アニーリング業界が発達するためには

1. より多くの人が関わること
   興味を持つきっかけの提供
2. アニーリングが実際に役立つこと
   役に立つアプリケーションを作成する

アニーリング業界が発達するためには

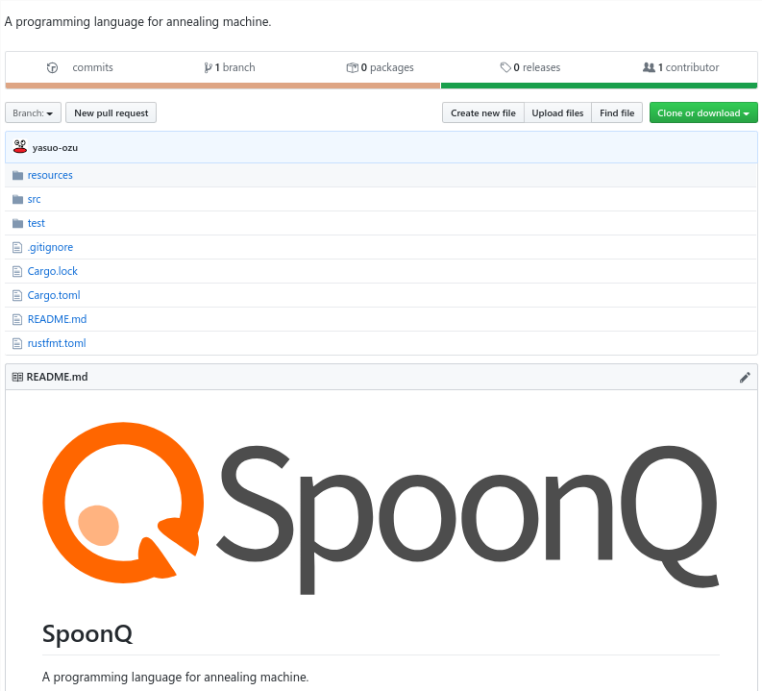
1 より多くの人が関わること
興味を持つきっかけの提供

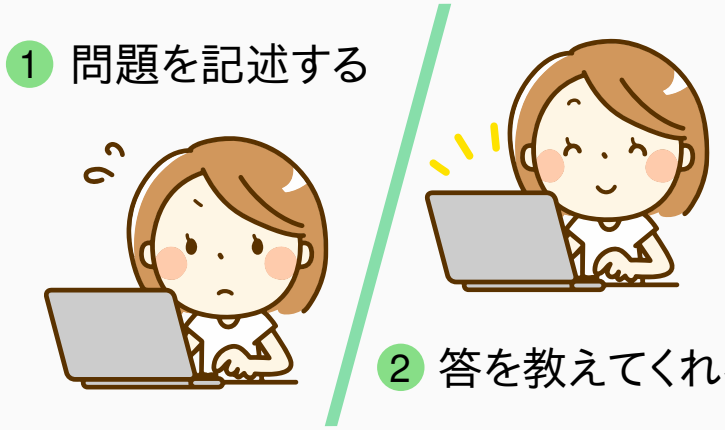
2 アニーリングが実際に役立つこと
役に立つアプリケーションを作成する

⇒ 現段階では、1に注力

昨年度のプロジェクト

2019 年度のプロジェクト・・・「SpoonQ」





初心者のユーザーがアニーリングに関する専門的な技術を学ぶこと無く、問題をアニーリングによって解くことができる

本プロジェクトの位置づけ

プロジェクトの対象

対象	2019 年度	本プロジェクト
問題の規模	小さい	大きい
ユーザー	新たに組合せ最適化問題を解く初心者ユーザー	既存の手段を活用して制約充足問題を解いているユーザー
アニーリングの知識	不要	不要

⇒ より多くの人をアニーリングの世界に引き込むことができる

本年度プロジェクト

2020 年度のプロジェクト

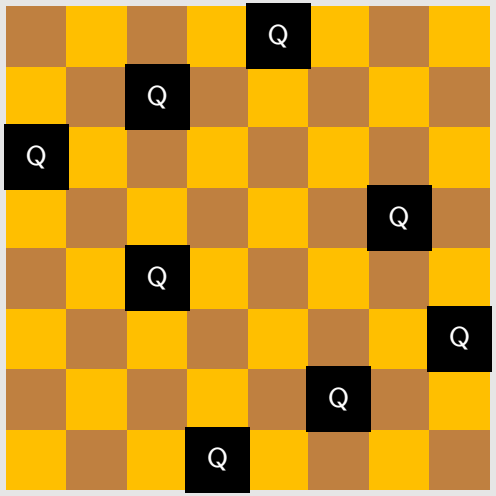
アニーリングを用いた効率的な制約充足問題ソルバの実装

制約充足問題

制約充足問題

変数と制約関数が与えられた時、制約関数を満たすような変数の値を求める問題。

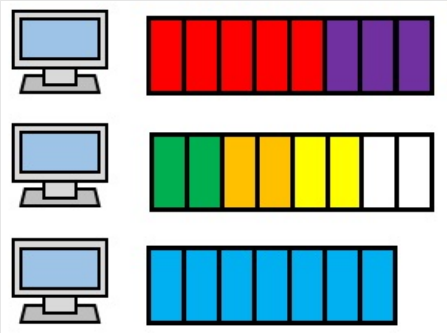
N クイーン問題



グラフ彩色問題

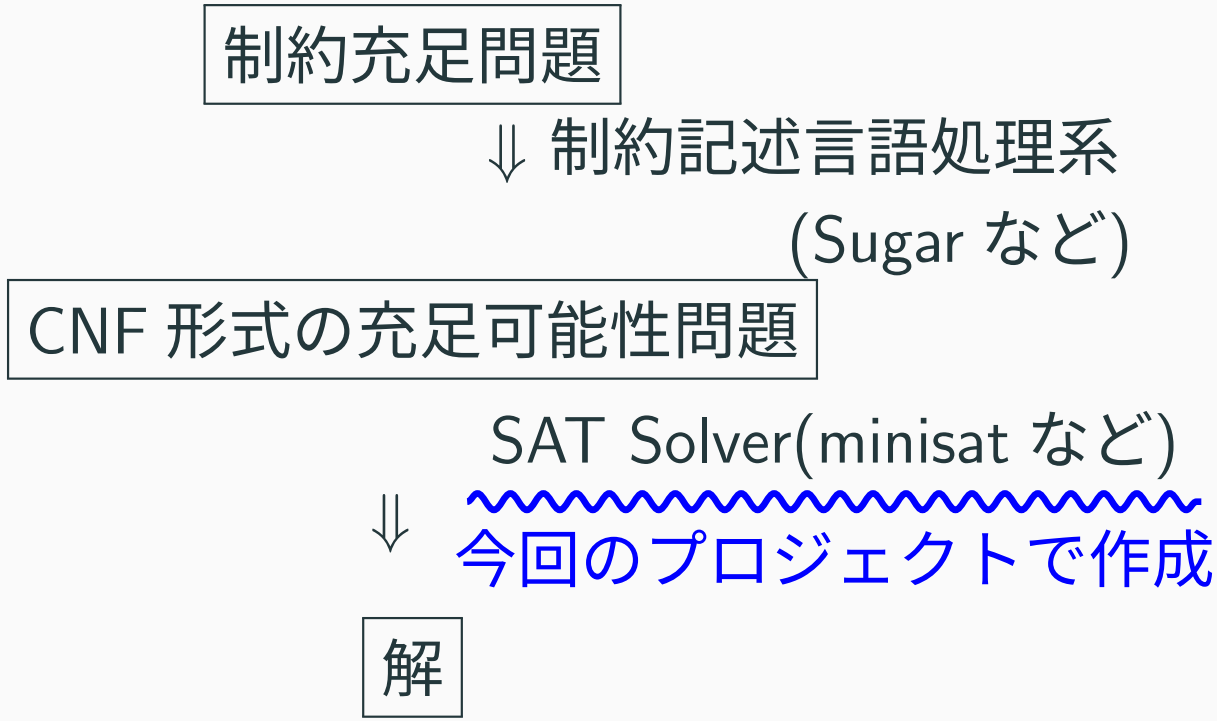
(Wikipedia より)

