Lecture 8

Friday, February 1, 2019 9:02 AM

<u>Reviewing Last class</u>

DFS – uninformed

doesn't halt unless that property is added

BFS – uses queue

shortest soln ≠ shortest procedure

↓ shortest path

↓ fewest computational steps

expanding a path vs. unfolding recursive calls (NOT backtracking)

<u>A* search</u>

if h is monotone, A* never repeats a state

monotone heuristics are non-increasing

<u>PC Trees</u>

$P,\ P \to Q \vdash Q$

P	Q	P → Q
$\boxed{T}$	T	$\boxed{T}$
T	⊥	⊥
⊥	T	T
⊥	⊥	T

← relevant valuation

when P is T and P → Q is T

Q is also T → checking all combos

is a BRUTE FORCE METHOD

Basics of PC Trees:

$A \vee B,\ \neg A \vdash B$

first write top to bottom

branch ↪

$A \vee B$ ← root

$\neg A$

B ← leaf

any path from root to leaf is called a path: sequence of adjacent statements

branches may split off into their own statements

guaranteed to halt (finite length)

tree rule: ① discharges a statement

② allows us to produce 1+ branches

check symbolizes discharge of statement and each statement can only be discharged once

<u>Test for validity</u>

Rule D (disjunction): if statement $P \vee Q$ ✓

P Q

specifies rule for discharge

$A \vee B$ ✓D

$\neg A$

$\neg B$ ← use negation to create proof by contradiction.

A ✗ B ✗

these paths are both closed bc $A \vee B$ is true, but $\neg A$ and $\neg B$

paths can be either open or closed

o ✗

↘ find a statement & its negation along a path, indicates contradiction

argument is valid by proof of contradiction IF:

① negate conclusion

② expand tree, check statements

③ all paths end up closed → valid

<u>Test for validity</u> (this argument is invalid)

$A \vee B,\ A \vdash B$

$A \vee B$ ✓D

A

$\neg B$

A o B ✗

since all paths are not closed in this proof by contradiction, argument is invalid

Negated disjunction (ND)

$\neg(P \vee Q)$ ✓

$\neg P$

$\neg Q$

goes along the same branch (all conjunction rules)

disjunctive statements split a branch

Conjunction (C)

$P \wedge Q$ ✓

P

Q

Negated Implication (NI)

$\neg(P \to Q)$ ✓

P

$\neg Q$

Negated Conjunction (NC)

$\neg(P \wedge Q)$

$\neg P$ $\neg Q$

Double Negation (DN)

$\neg\neg P$

P

Implication (I)

$P \to Q$ ✓

$\neg P$ Q

two potential issues of PC trees

① Does PCT properly represent PC?

② What properties does PCT have?

↳ efficient (not a brute force soln)

<u>Rules involving the closure of tree paths</u>

negation 1: P & ¬P in the same path

negation 2: ¬P and then P in the same path

constructed def for closed; anything else is open

PCT can be used for more than just validity tests:

- tautology
- satisfiability
- ...

<u>Test for Tautology</u>

Let $S \equiv (A \wedge B) \vee (\neg A \vee \neg B)$

show S is a tautology using PCT

① negate S

② show all paths are closed ∴ S is a tautology

$\neg[(A \wedge B) \vee (\neg A \wedge \neg B)]$ ✓ND ①

$\neg(\neg A \vee \neg B)$ ✓ ND ②

$\neg(A \wedge B)$ ✓ NC ③

$\neg\neg A$

$\neg\neg B$

$\neg A$ ✗ $\neg B$ ✗

since all paths are closed, original statement $(A \wedge B) \vee (\neg A \vee \neg B)$ is a tautology

P ← premises

Q Q'

R R'

conclusions

a given tree Rule R is correct iff

$(\llbracket P \rrbracket^v = T) \Rightarrow (\llbracket S \rrbracket^v = T)$

for $S \in L$, where L is one of our tree's lists of conclusions

definition: R is complete iff the premises of our tree are true whenever $\llbracket S \rrbracket^v = T$ for all $S \in L$.

we're guaranteed to halt w/ PCT rules ← Why?