論理と計算

第３回
命題論理：推論

担当：尾崎　知伸
ozaki.tomonobu@nihon-u.ac.jp

講義予定　※一部変更（前倒し）になる可能性があります

09/22	01. オリエンテーション と 論理を用いた問題解決の概要
09/29	02. 命題論理：構文・意味・解釈
10/06	03. 命題論理：推論
10/13	04. 命題論理：充足可能性問題
10/20	05. 命題論理：振り返りと演習（課題学習）
10/27	06. 述語論理：構文・意味・解釈
11/03	07. 述語論理：推論　※文化の日，文理学部授業日
11/10	08. 述語論理：論理プログラムの基礎
11/17	09. 述語論理：論理プログラムの発展
11/24	10. 述語論理：振り返りと演習（課題学習）
12/01	11. 高次推論：発想推論
12/08	12. 高次推論：帰納推論の基礎
12/15	13. 高次推論：帰納推論の発展
12/22	14. 高次推論：振り返りと演習（課題学習）
01/19	15. まとめと発展的話題

目次：今回の授業の内容

- 命題論理の推論
  - 推論と伴意
  - 推論手続き
    - モーダスポーネンス
    - 公理的証明法
    - 融合法（今日のメイン）

- ホーン節集合に対する推論
  - 前向き推論
  - 後ろ向き推論

推論と伴意

伴意・トートロジー・推論

問題を論理で表現し，
伴意関係を調べることで
問題を解決する

- そもそもやりたいこと：伴意関係$G \vDash \alpha$が成り立つかを調べる
  - 文集合Gのもとで，どんな文αが論理的に成り立つかを知りたい
    - 既知の知識から，どんな知識や事実が導かれるのか知りたい
  - 文αがGのもとで，論理的に正しいかを知りたい
    - 新しく考えた知識や事実が，既知の知識と矛盾しないか知りたい

- 定理：$G = \{\gamma_1, \gamma_2, \dots\}$を文の集合，$\alpha$を文とする．伴意式$G \vDash \alpha$が成り立つのは，$(\gamma_1 \wedge \gamma_2 \wedge \cdots) \Longrightarrow \alpha$がトートロジーのとき，かつ，そのときのみである
  - （命題文ではない）伴意テストを，含意文のトートロジーテスト（命題文の証明問題）に還元

- 上記の定理より，
  - $\alpha \Leftrightarrow \beta$ は，$\beta \vDash \alpha$かつ$\alpha \vDash \beta$のとき，またそのときに限り成り立つ

- 上記の定理は，反駁証明（背理法）の指針を与える
  - $G \vDash \alpha$ であるには，$G \Longrightarrow \alpha$が恒真（トートロジー）である必要がある
  - $G \Longrightarrow \alpha$ が恒真なら，その否定$\neg(G \Longrightarrow \alpha)$は恒偽
  - $\neg(G \Longrightarrow \alpha)$を変形すると$\neg(\neg G \vee \alpha)$となり $(G \wedge \neg\alpha)$となる
    - G にαの否定を追加して，矛盾が導かれるか確認すればよい

伴意式の証明

- （これまでの議論では）トートロジーテストは，命題文に対する真理値表の作成で達成
  - 真理値表の大きさは，命題記号数の指数関数（N個の命題記号に対し2^Nの解釈）
  - Nが大きいと現実的ではない（命題論理の推論はNP完全）
  - 真理値表を作成するアルゴリズムは，推論手法としては健全かつ完全

- モデル論 vs 証明論
  - モデル論（model theory）
    - 「論理式の正しさを示すのに伴意関係を調べる」という立場
    - 理論Gのすべてのモデルが，文αのモデルでもあることを調べることに相当
    - 手間がかかりすぎる
  - 証明論（proof theory）
    - 理論G＋公理から，いくつかの文を選択し，推論規則を適用することで，新たな文を生成する．この過程を繰り返すことによって，目的とする文αを導く
    - 推論式：$G \vdash \alpha$「理論Gから文αが推論される（証明される）」
      - 完全性，健全性には注意が必要（後述）

- より効率の良い証明手続きが必要
  - モーダスポーネンス，AND除去，融合法，公理的証明法など

モデル論と証明論

- モデル論と証明論の役割の違い
  - モデル論：
    - 論理式の真偽の拠り所を，その論理式が成り立つ世界（モデル）で与えることにより，推論の妥当性の根拠を与える
    - 証明論における証明手続きの正当性を保証する
  - 証明論
    - 証明手続きを厳密に定めることによって，得られる結論の妥当性を保証しようとする
    - 正しさが保証された証明手続きを用いることによって，効率的に証明を行うことが可能である

