充足可能性ソルバ （SAT ソルバ） の原理

Hiromi ISHII

2024-03-10

Tsukuba Computer Mathematics Seminar 2024

自己紹介

自己紹介

- 石井大海（いしい ひろみ）

自己紹介

- 石井大海（いしい ひろみ）
- 2018 年度 筑波大学数学専攻博士後期課程修了（照井研）

自己紹介

- 石井大海（いしい ひろみ）
- 2018 年度 筑波大学数学専攻博士後期課程修了（照井研）
- 計算機合宿には 2014 年から参加

自己紹介

- 石井大海（いしい ひろみ）
- 2018 年度 筑波大学数学専攻博士後期課程修了（照井研）
- 計算機合宿には 2014 年から参加
- 現職： Haskell 製大規模数値計算ベンチャー研究開発職

自己紹介

- 石井大海（いし い ひろ み）
- 2018 年度 筑波大学数学専攻博士後期課程修了（照井研）
- 計算機合宿には 2014 年から参加
- 現職： Haskell 製大規模数値計算ベンチャー研究開発職
- 宣伝：今年 05/11, 12 に横浜でお芝居をするので興味のある方は是非観にきてください

充足可能性ソルバ （SAT ソルバ） の原理

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$ (かつ)、$\vee$ (または)、$\rightarrow$ (ならば)、$\neg$ (でない) で結んで得られる論理式

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$ (かつ)、$\vee$ (または)、$\rightarrow$ (ならば)、$\neg$ (でない) で結んで得られる論理式
  - 古典的充足可能性 : 与えられた式を真とするような、 命題変数への真偽値
    - (真) または $\times$ (偽) の割り当てが存在するか？

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$（かつ）、$\vee$（または）、$\rightarrow$（ならば）、$\neg$（でない）で結んで得られる論理式
  - 古典的充足可能性 : 与えられた式を真とするような、 命題変数への真偽値
    - （真）または $\times$（偽）の割り当てが存在するか？
- 充足可能性（SATisfiability）を略して SAT と呼ぶ。

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$（かつ）、$\vee$（または）、$\rightarrow$（ならば）、$\neg$（でない）で結んで得られる論理式
  - 古典的充足可能性 : 与えられた式を真とするような、 命題変数への真偽値
    - （真）または $\times$（偽）の割り当てが存在するか？
- 充足可能性（SATisfiability）を略して SAT と呼ぶ。
- NP- 完全：総当たりで解けるような任意の問題が SAT に帰着できる
  - 本来 SAT は判定問題だが、具体的な求解まで含めて SAT と呼ぶ事が多い

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$（かつ）、$\vee$（または）、$\rightarrow$（ならば）、$\neg$（でない）で結んで得られる論理式
  - 古典的充足可能性 : 与えられた式を真とするような、 命題変数への真偽値
    - （真）または $\times$（偽）の割り当てが存在するか？
- 充足可能性（SATisfiability）を略して SAT と呼ぶ。
- NP- 完全：総当たりで解けるような任意の問題が SAT に帰着できる
  - 本来 SAT は判定問題だが、具体的な求解まで含めて SAT と呼ぶ事が多い
- 色々な問題が SAT（やその拡張である SMT ソルバ）で解け、実用上も重要

SAT で解ける問題の例：論理パズル

問 1 (三人の島民 [1])

常に嘘だけをいう嘘吐きと、 本当のことだけをいう正直者だけが住む島で、A, B, C 三人の島民に出会った。彼らのいうことには：

- A：「B と C はどちらも正直者だ」
- B：「A は嘘吐きで、C は正直者だ」

A, B, C はそれぞれ正直者か、嘘吐きか？

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする
- 情報を命題論理式に変換して $(1) \ \wedge \ (2)$ を充足する解を求めればよい：

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする
- 情報を命題論理式に変換して $(3) \land (4)$ を充足する解を求めればよい：

$$A \longleftrightarrow B \land C \tag{5}$$

$$B \longleftrightarrow \neg A \land C \tag{6}$$

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする
- 情報を命題論理式に変換して $(5) \land (6)$ を充足する解を求めればよい：

$$A \longleftrightarrow B \land C \tag{7}$$

$$B \longleftrightarrow \neg A \land C \tag{8}$$

- 真理表を書いてみると、全員嘘吐きだとわかる。

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする
- 情報を命題論理式に変換して $(7) \land (8)$ を充足する解を求めればよい：

