Lecture 08

Automated Reasoning

KE for Propositional, Predicate and Modal Logic

Jeremy Pitt
Department of Electrical & Electronic Engineering
Imperial College London, UK
Email: j.pitt _at_ imperial _dot_ ac _dot_ uk
Phone: (int) 46318
Office: 1010 EEE

Aims and Objectives

- Aims
  - To introduce a calculus for sound and complete reasoning in propositional, predicate and modal logics
    * Calculus: any system of symbolic representation admitting manipulating according to given rules
  - To discuss data structures and algorithms for a Prolog implementation

- Objectives
  - Understand the issues in the specification and implementation of proof procedures for various logics

Motivation

- Compute **logical consequence**, i.e. the **entailment** relation $\models$
  - $\models$ is a 2-place (binary) relation, between sets of formulas and formulas
  - A set of formulas S (**premises**) and a single formula p (**conclusion**) are in this relation ($S \models p$) if every valuation that makes each member $s \in S$ true, also makes p true
  - The Deduction Theorem states

$$(S \models p) \cong (S - \{s\}) \models (s \to p)$$

  - So let

$$S' = \bigwedge_{i=1}^{n} s_i = s_1 \wedge s_2 \wedge \ldots \wedge s_n$$

  - Then

$$(S \models p) \quad \cong \quad \{\} \models (S' \to p)$$

Issues with Entailment

- Problems with using semantics
  - We can't pre-compute $\models$ and look up membership
  - Truth tables for propositional logic are exponential in the number of symbols
  - Can't even use truth tables for predicate logic; can't check an infinite number of interpretations

- Therefore use syntax instead, i.e. the **proves** relation $\vdash$

- However
  - Many different methods of specifying/computing a proves relation
    - Sequent calculus, natural deduction, conversion to normal form, ...
    - $\vdash_{SC}$, $\vdash_{ND}$, $\vdash_{NF}$, ...
  - Have to show that the method is **sound** and **complete**

- We will use **proof by refutation**

Proof by Refutation

- Prolog's proof procedure
  - Representation of formulas: Horn clauses (certain type of normal form)
  - Query Q treated as $\neg Q$
  - Algorithm: Depth first search
  - Use resolution as the inference rule
  - Build: a search tree
  - Aim: derive empty query (contradiction)
  - So deduce Q (and if Q was $p(X)$, instances of X which satisfy $p(X)$)

By Contrast

- The calculus KE
  - Representation of formulas: standard form (using Prolog operators)
  - Query Q treated as $\neg Q$
  - Algorithm: search for a refutation by expanding $\neg Q$ according to, but 'clearing away', the logical structure of Q
  - Inference rules: elimination rules (resolution is just one instance)
  - Build: a tree, called a **tableau** or a **KE-tree**, with formulas labelling nodes
  - Aim: derive closure
  - And so:
    - Theorem proving (true for all valuations, interpretations or models)
    - Model building (find a valuation, interpretation or model that satisfies the formulas)

Basis of KE

- Consider the formula $F = ((p \to q) \wedge (q \to r)) \to (p \to r)$
  - A complex formula F consists of sub-formulas joined by a connective
  - Likewise sub-formulas, until you get to the literals

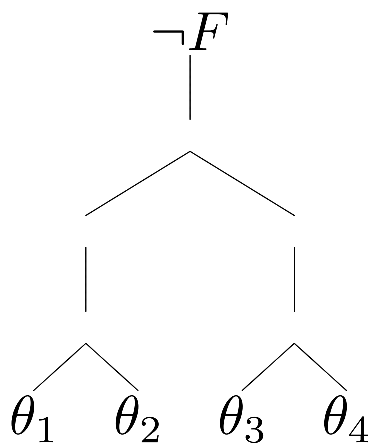
formula	sub-formulas				literals			
F	$SF_1,$	$SF_2,$	$\ldots$	SF_n	$l_1,$	$l_2,$	$\ldots$	l_m

formula	sub-formulas				literals		
F	$(p \to q) \wedge (q \to r)$	$p \to q$	$q \to r$	$p \to r$	p	q	r
1	0	1	0	0	1	1	0