- 論理式・推論式・伴意式
  - 論理式：一定の構文によって作られる真偽が定まる式
  - 推論式：論理式の集合が与えられたとき，それらがすべて正しい場合にそこからどの様な論理式が導かれるのかを示す式．それ自身は論理式ではない
  - 伴意式：伴意関係を表す式．伴意式の成否は，左辺を理論，右辺を文とする含意文（論理式）のトートロジー判定に還元できる

推論手続き

推論

- 実世界の事実を論理で表す目的・利点
  - 論理が持つ，表現対象によらない不偏的な推論手続きを利用する
    - 「機械的な記号処理」のみを用いて推論を行う

- 推論：理論Gから与えられた手続きに従って文 α を導く過程
  - 一般に，導出には複数のステップが必要 ／ 各ステップ＝推論規則の適用

- 推論図式：$\frac{\Gamma}{\alpha}$　論理式の集合Γから論理式αを導出する

推論手続きの健全性と完全性

- 推論手続きの**健全性**（soundness）
  - 推論手続きによって得られた結果がモデル論に照らして正しくなければならない
  - 推論手続きによって間違った結果が得られることはない
- 推論手続きの**完全性**（completeness）
  - 推論手続きによって，モデル論において正しいとされるすべての結果を導くことができる
  - 推論手続きによって，漏れなく網羅的に正しい結果を得ることができる

- 健全性・完全性の両方を備えることが望ましい
  - 融合法（次回以降学修）は命題論理において，健全かつ完全
    - 述語論理において，健全かつ，反駁完全（反駁証明に対して完全）

- 伴意関係を調べる手続き
  - 命題論理は決定的（決定可能）：任意の文から他の文が伴意されるか・されないかを調べる有限時間で調べることができるアルゴリズムが存在
  - 述語論理は準決定可的（準決定可能）：伴意される場合は，そのことを有限時間で示すことはできる．伴意されない場合は，それを有限時間で示すことが保証できない

推論図式：$\frac{\Gamma}{\alpha}$　論理式の集合Γから論理式αを導出する

推論規則：モーダスポーネンス（Modus Ponens）

- いわゆる三段論法：含意文$\alpha \Rightarrow \beta$と文αから文βを導出する
  - α, βは（命題記号ではなく）文である

$$\frac{\alpha \Rightarrow \beta \qquad \alpha}{\beta}$$

$\alpha \Rightarrow \beta$ が真 かつ α が真なら β は真である

- KB= {(A ∧ B), (A ∧ B) ⇒ (C ∨ D)} から C ∨ D が導かれる　$\frac{(A \wedge B) \Rightarrow (C \vee D) \quad (A \wedge B)}{C \vee D}$
- モーダスポーネンスは健全である
  - α が真で，$\alpha \Rightarrow \beta$ が真であれば，β が真でなければならない
  - $\alpha \wedge (\alpha \Rightarrow \beta) \rightarrow \alpha \wedge (\neg \alpha \vee \beta) \rightarrow (\alpha \wedge \neg \alpha) \vee (\alpha \wedge \beta)$
- モーダスポーネンスは完全ではない
  - w：芝生がぬれる
  - r：雨が降る　r ⇒ w：雨が降れば芝生はぬれる
  - s：スプリンクラーが動作する　s⇒ w：スプリンクラーが動作すれば芝生はぬれる

G⇒wの真理値表を書いて，Gからwが論理的に導かれることを確認してみよう

  - 「雨が降っているかスプリンクラーが動作している」とき，すなわち（r∨s）が真のときでも，モーダスポーネンス（だけ）ではwを導くことはできない
  - 形式的には，G={(r ⇒ w), (s⇒ w) , (r∨s) } からwを導出することはできない
    - 完全性に対する反例

推論規則の例

- And-除去（$\alpha \wedge \beta$ から α を導く）

- トートロジーである同値文は推論規則して用いることができる
  - 二つの文 α と β の間の同値文 $\alpha \Leftrightarrow \beta$ がトートロジーであれば，

推論規則として，$\frac{\alpha}{\beta}$ と $\frac{\beta}{\alpha}$ を用いることができる

$$\frac{\alpha \wedge \beta}{\alpha}\ (And-除去)$$

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}\ (双条件除去)$$

$$\frac{\alpha \Rightarrow \beta}{\neg\beta \Rightarrow \neg\alpha}\ (対偶)$$

$$\frac{\neg(\alpha \wedge \beta)}{\neg\alpha \vee \neg\beta}\ (ド・モルガンの法則)$$

$$\frac{\neg(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta}\ (ド・モルガンの法則)$$

