15-780: Graduate AI
Lecture 1. Logic

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman

Logic

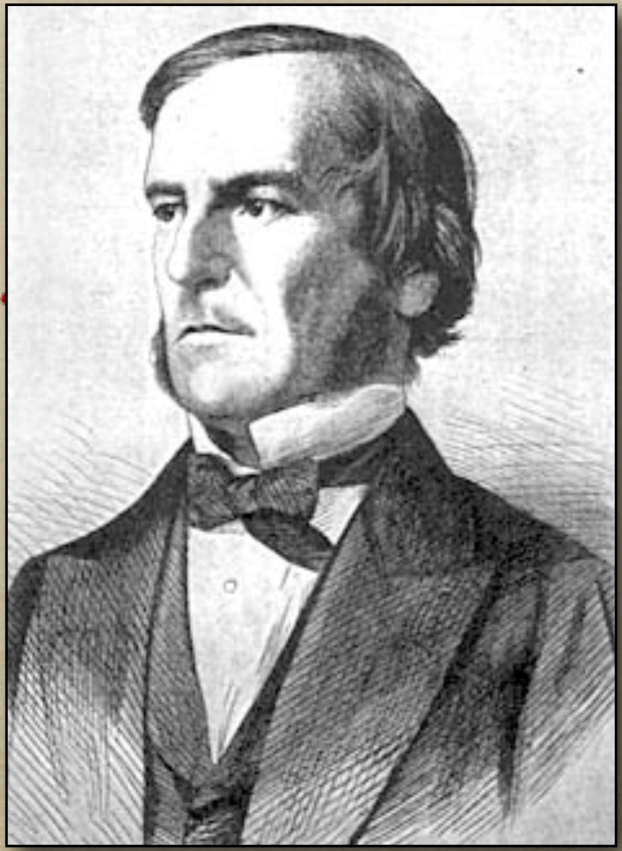
Why logic?

- *Search: can compactly write down, solve problems like Sudoku*
- *Reasoning: figure out consequences of the knowledge we've given our agent*
- *…and, logical inference is a special case of probabilistic inference*

Propositional logic

- *Constants: T or F*
- *Variables: x, y (values T or F)*
- *Connectives:* $\wedge$, $\vee$, $\neg$
  - *Can get by w/ just NAND*
  - *Sometimes also add others:*

    $\oplus$, $\Rightarrow$, $\Leftrightarrow$, …



George Boole
1815–1864

Propositional logic

- *Build up expressions like* $\neg x \Rightarrow y$
- *Precedence:* $\neg$, $\wedge$, $\vee$, $\Rightarrow$
- *Terminology: variable or constant with or w/o negation =* ***literal***
- *Whole thing =* ***formula*** *or* ***sentence***

Expressive variable names

- *Rather than variable names like x, y, may use names like "rains" or "happy(John)"*
- *For now, "happy(John)" is just a string with no internal structure*
  - *there is no "John"*
  - $happy(John) \Rightarrow \neg happy(Jack)$ *means the same as* $x \Rightarrow \neg y$

But what does it mean?

- *A formula defines a mapping*

  (assignment to variables) $\mapsto$ *{T, F}*

- *Assignment to variables =* ***model***

- *For example, formula* $\neg x$ *yields mapping:*

truth table

x	$\neg x$
T	*F*
F	*T*

model

More truth tables

x	y	$x \wedge y$
T	T	T
T	F	F
F	T	F
F	F	F

model

x	y	$x \vee y$
T	T	T
T	F	T
F	T	T
F	F	F

Truth table for implication

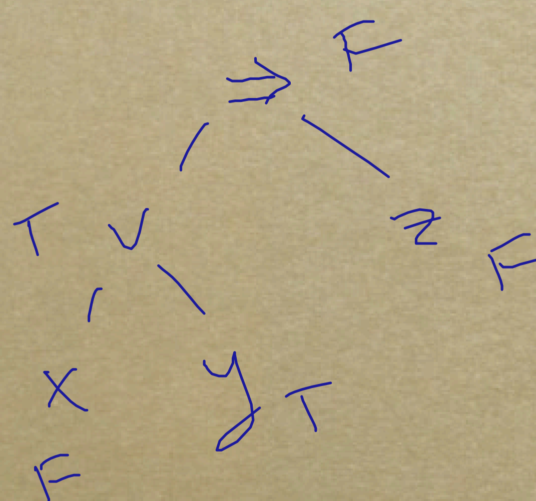
- *$(a \Rightarrow b)$ is logically equivalent to $(\neg a \vee b)$*
- *If a is True, b must be True too*
- *If a False, no requirement on b*
- *E.g., "if I go to the movie I will have popcorn": if no movie, may or may not have popcorn*

