COSC 290 Discrete Structures

Lecture 7: Argument Checking

Prof. Michael Hay

Monday, Sep. 13, 2017

Colgate University

Plan for today

1. Entailment and tautologies
2. Proving a sentence is a tautology
3. Converting to CNF

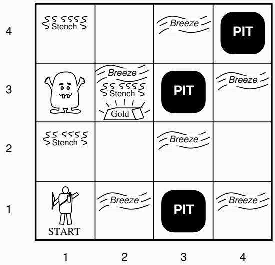
Entailment and tautologies

Wumpus World

An logical agent would like to use its

$KB =$ wumpus-world rules + observations

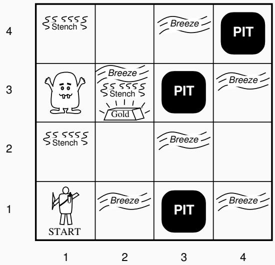
to safely navigate the world and gather the gold.



Example from Wumpus World

Let

- $\varphi_1 := b_{1,1} \iff (p_{1,2} \vee p_{2,1})$
  ("A square is breezy iff there is an adjacent pit.")
- $\varphi_2 := \neg b_{1,1}$ ("No breeze in [1,1]")
- $KB := \varphi_1 \wedge \varphi_2$
- $\alpha := \neg p_{1,2}$ ("[1,2] has no pit")



If $KB \models \alpha$, then agent is 100% certain that [1,2] is safe.

(Recall from Lecture 6) Entailment

Entailment means that one thing *follows from* another:

$$KB \models \alpha$$

Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true.

Ex: the KB containing "the Patriots lost" entails "Either the Patriots lost or the Seahawks won."

Ex: the KB containing rules of algebra and the fact $x + y = 4$ entails $y = 4 - x$.

(Recall from Lecture 6) Tautology

A proposition ψ is a tautology if it is true under every assignment of its variables. In other words, ψ is a tautology if $\psi \equiv \mathit{true}$.

Examples:

- $p \vee \neg p$
- $q \implies q$
- $p \wedge (p \implies q) \implies q$ (modus ponens)
- $(p \implies q) \wedge \neg q \implies \neg p$ (modus tollens)

Entailment and propositional logic

Let *KB* be a sentence in propositional logic.

Let α be a sentence in propositional logic.

The deduction theorem states that

$$KB \models \alpha \text{ if and only if } (KB \implies \alpha) \text{ is a tautology.}$$

The agent needs an inference algorithm to show that ($KB \implies \alpha$) is a tautology.

Today's lecture: a look at algorithms for proving tautologies

Lab 2: implementing a specific algorithm

Proving a sentence is a tautology

Ways to prove a sentence is a tautology

There are basically two ways to show $\psi \implies \varphi$ is a tautology:

Ways to prove a sentence is a tautology

There are basically two ways to show $\psi \implies \varphi$ is a tautology:

1. Using a truth table.
   - Make a truth table with columns for ψ and φ.
   - Check that whenever ψ is true, φ is also true.
     (What about when ψ is false?)

Ways to prove a sentence is a tautology

There are basically two ways to show $\psi \implies \varphi$ is a tautology:

1. Using a truth table.
   - Make a truth table with columns for ψ and φ.
   - Check that whenever ψ is true, φ is also true.
     (What about when ψ is false?)
2. Using known logical equivalences.
   - Step-by-step approach, resembling a proof.
   - Start with this sentence $\psi \implies \varphi$ and gradually transform it into simpler but equivalent sentence until eventually the sentence reduces to *True*.

Today: we will focus on approach 2.

Our Approach

1. Given sentence $S_1 := (\psi \implies \varphi)$, convert S_1 into an equivalent sentence S_2 where S_2 is in conjunctive normal form.
2. Check whether S_2 is a tautology. (This step is easy.)

Conjunctive Normal Form

A proposition is in conjunctive normal form (CNF) if it is the conjunction of one or more clauses where each clause is the disjunction of one or more *literals*.

