NASSLLI 2016—Multi-Modal Logic

Reinhard Muskens

Tilburg Center for Logic, Ethics, and Philosophy of Science (TiLPS)

Part I: Tableaux for Predicate Logic

Welcome to the Multi-Modal Logic Course!

- This course will give you a quick introduction to multi-modal logic as a tool in linguistics, philosophy, and artificial intelligence.
- We will focus on techniques that will let you define useable logics and will de-emphasise techniques for proving things about these logics.
- Modal logics will be introduced as fragments of predicate logic. This departs from standard didactic procedures but has some advantages given our goals. If you are already familiar with modal logics you may still want to see how this is done.
- Another focus will be on tableau systems. Tableaux are handy tools for checking entailment and consistency and can easily be implemented.

Prerequisites

- I will assume familiarity with predicate logic, but not with much beyond that. The pace will be fast, though.
- You are advised to take a course on logic or to work through a good introduction to predicate logic, should that knowledge be lacking. Three of my favourite introductory texts are Hodges (2001), Gamut (1991), and Smith (2003), but there are many others.

Modal Logics as Fragments of Predicate Logic

My goal here is to develop modal logics for the working linguist, philosopher, or computer scientist, not to study them from a logician's point of view. Given this goal, introducing them as fragments of predicate logic has certain advantages:

- The approach leads to fewer technicalities, so we can focus on ideas instead.
- In applications modal logics typically need to be combined. This is usually difficult, but not if the logics are all introduced as fragments of classical logic in the first place.
- An upgrade to the simply typed λ-calculus will be easy and can be used for the logical analysis of language.

For purposes like showing sets of proof rules complete wrt a class of models or showing them to be decidable one should move to a more standard set-up.

Parts of the Course I

The course will consist of five parts.

- Part 1 will rehearse the theory of tableau systems for predicate logic. We will also consider the idea of axioms in this part.
- In Part 2 we will introduce basic temporal logic.
- Part 3 looks at basic modal logics.
- In Part 4 we will show that at least one modal theory that is more sophisticated than one of the basic ones can be treated using our method. We will have a look at conditional modal logics in the tradition of David Lewis and others and apply them to counterfactuals and conditional obligation. There is also a connection with belief revision.

Parts of the Course II

- Part 5 will consider extensions and combinations of the logics treated thus far. In particular, we will combine temporal and modal logics, introduce quantification, and perhaps have a look at Kaplanian contexts.
- There will be five parts and five days, but since some parts contain more material than others we will not necessarily do one part per day. There may also simply be too much material to cover in five slots. We'll see.
- I hope to upload the slides to my new `freevariable.nl` site.

Tableaux for Predicate Logic

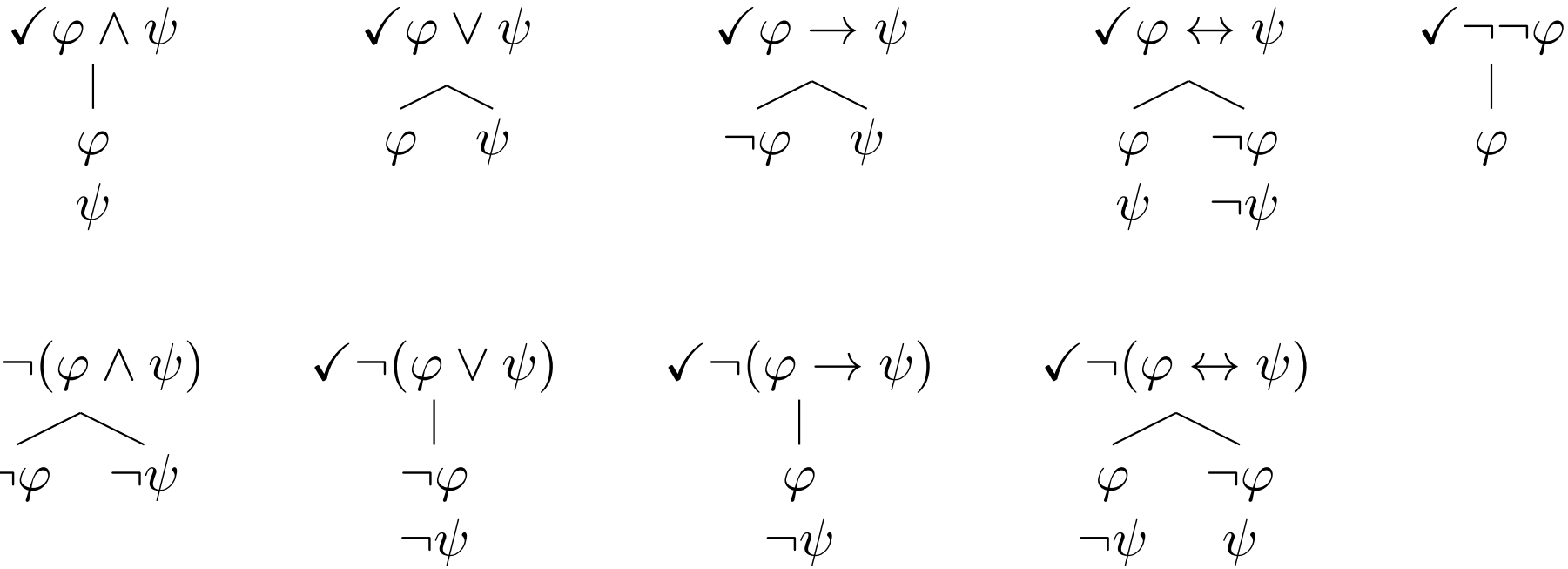
- Entailment (and consistency) are pivotal notions in any logic.
- There are various techniques to prove entailment. Here we focus on tableau systems (Beth 1955).
- We will focus on predicate logic first, in order to obtain tableaux for modal logics later.
- Tableaux can be interpreted as systematic searches for models.
- But they need not be so interpreted. They are defined by purely formal rules.

