Introduction to Logic in Computer Science: Autumn 2007

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Tableaux for Propositional Logic

The first part of the course will be devoted to

Automated Reasoning with Analytic Tableaux

This is a family of mechanical proof methods, developed for a variety of different logics. Tableaux are nice, because they are both easy to grasp for *humans* and easy to implement on *machines*.

Today we will be looking into tableau methods for classical *propositional* logic (we'll discuss first-order tableaux later).

Automated Reasoning

What the dictionaries say:

- **reasoning:** the process by which one judgement is deduced from another or others which are given
  (Oxford English Dictionary)

- **reasoning:** the drawing of inferences or conclusions through the use of *reason*
  reason: the power of comprehending, inferring, or thinking, esp. in orderly rational ways (cf. *intelligence*)
  (Merriam-Webster)

The scientific discipline of *Automated Reasoning* is concerned with the study of *reasoning processes* as *computational processes.*

Different Forms of Reasoning

Automated Reasoning covers a wide range of tasks, such as:

- *Deduction:* given a set of premises Δ and a conclusion φ, show that indeed $\Delta \models \varphi$ (this includes *theorem proving:* $\Delta = \{\,\}$)
- *Abduction/Induction:* given a theory T and an observation φ, find an explanation Δ such that $T \cup \Delta \models \varphi$
- *Satisfiability Checking:* given a set of formulas Δ, check whether there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \varphi$ for all $\varphi \in \Delta$
- *Model Checking:* given a model $\mathcal{M}$ and a formula φ, check whether $\mathcal{M} \models \varphi$

All of this for different *logics:* propositional or first-order, classical, intuionistic, modal, temporal, non-monotonic, . . .

Satisfiability Checking

An article on computational complexity in the *New York Times* from 13 July 1999 starts like this:

> *"Anyone trying to cast a play or plan a social event has come face-to-face with what scientists call a satisfiability problem. Suppose that a theatrical director feels obligated to cast either his ingénue, Actress Alvarez, or his nephew, Actor Cohen, in a production. But Miss Alvarez won't be in a play with Cohen (her former lover), and she demands that the cast include her new flame, Actor Davenport. The producer, with her own favors to repay, insists that Actor Branislavsky have a part. But Branislavsky won't be in any play with Miss Alvarez or Davenport. [...]"*

Is there a possible casting (and if there is, who will play)?

Tableaux

Early work by Beth and Hintikka (around 1955). Later refined and popularised by Raymond Smullyan:

- R.M. Smullyan. *First-order Logic*. Springer-Verlag, 1968.

Modern expositions include:

- M. Fitting. *First-order Logic and Automated Theorem Proving*. 2nd edition. Springer-Verlag, 1996.
- M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga (eds.). *Handbook of Tableau Methods*. Kluwer, 1999.
- R. Hähnle. *Tableaux and Related Methods*. In: A. Robinson and A. Voronkov (eds.), *Handbook of Automated Reasoning*, Elsevier Science and MIT Press, 2001.
- Proceedings of the yearly *Tableaux* conference: `http://i12www.ira.uka.de/TABLEAUX/`

What is it for?

The tableau method is a method for proving, in a mechanical manner, that a given set of formulas is *not satisfiable*. In particular, this allows us to perform automated *deduction:*

Given: set of *premises* Δ and *conclusion* φ

Task: prove $\Delta \models \varphi$

How? show $\Delta \cup \{\neg\varphi\}$ is not satisfiable (which is equivalent), *i.e.* add the complement of the conclusion to the premises and derive a contradiction (*“refutation procedure”*)

If a proof *fails*, we can *sometimes* conclude that the set in question *is* satisfiable, or that the conclusion in question does *not* follow (ok for propositional logic; for first-order logic only in special cases).

Constructing Tableau Proofs

- **Data structure:** a proof is represented as a *tableau*—a binary tree, the nodes of which are labelled with formulas.
- **Start:** we start by putting the premises and the negated conclusion into the root of an otherwise empty tableau.
- **Expansion:** we apply *expansion rules* to the formulas on the tree, thereby adding new formulas and splitting branches.
- **Closure:** we *close* branches that are obviously contradictory.
- **Success:** a proof is *successful* iff we can close all branches.

