Unit Resolution for a Subclass of the Ackermann Class

YAMASAKI, Susumu; ISHIBASHI, Toshihiko; DOSHITA, Shuji

Unit Resolution for a Subclass of the Ackermann Class

By

Susumu YAMASAKI*, Toshihiko ISHIBASHI* and Shuji DOSHITA*

(Received September 29, 1979)

Abstract

The Ackermann class and the Gödel class are typical subclasses of pure first-order logic. The unsatisfiability problems for the Ackermann class and the Gödel class of formulas are decidable and resolution strategies to the unsatisfiability problems for the Ackermann class and the Gödel class were constructed by W. H. Joyner.

Applying unit resolution of C. L. Chang, we construct a preprocessor to Joyner's resolution strategy for a subclass of the Ackermann class, since his strategy may necessitate too much time and space from the practical point of view.

In this paper, we describe an algorithm to decide whether there is a unit resolution refutation from a set of clauses in a subclass ACK_2 of the Ackermann class, in which at most two literals with variables appear in each clause. In this algorithm, we represent the unit clause resolvents generated by unit resolution by means of finite automata. Also, we transform the decision problem of a unit resolution refutability for ACK_2 to the emptiness problem of intersections of two regular languages. We give the time complexity and the space complexity of the constructed algorithm.

This result is an extension of the result by N. D. Jones namely that it can be decided in deterministic polynomial time whether or not ther is a unit resolution refutation for the propositional logic.

1. Introduction

Pure first-order logic is a subclass of the first-order logic in which no function symbols appear in prenex normal form, where the prenex normal form consists of the prefix containing quantifiers and the matrix without quantifiers.

The Ackermann class, the Gödel class and the propositional logic are its typical subclasses for which the unsatisfiability problems are solvable.

The Ackermann class consists of formulas having the prefix such as $\exists^* \forall \exists^*$ (* denotes arbitrarily many times of occurrences of quantifiers) in prenex normal form. The Gödel class consists of formulas having the prefix such as $\exists^* \forall \forall \exists^*$ in prenex

* Department of Information Science.

normal form. The halting problem for the class of program schemes, in which only one program variable appears and function symbols but variables must be one-place, can be transformed to the unsatisfiability problem for the Ackermann class. The halting problem for the class of program schemes, in which only two program variables appear and function symbols but variables must be two-place can be transformed to the unsatisfiablity problem for the Gödel class.

Resolution strategies are constructed in [7] as decision procedures to solve the unsatisfiability problems for the Ackermann class and for the Gödel class. Since it seems impossible that those 'complete' resolution strategies work in deterministic polynomial time, more effective means are necessary.

On the other hand it, was shown in [6] that unit resolution refutation (proof) by C. L. Chang works in deterministic polynomial time for propositional logic.

In this paper, we apply unit resolution to a subclass of the Ackermann class as a preprocessing of the unsatisfiablity problem for the class, which properly includes propositional logic.

The terminology on the resolution in the first-order logic is used according to [4]. The resolution is applied to a set of clauses which corresponds to a formula in Skolem standard form. The Skolem standard form of a formula is obtained from its prenex normal form, the existential quantifiers being replaced by the Skolem functions. The Ackermann class is abbreviated to ACK after this.

We assume that no common variables are contained in distinct clauses for the unification algorithm to be effective. The Skolem functions are either constants or one-place functions for the set of clauses in ACK. The subclass ACK_2 of ACK is defined as the class of formulas which take the form of the set of clauses each of which contains at most two literals with a variable.

We discuss the unit resolution refutation for ACK_2 and construct an algorithm to decide the unit resolution refutability in $O(n^9)$ time and $O(n^3)$ space for a set of clauses of length n in ACK_2. In constructing the algorithm, we utilize regular expressions to represent unit clause resolvents generated by unit resolution for ACK_2.

2. Unit Resolution Refutability for a Subclass (ACK_2) of the Ackermann Class

Definition1 [4] A unit resolution is a resolution in which a resolvent is obtained by using at least one unit parent clause or a unit factor of a parent clause. A unit deduction is a deduction in which every resolution is a unit resolution. A unit (resolution) refutation is a deduction of (the empty clause).

Definition 2 The Ackermann class, abbreviated to ACK, is the subclass of the pure first-order logic which consists of formulas having the prefix such as $\exists^* \forall \exists^*$ in prenex normal form, where* denotes arbitrarily many times of occurrences of quantifiers.

Each clause contains at most one variable in any set of clauses in ACK.