- Proof by refutation
  - If F is a tautology, then always a 1 in the final (leftmost) column
  - Look for a 1 for $\neg F$, and **fail to find it**
  - Provided you are guaranteed to find it if it exists
  - If a formula is a contradiction, then its DNF $\cong 0$
  - Therefore construct the DNF of $\neg F$, and show it is $\cong 0$

KE Proof Procedure (1)

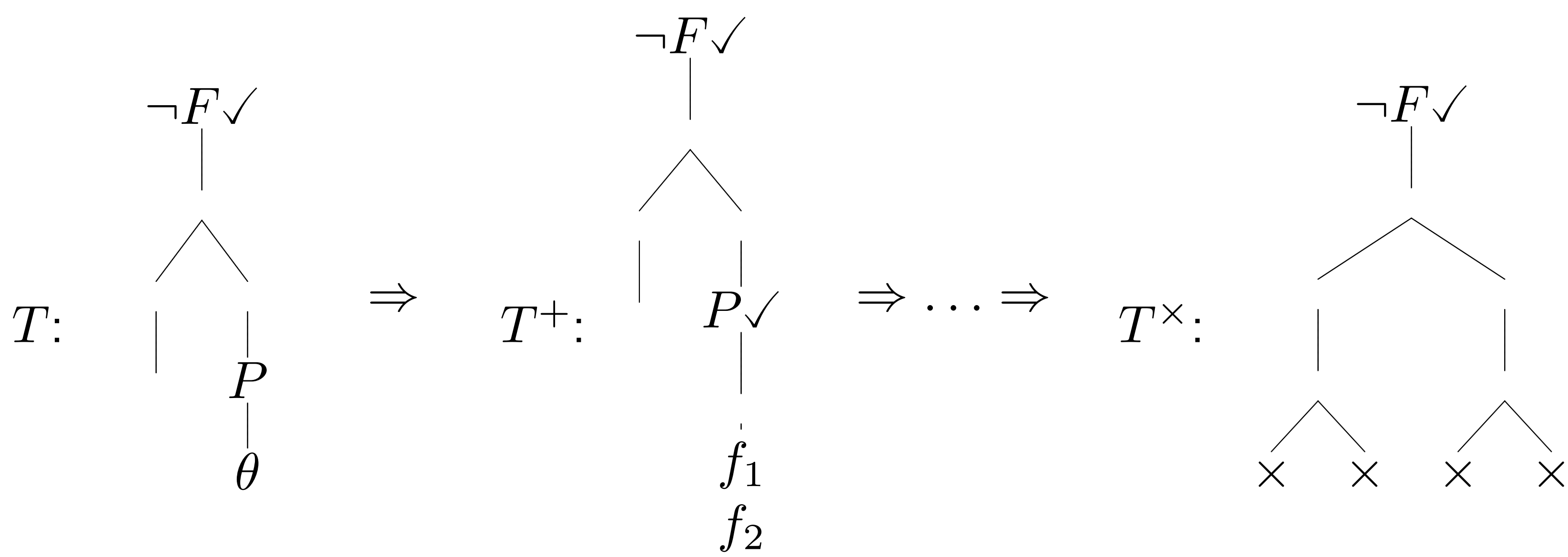
- KE Proof Procedure: To prove $\vdash_{KE} F$
  - Start from $\neg F$
  - Build a KE-tree, with nodes labelled by formulas (nodes=formulas)



- In a KE-tree
  - Each branch θ_i represents a conjunction of the formulas
  - KE-tree T represents a disjunction of branches $T = \theta_1 \vee \theta_2 \vee \ldots \vee \theta_n$

KE Proof Procedure (2)

- KE Calculus specifies a number of rules
  - The rules specify how to convert one KE-tree T into another T^+
    - Select a branch θ
    - Analyse a formula P on the branch (according to the rules)
    - Extend the branch with new formulas (according to the rules)
  - Aim is to construct a **closed** KE-tree $T^\times$



