Attendance

Are you here on time? (Timely attendance counts positively towards your participation grade.)

A) Yes

B) No

COSC 290 Discrete Structures

Lecture 18: Wrap up induction

Prof. Michael Hay

Monday, Mar. 19, 2018

Colgate University

Plan for today

1. Converting to CNF
2. Structural induction: propositions expressible in NNF
3. Structural induction: propositions expressible in CNF
4. Lab 3 Implementation tips

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" connectives 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.[1]
   Result: φ is in CNF.

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

[1] Recall $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Example of converting to CNF

$(p \wedge (p \implies q)) \implies q$	
$\equiv \neg(p \wedge (\neg p \vee q)) \vee q$	1. replace $\implies$
$\equiv (\neg p \vee \neg(\neg p \vee q)) \vee q$	2. push negation down
$\equiv (\neg p \vee (p \wedge \neg q)) \vee q$	2. push another negation down
$\equiv ((\neg p \vee p) \wedge (\neg p \vee \neg q)) \vee q$	3. distribute OR
$\equiv ((\neg p \vee p \vee q) \wedge (\neg p \vee \neg q \vee q))$	3. distribute another OR

Lab 3

Three key tasks:

1. Simplify: substitute connectives to get proposition containing *only* $\{\wedge, \neg\}$.[2]
2. toNNF: take simplified proposition and "push negations down" to get proposition in NNF.
3. fromNNFtoCNF: take proposition in NNF and convert to CNF. A key step is *distributing OR over AND* connectives.

Each task is can be solved recursively.

[2] Note: step 1 simplifies more than what is necessary for later steps. But it's a good "warm up" problem for this lab.

Structural induction: propositions expressible in NNF

Recall: propositions can be recursively defined

A proposition φ is a well-formed formula (wff) over the variables in the set $P := \{ p_1, \ldots, p_n \}$, is one of the following:

- (base case) $\varphi := p$ for some $p \in P$
- (inductive cases)
  - $\varphi := \alpha \vee \beta$
  - $\varphi := \alpha \wedge \beta$
  - $\varphi := \alpha \implies \beta$
  - $\varphi := \neg\alpha$

  where α and β are well-formed formulas.

Notation detail: Greek letters (α, β, etc.) represent *propositions*.

Recall: Negation Normal Form

Definition (Negation Normal Form (NNF))

A proposition φ is in negation normal form if the negation connective is applied only to variables and not to more complex expressions, and furthermore, the only connectives allowed are in the set $\{\wedge, \vee, \neg\}$.

Recall claim from last class

Claim: For any well-formed formula φ, there exists a proposition φ' that is in negation normal form and is logically equivalent to φ.

Notation:

- *isNNF*(φ) denotes the predicate: φ is in NNF.
- *hasNNF*(φ) denotes the predicate: there exists a proposition φ' that is in NNF and $\varphi' \equiv \varphi$.
- $\mathcal{W}$ denotes the set of all well-formed formulas.

Thus, our claim can be restated as $\forall\varphi \in \mathcal{W}$: *hasNNF*(φ).

Proof

Claim A: $\forall\varphi \in \mathcal{W}$: *hasNNF*(φ).

We will instead prove the *stronger* claim:

Claim B: $\forall\varphi \in \mathcal{W}$: *hasNNF*(φ) $\wedge$ *hasNNF*($\neg\varphi$).

(See last class slides and book discussion on p. 540.)

Proof structure

Base cases:

1. Variable: $\varphi := p$

Inductive cases:

1. AND: $\varphi := \alpha \wedge \beta$
2. OR: $\varphi := \alpha \vee \beta$
3. NOT: $\varphi := \neg\alpha$
4. IMPLIES: $\varphi := \alpha \implies \beta$

For each case, show that *hasNNF*(φ) and *hasNNF*($\neg\varphi$)

Proof: Inductive case 1, what to show?

Claim B: $\forall \varphi \in \mathcal{W}$: *hasNNF*(φ) $\wedge$ *hasNNF*($\neg\varphi$).

Inductive cases: We focus on case 1: $\varphi := \alpha \wedge \beta$.

Want to show: *hasNNF*($\alpha \wedge \beta$) $\wedge$ *hasNNF*($\neg(\alpha \wedge \beta)$).

