論理と計算

第４回
充足可能性問題

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

充足可能性問題

充足可能性問題

- SAT，satisfiability problem
- 与えられた命題論理式を「真」とするような，命題記号への真偽値割り当てがあるかを判定する問題．多くの場合，その真理値割り当てを求める

例題 $$(p_1 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_2 \vee \neg p_3) \wedge (p_1 \vee p_2 \vee \neg p_3)$$

- 命題記号：p1, p2, p3
- ∨ ：または（|| ），∧：かつ（&&），¬：否定（!）
- 求めるもの： 式を真にするためのp1, p2, p3への真偽値割り当て
  - 例えば，：p1 = true, p2 = true, p3 = true とすると．．． 全体は偽となる

$$\overbrace{(\overset{t}{p_1} \vee \overset{t}{p_3})}^{t} \wedge \overbrace{(\underbrace{\neg \overset{t}{p_1}}_{f} \vee \overset{t}{p_2})}^{t} \wedge \overbrace{(\underbrace{\neg \overset{t}{p_2}}_{f} \vee \underbrace{\neg \overset{t}{p_3}}_{f})}^{f} \wedge \overbrace{(\overset{t}{p_1} \vee \overset{t}{p_2} \vee \underbrace{\neg \overset{t}{p_3}}_{f})}^{t}$$

充足可能性問題（Boolean Satisfiability Testing）

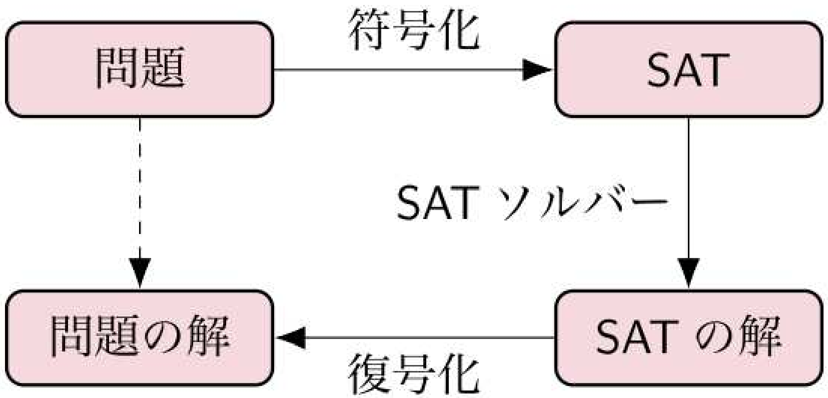
- 与えられた命題論理式を「真」にする命題変数への割り当て（＝モデル）が存在するかを判定する問題
  - SATは，NP完全であることが最初に証明された問題
  - 「SAT問題は，非常に多くの問題を解くための鍵となることから，明らかにkiller appだ」（The Art of Computer Programmingの序文 by Knuth先生(チューリング賞受賞者)）
  - 多くの場合，sat/unsatに加えてモデルも表示する

- 確認
  - 解釈＝命題変数に対する真理値の割り当て
  - モデル＝論理式が真となる解釈
  - 命題論理式の分類
    - 恒真（トートロジー）：すべての解釈で真
    - 充足可能：真にする解釈が存在する
    - 恒偽（矛盾）：すべての解釈で偽
- SAT：問題設定はシンプル
  - モデルが存在するかしないかを判定する
  - モデルが存在しない（恒偽）
    - 背理法を用いた伴意式の証明にも利用可能

SAT型システム

- 問題をSATへ変換し，SATソルバーを用いて解くシステム
  - 符号化：SATへの変換 ←→復号化：SAT解からの変換
  - SATソルバー：SAT問題を解くソフトウェア

- 問題毎に特化したアルゴリズムを使う（作る）のではなく，SATに解いてもらう



- SATへの入力
  - 連言標準形（CNF）＝節の連言（AND結合）＝節集合
    - 節（clause）：リテラルの選言（OR結合）
    - リテラル（Literal）：命題変数，命題変数の否定

SAT 型システムの成功事例

- プランニング (SATPLAN, Blackbox) [Kautz+, 1992] ▸ web
- 自動テストパターン生成 [Larrabee, 1992]
- ジョブショップスケジューリング [Crawford+, 1994]
- 有界モデル検査 [Biere, 2009]
- ソフトウェア検証 (Alloy) [Jackson, 2006] ▸ web
- 書換えシステム (AProVE) [Jürgen+, 2004] ▸ web
- インテル社の i7 プロセッサの検証 [Kaivola+, 2009]
- Eclipse のコンポーネント間の依存解析 [Le Berre+, 2009] ▸ web
- 解集合プログラミング (clasp) [Gebser+, 2012] ▸ web
- Linux のパッケージマネージャである DNF の依存性解決 ▸ web
- 制約充足問題 (Sugar) [Tamura+, 2009] ▸ web
    - オープンショップスケジューリング問題の未解決問題の求解 [Tamura+, 2009]
    - パッキング配列問題の未解決問題の求解 [則武+, 2013]
- この他ペトリネットの検証，システム生物学，グラフ理論の問題などにも応用されている [Ogata+, 2004, Soh+, 2010, Soh+, 2014].