KE Calculus Rules (1)

- Single premise rules: only extend the branch on which the formula occurs

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

- Two premise rules (major and minor premise): both premises must be on the same branch – only extend that branch

$$\frac{\begin{array}{l} p \rightarrow q \\ p \end{array}}{q} \qquad \frac{\begin{array}{r} p \rightarrow q \\ \neg q \end{array}}{\neg p} \qquad \frac{\begin{array}{l} p \vee q \\ \neg p \end{array}}{q} \qquad \frac{\begin{array}{r} p \vee q \\ \neg q \end{array}}{p} \qquad \frac{\begin{array}{l} \neg(p \wedge q) \\ p \end{array}}{\neg q} \qquad \frac{\begin{array}{r} \neg(p \wedge q) \\ q \end{array}}{\neg p}$$

- Branching rule: 0-premise rule, any branch, any time, any formula (but see later)

$$\begin{array}{ccc} & \wedge & \\ p & & \neg p \end{array}$$

KE Calculus Rules (2)

- More two premise rules (same branch, etc.)

$$\frac{p \leftrightarrow q \quad p}{q} \qquad \frac{p \leftrightarrow q \quad \neg p}{\neg q} \qquad \frac{p \leftrightarrow q \quad q}{p} \qquad \frac{p \leftrightarrow q \quad \neg q}{\neg p}$$

$$\frac{\neg(p \leftrightarrow q) \quad p}{\neg q} \qquad \frac{\neg(p \leftrightarrow q) \quad \neg p}{q} \qquad \frac{\neg(p \leftrightarrow q) \quad q}{\neg p} \qquad \frac{\neg(p \leftrightarrow q) \quad \neg q}{p}$$

- Closure rule (same branch)

$$\frac{p \quad \neg p}{\times}$$

Smullyan's Uniform Notation

- A useful shorthand identifying **components** of a formula
  - Conjunctive **alpha** formulas

α	α_1	α_2
$p \wedge q$	p	q
$\neg(p \rightarrow q)$	p	$\neg q$
$\neg(p \vee q)$	$\neg p$	$\neg q$

  - Disjunctive **beta** formulas

β	β_1	β_2
$\neg(p \wedge q)$	$\neg p$	$\neg q$
$p \rightarrow q$	$\neg p$	q
$p \vee q$	p	q

- Complements
  - The complement of a positive formula P is $\neg P$
  - The complement of a negative formula $\neg P$ is P
  - Denote the complement of a formula p by p^c

- The rules can be expressed succinctly

alpha	beta		closure	PB (branching)	beta simplification	
$\dfrac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}}$	$\dfrac{\begin{array}{c}\beta\\ \beta_1^c\end{array}}{\beta_2}$	$\dfrac{\begin{array}{c}\beta\\ \beta_2^c\end{array}}{\beta_1}$	$\dfrac{\begin{array}{c}P\\ P^c\end{array}}{\times}$	$\begin{array}{c}\diagup\ \diagdown\\ P\quad P^c\end{array}$	$\dfrac{\begin{array}{c}\beta\\ \beta_1\end{array}}{\ }$	$\dfrac{\begin{array}{c}\beta\\ \beta_2\end{array}}{\ }$

- Beta simplification
  - If a component of a beta formula is already on the branch, the formula can be analysed without extending the branch
  - For example, $p \vee q$ and p occurs on a branch. Applying PB gives
    * Either:
      p (no new information) and $\neg p$ (immediate closure)
    * Or:
      q (must have $\neg p$, so could have closed) and $\neg q$ (no new information)