$$A \longleftrightarrow B \land C \tag{9}$$

$$B \longleftrightarrow \neg A \land C \tag{10}$$

- 真理表を書いてみると、全員嘘吐きだとわかる。

A	B	C	(9)	(10)	(9) ∧ (10)
○	○	○	○	×	×
○	○	×	×	×	×
○	×	○	×	○	×
○	×	×	×	○	×

A	B	C	(9)	(10)	(9) ∧ (10)
×	○	○	×	○	×
×	○	×	○	×	×
×	×	○	○	×	×
×	×	×	○	○	○

SAT で解ける問題の例：数独

- 簡単な例として、4×4 の小さな数独の問題を SAT で解くことを考える。

d_{14}	d_{24}	d_{34}	d_{44}
d_{13}	d_{23}	d_{33}	d_{43}
d_{12}	d_{22}	d_{32}	d_{42}
d_{11}	d_{21}	d_{31}	d_{41}

SAT で解ける問題の例：数独

- 簡単な例として、4×4 の小さな数独の問題を SAT で解くことを考える。
- $i, j, k \leq 4$ に対し命題変数 $P_{ij}^{=k}$ を用意する（$d_{ij} = k$ というきもち）。

d_{14}	d_{24}	d_{34}	d_{44}
d_{13}	d_{23}	d_{33}	d_{43}
d_{12}	d_{22}	d_{32}	d_{42}
d_{11}	d_{21}	d_{31}	d_{41}

SAT で解ける問題の例：数独

- 簡単な例として、4×4 の小さな数独の問題を SAT で解くことを考える。
- $i, j, k \leq 4$ に対し命題変数 $P_{ij}^{=k}$ を用意する（$d_{ij} = k$ というきもち）。
- 各マスには $1, 2, 3, 4$ のいずれかの数字一つを入れる。

d_{14}	d_{24}	d_{34}	d_{44}
d_{13}	d_{23}	d_{33}	d_{43}
d_{12}	d_{22}	d_{32}	d_{42}
d_{11}	d_{21}	d_{31}	d_{41}

$$\bigwedge_{i \leq 4} \bigwedge_{j \leq 4} \left\{ \left(P_{ij}^{=1} \vee P_{ij}^{=2} \vee P_{ij}^{=3} \vee P_{ij}^{=4} \right) \wedge \bigwedge_{k \leq 4} \bigwedge_{l \neq k} \left(P_{ij}^{=k} \longrightarrow \neg P_{ij}^{=l} \right) \right\} \tag{13}$$

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, ..., 4$ が一つずつ入る。

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, \ldots, 4$ が一つずつ入る。
  - 「各行」は次のように書ける（「各列」も i, j の役割を入れ換え同様）：

$$\bigwedge_{i \leq 4} \bigwedge_{k \leq 4} (P_{i1}^{=k} \vee P_{i2}^{=k} \vee P_{i3}^{=k} \vee P_{i4}^{=k}) \tag{15}$$

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, \ldots, 4$ が一つずつ入る。
  - 「各行」は次のように書ける（「各列」も i, j の役割を入れ換え同様）：

$$\bigwedge_{i \leq 4} \bigwedge_{k \leq 4} (P_{i1}^{=k} \vee P_{i2}^{=k} \vee P_{i3}^{=k} \vee P_{i4}^{=k}) \tag{16}$$

  - 演習問題：「一意性」の条件が要らない理由を考えてみよう。

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, \ldots, 4$ が一つずつ入る。
  - 「各行」は次のように書ける（「各列」も i, j の役割を入れ換え同様）：

$$\bigwedge_{i \leq 4} \bigwedge_{k \leq 4} (P_{i1}^{=k} \vee P_{i2}^{=k} \vee P_{i3}^{=k} \vee P_{i4}^{=k}) \tag{17}$$

  - 演習問題：「一意性」の条件が要らない理由を考えてみよう。
  - 演習問題：「各ブロック」の条件を書き下してみよう。

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, \ldots, 4$ が一つずつ入る。
  - 「各行」は次のように書ける（「各列」も i, j の役割を入れ換え同様）：

$$\bigwedge_{i \leq 4} \bigwedge_{k \leq 4} (P_{i1}^{=k} \vee P_{i2}^{=k} \vee P_{i3}^{=k} \vee P_{i4}^{=k}) \tag{18}$$

  - 演習問題：「一意性」の条件が要らない理由を考えてみよう。
  - 演習問題：「各ブロック」の条件を書き下してみよう。
- あとは盤面の情報を $P_{ij}^{=k}$ の and で与えれば個別の問題を解ける！