Propositional Tableau Rules

Alpha Rules:

$$\frac{A \wedge B}{\begin{array}{c} A \\ B \end{array}} \qquad \frac{\neg(A \vee B)}{\begin{array}{c} \neg A \\ \neg B \end{array}} \qquad \frac{\neg(A \rightarrow B)}{\begin{array}{c} A \\ \neg B \end{array}}$$

¬¬-Elimination:

$$\frac{\neg\neg A}{A}$$

Beta Rules:

$$\frac{A \vee B}{\begin{array}{c|c} A & B \end{array}} \qquad \frac{A \rightarrow B}{\begin{array}{c|c} \neg A & B \end{array}} \qquad \frac{\neg(A \wedge B)}{\begin{array}{c|c} \neg A & \neg B \end{array}}$$

Branch Closure:

$$\frac{\begin{array}{c} A \\ \neg A \end{array}}{\times}$$

Note: These are the standard ("Smullyan-style") tableau rules.

Smullyan's Uniform Notation

Formulas of conjunctive (α) and disjunctive (β) type:

α	α_1	α_2
$A \wedge B$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$

β	β_1	β_2
$A \vee B$	A	B
$A \rightarrow B$	$\neg A$	B
$\neg(A \wedge B)$	$\neg A$	$\neg B$

We can now state alpha and beta rules as follows:

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

Exercises

Show that the following are valid arguments:

- $\models ((P \rightarrow Q) \rightarrow P) \rightarrow P$
- $P \rightarrow (Q \wedge R), \neg Q \vee \neg R \models \neg P$

Termination

Assuming we analyse each formula at most once, we have:

Theorem 1 (Termination) *For any propositional tableau, after a finite number of steps no more expansion rules will be applicable.*

This must be so, because each rule results in ever shorter formulas.

Note: Importantly, termination will *not* hold in the first-order case.

Soundness and Completeness

Let φ be a propositional formula and Δ a set of such formulas. We write $\Delta \vdash \varphi$ to say that there exists a closed tableau for $\Delta \cup \{\neg\varphi\}$.

Before you can believe in the tableau method you need to prove:

Theorem 2 (Soundness) *If $\Delta \vdash \varphi$ then $\Delta \models \varphi$.*

Theorem 3 (Completeness) *If $\Delta \models \varphi$ then $\Delta \vdash \varphi$.*

Remember: $\Delta \vdash \varphi$ means that φ follows from Δ *according to the tableau method*; $\Delta \models \varphi$ means that φ *really* follows from Δ.

Proof of Soundness

We say that a *branch* is *satisfiable* iff the set of formulas on that branch is satisfiable. First prove the following lemma:

Lemma 1 (Satisfiable branches) *If a non-branching rule is applied to a satisfiable branch, the result is another satisfiable branch. If a branching rule is applied to a satisfiable branch, at least one of the resulting branches is also satisfiable.*

Now we can prove soundness by contradiction: assume $\Delta \vdash \varphi$ but $\Delta \not\models \varphi$ and try to derive a contradiction.

$\Delta \not\models \varphi \quad \Rightarrow \quad \Delta \cup \{\neg\varphi\}$ satisfiable $\quad \Rightarrow \quad$ initial branch satisfiable
$\Rightarrow \quad$ always at least one branch satisfiable (by the above lemma)

This contradicts our assumption that at one point all branches will be closed ($\Delta \vdash \varphi$), because a closed branch clearly is not satisfiable.

Hintikka's Lemma

We'll need this for the completeness proof:

Definition 1 (Hintikka set) *A set of propositional formulas H is called a (propositional) Hintikka set provided the following hold:*

(1) not both $P \in H$ and $\neg P \in H$ for propositional atoms P;

(2) if $\neg\neg\varphi \in H$ then $\varphi \in H$ for all formulas φ;

(3) if $\alpha \in H$ then $\alpha_1 \in H$ and $\alpha_2 \in H$ for alpha formulas α;

(4) if $\beta \in H$ then $\beta_1 \in H$ or $\beta_2 \in H$ for beta formulas β.