Proofs in KE

- A branch θ of a KE-tree is **closed** if both P and $\neg P$ occur on the branch
- A KE-tree is (atomically) **closed** of every branch is closed
  - Atomic closure requires P to be a literal
- a KE **proof** of P is a closed KE-tree for $\neg P$
- P is a theorem if $\vdash_{KE} P$
- P is a logical consequence of S if $S \vdash_{KE} P$
  - This is the same as $\vdash_{KE} S' \rightarrow P$
  - So we are looking for a closed KE-tree for $\neg(S' \rightarrow P)$
  - So the root of the tree is all the premises in S, and the negation of the conclusion $\neg P$

Starting a KE Proof

$$\neg(S' \to P)$$

$$\neg((s_1 \land s_2 \land \ldots \land s_n) \to P)$$

$$\neg((s_1 \land s_2 \land \ldots \land s_n) \to P)$$
$$(s_1 \land s_2 \land \ldots \land s_n)$$
$$\neg P$$

$$\neg((s_1 \land s_2 \land \ldots \land s_n) \to P)$$
$$(s_1 \land s_2 \land \ldots \land s_n)$$
$$\neg P$$
$$s_1$$
$$(s_2 \land \ldots \land s_n)$$

Examples

- Try proving some theorems
  - $p \to (q \to p)$
  - $((p \to q) \to p) \to p$
  - $((p \to q) \land (q \to r)) \to (p \to r)$
  - $((p \to r) \land (q \to r)) \to ((p \lor q) \to r)$
  - $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
  - $(p \land (q \lor r)) \leftrightarrow ((p \land q) \lor (p \land r))$
  - $(p \lor (q \land r)) \leftrightarrow ((p \lor q) \land (p \lor r))$
- Try proving some logical consequences

$\leftrightarrow$ and eta formulas

η	$\eta_{1,1}$	$\eta_{1,2}$	$\eta_{2,1}$	$\eta_{2,2}$
$P \leftrightarrow Q$	P	Q	$\neg P$	$\neg Q$
$\neg(P \leftrightarrow Q)$	P	$\neg Q$	$\neg P$	Q

$$\begin{array}{cc} \multicolumn{2}{c}{\text{eta}} \\ \hline \eta & \eta \\ \dfrac{\eta_{1,x}}{\eta_{1,y}} & \dfrac{\eta_{2,x}}{\eta_{2,y}} \end{array}$$

- With the note that "if $x = 1$ then $y = 2$ else [$x = 2$] $y = 1$"

- If (on some branch) you have an η formula (negated or un-negated) and one component of either pair (on the same branch), then that branch can be extended with the other component (of the same pair).

- Proving a formula $\neg(P \leftrightarrow Q)$? – think PB

Open Branches and Model Building

- In a KE-tree
  - Each branch represents the conjunction of formulas appearing on it
  - The tree itself represents the disjunction of its branches
  - If a branch is closed, it is because some p and -its complement cannot both be true
  - Otherwise, if we have analysed all the formulas on the branch, then there is a valuation for all the literals which
    * Satisfies the premises, and
    * Falsifies the conclusion

- Model building
  - Rather than looking to see if every branch is closed, check to see if any branch (ideally just one) is open
  - Instead of adding the negated conclusion, add the formula and look for an open branch

Consistency Checking

- In the *Obligatio Game*, a finite number n of rounds is chosen, depending on the severity of the exam. The examiner then gives the candidate successive assertions, $\phi_1, \phi_2, \ldots, \phi_n$ that she has to either $accept$ or $reject$ as each one is put forward. In the former case, ϕ_i is added to the candidate's stock of commitments; in the latter case, the negation $\neg\phi_i$ is added.

- The candidate passes the exam if she maintains the consistency of her stock of commitments throughout all n assertions.

- The KE calculus can be used show which sequences of answers will ensure that the student passes the exam.

- Try to construct a KE-proof for each possible sequence of $accept$ and $reject$.

