記述計算量と論理式の発見

Skip Jordan
JST ERATO湊離散構造処理系プロジェクト

概要

計算量理論では帰着が非常に重要なものになる。何かを証明する時は帰着の存在、帰着で使用できるガジェット、又は写像が帰着ではないと証明するための反例等を考えることもよくある。近年は充足可能性問題（SAT）ソルバを応用し、ガジェット[2]、帰着の発見[1, 3]やプログラム合成の研究がある。記述計算量理論では、写像（ブール写像、つまり決定問題も含む）を論理式で表し、写像の複雑さを定義する論理式の必要な記述能力によって定義する。ここでは、記述計算量の視点から始まり論理式の発見を課題にし、計算量理論の応用を紹介する。帰着の発見に集中するが、それ以外にも応用が様々ある[4]。

プロパティーPからQへの帰着$\mathbf{r}$は、

$$\exists \mathbf{r}\, \forall \mathbf{x} : \mathbf{x} \in P \leftrightarrow \mathbf{r}(\mathbf{x}) \in Q \qquad (1)$$

を満たす写像である。一般的には$\mathbf{r}$の存在は決定不可能であるが、$\mathbf{x}$と$\mathbf{r}$を有限クラスに限って考えると$\mathbf{\Sigma_2^p}$のような問題になる。帰着以外にも似ている条件になる。例えば、ある二階述語論理式$\Phi(\mathbf{x}, \mathbf{y})$と同値なLFP式$\varphi(\mathbf{x}, \mathbf{y})$を探す時は

$$\exists \phi \forall \mathbf{x}, \mathbf{y} : \Phi(\mathbf{x}, \mathbf{y}) \iff \phi(\mathbf{x}, \mathbf{y})$$

を満たすφが探したい。ϕとはφのアウトラインを決めてから内容を表すブール変数のタプルである[4]。

$\mathbf{\Sigma_2^p}$の問題を解く方法

1. QBFソルバ：qdimacs（CNF）やqpro（NNF）
2. ASPソルバ：disjunctiveソルバ
3. SATソルバやBDDとCEGAR：

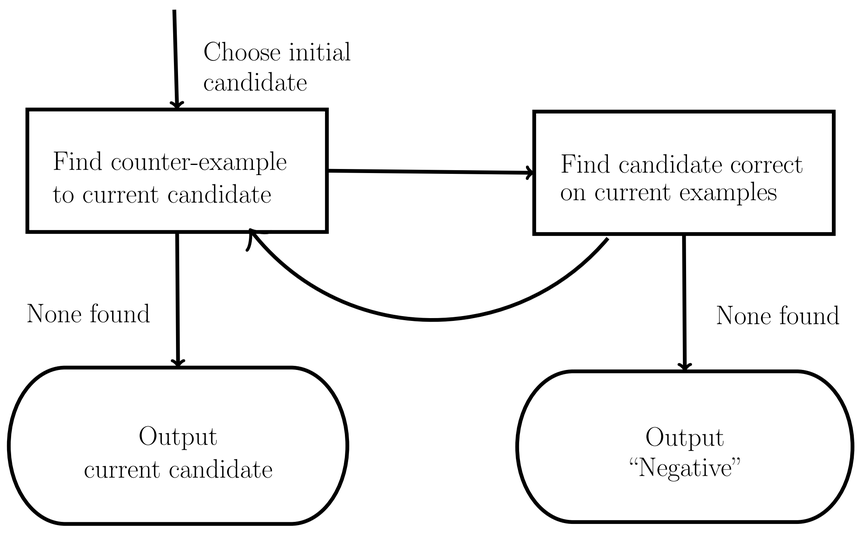

Figure 1: **C**ounter-**e**xample **g**uided **a**bstraction **r**efinement (CEGAR)

ここでは、反例が関係ストラクチャーでcandidateが論理で表すクエリである。我々の実装ではクエリのアウトラインと反例を計算する手法（論理式等）を与えると上の様に学習ができる。

なぜ記述計算量？

記述計算量では帰着等は論理式として表すので、SATやQBFに書き換えやすい。しかしそれだけではなく、様々な標準形もあり学習での仮説空間を決める時有効だと考えられる。ある計算が多項式時間でできないと証明するのが難しいが、一階述語論理で表せないと証明する方法がいくつかある。つまり記述計算量理論で使用する帰着は多項式時間の帰着等より弱く期待できるかもしれないが、完全性等に対して力が充分ある。