An Example of a Tableau Rule

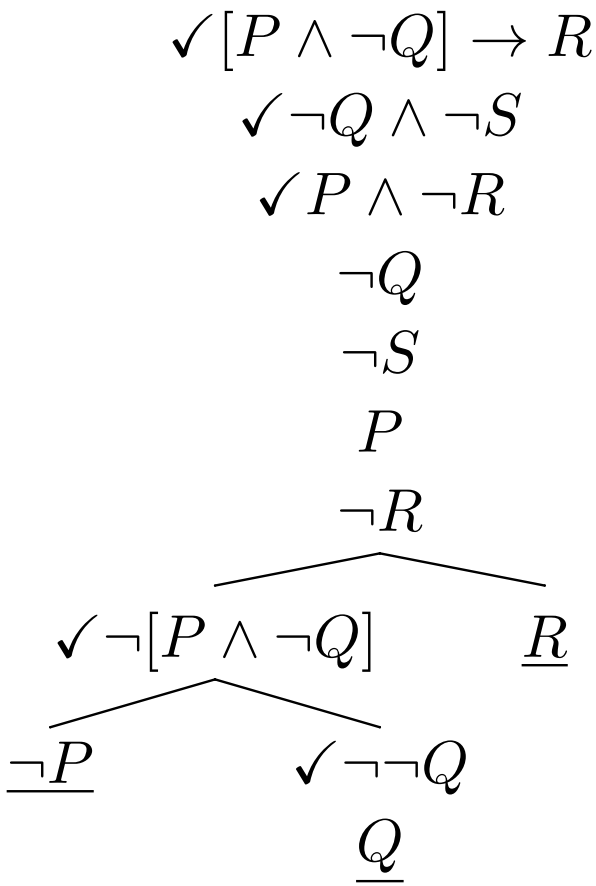
- As an example of a tableau rule, here is one for $\neg(\varphi \wedge \psi)$:
- $\neg(\varphi \wedge \psi)$

  $\neg\varphi \quad \neg\psi$
- If $\neg(\varphi \wedge \psi)$ is true at least one of $\neg\varphi$ and $\neg\psi$ must be true.
- (And in fact if $\neg\varphi$ or $\neg\psi$ is true, $\neg(\varphi \wedge \psi)$ is.)
- The following slide gives an overview of all propositional tableau rules.
- (A ✓ signals that the formula can be discarded once the rule has been applied.)

Tableau Rules for Propositional Logic

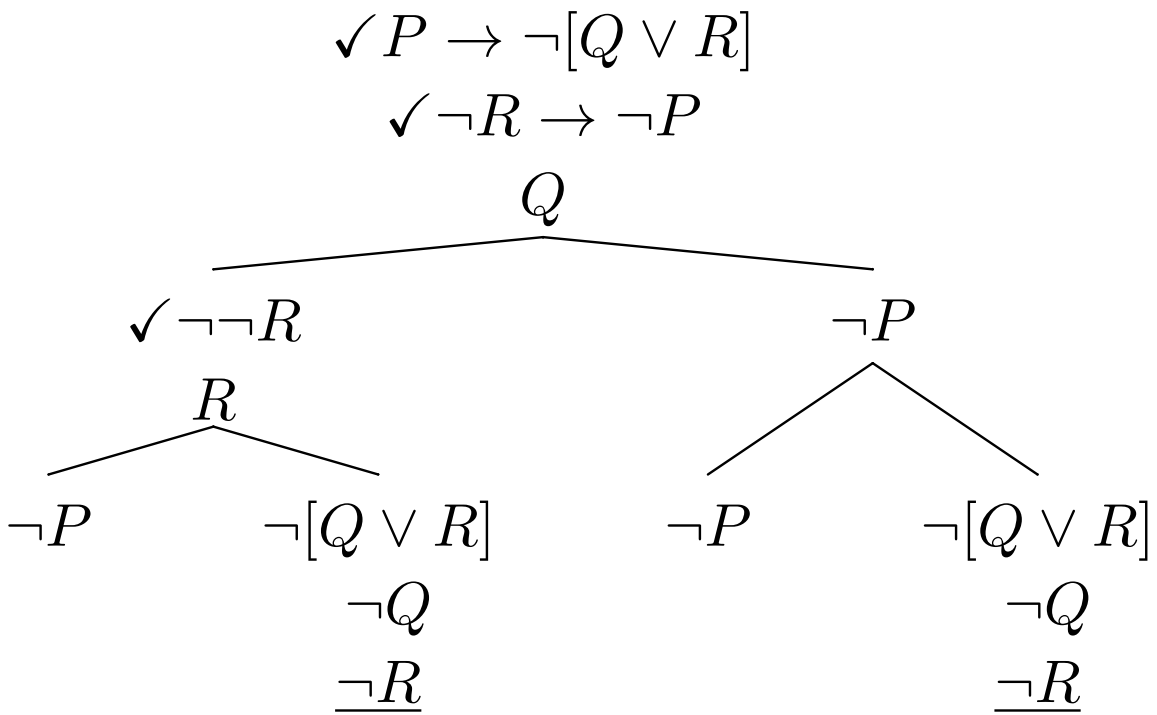


Testing for Consistency 1



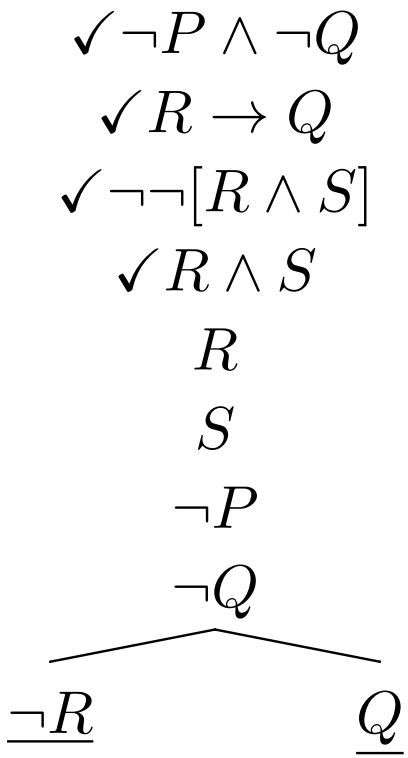
- A semantic tableau for $\{[P \wedge \neg Q] \rightarrow R,\ \neg Q \wedge \neg S,\ P \wedge \neg R\}$.
- A semantic tableau is a systematic search for a model satisfying the given sentences.
- The tableau closes. So there is no structure in which the given sentences are simultaneously true.
- The set of sentences is shown to be inconsistent.