Obligatio Game: Example (1)

- Suppose a candidate is exposed (successively) to the following three ($n = 3$) assertions:

  1. $q \vee \neg(p \vee r)$
  2. $p \rightarrow q$
  3. q

- It is necessary to check

  - $q \vee \neg(p \vee r)$, $p \rightarrow q$, q
  - $q \vee \neg(p \vee r)$, $p \rightarrow q$, $\neg q$
  - $q \vee \neg(p \vee r)$, $\neg(p \rightarrow q)$, q
  - $q \vee \neg(p \vee r)$, $\neg(p \rightarrow q)$, $\neg.q$
  - $\neg(q \vee \neg(p \vee r))$, $p \rightarrow q$, q
  - $\neg(q \vee \neg(p \vee r))$, $p \rightarrow q$, $\neg q$
  - $\neg(q \vee \neg(p \vee r))$, $\neg(p \rightarrow q)$, q
  - $\neg(q \vee \neg(p \vee r))$, $\neg(p \rightarrow q)$, $\neg q$

Obligatio Game: Example (2)

1	$q \vee \neg(p \vee r)$	assertion 1
2	$p \to q$	assertion 2
3	q	assertion 3
	open	

1	$q \vee \neg(p \vee r)$	assertion 1
2	$p \to q$	assertion 2
3	$\neg q$	assertion 3
4	$\neg(p \vee r)$	$(\beta, 1, 3)$
5	$\neg p$	$(\alpha, 4)$
6	$\neg r$	$(\alpha, 4)$
	open	

1	$q \vee \neg(p \vee r)$	assertion 1
2	$\neg(p \to q)$	assertion 2
3	q	assertion 3
4	p	$(\alpha, 2)$
5	$\neg q$	$(\alpha, 2)$
	$\times$	$(3, 5)$

1	$q \vee \neg(p \vee r)$	assertion 1
2	$\neg(p \to q)$	assertion 2
3	$\neg q$	assertion 3
4	$\neg(p \vee r)$	$(\beta, 1, 3)$
5	$\neg p$	$(\alpha, 4)$
6	$\neg r$	$(\alpha, 4)$
7	p	$(\alpha, 2)$
8	$\neg q$	$(\alpha, 2)$
	$\times$	$(5, 7)$

Soundness

- Some definitions/observations
  - A set of formulas S is satisfiable if there is valuation which makes all the formulas in S true
  - A branch θ on a KE-tree T is satisfiable if every formula on the branch is satisfiable
  - A KE-tree T is satisfiable if some branch of T is satisfiable

- 'Lemmas':
  - 1. Applying a KE rule to a satisfiable KE-tree yields another satisfiable KE-tree (check each rule)
  - 2. A closed KE-tree is not satisfiable
    - Every branch contains P and $\neg P$ (for some P)
    - No valuation makes both P and $\neg P$ true

- If there is a closed KE-tree for S, then S is not satisfiable
  - Suppose there is a closed KE-tree for S, and S is satisfiable
  - Then by (1), every KE-tree derived from S is satisfiable, including $T^{\times}$
  - But by (2), there are no closed, satisfiable tableau
  - Contradiction
- Therefore, if $\vdash_{KE} S$, then $\models S$
  - If $\vdash_{KE} S$, then there is a closed KE-tree for $\neg S$
  - If there is a closed KE-tree for $\neg S$, then $\neg S$ is not satisfiable
  - Therefore S is a tautology, so $\models S$

Completeness (Sketch)

- **Analytic restriction**: Only apply the PB branching rule to sub-formulas

- Suppose S is a tautology, then construct a (restricted) KE-tree for $\neg S$
  - Apply all the rules to all the formulas
  - Continue until no more rules can be applied – all the formulas have been **analysed**
  - This process must terminate (the rules only ever generate sub-formulas)
  - Let $T^{\times}$ be the final tableau

- Suppose $T^{\times}$ is not atomically closed
  - Let θ be one of the non-closed branches
    - If $\neg\neg p$ occurs on θ, and has been analysed, then so does p
    - If α occurs on θ, and has been analysed, then so do α_1 and α_2
    - If β occurs on θ, and has been analysed, then so do either β_1 or β_2

