# Introduction to Logic

Jake Chandler

*Tableaux for Predicate Logic*

LA TROBE UNIVERSITY

## Extending the Tableau Method

## Last week

- The idea that some arguments are valid because of their subsentential (rather than merely sentential) form
- A notational system with which to express subsentential structure: the language of predicate logic
- Some potential issues in translating from English to the language of predicate logic
- The concept of a model in predicate logic



## Tableaux for $\mathscr{L}_P$

- Similar principle to tableaux for $\mathscr{L}_S$:

  Try to find a model $M$ such that $v_M$ assigns the value 1 to the premises and negation of the conclusion
- Same
  (a) way of starting the tableau,
  (b) closing rules
  (c) resolving rules for the connectives
- New extra rules for universal and existential sentences, both negated and non-negated: instantiation rules

## Reminder: Instances

- Recall:

  For any sentence $\varphi$ of the form $(\forall x)A$ or $(\exists x)A$, we denote by $A(x := a)$ the result of

  (i) substituting $a$ for every (free) occurrence of $x$ in $A$

  (ii) subsequently deleting the leftmost quantifier $(\forall x)$ or $(\exists x)$.

  We say that $A(x := a)$ is an instance of $\varphi$.

**Instances**

$Fc\&Gb$ is an instance of $(\forall x)(Fx\&Gb)$

$(\forall y)(Lby)$ is an instance of $(\exists x)(\forall y)(Lxy)$



## Existential sentences

- Rule:

$$\begin{array}{c}(\exists x)A\\ |\\ A(x := a)\\ (\text{new } a)\end{array}$$

**Existential sentences**

$$\begin{array}{c}(\exists x)(Gb \supset Lbx)\\ |\\ Gb \supset Lba\end{array}$$

- Rationale:
  - If $(\exists x)A$ is true, then one of its instances is true.
  - This might not be an instance that involves a name already used on the branch. So we introduce a new name, just in case.



## Universal sentences

- Rule:

$$\begin{array}{c}(\forall x)A\\ |\\ A(x := a)\\ (a \text{ already on branch})\end{array}$$

**Universal sentences**

$$\begin{array}{c}(\forall x)(Fax\&(\exists y)(Rxy))\\ |\\ Faa\&(\exists y)(Ray)\end{array}$$

- Rationale:
  - If $(\forall x)A$ is true, then all of its instances are true.
  - This includes all those instances involving names already used on the branch.



## Negated quantified sentences

- Rules:

$$\begin{array}{c}\sim(\exists x)A\\ |\\ (\forall x)\sim A\end{array} \qquad \begin{array}{c}\sim(\forall x)A\\ |\\ (\exists x)\sim A\end{array}$$

- Rationale: intuitive logical equivalence

## Negated quantified sentences: alternative

- In the textbook, Restall uses:

$$\begin{array}{c} \sim(\exists x)A \\ | \\ \sim A(x := a) \\ \text{(any } a \text{ already on branch)} \end{array} \qquad \begin{array}{c} \sim(\forall x)A \\ | \\ \sim A(x := a) \\ (a \text{ new}) \end{array}$$

**Negated quantified sentences**

$$\begin{array}{c} \sim(\exists x)(Gb \supset Lbx) \\ | \\ \sim(Gb \supset Lbb) \end{array} \qquad \begin{array}{c} \sim(\forall x)(Fax \& (\exists y)(Rxy)) \\ | \\ \sim(Fab \& (\exists y)(Rby)) \end{array}$$

- These are clearly equivalent (can you see why?)



## Particular vs general rules

- The rules for $(\exists x)A$ and $\sim(\forall x)A$ are known as a particular rules
- The rules for $(\forall x)A$ and $\sim(\exists x)A$ are known as general rules
- The rule for $(\exists x)A$ is applied only once to a given sentence.

  Once used, we put a tick next to the sentence alongside the name introduced: e.g. $\checkmark a$
- The rule for $(\forall x)A$ can be applied repeatedly to one same sentence

  After first use, we put a backslash next to the sentence, alongside the relevant name: e.g. $\backslash a$.

  Upon subsequent uses, we add the relevant names: e.g. $\backslash a, b, c \ldots$



## Note on general rules

- Question:

  What happens when all the sentences at the root of the tableau are of the form $\sim(\exists x)A$ or $(\forall x)A$ and do not contain any names?
- Answer:
  - Since every domain has at least one element and every element is named, we have to have at least one name by default, say '$a$'.
  - We then substitute this into all the relevant sentences, using the relevant rules.