Lemma 2 (Hintikka) *Every Hintikka set is satisfiable.*

Proof sketch. (1) Build a model $\mathcal{M}$ from H: every atom $P \in H$ is true in $\mathcal{M}$, and only those. (2) Show by structural induction that $\mathcal{M} \models \varphi$ for *all* formulas $\varphi \in H$. $\Box$

Proof of Completeness

We will show the contrapositive of the claim: if $\Delta \nvdash \varphi$ then $\Delta \not\models \varphi$. That is, we will show: if there is a tableau for $\Delta \cup \{\neg\varphi\}$ that cannot be closed, then $\Delta \cup \{\neg\varphi\}$ is satisfiable.

Now suppose $\Delta \nvdash \varphi$. Let B be the branch that cannot be closed.

Observe that any branch that cannot be closed is a Hintikka set. Hence, by Hintikka's Lemma, B is satisfiable. But then $\Delta \cup \{\neg\varphi\}$ must be satisfiable as well.

Decidability

Our discussion confirms the decidability of propositional logic:

Theorem 4 (Decidability) *The tableau method is a decision procedure for classical propositional logic.*

Proof. To check validity of φ, develop a tableau for $\neg\varphi$. Because of termination, we will eventually get a tableau that is either (1) closed or (2) that has a branch that cannot be closed.

- In case (1), the formula φ must be valid (soundness).
- In case (2), the branch that cannot be closed shows that $\neg\varphi$ is satisfiable (see completeness proof), *i.e.* φ cannot be valid.

That's it. $\square$

KE: Analytic Tableaux with Semantic Branching

We are now going to introduce a variant of the tableau method, called KE, which includes a special branching rule:

$$\begin{array}{c|c} \hline A & \neg A \end{array}$$

We call this rule PB (for *principle of bivalence*). Sometimes it is also called a *semantic branching* rule, as opposed to the beta rule of Smullyan's tableau method, which is a *syntactic branching* rule.

PB is clearly sound (why?). But, usually, not requiring a cut rule such as PB is considered a great advantage ($\rightsquigarrow$ cut elimination). However, as we shall see, if applied properly it can be very useful.

The obvious danger is that PB could be used with *any* formula A. So we might lose the *analytic* character of the method, which has so far given clear guidelines on how to construct a proof …

Propositional KE Rules

Alpha Rules:

$$\frac{A \wedge B}{\begin{array}{c}A\\B\end{array}} \qquad \frac{\neg(A \vee B)}{\begin{array}{c}\neg A\\\neg B\end{array}} \qquad \frac{\neg(A \rightarrow B)}{\begin{array}{c}A\\\neg B\end{array}}$$

¬¬-Elimination:

$$\frac{\neg\neg A}{A}$$

Beta Rules:

$$\frac{\begin{array}{c}A \vee B\\\neg A\end{array}}{B} \qquad \frac{\begin{array}{c}A \rightarrow B\\A\end{array}}{B} \qquad \frac{\begin{array}{c}\neg(A \wedge B)\\A\end{array}}{\neg B}$$

$$\frac{\begin{array}{c}A \vee B\\\neg B\end{array}}{A} \qquad \frac{\begin{array}{c}A \rightarrow B\\\neg B\end{array}}{\neg A} \qquad \frac{\begin{array}{c}\neg(A \wedge B)\\B\end{array}}{\neg A}$$

PB:

$$\frac{}{A \mid \neg A}$$

Branch Closure:

$$\frac{\begin{array}{c}A\\\neg A\end{array}}{\times}$$

KE: Smullyan's Uniform Notation

Recall the uniform notation for alpha and beta formulas:

α	α_1	α_2
$A \wedge B$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$

β	β_1	β_2
$A \vee B$	A	B
$A \rightarrow B$	$\neg A$	B
$\neg(A \wedge B)$	$\neg A$	$\neg B$

We can also state alpha and beta rules for KE as follows:

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\begin{array}{c}\beta\\ \beta_1^c\end{array}}{\beta_2} \qquad \frac{\begin{array}{c}\beta\\ \beta_2^c\end{array}}{\beta_1} \qquad \text{where } A^c = \begin{cases} A' & \text{for } A = \neg A', \\ \neg A & \text{otherwise} \end{cases}$$

Eta Rules for KE

$$\frac{\begin{array}{c} A \leftrightarrow B \\ A \end{array}}{B} \qquad \frac{\begin{array}{c} A \leftrightarrow B \\ B \end{array}}{A} \qquad \frac{\begin{array}{c} \neg(A \leftrightarrow B) \\ A \end{array}}{\neg B} \qquad \frac{\begin{array}{c} \neg(A \leftrightarrow B) \\ B \end{array}}{\neg A}$$

$$\frac{\begin{array}{c} A \leftrightarrow B \\ \neg A \end{array}}{\neg B} \qquad \frac{\begin{array}{c} A \leftrightarrow B \\ \neg B \end{array}}{\neg A} \qquad \frac{\begin{array}{c} \neg(A \leftrightarrow B) \\ \neg A \end{array}}{B} \qquad \frac{\begin{array}{c} \neg(A \leftrightarrow B) \\ \neg B \end{array}}{A}$$

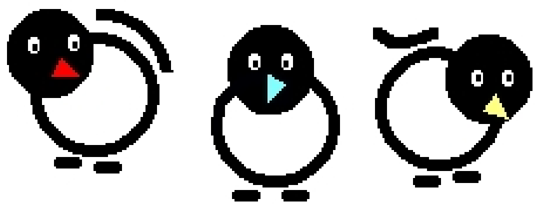
WinKE: An Interactive Proof Assistant

You can practice constructing tableau proofs using WinKE:

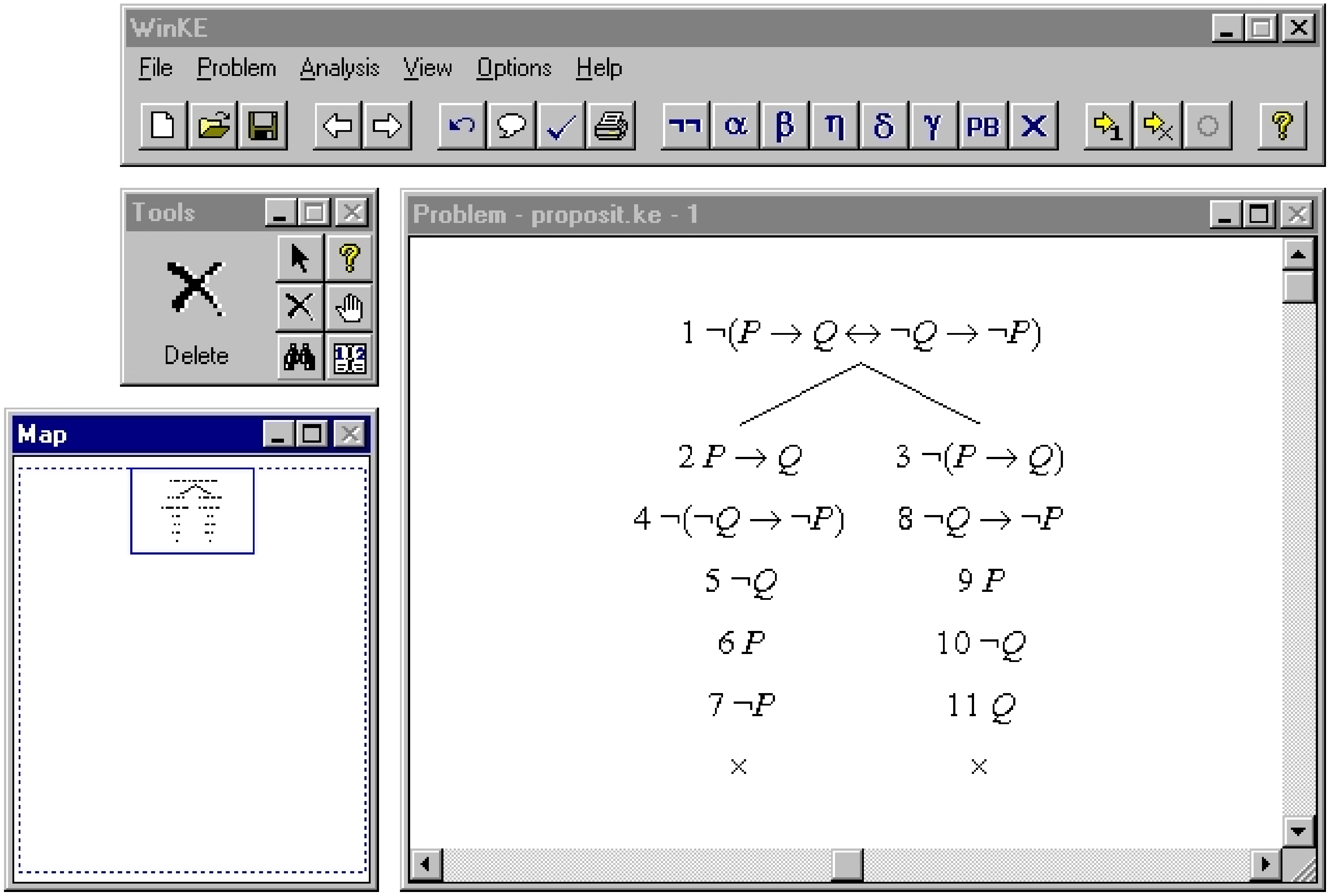
`http://www.illc.uva.nl/~ulle/WinKE/`

Download it from Blackboard (the beginners' version). Note:

- WinKE only works for the KE-variant of the tableau method (with semantic branching).
- You can also use it to construct tableaux for first-order logic.
- Beware: some bugs (see list at website) and it only works under Windows (sorry!).