Definition 3 The subclass ACK_2 of ACK is defined as the class of formulas which take the form of the set of clauses each of which contains at most two literals with a variable.

We suppose in this section that the variable in the non-unit clause is not substituted in obtaining resolvents as long as there are resolvensts from the parent clause by unit resolution keeping its variables unchanged. Cleary this assumption does not influence the existence of unit refutations at all. The following remark is taken into considerations. Unit factors from a clause can be obtained after the variable in the clause being unified to constnts, since there is at most one variable in a clause in ACK (ACK_2).

Infinitely many resolvents can be generated by unit resolution, only for the reason that the function nesting of resolvents may be arbitrarily deep. The unit clause resolvent in ACK_2 has a term prescribed by the term in another unit clause resolvent. Also, unit clause resolvents whose terms constitute periodic relations each other will be generated by unit resolution. To check this case, we will represent the predicate symbol and a part of the terms in the unit clauses as the state of an automaton and the generated unit clause as the transition in the automaton from the intial state to the state which consists of the predicate symbol and a part of the terms in the unit clause. The transition from one state g_1 to another state g_2 in the automaton corresponds to the relation between a part of the terms in the unit clause for g_1 and a part of the terms in the unit clause for g_2. The relation is expressed by an addition or deletion of function symbols.

Based on the above considerations, we construct a finite automaton simulating unit resolution for ACK_2 and represent unit clause resolvents by regular expressions. Then we transform the decidability problem of unit refutablity for ACK_2 to the emptiness problem of intersections of two regular languages.

2.1 Representation of Sets of Clauses in ACK_2

For further discussions, the input set of clauses in ACK_2 are represented as follows. Parentheses are sometimes omitted.

Definition 4 (1) For a literal L, let L be in $Pr\times T_1\circ T_2$: (a) Pr is a set of predicate symbols with or without negation signs. (b) $T_1\circ T_2$ is a decomposition form of terms such $T_2\subset Fn^*\circ(V\cup Ct)$, and T_1 is a set of tuples consiting of prefixes of terms, where Fn is a set of function symbols, Fn^* denotes $\varepsilon\cup Fn\cup Fn\circ Fn\cup\ldots$ (ε denotes the identity function), V is a set of variables, Ct is a set of constants and the operation '$\circ$' is as follows, being identified with the concatenation; $a\circ\alpha=a$, $\varepsilon\circ\alpha=\alpha$, $f\circ\alpha=f(\alpha)$, $(f\cup g)\circ\alpha=f\circ\alpha\cup g\circ\alpha$, and $f^{-1}\circ f\circ\alpha=\alpha$ for a in Ct, f and g in Fn, α in $Fn^*\circ(V\cup Ct)$. $(\alpha_1, \alpha_2, \ldots,$

$\alpha_n)\circ(\beta\cup\gamma)=(\alpha_1\circ\beta\cup\alpha_1\circ\gamma,\ \alpha_2\circ\beta\cup\alpha_2\circ\gamma,\dots,\alpha_n\circ\beta\cup\alpha_n\circ\gamma)$, and $P(\alpha\cup\beta)=P(\alpha)\cup P(\beta)$ for α and β in $T_1\circ T_2$, and P in Pr.

The part $T_1\circ T_2$ of L is defined as follows. The part T_2 is related to the substitutions in the unification of terms.

The part T_1 is an invariant of terms. (i) ε is not contained in T_2. (ii) If a term of L contains the function symbol and the variable (the variable x is regarded as $\varepsilon(x)$), the terms of L are decomposed so that the part T_2 consains a variable and as many function symbols as possible. The constants are contained in T_1. (iii) If the terms of L contain no function symbols and no variables, and $L=P\ (a_1, a_2,\dots, a_n)$ for P in Pr and each a_i $(1\leq i\leq n)$ in Ct, then the part $T_1\circ T_2$ of L can be decomposed as $(a_1', a_2',\dots,a_n')\circ(\mathrm{b})$ for each constant symbol b where only the occurrence of the common constant symbol b among $a_1, a_2,\dots, a_n$ is replaced by ε. The other constant symbols are left unchanged. (L is represented as a subset of $Pr\times T_1\circ T_2$).

(2) The clause is represented as a union of representations of all literals in the clause.

(3) The set of clauses is represented as a set consisting of clauses represented by (2).