We will break this up into parts:

1. Showing *hasNNF*($\alpha \wedge \beta$)
2. Showing *hasNNF*($\neg(\alpha \wedge \beta)$)

Poll: Inductive case 1, what can we assume?

Claim B: $\forall \varphi \in \mathcal{W}$: *hasNNF*(φ) $\wedge$ *hasNNF*($\neg\varphi$).

Proof continued...

Inductive cases: We focus on case 1: $\varphi := \alpha \wedge \beta$.

Want to show: *hasNNF*($\alpha \wedge \beta$) $\wedge$ *hasNNF*($\neg(\alpha \wedge \beta)$). Which of the following can we assume is true (by the inductive hypothesis)?

A) *hasNNF*(α) ... recall this means that α is logically equivalent to some NNF proposition.

B) *hasNNF*($\neg\alpha$)

C) *isNNF*(α) ... recall this means that α is an NNF.

D) A and B

E) A, B, and C

Proof for inductive case 1

Claim B: $\forall \varphi \in \mathcal{W}$: *hasNNF*(φ) $\wedge$ *hasNNF*($\neg\varphi$).

Inductive cases: We focus on case 1: $\varphi := \alpha \wedge \beta$.

Want to show: *hasNNF*($\alpha \wedge \beta$) $\wedge$ *hasNNF*($\neg(\alpha \wedge \beta)$).

Assume by inductive hypothesis:

- *hasNNF*(α), *hasNNF*(β), *hasNNF*($\neg\alpha$), *hasNNF*($\neg\beta$)

Part 1: Since *hasNNF*(α), there exists α' such that $\alpha' \equiv \alpha$ and *isNNF*(α'). Similarly for β. Let $\varphi' := \alpha' \wedge \beta'$. We have *isNNF*($\varphi'$) and $\varphi' \equiv \alpha \wedge \beta$. Thus *hasNNF*($\alpha \wedge \beta$).

Part 2: $\neg\varphi = \neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ by DeMorgan's law. Since *hasNNF*($\neg\alpha$), there exists $\bar{\alpha}$ such that $\bar{\alpha} \equiv \neg\alpha$ and *isNNF*($\bar{\alpha}$). Similarly for β. Thus, let $\bar{\varphi} := \bar{\alpha} \vee \bar{\beta}$. We have *isNNF*($\bar{\varphi}$) and $\bar{\varphi} \equiv \neg(\alpha \wedge \beta)$. Thus *hasNNF*($\neg(\alpha \wedge \beta)$).

Proof for inductive case 2

Claim B: $\forall \varphi \in \mathcal{W} : \textit{hasNNF}(\varphi) \land \textit{hasNNF}(\neg\varphi)$.

Inductive cases: Case 2: $\varphi := \alpha \lor \beta$.

Proof is identical to case 1, just replace ANDs with ORs and vice versa.

Poll: Inductive case 3, what to show?

Claim B: $\forall \varphi \in \mathcal{W} : \textit{hasNNF}(\varphi) \land \textit{hasNNF}(\neg\varphi)$.

Inductive cases: Case 3: $\varphi := \neg\alpha$.

What do we want to show?

A) $\textit{hasNNF}(\alpha)$
B) $\textit{hasNNF}(\neg\alpha)$
C) $\textit{hasNNF}(\neg\neg\alpha)$
D) B and C
E) A, B, and C

Proof for inductive case 3

Claim B: $\forall \varphi \in \mathcal{W} : \textit{hasNNF}(\varphi) \land \textit{hasNNF}(\neg\varphi)$.

Inductive cases: Case 3: $\varphi := \neg\alpha$.

Want to show: $\textit{hasNNF}(\neg\alpha) \land \textit{hasNNF}(\neg\neg\alpha)$.

Assume by inductive hypothesis:

- $\textit{hasNNF}(\alpha)$, $\textit{hasNNF}(\neg\alpha)$

Still need to show: $\textit{hasNNF}(\neg\neg\alpha)$.

Since $\neg\neg\alpha \equiv \alpha$ and $\textit{hasNNF}(\alpha)$, then let α' be such that $\alpha' \equiv \alpha$ and $\textit{isNNF}(\alpha')$. Let $\bar{\varphi} := \alpha'$. Since $\bar{\varphi} \equiv \neg\neg\alpha$ and $\textit{isNNF}(\bar{\varphi})$, thus $\textit{hasNNF}(\neg\neg\alpha)$.