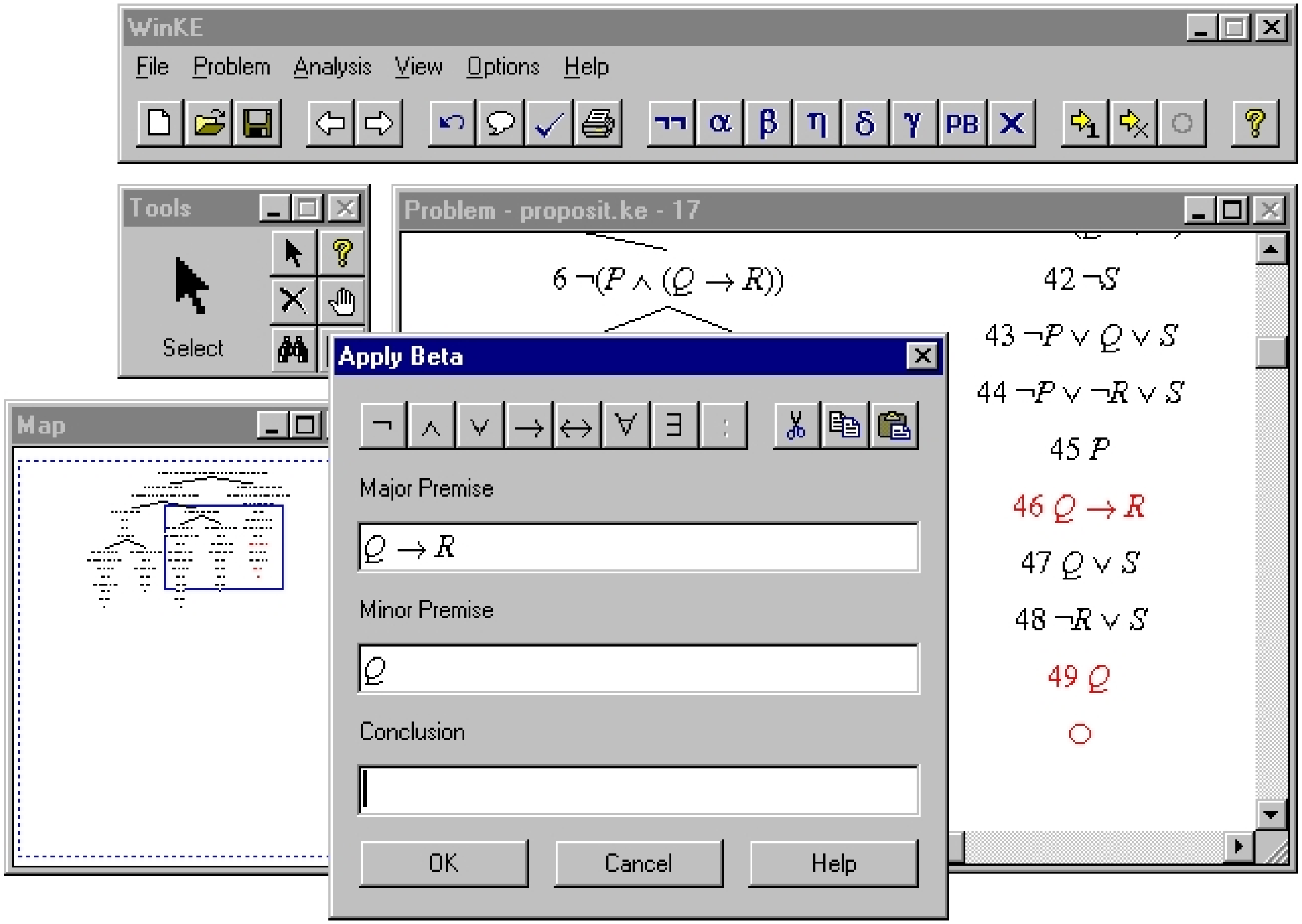
WinKE: Example


WinKE
File Problem Analysis View Options Help
¬¬ α β η δ γ PB ✕
Tools
Delete
Map
Problem - proposit.ke - 1

1 $\neg(P \to Q \leftrightarrow \neg Q \to \neg P)$

2 $P \to Q$ | 3 $\neg(P \to Q)$
4 $\neg(\neg Q \to \neg P)$ | 8 $\neg Q \to \neg P$
5 $\neg Q$ | 9 P
6 P | 10 $\neg Q$
7 $\neg P$ | 11 Q
$\times$ | $\times$

WinKE: Another Example



6 $\neg(P \wedge (Q \to R))$

42 $\neg S$

43 $\neg P \vee Q \vee S$

44 $\neg P \vee \neg R \vee S$

45 P

46 $Q \to R$

47 $Q \vee S$

48 $\neg R \vee S$

49 Q

$\circ$

Major Premise

$Q \to R$

Minor Premise

Q

Conclusion

General KE is not Analytic

PB makes our tableau calculus *non-analytic:* any formula A could be used for splitting. This generates some problems:

- How do you decide which formula to use for splitting? (And how do you decide when to apply PB in the first place?)
- We lose the termination property.

We should somehow restrict the range of formulas to which we apply PB. Any ideas how?

Analytic PB and the Subformula Property

We'd like to have the *subformula property:* the formulas used in a proof are all subformulas of the original statement to be shown.

- Smullyan-style tableaux have it.
- KE does not.

Fortunately, we can refine the PB rule:

- Obviously, only applications of PB to subformulas of formulas already on the branch could ever contribute to its closure . . .
- In fact, we can even restrict PB to the immediate subformulas of non-analysed beta formulas. This is called *analytic PB*.

It is not difficult to show that KE with analytic PB is (sound and) *complete* for classical propositional logic.

Note: In the presence of $\leftrightarrow$, the (analytic) PB rule should (only) be applied to subformulas of non-analysed beta and eta formulas.

Exercise

Give a KE proof for the following theorem:

$$(P \wedge Q) \vee (R \wedge \neg R) \;\rightarrow\; \neg(\neg P \vee \neg Q)$$

What happens if we use different "strategies" concerning the order in which we pick rules to apply and formulas to analyse?