Testing for Consistency 2



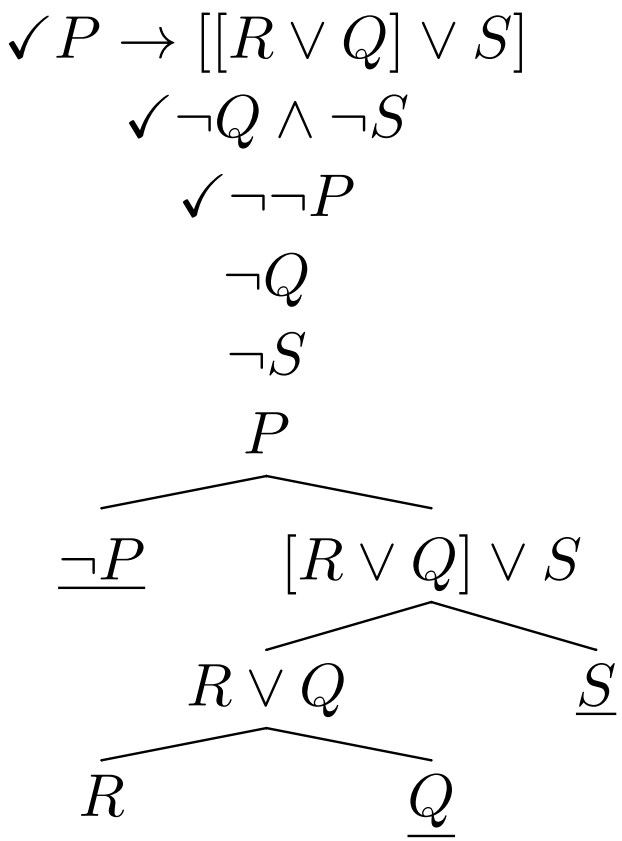
- The tableau does not close. $\{P \to \neg[Q \vee R],\ \neg R \to \neg P,\ Q\}$ is a consistent set of sentences. A model satisfying $\neg P$ and Q will do the trick. This can be read off the second open branch. In fact every open branch will give at least one model.

Testing for Validity 1



- A tableau for the argument $\neg P \wedge \neg Q,\ R \rightarrow Q \vdash \neg[R \wedge S]$ is made
- by taking the counterexample set $\{\neg P \wedge \neg Q,\ R \rightarrow Q,\ \neg\neg[R \wedge S]\}$ and making a tableau for that set.
- The tableau closes. There is no counterexample.
- The argument is thus shown to be valid.

Testing for Validity 2



- A tableau for $P \to [[R \vee Q] \vee S],\ \neg Q \wedge \neg S \vdash \neg P$ is obtained
- by making one for the counterexample set $\{P \to [[R \vee Q] \vee S],\ \neg Q \wedge \neg S,\ \neg\neg P\}$.
- The tableau does not close. An open branch yields the counterexample R, P, $\neg Q$, $\neg S$.

Tableau Rules for $\forall$ and $\exists$

- We may write $\varphi(x)$ for φ in order to draw attention to the fact that x may be free in φ. We then write $\varphi(c)$ for the result of replacing each free occurrence of x in φ by c.
- In the first rule below 'c old/first' is short for: c either already has an earlier occurrence on the branch where $\forall x\varphi(x)$ occurs, or there are no occurrences of constants on this branch before the point the rule is applied and c is freshly introduced.

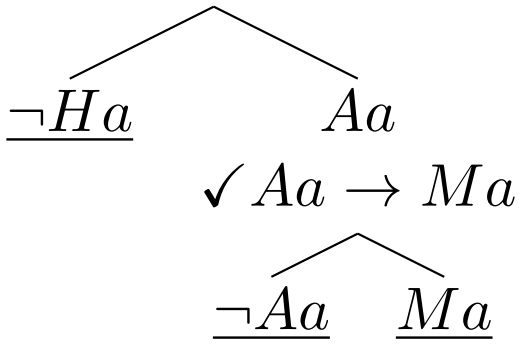
$$\begin{array}{c} \forall x\varphi(x) \\ | \\ \varphi(c) \\ (c \text{ old/first}) \end{array} \qquad \begin{array}{c} \checkmark\exists x\varphi(x) \\ | \\ \varphi(c) \\ (c \text{ new}) \end{array} \qquad \begin{array}{c} \checkmark\neg\forall x\varphi \\ | \\ \exists x\neg\varphi \end{array} \qquad \begin{array}{c} \checkmark\neg\exists x\varphi \\ | \\ \forall x\neg\varphi \end{array}$$

$\forall$ Sentences: A Standing Obligation

- Note that in the rule for $\forall x\varphi$ no $\checkmark$ was placed in front of the universal sentence.
- Until now all rules had the property that they had to be applied to a relevant sentence only once. That sentence then became irrelevant to the rest of the tableau.
- The rule for $\exists x\varphi$ also has this property. Once $\varphi(c)$ is introduced (new c), the information that $\exists x\varphi$ is not lost.
- But if $\forall x\varphi$ occurs on a branch the conclusion that $\varphi(c)$ does not exhaust the original information. If some b also occurs on the branch (or comes to occur on it later), we can conclude that $\varphi(b)$, and that info need not be contained in $\varphi(c)$.
- A sentence $\forall x\varphi$ creates a standing obligation to continue applying the $[\forall]$ rule as long as there are constants on the branch.

