Segment 0
1/9/14

Theory allows for a much higher level of certainty then what would otherwise be possible or expected. For this reason it is used for "mission critical" software right now. But it is likely to be more widespread as computing science matures.

Programs:
- commands
  or
- equations

formal theory - rules of proof
?

Impt: Formal is just a different thought process/language ⇐ mathimatical of correctness

- Only certainty can be proved w/ formal
- Proving go step by step.

- Model checking - i/p 4 many states (10^{60}) (small) → 2^{200} - what is state? - [wiki] = all stored info. at one instant
  - Can abstract, still must prove.

- We will use boolean
  - simple
  - general
  - includes time and space bounds
  - includes prob

Segment 1
0 operand

Theorems: True statement / high voltage / $\top$
Antitheorems: False state / low voltage / $\bot$

2: $x \Rightarrow y$ / x stronger or = y / x implies y
$x \Leftarrow y$ / x implied by y / s weaker = y + more easy.

3: if x then y else z

* associative: $\wedge \vee = \neq$
* Continuing opps: $\Rightarrow, \Leftarrow, =, \neq$
  $x=y=z$ iff $x=y \wedge y=z$
  $x \Rightarrow y \Rightarrow z$ iff $(x \Rightarrow y) \wedge (y \Rightarrow z)$

* Big opperators: $=, \Rightarrow, \Leftarrow$: same just written bigger so they have "later" precidence. ..., or, come first in order $x = y \Rightarrow z$ means $(x=y) \wedge (y \Rightarrow z)$

- if then else

$\top\top\top$	$\top\top\bot$	$\top\bot\top$	$\top\bot\bot$	$\bot\top\top$	$\bot\top\bot$	$\bot\bot\top$	$\bot\bot\bot$
$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$

if the "if" = $\top$, then result = then, if = $\bot$, result = else!

- var sub (instantiation)
  - maintain presidence
  - must be consistant w/ substitution

Boolean Expressions: (grass is green) or 1+1=2
- complete = fully instant or sub all are thermnor anti

- you must be consistant
  ||
  ea/ express is thernmor anti

Axiom = choice. only axiom = $\top$ and anti = $\bot$
but ea/ app can choose it's own
- axioms, like boolean expressions can be left unclassified.

- Evaluation Rule: all subexpressions known, then it is classified
- Completion Rule: You don't need all subexpressions to be classified. $x \vee \top$ can be classified as a theorem because for all assign of x, the expression = $\top$
- Consistency Rule: if you classify an expression and only one way of classifying it's sub-expressions is consistant, then that will be their classification.

Note: subexpressions go all the way to things like "x" and x can be a theorm

- Instance Rule: ∀ classified expressions, all instances have same class

ex. $x = x$ so $\top = \top \vee \top = \bot = \bot \vee \bot$ ← interesting

Classical Logic: 5 rules
Constructive: └ completion
evaluation: └ (consistancy ∧ completion)

* Format: Add spacing to rep precidence
✓ $x \wedge y \; \vee z$ ✗ $x \; \wedge y \vee z$

Some Laws: Material Implication: $a \Rightarrow b = \neg a \vee b$
Duality: deMorgan's law $\neg(a \wedge b) = \neg a \vee \neg b$

---

* Monotonicity vs Antimonotonicity: (aka covariance + contravarance) – mon: $x \leq y \Rightarrow f(x) \leq f(y)$ anti: $x \geq y \Rightarrow f(x) \leq f(y)$ (with numbers)

- Boolean: $x \Rightarrow y$ – x implies y – x is stronger than or equal to
  so.... that means you just use implication instead of inequality ($\geq$)

* Mon: $x \Rightarrow y \Rightarrow f(x) \Rightarrow f(y)$
Ant: $x \Rightarrow y \Rightarrow f(x) \Leftarrow f(y)$

Segment 2

- This can be a little tricky. here is an example:

1. $\neg(a \wedge \neg(a \vee b))$ law of generalization ($a \Rightarrow a \vee b$)

a is stronger than $a \vee b$ because of generalization

$\neg(a \wedge \neg a)$ law of noncontradictions

$\top$ (tricky part):

2. $\neg(a \wedge \underbrace{\neg(a \vee b)}_{\Downarrow})$ because negation is antimonotonic
$\neg(a \wedge \neg a)$

3. $\neg(\underbrace{a \wedge \neg(a \vee b)}_{\Downarrow})$ still same because conjunction is monotonic
$\neg(a \wedge \neg a)$

4. $\neg(\underbrace{a \wedge \neg(a \vee b)}_{\Uparrow})$ negation is antimonotonic, so you flip it. Now that you know what the correct relationship is for the entire expressions you can say:
$\neg(a \wedge \neg a)$

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

* After those steps you have: $\neg(a \wedge \neg(a \vee b)$ – Law of generalization
$\Leftarrow \neg(a \wedge \neg a)$ – Noncontradiction
$= \top$

We can now find the class of $\neg(a \wedge \neg(a \vee b))$ because it is weaker than a expression whose class is true, so it too must be true.

* Context: When in a conjunct, when substituting, you can assume the other value is true
* still don't fully understand how this works...
– look back on this

$\neg(a \wedge \neg(\underbrace{a}_{} \vee b))$ – assume a
$\neg(a \wedge \neg(\top \vee b))$ – symmetry Law and Base Law $\vee$
⋮
$\top$

* Number expressions
– don't leave out multiplication sign

* Char Theory:
"A", "a", " ", """""" and norm symbols