Software Engineering and Applications 软件工程与应用, 2020, 9(1), 49-55
Published Online February 2020 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2020.91006

Hans汉斯

A Decision Procedure of EUF Formulas Based on Eager Encoding

Mengqi Cheng, Siran Lei, Jianguo Jiang

School of Mathematics, Liaoning Normal University, Dalian Liaoning
Email: cmq215box@sina.com, siranlei0922@gmail.com, jjgbox@sina.com

Received: Jan. 23rd, 2020; accepted: Feb. 6th, 2020; published: Feb. 13th, 2020

Abstract

In the formal verification community, a decision procedure of EUF formulas based on eager encoding method has practical advantages in solving certain problems. It also promotes the option of deciding a combination of theories by reducing them to this logic. In this paper, in order to find an efficient decision procedure of EUF formulas based on eager encoding, an eager approach is studied that an EUF formula is reduced to propositional logic formula by Ackermann's method and an algorithm based on a nonpolar equality graph. The DP_{LIB} library written in C++ language is provided for simplifying the development of the procedure. After adding the code to implement the above algorithms and compiling it on Linux system, it can be called to solve an EUF formula faster.

Keywords

EUF Formula, Ackermann's Reduction, Equality Graph

基于活性编码的EUF公式的判定过程

程梦奇，雷斯然，江建国

辽宁师范大学数学学院，辽宁 大连
Email: cmq215box@sina.com, siranlei0922@gmail.com, jjgbox@sina.com

收稿日期：2020年1月23日；录用日期：2020年2月6日；发布日期：2020年2月13日

摘　要

在形式化验证领域，EUF公式的活性编码判定过程解决某些问题具有实际的优势，判定组合理论时，该方法选择将其归约为命题公式，更易于求解。本文以寻找高效的EUF公式的活性编码判定过程为目的，研究了一个活性编码方法，即通过Ackermann归约算法以及基于非极性等式图的归约算法，将EUF公式

文章引用: 程梦奇, 雷斯然, 江建国. 基于活性编码的 EUF 公式的判定过程[J]. 软件工程与应用, 2020, 9(1): 49-55.
DOI: 10.12677/sea.2020.91006

归约为等可满足性的命题公式。为了简化这一方法的实现进程，本文基于C++语言编写的DP_{LIB}库，根据上述算法添加程序并成功编译于Linux系统，可以调用它更快的求解EUF公式。

关键词

EUF公式，Ackermann归约，等式图

1. 引言

在形式化验证领域，带未解释函数的等式逻辑公式(EUF)的判定过程得到了广泛的应用，很多实际问题可以建模成 EUF 公式的可满足性判定问题，例如：电路的等价性验证、程序的等价性验证、编译的转换有效性验证等[1] [2] [3]。

到目前为止，针对 EUF 公式的可满足性求解问题，它的判定过程大体上分为两类：惰性编码(lazy encoding)和活性编码(eager encoding)。惰性编码判定过程是目前 SMT 领域内的主流方法，该方法基于 SAT 求解器与理论求解器之间的相互作用，并且理论求解器仅仅在需要时被引入，目的是检测理论文字集的可满足性。对 EUF 公式而言，该方法的主要思想是：首先找到满足公式命题结构的赋值，然后根据理论验证这些赋值。这种方法的缺点是，满足公式命题结构的赋值的数量在公式的大小上可能是指数级的，当理论求解器需要被调入时，可能会遇到瓶颈[4] [5]。

基于活性编码判定 EUF 公式，可以避免遇到这种情况。因为它不依赖于 SAT 求解器和理论求解器的交替使用，而是将一个理论公式归约成一个等可满足性的命题公式，仅仅使用 SAT 求解器判定该命题公式的可满足性即可判定原始公式的可满足性。这个方法的好处在于，它可以充分地利用高效的 SAT 求解器，不用针对 EUF 理论去开发特定的理论求解器。

活性编码方法是早期 SMT 领域内的主要方法，主要包括：per-constraint 方法[6]和 small-domain instantiation 方法[7]。一个高效的将 EUF 公式归约为命题公式的方法，不仅能够提高其求解效率，还能够将它与命题逻辑级的其他理论整合在一起[8] [9] [10]。该方法的求解效率依赖于 EUF 公式归约为命题公式的效率和 SAT 求解器自身的效率。因此寻找一个好的活性编码方法对 EUF 公式的求解至关重要。