$\forall x(Hx \to Ax),\ \forall x(Ax \to Mx) \vdash \forall x(Hx \to Mx)$

$$\forall x(Hx \to Ax)$$
$$\forall x(Ax \to Mx)$$
$$\checkmark \neg\forall x(Hx \to Mx)$$
$$\checkmark \exists x\neg(Hx \to Mx)$$
$$\checkmark \neg(Ha \to Ma)$$
$$Ha$$
$$\neg Ma$$
$$\checkmark Ha \to Aa$$



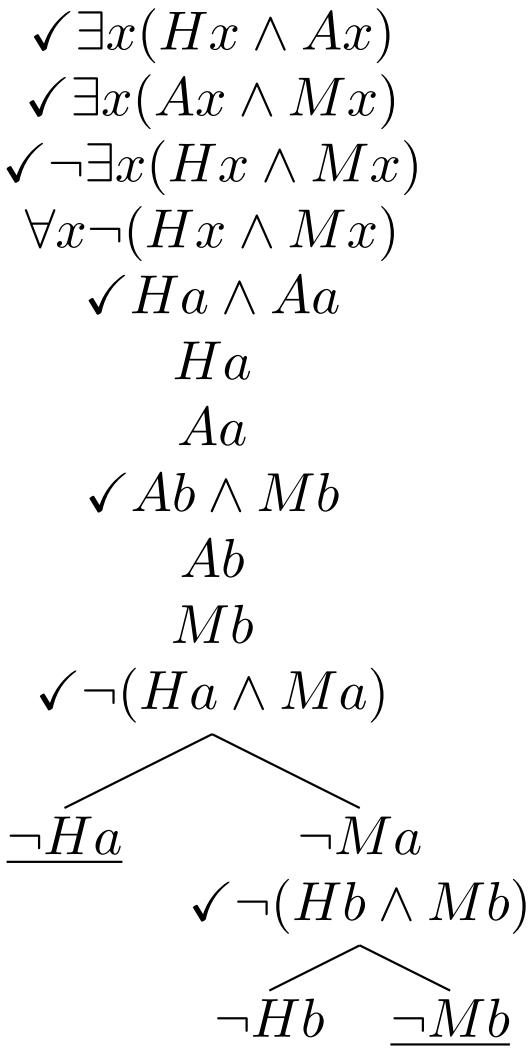
This is the Barbara syllogism:

All humans are animals
All animals are mortal
$\therefore$ All humans are mortal

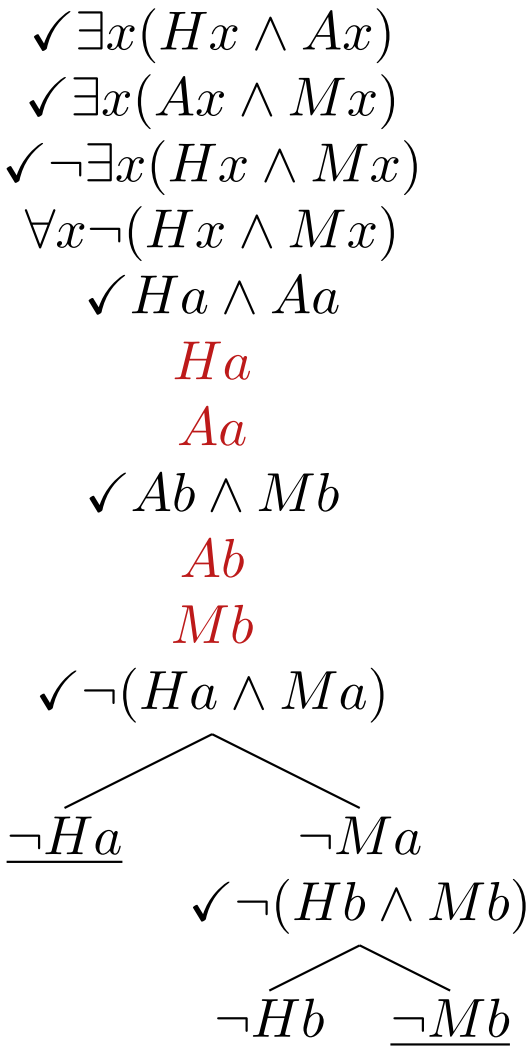
Reading Off Counterexamples from Open Branches

- The following slide gives a tableau for an invalid syllogism: Some human is an animal. Some animal is mortal. $\therefore$ Some human is mortal.
- An open branch is reached that is finished in the sense that no further rule can be applied.
- In such a case it is possible to read off a counterexample from the open branch by constructing a model that satisfies all atomic sentences on the branch, but does not satisfy any atomic sentence not on the branch.

$\exists x(Hx \wedge Ax),\ \exists x(Ax \wedge Mx) \not\vdash \exists x(Hx \wedge Mx)$

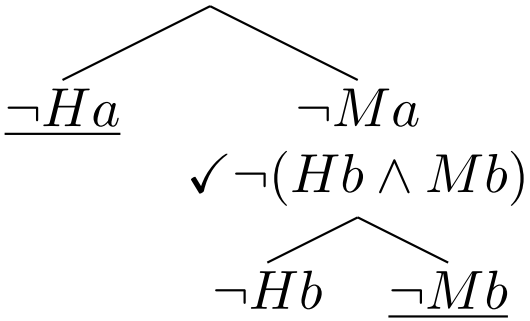


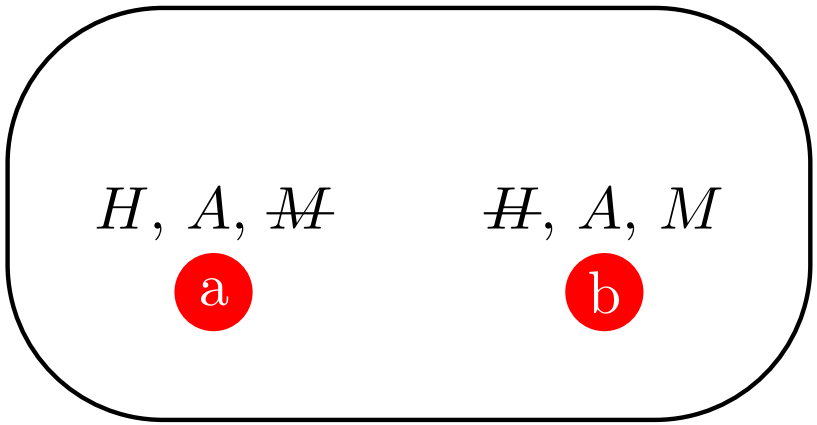
$\exists x(Hx \wedge Ax),\ \exists x(Ax \wedge Mx) \nvdash \exists x(Hx \wedge Mx)$