A literal is an atomic proposition or the negation of an atomic proposition (i.e. it's either p or $\neg p$ for some variable p).

Poll

Which of these propositions is *not* in CNF?

A) $\neg p$

B) $p \vee q$

C) $(p \vee q) \wedge (r \vee s)$

D) $(p \wedge q) \vee (r \wedge \neg p)$

E) More than one is *not* CNF / All are in CNF

(Definition restated here)

A proposition is in CNF if it is the conjunction of one or more clauses where each clause is the disjunction of one or more literals.

A literal is an atomic proposition or the negation of an atomic proposition (i.e. it's either p or $\neg p$ for some variable p).

Checking a CNF sentence for tautology

If S is a proposition in CNF. Then checking for a tautology is easy.

- S is a tautology if and only if each clause is a tautology.
- A clause from a CNF is a tautology if and only if it contains a literal and its opposite.

illustrate this on the board

Poll: is this CNF a tautology?

Consider

$$\varphi := (p \vee q \vee \neg p) \wedge (r \vee p \vee q \vee \neg q) \wedge \neg r$$

Is φ in CNF? Is φ a tautology?

A) CNF: yes, tautology: yes
B) CNF: yes, tautology: no
C) CNF: no, tautology: yes
D) CNF: no, tautology: no

Converting to CNF

Conversion process

Given φ not in CNF, we can convert to an equivalent proposition in CNF by following these steps:

1. Replace "unnecessary" operators like $\iff$, $\implies$, $\oplus$ with a logically equivalent expression.
   Result: φ has only $\{\vee, \wedge, \neg\}$ connectives.
2. Push negations down to obtain negation normal form.
   Result: the *only* places where $\neg$ appears in φ is on a literal.
3. Distribute Or over And.
   Result: φ is in CNF.

Let's apply steps to: $(p \wedge (p \implies q)) \implies q$.

ReplaceIf

Last lecture we looked closely at operators $\iff$, $\implies$, $\oplus$ and showed each operator is logically equivalent to some expression involving only $\{\vee, \wedge, \neg\}$.

Example: $\varphi \implies \psi \equiv \neg\varphi \vee \psi$

Let's think about how we could write a *recursive* algorithm for replacing every "if" statement (i.e., $\implies$ operator).

Shown on board

Negation Normal Form

A sentence is in negation normal form if the negation connective is applied only to atomic propositions (i.e. variables) and not to more complex expressions. Furthermore, the only connectives allowed are $\wedge$, $\vee$, and $\neg$.

Yes: $(\neg p \wedge (\neg p \vee q)) \vee \neg q$

No: $\neg(p \wedge (\neg p \vee q)) \vee q$

Exercise: Negation Normal Form

Given

$$\varphi := \neg(p \land (\neg p \lor q)) \lor q$$

let's write it in negation-normal form by "pushing negations down."

Hint: double negation and De Morgan's laws are useful.

$$\begin{array}{rclr}
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{De Morgan's law \#1} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{De Morgan's law \#2}
\end{array}$$

Distributing OR over AND

The last step is to distribute OR over AND

$$(\alpha \vee (\beta \wedge \gamma)) \quad \equiv \quad ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

Example shown on board.

Poll: thinking recursively

Imagine we are defining a recursive function `DistOrOverAnd` that takes in a sentence φ that is in negation-normal form and returns a sentences in CNF. In other words, the function distributes ORs over ANDs.

Suppose $\varphi := \varphi_1 \vee \varphi_2$ and we make recursive calls

- $S_1 = \texttt{DistOrOverAnd}(\varphi_1)$
- $S_2 = \texttt{DistOrOverAnd}(\varphi_2)$

Suppose you inspect the *returned* propositions S_1 and S_2, and it turns out that...

- S_1 is of the form $S_{11} \vee S_{12}$
- S_2 is of the form $S_{21} \vee S_{22}$

Then is $\varphi = S_1 \vee S_2$ in CNF?

A) Yes
B) Not necessarily