https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdfより

SAT に関連する特に最近の話題

- **2014 年 2 月**
  - Erdös Discrepancy Conjecture の $C = 2$ の場合の解決 [Konev+, 2014]
- **2015 年 12 月**
  - The Art of Computer Programming 最新分冊で SAT が取り上げられる [Knuth, 2015]
- **2016 年 5 月**
  - Boolean Pythagorean Triple 問題の解決 [Heule+, 2016]
    - ★ 発表当時 Nature 誌へこの話題が掲載される ▸ web
- **2017 年 2 月**
  - SHA-1 の衝突メッセージ作成の過程で SAT ソルバーが使われる ▸ web

https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdfより

SHA-1 ハッシュ値の衝突

- 2017 年 2 月に同一の SHA-1 (Security Hash Algorithm 1) ハッシュ値をもつ 2 つのファイルが実際に作成された．

CV_0	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
$M_1^{(1)}$			7f	46	dc	93	a6	b6	7e	01	3b	02	9a	aa	1d	b2	56	0b		
			45	ca	67	d6	88	c7	f8	4b	8c	4c	79	1f	e0	2b	3d	f6		
			14	f8	6d	b1	69	09	01	c5	6b	45	c1	53	0a	fe	df	b7		
			60	38	e9	72	72	2f	e7	ad	72	8f	0e	49	04	e0	46	c2		
$CV_1^{(1)}$	8d	64	d6	17	ff	ed	53	52	eb	c8	59	15	5e	c7	eb	34	f3	8a	5a	7b
$M_2^{(1)}$			30	57	0f	e9	d4	13	98	ab	e1	2e	f5	bc	94	2b	e3	35		
			42	a4	80	2d	98	b5	d7	0f	2a	33	2e	c3	7f	ac	35	14		
			e7	4d	dc	0f	2c	c1	a8	74	cd	0c	78	30	5a	21	56	64		
			61	30	97	89	60	6b	d0	bf	3f	98	cd	a8	04	46	29	a1		
CV_2	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5

CV_0	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
$M_1^{(2)}$			73	46	dc	91	66	b6	7e	11	8f	02	9a	b6	21	b2	56	0f		
			f9	ca	67	cc	a8	c7	f8	5b	a8	4c	79	03	0c	2b	3d	e2		
			18	f8	6d	b3	a9	09	01	d5	df	45	c1	4f	26	fe	df	b3		
			dc	38	e9	6a	c2	2f	e7	bd	72	8f	0e	45	bc	e0	46	d2		
$CV_1^{(2)}$	8d	64	c8	21	ff	ed	52	e2	eb	c8	59	15	5e	c7	eb	36	73	8a	5a	7b
$M_2^{(2)}$			3c	57	0f	eb	14	13	98	bb	55	2e	f5	a0	a8	2b	e3	31		
			fe	a4	80	37	b8	b5	d7	1f	0e	33	2e	df	93	ac	35	00		
			eb	4d	dc	0d	ec	c1	a8	64	79	0c	78	2c	76	21	56	60		
			dd	30	97	91	d0	6b	d0	af	3f	98	cd	a4	bc	46	29	b1		
CV_2	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5

実際の衝突メッセージ． web より引用

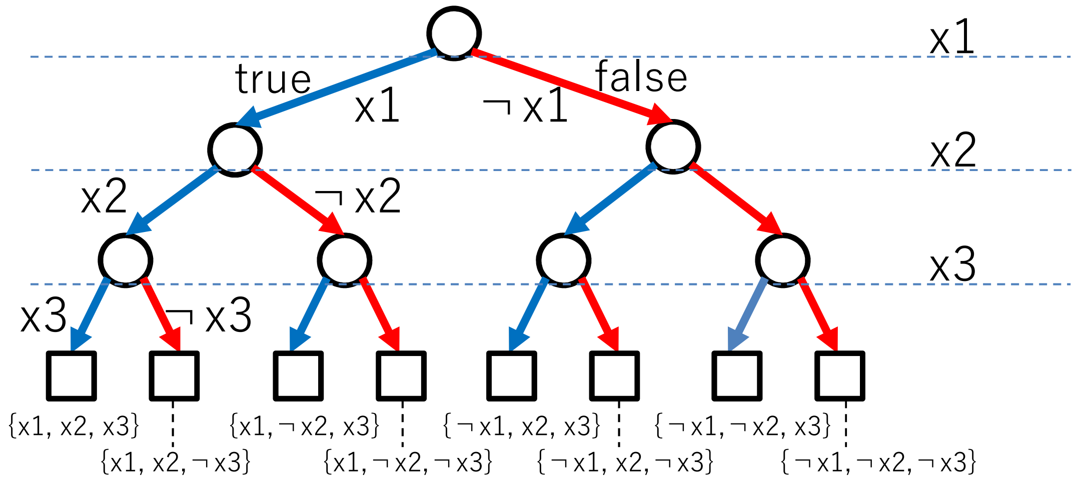
- 報告したのは Google とオランダ国立数学・計算機科学研究センター（CWI）のチームで，2 番目の near-collision ブロックペアを見つける計算の一部で SAT ソルバーが使われた．