トートロジーである同値文の例

$\alpha \Rightarrow \alpha$	
$\alpha \vee \neg \alpha$	(排中律)
$\alpha \Leftrightarrow \neg \neg \alpha$	(二重否定)
$(\alpha \Rightarrow \beta) \Leftrightarrow (\neg \alpha \vee \beta)$	(含意記号の定義)
$(\alpha \Leftrightarrow \beta) \Leftrightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$	(同値記号の定義)
$(\alpha \Rightarrow \beta) \Leftrightarrow (\neg \beta \Rightarrow \neg \alpha)$	(対偶)
$\alpha \wedge \beta \Leftrightarrow \beta \wedge \alpha$	(連言の交換率)
$\alpha \vee \beta \Leftrightarrow \beta \vee \alpha$	(選言の交換率)
$\neg (\alpha \wedge \beta) \Leftrightarrow \neg \alpha \vee \neg \beta$	(ド・モルガンの法則)
$\neg (\alpha \vee \beta) \Leftrightarrow \neg \alpha \wedge \neg \beta$	(ド・モルガンの法則)
$\alpha \wedge (\beta \wedge \gamma) \Leftrightarrow (\alpha \wedge \beta) \wedge \gamma$	(連言の結合律)
$\alpha \vee (\beta \vee \gamma) \Leftrightarrow (\alpha \vee \beta) \vee \gamma$	(選言の結合律)
$\alpha \wedge (\beta \vee \gamma) \Leftrightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	($\wedge$ の $\vee$ への分配率)
$\alpha \vee (\beta \wedge \gamma) \Leftrightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	($\vee$ の $\wedge$ への分配率)

Wumpus Worldにおける推論の例

- R1：$\neg P_{1,1}$　　部屋[1,1]には穴がない
- R2：$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$　隣の部屋に穴があるとき（またそのときに限り）風を感じる
- R4：$\neg B_{1,1}$　　[1,1]で風を感じない
- 証明したいこと：$\neg P_{1,2} \wedge \neg P_{2,1}$（[1,1]で風を感じないので，となりの[1,2],[2,1]には穴がない）
- R2に双条件除去を適用：$\dfrac{B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})}{(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})}$
- And除去を適用：$\dfrac{(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})}{(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}}$，　対偶を適用：$\dfrac{(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}}{\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})}$
- モーダスポーネンスを適用：$\dfrac{\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}),\quad \neg B_{1,1}}{\neg (P_{1,2} \vee P_{2,1})}$
- ド・モルガンの法則を適用：$\dfrac{\neg (P_{1,2} \vee P_{2,1})}{\neg P_{1,2} \wedge \neg P_{2,1}}$

$$\dfrac{\dfrac{\dfrac{\dfrac{B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})}{(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})}\ (\text{双条件除去})}{(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}}\ (And\ \text{除去})}{\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})}\ (\text{対偶}) \qquad \neg B_{1,1}}{\neg (P_{1,2} \vee P_{2,1})}\ (\text{モーダスポーネンス})}{\neg P_{1,2} \wedge \neg P_{2,1}}\ (\text{ド・モルガンの法則})$$

命題論理の推論

- 命題論理における推論はNP完全である
  - 現実的な場面では，証明の探索が非常に効率よく行える可能性がある
  - ゴールとは関係のない命題・式は無視することができる

- Wumpus Worldの例
  - 導きたい命題$\neg P_{1,2} \wedge \neg P_{2,1}$の証明に，

    $B_{2,1}, P_{1,1}, P_{2,2}$ などの命題や式$B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$は出てこない

- 単調性（monotonicity）
  - 情報が知識ベースに追加されたときに，そこから伴意される文の集合が増加する
    - 文が増えても，既に伴意された文が，伴意されなくなることはない
  - 知識ベースKBと任意の文αとβに関し，以下の関係が成り立つ
    - $KB \vDash \alpha$ ならば $KB \wedge \beta \vDash \alpha$