SAT で解ける問題の例：命題論理の定理自動証明

- SAT ソルバは定理の自動証明にも使える

SAT で解ける問題の例：命題論理の定理自動証明

- SAT ソルバは定理の自動証明にも使える

定理 4 (古典命題論理の完全性定理および健全性定理 (Gödel))

任意の古典命題論理式 φ に対し、φ がどんな真偽値の割り当てに対しても真であることと、φ が証明可能であることは同値である。

SAT で解ける問題の例：命題論理の定理自動証明

- SAT ソルバは定理の自動証明にも使える

定理 6 (古典命題論理の完全性定理および健全性定理 (Gödel))

任意の古典命題論理式 φ に対し、φ がどんな真偽値の割り当てに対しても真であることと、φ が証明可能であることは同値である。

系 7

任意の古典命題論理式 φ が証明可能である必要十分条件は、$\neg\varphi$ が充足可能でないことである。

SAT で解ける問題の例：命題論理の定理自動証明

- SAT ソルバは定理の自動証明にも使える

定理 8 (古典命題論理の完全性定理および健全性定理 (Gödel))

任意の古典命題論理式 φ に対し、φ がどんな真偽値の割り当てに対しても真であることと、φ が証明可能であることは同値である。

系 9

任意の古典命題論理式 φ が証明可能である必要十分条件は、$\neg\varphi$ が充足可能でないことである。

- $\neg\varphi$ が充足可能でない事がわかれば、φ が証明できたことになる！

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要
  - 定理証明系で人間が手で書きたくない補題の自動証明にも使える

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要
  - 定理証明系で人間が手で書きたくない補題の自動証明にも使える
- SMT は SAT ソルバを外部の理論ソルバと協業させこれらの問題を解く

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要
  - 定理証明系で人間が手で書きたくない補題の自動証明にも使える
- SMT は SAT ソルバを外部の理論ソルバと協業させこれらの問題を解く
  - SAT を如何に速く解くかが SMT の実装にも重要

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要
  - 定理証明系で人間が手で書きたくない補題の自動証明にも使える
- SMT は SAT ソルバを外部の理論ソルバと協業させこれらの問題を解く
  - SAT を如何に速く解くかが SMT の実装にも重要
- Disclaimer: 今回は SMT の話はしません

SAT ソルバの原理

SAT は 「難しい」

- 論理パズルの解法で見たように、すべての命題変数の有り得る真偽値の割り当てを総当たりして、真理表をつくれば原理的には解ける
  - これは命題変数の数を n とすると 2^n 通りの場合分けが必要になる
  - もっと効率的な方法はないか？
- 実は「総当たりで解ける問題」は全て SAT に帰着できる（SAT は NP- 完全である）ことが知られている
  - $P \neq \mathrm{NP}$ 予想が正しければ、本質的に効率的な解法は存在しない
- 本質的に速くなくても実用的な問題に対し十分速い手法が研究されている
  - 以下ではその中でも主流な CDCL 法の原理を簡単に紹介（参考文献 [2–5]）

連言標準形 （CNF）

- SAT ソルバは連言標準形 (CNF) と呼ばれる形の論理式を扱うことが多い

定義 10

1. 命題変数 P_i およびその否定 $\neg P_i$ を合わせてリテラルと呼ぶ。 以下、リテラルを表すメタ変数を l, l_i などと書く。
2. 0 個以上のリテラルを 「または」 で繋いだもの $l_1 \lor \ldots \lor l_k$ を節と呼ぶ。以下、節を表すメタ変数を C, C_i などと書く。
3. 0 個以上の節を「かつ」で繋いだ $C_1 \land \ldots \land C_m$ の形の論理式を連言標準形（Conjunctive Normal Form, CNF）と呼ぶ。

注意：空の節は矛盾に、空の CNF は恒真に対応する。

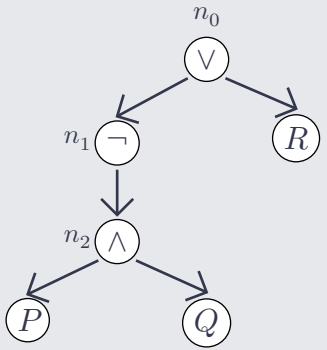
任意の命題論理式は CNF で表せる

定理 11 (命題論理式の CNF への変換)