https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdfより

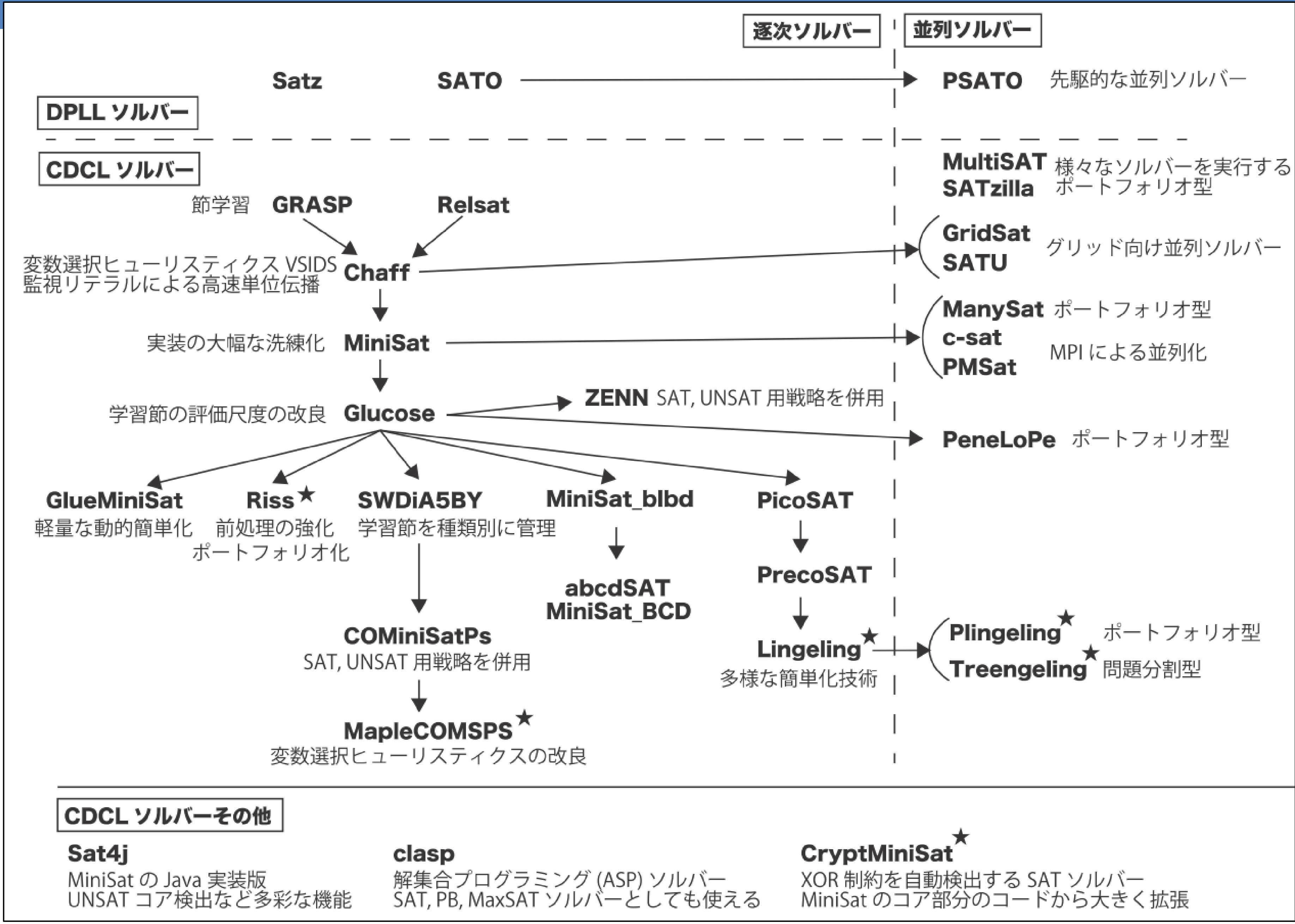
基本アルゴリズム

SATの基本アルゴリズム

- 真となる割り当てを調べる：原理的には，真理値表を求める場合と同じ
  - 真理値表を表示するプログラム（TruthTable.pde）を確認してみよう
    - 各命題記号に対し，trueを割り当てる場合，falseを割り当てる場合の2通り
    - 命題記号をノード，割り当てを分岐とする二分木の深さ優先探索