a	b	$a \Rightarrow b$
T	T	T
T	F	F
F	T	T
F	F	T

Complex formulas

- *To evaluate a bigger formula*
  - $(x \lor y) \land (x \lor \neg y)$ *when* $x = F, y = F$
- *Build a parse tree*
- *Fill in variables at leaves using model*
- *Work upwards using truth tables for connectives*

Another example



$$(x \vee y) \Rightarrow z$$

x = F, y = T, z = F

Questions about models and sentences

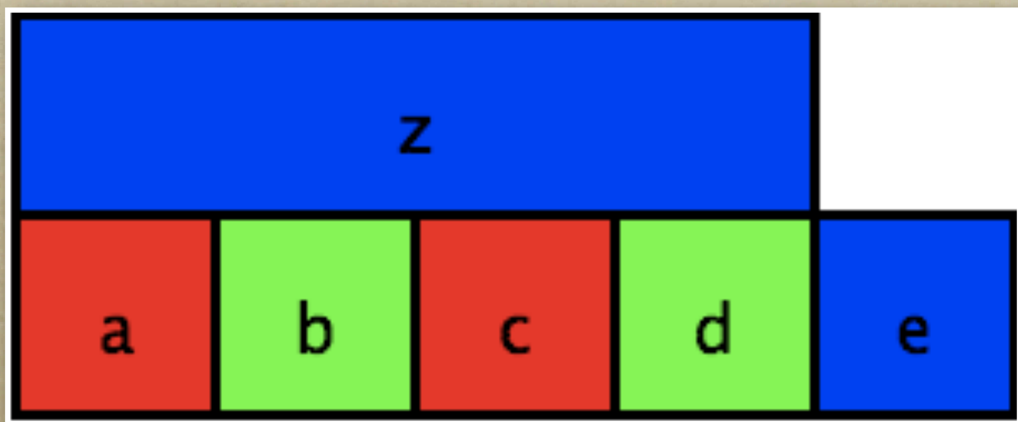
- *How many models make a sentence true?*
  - *Sentence is **satisfiable** if true in some model (famous NP-complete problem)*
  - *If not satisfiable, it is a **contradiction** (false in every model)*
  - *A sentence is **valid** if it is true in every model (called a **tautology**)*

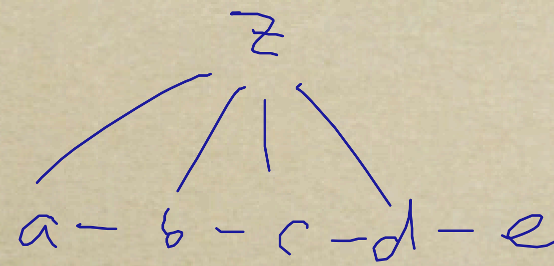
Questions about models and sentences

- *How is the variable X set in {some, all} satisfying models?*
- *This is the most frequent question an agent would ask: given my assumptions, can I conclude X? Can I rule X out?*
- *SAT answers all the above questions*

Bigger Examples

3-coloring





Vars: aR, aG, aB, bR, bG, bB .....

$(aR \vee aG \vee aB) \wedge (bR \vee bG \vee bB) \wedge \dots$

$(\overline{aR} \vee \overline{bR}) \wedge (\overline{aG} \vee \overline{bG}) \wedge \dots$

Sudoku

SuDoku Puzzle

		6	3			4	7	
		5	8		7			
1							2	3
	6		1	9				
4	9							
						1	9	8
6					3	5		
		8		5				2
	7	4			6		8	

http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html

Constraint satisfaction problems

$(aR \vee aB \vee aG) \wedge$
$(bR \vee bB \vee bG) \wedge \ldots$
$\wedge (\overline{aR} \vee \overline{bR}) \wedge (\overline{aB} \vee \overline{bB}) \wedge (\overline{aG} \vee \overline{bG})$
$\wedge (\overline{aR} \vee \overline{zR}) \ldots$

