AUTOMATED REASONING

SLIDES 2:

PROPOSITIONAL TECHNIQUES (2)

Model Generation Procedure
Solving the "Three Little Girls" Problem
OBDDs (Ordered Binary Decision Diagrams)

KB - AR - 09

Model Generation Method

The Model Generation (MG) method is an easy method for humans to apply, but it has not been so widely implemented as DP.

Similar to DP, MG returns True if given clauses are **satisfiable**.

MG constructs a tree in which each branch tries to maintain a partial model of the initial clauses (but in a different and more constrained way to DP).

Branches of MG trees contain only atoms: the atoms in a branch give a partial model. Branches that cannot be extended to a full model are called *closed*.

eg Given a branch with atoms A and B and clauses ¬A∨C, ¬A∨¬B

We could extend the branch with atom C, since any model that makes A true will be forced to make C true because of ¬A∨C

However, better we can close the branch since no model that makes A and B true can make ¬A∨¬B true.

Branches that are not yet closed are called *open*.
A branch is *completed* if all initial clauses have been considered by the branch.

MG is related to the tableau method, which we study later.

Model Generation Procedure

procedure MG(S,B): boolean

%S are clauses still to make true and B is the branch (model) so far

1. If no clause in S contains a negative literal then return true.
(B is a partial model of the initial clauses - see Note 1 to make B a model)

2. If S contains a clause C with ≥1 positive literals and is such that all negative literals ¬L in C (if any) satisfy L occurs in B, then return the disjunction of MG(S(A),B+A) , for each positive literal A in C,
where S(A) = S-{X|X in S and X is made true by the assignments in B+A}
(See Note 2)

3. If S contains a clause C with only negative literals and and all literals ¬L in C satisfy L occurs in B, then return false.
(B makes C false - see Note 3)

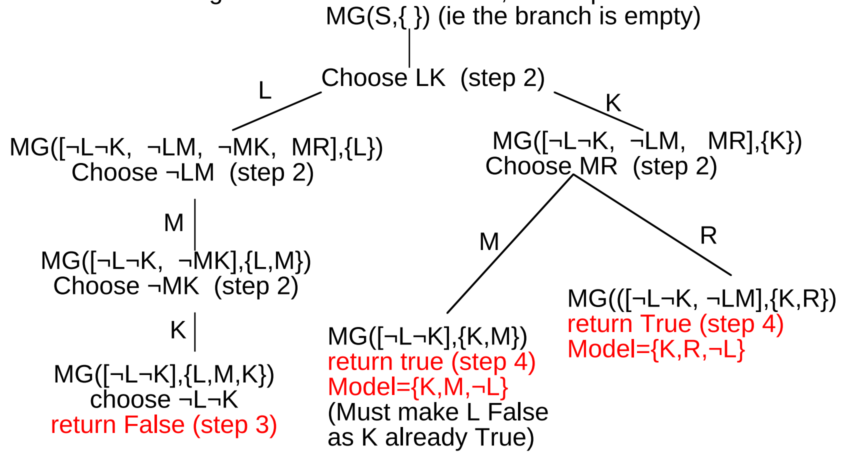
4. If none of 1, 2 or 3 occurs then return true.
(Every remaining clause has ≥1 unassigned negative literal) – See note 4)

MG is initially called with S=given clauses and B= [ ].

(**Note**: Can assume there are no tautologies or subsumed clauses at the start and that all identical literals have been merged.)

Model Generation Example as a tree

Given clauses: LK, ¬L¬K, ¬LM, ¬MK, MR (ie L∨K etc.)
No tautologies or subsumed clauses; not step 1



Return (False or True or True)=True

Notes about MG

procedure MG(S,B): boolean

1. If no clause in S contains a negative literal return true.

Note1: B satisfies all clauses so far removed from the original argument S. B can be made into a model of the remaining clauses by assigning true to any atoms not yet assigned, since they either occur positively in S, or do not occur at all.

2. If C=L∨D (L positive) is in S and for all negative literals ¬L in D L is in B, then return the disjunction of MG(S(A),B+A) , for each positive literal A in C

Note2: The branches of the form B+A extending B are the only ways to make C true. Can remove any clauses in S also made true by B+A

3. If C in S has only negative literals and for all literals ¬L in C L is in B, then return false.

Note3: B cannot be extended to make C true as it already makes C false.