为验证 EUF 公式的可满足性，本文给出了一种高效的活性编码判定方法，首先消去公式中的未解释函数，依靠函数的共同属性——函数一致性，在第二节主要介绍把判定 EUF 公式归约为判定等式公式的 Ackermann 归约算法，这个方法通过增加一些约束来加强函数一致性。然后消去公式中的等式谓词，依靠等式的传递性，在第三节提出基于非极性等式图将等式逻辑公式归约为命题逻辑公式的算法。最后，在 DP_{LIB} 库[1]的基础上，添加实现上述算法的 C++程序，并成功在 Linux 系统上编译，可以调用它更快的求解 EUF 公式。

2. Ackermann 归约

本节研究的是将 EUF 公式归约为等可满足性的等式公式的方法。首先，通过一个例子简要介绍 EUF 公式，$x = y \wedge G(F(x)) = G(y)$ 是一个 EUF 公式，其中 x 和 y 是无限数域上的变量，F 和 G 是未解释函

[1] http://www.decision-procedures.org/

数， $x=y$ 和 $G\left(F\left(x\right)\right)=G\left(y\right)$ 是原子公式。然后，给出 EUF 公式的语法定义。

定义 2.1. 带未解释函数的等式逻辑公式(EUF)的语法定义如下：

公式：公式∧公式 |¬公式 |(公式)| 原子

原子：项 = 项 | 谓词符号(项列表)

项：标识符 | 函数符号(项列表)

给定一个 EUF 公式，通常用 φ^{UF} 表示。其中，标识符是定义在一个无限数域上的变量。

为验证 EUF 公式的有效性，本文提到了 Ackermann 归约算法，该方法将判定该公式归约为判定等式逻辑公式，其中涉及一个重要的内容，即函数一致性规则：同一个函数的不同实例参数相等时，则返回相等的函数值[11]。

为消去 EUF 公式中的未解释函数，Ackermann 归约对公式添加了明确的约束，目的是加强函数一致性。该算法读到了一个 EUF 公式 φ^{UF}，验证该公式是否有效，下面把它转化为一个等式公式 φ^{E}，如下所示：

$$\varphi^{E} \coloneqq FC^{E} \Rightarrow flat^{E}, \tag{2.1}$$

其中，FC^{E} 是函数一致性约束的合取式，$flat^{E}$ 是 φ^{UF} 的扁平化公式，公式中每一个函数实例用一个新变量来替换。

根据输入的公式做出一些假设：该公式只含有一个未解释函数且单参，并且不存在有相同参数值的两个函数实例。然后，给出简化的 Ackermann 归约算法。

算 法 2.1：Ackermann 归约

输 入： 带未解释函数 F 的 m 个函数实例的 EUF 公式 φ^{UF}

输 出： 等式公式 φ^{E}，当且仅当 φ^{UF} 有效时，φ^{E} 有效

1. 由子表达式向外将索引赋给未解释函数实例，并且用 F_i 表示 F 的索引为 i 的函数实例，$\arg\left(F_i\right)$ 表示 F_i 的单参.
2. 令 $flat^{E} \doteq \mathcal{T}\left(\varphi^{UF}\right)$，$\mathcal{T}$ 是一个函数。该函数的作用是用新的项变量 f_i 替换每一个未解释函数实例 F_i（在有嵌套函数的情况下，只有对应最外层函数实例的项变量被保留下来）将输入的 EUF 公式（或项）转换为一个等式公式（或项）.
3. 令 FC^{E} 表示函数一致性约束的合取式：
$$FC^{E} \coloneqq \bigwedge_{i=1}^{m-1}\bigwedge_{j=i+1}^{m}\left(\mathcal{T}\left(\arg\left(F_i\right)\right)=\mathcal{T}\left(\arg\left(F_j\right)\right)\right)\Rightarrow f_i=f_j.$$
4. 令 $\varphi^{E} \coloneqq FC^{E} \Rightarrow flat^{E}$，并返回 φ^{E}.