$\exists x(Hx \wedge Ax),\ \exists x(Ax \wedge Mx) \not\vdash \exists x(Hx \wedge Mx)$

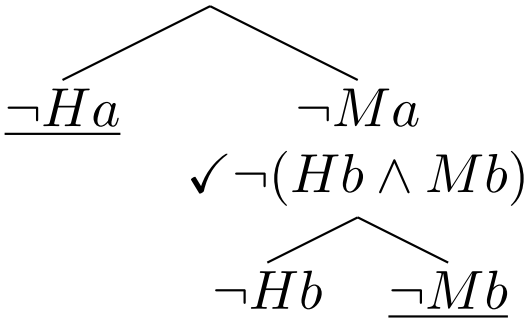
$\checkmark \exists x(Hx \wedge Ax)$
$\checkmark \exists x(Ax \wedge Mx)$
$\checkmark \neg\exists x(Hx \wedge Mx)$
$\forall x \neg(Hx \wedge Mx)$
$\checkmark Ha \wedge Aa$
Ha
Aa
$\checkmark Ab \wedge Mb$
Ab
Mb
$\checkmark \neg(Ha \wedge Ma)$

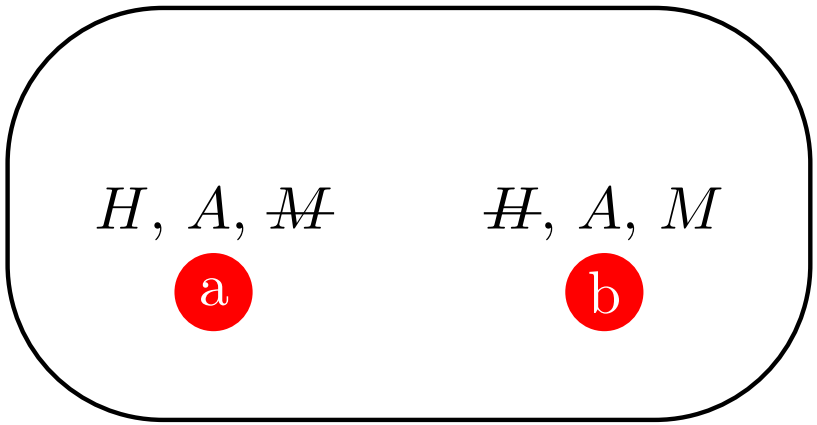




$\exists x(Hx \wedge Ax),\ \exists x(Ax \wedge Mx) \not\vdash \exists x(Hx \wedge Mx)$

$\checkmark \exists x(Hx \wedge Ax)$
$\checkmark \exists x(Ax \wedge Mx)$
$\checkmark \neg\exists x(Hx \wedge Mx)$
$\forall x \neg(Hx \wedge Mx)$
$\checkmark Ha \wedge Aa$
Ha
Aa
$\checkmark Ab \wedge Mb$
Ab
Mb
$\checkmark \neg(Ha \wedge Ma)$

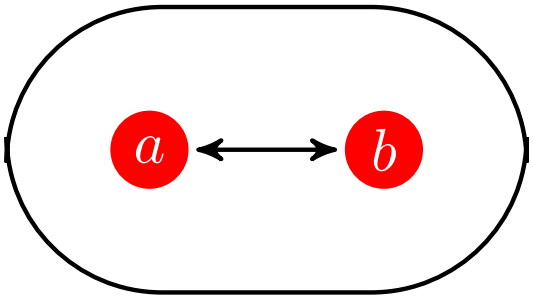




$\exists x \forall y Rxy \vdash \forall y \exists x Rxy$

- Here is an example of a valid quantifier shift.
- $$\begin{array}{c} \checkmark \exists x \forall y Rxy \\ \checkmark \neg \forall y \exists x Rxy \\ \forall y Ray \\ \checkmark \exists y \neg \exists x Rxy \\ \checkmark \neg \exists x Rxb \\ \forall x \neg Rxb \\ Rab \\ \underline{\neg Rab} \end{array}$$
- In general $\exists x \forall y Rxy$ is stronger than $\forall y \exists x Rxy$: $\exists x \forall y Rxy$ does not follow from $\forall y \exists x Rxy$ as we will see on the next slide.

$\forall y \exists x Rxy \not\vdash \exists x \forall y Rxy$

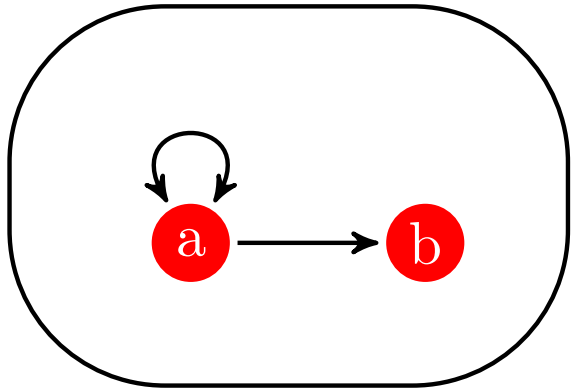
- The tableau rules formulated thus far do not always provide the easiest method to find counterexamples.
- Consider, for example, the question whether $\forall y \exists x Rxy \vdash^{?} \exists x \forall y Rxy$.
- A tableau gets rather involved, but some thought provides the following counterexample:
- 

- The rules for $=$ that will be discussed shortly will make it possible to use tableaus to generate finite counterexamples where they exist.