4. If none of 1, 2 or 3 occurs then return true.

Note4: Every remaining clause in S has ≥1 unassigned negative literal ¬L. Assign false to all such L to make the remaining clauses true. Can assign either true or false to any remaining atoms in the language not yet assigned.

Example: {LK, ¬L¬K, ¬LM, ¬MK, MR} (also shown in tree form on 2aiii)

(Step 2) Choose MR (could also have chosen LK instead as on Slide 2aiii).
Return MG([LK,¬L¬K, ¬MK], {M}) **or** MG([LK,¬L¬K, ¬LM, ¬MK], {R});
i.e. start a tree with two branches.
Consider the first disjunct and (Step 2), clause ¬MK.
Return MG([¬L¬K], {M,K}).
ie still to make ¬L∨K true

(Step 4) return true - assign false to L.
Check that {M,K,¬L} is a (partial) model for the initial clauses.

If the second disjunct had been chosen,
then apply (Step 2) and clause L∨K and return
MG([¬L¬K, ¬LM, ¬MK], {R,L}) or MG([¬L¬K, ¬LM], {R,K});
i.e. extend the second branch of the tree by two branches.

The first disjunct will eventually return false (show this yourself)
and the second will return the model {R,K,¬L}.
Check that {R,K,¬L} is a (partial) model for the initial clauses.

You can also try the "three little girls" problem as an exercise.
The MG procedure can be generalised to first order clauses with care.
e.g. use (2) in place of (2a)-(2c). See exercise sheet.

Model Generation Procedure (MG):

Another procedure appropriate for propositional clauses is *model generation* (MG), which is a special case of the tableau method (to be covered in detail later in the course). Although the MG method can be generalised to arbitrary propositional sentences, it then loses some of its simplicity. MG is very easy for humans to use as it is more restricted than DP.

The MG procedure attempts to build partial models. It constructs a tree in which each branch tries to maintain a partial model of the initial sentences (but in a different way to DP).

The tree built by the MG procedure for an initial set of clauses S consists of nodes each labelled by an atom. There are 2 kinds of branches, called *open* and *closed*. The atoms in each open branch B give a model of the set of clauses processed in B; when all of S have been processed in an open branch B then the branch is called *completed* and the nodes in B give rise to a (partial) model of S. A closed branch B is also called completed and indicates that the atoms in B cannot be extended to be a model of S (i.e. they make some clause in S false).

Let closed branches be labelled by false and completed open branches be labelled by true. Then the given clauses S have no model if the disjunction of all labels in a tree in which all branches are completed is false. There is a model if the disjunction is true, as at least one branch must then have been completed and open.

Another simple normalisation procedure for ground clauses is the method of OBDDs (Ordered Binary Decision Diagrams). A related extension is used in model checking programs to obtain neat representations for propositional sentences. In essence it reduce a sentence to an *if-then-else* normal form, from which a model can easily be obtained. If the normal form reduces to false then there is no model. See slides 2d.

Why does MG Work (1)? (An illustration)

Assume clauses S, and a branch B containing positive atoms

The termination cases (1,3,4):

Case 1

All clauses in S have only positive literals

Already processed (ie removed) clauses that are true using atoms in B

$\boxed{\text{Model is: B + remaining atoms in clauses}}$

Remaining clauses will be true by above assignments

Case 3

S includes ¬B1∨ ... ∨ ¬Bn where B1, ..., Bn all in B

$\boxed{\text{S cannot be made true given B}}$

NOTE:
In Cases 1 and 4 may have to extend model to cover all atoms in language. Can do so arbitrarily.

Case 4

All clauses include some ¬Ci, where Ci is not in B
eg ¬Ci ∨ ...

Otherwise would be in Cases 1,2 or 3

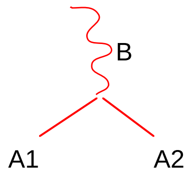
$\boxed{\text{Model is: Atoms in B true; Each Ci is false}}$

Why does MG Work (2)?

Assume clauses S, and a branch B containing positive atoms

Non-terminating case 2

Case 2



S includes
$C = \neg B1 \vee \neg B2 \vee A1 \vee A2$
and B1, B2 in B

C is made true in each of the new branches by extending B

In the branch extended by A1 all clauses in S also including A1 are true and can be removed

Proof that the Model Generation procedure is correct.