- *Like SAT, but:*
  - *variable domains are arbitrary (vs. TF)*
  - *complex constraints (vs. $a \vee b \vee \neg c$)*
- *Sudoku: "at most one 3 in row 5"*
- *Can translate SAT ⇔ CSP*
  - *often CSP more compact*

Minesweeper

0	0	1	v1		
0	0	1	v2		
0	0	1	v3		
1	1	2	v4		
v8	v7	v6	v5		

V = { v1 , v2 , v3 , v4 , v5 , v6 , v7 , v8 }, D = { B (bomb) , S (space) }

C = { (v1,v2) : { (B , S) , (S,B) } ,(v1,v2,v3) : { (B,S,S) , (S,B,S) , (S,S,B)},...}

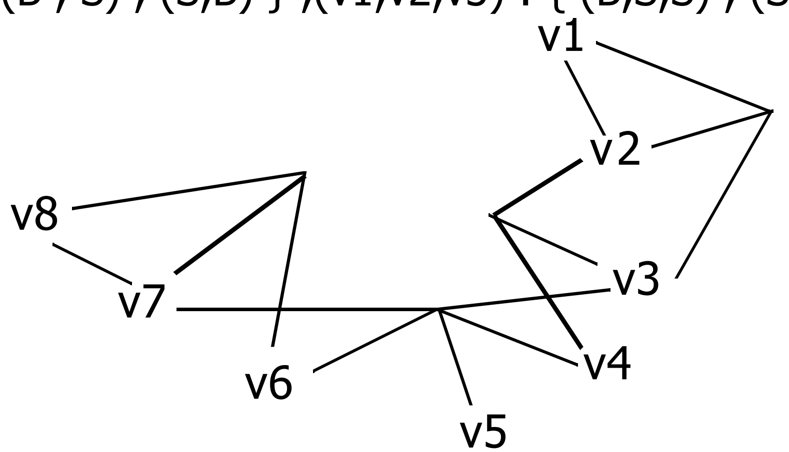


image courtesy Andrew Moore

Propositional planning

init: have(cake)

goal: have(cake), eaten(cake)

eat(cake):

pre: have(cake)

eff: -have(cake), eaten(cake)

bake(cake):

pre: -have(cake)

eff: have(cake)

fluents

actions

Other important logic problems

- *Scheduling (e.g., of factory production)*
- *Facility location*
- *Circuit layout*
- *Multi-robot planning*

Handling uncertainty

- *Minesweeper: what if no safe move?*
- *Say each mine initially present w/ prob p*
- *Common situation: independent "Nature" choices, deterministic rules thereafter*
- *Logic represents deterministic rules ⇒ use logical reasoning as subroutine*

	⊗	1			1			1	
1	1	1	1	1	1	1	1	1	

Handling uncertainty

- *Minesweeper: what if no safe move?*
- *Say each mine initially present w/ prob p*
- *Common situation: independent "Nature" choices, deterministic rules thereafter*
- *Logic represents deterministic rules ⇒ use logical reasoning as subroutine*

	⊗	1		⊗	1		⊗	1	
1	1	1	1	1	1	1	1	1	

Handling uncertainty

- *Minesweeper: what if no safe move?*
- *Say each mine initially present w/ prob p*
- *Common situation: independent "Nature" choices, deterministic rules thereafter*
- *Logic represents deterministic rules* $\Rightarrow$ *use logical reasoning as subroutine*

⊗		1	⊗		1	⊗		1	⊗
1	1	1	1	1	1	1	1	1	

Handling uncertainty

- *Minesweeper: what if no safe move?*
- *Say each mine initially present w/ prob p*
- *Common situation: independent "Nature" choices, deterministic rules thereafter*
- *Logic represents deterministic rules ⇒ use logical reasoning as subroutine*

⊗		1	⊗		1	⊗		1	
1	1	1	1	1	1	1	1	1	⊗

Working with formulas

Truth tables get big fast

x	y	z	$(x \vee y) \Rightarrow z$
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