- By definition, this is a **Hintikka Set**
- By Hintikka's Lemma, this is set is satisfiable (sketch later)
- This set must include $\neg S$, because $\neg S$ is on every branch
- Therefore there is a valuation that satisfies $\neg S$, so S is not a tautology
- This is a contradiction

- Therefore $T^{\times}$ is atomically closed

- Therefore, if $\models S$, then $\vdash_{KE} S$

Completeness (Sketch Sketch)

- Hintikka set: maximally consistent set satisfying sub-formula property
- A set of propositional formulas H is a (propositional) Hintikka set provided the following four conditions hold:
  - For propositional atoms p, not both $p \in H$ and $-p \in H$
  - For all formulas ϕ, if $\neg\neg\phi \in H$, then $\phi \in H$
  - For alpha formulas α, if $\alpha \in H$, then $\alpha_1 \in H$ and $\alpha_2 \in H$
  - For beta formulas β, if $\alpha \in H$, then $\beta_1 \in H$ or $\beta_2 \in H$
- Hintikka's Lemma: Proof (sketch)
  - Construct a model $\mathcal{M}$ from H such that for every atom $p \in H$, and only those atoms, $\models_{\mathcal{M}} p$
  - Show by structural induction for all formulas $\phi \in H$ that $\models_{\mathcal{M}} \phi$

Notes on Proof Method

- KE rules are non-deterministic
  - Say what can be done, not what should be done
    - Pick any formula to analyse next
    - Not use some formulas at all
    - Use some formulas more than once
    - Close on complex formulas
    - Branch on any formula

- Usual to apply restrictions
  - Analytic restriction: obey the sub-formula property (for branching)
  - Strictness: only analyse a formula once on a branch
  - Closure: only close on literal (atomic closure)
  - Order does not matter, provided every formula is analysed at least once

- These restrictions don't affect soundness and completeness

Algorithm for Satisfiability

- Data Structure
  - U – list of ‘unvisited’ formulas
  - V – list of ‘visited’ formulas
  - L – list of literals
  - G – list of analysed formulas

- (In the sequel, L1++L2 is a function which returns L3 in the relation append(L1, L2, L3).)

- Query: ?- ke([-F], [], [], []).

- Case 1: ke([F|U], V, L, G)
  - if F is a literal
    * if $F^c \in L$ return unsatisfiable
    * otherwise return ke(U, V, [F|L], G)
  - if F is an alpha formula (alpha(F, A1, A2))
    * return ke([A1, A2|U]++V, [], L, [F|G])
  - if F is a beta formula (beta(F, B1, B2))
    * if $B1 \in U +\!\!+ V +\!\!+ L +\!\!+ G$ return ke(U, V, L, [F|G])
    * if $B1^c \in U +\!\!+ V +\!\!+ L +\!\!+ G$ return ke([B2|U]++V, [], L, [F|G])
    * if $B2 \in U +\!\!+ V +\!\!+ L +\!\!+ G$ return ke(U, V, L, [F|G])
    * if $B2^c \in U +\!\!+ V +\!\!+ L +\!\!+ G$ return ke([B1|U]++V, [], L, [F|G])
    * otherwise, return ke(U, [F|V], L, G)

- Case 2: ke([], [F|V], L, G) returns
  - F is a beta formula (beta(F, B1, B2))
  - return ke([F, B1|V], [], L, G) AND ke([F, B1c|V], [], L, G)

KE for (Propositional) Modal Logic

- Need to know in which world a formula is true

- To do this, use **prefixes**
  - A prefix σ is a name for a possible world
  - Formulas will be **labelled** by a prefix, $\sigma : P$
  - $\sigma : P$ means P is true in the world labelled (named) σ

- The prefix system is designed so that it can syntactically determined if worlds so labelled are in the accessibility relation R
  - A prefix is a non-empty finite sequence of integers
  - $[1]$, $[1, 2]$, $[1, 4, 17]$ are all valid prefixes
  - $[1]$ labels the 'real' world

- This system allows syntactic recognition of accessibility – although different prefixes may name the same world