We show that, if no atom in B occurs positively in S, then the call MG(S,B) returns true iff B can be extended to give a model of S (**).

Then, since the initial call MG(S,$\emptyset$) clearly satisfies the assumption of (**) (there are no atoms in B=$\emptyset$), we can conclude that the MG procedure returns true iff S has a model.

To show (**) we use induction on *n*, the number of clauses in the argument S.
Case n=0. Step 1 applies and since S is empty and result = true, then (**) will be true whatever B is.

Case n>0.
Assume as Ind. Hyp. (IH) that for any call MG(S,B), where S has fewer than n clauses and such that no atom in B occurs positively in S, that MG(S,B) returns true iff the set of atoms in B can be extended to give a model of the clauses in S.

If Step 1 applies, then can make clauses in S true by assigning true to one atom from each clause in S. This is an extension of B and so the result returned (true) is correct.

If Step 3 applies, then some clause C in S contains only negative literals ¬X, where X is in B. Clearly B cannot be extended to be a model of S, so the result (false) is correct.

Correctness of MG continued:

If Step 4 applies, then every clause in S contains some negative literal ¬X, where X is not in B. B can be extended to be a model of S by making such atoms X false and hence the returned result (true) is correct.

If Step 2 applies for clause C, then MG(S,B)=disjunction of MG(S(A),A+B), for each atom (positive literal) A in C. Note that S(A) satisfies the conditions of (IH) -
(i) it has <n clauses,
(ii) by construction there are no positive occurrences of A in S(A), and
(iii) there are no positive occurrences of atoms in B in S(A) since there were none in S by assumption.

Therefore, by (IH), MG(S(A),A+B) returns the correct result for S(A).

Next we show that this is the correct result for S as well. If, for some A, MG(S(A), A+B) returns true, then MG(S,B) will return true. This is correct, for B can be extended (by assigning true to A) to be a model of all S(A) by assumption, and of all clauses in S but not in S(A), which are clauses containing A. If, for every A, MG(S(A), A+B) returns false, then MG(S,B) will return false. This is also correct since B cannot be extended to make S true. To make S true, C must be true, which requires at least one of the atoms A to be true. By assumption, if the atoms A+B are all true, then the subset S(A) of S cannot be made true. So S cannot be made true by extending B. Therefore, by induction we can conclude (**).

"The three little girls" problem again!

The data

(1) $C(d) \vee C(e) \vee C(f)$	One of the threee girls was the culprit
(2) $C(x) \rightarrow H(x)$	$\{ C(d) \rightarrow H(d), C(e) \rightarrow H(e), C(f) \rightarrow H(f) \}$ To convert into propsitional form
(3) $\neg(C(d) \wedge C(e))$	
(4) $\neg(C(d) \wedge C(f))$	
(5) $\neg(C(f) \wedge C(e))$	Only one of the three girls was the culprit
(6) $C(d) \vee H(d) \vee \neg C(e)$	(Dolly's statement negated)
(7) $C(e) \vee C(f) \vee \neg(C(e) \rightarrow (C(d) \vee H(d)))$	(Ellen's negated)
(8) $C(f) \vee \neg H(d) \vee \neg((H(d) \wedge C(d)) \rightarrow C(e))$	(Frances's negated)

This time we'll try to find a model and return True (and we hope the model will make C(f) true).

So there is no need to include (9) ¬C(f).

We convert to clauses and can remove any tautologies or subsumed clauses at the start. Also merge identical literals

Solution to "The three little girls" by MG

2cii

(1) C(d) ∨ C(e) ∨ C(f) (2a) ¬C(d) ∨ H(d) (2b) ¬C(e) ∨ H(e)
(2c) ¬C(f) ∨ H(f) (3) ¬ C(d) ∨ ¬C(e) (4) ¬ C(d) ∨ ¬C(f)
(5) ¬ C(e) ∨ ¬C(f) (6) C(d) ∨ H(d) ∨ ¬C(e) (7a) C(e) ∨ C(f) ∨ C(e)
(7b) C(e) ∨ C(f) ∨ ¬C(d) (7c) C(e) ∨ C(f) ∨ ¬ H(d) (8a) C(f) ∨ ¬H(d) ∨ H(d)
(8b) C(f) ∨ ¬H(d) ∨ C(d) (8c) C(f) ∨ ¬H(d) ∨ ¬C(e)