例 2.2. 考虑公式，

$$\left(x=y\right)\vee\left(F\left(x\right)\neq F\left(y\right)\right)\vee\left(F\left(x\right)=F\left(z\right)\right), \tag{2.2}$$

使用算法 2.1 将该公式归约为等式公式。

观察该公式，假设从左到右依次找到关于 F 的函数实例，并且将索引赋给它们，根据这个步骤可以计算出 $flat^{E}$ 和 FC^{E}，如下：

$$flat^{E} \coloneqq \left(x=y\right)\vee\left(f_1\neq f_2\right)\vee\left(f_1=f_3\right), \tag{2.3}$$

$$FC^E := (x = y \Rightarrow f_1 = f_2) \wedge (x = z \Rightarrow f_1 = f_3) \wedge (y = z \Rightarrow f_2 = f_3). \tag{2.4}$$

当且仅当等式公式(2.1) φ^E 有效，原始 EUF 公式(2.2)是有效的。

对于更为复杂的 EUF 公式而言，如何通过 Ackermann 归约算法将它们归约为等式公式，接下来给出两个例子，展示该算法的扩展应用过程：

例 2.3. 给定公式：

$$\big((x = v \wedge y = F(x,v) \wedge z = F(y,v)) \wedge w = F(F(v,v),v)\big) \Rightarrow z = w. \tag{2.5}$$

用对应的项变量替换每一个函数实例，计算出 $flat^E$：

$$flat^E := \big((x = v \wedge y = f_1 \wedge z = f_2) \wedge w = f_4\big) \Rightarrow z = w. \tag{2.6}$$

计算出函数一致性约束的合取式，即：

$$\begin{aligned} FC^E := {} & \big((x = y \wedge v = v) \Rightarrow f_1 = f_2\big) \wedge \big((x = v \wedge v = v) \Rightarrow f_1 = f_3\big) \\ & \wedge \big((x = f_3 \wedge v = v) \Rightarrow f_1 = f_4\big) \wedge \big((y = v \wedge v = v) \Rightarrow f_2 = f_3\big) \\ & \wedge \big((y = f_3 \wedge v = v) \Rightarrow f_2 = f_4\big) \wedge \big((v = f_3 \wedge v = v) \Rightarrow f_3 = f_4\big). \end{aligned} \tag{2.7}$$

最后，等式公式为 $FC^E \Rightarrow flat^E$，接下来验证它的有效性。

上面这个例子展示了，对于多参数函数如何来扩展这个归约法：仅当一对函数实例的所有参数都成对相等时，返回相等的函数值。

事实上读者可以观察到这些约束中的大多数是多余的，公式的有效性仅仅取决于 $F(x,v)$ 与 $F(v,v)$，和 $F(y,v)$ 与 $F(F(v,v),v)$ 这两对函数实例是相等的。因此，在(2.7)式中，只有第二个和第五个约束是必要的。实践中，这样的观察是有必要的，因为函数一致性约束的数量以平方式增长可能会成为一个瓶颈。在这种情况下，通常是在检测公式的有效性之前检测出能够在多项式时间内被移除的比较大的约束集，这种方法的更多细节参考文献[12]。

例 2.4. 检测公式的有效性：

$$x = y \Rightarrow \underbrace{\underbrace{F\Big(\overbrace{G(x)}^{g_1}\Big)}_{f_1}}_{f_2} = \underbrace{\underbrace{F\Big(\overbrace{G(y)}^{g_2}\Big)}_{f_3}}_{f_4}. \tag{2.8}$$

由内而外将索引赋给函数实例，计算得：

$$flat^E := x = y \Rightarrow f_2 = f_4, \tag{2.9}$$

$$\begin{aligned} FC^E := {} & x = y \Rightarrow g_1 = g_2 \wedge g_1 = f_1 \Rightarrow f_1 = f_2 \\ & \wedge g_1 = g_2 \Rightarrow f_1 = f_3 \wedge g_1 = f_3 \Rightarrow f_1 = f_4 \\ & \wedge f_1 = g_2 \Rightarrow f_2 = f_3 \wedge f_1 = f_3 \Rightarrow f_2 = f_4 \\ & \wedge g_2 = f_3 \Rightarrow f_2 = f_4. \end{aligned} \tag{2.10}$$

最后，得：

$$\varphi^E := FC^E \Rightarrow flat^E. \tag{2.11}$$

从这个例子可以很清晰地看出，如何将 Ackermann 归约算法推广应用于多重未解释函数。

假设已经用 Ackermann 归约方法消去公式中的未解释函数，得到了一个等式公式。接下来，提出一个基于非极性等式图将等式逻辑公式归约到命题逻辑公式的方法。

3. 基于非极性等式图的归约

假设给出的等式公式是没有常量的否定范式(NNF)，记为 φ^E， $AT\left(\varphi^E\right)$ 表示 φ^E 的原子集。

定义 3.1. 给定一个等式公式 φ^E，存在对应 φ^E 的非极性等式图，是一个无向图 (V,E)，其中 V 中的节点对应 φ^E 中的变量，E 中的边对应 φ^E 中的等式谓词 $AT\left(\varphi^E\right)$，用 $G_{NP}^E\left(\varphi^E\right)$ 表示。

