.
Nontautology: F in some or all rows - not a tautology.

By constructing a joint truth table we are able to represent multiple propositions in an argument. We simply do this by using one truth table with multiple propositions.

Equivalent: Same values in every row
Jointly satisfiable: Some row in which both propositions are T
Jointly unsatisfiable: No rows in which both are T
Contradictory: Jointly unsatisfiable and no row in which both are F.
Contraries: Jointly unsatisfiable and some row in which both are F.

We can use the truth tables as a proof of validity or invalidity. To do this we take the entire argument and put it within a truth table. Once we have solved each section of the truth table we look at all the possibnilities, and if there is a case where the premises are all true, and the conclusion is false - then it is invalid.

A counter example can be found once we have found asserted that the argument is invalid. The counter argument is just a set of the value of basic propositions which result in the argument having all true propsositions and a false conclusion.

A truth table camn also be used to prove if an argument is unsound (note that it cannot prove soundness). If the argument is invalid it must be unsound - but this does not count for all unsound cases.

A general result can be found when we substitute basic propositions with a variable in some argument - these variables can contain a basic or complex proposition. By solving the substituted argument we find a general result for a problem - potentially that it is a tautology or a contradiction etc.

1.7 Truth trees

Note: View the attatched sheet for the truth tree rules

Truth trees provide a more efficient method of generating the validity or invalidity of an argument. The closure rule is important for determining if a tree is valid or not, and is broken into two subrules:

1. Whenever we find a basic proposition and its negation on a branch in a tree we close the branch with an X
2. A tree is completed when either every branch closes or when every tree rule that can be applied has been applied.

An argument is valid when every branch on the tree for an argument closes. An argument is invalid when the tree is complete and atleast one branch on the truth tree for that argument remains opened - this is the line for our counter example.

To find the validity of the argument we must assume that all of the premises of an argument are true, and that the negation of the conclusion of the argument is true (that is that the conclusion is false). If the tree closes under this assumption then it is not possible for the premises to be true and the conclusion to be false - hence is valid.

The counter example of an invalid tree is essentially the branches (and the values of the basic propositions along this) it takes to go from an opened end back to the base of the tree.

1.8 Steps to solve PL trees

0. Negate the conclusion
1. Generate a basic truth tree for each of the popositions
2. Link the mini truth trees into one mega tree
3. Close off any closed branches when there is a contradiction of basic propositions
4. Once all trees have been added together any opened ends should be noted as counter examples
5. compare counter examples to the given answers and select the most appropriate

2 Monadic predicate logic

2.1 Syntax of MPL

Propositional logic cannot cover a situation like the following:

All people are animals. Sally is a person. Therefore, sally is an animal.

This is because the above argument would need to be represented as:

$$A$$

$$B$$

$$\therefore C$$

Which even though is logically valid (based on the written version) - is not valid by using a truth trees. To suppliment propositional logic we must use monadic predicate logic.

Monadic predicate logic uses names in lower case letters a,b,c etc. and variables in the reserved lower case letters t,u,v,w,x,y,z. The unreserved letters (names) can take on some object within the world. Uppercase letters are used to denote a preditcate, which holds a particular property for a name. These uppercase letters are followed by a subscript lower case letter.

$$P_x$$

is a predicate P which has a variable x.

To handle the scope of some variable within the world we must use quantifiers. $\forall$ denotes "for all" and means that "for every object within the universe (of the question), the following will apply. This is called the universal quantifier. The universal quantifier is followed by a variable - $\forall_x$ will denote "for every object, x, in the universe...".

There is also a second quantifier - the existential quantifier $\exists_x$ which translates to "there exists some x in the universe such that...".

There are three important points about quantifiers:

- Quantifiers can be translated into one another via the use of the negation symbol.
- Quantifiers have a scope which they range over. Essentially the quantifier will only work on the predicate immediately next to it or the brackets next to it.
- Bound and unbounded variables can occur. A well formed formula with no free ariables is a closed WFF. and free variables creates an open WFF. Open variables occur when there is no quantifier that is binding to a variable - a free variable.

Note that a variable is bound to the closest quantifier which affects its variable.

Extensions are a concept of a thing that possessess the property in question - that is that an extension is a little sub universe in which the predicates apply. Every predicate corresponds to some set of objects, the set is its extension.

Expression	Meaning
$\forall_x P_x$	Everything in the universe has property P
$\exists_x P_x$	Something in the universe has propert P
$\neg\forall_x P_x$	Not everything in the universe (atleast one is not included)
$\neg\exists_x P_x$	Nothing in the universe (no objects at all)

The complex quantified propositions will say something about the extensions of various predicates, which is a way of talking about what things have what properties.