Merge literals in (7a) to obtain C(e) ∨ C(f) and remove (8a) (tautology);
(7a) subsumes (7b), (7c), (1);

call MG([2a, 2b, 2c, 3, 4, 5, 6, 7a, 8b, 8c], []);
Apply (Step 2) on 7a (C(e) ∨ C(f)); evaluate
MG([2a, 2b, 2c, 3, 4, 5, 6, 8b,8c], [C(e)]) (i) **or** MG([2a,2b,2c,3,4,5,6], [C(f)]) (ii);

In (i): apply (Step 2) to 2b and call MG([2a,2c,3,4, 5, 6,8b, 8c], [H(e), C(e)]);
Apply (Step 2) to 6 and evaluate
MG([2a,2c,3,4, 5,8c], [C(d), H(e), C(e)]) **or** MG([2c,3,4,5,8b, 8c], [H(d),H(e), C(e)]);
Both eventually return false.

In (ii): apply (Step 2) to (2c) and call MG([2a, 2b, 3, 4, 5, 6], [H(f), C(f)]);
Apply (Step 4) to return true; get the partial model [H(f), C(f), ¬C(d), ¬C(e)].
H(e) and H(d) can be either true or false.

Correctness Theorem for MG

2ciii

MG(S,B) halts with *true* if S+B has a model and it returns a partial model which includes B (ie makes atoms in B true).

This partial model can be extended to a complete model for S+B (see procedure).

MG(S,B) halts with *false* if S +B has no models.

Questions:

Can you think of possible “tricks” to enable MG to be implemented efficiently?

How could MG be extended to use general propositional sentences?

What can you do with pure literals in clauses?

What about first order clauses?

Ordered Binary Decision Diagrams (OBDDs)

2di

- The Davis Putnam method processes the given clauses *top - down* ;
- The initial set of clauses is gradually simplified, until either the set becomes empty or contains the empty clause.
- This allows simplifying steps (e.g. subsumption) to be performed on the whole set.

- A different approach is to process the given set of sentences *bottom - up*.
- Each sub-sentence is turned into an IF-THEN-ELSE triple and triples are combined.

OBDDs are based on the If –Then – Else connective:

- A ∧ B ≡ If A Then B Else false
- A ≡ If A Then true Else false
- A → B ≡ If A Then B Else true
- A ∨ B ≡ If A Then true Else B
- ¬ A ≡ If A Then false Else true
- A ↔B ≡ If A Then B Else (¬B)

OBDDs are *ordered* – the test literals always occur in some order (using a single total ordering, for all sub-sentences in some OBDD). Combining two OBDDs :

If A Then X Else Y ∧/∨ If A Then W Else Z ≡ If A Then X ∧/∨ W Else Y ∧/∨ Z
If A Then X Else Y ∧ B (A not tested in B) ≡ If A Then X ∧/∨ B Else Y ∧/∨ B, etc.
If A Then X Else X ≡ X (A simplification step)

You can work out the combinations for -> and <-> yourself.
By storing OBDDs in a table, duplications can be obtained by lookup.

<u>e.g. OBDD for X ∧ (Y ∨ ¬ Z):</u>

2dii

Y ∨ ¬Z= (If Y Then true Else false) ∨ (If Z Then false Else true) =

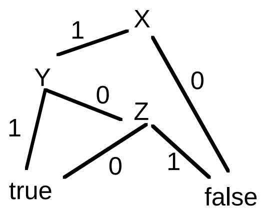
If Y Then (true ∨ (If Z Then false Else true)) Else (false or (If Z Then false ....) =

If Y Then true Else (If Z Then false Else true)

X ∧ (Y ∨ ¬ Z) =

If X Then (true and If Y …)
  Else (false and If Y …) =

If X Then (If Y Then true
      Else (If Z Then false Else true))
  Else false



See Moore, JAR, 1994
Also, E. Clark, Bryant.
Ed Clark recently won the Turing Award

OBDD Example: [LK, ¬L¬K, ¬LM, ¬MK, MR]
(All clauses anded together; t is true, f is False;
atoms tested alphabetically; If X Then Y Else Z represented as if(X,Y,Z))

(1) LK = if(K, t, IF(L , t , f))
(2) ¬L¬K = if(K, IF(L , f , t) , t)
(3) ¬LM = if(L, IF(M ,t, f) , t)
(4) ¬MK = if(K , t , IF(M ,f ,t))
(5) MR = if(M , t , IF(R, t , f))