- 真理値表：すべての命題変数に値を割り当てた後，解釈を表示
- SAT：モデル（真となる解釈）のみが必要
  - SAT/UNSATだけで良いなら，一つのモデルが見つかったら終了すればよい
  - モデルとならないことが分かった時点でその先の探索（割り当て）を打ち切る
  - 更なる工夫
    - 単位伝播：「必然的な真偽値割り当て」を行うことで分岐を減らす
    - 節学習：「探索中に得た情報」を利用する



宋 他，SATソルバーの最新動向と利用技術，コンピュータソフトウェア，35(4):72-92, 2018より

DPLL（Davis-Putnam-Logemann-Loveland）

- 二分木の深さ優先探索＋単位伝播
  - ＋早期停止＋純粋記号ヒューリスティクス

clauses ：節集合
symbols：命題変数の集合
model：（部分）真偽値割り当て

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, { })

このアルゴリズムは純粋にsat/unsatを返す
（modelを表示すれば割り当ては分かる）

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
早期停止
  P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*) 純粋記号ヒューリスティクス
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*) 単位伝播
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, *model* ∪ {P=*true*}) **or**
      DPLL(*clauses*, *rest*, *model* ∪ {P=*false*})

二分木の深さ優先探索

S. Russell, P. Norvig 著，古川康一監訳：エージェントアプローチ人工知能，7章より

$$\overbrace{\underbrace{(\overbrace{\alpha_1}^{\text{リテラル}} \vee \dots \vee \alpha_m)}_{\substack{\text{最低一つのリテラルが} \\ \text{true であればよい}}} \wedge \dots \wedge \underbrace{(\beta_1 \vee \dots \vee \beta_n)}_{\text{節}}}^{\text{すべての節が true になる必要がある}}$$

早期停止

- 部分的に出来上がったモデルに対して真偽を検出する
- すべての変数に真偽値を割り当てる前に，SAT/UNSATが決定できる場合がある

- SAT/UNSATの条件
  - SATであるためには，すべての節がtrueである必要がある

    ➔節はリテラルの選言；最低一つのリテラルがtrueであれば節はtrue
  - falseとなる節があればSATにはならない

- 例：(A∨B)∧(A∨C)
  - A = true とすると，(A∨B)(A∨C)は共にtrueとなる ➔ SATであることが確定
    - B, C への値の割り当てを行う前にSATが確定する.
    - また，B,Cの割り当ては任意でOK
  - A =false, B = falseとすると，(A∨B)はfalseとなる

    ➔ (現在の割り当てでは)satにならないことが確定
    - Cへの値の割り当てを行う前に，SATにならないことが確定する.
    - このとき，バックトラックしてA, B の真偽値の割り当てをやり直す

純粋記号ヒューリスティクス

- 純粋な記号（pure symbol）：すべての節で同じ符号を持つ命題記号
  - 現在の割り当てにおいて，trueが確定している節は無視して考える
  - 例：$(A \lor \neg B) \land (\neg B \lor \neg C) \land (A \lor C)$　におけるAとB
    - Aは正リテラルとしてのみ現れる．Bは負リテラルとしてのみ現れる

- 「純粋な記号のリテラルをtrueにする」モデルが存在する
  - 純粋な記号のリテラルに対するtrue割り当てが節をfalseにすることはない
    - 節中の最低一つのリテラルがtrueであれば，節自体はtrueになる
    - 純粋な記号にtrueを割り当てれば，節がtrueとなりその節を無視することができる
  - 例：$(A \lor \neg B) \land (\neg B \lor \neg C) \land (A \lor C)$
    - A=true, B=false ($\neg$B=true)を割り当てる➔すべての節がtrueになりsat

- 現在の割り当てにおいて，trueが確定している節は無視して考える
  - trueが確定している節は，真理値未割当て変数に任意の割り当てをしてもtrueのまま
    - 既に最低一つのリテラルがtrueなので，節がtrueになっている．
    - 残りのリテラルがtrueになってもfalseになっても，節の真理値は変わらない
  - 例：B=falseの下での$(A \lor \neg B) \land (\neg B \lor \neg C) \land (A \lor C)$➔ $(A \lor \neg B) \land (\neg B \lor \neg C) \land (A \lor C)$
    - A，Cも純粋なリテラルとなる

単位伝播（unit propagation）

- 単位節（Unit clause）
  - 一つを除いてすべてのリテラルにfalseが割り当てられている節
  - 例：B=falseにおける（B∨¬C）
- 充足可能であるためには，すべての節がtrueである必要がある
  - 当然，単位節もtrueとなる必要がある
  - 単位節に現れる（値未割当ての）リテラルはtrueにしなければならない
    - 他のリテラルはすべてfalseなので，残ったリテラルはtrueでないと節がtrueにならない
  - ➔ 単位節中のリテラルは（自動的に）真偽値が決まる
    - 例：B=falseにおける(B∨¬C)なる単位節から割当てC=falseが確定
  - ➔すべての単位節中のリテラルに，最初に値を割り当てる
    - この時点で矛盾が生じる＝それまでの割り当てが不適 → バックトラック
- 単位節の伝播：一つの単位節に真理値割り当てが，別の単位節を生む
  - B=falseにおける(B∨¬C)，(C∨A)
    - 単位節(B∨¬C)より，C=falseを割り当てる．
    - 結果{B=false, C=false}となり，（C∨A)も単位節となる．A=trueが自動的に決まる