非极性等式图是等式图的一个退化版本，因为它忽略了等式谓词的极性[13] [14]。等式公式基于图方法归约到命题公式是由 Bryant 和 Velev 提出的一个判定过程[15]。

给定一个等式公式 φ^E，由该公式产生两个命题公式 $e\left(\varphi^E\right)$ 和 B_{trans}，

$$\varphi^E \text{ 是可满足的} \Leftrightarrow \quad e\left(\varphi^E\right)\wedge B_{trans} \text{ 是可满足的。} \tag{3.1}$$

$e\left(\varphi^E\right)$ 和 B_{trans} 定义如下：

1) $e\left(\varphi^E\right)$ 是 φ^E 的命题架，φ^E 中形如 $x_i = x_j$ 的每个等式谓词由一个新的布尔变量 $e_{i,j}$ 来替换。举例，

$$\varphi^E \coloneqq x_1 = x_2 \wedge \left(\left(\left(x_2 = x_3\right)\wedge\left(x_1 \neq x_3\right)\right)\vee\left(x_1 \neq x_2\right)\right). \tag{3.2}$$

编码，得

$$e\left(\varphi^E\right) \coloneqq e_{1,2} \wedge \left(\left(e_{2,3}\wedge\neg e_{1,3}\right)\vee\neg e_{1,2}\right). \tag{3.3}$$

由此可见，如果 φ^E 是可满足的，那么 $e\left(\varphi^E\right)$ 是可满足的。然而，其它情况是不成立的，比如在上述示例中，虽然公式(3.2)是不可满足的，但是它的编码(3.3)是可满足的，这是因为在编码过程中丢失了等式关系的传递。为了保持等可满足性，则需要添加一些约束，即 B_{trans}。

2) B_{trans} 是一个蕴含合取式，被称为传递约束。每个蕴含合取式都与非极性等式图中的一个回路相关，一个具有 n 个边的回路，当其中 $n-1$ 个边被赋值为 TRUE 时，B_{trans} 禁止赋值 FALSE 给剩余的那个边。给每个回路中的每个边强加此约束是(3.1)成立的充分条件。

例 3.2. 在一个非极性等式图中，原子 $x_1 = x_2, x_2 = x_3, x_1 = x_3$ 形成了一个具有 3 个边的回路。下面的约束充分满足了公式(3.1)中等式关系的传递性：

$$B_{trans} = \left(\left(e_{1,2}\wedge e_{2,3}\Rightarrow e_{1,3}\right)\wedge\left(e_{1,2}\wedge e_{1,3}\Rightarrow e_{2,3}\right)\wedge\left(e_{2,3}\wedge e_{1,3}\Rightarrow e_{1,2}\right)\right). \tag{3.4}$$

然而，给每个回路增加 n 个约束并不实际，因为在一个给定的无向图上可能会存在指数式数量的回路。

定义 3.3. 回路中两个不相邻节点连成的边，被称为回路的弦。在给定的图中，一个没有弦的回路被称为无弦回路。

定理 3.4. 等式公式与编码后的命题公式等可满足性成立的充分条件是，在简单无弦回路上增加传递约束。

例 3.5. 考虑图 1 中的回路 $\left(x_3, x_4, x_8, x_7\right)$，该回路包含弦 $\left(x_3, x_8\right)$，因此，它不是无弦回路。假设赋 TRUE 给这个回路中除了 $\left(x_3, x_4\right)$ 的所有边。如果 $\left(x_3, x_8\right)$ 被赋值为 TRUE，那么对简单无弦回路 $\left(x_3, x_4, x_8\right)$ 的赋值与传递性冲突。如果 $\left(x_3, x_8\right)$ 被赋值为 FALSE，那么对简单无弦回路 $\left(x_3, x_7, x_8\right)$ 的赋值与传递性冲突。因此，在无弦回路上的约束足够阻止赋给包含一个弦的回路违反传递性的值。

图中简单无弦回路的数量在顶点数量内仍然可以是指数式的。因此，构造 B_{trans} 使其直接约束每个这样的回路在变量数量内都可以使这个公式的大小成为指数式的。有：