(1) $\wedge$ (2): if(K, IF(L, f, t), IF(L, t, f))

$\wedge$ (4): if(K, if(L, f, t), if(L, t, f) $\wedge$ if(M, f, t)) = if(K, if(L, f, t), if(L, if(M, f, t), f))

$\wedge$ (3): if(K, if(L, f,t) $\wedge$ (3), if(L, if(M, f, t) $\wedge$ if(M, t, f), f))
= if(K, if(L, f, t) , if(L, if(M, f, f), f))
= if(K, if(L, f, t) , if(L, f, f)) = if(K, if(L, f, t) , f)

$\wedge$ (5): if(K, if(L, f, t) $\wedge$ if(M, t, if(R, t, f)) , f)
= if(K, if(L, f, if(M, t, if(R, t, f))), f)

Gives models [K, ¬ L, M] or [K, ¬L, ¬M, R]
Can you see how to read off the model from the final expression?

Exercise: Draw the graphical form of the final result (Answer on next slide).

Correctness Theorem for OBDD

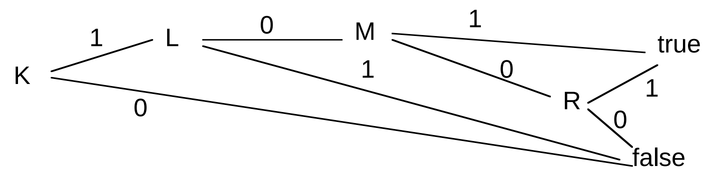
OBDD(S) ≡ S

hence if OBDD(S) = f (no arcs enter True) then S has no models;

otherwise OBDD(S) ≠f and S has a model that can be read from the tree.
(Simply follow the decisions from the root to the True node – if there are several paths then there are several models);

if OBDD(S)=t then S is a tautology.

Graphical form of if(K, if(L, f, if(M, t, if(R, t, f))), f)



Summary of Slides 2

1. A second method for testing satisfiability of propositional clauses was described; the Model Generation (MG) procedure is good for humans, but less well investigated in optimised implementations.

2. MG returns True (and a partial model) for given clauses S if there is one. It returns False if there are no models of S. The returned model (if any) can be extended to all atoms by assigning T or F to unasigned atoms.

3. MG can be extended to first order clauses (see later).

4**.** The state of MG can be represented as a tree, in which each node N is labelled by the current set of clauses S (those still to be made True in the branch B ending at N). The arcs of the tree are labelled by atoms and the atoms labelling arcs in B are a partial assignment that satisfy those clauses in the initial set which are not still in S (for branch B). This property is maintained as an invariant.

5. MG was used to solve the “Three Little Girls” problem.

6. The correctness property of MG is proved by induction. The induction is on the number of clauses in the current clause set.

7. The OBDD (Ordered Binary Decision Tree) was introduced.

8. An informal method for generating the OBDD form of a proposition using simplification rules was given.

9. An OBDD can be represented as a tree, in which each node is labelled by an atom, the one “decided” at that node. A given order of atoms is used in any particular problem. The arcs in a branch are labelled by T or F. Two special nodes are True and False.

For a given proposition S:
- if no arcs in the OBDD(S) enter False then S is equivalent to true (i.e. is a tautology);
- if at least one arc in OBDD(S) enters True then S has a model that can be read from the tree;
- if no arcs in the OBDD(S) enter True then S is equivalent to false and has no models.

10. The correctness property of OBDD relies on the preservation of equivalence by the simplication rules. Various optimisations are possible.

Some more questions

Write a prolog program to generate OBDDs (bottom up).
Represent them by a triple (test literal, true case, false case)

Where is the best place in the literal order to consider pure literals?

Question to think about for next week

The “3 girls” problem is not propositional, but we used DP to solve it, as the set of ground instances of the given clauses was not too large.

Suppose the set of ground instances is large (or even very large - infinite!)

One approach is to hope the data is Horn clauses, as you can then use Prolog (perhaps). But if the data is not Horn clauses?

Any ideas?