Truth tables get big fast

x	y	z	a		$(x \vee y \vee a) \Rightarrow z$
T	T	T	T		
T	T	F	T		
T	F	T	T		
T	F	F	T		
F	T	T	T		
F	T	F	T		
F	F	T	T		
F	F	F	T		
T	T	T	F		
T	T	F	F		
T	F	T	F		
T	F	F	F		
F	T	T	F		
F	T	F	F		
F	F	T	F		
F	F	F	F		

Definitions

- *Two sentences are **equivalent**, A ≡ B, if they have same truth value in every model*
  - *(rains ⇒ pours) ≡ (¬rains ∨ pours)*
  - *reflexive, transitive, symmetric*
- ***Simplifying** = transforming a formula into a simpler, equivalent formula*

Transformation rules

$$(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land$$
$$(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor$$
$$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land$$
$$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor$$
$$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land$$

α, β, γ are arbitrary formulas

More rules

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$

α, β are arbitrary formulas

Still more rules…

- *…can be derived from truth tables*
- *For example:*
  - $(a \vee \neg a) \equiv True$
  - $(True \vee a) \equiv True$ ~~*(T elim)*~~ → $T \wedge a \equiv a$
  - $(False \wedge a) \equiv False$ ~~*(F elim)*~~ → $F \vee a \equiv a$

Example

$$(a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg(b \vee c) \vee \neg a)$$

$$a \vee (\neg b \vee \neg c)$$

$$\neg (b \wedge c)$$

$$\neg (b \wedge c) \wedge (a \vee \neg a)$$

$$\top$$

$$\neg (b \wedge c)$$

Normal Forms

Normal forms

- *A normal form is a standard way of writing a formula*
- *E.g., **conjunctive normal form** (CNF)*
  - *conjunction of disjunctions of literals*
  - $(x \vee y \vee \neg z) \wedge (x \vee \neg y) \wedge (z)$
  - *Each disjunct called a **clause***
- *Any formula can be transformed into CNF w/o changing meaning*

CNF cont'd

happy(John) ∧
(¬*happy(Bill)* ∨ *happy(Sue)*) ∧
man(Socrates) ∧
(¬*man(Socrates)* ∨ *mortal(Socrates)*)

- *Often used for storage of knowledge database*
  - *called **knowledge base** or KB*
- *Can add new clauses as we find them out*
- *Each clause in KB is separately true (if KB is)*

Another normal form: DNF

- *DNF = disjunctive normal form = disjunction of conjunctions of literals*
- *Doesn't compose the way CNF does: can't just add new conjuncts w/o changing meaning of KB*

(rains ∨ pours) ∧ (¬pours ⇒ fishing)

pours ∨ fishing

$$rains \land (p \lor f) \lor (p \lor (p \land f))$$

$$r \land p \lor r \land f \lor \underbrace{p \land p}_{p} \lor \underbrace{p \land p \land f}_{p \land f}$$

Transforming to CNF or DNF

- *Naive algorithm:*
  - *replace all connectives with* ∧∨¬
  - *move negations inward using De Morgan's laws and double-negation*
  - *repeatedly distribute over* ∧ *over* ∨ *for DNF (*∨ *over* ∧ *for CNF)*

Example

- *Put in CNF:*

  $(a \vee \neg c) \wedge \neg(a \wedge b \wedge d \wedge \neg e)$

  $(\bar{a} \vee \bar{b} \vee \bar{d} \vee e)$

Discussion

- *Problem with naive algorithm: it's exponential! (Space, time, size of result.)*
- *Each use of distributivity can almost double the size of a subformula*