DPLL（Davis-Putnam-Logemann-Loveland）

- 二分木の深さ優先探索＋単位伝播
  - ＋早期停止＋純粋記号ヒューリスティクス

clauses ：節集合
symbols：命題変数の集合
model：（部分）真偽値割り当て

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of s
  $symbols \leftarrow$ a list of the proposition symbols in s
  return DPLL($clauses, symbols$, { })

このアルゴリズムは純粋にsat/unsatを返す
（modelを表示すれば割り当ては分かる）

function DPLL($clauses, symbols, model$) **returns** *true* or *false*

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
早期停止
  $P, value \leftarrow$ FIND-PURE-SYMBOL($symbols, clauses, model$) 純粋記号ヒューリスティクス
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P, value \leftarrow$ FIND-UNIT-CLAUSE($clauses, model$) 単位伝播
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
  return DPLL($clauses, rest, model \cup \{P{=}true\}$) **or**
      DPLL($clauses, rest, model \cup \{P{=}false\}$))

二分木の深さ優先探索

S. Russell, P. Norvig 著，古川康一監訳：エージェントアプローチ人工知能，7章より

DPLL（Davis-Putnam-Logemann-Loveland）

procedure $DPLL(Th:$ set of clauses):
  if Th is empty
    return *true*
  else if Th contains an empty clause
    return *false*
  else if there exists a literal l in Th such that $\bar{l}$ does not appear in Th
    $Th' := Th - \{C \mid C \in Th \text{ and } C \text{ contains } l\}$
    return DPLL(Th')
  else if Th contains a unit clause (a clause with one literal l)
    $Th' := \{C - \{\bar{l}\} \mid C \in Th \text{ and } C \text{ does not contain } l\}$
    return DPLL(Th')
  else choose a proposition l in Th
    $Th_t := \{C - \{\bar{l}\} \mid C \in Th \text{ and } C \text{ does not contain } l\}$
    $Th_f := \{C - \{l\} \mid C \in Th \text{ and } C \text{ does not contain } \bar{l}\}$
    return $DPLL(Th_t) \vee DPLL(Th_f)$

Th : 節集合
C : 節（＝ リテラル集合）

*l*にtrueを割り当てる．
*l*を含む節はtrueとなるので，
以降考慮する必要はない

*l*はtrue．従って*l*を含む節もtrueで無視できる
一方，¬lはfalse．
以降¬lは，¬lを含む節のtrue,falseに影響なし

リテラル*l*をfalseを割当て．
¬*l*を含む節はtrueとなる．
*l*はfalse．*l*を含む節の真理値に影響なし

Luc De Raedt et. al, Statistical Relational Artificial Intelligence, Morgan & Claypool Publishers, 2016より

「純粋記号ヒューリスティクスなしで」同じことを行っています．内容を確認してみましょう

DPLL（Davis-Putnam-Logemann-Loveland）

ψ ：論理集合
□ ：矛盾

DPLL(CNF ψ)
begin
  if ψ is empty **then** return SAT;
  ψ := **UnitPropagation**(ψ);
  if □ exists in ψ **then** return UNSAT;
  Choose variable x in ψ by some hueristics;
  if DPLL($\psi \wedge x$) returns SAT **then** return SAT;
  else return the result of **DPLL**($\psi \wedge \overline{x}$)
end

xにtrueを割り当てる

xにfalseを割り当てる

図 1 DPLL algorithm

UnitPropagation(CNF ψ)
begin
  while □ doesn't exist and a unit clause l exists in ψ **do**
   Assign 1 to l and simplify ψ;
  return ψ
end

simplify：
(1) ψ から，「trueが確定した節」を除去
(2) 各節から，「¬l」を除去

井上克巳，田村直之：SATソルバーの基礎，人工知能学会誌，25(1):57-67, 2010より

CDCL（Conflict Driven Clause Learning）

- 矛盾からの節学習
  - 矛盾が生じたときに，その「原因」を解析する．
  - 「原因」を否定した節を論理式に追加する．

- 例：$(x1 \lor x2), (\neg x2 \lor \neg x3 \lor \neg x4), (x1 \lor x4), (\neg x2 \lor x3 \lor \neg x4)$
  - $x2 = true, x1 = true, x4 = true$ の場合を考える．
  - $(\neg x2 \lor \neg x3 \lor \neg x4)$と$(\neg x2 \lor x3 \lor \neg x4)$は同時にはtrueに出来ない
    - （false ∨ ￢x3 ∨ false）と（false ∨ x3 ∨ false）なので，x3をtrueにしてもfalseにしてもだめ