スケジュール問題



制約充足問題の解き方

制約充足問題の解き方 (SAT ソルバを使う場合)



→ 既に SAT Solver を問題解決に用いている人がアニーリングに興味を持つきっかけ

制約充足問題の解き方

制約充足問題の解き方 (SAT ソルバを使う場合)

制約充足問題

⇓ 制約記述言語処理系

(Sugar など)

CNF 形式の充足可能性問題

SAT Solver(minisat など)

のプロジェクトで作成

CNF 形式の例

$x_1, \ldots, x_n \in \{True, False\}$

$x_1 \vee x_2 \vee x_3$

$\neg x_2 \vee x_4$

→ 既

いている人がアニーリン

グに

より効率的なアニーリングのために

古典 SAT ソルバ

節の数、変数の数のバランスが重要

アニーリング

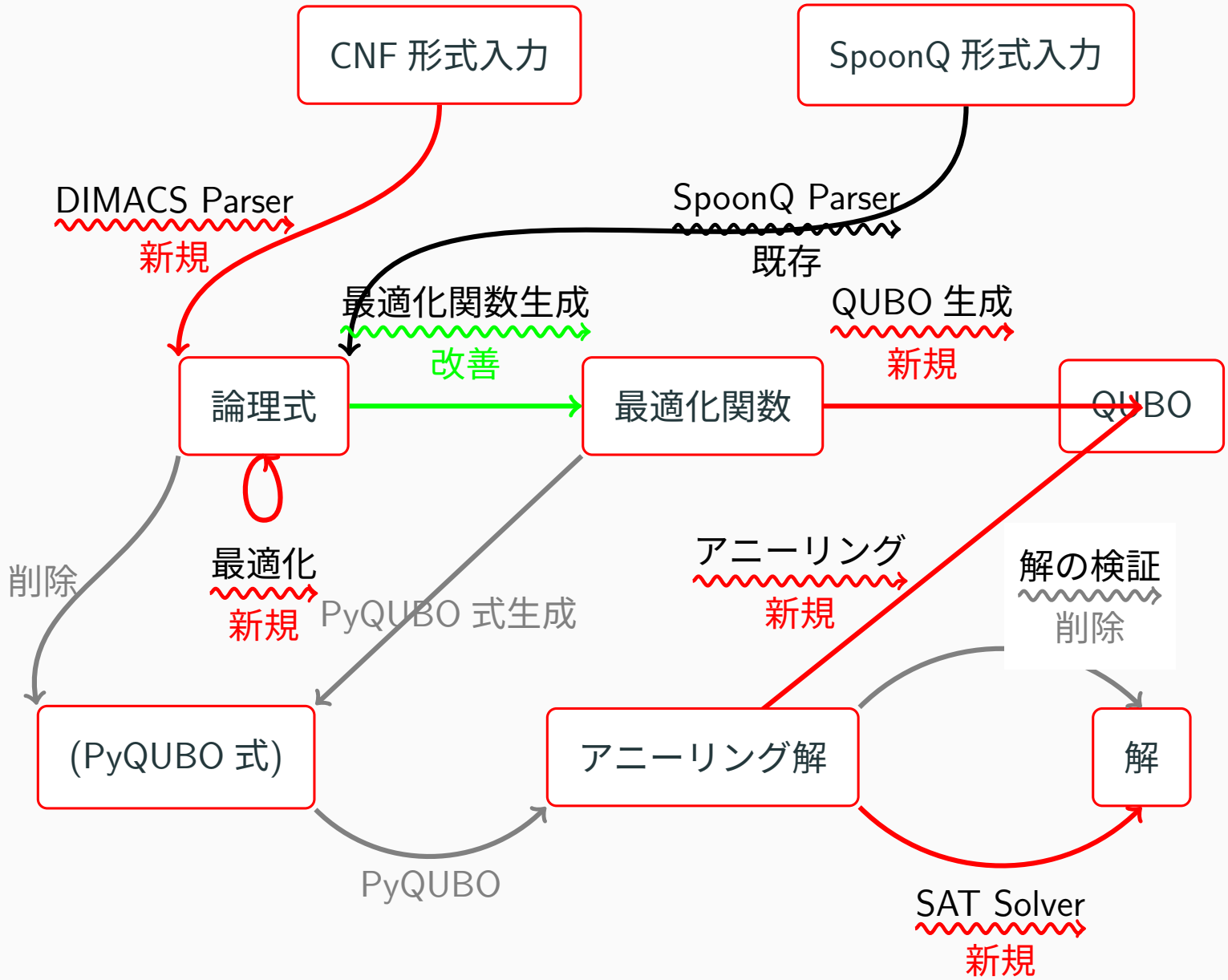
変数の数を減らしたい (節の数は問題ない)