## Note on general rules (ctd.)

**Handling roots without names**

$$\begin{array}{c} (\forall x)Gx \ \backslash a \\ \sim(\exists x)Gx \ \checkmark \\ | \\ (\forall x) \sim Gx \ \backslash a \\ | \\ Ga \\ | \\ \sim Ga \\ \times \end{array}$$

# Completedness

- Note: there are no repeatable rules in sentential logic.
- The introduction of these rules makes a difference to the definition of a completed tableau in pred. logic.
- In sent. logic:

  A tableau is completed iff (i), in every open branch $b$, every sentence on $b$ that could have had a rule applied to it has had a rule applied

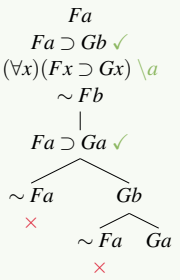
- Not good enough here: sentences of the form $(\forall x)A$ sometimes need to have rules applied to them more than once.
- In pred. logic:

  A tableau is completed iff, in every open branch $b$, (i) (as above) *and* (ii) every name on $b$ (and at least one name) has been substituted into every sentence of the form $(\forall x)A$

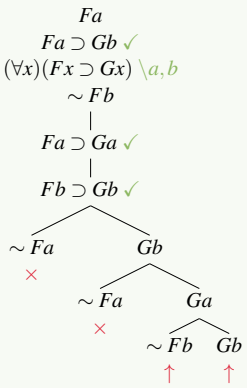


# Completedness (ctd.)

**Completed vs non-completed tableaux**




Non-completed! ($b$ not substituted in)




Completed ($b$ substituted in)



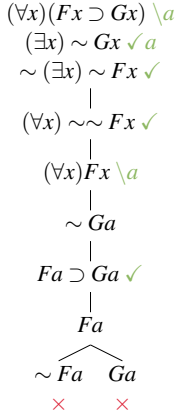
# Brief general tips

- Recommendations:
  - Apply rules for connectives first, with, among these, non-branching rules first
  - Apply instantiation rules second, with, among these, particular rules first
- These recommendations are defeasible, however: e.g. the application of a general rule may immediately lead to tableau closure
- Now for 2 examples:
  (1) a tableau that closes,
  (2) an open tableau with countermodel



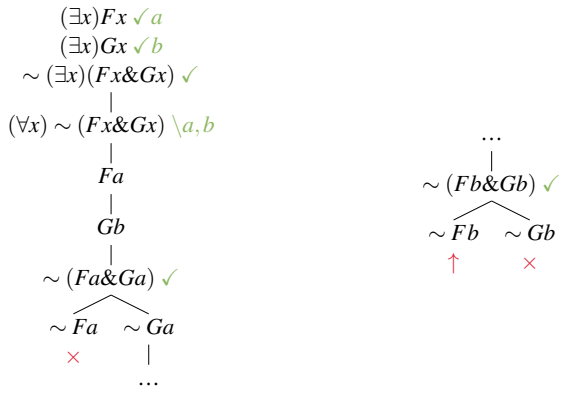
# A closed tableau

- We show that $(\forall x)(Fx \supset Gx), (\exists x) \sim Gx \vdash (\exists x) \sim Fx$



## A completed open tableau

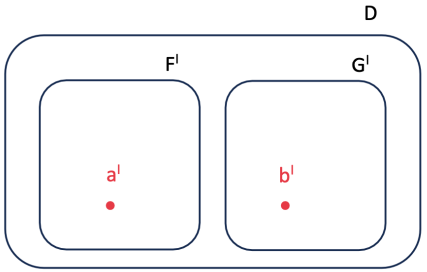
- We show that $(\exists x)Fx, (\exists x)Gx \nvdash (\exists x)(Fx \& Gx)$






## A completed open tableau: the countermodel

- Our open branch contains: $Fa$, $Gb$, $\sim Ga$ and $\sim Fb$.
- Countermodel:






# Introduction to Logic

Jake Chandler

*Tableaux for Predicate Logic*

## Soundness, Completeness and Decidability

## A curious tableau…

- We check whether or not $\vdash \sim (\forall x)(\exists y)Lxy$

$$
\begin{array}{l}
(\forall x)(\exists y)Lxy \;\backslash a_1, a_2, a_3, \ldots \\
\quad | \\
(\exists y)La_1y \;\checkmark a_2 \\
\quad | \\
La_1a_2 \\
\quad | \\
(\exists y)La_2y \;\checkmark a_3 \\
\quad | \\
La_2a_3 \\
\quad | \\
(\exists y)La_3y\ldots
\end{array}
$$



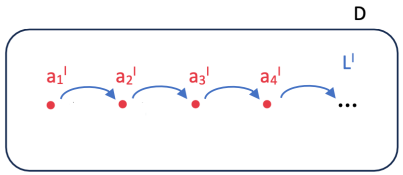
## An infinite open tableau (!)

- What's going on here?
- There exists a completed tableau and it is open: there *exists* a proof of invalidity
- Problem: the completed open tableau is infinitely long!
- We cannot construct the proof in a finite number of steps, by sequentially applying rules until we complete the tableau: the tableau method cannot decide validity here
- This is *not* the case for propositional logic, nor is it the case for the restriction of predicate logic to unary predicates
- In both those cases, completed tableaux are always finite



## An infinite open tableau (!) (ctd.)

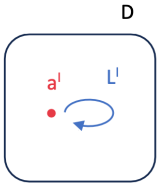
- Our non-terminating branch will contain $La_1a_2, La_2a_3, La_3a_4, \ldots$
- This gives us an infinite countermodel






## An infinite open tableau (!) (ctd.)

- As Restall notes, in this particular case, we can also use our initiative to find a *finite* countermodel:




- This is *not* always possible however.
- For instance, one can show that there is *no* finite model in which the following is true:

$$(\forall x)(\exists y)Rxy\&(\forall x) \sim Rxx\&(\forall x)(\forall y)(\forall z)((Rxy\&Ryz) \supset Rxz)$$

(Can you see why?)

## Decidability

- The tableau method is sound and complete (see Restall)
- But we saw that our tableau method can't decide the satisfiability of sentences in $\mathscr{L}_P$ (the "FO-SAT problem")
- What about *other* procedures?
- We've seen that Turing's Halting Problem is undecidable
- It turns out that we can reformulate the Halting Problem as a specific predicate logic satisfiability problem
- Since the Halting Problem is undecidable, so too is that satisfiability problem
- Hence FO-SAT is undecidable



# Introduction to Logic

Jake Chandler

*Tableaux for Predicate Logic*



## Some noteworthy (in)validities

## Equivalences

- Quantifier order:

  $(\forall x)(\forall y)A \Leftrightarrow (\forall y)(\forall x)A$

  $(\exists x)(\exists y)A \Leftrightarrow (\exists y)(\exists x)A$

  (But $(\forall y)(\exists x)A \not\Leftrightarrow (\exists x)(\forall y)A$

    $(\forall y)(\exists x)Lxy$: Everyone is loved by someone or other.

    $(\exists x)(\forall y)Lxy$: There is someone who loves everyone.)

- Quantifier duality:

  $(\forall x)A \Leftrightarrow\sim (\exists x) \sim A$

  $(\exists x)A \Leftrightarrow\sim (\forall x) \sim A$

## Failure of existential import

- Some valid forms, according to Aristotle:

  $(\forall x)(Dx \supset Rx) \therefore (\exists x)(Dx \& Rx)$

  $\sim (\exists x)(Gx \& Tx) \therefore (\exists x)(Gx \& \sim Tx)$

**Existential import**

All flying donkeys come home to roost. Therefore some flying donkeys come home to roost.

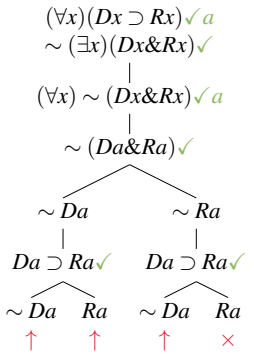
No goblin drives a pickup truck. Therefore some goblins don't drive pickup trucks.

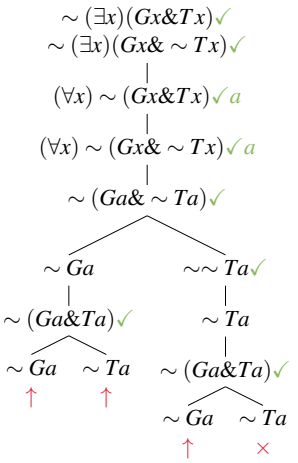
- According to him, $(\forall x)$ and $\sim (\exists x)$ have existential import



## Failure of existential import (ctd)

- These argument forms are not valid in predicate logic!!