  - 原因： x2とx4を同時にtrueにしたこと．$(x2 \land x4)$が原因
  - 原因の否定を追加：$\neg (x2 \land x4) \rightarrow (\neg x2 \lor \neg x4)$を追加する
    - これにより，x2をtrueとしたとき，x4のfalseが確定（単位伝播）

- 矛盾の解析：含意グラフ（単位伝播による含意関係を表すグラフ）の分析
  - 詳細は「鍋島英知，宋剛秀：高速SATソルバーの原理,人工知能学会誌，25(1):68-76, 2010 」参照

確率的ソルバーとSATの拡張

確率的ソルバー（stochastic solver）

- 系統的ソルバー：系統的探索を行う完全なアルゴリズムに基づくソルバー
  - DPLLなど
  - 充足可能・不能を判定可能

- 確率的ソルバー：確率的局所探索を行う不完全なアルゴリズムに基づく
  - WalkSatなど
  - 充足可能性は判定可能／充足不能性は一般に判断できない（その分高速）

- WalkSat：確率的局所探索
  - 具体的なアルゴリズムは次ページ

  - 局所探索：現在の割り当てと一ヶ所だけが異なる割当てを考える
  - 確率的：ランダムに，割り当てを変更する命題記号を選択する

  - 繰り返し上限を∞にすると，充足可能な場合は 必ずSATと判定できる
  - 様々な変種が考えられる

WalkSat

S. Russell, P. Norvig 著，古川康一監訳：エージェントアプローチ人工知能，7章より

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of value flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*　ランダム割当てでスタート
  for each *i* = 1 **to** *max_flips* **do**　繰り返しの上限数を与える
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*　false節の選択
    if RANDOM(0, 1) ≤ *p* **then**　ランダムウォーク
      flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses　制約違反最小化
  return *failure*

- 命題記号に対するランダムな真偽値割り当てからスタート
- 繰り返し上限に達する or モデルが見つかるまで 以下を繰り返す
  - falseである節をランダムに選択
  - 確率pでランダムウォーク
    - 「節中の命題記号をランダムに選択し，その真偽値を反転する」
  - 確率(1-p)で制約違反最小化
    - 「節中の命題記号の内，『反転することでtrueとなる節が最大となる命題記号』の真偽値を反転する」

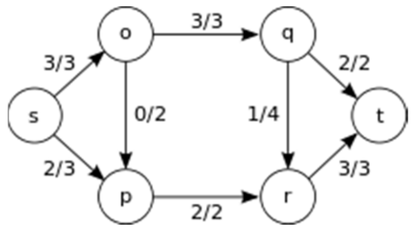
SATの発展：MaxSat（Maximal satisfiability）

- SATを最適化問題に拡張したもの
- モデルに優劣をつけ，最適なモデルを出力する
  - 元のSATではモデルの優劣は考えない

- SATはすべての節を真にすることを要求する（例外を許さない）
- MaxSatでは，ハード節とソフト節を考える
  - ソフト節：満たさなくても良い節の集合．重みを用いて重要性を表す
  - ハード節：真とならなければいけない節の集合．重みはソフト節の最大値よりも大きい

- 最適解
  - すべてのハード節を満たし，満たされるソフト節の重みの総和を最大にする割り当て

- 多岐の応用
  - スケジューリング，プラニング，ゲーム理論，ルーティング，バイオインフォマティクス，ハードウェア・ソフトウェアのデバッグ，など
    - 最大フロー問題：単一の始点から終点へのフローネットワークで最大となるフローを求める問題



https://ja.wikipedia.org/wiki/%E6%9C%80%E5%A4%A7%E3%83%95%E3%83%AD%E3%83%BC%E5%95%8F%E9%A1%8C より

演習：SATソルバーを動かしてみよう

SATソルバーを動かしてみよう

- clingo：解集合プログラミングシステム
  - https://github.com/potassco/clingo/releases/
  - gringo : グラウンダー（変数の基礎化）
  - claps : ソルバ　（SATソルバーとしても利用可能）

・一回目資料より．．．

【演習環境】

- レポートは，Latexを用いて作成しPDFファイルを提出すること
- 一部の例題に Processing プログラムを用いる
- SAT演習にはclasp，ASP演習にはclingo，ILP演習にはILASPを用いる
  - 参考：https://doc.ilasp.com/installation.html　ですべてインストール可能
    - 管理者として実行しましょう
  - Ubuntu on VirtualBox / WSL2 / Mac のいずれかを利用する

SATソルバーを動かしてみよう

- 入力表現：DIMACS CNF
  - http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/satformat.ps
  - 先頭行： p cnf #ofN #ofC ; #ofNは命題変数の数, #ofCは節の数
  - 各行：節 ；節は0で終わる．各リテラルは正整数，¬は-で表現
  - ；（セミコロン）；コメントアウト

p cnf 3 4	; 命題変数の数␣節の数
1 2 3 0	; $p_1 \vee p_2 \vee p_3$
-1 -2 0	; $\neg p_1 \vee \neg p_2$
-1 -3 0	; $\neg p_1 \vee \neg p_3$
-2 -3 0	; $\neg p_2 \vee \neg p_3$