任意の命題論理式 φ は同値な CNF φ_{CNF} を持つ。

- de Morgan 則と分配則を使って変換すればよいが、自乗のオーダーかかる
- 充足可能性だけを考えればよいのなら、結合子ごとに追加の命題変数 n_i を導入し、各節のサイズが高々 3 の CNF（3-CNF）に変換できる
- 論理式 $\neg(P \wedge Q) \vee (R \wedge S)$ の CNF への変換例：

$$
\begin{aligned}
& n_0 \wedge (\neg n_0 \vee n_1 \vee R) \wedge (\neg n_1 \vee n_0) \wedge (\neg R \vee n_0) \\
& \quad \wedge (\neg n_1 \vee \neg n_2) \wedge (n_1 \vee n_2) \\
& \quad \wedge (\neg n_2 \vee P) \wedge (\neg n_2 \vee Q) \wedge (\neg P \vee \neg Q \vee n_2)
\end{aligned}
$$



- 演習問題：他の結合子の変換規則を考えてみよう。

DPLL: CDCL の祖先

- CDCL は Davis–Putnam–Logemann–Loveland (DPLL) を基礎としている
- 基本的な考え方： CNF の節やリテラルを削れるまで削ってから場合分け
  - 充足された節は削除できる $\rightarrow$ 節が一つもなくなれば充足可能！
  - 偽なリテラルがあれば、節から削除できる $\rightarrow$ 空な節ができたら充足不能！
- 以下の規則に従って CNF を変形する：

(a) 単位節伝播：リテラル一つだけの節 $\{l\}$ があれば $l \equiv \top$ とし $\neg l$ を削除

(b) 純リテラル：肯定または否定のみ出現するリテラル l があれば $l \equiv \top$ としそのリテラルを含む節を削除（コスト面から実装しない事が多い）

(c) Decide：他規則が適用できないならリテラルを一つ選び場合分け

- 真偽を仮決めし継続。失敗したら巻き戻し (Backtrack) 否定で確定し継続

DPLL の動作例

- $(\neg P_1 \vee P_2) \wedge (P_3 \vee P_4) \wedge (\neg P_5 \vee \neg P_6) \wedge (\neg P_2 \vee \neg P_5 \vee P_6)$ を DPLL で解いてみよう（[2] の例）。
- 今後は CNF しか出て来ないので、$(\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6)$ と略記する

$$
\begin{array}{rll}
 & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid \\
\xrightarrow{\text{(Dec 1)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}} \\
\xrightarrow{\text{(Unit 2)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}}\ 2 \\
\xrightarrow{\text{(Dec 3)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}} \\
\xrightarrow{\text{(Dec 5)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ 5^{\mathrm{d}} \\
\xrightarrow{\text{(Unit 6)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \underline{\bar{6}})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ 5^{\mathrm{d}}\ \underline{\bar{6}} \\
\xrightarrow{\text{(Bktrk)}} & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) & \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ \bar{5}
\end{array}
$$

Backtrack しまくる DPLL の動作例

$$
\begin{array}{rl|l}
 & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \\
\text{–(Dec [1-4])}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ 4^{\mathrm{d}} \\
\text{–(Unit [5-11])}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ 4^{\mathrm{d}}\ \dots\ 11 \\
\text{–(Unit 12)}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}}) & \dots\ 4^{\mathrm{d}}\ \dots\ 11\ \underline{\underline{12}} \\
\text{–(Bktrk)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{4} \\
\text{–(Dec [5-7])}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{4}\ \dots\ 7^{\mathrm{d}} \\
\text{–(Unit [8-11])}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}}) & \dots\ \bar{4}\ \dots\ 11 \\
\text{–(Bktrk)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{4}\ \dots\ \bar{7} \\
\text{–(}\cdots\text{)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}}) & \dots\ \bar{4}\ \dots\ 11 \\
\text{–(Bktrk)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{4}\ \dots\ \bar{6} \\
\text{–(}\cdots\text{)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}}) & \dots\ \bar{4}\ \dots\ 11 \\
\text{–(Bktrk)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{4}\ \dots\ \bar{5} \\
\text{–(Bt on } 7,8,10,11\text{)}\longrightarrow & (\bar{1},\bar{2},\bar{3},4,5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & \dots\ \bar{5}\ \bar{7}\ \bar{8}\ \overline{10}\ \overline{11}
\end{array}
$$

CDCL: Conflict-Driven Clause Learning

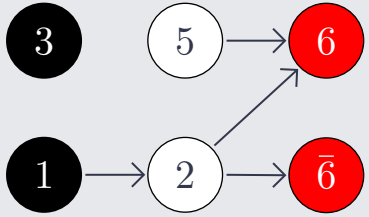
- 単純総当たりより DPLL は賢いが、Backtrack で得られる情報が少ない
  - 折角 Backtrack でいったん真偽値が確定しても、更に Backtrack すればその情報は消えてしまう
- Backtrack は一段階ずつしか戻らない
  - 場合分けが入れ子になっている場合、実は直近の場合分けよりもっと早くに矛盾が起きていても延々と一段階ずつ Backtrack してしまう
- 矛盾からに過程を辿っていって矛盾の原因となった「理由」を見付け。学習節として CNF に追加してそこまで一気に巻き戻そう (Backjump) ！
  - これが矛盾からの節学習 (Conflict-Driven Clause Learning, CDCL)