2.2 Validity in MPL

Recall that validity is a state in which the conclusion cannot be false given that the premises are true. In PL validity of an argument was based on charting the possible combinations of truth and falsity of the basic propositions in an argument. For MPL, validity is still a function of possible situations. An argument is valid in MPL when there is no model on which all the premises are

true and the conclusion false, which is to say that we cannot build any such model whatsoever (models are explained below). If there is no model on which all of the premises are true and the conclusion is false, then it follows that there is no possible way to make all of the premises true and the conclusion false.

2.2.1 Actuality

The concept of actualatiy has three parameters: 1. the world contains a range of objects 2. We use names to pick out some of those objects 3. The objects that exist possess various properties

The actual world sets these parameters in a certain way, the actual world is one in which a particular group of objects exist, that are named in a particular way and that possess certain properties.

2.2.2 Possibilities

The possibilities that were interested in are like the actual world:

- Each possibility has some objects
- Each possibility is one in which the objects have certain properties
- Each possibility is one in which some of the objects are named.

For each possibility we are interested in:

1. There exists some number of objects n, where $n > 1$. We call this the domain of objects
2. Every name that occurs at that possibility picks out some object. That is every name refers to something in the domain. Note that we impose the following simplifcation: multiple names can refer to the same object, but the same name cannot refer to multiple objects
3. Every property that exists at a possibility is assigned to some predicate, which has an extension: a set of objects from the domain. Note that an extension can be empty.

Instead of working with possibilities we are working with models - a model is a specification of a possiblity where the specification does not need to specify the possiblity completely.

2.2.3 Models

A model is essentially a subset of a universe - a defined subset in which the argument takes place. A model is:

- A specification of a domain of objects
- A specification of the extension of any predicates
- The assignment of an object to each name, where the same object can be named twice, but the same name cannot name two objects

For example:

Objects that exist: 1st plaent from the sun, 2nd planet from the sun, 3rd planet from the sun, 4th planet from the son.

Properties that exist: is red, is blue, is green

Names that refer: Mars: the first planet from the sun Jupiter: The second planet from the sun Saturn: the third planet from the sun

Objects possessing properties: The first planet from the sun is blue, the second planet from the sun is blue, the third planet from the sun is red, the forth planet from the sun is green.

This domain can be more formalised in the following way:

$$Domain : \{1st\ planet,\ 2nd\ planet,\ 3rd\ planet,\ 4th\ planet\}$$
$$B_x : \{1st, 2nd\}$$
$$R_x : \{3rd\}$$
$$G_x : \{4th\}$$

$$a = 1st$$
$$b = 2nd$$
$$c = 3rd$$

With this given model we are able to do work on the MPL propositions. For example:

$$\forall_x(G_x \rightarrow R_x)$$

This is not true because there is atleast one green thing that is not red.

$$\exists_x R_x$$

This is true because there atleasts atleast one thing that is red.

We need to determine whether an argument is valid or not - to do this we use a simple extension of the truth tree method. Akk the same rules of truth trees apply, with the following rules:

Closure rule: Whenever we find a basic proposition and its negation on a branch in our tree we must close the branch (same as in propositional logic).

Completion rule: A tree is completed either when every branch closes or every tree rule that can be applied has been applied. This rule must be adapted for MPL.

Negated quantifier rule: Whenever a quantifier is negated (like $\neg\forall$) it will transform into the other quantifier, with negated predicats following ($\exists\neg P$).

Existential rule: When a existential is reached we must apply a new name (one that hasnt been used in preceeding branches of the tree) to the variable owned by the existential.

Universal rule: When an universal quantifier is reached we can apply any name we would like to it - it makes sense to apply a name which has already been used in the tree.

2.2.4 Completion and saturation

We face a problem with the existential quantifier where we could potentially just keep calling new variables on into infinity. We need a rule to deal with this. A tree in MPL is completed when:

1. Every branch is closed

Or

2. Every branch in the tree is saturated

A branch will be saturated when: 1. Every formula on it has had the relevent rule applied to it 2. Every formula on it whose major operator is a universal quantifier: * Has had the universal quantifier applied to it atleast once and, * Has had the rule applied to it once for each name that appears previously on the same branch

2.3 MPL Trees

MPL trees have an order of operations:

1. Propositional logic rules
2. Negated quantifier rules
3. Unnegated existential quantifier rules
4. Unnegated universal quantifier rules
5. Return to step 1 and continue to cycle until no more rules can be applied, or the tree is saturated.