具体例 ・・・8-Queen 問題による試算

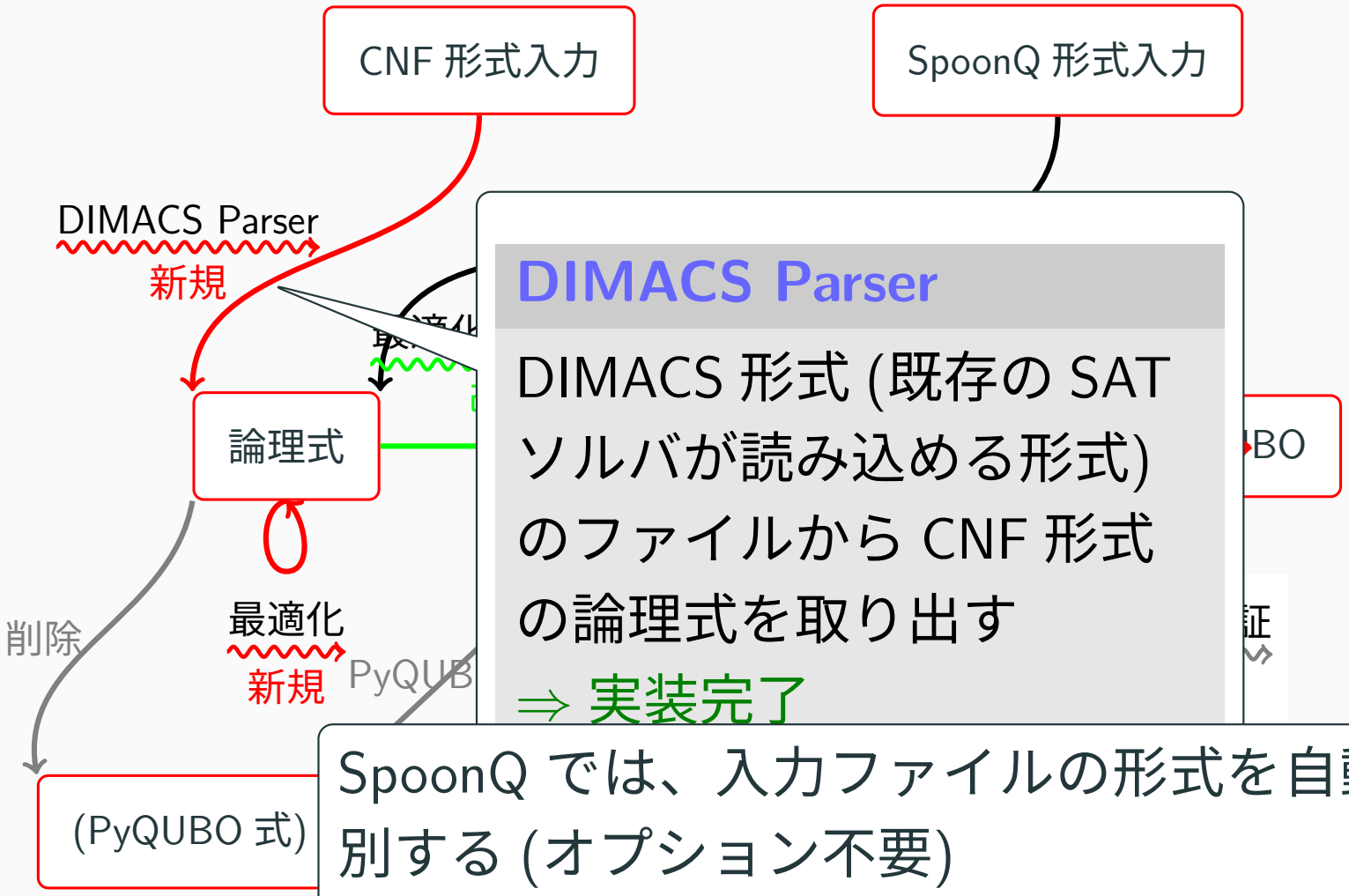
	節数	変数の数
CNF を直接用いる場合 (現行手法)	519	460
理想的なハミルトニアン (これを目指す)	16	64

※ 8-Queen 問題の CNF 生成には Sugar を使用 ※節数 ・・・ トップレベルの論理和の数

本プロジェクトによる SpoonQ の改造



本プロジェクトによる SpoonQ の改造

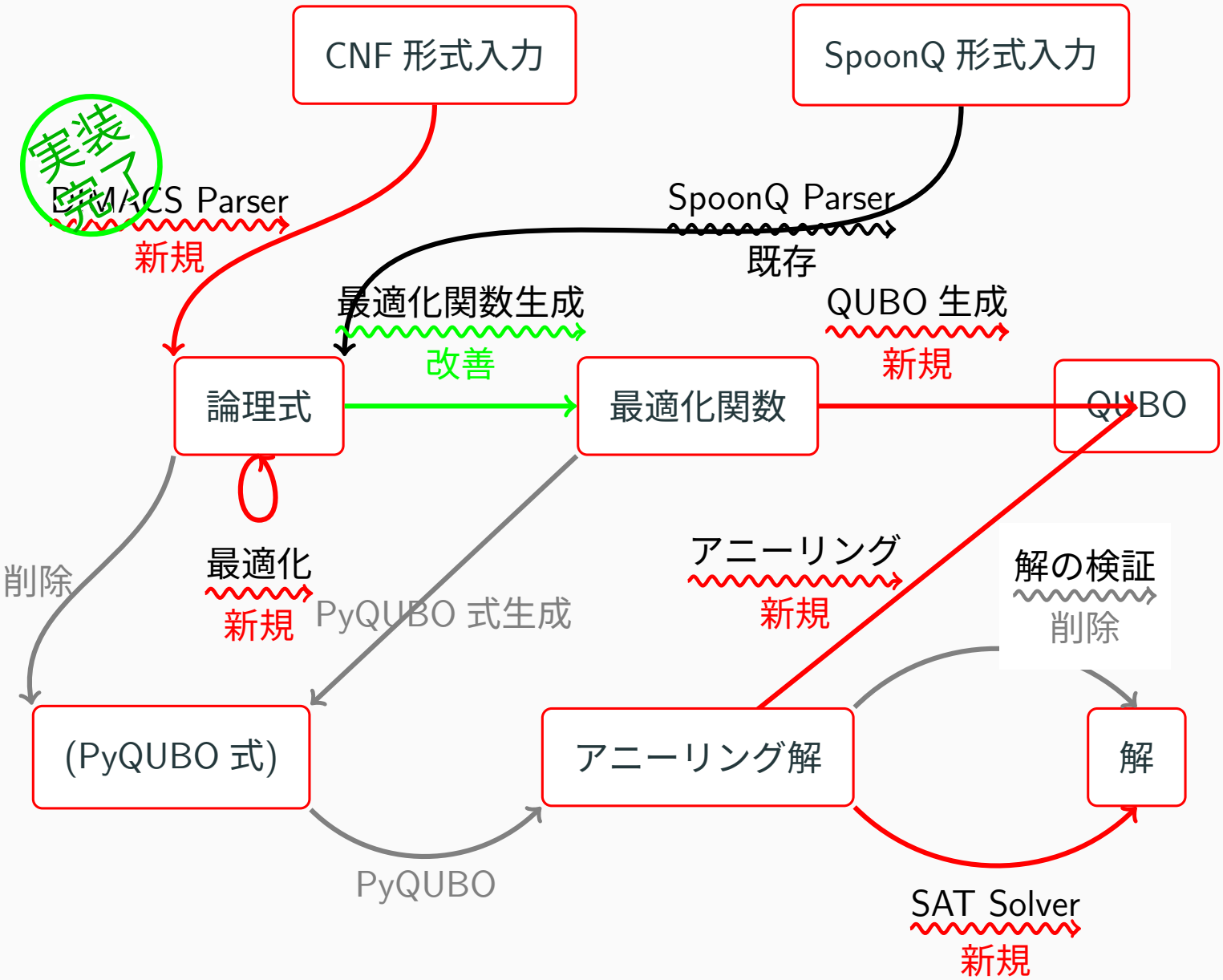


SpoonQ では、入力ファイルの形式を自動で判別する (オプション不要)