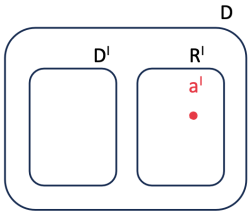


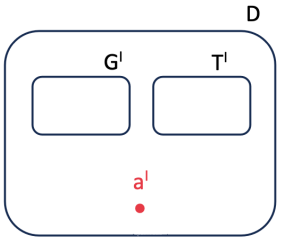






## Failure of existential import (ctd)

- Countermodels




- Note that we have $\sim (\exists x)D$ and $\sim (\exists x)Gx$, respectively



## Trivially true universal generalisations

- Not only do we have

  $(\forall x)(Dx \supset Rx) \not\models (\exists x)(Dx \& Rx)$

  but we actually have:

  $\sim (\exists x)Fx \models (\forall x)(Fx \supset Gx)$

**Trivially true universal generalisations**

There are no flying donkeys. Therefore all flying donkeys come home to roost.

- Many people find this counterintuitive and it seems related to the following paradox of material implication: $\sim p \models p \supset q$

## Trivially true universal generalisations (ctd.)

- Proof:

$$
\begin{array}{c}
\sim(\exists x)Fx \checkmark \\
\sim(\forall x)(Fx \supset Gx) \checkmark \\
| \\
(\forall x) \sim Fx \checkmark a \\
| \\
(\exists x) \sim (Fx \supset Gx) \checkmark a \\
| \\
\sim (Fa \supset Ga) \checkmark \\
| \\
Fa \\
\sim Ga \\
| \\
\sim Fa \\
\times
\end{array}
$$



# Introduction to Logic

Jake Chandler

*Tableaux for Predicate Logic*



## Identity

## Identity statements

*Superman*

Superman loves Lois. Superman is Clark Kent. Therefore Clark Kent loves Lois.

- This seems valid
- In standard predicate logic: $Lsl, Isc \therefore Lcl$
- Problem: $Lsl, Isc \not\models Lcl$
- Suggestion: We have the hidden premise $Lsl \supset Lcl$
- Implausible: $Lsl \supset Lcl$ arguably merely *follows* from $Isc$

## New quantifiers

*Problems*

John has three problems. Therefore John has at least two problems.

- Also seems valid. What's the logical form?
- Suggestion: $(\exists x)(Px \& Hjx) \& (\exists y)(Py \& Hjy) \& (\exists z)(Pz \& Hjz) \therefore (\exists x)(Px \& Hjx) \& (\exists y)(Py \& Hjy) \vee \ldots$
- Problem #1: The conclusion is an infinite disjunction
- Problem #2: This is simply equivalent to $(\exists x)(Px \& Hjx) \therefore (\exists x)(Px \& Hjx)$!
- Suggestion: use two unary predicates: '…has three problems' ($F$) and '…has at least two problems' ($G$).
- Problem: $Fj \not\models Gj$



## Expanding the language

- We are going to add a special symbol '=' to $\mathscr{L}_P$ to yield $\mathscr{L}_{PI}$ (for 'predicate logic with identity')
- New syntactic rule:

  If $a$ and $b$ are names, then $a = b$ is a wfs.

- New rule for evaluating truth in a model

  Where $a$ and $b$ are names, $v_M(a = b) = 1$ iff $a^I = b^I$.

- We write '$a \neq b$' as shorthand for $\sim a = b$



## Amending the tableau system

- New rule for closure:

  For any name $a$, if $a \neq a$ is on a branch $b$, then $b$ closes.

- New tableau rule, where $\varphi$ is a wfs, $a$ and $b$ are names and $\varphi(a := b)$ denotes the result of substituting $b$ for the instances of $a$ in $\varphi$:

$$\begin{array}{c} \varphi \\ a = b \\ | \\ \varphi(a := b) \end{array}$$

- We can now translate *Superman*: $Lsl, s = c \therefore Lcl$. Proof of validity left as a (trivial) exercise.