A smarter transformation

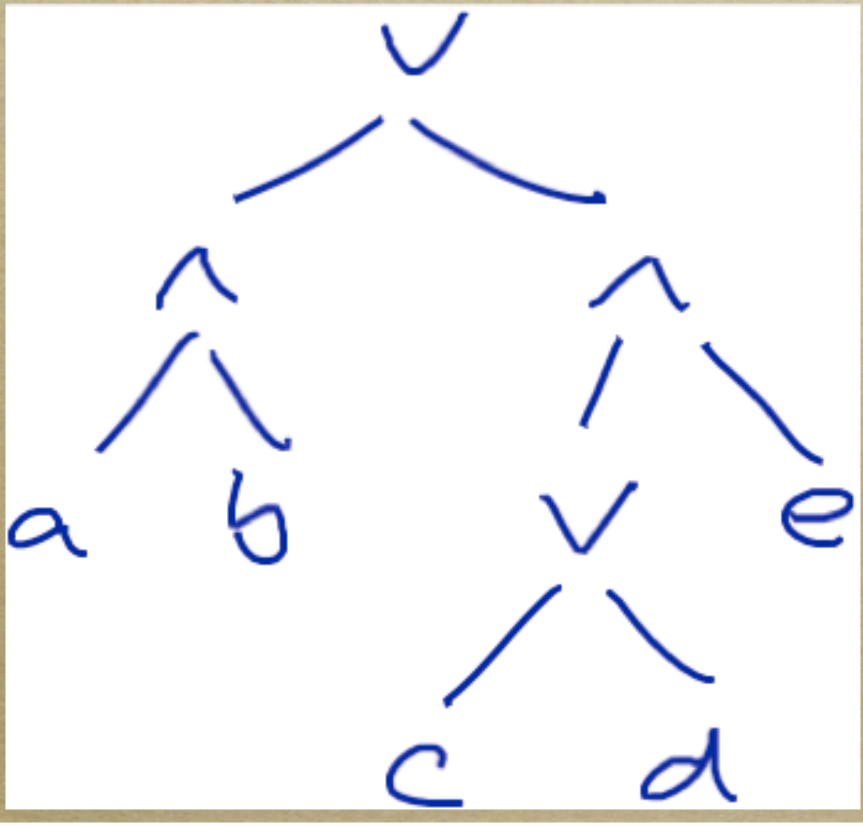
- *Can we avoid exponential blowup in CNF?*
- *Yes, if we're willing to introduce new variables*
- *G. Tseitin. On the complexity of derivation in propositional calculus. Studies in Constrained Mathematics and Mathematical Logic, 1968.*

Tseitin example

- *Put the following formula in CNF:*

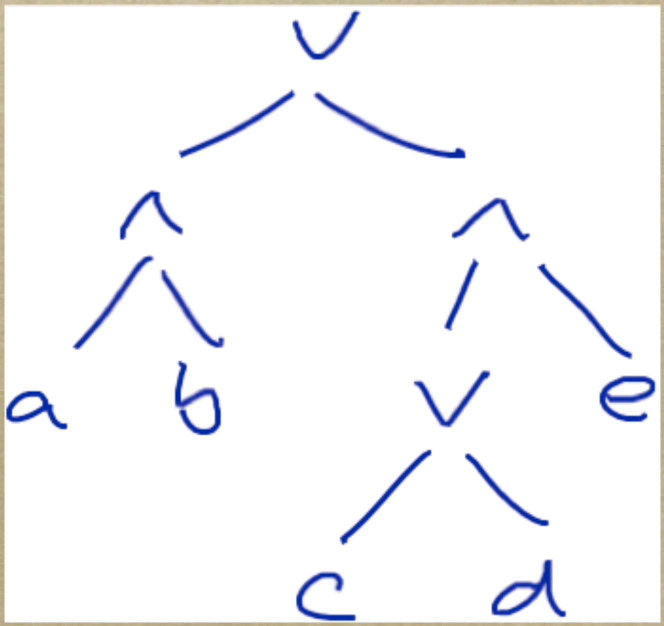
  $(a \wedge b) \vee ((c \vee d) \wedge e)$