CDCL: どういう学習節が望ましいか？

- 学習節が元々の入力の帰結であること
  - 任意のタイミングで Backjump が起き得る
  - いつでも使えるように、仮定が本質的には増えていてはならない
- 「すぐに使える」学習節であること
  - Backjump した直後に、新しい情報が得られないと意味がない
  - 特に、巻き戻した段階で単位節になっているようにしたい
- なるべく小さいほうがよい
  - 単位節や矛盾が比較的すぐ出るようにしたい
- 含意グラフを考えるとこれらを満たす学習節を得られる

CDCL：含意グラフ

- これまでに真偽を割り当てたリテラルを頂点として持つ有向グラフ
- 単位伝播で真になったリテラルへ、根拠となったリテラルから辺を伸ばす
  - 単位伝播＝「他のリテラルが真にならなかったので最後の一つを真にする」
  - 「根拠」：つまり、真偽を確定させた節に含まれる他のリテラルの否定
  - ただ一つのリテラルを持つ単位節のリテラルや、場合分けリテラルの場合入次数はゼロ
- 以下は最初の $(\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6)$ に対する含意グラフの例
- 黒丸はこれよりも前の場合分けで仮置きされたリテラル
- 赤丸は互いに矛盾したリテラル

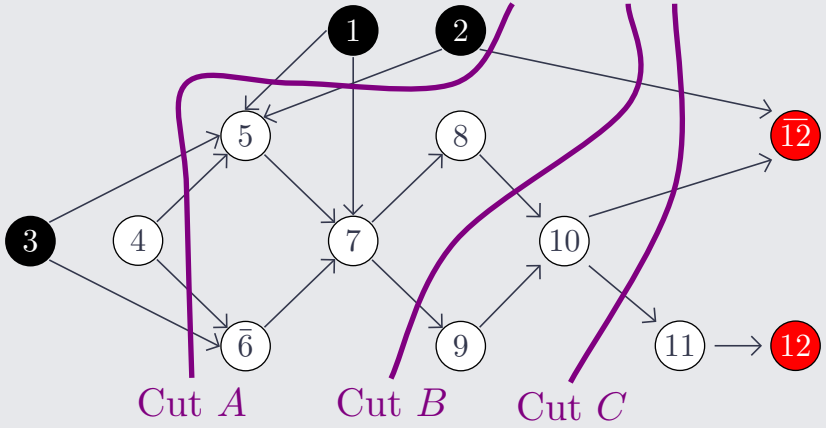


CDCL：含意グラフと学習節 1 - カットと矛盾原因

- 含意グラフを以下の二つの集合にカットすると、矛盾への原因が得られる
  (a) conflict side: 直接矛盾した頂点をどちらも含む
  (b) reason side: 現在の場合分けの起点と以前の場合分け対象全部を含む
- このとき、reason side のリテラルは conflict side の原因になっている
  - 特に conflict side に向かう辺を持つ reason side の頂点があれば十分

Cut と学習節の例：多段 Backtrack の例の序盤

$(\bar{1}, \bar{2}, \bar{3}, \bar{4}, 5)$ $(\bar{3}, \bar{4}, \bar{6})$ $(\bar{5}, 6, \bar{1}, 7)$ $(\bar{7}, 8)$ $(\bar{2}, \bar{7}, 9)$ $(\bar{8}, \bar{9}, 10)$ $(\bar{10}, 11)$ $(\bar{11}, \underline{\underline{12}})$ $(\bar{10}, \bar{2}, \underline{\underline{\bar{12}}})$



- Cut A: $(\bar{1}, \bar{2}, \bar{3}, \bar{4}) \to 3$ の場合分け直後へ飛び、$\bar{4}$ が単位節伝播
- Cut B: $(\bar{2}, \bar{7}, \bar{8}) \to 2$ の場合分け直後へ飛ぶが何もわからない
- Cut C: $(\bar{2}, \overline{10}) \to 2$ の場合分け直後へ飛び、$\overline{10}$ が単位節伝播
  - 単位節になっているものとなっていないものの違いは何か？