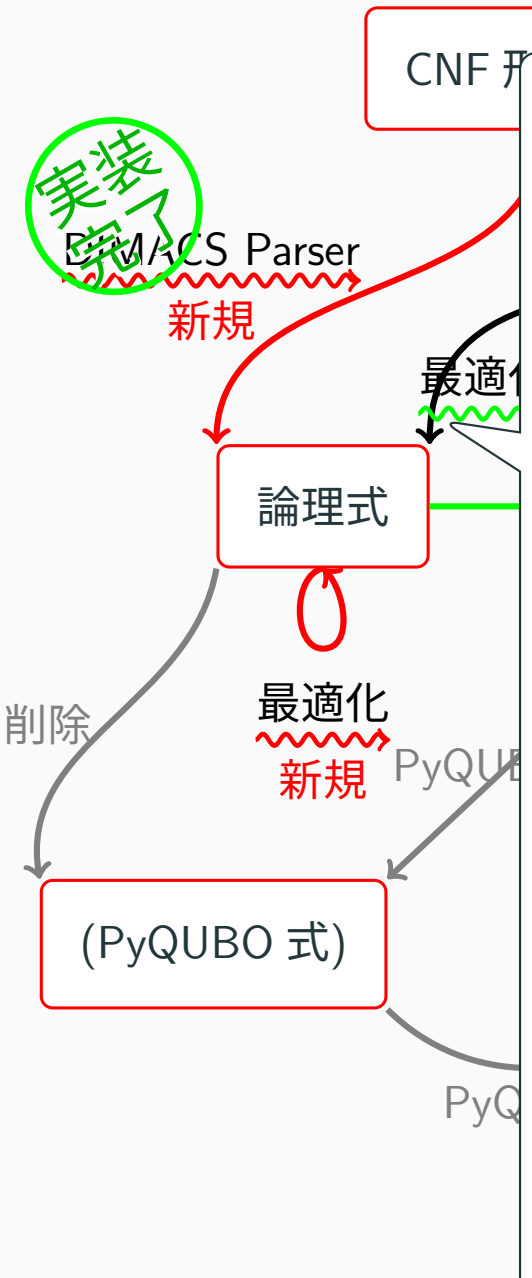
DIMACS 形式 (今年度)▷ `spoonq file.dimacs`

SpoonQ 形式 (昨年度)▷ `spoonq file.txt`

本プロジェクトによる SpoonQ の改造



本プロジェクトによる SpoonQ の改造



論理式について

CNF 由来の論理式

$$\mathrm{Val}(x) \in \{True, False\},\ \mathrm{Not}(x),$$
$$\mathrm{And}(x_1, x_2, \ldots),\ \mathrm{Or}(x_1, x_2, \ldots)$$

アニーリングマシンが解きやすい論理式

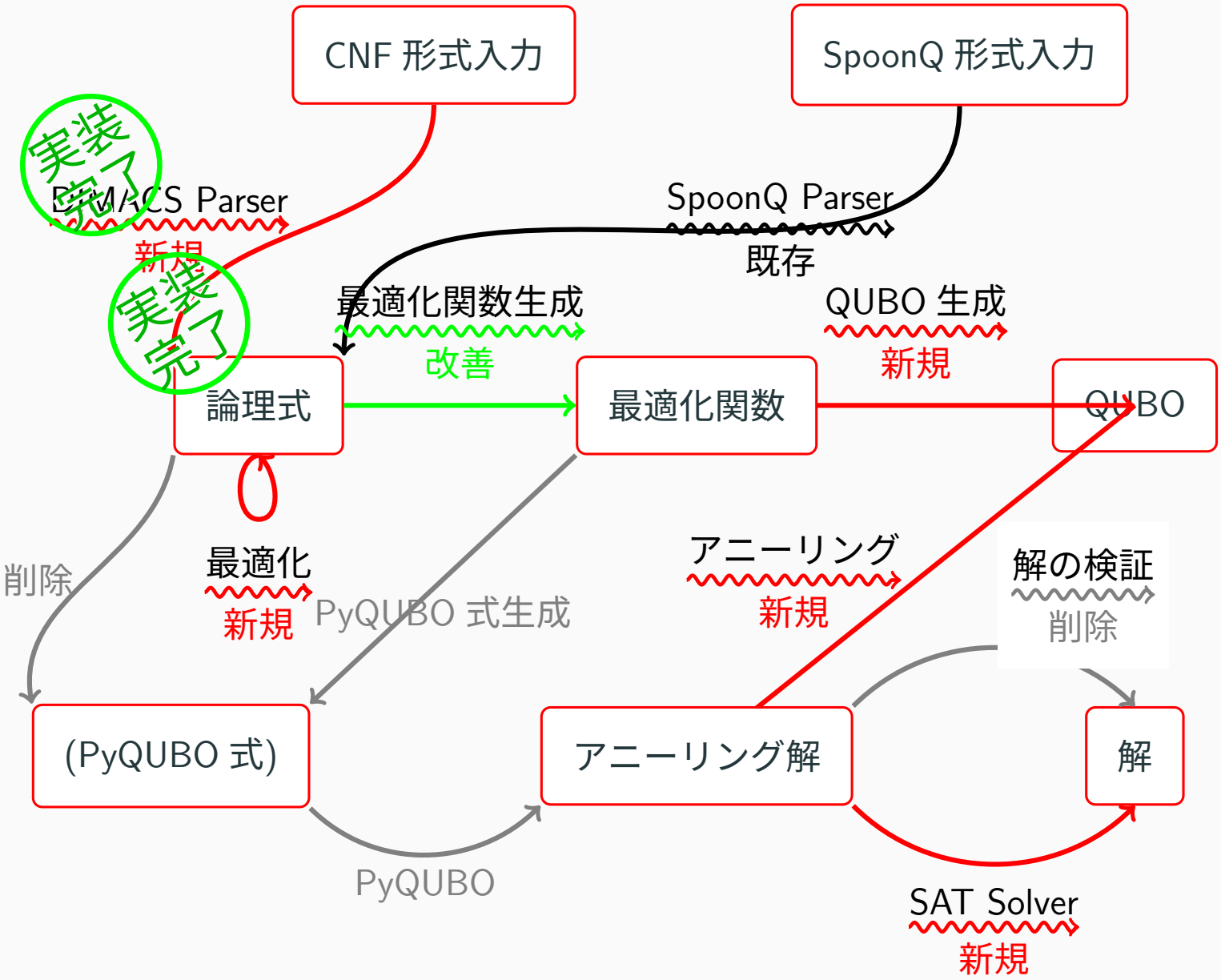
$$\mathrm{CountEq}(x_1, \ldots, x_n, cnt)$$