- 実行方法：
  - $ ./clasp [オプション] ファイル名
    - オプションなしだと，最初に見つけた解を表示
    - オプション位置に “0” を入れるとすべての解を表示

- 試してみよう
  1. 右上のCNF
  2. (x1∨x2)∧(¬x2∨¬x3∨¬x4)∧(x1∨x4)∧(¬x2∨x3∨¬x4)
  3. (p1∨p3)∧（¬p1∨p2)∧（¬p2∨¬p3)∧（p1∨p2∨¬p3)

SAT符号化して問題を解いてみよう

入試問題：2016年度青山学院大学(全学部日程)

- A，B，Cの3名がいて，正直者が2人，残りの1人が嘘つきである．
- そして3名とも誰が正直者で，誰が嘘つきかは知っているとする．
- ここで，正直者とは常に真実をいう人，嘘つきとは常に真実と反対のことをいう人である．
- このとき，つぎのようなA，B，Cの証言が得られた．
  - Aの証言：Cは嘘つきである．
  - Bの証言：Aは正直者である．
  - Cの証言：Bは嘘つきである．
- 誰が嘘つきかを決定しよう．

何を命題変数にしますか？
「嘘つきは一人」をどう表しますか？

今回はSAT符号化という大げさなものではなく，単なる命題論理表現をCNFに変換する

SAT符号化して問題を解いてみよう(1)

命題論理（CNF）で表現してみよう

- 誰が嘘つきかを決定しよう

  ➔ 「嘘つき」なら真，（反対の）「正直者」なら偽を取る命題変数

  L_A : Aは嘘つきである / L_B : Bは嘘つきである / L_C : Cは嘘つきである

- A，B，Cの3名がいて，正直者が2人，残りの1人が嘘つきである．

  ➔ 誰か一人は嘘つき：（L_A∨L_B∨L_C）

  ➔ 同時に2名が嘘つきにはならない

  ➔ AとBが同時に嘘つきではない：¬（L_A∧L_B） より（¬L_A∨¬L_B）

  ➔ BとCが同時に嘘つきではない：¬（L_B∧L_C） より（¬L_B∨¬L_C）

  ➔ CとAが同時に嘘つきではない：¬（L_C∧L_A） より（¬L_C∨¬L_A）

- Aの証言：Cは嘘つきである

  ➔Aが正直者⇔Cは嘘つき：¬L_A⇔L_C より （L_A∨L_C） ∧ （¬L_A∨¬L_C）

  ➔Aが嘘つき⇔Cは正直者：L_A⇔¬L_C より （L_A∨L_C） ∧ （¬L_A∨¬L_C）

- Bの証言：Aは正直者である．

  ➔Bが正直者⇔Aは正直者：¬L_B⇔¬L_A より （L_B ∨¬L_A） ∧ （L_A∨¬L_B）

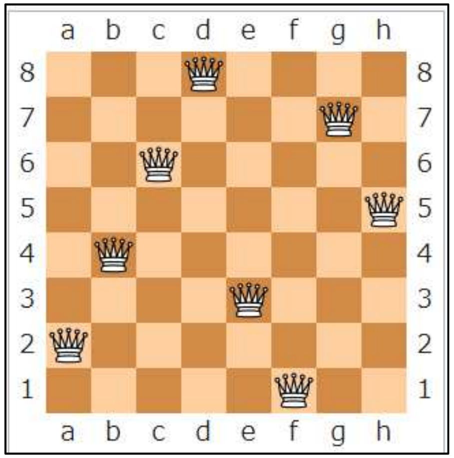
- Cの証言：Bは嘘つきである．

  ➔Cが正直者⇔Bは嘘つき：¬L_C⇔L_B より （L_C∨L_B） ∧ （¬L_C∨¬L_B）

- 動かしてみよう：SAT/Lair_or_honest/ Liar_or_honest.cnf

N人の女王

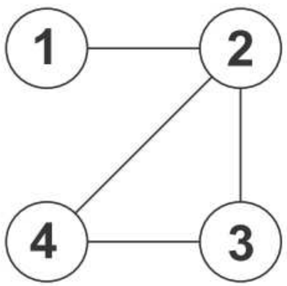
- n個のクイーンを，n × nのチェス盤に，お互いに取られないように並べる
- SAT符号化
  - 命題記号：q[i, j]「i行j列にクイーンがある」
  - 制約１：各行でクイーンは一つ
    - $(q[1,1] \lor \cdots \lor q[1,n]) \land \cdots (q[n,1] \lor \cdots \lor q[n,n])$
    - $(\neg q[1,1] \lor \neg q[1,2]) \land (\neg q[1,n-1] \lor \neg q[1,n]) \land$
      $\cdots \land (\neg q[n,1] \lor \neg q[n,2]) \land (\neg q[n,n-1] \lor \neg q[n,n])$
  - 制約2：各列でクイーンの重複はNG
    - $(\neg q[1,1] \lor \neg q[2,1]) \land (\neg q[n-1,1] \lor \neg q[n,1]) \land$
      $\cdots \land (\neg q[1,n] \lor \neg q[2,n]) \land (\neg q[n-1,n] \lor \neg q[n,n])$
  - 制約3：斜めでクイーンの重複はNG
    - $(\neg q[i,j] \lor \neg q[i+k, j \pm k])$　マス(i, j)の右（左）斜め下
- 動かしてみよう：SAT/nQueen/queen_8.cnf（N=8クイーン）