単位節と UIP

- Cut に対応する学習節が単位節になるのは、 一意相互作用点 （Unique Interaction Point, UIP）と呼ばれる頂点の直後で Cut した時
  - 頂点 l が UIP である $\iff$ グラフから l を取り除くと、直近の決定リテラルから少なくとも一方の矛盾頂点へのパスが消える（「必ず通る点」）
- 前の例では $4, 7, 10$ が UIP で、Cut A, C が頂点 $4, 10$ の直後の Cut に相当
- 「学習節が単位節になる」ためには UIP で切ればよい（逆もまた真）
  - 決定リテラル自身も UIP なので、少なくとも一つ UIP は存在する
  - 矛盾に一番近い UIP、1-UIP の Cut がよいことが経験的に知られている [6]
  - 前の例では、$(\bar{2}, \overline{10})$ に対応する 10 で切った Cut C が 1-UIP
  - conflicting side が増えるほど必要な仮定も増えそうなので、直感的ではある

1-UIP 学習節の生成と導出規則

- 含意グラフは概念上でわかりやすいが、直接実装する必要はない
  - 矛盾節に導出規則 (Resolution) を単位節になるまで繰り返し適用すればよい

定理 12 (導出規則)

以下の導出規則は古典論理の派生規則：

$$\frac{c \vee \Phi, \quad \neg c \vee \Psi}{\Phi \vee \Psi}$$

- 古典的に $c \vee \Phi$ は $\neg c \to \Phi$ と、$\neg c \vee \Psi$ は $c \to \Psi$ とそれぞれ同値
- 更に排中律があるので $c \vee \neg c$ が成り立つから、結局 $\Phi \vee \Psi$ が結論できる

導出の繰り返しで 1-UIP 学習節を作る

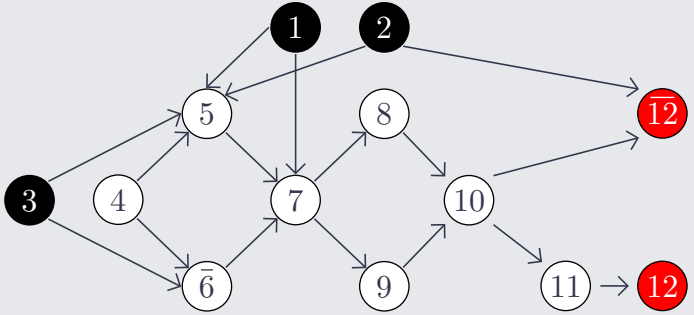
1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

- 学習節の導出の例：

導出の繰り返しで 1-UIP 学習節を作る

1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

◆ 学習節の導出の例：



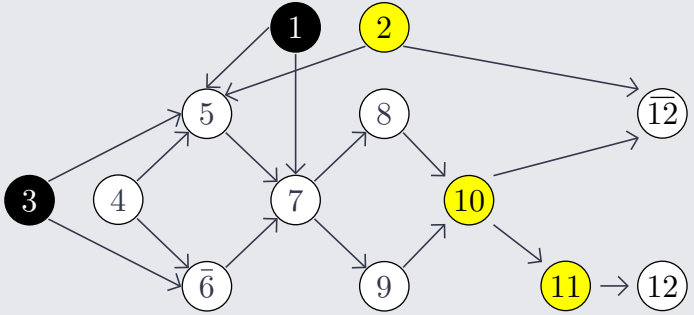
$(\bar{2}, \overline{10}, \overline{12})$