Proof Rules for Modal KE (1)

- To prove P, build a KE-tree for $[1] : \neg P$
- Try to construct a model for $\neg P$ such that we are guaranteed to find it, if it exists
- If the process fails, so there is no model for $\neg P$, then P must be true in every model
- In any one world, normal KE inference rules apply, e.g.:

$$\frac{\sigma : p \wedge q}{\begin{array}{c}\sigma : p \\ \sigma : q\end{array}} \qquad \frac{\begin{array}{c}\sigma : p \to q \\ \sigma : p\end{array}}{\sigma : q} \qquad \frac{\sigma : \neg\neg p}{\sigma : p} \qquad \frac{\begin{array}{c}\sigma : p \\ \sigma : p^c\end{array}}{\times} \qquad \begin{array}{cc} & \\ \sigma : p & \sigma : p^c \end{array}$$

- Must be in the same world to close a branch

Proof Rules for Modal KE (2)

- To define proof rules for modal formulas, need the following terminology
  - A prefix σ **occurs** on a branch if it labels a formula on the branch
  - A prefix σ is **available** on a branch if it occurs on a branch
  - A prefix σ is **unrestricted** on a branch if it not an initial segment (strict sub-segment or equals) of any prefix that occurs on the branch
    - Suppose $[1, 2, 3]$ occurs on a branch
    - $[1, 2, 3]$ is available on the branch
    - $[1]$, $[1, 2]$ and $[1, 2, 3]$ are restricted on the branch

- Then the rules are:

$$\frac{\sigma : \Box p}{\sigma n : p} \qquad \frac{\sigma : \neg\Diamond p}{\sigma n : \neg p} \qquad \frac{\sigma : \Diamond p}{\sigma n : p} \qquad \frac{\sigma : \neg\Box p}{\sigma n : \neg p}$$

for any available σn for σn unrestricted

- Global Rule: any branch can be extended with $\sigma : F$, for any available σ and instance F of the characteristic formula for the specific modal logic

Example

1	$[1]:$	$\neg(\Box(p \to q) \to (\Box p \to \Box q))$	$\neg$ conc
2	$[1]:$	$\Box(p \to q)$	α, 1
3	$[1]:$	$\neg(\Box p \to \Box q)$	α, 1
4	$[1]:$	$\Box p$	α, 3
5	$[1]:$	$\neg\Box q$	α, 3
6	$[1,1]:$	$\neg q$	$\Diamond$, 5
7	$[1,1]:$	$p \to q$	$\Box$, 2
8	$[1,1]:$	p	$\Box$, 4
9	$[1,1]:$	q	β, 7, 8
		$\times$	close, 6, 9

Counter Example

1	$[1]:$	$\neg((\Box p \to \Box q) \to \Box(p \to q))$	$\neg$ conc
2	$[1]:$	$\Box p \to \Box q$	α, 1
3	$[1]:$	$\neg\Box(p \to q)$	α, 1
4	$[1,1]:$	$\neg(p \to q)$	$\Diamond$, 3
5	$[1,1]:$	p	α, 4
6	$[1,1]:$	$\neg q$	α, 4

7	$[1]:$	$\Box p$	PB1
9	$[1]:$	$\Box q$	β, 2, 7
10	$[1,1]:$	q	$\Box$, 9
	$\times$	close, 6, 9	

8	$[1]:$	$\neg\Box p$	PB2
11	$[1,2]:$	$\neg p$	$\Diamond$, 8
	open		

- Can now build a counter-model
  - $W = \{[1], [1,1], [1,2]\}$
  - $R = \{([1], [1,1]), ([1], [1,2])\}$
  - $\mid p \mid = \{[1,1]\}$, $\mid \neg q \mid = \{[1,1]\}$

General Proof System

- Reflect the condition(s) on the accessibility relation R by accessibility conditions on prefixes
  - A prefix satisfies the —
    - **general** condition if σn is accessible from σ, for every integer n
    - **reverse** condition if σ is accessible from σn, for every integer n
    - **identity** condition if σ is accessible from itself
    - **transitive** condition if σ is accessible from τ, and τ is a strict initial segment of σ
    - **universal** condition if σ is accessible from τ, every σ and every τ