  - 単調性が成り立つので，他にどのような文が知識ベースにあっても，規則の適用により導かれた結論は知識ベースに従う

推論規則：公理論的証明法

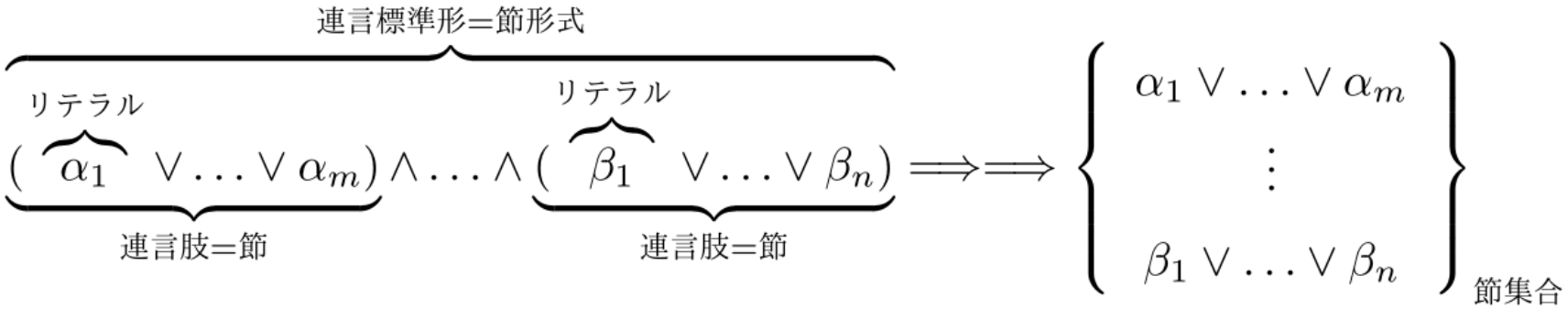
- 結合子として，含意（⇒）と否定（¬）のみを必要不可欠なものと考える
  - 残りの結合子は，その二つから導いている

- 命題論理の場合
  - 公理
    - $\alpha \Rightarrow (\beta \Rightarrow \alpha)$
    - $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$
    - $(\neg \beta \Rightarrow \neg \alpha) \Rightarrow ((\neg \beta \Rightarrow \alpha) \Rightarrow \beta)$
  - 推論規則：モーダスポーネンス
  - 公理＋推論規則のみを用いて，命題文の集合から伴意されるすべての文を導き出すことができる（完全である）

推論規則：融合法

- 融合法（resolution）：節形式・節集合を対象とした推論規則
  - 節形式（=連言標準形）または節集合（連言標準形の集合表現）で知識ベースを準備する
  - 知識ベースは，命題文の”連言”
  - 各命題文は，連言標準形で表現することが可能.

- 準備１：節形式（＝連言標準形）と節集合
  - すべての連言肢がリテラルの選言である連言文は，連言標準形をしているという
    - 「リテラルの選言」の連言＝「リテラルの∨」を∧で繋いだもの
  - 0個以上のリテラルの選言を節（clause）と呼ぶ
  - 0個以上の節の連言（＝連言標準形）を節形式（clausal form）と呼ぶ
  - 節形式を（連言ではなく）節の集合として表したものを節集合（clause set）と呼ぶ



- 準備２：相補リテラル（complementary literal）
  - 同じ命題変数に対する正負が逆のリテラル（例： p に対する ¬p，¬qに対する q）

推論規則：融合法

- 融合規則
  - 互いに相補リテラルを含む2つの節（融合可能な節の対）を融合する
    ＝2節を結合子∨で連結し，相補リテラルの除去とファクタリングを行う．
    → 結果として得られる節を融合節（resolvent）と呼ぶ（∨連結なので結果も節になる）
    - 「正リテラルと負リテラルで互いに打ち消す」というイメージ

$$\frac{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m \vee l \quad \neg l \vee \beta_1 \vee \beta_2 \vee \ldots \vee \beta_n}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m \vee \beta_1 \vee \beta_2 \vee \ldots \vee \beta_n} \text{（融合規則）}$$

  - 特に，m=0 またはn=0の場合を単位節融合（unit resolution）と呼ぶ
    - この場合，片方の節は消えることになる

$$\frac{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m \vee l \quad \neg l}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m} \text{（単位節融合）}$$

  - ファクタリング（factoring）：同じリテラルを一つする

$$\frac{(A \vee B) \quad (A \vee \neg B)}{A \vee A} \Longrightarrow \frac{(A \vee B) \quad (A \vee \neg B)}{A}$$

融合証明手続き

- 定義：与えられた節集合G_0から，節αの融合証明手続きは，次の通りである
  1. G := G_0
  2. G中から融合可能な節の対を選び，融合規則を適用して融合節Rを得る
  3. もしR＝αなら停止する
  4. そうでないならG := G∪{R}とし，ステップ2に戻る

     ※厳密に言えば，上記の手続きは，停止しない場合がある

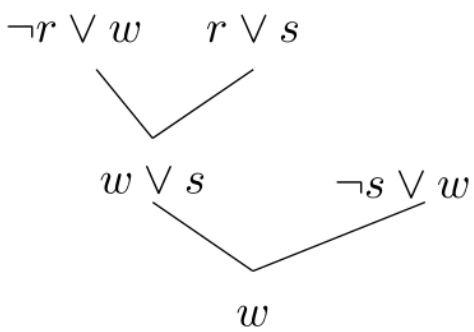
- 定義：$\vdash_r$
  - 融合証明手続きにより，節集合Gから節αが導出されることをと$G \vdash_r \alpha$と表す

r：雨が降る
s：スプリンクラーが動作している
w：芝生がぬれる
r ⇒ w：雨が降れば芝生はぬれる
s⇒ w：スプリンクラーが動作していれば芝生はぬれる

「雨が降っているかスプリンクラーが動作している」とき，
すなわち（r∨s）が真のときに，wを導く
形式的には，G={(r ⇒ w), (s⇒ w) , (r∨s) } からwを導出する