定义 3.6. 在一个给定的无向图中，只含有长度小于 4 的回路(无弦回路)，称之为弦图。

每个图形可以在多项式时间内以顶点数构成，由于弦图中唯一的无弦回路是三角形，这表示将定理 3.4 应用于这样的图形会导致在变量数量内的公式大小不超过三次方(为图中的每个三角形构造三个约束)。

算 法 3.1： 等式公式归约为命题公式

输 入： 等式公式 φ^E

输 出： 与等式公式 φ^E 等可满足性的命题公式

1. 用新的布尔变量 $e_{i,j}$ 来替换 φ^E 中形如 $x_i = x_j$ 的每个原子来重构布尔公式 $e\left(\varphi^E\right)$.
2. 构造非极性等式图 $G_{NP}^E\left(\varphi^E\right)$.
3. 使 $G_{NP}^E\left(\varphi^E\right)$ 成为有弦图.
4. $B_{trans} := TRUE$.
5. 对在 $G_{NP}^E\left(\varphi^E\right)$ 中的每一个三角形 $\left(e_{i,j}, e_{j,k}, e_{i,k}\right)$，

$$B_{trans} := B_{trans} \wedge \left(e_{i,j} \wedge e_{j,k} \Rightarrow e_{i,k}\right) \wedge \left(e_{i,j} \wedge e_{i,k} \Rightarrow e_{j,k}\right) \wedge \left(e_{i,k} \wedge e_{j,k} \Rightarrow e_{i,j}\right). \quad (3.5)$$

6. 返回 $e\left(\varphi^E\right) \wedge B_{trans}$.

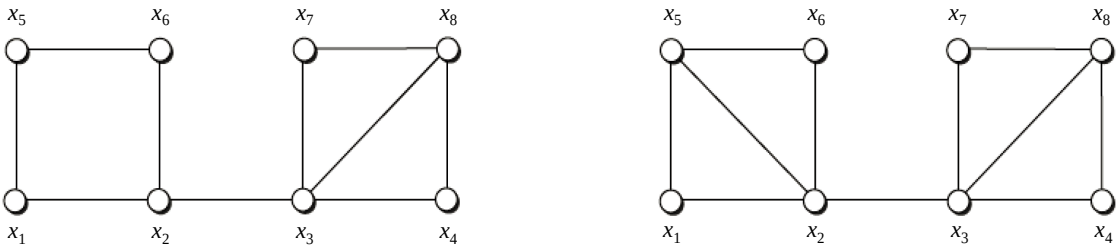

Figure 1. A nonchordal nonpolar equality graph corresponding to φ^E (left), and a possible chordor version of it (right)

图 1. 左图是对应 φ^E 的一个非极性无弦等式图，右图是它的一个可能的弦图

给定一个等式公式 φ^E，算法 3.1 的第二步构造出了对应 φ^E 的非极性等式图，若该图是一个无弦图，则在第三步添加新弦将其弦化，使其成为一个弦图，在这个过程中新添加的弦用出现在 B_{trans} 但不在 $e\left(\varphi^E\right)$ 里的新变量来表示。

例 3.7. 图 1 描绘了使其弦化之前和之后的非极性等式图，用实边表示。图形弦化后，它包含四个三角形，因此，B_{trans} 包含了 12 个约束。例如，三角形 $\left(x_1, x_2, x_5\right)$，它的约束条件是

$$\left(e_{1,2} \wedge e_{2,5} \Rightarrow e_{1,5}\right) \wedge \left(e_{1,5} \wedge e_{2,5} \Rightarrow e_{1,2}\right) \wedge \left(e_{1,2} \wedge e_{1,5} \Rightarrow e_{2,5}\right). \quad (3.6)$$

图形弦化后，添加的弦用一个新的辅助变量 $e_{2,5}$ 表示，出现在 B_{trans} 里而不在 $e\left(\varphi^E\right)$ 里。

4. 结论

本文针对 EUF 公式的可满足性求解问题，给出了一种可实现的活性编码判定过程。DP_{LIB} 库提供了部分基础程序，以简化该判定过程的开发。在这个基础上，添加有关活性编码算法的相关实现，并已于 Linux 系统中成功编译，加快了 EUF 公式的求解速度。但是本文中的两个算法中分别添加的函数一致性约束合取式与传递约束合取式的数量对求解效率可能有一定的阻碍，如何优化本文中的活性编码方法使其求解效率更高，还有待进一步地研究。

参考文献

[1] Kroening, D. and Strichman, O. (2016) Equality Logic and Uninterpreted Functions. In: *Decision Procedures. Texts in Theoretical Computer Science. An EATCS Series*, Springer, Berlin, 77-95. https://doi.org/10.1007/978-3-662-50497-0_4

[2] Lei, B.U. and Bao, X.D. (2014) Formal Verification of Hybrid System. *Journal of Software*, **25**, 219-233.