Components of complete proof

Base cases:

1. Variable: $\varphi := p$

Inductive cases:

1. AND: $\varphi := \alpha \land \beta$
2. OR: $\varphi := \alpha \lor \beta$
3. NOT: $\varphi := \neg\alpha$
4. IMPLIES: $\varphi := \alpha \implies \beta$

For each case, show that $\textit{hasNNF}(\varphi)$ and $\textit{hasNNF}(\neg\varphi)$

Structural induction: propositions expressible in CNF

From NNF to CNF

Claim: For any φ that is in NNF, there exists a proposition φ' that is in CNF and is logically equivalent to φ.

Notation used in proof:

- *isCNF*(φ) denotes the predicate: φ is in CNF.
- *hasCNF*(φ) denotes the predicate: there exists a proposition φ' that is in CNF and $\varphi' \equiv \varphi$.

Poll: what cases to consider?

Claim: For any φ that is in NNF, there exists a proposition φ' that is in CNF and is logically equivalent to φ.

We can break the proof of this claim into cases (base, inductive and sub-cases of each). Which of the following cases should *not* be included in the proof?

A) $\varphi \coloneqq \alpha \implies \beta$ (where α and β are propositions)
B) $\varphi \coloneqq \alpha \wedge \beta$
C) $\varphi \coloneqq \neg\alpha$
D) $\varphi \coloneqq \neg p$ (where p is a variable)
E) A and C
F) A and D

Proof structure

Base cases:

1. Positive literal: $\varphi \coloneqq p$
2. Negative literal: $\varphi \coloneqq \neg p$

Inductive cases:

1. $\varphi \coloneqq \alpha \wedge \beta$
2. $\varphi \coloneqq \alpha \vee \beta$

We don't need to consider anything else because φ is in NNF!

Poll: inductive case 1, what can we assume?

Want to show: *hasCNF*$(\alpha \wedge \beta)$. Which of the following can we assume is true (by the inductive hypothesis)?

A) *hasCNF*(α), *hasCNF*(β)
B) *hasCNF*$(\neg\alpha)$, *hasCNF*$(\neg\beta)$
C) *isCNF*(α), *isCNF*(β)
D) A and C
E) A, B, and C

Inductive case 2

Case 2: $\varphi \coloneqq \alpha \vee \beta$

Inductive hypothesis tells us that *hasCNF*(α) and *hasCNF*(β). Let α' be such that $\alpha \equiv \alpha'$ and *isCNF*(α'); similarly for β'.

Let $\alpha' \coloneqq c_1 \wedge c_2 \wedge \cdots \wedge c_m$ where each c_i is a clause (disjunction of one or more literals).

Let $\beta' \coloneqq d_1 \wedge d_2 \wedge \cdots \wedge d_n$ where each d_j is a clause.

Inductive case continued...

Inductive cases: case 2 continued...

$$\begin{aligned}
\varphi &\equiv \alpha' \vee \beta' && \text{inductive hypothesis}\\
&\equiv \left(\bigwedge_{i=1}^{m} c_i\right) \vee \beta' && \text{definition of } \alpha'\\
&\equiv \bigwedge_{i=1}^{m} (c_i \vee \beta') && \text{distribute OR over ANDs of } \alpha'\\
&\equiv \bigwedge_{i=1}^{m} \left(c_i \vee \left(\bigwedge_{j=1}^{n} d_j\right)\right) && \text{definition of } \beta'\\
&\equiv \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{n} (c_i \vee d_j) && \text{distribute OR over ANDs of } \beta'
\end{aligned}$$

The last line shows a proposition in CNF! How so? Recall c_i and d_j are both clauses – i.e. disjunctions of literals. ORing two clauses together effectively makes new, bigger clause. All of these new clauses are being ANDed together.

Lab 3 Implementation tips

Suggestions for lab 3

Have faith: trust that recursive call will work correctly. Focus on what to do with the result.

Start small (base cases).

Add complexity slowly.

Test your code often.

- *Before* you modify your code to add support a new case, write a test example for that case.
- *After* each change to your code, re-run all of your old cases to make sure changes didn't "break".