$x_1, \ldots, x_n$ の中で $True$ となる変数が cnt 個である条件

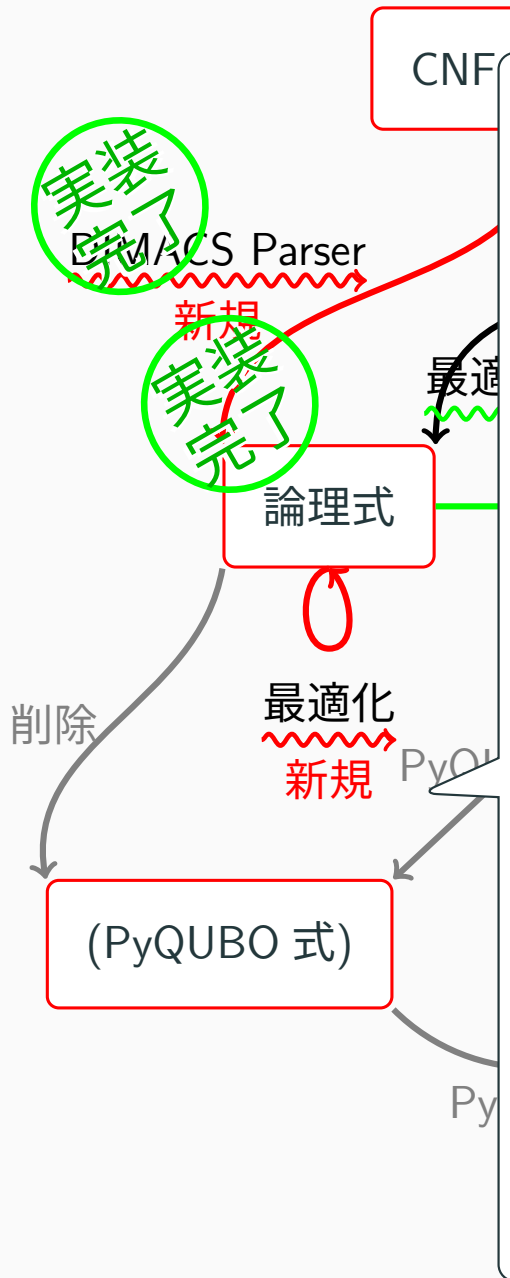
$$\mathrm{CountLeq}(x_1, \ldots, x_n, cnt)$$

$True$ となる変数が cnt 個以下である条件

本プロジェクトによる SpoonQ の改造



本プロジェクトによる SpoonQ の改造



最適化の方法

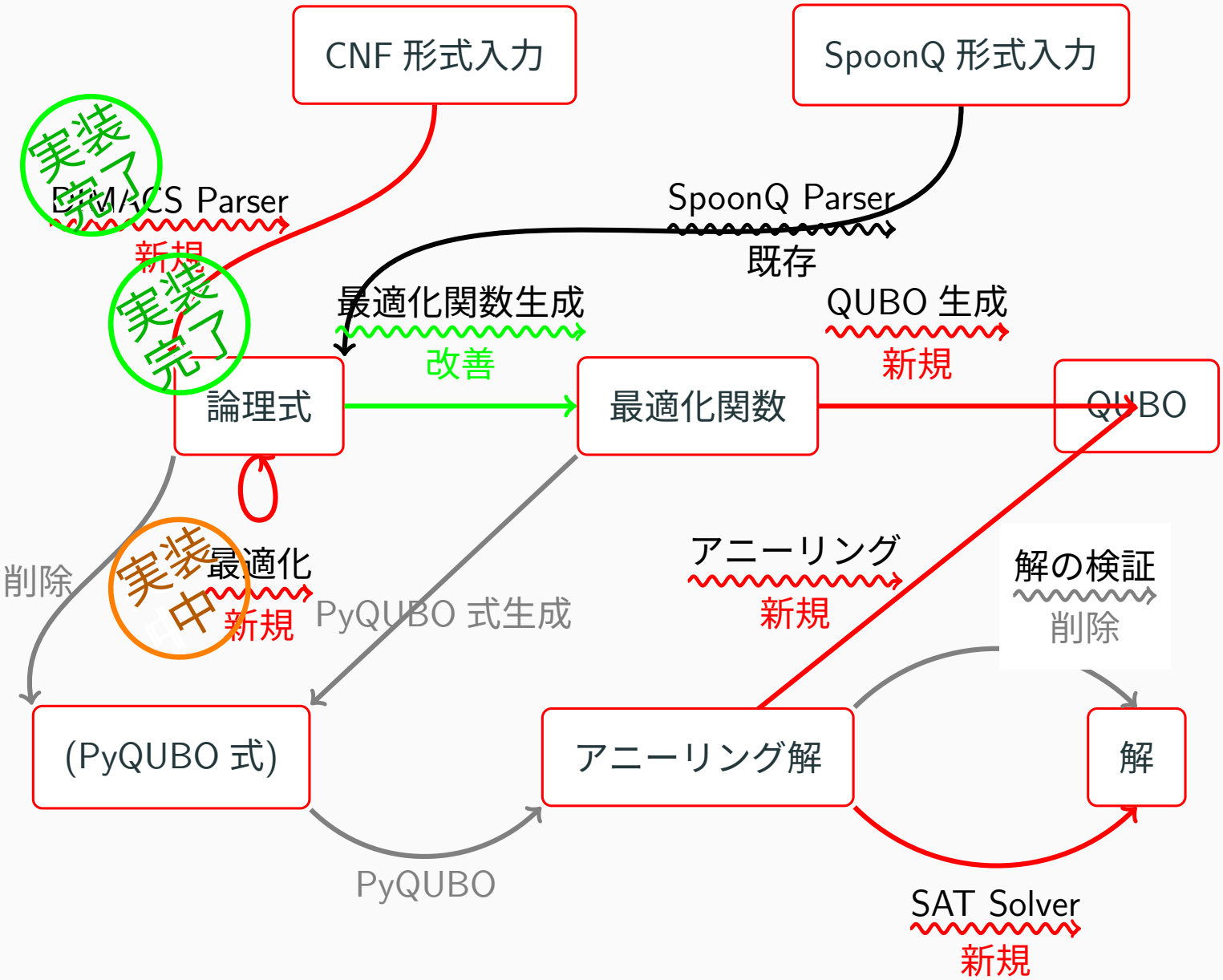
- And, Or をまとめる
- 同じ解をとる条件同士を見つけてまとめる
- パターンマッチングにより CountEq, CountLeq を生成する (実装中)

パターンマッチングの例:

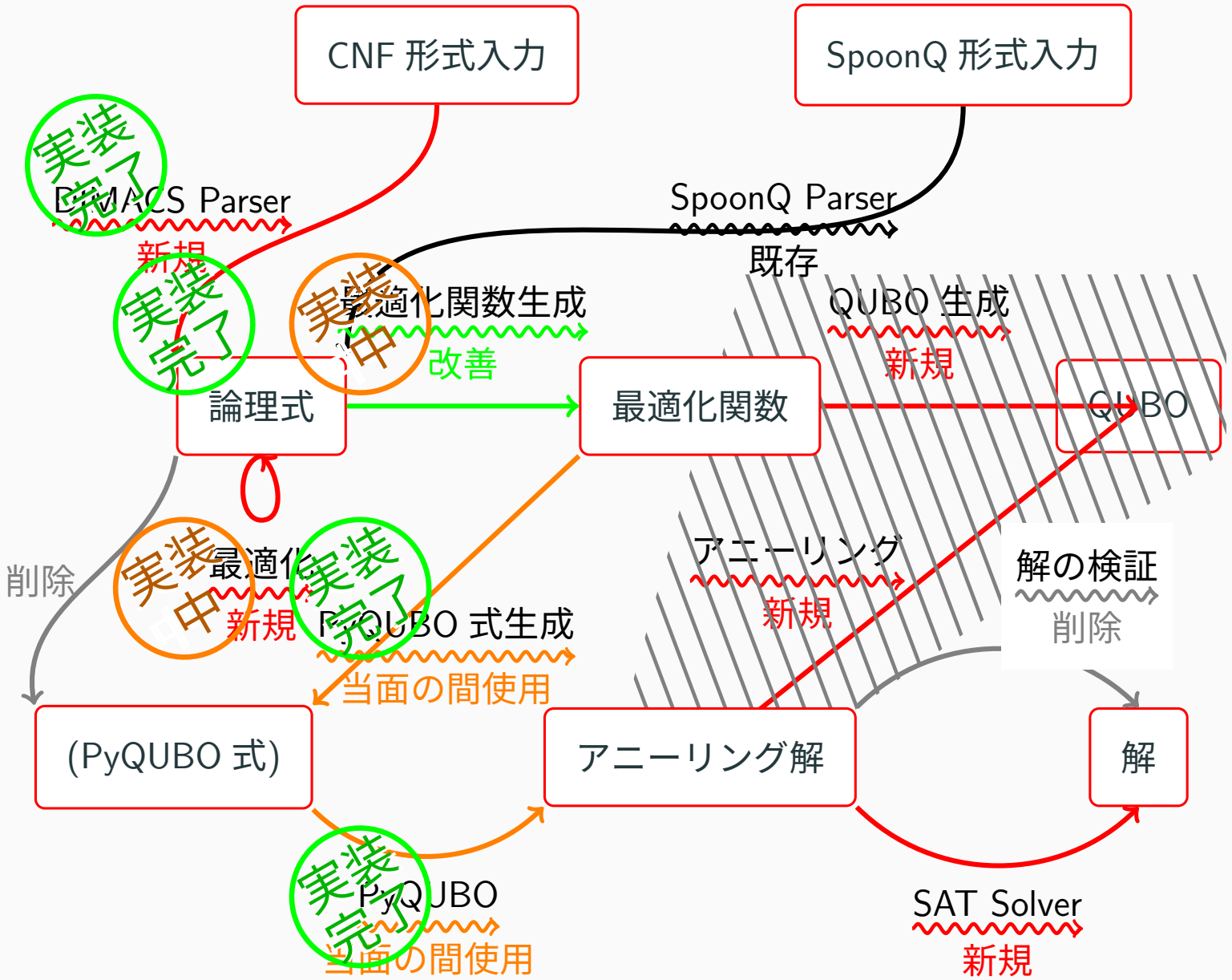
$(x \vee y \vee z) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (\neg x \vee \neg z)$