$\exists x \forall y Rxy \not\vdash \forall y \exists x Ryx$

Here is another tableau example. A few slides back we found that $\exists x \forall y Rxy \vdash \forall y \exists x Rxy$, but shifting the position of two variables leads to invalidity.

$$
\begin{array}{c}
\checkmark \exists x \forall y Rxy \\
\checkmark \neg \forall y \exists x Ryx \\
\forall y Ray \\
Raa \\
\checkmark \exists y \neg \exists x Ryx \\
\checkmark \neg \exists x Rbx \\
\forall x \neg Rbx \\
\neg Rba \\
\neg Rbb \\
Rab
\end{array}
$$

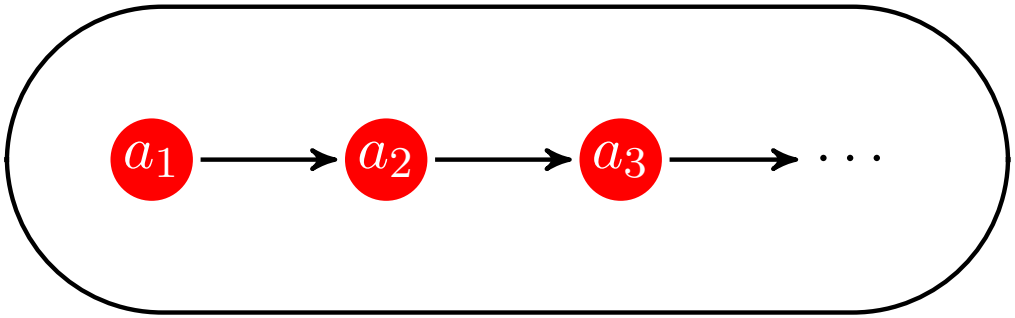


Why Does Rule [∃] Require a New Constant?

- The intuitive motivation of the [∃] rule is that if something satisfies φ we may as well give that thing a name.
- But it must be a new name. If an old name would be used, we run the risk of ascribing φ to something that can be shown not to satisfy φ, thus introducing an inconsistency.
- In the following we "prove" that $\forall x Ax$, follows from $\exists x Ax$ using an incorrect instantiation of [∃].
-
$$\begin{array}{c} \checkmark \exists x Ax \\ \checkmark \neg \forall x Ax \\ Aa \\ \checkmark \exists x \neg Ax \\ \underline{\neg Aa} \end{array}$$

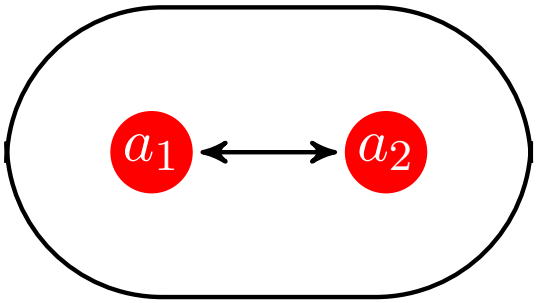
$\forall x \exists y Rxy \not\models \exists x Rxx$

$$
\begin{array}{c}
\forall x \exists y Rxy \\
\checkmark \neg \exists x Rxx \\
\forall x \neg Rxx \\
\neg Ra_1 a_1 \\
\checkmark \exists y Ra_1 y \\
Ra_1 a_2 \\
\neg Ra_2 a_2 \\
\checkmark \exists y Ra_2 y \\
Ra_2 a_3 \\
\neg Ra_3 a_3 \\
\checkmark \exists y Ra_3 y \\
\vdots
\end{array}
$$



An infinite tableau and an infinite counterexample.

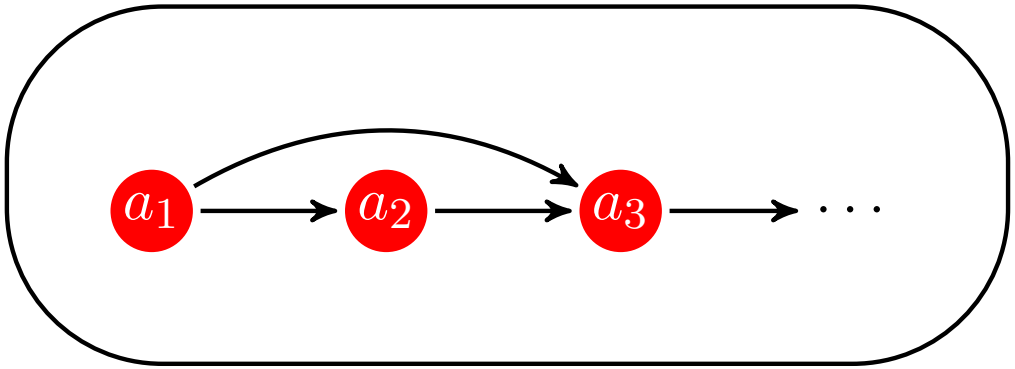
$\forall x \exists y Rxy \not\models \exists x Rxx$: A Finite Counterexample

- The previous slide made it clear that tableau computations can loop.
- For each constant a in the tableau, the [$\forall$] rule applied to $\forall x \exists y Rxy$ led to the creation of a new existential sentence $\exists y Ray$ and for each of these [$\exists$] spawned a new constant, so that [$\forall$] could fire again . . .
- In this case there is also a finite counterexample:
- 

- But the next slide illustrates that a tableau may have an infinite open branch, but no finite one.

$\forall x \forall y \forall z((Rxy \wedge Ryz) \rightarrow Rxz),\ \forall x \exists y Rxy \not\models \exists x Rxx$

$$\forall x \forall y \forall z((Rxy \wedge Ryz) \rightarrow Rxz)$$
$$\forall x \exists y Rxy$$
$$\checkmark \neg \exists x Rxx$$
$$\forall x \neg Rxx$$
$$\neg Ra_1a_1$$
$$\checkmark \exists y Ra_1y$$
$$Ra_1a_2$$
$$\neg Ra_2a_2$$
$$\checkmark \exists y Ra_2y$$
$$Ra_2a_3$$
$$\neg Ra_3a_3$$
$$\checkmark \exists y Ra_3y$$
$$\vdots$$



This time a finite counterexample no longer exists!

Predicate Logic is Undecidable

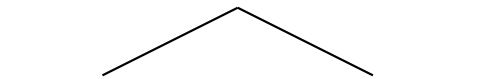
- The possibility that an infinite tree is obtained makes reasoning with predicate logic essentially more complex than reasoning with propositional logic.
- It can be shown that if a given argument is valid, there is a closed tableau for it. But that finite tableau could be very large.
- We have seen that an invalid argument does not have to correspond to a finite open tableau that is finished.
- Working on a tableau we may be unable to tell whether it is going to close at some point, whether it will be open but finished at some point, or whether the process will never stop either way.
- Computer programs have a similar property: maybe your program will return an answer in the next few minutes, maybe it will return one in the next few years, maybe it is in an infinite loop.