- Idea originates with Fitting (1993)

- Generalised by Pitt and Cunningham (1996) to include empty sequences, sequences with variables, and sequence unification, to provide a modal KE theorem prover for all 15 normal modal logics

Proof Rules in the General System

- For modal logic **K**, the rules remain the same
- For modal logic **S5** (only), the rules can be simplified

$$\frac{i : \Box p}{j : p} \qquad \frac{i : \neg\Diamond p}{j : \neg p} \qquad \frac{i : \Diamond p}{j : p} \qquad \frac{i : \neg\Box p}{j : \neg p}$$

for any available integer j for j new to the branch

- For the other logics, see Pitt and Cunningham (1996)

KE for Predicate Logic

- Unlike in propositional logic
  - Models can be infinite
  - There may be an infinite number of models
- So we have to use proof systems
- Extend KE proof rules for analysing quantified formulas
  - Notation: write $A_{\{c/x\}}$ for the formula resulting from replacing every occurrence of the variable x with the constant term c

γ rule $\dfrac{\forall x.A}{A_{\{c/x\}}}$ $\dfrac{\neg\exists x.A}{\neg A_{\{c/x\}}}$ for any constant c

δ rule $\dfrac{\exists x.A}{A_{\{c/x\}}}$ $\dfrac{\neg\forall x.A}{\neg A_{\{c/x\}}}$ for a constant c new to the branch

Example

- Consider the syllogism
  - No reptile has fur.
  - All snakes are reptiles.
  - Therefore: No snake has fur.

- Formally: prove using **KE** for predicate logic

- $\{\neg\exists x.reptile(x) \wedge furry(x), \forall y.snake(y) \rightarrow reptile(y)\}$
  $\models$
  $\neg\exists z.snake(z) \wedge furry(z)$

KE Proof

1	premise	$\neg\exists x.reptile(x) \wedge furry(x)$
2	premise	$\forall y.snake(y) \rightarrow reptile(y)$
3	¬conc	$\neg(\neg\exists z.snake(z) \wedge furry(z))$
4	dne, 3	$\exists z.snake(z) \wedge furry(z)$
5	$\exists, 4$	$snake(c) \wedge furry(c)$
6	$\alpha, 5$	$snake(c)$
7	$\alpha, 5$	$furry(c)$
8	$\forall, 2$	$snake(c) \rightarrow reptile(c)$
9	$\beta, 8, 6$	$reptile(c)$
10	$\forall, 1$	$\neg(reptile(c) \wedge furry(c))$
11	$\beta, 10, 9$	$\neg furry(c)$
	close, 7,11	

Implementation Routes

- Chief practical difficulty in implementing KE for predicate logic comes from the γ-rule

- In domains where all the constants are known, could use (a variation of) the 'Hintikka' rules
  - γ-rule: extend the branch with $A_{\{c/x\}}$ for every constant c
  - δ-rule: open n alternative branches with $A_{\{c/x\}}$ for each constant c

- The general problem is that universal quantifier rule can be used as often as is liked, and predicate logic is **semi-decidable**
  - If the formula is unsatisfiable, the proof procedure will stop with a closed tableau
  - If the formula is unsatisfiable, the proof procedure might stop with an open branch and all formulas analysed, or it might extend infinitely

- Various solutions, including free variables (but generally it's just nasty)

KE for Predicate Modal Logic

- This slide intentionally left blank
- Here be dragons
- And various other flimsy excuses too numerous to mention

Summary and Conclusions

- Introduced the calculus KE, a sound and complete proof procedure for automated reasoning in propositional, predicate and modal logics

- Been the basis of several implemented systems, including $\Box$KE, a theorem prover for all 15 normal modal logics, and WinKE, a windows-based pedagogic tool for teaching logic and reasoning