- *Parse tree:*



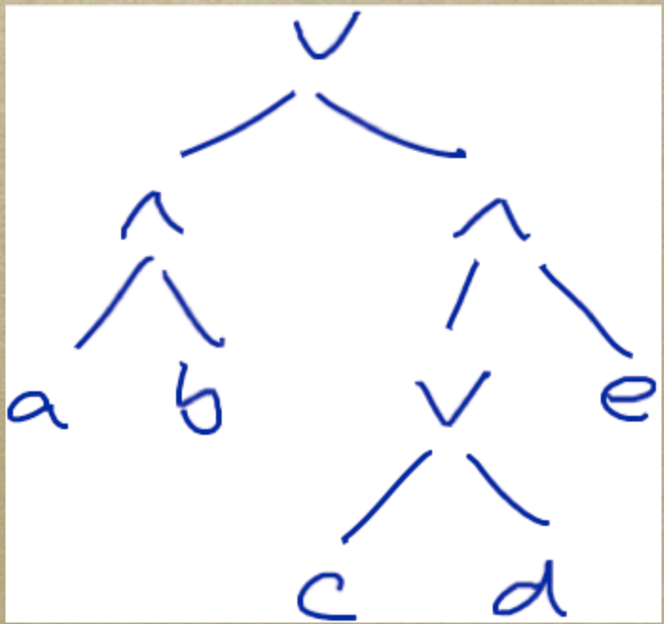
Tseitin transformation

- *Introduce temporary variables*
  - $x = (a \wedge b)$
  - $y = (c \vee d)$
  - $z = (y \wedge e)$



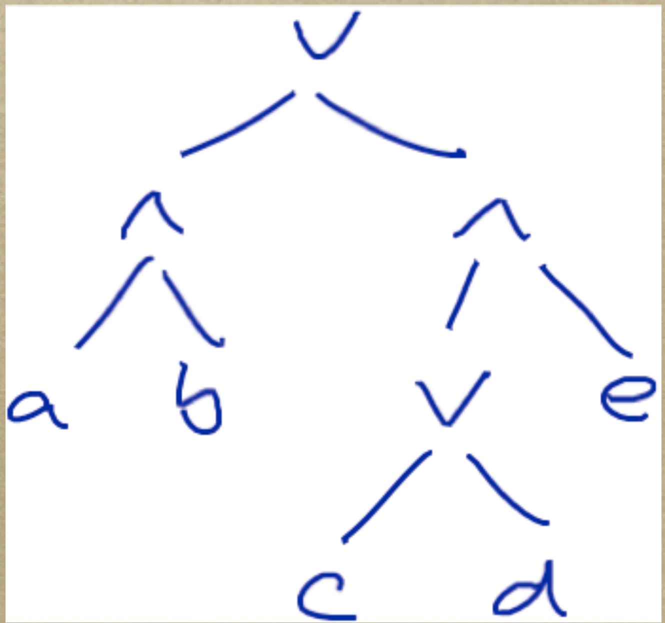
Tseitin transformation

- *To ensure* $x = (a \wedge b)$*, want*
  - $x \Rightarrow (a \wedge b)$
  - $(a \wedge b) \Rightarrow x$



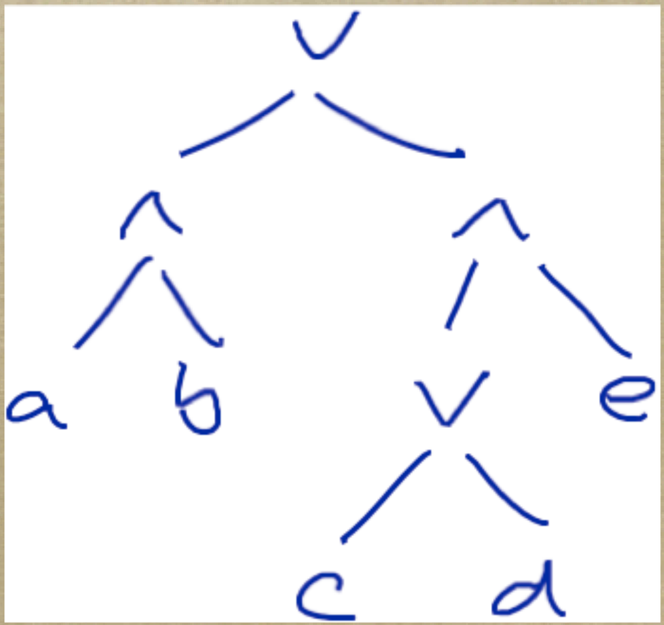
Tseitin transformation

- $x \Rightarrow (a \wedge b)$
- $(\neg x \vee (a \wedge b))$
- $(\neg x \vee a) \wedge (\neg x \vee b)$