$\Rightarrow \mathrm{CountEq}(x, y, z, 1)$

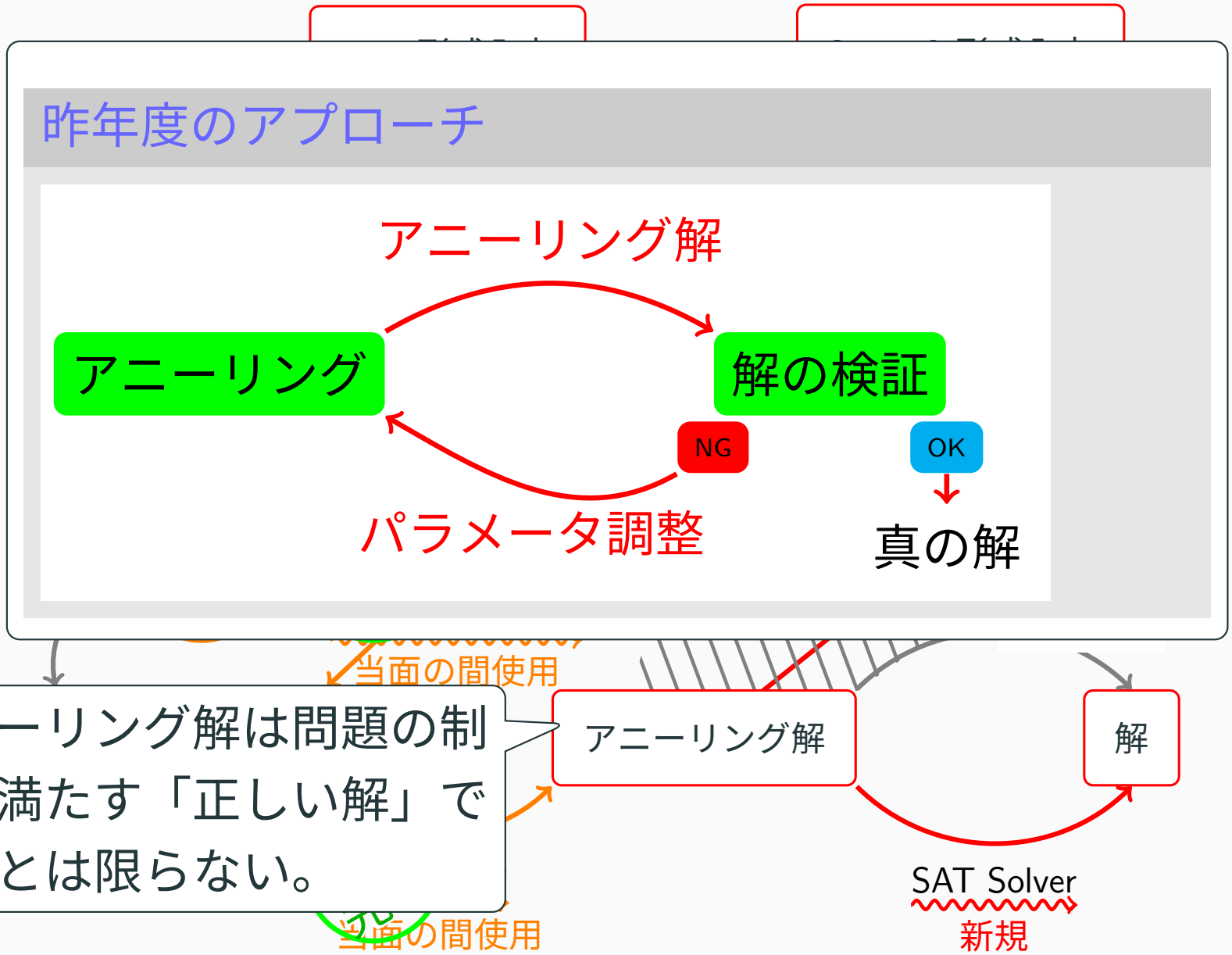
本プロジェクトによる SpoonQ の改造



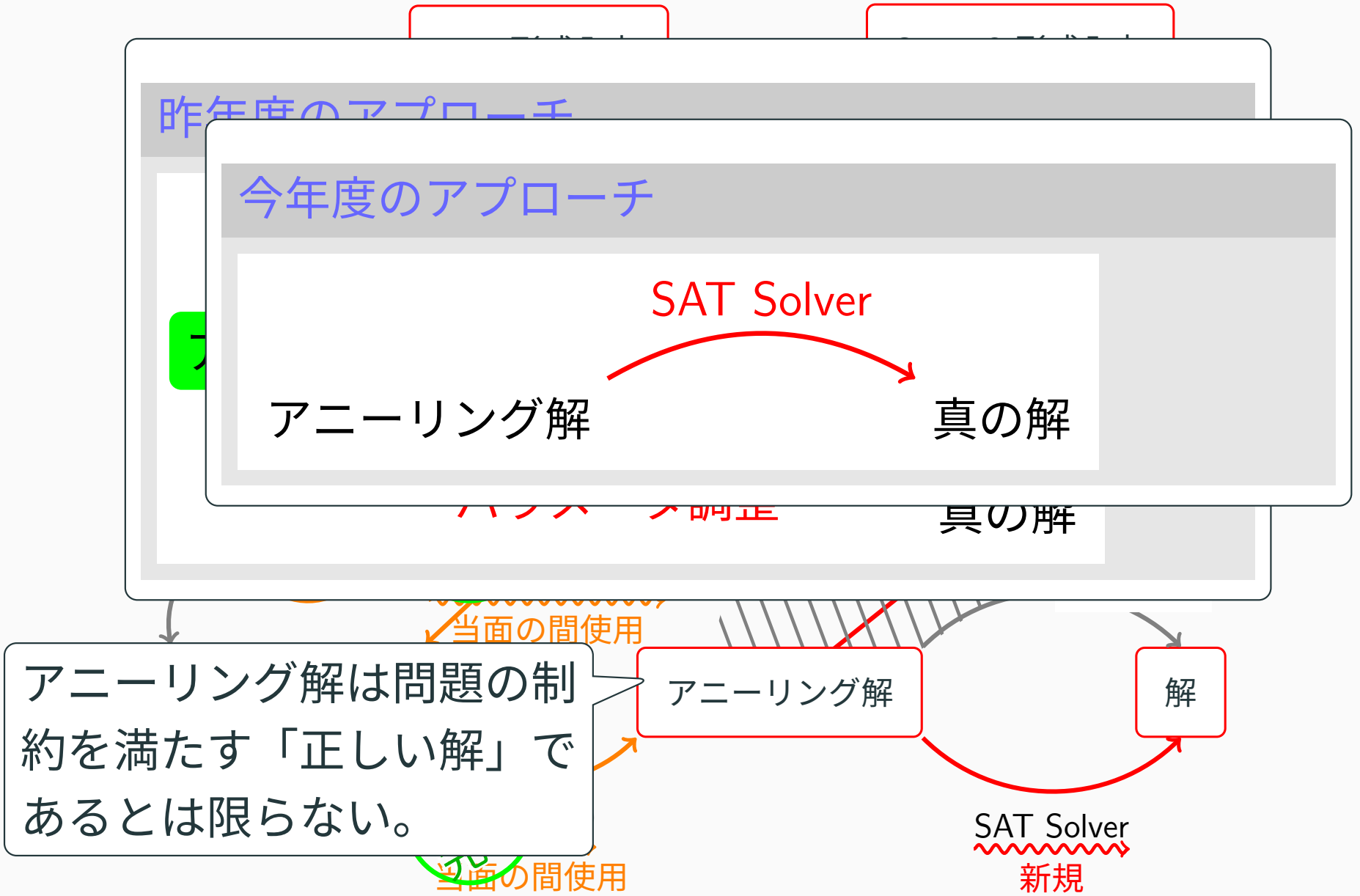
本プロジェクトによる SpoonQ の改造



本プロジェクトによる SpoonQ の改造

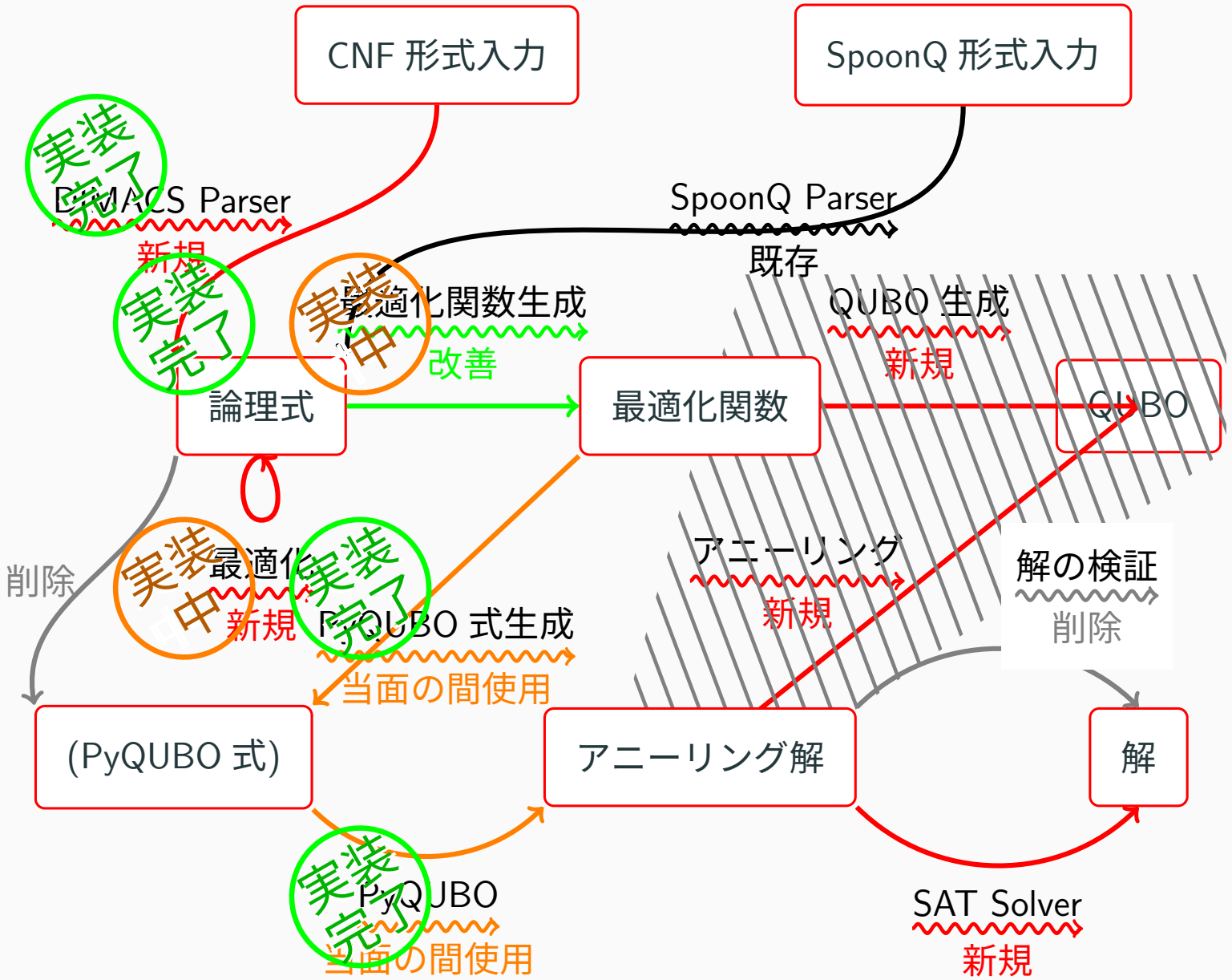


本プロジェクトによる SpoonQ の改造

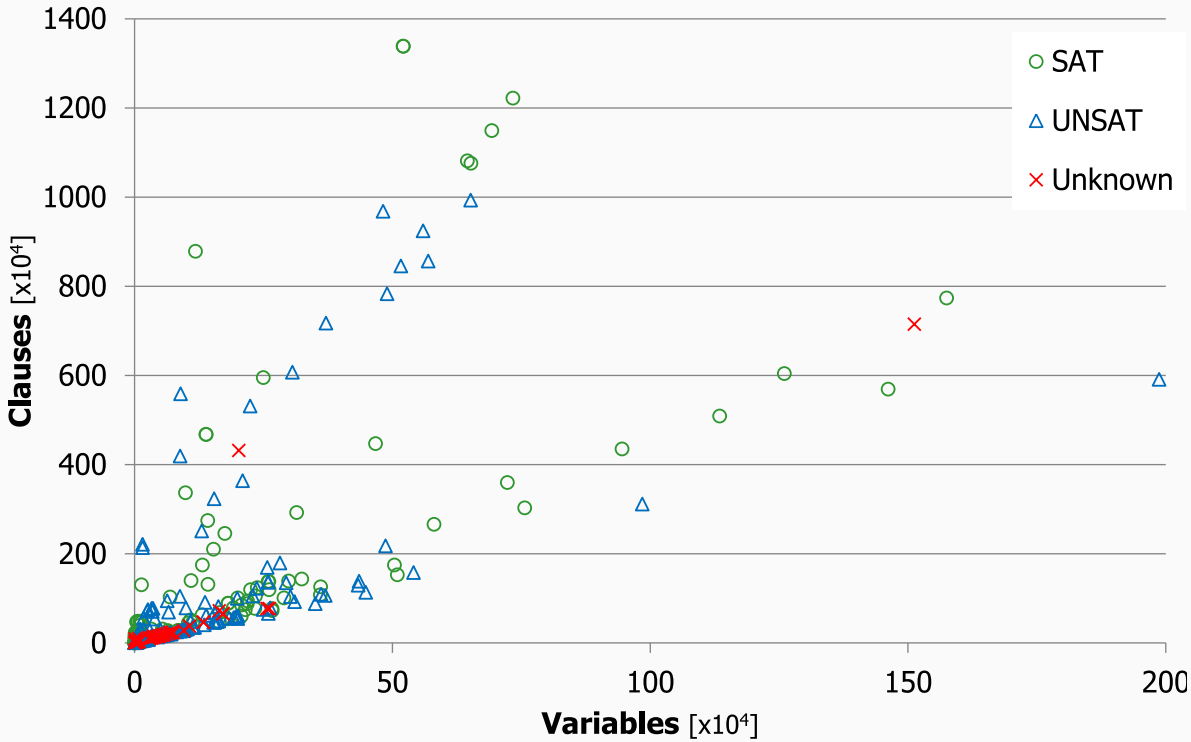


アニーリング解は問題の制約を満たす「正しい解」であるとは限らない。

本プロジェクトによる SpoonQ の改造



SAT ソルバの現状



SAT 2011 競技会 Application 部門 300 問

http://www-erato.ist.hokudai.ac.jp/docs/seminar/nabeshima.pdf

効果と展望

期待される効果

- SAT Solver としてのアニーリングマシンの活用推進
  既存の SAT ソルバ利用者をアニーリングに引き込む
- 多くの人が関わることによって、将来的に既存の SAT Solver とアニーリングを競争させることができる

今後の予定

- 11 月まで · · · 作成中のソルバが一通り機能する状態
- 未踏期間 · · · さらなる最適化の実装や性能評価、ドキュメント作成

QA

最適化関数の生成

論理関数を実数値最適化関数に変換する。
論理変数 $x_i \in \{True, False\}$ を実数変数 $x_i^{(A)}, x_i^{(B)}, \ldots \in \mathbb{R}$ に変換する際に以下の戦略をとるとする。

戦略	$x =$ *True*	$x =$ *False*
A	$x^{(A)} = 0$	$x^{(A)} = 1$
$\overline{A}$	$x^{(\overline{A})} = 1$	$x^{(\overline{A})} = 0$