Gからα=wを導く例

$$G = \{\neg r \vee w, \quad \neg s \vee w, \quad r \vee s\}, \quad \alpha = w$$

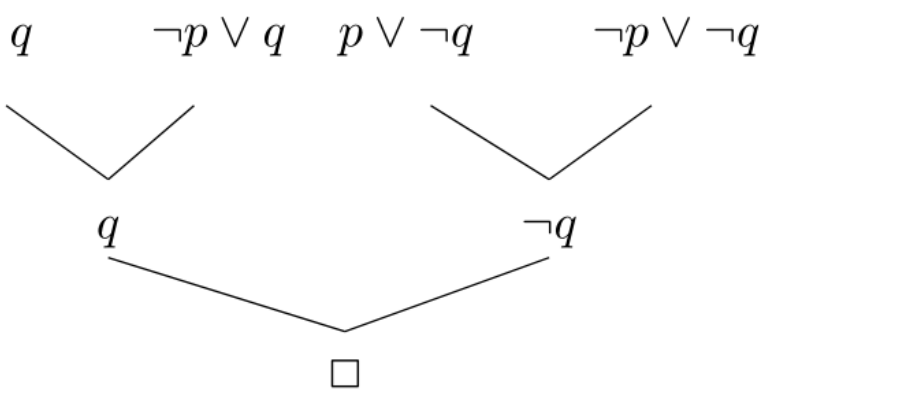


矛盾の導出

- 0個の文の選言（＝連言肢）を□で表し，その意味をfalseとする
  - $(p_1 \vee \cdots \vee p_m)$と$(p_1 \vee \cdots \vee p_m \vee \text{false})$ は同値

- □ (false)が導出されるとき，Gは恒偽である（矛盾している）
  - Gを真とする解釈（モデル）は存在しない

$$G = \{p \vee q, \quad p \vee \neg q, \quad \neg p \vee q, \quad \neg p \vee \neg q\}, \quad \alpha = \square$$



融合法の健全性と完全性

$$\frac{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m \vee l \quad \neg l \vee \beta_1 \vee \beta_2 \vee \ldots \vee \beta_n}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m \vee \beta_1 \vee \beta_2 \vee \ldots \vee \beta_n}$$

- 融合法は健全である（$G \vdash_r \alpha$ ならば $G \models \alpha$）
  - 相補リテラルに着目して考える
  - lが真であれば，¬lは偽である．
    - ¬l∨ $\beta_1 \vee \cdots \vee \beta_n$は真であるので，$\beta_1 \vee \cdots \vee \beta_n$は真でなければならない
    - （l∨ $\alpha_1 \vee \cdots \vee \alpha_m$は真であるが，$\alpha_1 \vee \cdots \vee \alpha_m$の真偽は不明．
      しかし$\beta_1 \vee \cdots \vee \beta_n$が真なら，$\alpha_1 \vee \cdots \vee \alpha_m \vee \beta_1 \vee \cdots \vee \beta_n$は真となる
  - lが偽であれば，¬lは真である．
    - l∨ $\alpha_1 \vee \cdots \vee \alpha_m$は真であるので，$\alpha_1 \vee \cdots \vee \alpha_m$は真でなければならない
    - （¬l∨ $\beta_1 \vee \cdots \vee \beta_n$は真であるが，$\beta_1 \vee \cdots \vee \beta_n$の真偽は不明．
      しかし$\alpha_1 \vee \cdots \vee \alpha_m$が真なら，$\alpha_1 \vee \cdots \vee \alpha_m \vee \beta_1 \vee \cdots \vee \beta_n$は真となる
- 融合法は完全ではない
  - $G = \{p, q\},\ \alpha = p \vee q$ のとき$G \models \alpha$ であるが　$G \nvdash_r \alpha$ である
    - p と q は相補リテラルを持たないので，融合操作を適用することができない
    - 真理値表を用いて $G \models \alpha$ であることを確認してみよう

p	q	$(p$	$\wedge$	$q)$	$\Rightarrow$	$(p$	$\vee$	$q)$
$false$	$false$	$false$	$false$	$false$	$true$	$false$	$false$	$false$
$false$	$true$	$false$	$false$	$true$	$true$	$false$	$true$	$true$
$true$	$false$	$true$	$false$	$false$	$true$	$true$	$true$	$false$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$true$	$true$