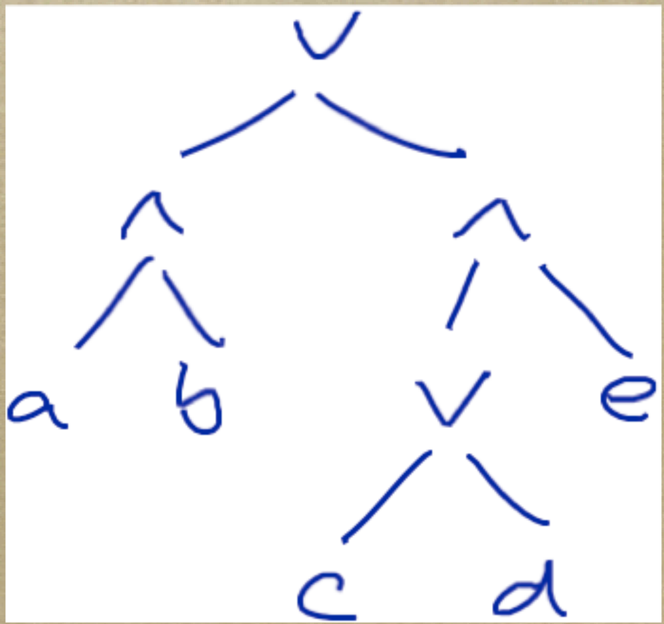
Tseitin transformation

- $(a \wedge b) \Rightarrow x$
- $(\neg(a \wedge b) \vee x)$
- $(\neg a \vee \neg b \vee x)$



Tseitin transformation

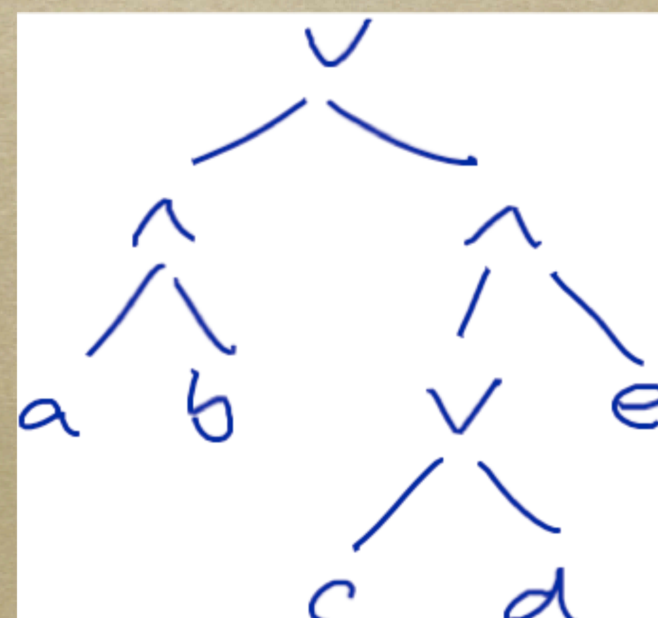
- *To ensure* $y = (c \vee d)$*, want*
  - $y \Rightarrow (c \vee d)$
  - $(c \vee d) \Rightarrow y$



Tseitin transformation

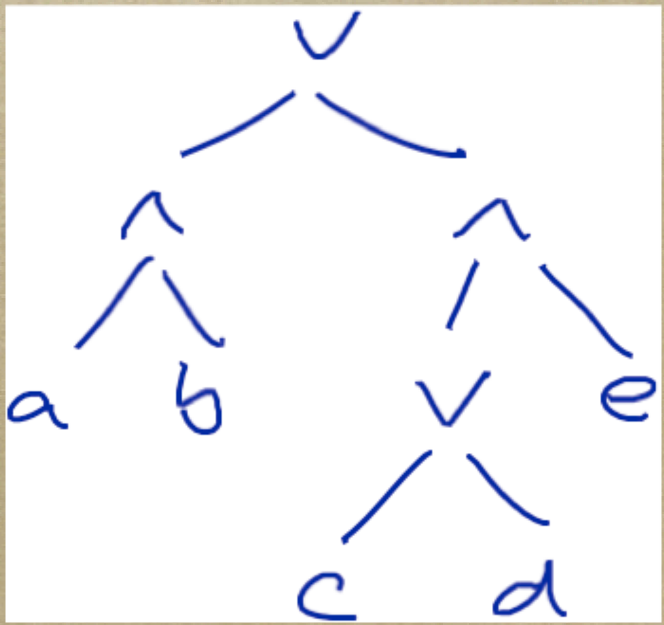
- $y \Rightarrow (c \vee d)$
- $(\neg y \vee c \vee d)$