No Mechanistic Method: Use Your Wits

- The tableau method is a very useful tool to find out whether a sequent is correct or not, to prove a sequent's correctness if it is correct, and to provide counterexamples if it is not.
- But tableaus do not provide us with a mechanistic method for testing the validity of a predicate logical sequent. There are no such methods!
- In difficult cases you will have to steer the process. Think about the meaning of premises and conclusion and first try to find an informal argument that shows that the conclusion follows from the premises. Or try to produce a counterexample.
- If you find a counterexample, you no longer need a tableau because you have shown the sequent invalid. If you have found an informal argument, try to convert it into a closed tableau.

Rules for $=$

- If φ is on a branch and some a occurs in φ, then that a may be replaced by a b if $a = b$ or $b = a$ is on the same branch.
- If c_1 and c_2 are both on a branch then we may distinguish between the case that $c_1 = c_2$ and the case that $\neg c_1 = c_2$.
- $$\begin{array}{c} \varphi(a) \\ a = b \\ | \\ \varphi(b) \end{array} \qquad \begin{array}{c} \varphi(a) \\ b = a \\ | \\ \varphi(b) \end{array}$$



(c_1 and c_2 old)

- A branch can be closed if a sentence of the form $\neg\, c = c$ occurs on it.
- On the following slides [which I will skip] it is shown that $=$ is an equivalence relation, i.e. that it has the properties of reflexivity ($\forall x\, x = x$), symmetry ($\forall x \forall y (x = y \rightarrow y = x)$), and transitivity ($\forall x \forall y \forall z ((x = y \wedge y = z) \rightarrow x = z)$).

$\vdash \forall x\, x = x$ and $\forall x \forall y (x = y \rightarrow y = x)$

- $$\begin{array}{c} \checkmark \neg \forall x\, x = x \\ \checkmark \exists x \neg\, x = x \\ \underline{\neg a = a} \end{array}$$

- $$\begin{array}{c} \checkmark \neg \forall x \forall y (x = y \rightarrow y = x) \\ \checkmark \exists x \neg \forall y (x = y \rightarrow y = x) \\ \checkmark \neg \forall y (a = y \rightarrow y = a) \\ \checkmark \exists y \neg (a = y \rightarrow y = a) \\ \checkmark \neg (a = b \rightarrow b = a) \\ a = b \\ \neg b = a \\ \underline{\neg b = b} \end{array}$$

$\vdash \forall x \forall y \forall z((x = y \wedge y = z) \rightarrow x = z)$

$$\checkmark \neg \forall x \forall y \forall z((x = y \wedge y = z) \rightarrow x = z)$$
$$\checkmark \exists x \neg \forall y \forall z((x = y \wedge y = z) \rightarrow x = z)$$
$$\checkmark \neg \forall y \forall z((a = y \wedge y = z) \rightarrow a = z)$$
$$\checkmark \exists y \neg \forall z((a = y \wedge y = z) \rightarrow a = z)$$
$$\checkmark \neg \forall z((a = b \wedge b = z) \rightarrow a = z)$$
$$\checkmark \exists z \neg((a = b \wedge b = z) \rightarrow a = z)$$
$$\checkmark \neg((a = b \wedge b = c) \rightarrow a = c)$$
$$\checkmark a = b \wedge b = c$$
$$\neg a = c$$
$$a = b$$
$$b = c$$
$$\underline{\neg b = c}$$

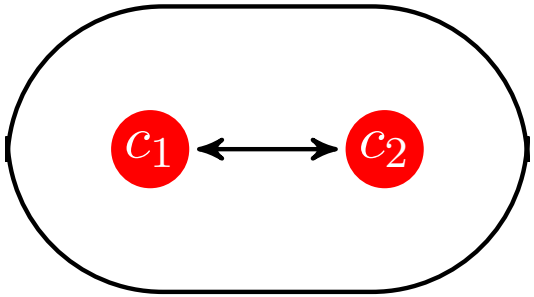
The Principle of Bivalence (PB)

- Our $\dfrac{}{c_1 = c_2 \quad \neg c_1 = c_2}$ rule (let's call it Identification) is a particular instance of a rule called the Principle of Bivalence (PB): $\dfrac{}{\varphi \quad \neg\varphi}$
- This more general rule is sound. Adding it to the calculus is a good idea if you wish to make tableaus smaller, but a bad idea if you want to use a variant of the calculus for automated theorem proving.
- Adding PB to the calculus is conservative: it is not the case that more things become provable as a result of the addition.
- Adding Identification, unlike unrestricted PB, does not wildly increase the search space.

Using Identification to Obtain Finite Models

$\forall x \exists y Rxy$
$\checkmark \neg \exists x Rxx$
$\forall x \neg Rxx$
$\neg Rc_1c_1$
$\checkmark \exists y Rc_1y$
Rc_1c_2

$c_1 = c_2$ | $\neg c_1 = c_2$
$\underline{\neg Rc_1c_2}$ | $\neg Rc_2c_2$
| $\checkmark \exists y Rc_2y$
| Rc_2c_3
| $c_3 = c_1$ | $\neg c_3 = c_1$
| Rc_2c_1 | $\vdots$

- Identification can be used to find a finite model if there is one.
- In the tableau on the left for $\forall x \exists y Rxy \nvdash \exists x Rxx$ we systematically try whether individuals can be identified.
- The second branch is open and finished and leads to the counterexample we have met before:
- 

Adding Axioms

- Predicate logic is a very general reasoning tool. It can be applied to many domains.
- More often than not we are working in a special domain, where additional reasoning principles obtain.
- The canonical way to deal with this situation is to write down a set of axioms.
- The axioms restrict the class of domains to those having the required structure.
- Axioms theoretically are just extra premises to our arguments, but in practice it is often wise to convert them into extra reasoning principles.