A Solution to Aunt Agatha's Burglary

Note: (x and y are implicitly universally quantified)

(1) $s(m,a)$ (someone stole from Agatha)
(2) $x=a \lor x=b \lor x=j$ (The only people are Agatha, James and the butler)
(3) $s(x,y) \rightarrow (d(x,y) \land \neg r(x,y))$
(thieves dislike, and are not richer than, their victims)
(4) $d(a,x) \rightarrow \neg d(j,x)$ (James dislikes no-one whom Agatha dislikes)
(5) $\neg x= b \rightarrow d(a,x)$ (Agatha dislikes all but the butler)
(6) $\neg r(x,a) \rightarrow d(b,x)$ (butler dislikes anyone not richer than Agatha)
(7) $d(a,x) \rightarrow d(b,x)$ (and also anyone Agatha dislikes)
(8) $\neg(\forall z\ d(x,z))$ (No-one dislikes everyone)
(9) $\neg a=b$
(10) Conclusion: $s(a,a)$ (Agatha burgled herself)

First simplify (8) to $\exists z \neg d(x,z)$ and then to $\neg d(x, f(x))$.
(For each x, f(x) is a person x doesn't dislike)

Put x as m in (2); m=b and m=j will lead to contradictions forcing m=a.

m=j: $s(j,a)$ (by equality substitution in (1)) and hence $d(j,a)$ and $\neg r(j,a)$ from (3);
From (5) and (9) $d(a,a)$ hence $\neg d(j,a)$ by (4);
Contradiction so $m\neq j$.

m=b: $\neg d(b,f(b))$ (*) (put x as b in (8));
hence $\neg d(a,f(b))$ by (7) and $\neg \neg f(b)=b$ by (5);
Therefore $f(b) = b$ and $\neg d(b,b)$ from (*);
$s(b,a)$ (by equality substitution in (1)) and hence $d(b,a)$ and $\neg r(b,a)$ from (3);
Hence $d(b,b)$ from (6);
Contradiction so $m\neq b$;

Therefore m=a;
$s(a,a)$ (by equality substitution in (1)) which is the conclusion.

(See slides further in the course for other approaches.)

A (Natural) Solution to the "three little girls"

(1) $C(d) \lor C(e) \lor C(f)$
(2) $C(x) \rightarrow H(x)$ (x is implicitly universally quantified)
(3) $\neg(C(d) \land C(e))$
(4) $\neg(C(d) \land C(f))$
(5) $\neg(C(f) \land C(e))$
(6) $C(d) \lor H(d) \lor \neg C(e)$ (Dolly's statement negated)
(7) $C(e) \lor C(f) \lor \neg(C(e) \rightarrow (C(d) \lor H(d)))$ (Ellen's negated)
(8) $C(f) \lor \neg H(d)$ or $\neg((H(d) \land C(d)) \rightarrow C(e))$ (Frances's negated)

By (1) it must have been C(d), C(e) or C(f).

Case 1: C(d). (Suppose Dolly did it.)
(7) cannot then be true as all 3 parts lead to a contradiction.
So Dolly is not the culprit.

($\neg(X \rightarrow Y)$ true means X true and Y false.)

Case 2: C(e) Suppose Ellen did it. Then H(d) follows from(6)
(remember $\neg C(d)$ from Case 1);
then (8) leads to contradiction.
So Ellen is not the culprit.

Hence Frances was the culprit. (See slides for other approaches.)

A Solution to the Mathematical problem

(1) $a \circ b = c$
(2) $\circ$ is an associative operator: $x \circ (y \circ z) = (x \circ y) \circ z$
(3) $x \circ x = e$
(4) $x \circ e = e \circ x = x$ (e is the identity of $\circ$)

Show $b \circ a = c$

$x \circ (a \circ b) = (x \circ a) \circ b$	(put y as a and z as b in (2))
$x \circ c = (x \circ a) \circ b$	(use $a \circ b = c$)
$c \circ c = (c \circ a) \circ b$	(put x as c)
$e = (c \circ a) \circ b$	(use (3) $c \circ c = e$)
$e \circ b = e \circ b$	(property of =)
$e \circ b = ((c \circ a) \circ b) \circ b$	(use $e = (c \circ a) \circ b$)
$b = ((c \circ a) \circ b) \circ b$	(use (4) $e \circ b = b$)
$b = (c \circ a) \circ (b \circ b)$	(use (2))
$b = (c \circ a) \circ e$	(use (3) $b \circ b = e$)
$b = c \circ a$	(use (4) $(c \circ a) \circ e = c \circ a$)

(See slides near the end of the course for other approaches.)