B	$x^{(B)} = 0$	$x^{(B)} \neq 0$
C	$x^{(C)} = 0$	$x^{(C)} > 0$

論理関数 $y = f(x_1, x_2)$ は y, x_1, x_2 の戦略に応じて複数通り (戦略が N 個なら最大 N^3 通り) の最適化関数で表される。

最適化関数の生成例

論理関数 $f(x_1, x_2) = x_1 \wedge x_2,\ g(x_1, x_2) = x_1 \vee x_2$ を変換する。最適化関数は

$$
\begin{aligned}
f^{(\overline{A})} &= x_1^{(\overline{A})} x_2^{(\overline{A})} = (1 - x_1^{(A)})(1 - x_2^{(A)}) \\
f^{(A)} &= 1 - x_1^{(\overline{A})} x_2^{(\overline{A})} = 1 - (1 - x_1^{(A)})(1 - x_2^{(A)}) \\
f^{(\overline{C})} &= x_1^{(\overline{C})} x_2^{(\overline{C})}, \quad f^{(\overline{B})} = x_1^{(\overline{B})} x_2^{(\overline{B})} \\
g^{(\overline{C})} &= x_1^{(\overline{A})} + x_2^{(\overline{A})} = x_1^{(\overline{C})} + x_2^{(\overline{C})}
\end{aligned}
$$

等が生成できる。
進捗状況: これから実装

SAT Solver

SAT Solver の単純化されたアルゴリズム

- 文字が一個のみの節がある場合は、その文字を True とみなす
- 全節の中に肯定と否定の両方が含まれない文字がある場合、それを True とみなす
- 適当な文字を選択し、True の場合と False の場合でそれぞれ探索する
- 以上を、解もしくは矛盾がみつかるまで繰り返す

minisat をベースにアルゴリズムを開発 (並列化 (多プロセッサ) 対応等が課題)
進捗状況: 調査中