An Example: Axioms for Time

- Suppose you want to talk about time and you want to assume that the temporal precedence relation $\prec$ is a dense linear ordering without endpoints (a very conservative choice). Then you can write up the axioms for those:

A1	$\forall x \neg x \prec x$	(irreflexivity)
A2	$\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$	(transitivity)
A3	$\forall x \forall y(x \prec y \vee y \prec x \vee x = y)$	(connectedness)
A4	$\forall x \forall y(x \prec y \rightarrow \exists z(x \prec z \wedge z \prec y))$	(density)
A5	$\forall x \exists y\ x \prec y$	(no end)
A6	$\forall x \exists y\ y \prec x$	(no beginning)

Working with Axioms

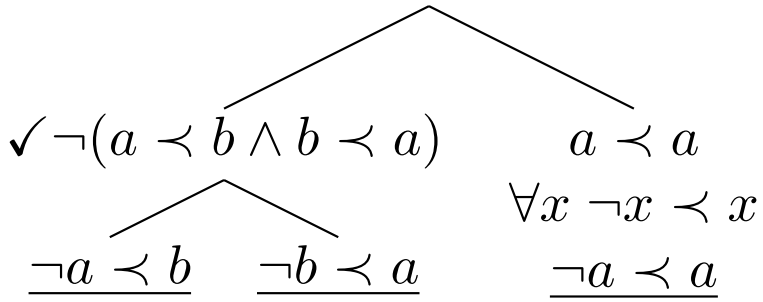
- Axioms such as the ones on the previous slide can be used as extra premises. When working with tableaus it is a bit more practical to add the rule:

$$\begin{array}{c} | \\ \varphi \end{array} \qquad \text{if } \varphi \text{ is an axiom.}$$

- This extra rule says that any axiom of the theory under consideration can be added at any point to any branch of a tableau.
- But this may still lead to rather involved tableaus. The following slide gives an example of a proof for A1,...,A6 $\vdash \forall x \forall y (x \prec y \rightarrow \neg y \prec x)$ (the asymmetry of precedence) that contains a lot of bookkeeping.

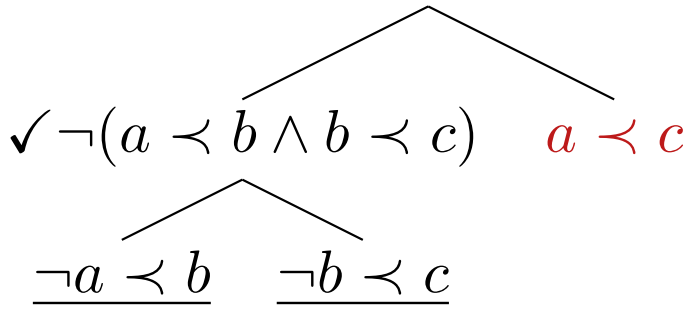
A1,. . . ,A6 $\vdash \forall x \forall y(x \prec y \rightarrow \neg y \prec x)$

$$\checkmark \neg \forall x \forall y(x \prec y \rightarrow \neg y \prec x)$$
$$\checkmark \exists x \neg \forall y(x \prec y \rightarrow \neg y \prec x)$$
$$\checkmark \neg \forall y(a \prec y \rightarrow \neg y \prec a)$$
$$\checkmark \exists y \neg(a \prec y \rightarrow \neg y \prec a)$$
$$\checkmark \neg(a \prec b \rightarrow \neg b \prec a)$$
$$a \prec b$$
$$\checkmark \neg \neg b \prec a$$
$$b \prec a$$
$$\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$$
$$\forall y \forall z((a \prec y \wedge y \prec z) \rightarrow a \prec z)$$
$$\forall z((a \prec b \wedge b \prec z) \rightarrow a \prec z)$$
$$\checkmark (a \prec b \wedge b \prec a) \rightarrow a \prec a$$



Derived Rules

$$a \prec b$$
$$b \prec c$$
$$\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$$
$$\forall y \forall z((a \prec y \wedge y \prec z) \rightarrow a \prec z)$$
$$\forall z((a \prec b \wedge b \prec z) \rightarrow a \prec z)$$
$$\checkmark(a \prec b \wedge b \prec c) \rightarrow a \prec c$$



- When working with certain axioms, it is often a good idea to derive new rules from them.
- Here it is shown how a rule

$$\begin{array}{c} a \prec b \\ b \prec c \\ \hline a \prec c \end{array}$$

can be derived in the presence of $\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$

Derived Rules in the Presence of Certain Axioms

Irreflexivity:

$$\mid$$
$$\neg a \prec a$$

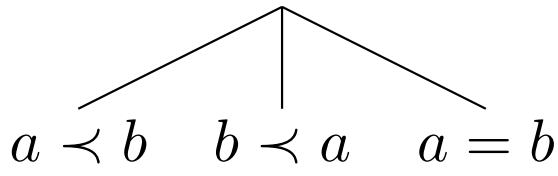
(a old)

Transitivity:

$$a \prec b$$
$$b \prec c$$
$$\mid$$
$$a \prec c$$

(a, b, c old)

Connectedness:



(a, b old)

Density:

$$a \prec b$$
$$\mid$$
$$a \prec c$$
$$c \prec b$$

(a, b old; c new)

No End:

$$\mid$$
$$a \prec b$$

(a old; b new)

No Beginning:

$$\mid$$
$$b \prec a$$

(a old; b new)

Looking Back and Forward

- In order to prepare the ground for our treatment of modal logics as fragments of predicate logic (and tableau systems for modal logics) we have looked at tableaux in predicate logic.
- We have also introduced the idea of axiomatic extensions and shown how useful axioms can often be compiled into rules.
- We will now move to Part 2 where it is shown how these techniques can be used to obtain basic temporal logics.

References I

Beth, E. (1955).
Semantic Entailment and Formal Derivability.
Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde 18(13), 309–342.

Gamut, L. T. F. (1991).
Logic, Language and Meaning, Volume 1.
Chicago: University of Chicago Press.

Hodges, W. (2001).
Logic.
Penguin Books.