- $(c \vee d) \Rightarrow y$
- $((\neg c \wedge \neg d) \vee y)$
- $(\neg c \vee y) \wedge (\neg d \vee y)$

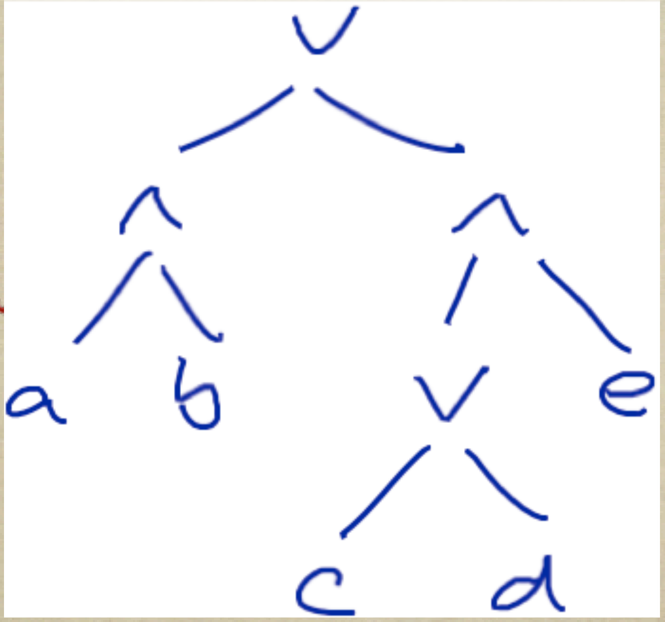


Tseitin transformation

- *Finally*, $z = (y \wedge e)$
- $z \Rightarrow (y \wedge e) \equiv (\neg z \vee y) \wedge (\neg z \vee e)$
- $(y \wedge e) \Rightarrow z \equiv (\neg y \vee \neg e \vee z)$



Tseitin end result



$(a \wedge b) \vee ((c \vee d) \wedge e) \equiv$

$(\neg x \vee a) \wedge (\neg x \vee b) \wedge (\neg a \vee \neg b \vee x) \wedge$

$(\neg y \vee c \vee d) \wedge (\neg c \vee y) \wedge (\neg d \vee y) \wedge$

$(\neg z \vee y) \wedge (\neg z \vee e) \wedge (\neg y \vee \neg e \vee z) \wedge$

$(x \vee z)$

Compositional Semantics

Semantics

- *Recall: meaning of a formula is a function*

  $models \mapsto \{T, F\}$

- *Why this choice? So that meanings are* ***compositional***
- *Write* $[\alpha]$ *for meaning of formula* α
- $[\alpha \wedge \beta](M) = [\alpha](M) \wedge [\beta](M)$
- *Similarly for* $\vee$, $\neg$, *etc.*

Proofs

Entailment

- *Sentence A* ***entails*** *sentence B*, $A \vDash B$, *if B is true in every model where A is*
  - *same as saying that* $(A \Rightarrow B)$ *is valid*

Proof tree

- *A tree with a formula at each node*
- *At each internal node, children* $\models$ *parent*
- *Leaves:* ***assumptions*** *or* ***premises***
- *Root:* ***consequence***
- *If we believe assumptions, we should also believe consequence*

Proof tree example

$rains \Rightarrow pours$

$pours \wedge outside \Rightarrow rusty$

$rains$

$outside$

Proof by contradiction

- *Assume opposite of what we want to prove, show it leads to a contradiction*
- *Suppose we want to show* $KB \models S$
- *Write KB' for* $(KB \wedge \neg S)$
- *Build a proof tree with*
  - *assumptions drawn from clauses of KB'*
  - *conclusion = F*
  - *so,* $(KB \wedge \neg S) \models F$ *(contradiction)*

Proof by contradiction

KB

$rains \Rightarrow pours$

$pours \wedge outside \Rightarrow rusty$

$rains$

$outside$

$\neg rusty$

← negation of desired conclusion

Proof by contradiction