融合法の健全性と完全性

- 融合法は反駁完全である($G \models \alpha$，すなわち$(G \wedge \neg\alpha) \models \square$ならば$(G \wedge \neg\alpha) \vdash_r \square$)
  - 反駁証明（背理法）
    - $G \models \alpha$ であるには，$G \Longrightarrow \alpha$が恒真（トートロジー）である必要がある
    - $G \Longrightarrow \alpha$ が恒真なら，その否定$\neg(G \Longrightarrow \alpha)$は恒偽
    - $\neg(G \Longrightarrow \alpha)$を変形すると$\neg(\neg G \vee \alpha)$となり $(G \wedge \neg\alpha)$となる
      - G にαの否定を追加して，矛盾が導かれるか確認すればよい

- 定理：節集合Hが 恒偽であれば，融合規則のみを用いてHから□を証明できる
  - $H = (G \wedge \neg\alpha)$とする．$(G \wedge \neg\alpha) \models \square$でならば $(G \wedge \neg\alpha) \vdash_r \square$
  - 証明（次ページ参照）

反駁証明の例

$G = \{p, q\}, \alpha = p \vee q$　より　　$(G \wedge \neg\alpha) = \{p, q, \neg p, \neg q\}$

（参考）証明：節集合Hが恒偽であれば，融合規則のみを用いてHから□を証明できる

- Hに含まれる原子文の数 N に関する数学的帰納法を用いる．
- 初期ステップ：N=0のとき k，節集合Hはfalseしか含まないので，定理は成り立つ
- 帰納ステップ：N = k – 1のとき，定理が成り立つとし，HがN = k 個の原子文を含む場合を考える
  - Hに含まれる原子文をaとする．
  - (1) aを真と仮定する
    - このとき，aを含む節は（a∨…の形をしているので）真となる．この様な節を除いても，（前提より）Hは恒偽であり，(またH= $\alpha \wedge \cdots$の形をしているので)Hの真偽に影響を与えない
    - 同様に，残った各節から（存在すれば）¬aを取り除いても（¬a∨b∨…とfalse∨b…は同値なので）Hの真偽に影響を与えない
    - 上記の除去操作によってHから得られた節集合をH’とすると，H’はk-1個の原子式を含み，また（仮定より）恒偽であるので， $\mathrm{H}' \vdash_r \square$である．
    - $\mathrm{H}' \vdash_r \square$の証明図に¬aを戻すと，結果として□もしくは¬aが導出される
  - (2) aを偽と仮定する
    - このとき，¬aを含む節は（¬a∨…の形をしているので）真となる．この様な節を除いても，（前提より）Hは恒偽であり，(またH= $\alpha \wedge \cdots$の形をしているので)Hの真偽に影響を与えない
    - 同様に，残った各節から（存在すれば）aを取り除いても（a∨b∨…とfalse∨b…は同値なので）Hの真偽に影響を与えない
    - 上記の除去操作によってHから得られた節集合をH’’とすると，H’’はk-1個の原子式を含み，また（仮定より）恒偽であるので， $\mathrm{H}'' \vdash_r \square$である．
    - $\mathrm{H}'' \vdash_r \square$の証明図にaを戻すと，結果として□もしくはaが導出される
  - 2つの証明図はそれぞれ対応するので，両者から□が証明されるか，それぞれで¬aとaが証明されるかでのどちらかである．
  - 後者の場合でも，¬aとaを融合させることで，□を導くことができる．

反駁証明のアルゴリズム

※余力があったら実装してみよう

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new $\leftarrow \{\}$
  while *true* **do**
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new $\leftarrow$ *new* $\cup$ *resolvents*
    if *new* $\subseteq$ *clauses* **then return** *false*
    clauses $\leftarrow$ *clauses* $\cup$ *new*

Figure 7.13 A simple resolution algorithm for propositional logic. PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

融合法の特徴

- 健全かつ反駁完全
  - 間違いのない推論・漏れのない推論
- 単一の推論規則
  - 推論アルゴリズムの実装が容易
  - 種々の推論アルゴリズムの基礎
- 多くの実用的状況では，融合法のすべての能力が必要とされるわけではない
  - 更なる表現の制限・より効率的な手続き（実装）の必要性
    - ナイーブな反駁証明アルゴリズム（前ページ）では，同じ操作を繰り返すことになる