Finding Short Proofs

It is an open problem (with no solution in sight) to find an efficient algorithm to decide in all cases which rule to use next in order to derive the shortest possible proof.

However, we can give some rough guidelines:

- Always apply any applicable *non-branching rules first.* In some cases, these may turn out to be redundant, but they will never cause an exponential blow-up of the proof.
- *Beta simplification:* Do not attempt to ananlyse beta formulas that are subsumed on the branch (e.g. if both $\beta_1 \vee \beta_2$ and β_1 are on the branch, then the former is redundant, so don't apply PB to β_1 or β_2 just because of $\beta_1 \vee \beta_2$).
- It seems like a good idea to prefer those candidate PB formulas that appear most frequently on a branch, but this is not always the best strategy . . .

Efficiency

Are either of the two variants of the tableau method *efficient* ways of checking whether a formula is a tautology?

It's pretty clear that *truth-tables* are *not* an efficient method: checking a formula involving n propositional variables requires filling in 2^n rows (exponential = very bad).

Are tableaux any better?

Exercise

Give proofs for the unsatisfiability of the following formula using (1) truth-tables, (2) Smullyan-style tableaux, and (3) KE:

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

P-Simulation

Intuitively, one proof system is at least as good as the next iff it never requires a longer proof for the same theorem. Actually, this is asking too much: we are satisfied if the increase in size is at most polynomial (rather than exponential).

Definition 2 (p-simulation) *A proof system $\mathcal{A}$ p-simulates another proof system $\mathcal{B}$ (deriving the same theorems) iff there is a function g, computable in polynomial time, that maps derivations for any formula φ in $\mathcal{B}$ to derivations for φ in $\mathcal{A}$.*

Essentially, this definition is about the comparative *length of proofs* in different proof systems.

Smullyan-style Tableaux and Truth-Tables

Rather surprisingly, we get:

Theorem 5 (D'Agostino, 1992) *Smullyan-style tableaux cannot p-simulate the truth-table method.*

Proof idea. Analyse the size of proofs for formulas of the form:

$$H_{P_1,\ldots,P_n} \quad = \quad \neg \bigwedge \{P_1^* \vee \cdots \vee P_n^* \mid P_i^* \in \{P_i, \neg P_i\}\}$$

That's the negation of the conjunction of all possible "complete" disjunctions of literals (involving either P_i or $\neg P_i$ for all i). [. . .] Note that $n!$ grows faster than any polynomial function of 2^n. $\square$

In fact, Smullyan tableaux and truth-tables are *incomparable* in terms of p-simulation (the other direction is homework). So neither method is better in all cases. Of course, in practice, the tableau method often *is* very much better than using truth-tables.

KE and Competitors

It's easy to see that KE can p-simulate (even *linearly* simulate) both truth-tables and Smullyan-style tableaux:

- *Truth-tables:* Start with PB for each propositional atom to build an initial tree with 2^n branches, enumerating all possible models. That's like setting up a truth-table; the rest is linear.
- *Smullyan-style tableaux:* Whenever Smullyan would use a beta rule on $\beta_1 \vee \beta_2$, first use PB with β_1 and then a KE-style beta rule with $\beta_1 \vee \beta_2$ and $\neg\beta_1$ on the righthand branch. In the sequel, simply ignore the additional formula $\neg\beta_1$.

So Smullyan-style tableaux cannot p-simulate KE (why?).

But note that this doesn't necessarily mean that KE will be *faster* than Smullyan-style tableaux (with syntactic branching).
For instance, a beta application in KE requires a complex search, while it is instant for syntactic branching.

Summary

- Two tableau methods for propositional logic: Smullyan-style (syntactic branching) and KE-style (semantic branching)
- Termination, soundness, completeness, decidability
- Efficiency issues: which rule?; which formula?; analytic PB
- Smullyan-style are less space-efficient than KE-style tableaux
- Much of what we have covered can be found in:
  - M. D'Agostino. Tableau Methods for Classical Propositional Logic. In *Handbook of Tableau Methods*, Kluwer, 1999.

  For further reading, refer to the references on slide 6.
- Next: tableaux for first-order logic