https://ja.wikipedia.org/wiki/エイト・クイーン より

グラフの頂点彩色

- 隣接する頂点同士が同じ色にならないように全頂点を彩色する
  - どんな（平面）グラフも，４色で塗り分けることができる
- SAT符号化（色の数をcとする）
  - 命題記号：p_{ i, k }「頂点 I の色は k である」
  - 制約１：各頂点の色は一つ
    - 各頂点の色は， j = 1 … c のいずれか． （p_{i,1} ∨… ∨p_{i,c}）
    - p_{ i, k_1 }とp_{ i, k_2 } は同時に成り立たない．
      - （¬p_{ i, 1 }∨¬p_{ i, 2 }）∧…∧ （¬p_{ i, c-1 }∨¬p_{ i, c }）
  - 制約2：隣接する頂点は異なる色
    - 隣接頂点(i,j) が同時に色 k になることはない
      - （¬p_{ i, 1 }∨¬p_{ j, 1 }）∧…∧ （¬p_{ i, c }∨¬p_{ j, c }）



3色で塗り分ける

$$
\begin{aligned}
&(\neg p_{1,1} \vee \neg p_{2,1}) \wedge (\neg p_{2,1} \vee \neg p_{3,1}) \wedge (\neg p_{2,1} \vee \neg p_{4,1}) \wedge (\neg p_{3,1} \vee \neg p_{4,1}) \wedge \\
&(\neg p_{1,2} \vee \neg p_{2,2}) \wedge (\neg p_{2,2} \vee \neg p_{3,2}) \wedge (\neg p_{2,2} \vee \neg p_{4,2}) \wedge (\neg p_{3,2} \vee \neg p_{4,2}) \wedge \\
&(\neg p_{1,3} \vee \neg p_{2,3}) \wedge (\neg p_{2,3} \vee \neg p_{3,3}) \wedge (\neg p_{2,3} \vee \neg p_{4,3}) \wedge (\neg p_{3,3} \vee \neg p_{4,3}) \wedge \\
&(p_{1,1} \vee p_{1,2} \vee p_{1,3}) \wedge (p_{2,1} \vee p_{2,2} \vee p_{2,3}) \wedge \\
&(p_{3,1} \vee p_{3,2} \vee p_{3,3}) \wedge (p_{4,1} \vee p_{4,2} \vee p_{4,3}) \wedge \\
&(\neg p_{1,1} \vee \neg p_{1,2}) \wedge (\neg p_{1,1} \vee \neg p_{1,3}) \wedge (\neg p_{1,2} \vee \neg p_{1,3}) \wedge \\
&(\neg p_{2,1} \vee \neg p_{2,2}) \wedge (\neg p_{2,1} \vee \neg p_{2,3}) \wedge (\neg p_{2,2} \vee \neg p_{2,3}) \wedge \\
&(\neg p_{3,1} \vee \neg p_{3,2}) \wedge (\neg p_{3,1} \vee \neg p_{3,3}) \wedge (\neg p_{3,2} \vee \neg p_{3,3}) \wedge \\
&(\neg p_{4,1} \vee \neg p_{4,2}) \wedge (\neg p_{4,1} \vee \neg p_{4,3}) \wedge (\neg p_{4,2} \vee \neg p_{4,3})
\end{aligned}
$$

動かしてみよう：
SAT/GraphColoring/sample_graph.cnf

宋 他「SAT型制約プログラミングシステムと周辺技術」，コンピュータソフトウェア，34(1):67-80, 2017より

数独，ナンバーリンク

- 以下の記事を読んで，自分で作成してみよう

- 田村他「SATとパズル –問題をいかにSATソルバーで解くか– 」情報処理, 57(8):710-715, 2016
  - https://ipsj.ixsq.nii.ac.jp/ej/index.php?active_action=repository_view_main_item_detail&page_id=13&block_id=8&item_id=169443&item_no=1

目次：今回の授業の内容

- 充足可能性問題
  - 定義・応用例・SAT型システム
  - 系統的ソルバーの基本アルゴリズム
    - 二分探索・DPLL・CDCL
  - 確率的ソルバー
  - SATの拡張
    - MaxSAT

- SATソルバを動かしてみよう
  - clasp
  - 簡単な例題
  - より現実的な例題