参考：演繹定理（Deduction Theorem）

- 定理： Gを命題論理式の集合， α， β を命題論理式とする．このとき，健全かつ完全な証明手続き⊢に対して，以下の関係が成り立つ

$$\{G \cup \{\alpha\}\} \vdash \beta \iff G \vdash (\alpha \Rightarrow \beta)$$

- 証明：証明手続きが健全かつ完全ならば⊢を⊨に置き換えることができる
  - $\{G \cup \{\alpha\}\} \models \beta \Leftarrow G \models (\alpha \Rightarrow \beta)$ の証明
    - Gのモデルは， α が真となるモデルと α が偽となるモデルに分けることができる
    - aが真の場合，$\{G \cup \{\alpha\}\}$は真であり，また $\alpha \Rightarrow \beta$ より β は真である．
    - aが偽の場合，$\{G \cup \{\alpha\}\}$は偽なので，|=に関係しない
  - $\{G \cup \{\alpha\}\} \models \beta \Rightarrow G \models (\alpha \Rightarrow \beta)$
    - Gのモデルは α が真となるモデルと α が偽となるモデルに分けることができる
    - Gのモデルの内， α が真となるモデルでは，（前提より） β は必ず真となる．よって $\alpha \Rightarrow \beta$ が真となる
    - Gのモデルの内， α が偽となるモデルでは，（α の値に関係なく） $\alpha \Rightarrow \beta$ が真となる

- 演繹定理より，以下の系が成り立つ
- 系：Gを命題論理式の集合， α， β を命題論理式とする．

  このときが成り立つ$\{G \cup \{\alpha\}\} \models \beta \iff \{G \cup \{\neg\beta\}\} \models \neg\alpha$

- 演繹定理とその系は，帰納推論の一つの基礎を与えることになる

ホーン節集合に対する推論

前向き推論と後ろ向き推論

- ホーン節（horn clause）：正リテラルを高々が一つ含む節（リテラルの選言）
  - 前提を正リテラルの連言，帰結を高々一つの正リテラルの含意文で表現可能
    - $(h \vee \neg b_1 \vee \cdots \vee \neg b_n)$ は，含意文$b_1 \wedge \cdots \wedge b_n \Rightarrow h$ で表現できる
  - 事実節（fact clause）：負リテラルを持たない（n=0）の節
  - 確定節（definite clause）：正リテラルを丁度一つ含む節
    - 確定節は，「確定的なルール」と見做すことが可能

- 対象をホーン節の集合で表現できる場合，より効率的な推論メカニズムが適用可能
  - 前向き推論（forward reasoning, forward chaining）
  - 後向き推論（backward reasoning , backward chaining ）

  - いずれの推論も．．．
    - 単一の命題記号qが，ホーン節集合から伴意されるかを調べる
      - 確定節集合を対象としている．文$(h \vee \neg b_1 \vee \cdots \vee \neg b_n)$の伴意関係を調べるには？
        → 文$(q \vee \neg h)$ を知識ベースに追加し，”q”の伴意関係を調べる

    - 推論ステップが自明であり，推論過程の把握が容易
    - 知識ベースのサイズに対して線形時間で伴意関係を決定できる

単純な前向き推論アルゴリズム

※余力があったら実装してみよう

- 以下の操作を，新しい帰結hが導出されなくなるまで繰り返す
  - 各ルール（含意文）$b_1 \wedge \cdots \wedge b_n \Rightarrow h$ に対し，すべての条件$b_1 \wedge \cdots \wedge b_n$が導出されたら，帰結hを導出する．※モーダスポーネンスを行っていることに相当
- もちろん，pが導出した時点でやめることも可能

KB：ホーン節集合，p：証明したい原子文

0: $fact_p := \emptyset$
1: $fact :=$ KB 中の事実節の全体集合
2: while($fact_p \neq fact$)
3: $\quad fact_p := fact$
4: $\quad fact := fact \cup \{ h \mid (h \vee \neg b_1, \ldots \vee \neg b_n) \in KB, \{b_1, \ldots, b_n\} \subseteq fact\}$
5: end while
6: return $p \in KB$

KBからqを求める例

$$KB = \begin{cases} a \\ b \\ a \wedge b \Rightarrow l \\ a \wedge p \Rightarrow l \\ b \wedge l \Rightarrow m \\ l \wedge m \Rightarrow p \\ p \Rightarrow q \end{cases}$$

$$\begin{aligned} fact &:= \{a, b\} \\ fact &:= \{a, b\} \cup \{l\} && \because a \wedge b \Rightarrow l \\ fact &:= \{a, b, l\} \cup \{m\} && \because b \wedge l \Rightarrow m \\ fact &:= \{a, b, l, m\} \cup \{p\} && \because l \wedge m \Rightarrow p \\ fact &:= \{a, b, l, m, p\} \cup \{q\} && \because p \Rightarrow q \end{aligned}$$