2.3.1 Counter models

The counter model can simply be read from a branch of the treen which is left open - With MPL we are locing for a model which is a counter model. To build a countermodel for MPL we must specify:

1. A domain of objects
2. Extension for each predicate in the tree
3. Referents for each name in the tree

2.4 Infinite trees

Consider the following argument:

$$\forall_x \exists_x (P_x \wedge Q_y)$$
$$\therefore \forall_x Q_x$$

The argument is invalid, but it is also infinite. This is because the universal quantifier in the first premise keeps calling an existential which calls up a new name, which forces us to run it through the universal quantifier again forever! There are two things to note about infinite trees:

1. If a tree has an infinite branch then it is invalid, and can have a countermodel read off of it
2. You cant really saturate the tree. So when we find it we must stop what we are doing and simply end the branch.

2.5 Steps to solve MPL trees

0. Negate the conclusion!
1. Start with any non branching propositions (which dont have a quantifier) and then move to branching propositions
2. Flip all negated quantifiers
3. Start with any existentials and give them names for their variables ASAP
4. Now start on universals and saturate each (with names from higher in the tree)
    - If you notice that the existential keeps being called by an internal Existential end the branch - this is an open end
5. Keep repeating until the tree is completely saturated
6. Either the tree is valid or read off a countermodel

3 General predicate logic and Identity

We need to expand our language so that it can adequately handle situations in which we require multiple place predicates such as in " Every person is taller than every cat". We cannot do this in MPL because it is unable to deal with things in relationship to eachother. There is a very distinct difference we must draw between relations versus properties:

Properties: Tells us about how something is. For instance, we say that a chair has the property of being red.

Relations: Tell us about a relationship or connection between two or more things. For instance, we say that Sally is taller than Billy, that the beer is in the fridge, that Canberra is between Sydney and Melbourne.

In GPL we add multiple place predicates (such as "T_{xy} : x is taller than y").

3.0.1 Relations

We must imagine the universe as a big bag - When we use a quantifier it tells us something about what we can pick out of the bag. For example: $\forall_x P_x$ tells us that for everything we pick out of the bag, it will have property P_x, and $\forall_x \forall_y (P_x \wedge Q_y)$ Tells us that if we pick one thing (x) out of the bag and put it back in, then pick a second thing (y) (note that these can be the same thing), x will have property P and y will have property Q.

3.1 Syntax of GPL

We now have a few rules for GPL:

1. Where P^n is any n-placed predicate and $t_1 ... t_n$ are any terms, the following is a wff: $P^n_{t_1 ... t_n}$. That is that an n-place predicate followed by any mixture of n names or variables is a well formed formula
2. The MPL WFF all apply
3. Nothing else is a well formed formula

The order of the terms after a predicate with two or more places really matter! This can be shown in the following:

let $T_{xy} = x$ is taller than y.

if we have a world in which daniel is taller than tina, then we must assign daniel as x and tina as y. and therefore when we call these, they must be in their appropriate order.

3.1.1 Nested quantifiers

Sometimes the order of the quantifiers also matters, we have the following possible combinations (for the following let P_{xy} be "x likes y"):

Nested quantifier	Meaning	Order matters
$\forall_x \forall_y P_{xy}$	Everything likes everything	NO
$\forall_y \forall_x P_{xy}$	Everything likes everything	NO
$\forall_x \exists_y P_{xy}$	Everything has atleast one thing it likes	YES
$\forall_y \exists_x P_{xy}$	Everything is liked by something	YES

Nested quantifier	Meaning	Order matters
$\exists_x \forall_y P_{xy}$	Something likes everything	YES
$\exists_y \forall_x P_{xy}$	Somethings area liked by everything	YES
$\exists_x \exists_y P_{xy}$	There is atleast one thing that likes atleast one thing	YES
$\exists_y \exists_x P_{xy}$	There is atleast one thing that likes atleast one thing	YES

3.2 Identity

The notion of identity relates to one thing being equivilatnt to another thing. For instance: the morning star,a, is the evening star, b:

$$a = b$$

The identity relation means the exact same thing no matter what the domain is and what the extension is. Because the identity is a relation it can take in names and have variable places like any other relation. The syntax has the exact same rules as the other areas weve focused on, and it is identical in use to any predicate. This is to say that the identity predicate can be negated:

$$\neg(x = y)$$

Which is the same as:

$$x \neq y$$

3.3 Semantics of GPL

The key to semantics of MPL wa sthe notion of an extension - which is a set of objects that possess a particular property, which we mark with a predicate letter. The semantics for GPL use the same general idea - the central difference, however, is that we must generalise the notion of an extension to handle n-place predicates.
GPL relations As with one place predicates, n-place predicates have extensions, however they are not just sets of objects. The extension of a two place predicate is a set of ordered pairs. An ordered pair is just a list with a kind of ordereing built into it. For the example of "Daniel is taller than Tina", which can be generalised to "x is taller than y" and can have the following properties:

$$Domain: \{Daniel, Tina\}$$

$$d = Daniel,\ t = Tina$$

$$Txy : \{x, y : x\ is\ taller\ than\ y\}$$

which means that T_{dt} is true, as Daniel is taller than Tina, but T_{td} is not true as Tina is not taller than Daniel. This can be represented in the ordered pair: $< Daniel,\ Tina >$. We need the set of ordered lists to be a set such that the claims are true. This means that the extension of a two-place predicate is a set of ordered pairs, where the pairs make the predicate true. This can be generalised for n-place predicates, simply by adding more items per ordered group (ie. an ordered triplet, etc.).

3.4 Models

A model is:

- A specification of a domain of objects
- A specification of the extension of any predicates
- The assignment of an object to each name, where the same object can be named twice, but the same name cannot name two objects

3.4.1 Validity in terms of models

Even though we are now thinking in terms of models, the understanding of validity of a GPL argument remains the same.

3.5 Semantics of Identity

The identity predicate is defined recusively. The identity predicate is just a two place relation, so in a basic sense the semantics of GPL carry over to the identity symbol as well. The identity predicate is a set of n-tuples, but it is special because it is interpreted in the same manner in every model. The identity symbol is given the following interpretation under very model: $=$: the set of ordered pairs containing one pair $< x, x >$ for each object x in the domain. There are three important formal properties of the identity:

1. **Reflexivity:** Every entity is identical to itself:

$$\forall_x(x = x)$$

2. **Transitivity:** If x is identical to y, and y is identical to z then x is identical to z:

$$\forall_x\forall_y\forall_z((x = y \wedge y = z) \rightarrow x = z)$$

3. **Symmetry:** if x is identical to y then y is identical to x

$$\forall_x\forall_y(x = y \rightarrow y = x)$$

3.6 GPL Trees

No new rules need to be applied for GPL predicates, except for identity. There are 3 new rules we must add:

1. The following case will lead to closure:

$$a \neq a$$

2. If at any point we have an equality between two terms, we are entitled to substitute one of those terms for the other in any formula on that branch in the tree:

$$Rab$$

$$a = b$$

means we now have:

$$Raa$$

$$Rbb$$

This also Means that we have substitution of identicals:

$$a = b(or\ b = a)$$

$$\alpha(b//a)$$

Where the double slashes denote bi directional substitution (b for a or a for b)

3. Negation of equalities - When we negate an equality we are basically saying that a and b are not the same thing. When we negate an inequality we are basically saying that a and b are the same thing.

3.6.1 Saturation

For GPL+I we need to take into account identity claims. In order to saturate a GPL+I tree we need to work through all of the equalities in the tree. That means you need to substitute all of the equivilances (ie. $a = b$) for all relevant spots in the branch - Thas is:

If

$$a = b$$

then we must change all examples of a to b and then all examples of b to a in order for the tree to be saturated. For example:

$$R_{ab}$$

$$a = b$$

$$R_{ab}$$

$$R_{aa}$$

$$R_{ba}$$

$$R_{bb}$$

This now gives us an updated set of saturation rules - a branch is saturated when:

1. Every formula has had the relevant rules applied to it
2. Every formula on it whos main operator is a universal quantifier
    - Has had the universal quantifier applied to it atleast once and,
    - Has had the rule applied to it once for each name that appears on the branch
3. Every application of sinstotiton of identicals that could be made on that branch and that would result in the addition to the path of a formula that does not already appear on the branch has been made.

3.6.2 Validity in GPL

Validity in GPL is fairly straight forward, but what is invalidity? We handle these the exact same way as we did in MPL arguments. A GPL argument is invalid if a tree is completed and atleast one branch remains open - we then use the open branch on the tree to read off a countermodel.

3.7 Steps to solve GPL trees

0. Negate the conclusion!
1. Start with any non branching propositions (which dont have a quantifier) and then move to branching propositions
2. Flip all negated quantifiers
3. Start with any existentials and give them names for their variables ASAP
4. Now start on universals and saturate each (with names from higher in the tree)
    - If you notice that the existential keeps being called by an internal Existential end the branch - this is an open end
5. Apply any identities, ensuring to check every combination for if it might saturate close the branch
6. Keep repeating until the tree is completely saturated
7. Either the tree is valid or read off a countermodel