## A tableau that closes

- We show that $\vdash (\forall x)(\forall y)(x = y \supset y = x)$:

$$\begin{array}{c} \sim (\forall x)(\forall y)(x = y \supset y = x) \checkmark \\ | \\ (\exists x) \sim (\forall y)(x = y \supset y = x) \checkmark a \\ | \\ \sim (\forall y)(a = y \supset y = a) \checkmark \\ | \\ (\exists y) \sim (a = y \supset y = a) \checkmark b \\ | \\ \sim (a = b \supset b = a) \checkmark \\ | \\ a = b \\ b \neq a \checkmark \\ | \\ b \neq b \\ \times \end{array}$$

## A completed open tableau

- We show that $\nvdash (\forall x)(\forall y)(\forall z)(((Rxy \& Rzx) \& y = z) \supset Rxx)$

$\sim (\forall x)(\forall y)(\forall z)(((Rxy \& Rzx) \& y = z) \supset Rxx)$ ✓

$(\exists x) \sim (\forall y)(\forall z)(((Rxy \& Rzx) \& y = z) \supset Rxx)$ ✓ $a$

$\sim (\forall y)(\forall z)(((Ray \& Rza) \& y = z) \supset Raa)$ ✓

$(\exists y) \sim (\forall z)(((Ray \& Rza) \& y = z) \supset Raa)$ ✓ $b$

$\sim (\forall z)(((Rab \& Rza) \& b = z) \supset Raa)$ ✓

$(\exists z) \sim (((Rab \& Rza) \& b = z) \supset Raa)$ ✓ $c$

$\sim (((Rab \& Rca) \& b = c) \supset Raa)$ ✓

$((Rab \& Rca) \& b = c)$ ✓

$\sim Raa$

…

…

$Rab$ ✓

$Rca$

$b = c$

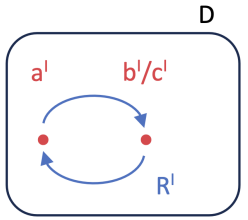
$Rac$

↑



## A completed open tableau

- What about the countermodel?
- Same as before *except* that identity sentences on the relevant open branch constrain the interpretation of names.
- Here, we must have $b^I = c^I$.
- Our countermodel:






## Handling our new quantifiers

- Recall:

*Problems*

John has three problems. Therefore John has at least two problems.

- Regarding the conclusion:

$(\exists x)(\exists y)(Pxj \& Pyj \& x \neq y)$

'There exists an $x$, such that there exists a $y$, such that $x$ and $y$ are both problems of John's and are non-identical.'



## Handling our new quantifiers (ctd.)

- More generally, we translate 'There are at least $n$ $F$'s' by

$(\exists x_1) \ldots (\exists x_n)(Fx_1 \& \ldots \& Fx_n \& x_1 \neq x_2 \& \ldots \& x_1 \neq x_n \& \ldots \& x_2 \neq x_3 \& \ldots \& x_2 \neq x_n \& \ldots \& x_{n-1} \neq x_n)$

'There exists an $x_1$, such that there exists an $x_2$, …, such that there exists an $x_n$, such that $x_1$ to $x_n$ are all $F$'s and are non-identical.'

- We can also translate 'There are at most $n$ $F$'s':

$(\forall x_1) \ldots (\forall x_{n+1})((Fx_1 \& \ldots \& Fx_{n+1}) \supset (x_1 = x_2 \vee \ldots \vee x_1 = x_n \vee \ldots \vee x_2 = x_3 \vee \ldots \vee x_2 = x_n \vee \ldots \vee x_n = x_{n+1}))$

'For all $x_1$ to $x_{n+1}$, if $x_1$ to $x_{n+1}$ are all $F$'s, then at least two of them are identical.'

# Handling our new quantifiers (ctd.)

- With this we can now translate 'There are (exactly) $n$ $F$'s' via its equivalence with 'There are at least and at most $n$ $F$'s'.
- So, for 'John has exactly two problems', we'd get:

  $$(\exists x)(\exists y)(Pxj\&Pyj\&x \neq y)\&(\forall x)(\forall y)(\forall z)((Pxj\&Pyj\&Pzj) \supset (x = y \vee x = z \vee y = z))$$

  'There exists an $x$, such that there exists a $y$, such that $x$ and $y$ are both problems of John's and are non-identical *and* such that for all $z$, if $z$ is a problem of John's then it is identical to either $x$ or $y$.'
- Our premise in *Problems* ("John has three problems") is handled similarly