帰着以外の応用

帰着以外にも応用がたくさんあり、最近[4]は次のような応用をしている。

1. 入力の2階述語論理式と同値になるLFP式の発見
   - 指数時間のプログラムと同値になる多項式時間プログラムの発見
   - 例：形式検証で使うゲームソルバ。パリティゲームはμ計算と同型で NP∩coNPにあるが、Pには？
2. ボードゲームをサンプルプレイから学習
   - 駒の動き等を定義する式等
   - 従来より速い場合がある。
3. 反例の発見
   - 証明のチェック
   - 論理式を書く時のチェック
   - 宿題等の採点の道具
   - ガジェット等の発見

パーフォーマンス比較

SATソルバの場合はGLUEMINISAT、QBFソルバはRAREQSやDEPQBF、ASPソルバはCLASPDが最もパーフォーマンスが良い。[1]のREDFINDと48個の決定問題の間2304個のサーチを、2分のタイムアウトに対していくつかのパラメータの結果を見る。他にQUBE、SKIZZO、CIRQIT、CMODELS、GNT2、CRYPTOMINISAT等も比較している。

	$c=1\ n=3$	$c=2\ n=3$	$c=3\ n=3$	$c=1\ n=4$	$c=2\ n=4$	$c=3\ n=4$
DE-GMS	0 (100.0%)	0 (100.0%)	10 (99.6%)	0 (100.0%)	5 (99.8%)	103 (95.5%)
DE-CUDD	0 (100.0%)	116 (95.0%)	537 (76.7%)	0 (100.0%)	186 (91.9%)	722 (68.7%)
RAREQS	0 (100.0%)	0 (100.0%)	16 (99.3%)	19 (99.1%)	65 (97.1%)	204 (91.1%)
DEPQBF	0 (100.0%)	142 (93.8%)	547 (76.2%)	16 (99.3%)	297 (87.1%)	711 (66.1%)
GRINGO	40 (98.3%)	393 (82.9%)	590 (74.4%)	72 (96.9%)	593 (74.3%)	836 (63.7%)
LPARSE	51 (97.8%)	396 (82.8%)	605 (73.7%)	75 (96.7%)	635 (72.4%)	850 (63.1%)
RedFind	1 (99.9%)	152 (93.4%)	396 (82.8%)	2 (99.9%)	347 (84.9%)	547 (76.3%)

Table 1: Number of timeouts (% solved), $k = 1$.

難しい場合はCEGARが最も速い。例えば、coNL完全なREACHからNL完全なREACHへの帰着（Immerman-Szelepcsényiの定理）に対し、パラメータを増やす。ここでは、DEPQBF等はかなり遅くなる。

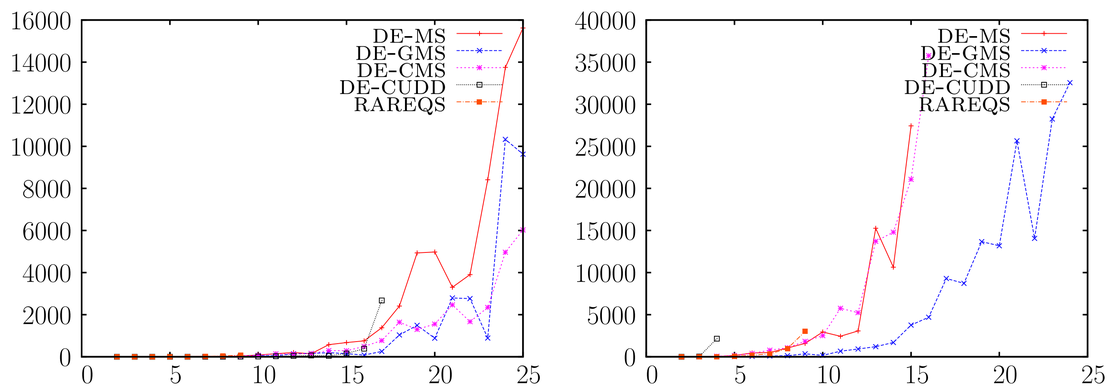

Figure 2: Performance (seconds) scaling n with $c = 1, 2$ (left,right)

参考文献

[1] Michael Crouch, Neil Immerman, and J. Eliot B. Moss. Finding reductions automatically. In *Fields of Logic and Computation – Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *LNCS*, pages 181–200. Springer, 2010.

[2] Alejandro Erickson and Frank Ruskey. Domino tatami covering is NP-complete. In *International Workshop on Combinatorial Algorithms (IWOCA) 2013, arXiv:1305.6669*, 2013.

[3] Charles Jordan and Łukasz Kaiser. Experiments with reduction finding. In *Theory and Applications of Satisfiability Testing (SAT) 2013*, volume 7962 of *LNCS*, pages 192–207. Springer, 2013.

[4] Charles Jordan and Łukasz Kaiser. Learning programs as logical queries. In *ICALP 2013 Satellite Workshop on Learning Theory and Complexity (LTC)*, 2013.