$(\overline{11}, 12)$

導出の繰り返しで 1-UIP 学習節を作る

1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

◆ 学習節の導出の例：



$(\bar{2}, \overline{10}, \overline{12})$

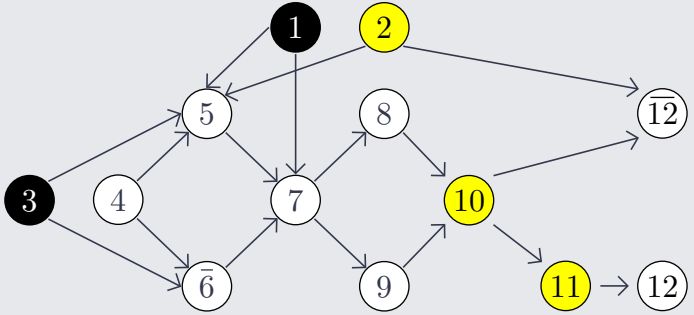
$(\overline{11}, 12)$

$(\bar{2}, \overline{10}, \overline{11})$

導出の繰り返しで 1-UIP 学習節を作る

1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

◆ 学習節の導出の例：



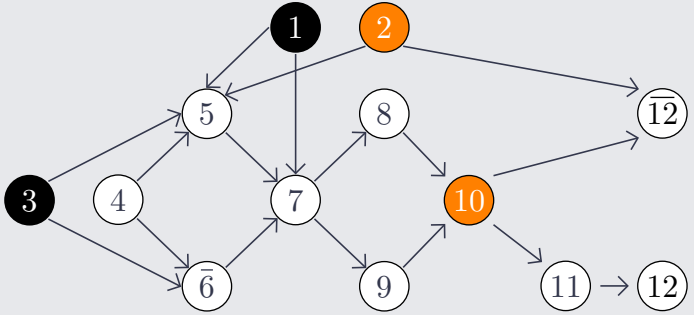
$(\bar{2}, \overline{10}, \overline{12})$ ↘

$(\overline{11}, 12)$ ↗ $(\bar{2}, \overline{10}, \overline{11})$ $\xrightarrow{(\overline{10}, 11)}$

導出の繰り返しで 1-UIP 学習節を作る

1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

◆ 学習節の導出の例：



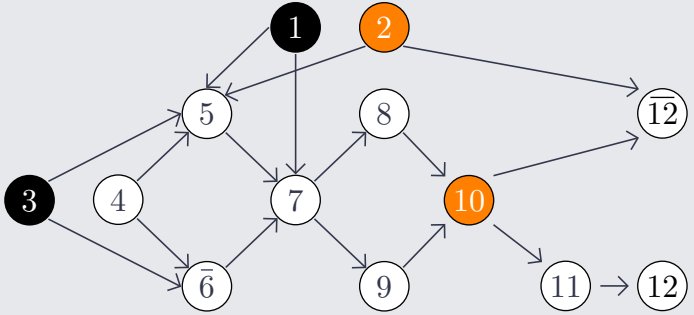
$$(\bar{2}, \overline{10}, \overline{12}) \searrow$$

$$(\overline{11}, 12) \nearrow \quad (\bar{2}, \overline{10}, \overline{11}) \xrightarrow{(\overline{10}, 11)} (\bar{2}, \overline{10})$$

導出の繰り返しで 1-UIP 学習節を作る

1. 矛盾した二つの節の矛盾リテラルに導出規則を適用
2. 単位節になるまで、仮学習節の中で「一番最後に真偽が確定した命題変数」を選び、その真偽を確定させた節と導出規則を適用
   - 「単位節になる」＝最後の場合分け以後の命題変数が一つになる

- 学習節の導出の例：



$$(\bar{2}, \overline{10}, \overline{12}) \searrow$$
$$(\overline{11}, 12) \nearrow \quad (\bar{2}, \overline{10}, \overline{11}) \xrightarrow{(\overline{10}, 11)} (\bar{2}, \overline{10})$$

- 根拠となる節で導出を繰り返す事が、「原因」を手繰っていく事に対応

同じ例の CDCL の動作

長い例で学習節ありの Backjump がどう振る舞うか見てみよう

	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})$	
$\longrightarrow$(…)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}})$	… 4^{d} … 11 $\underline{\underline{12}}$
$\longrightarrow$(Bkjmp)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})$	1^{d} 2^{d}
$\longrightarrow$(Unit $\overline{10}$)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})$	1^{d} 2^{d} $\overline{10}$
$\longrightarrow$(Dec [3-4])	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})$	$\overline{10}$ … 4^{d}
$\longrightarrow$(Unit [5-8])	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},\underline{9})\ (\bar{8},\underline{\bar{9}},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})$	$\overline{10}$ … 8
$\longrightarrow$(Bkjmp)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	1^{d} 2^{d} $\overline{10}$
$\longrightarrow$(Unit $\bar{7}$)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… $\bar{7}$
$\longrightarrow$(Dec [3-4])	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… 4^{d}
$\longrightarrow$(Unit [5-6])	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… $\bar{6}$
$\longrightarrow$(Dec 8)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… 8^{d}
$\longrightarrow$(Unit $\bar{9}$)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… 4^{d} … $\bar{6}$ 8^{d} $\bar{9}$
$\longrightarrow$(Dec 11)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	… 11
$\longrightarrow$(Unit 12)	$(\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12})\ (\overline{10},\bar{2})\ (\bar{2},\bar{7},10)$	1^{d} … 12