[3] Bryant, R.E., German, S. and Velev, M.N. (2001) Processor Verification Using Efficient Reductions of the Logic of Uninterpreted Functions to Propositional Logic. *ACM Transactions on Computational Logic*, **2**, 1-41. https://doi.org/10.1145/371282.371364

[4] Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A. and Peled, D. (2004) DPLL(T): Fast Decision Procedures. In: Alur, R. and Peled, D.A., Eds., *Computer Aided Verification. CAV* 2004. *Lecture Notes in Computer Science*, Volume 3114, Springer, Berlin, Heidelberg, 175-188. https://doi.org/10.1007/978-3-540-27813-9_14

[5] 金继伟, 马菲菲, 张健. SMT 求解技术简述[J]. 计算机科学与探索, 2015, 9(7): 769-780.

[6] Seshia, S.A., Lahiri, S.K. and Bryant, R.E. (2003) A Hybrid SAT-Based Decision Procedure for Separation Logic with Uninterpreted Functions. In: *Proceedings of the* 40*th Annual Design Automation Conference*, ACM, New York, 425-430. https://doi.org/10.1145/775832.775945

[7] Pnueli, A., Rodeh, Y. and Shtrichman, O. (1999) Deciding Equality Formulas by Small Domains Instantiations. In: Halbwachs, N. and Peled, D., Eds., *Computer Aided Verification. CAV* 1999. *Lecture Notes in Computer Science*, Volume 1633, Springer, Berlin, Heidelberg, 455-469. https://doi.org/10.1007/3-540-48683-6_39

[8] 王翀, 吕荫润, 陈力, 王秀利, 王永吉. SMT 求解技术的发展及最新应用研究综述[J]. 计算机研究与发展, 2017, 54(7): 1405-1425.

[9] Kroening, D. and Strichman, O. (2009) A Framework for Satisfiability Modulo Theories. *Formal Aspects of Computing*, **21**, 485-494. https://doi.org/10.1007/s00165-009-0105-z

[10] Strichman, O. (2002) On Solving Presburger and Linear Arithmetic with SAT. In: Aagaard, M.D. and O'Leary, J.W., Eds., *Formal Methods in Computer-Aided Design. FMCAD* 2002. *Lecture Notes in Computer Science*, Volume 2517, Springer, Berlin, Heidelberg, 160-170. https://doi.org/10.1007/3-540-36126-X_10

[11] Bryant, R.E., German, S. and Velev, M.N. (1999) Exploiting Positive Equality in a Logic of Equality with Uninterpreted Functions. In: Halbwachs, N. and Peled, D., Eds., *Computer Aided Verification. CAV* 1999. *Lecture Notes in Computer Science*, Volume 1633, Springer, Berlin, Heidelberg, 470-482. https://doi.org/10.1007/3-540-48683-6_40

[12] Pnueli, A. and Strichman, O. (2006) Reduced Functional Consistency of Uninterpreted Functions. *Electronic Notes in Computer Science*, **144**, 53-65. https://doi.org/10.1016/j.entcs.2005.12.006

[13] Rodeh, Y. and Shtrichman, O. (2001) Finite Instantiations in Equivalence Logic with Uninterpreted Functions. In: Berry, G., Comon, H. and Finkel, A., Eds., *Computer Aided Verification. CAV* 2001. *Lecture Notes in Computer Science*, Volume 2102. Springer, Berlin, Heidelberg, 144-154. https://doi.org/10.1007/3-540-44585-4_13

[14] Rodeh, Y. and Strichman, O. (2006) Building Small Equality Graphs for Deciding Equality Logic with Uninterpreted Functions. *Information and Computation*, **204**, 26-59. https://doi.org/10.1016/j.ic.2005.08.001

[15] Bryant, R.E. and Velev, M.N. (2002) Boolean Satisfiability with Transitivity Constraints. *ACM Transactions on Computational Logic*, **3**, 604-627. https://doi.org/10.1145/566385.566390