単純な後向き推論アルゴリズム

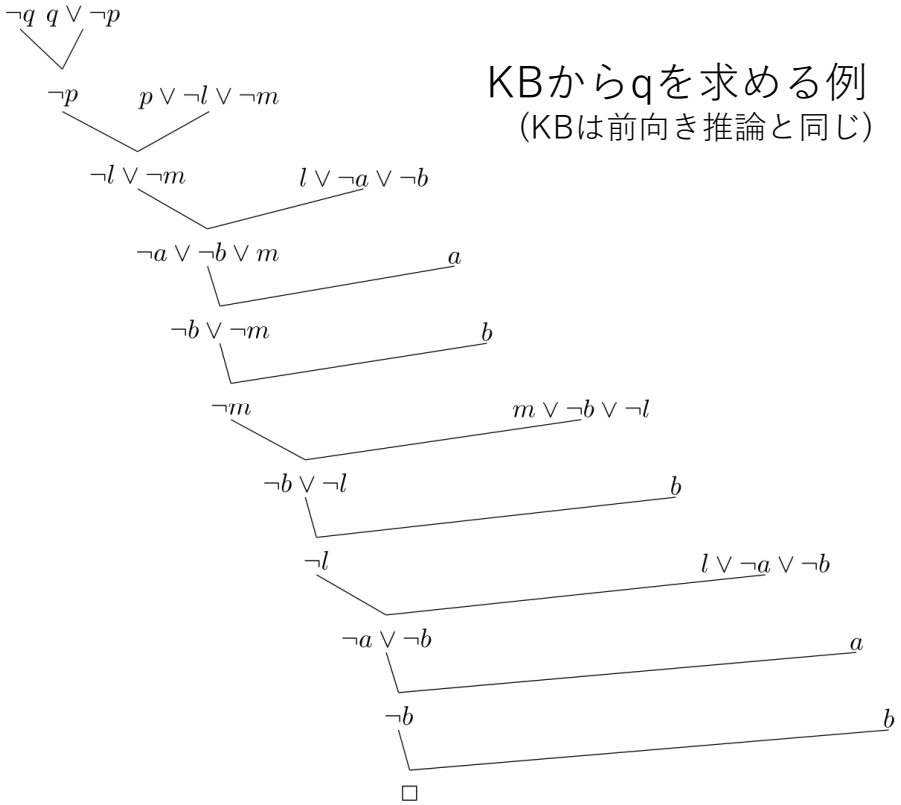
※余力があったら実装してみよう

- 融合法による反駁証明
  - 最初の融合操作には，証明したい原子文の否定を用いる
  - 以降の融合操作には，直前の融合操作から得られる融合節を用いる
  - 矛盾が導出されれば，証明完了
  - （上記に「最左リテラルから順番に」を追加するとPrologの動作となります）

KB：ホーン節集合，p：証明したい原子文

```
0:  return solve(¬p, KB)

1:  boolean solve(c, KB)
2:      for each c' ∈ KB s.t. c と c' は融合可能
3:          r := c と c' の融合節
4:          if( solve(r, KB) ) then return true
5:      end for
6:      return false
```



KBからqを求める例
（KBは前向き推論と同じ）

（参考）より洗練された前向き推論アルゴリズム

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*
  inputs: KB, the knowledge base, a set of propositional definite clauses
    q, the query, a proposition symbol
  count ← a table, where *count*[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where *inferred*[s] is initially *false* for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while *queue* is not empty **do**
    p ← POP(*queue*)
    if $p = q$ **then return** *true*
    if *inferred*[p] = *false* **then**
      inferred[p] ← *true*
      for each clause c in KB where p is in c.PREMISE **do**
        decrement *count*[c]
        if *count*[c] = 0 **then** add c.CONCLUSION to *queue*
  return *false*

Figure 7.15 The forward-chaining algorithm for propositional logic. The *agenda* keeps track of symbols known to be true but not yet "processed." The *count* table keeps track of how many premises of each implication are not yet proven. Whenever a new symbol p from the agenda is processed, the count is reduced by one for each implication in whose premise p appears (easily identified in constant time with appropriate indexing.) If a count reaches zero, all the premises of the implication are known, so its conclusion can be added to the agenda. Finally, we need to keep track of which symbols have been processed; a symbol that is already in the set of inferred symbols need not be added to the agenda again. This avoids redundant work and prevents loops caused by implications such as $P \Rightarrow Q$ and $Q \Rightarrow P$.

まとめ：今回の授業の内容

- 命題論理の推論
  - 推論と伴意
  - 推論手続き
    - モーダスポーネンス
    - 公理的証明法
    - 融合法

- ホーン節集合に対する推論
  - 前向き推論
  - 後ろ向き推論