Backjump 二回だけですんなり解けている！

CDCL：更なる最適化手法

現代の CDCL では以下のようなテクニックで効率化が図られている：

- 監視リテラル：単位節または矛盾節の検出方法の効率化
  - 観察：単位節・矛盾節確定するのは、リテラル数が 2 以下になったときだけ
  - そこで、節ごとに最大 2 つのリテラルを監視リテラルとして指定
  - 監視リテラルの真偽が確定したときだけ、その節の状態を確かめて単位節・矛盾節の検出を行えばよい！
- 変数選択ヒューリスティクス：場合分けする変数と真偽には任意性がある
  - どのリテラル、値を選ぶかで性能が大きく変わる
  - 時間減衰する優先度をつかう VSIDS など変数選択の指標が提案されている
  - 選択の指標に関しては、例えば Liang らの論文 [7] が詳しい

CDCL：更なる最適化手法　（続）

- リスタート：履歴の破棄
  - 辿った履歴によって簡単にとけたり、逆に永遠に解けなかったりする
  - そこで矛盾が起きたら一定間隔で状態を破棄し、ゼロから解きなおす
  - 学習節は捨てず、間隔はだんだん長くするので、完全性は担保される
  - リスタート間隔のヒューリスティクスは Wu らの論文 [8] が詳しい
- 忘却：増えすぎた節の枝刈りをする
  - 矛盾のたびに学習節が増えるので、節数が爆発しがち
  - そこで、適切な指標の下で優先度の低い学習節を定期的に削除
  - 削除するのは学習節だけなので、完全性に影響しない

実装、　まとめ

CDCL 型 SAT ソルバの実装

- MiniSat [9] (`http://minisat.se`) がシンプルかつ高速で、実装の参考になる
- herbrand [10]: MiniSAT や参考文献を基に私が Haskell で実装してみたもの
  - `https://github.com/konn/herbrand`
  - Haskell を使って実装
  - それでも GC は無効化されないのでまだまだ遅い
  - メモリまわりの管理を MiniSat を参考に改築したい

まとめ

- SAT ソルバは古典命題論理式の充足可能性を判定するプログラム
  - NP- 完全なので本質的に難しい
  - CDCL: 矛盾するごとに学習節を作り、必要なところまで一気に巻き戻す
  - 監視リテラルや Restart など、色々なヒューリスティクスが提案されている
- CDCL は述語論理上の理論に一般化した SMT ソルバの基礎でもある
  - 私の動機： Presburger 算術 （整数計画法） と未解釈関数つき同値関係の理論の組合せソルバを実装したい
  - 整数計画法のインクリメンタルでバックトラック可能で完全なソルバが必要
    - CVC4 [11] のサイトなどを見てみたがパッと見付からず
    - よい文献をご存知の方おしえてください

参考文献

[1] レイモンド・スマリヤン , “スマリヤンの決定不能の論理パズル ゲーデルの定理と様相論理 ,” 白揚社 , 2008.

[2] Aleksandar Zeljić, "CS357: Advanced Topics in Computer Science". Course Material. Stanford University. 2019. https://web.stanford.edu/class/cs357/.

[3] 鍋島秀和 and 宋剛秀 , “高速 SAT ソルバーの原理 ,” 人工知能 , pp. 68–68, vol. 25, no. 1, 2010, https://www.jstage.jst.go.jp/article/jjsai/25/1/25_68/_article/-char/ja/.

[4] 岩沼宏治 and 鍋島秀和 , “SMT : 個別理論を取り扱う SAT 技術 ,” 人工知能 , pp. 86–86, vol. 25, no. 1, 2010, https://www.jstage.jst.go.jp/article/jjsai/25/1/25_86/_article/-char/ja/.

[5] Enuba Torlak, “A Modern SAT Solver”. Course Material. 2003. https://courses.cs.washington.edu/courses/cse507/17wi/lectures/L02.pdf.

[6] L. Zhang et al., “EffEfficient Conflict Driven Learning in a Boolean Satisfiability Solver,” in *Proceedings of ICCAD-01*, November 2001, pp. 279–279.

[7] J. H. Liang et al., “Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers,” in *Hardware and Software: Verification and Testing*, N. Piterman, Eds., Cham: Springer International Publishing, November 2015, pp. 225–225.

[8] H. Wu, “Randomization and Restart Strategies,” *Master's Thesis, University of Waterloo*, pp. 1–1, 2006.

[9] Niklas Eén and Niklas Sörensson. MiniSat. http://minisat.se/.

[10] Hiromi ISHII. `herbrand`. https://github.com/konn/herbrand.

[11] Clark W. Barrett, et al. "CVC4". in *Computer Aided Verification - 23rd International Conference, CAV 2011*, Snowbird, UT, USA, July 14-20, 2011. Proceedings. pp. 171-177. https://doi.org/10.1007/978-3-642-22110-1_14. https://cvc4.github.io/publications.html.