Example 1 For $L=Q(f(x), g(x), x)$ in whith Q denotes a predicate symbol, f and g denote function symbols, and x denotes a variable, L is represented as $Q(f, g, \varepsilon)\circ(x)$. For $L=Q(a, f(x))$ in which Q denotes a predicate symbol, f denotes a function symbol, x denotes a variable and a denotes a constant, L is represented as $Q(a, \varepsilon)\circ(fx)$.

For $L=R(a, a, b)$ in which R denotes a predicate symbol, and a and b denote constants, L is represented as $\{R(\varepsilon, \varepsilon, b)\circ(a), R(a, a, \varepsilon)\circ(b)\}$.

The following property is easily derived.

Lemma 1 In the literal represented by Definition 4, the length of the part T_2 is at most 2, and the length of each element of tuples belonging to the part of T_1 is at most one. The expression by Definition 4 is obtained in time $O(n^2)$ and in space $O(n)$ for the set of clauses of length n in ACK_2, where the time complexity and the space complexity are defined by multitape Turing machines.

2.2. Construction of a Finite Automaton Simulating Unit Resolution

We construct a finite automaton simulating the generation of unit clause resolvents by unit resolution from a set of clauses in ACK_2 as is already expressed in Definition 4.

To represent unit clause resolvents by regular expressions, we define the following operation on languages.

Definition 5 Let Σ and Σ^{-1} be alphabets scuh that a one-to-one mapping In from Σ into Σ^{-1} is defined. Let In (a) in Σ^{-1} for a in Σ be denoted by a^{-1}. The operation Op consists of such a reduction on concatenations as follows.

(1) $a\circ a^{-1}=\varepsilon$(empty string) for any a in Σ.

(2) $a\circ b^{-1}\neq\varepsilon$ for $a\neq b$ in Σ.

(3) $b^{-1} \circ a \neq \varepsilon$ for a and b in Σ.

If w in Σ^* is a reduced string by using Op repeatedly from w_0 in $(\Sigma \cup \Sigma^{-1})^*$, such a representation as Op $(w_0) = w$ is used. Let L be a language over $(\Sigma \cup \Sigma^{-1})$ and Red (L) be $\{w \text{ in } \Sigma^* \mid w_0 \text{ in } L, \text{Op}(w_0) = w\}$.

The next theorem is easily obtained from the following algorithm.

Theorem 2 Let L be a regular language over $(\Sigma \cup \Sigma^{-1})$, where Σ and Σ^{-1} are alphabets as in Definition 5. Then Red (L) is also a regular language.

We can construct an algorithm to provide a finite automaton accepting Red (L) from a finite automaton accepting L over $(\Sigma \cup \Sigma^{-1})$.

Algorithm 1 (An algorithm to provide a finite automaton accepting Red (L) from a finite automaton accepting L over $(\Sigma \cup \Sigma^{-1})$)

Input : A finite automaton A_0 accepting L (the transitions of A_0 are given as the set *Inp* of the forms (p, f, q), where p and q are states and f is a transition). Let the length of the input be n.

Output: *A* finite automaton A accepting Red (L) (the set of initial states and set of final states are the same as those of A_0).

Method: Execute the next steps. The procedure is represented by means of Pidgin ALGOL in [1].

```
procedure Red (Inp):
begin
    Out←Inp
    for k←1 step 1 until n do
        for each (p, f⁻¹, q) in Inp containing a symbol f⁻¹ in
            Σ⁻¹ do
            begin
                Out←Out-{(p, f⁻¹, q)};
                for i←0 step 1 until k do
                    begin
                        S₁←{all the states reachable to p with εⁱ};
                        S₂←{all the states reachable to some state in S₁ with f};
                        Out←Out ∪ {r, ε, q) | r∈S₂}
                    end
            end;
    return Out
end
```

Next we provide an algorithm to construct a finite automaton simulating the generation of unit clause resolvents by unit resolution from an input set of clauses in ACK_2, which has the form in Definition 4.

The set of parts $Pr\times T_1$ of literals with some indications corresponds to the set of states of the above automaton. Function symbols, inverse function symbols, constants and variables in T_2 form an input alphabet for the transitions of the automaton. We will represent a unit clause resolvent C by a state g for the part $Pr\times T_1$ of clause C and the part T_2 of terms, which is a reverse string from an initial state to the state g. Therefore, we recognize that a unit clause C can be generated by detecting a string γ from an initial state to the part $Pr\times T_1$ of the clause C, such that γ^{Rev} is the part T_2 of the terms of C.

The outline of constructing a finite automaton is as follows:

(1) For the unit clause C, construct a transition by the part T_2 of the terms of C from an initial state to the part $Pr\times T_1$ of C.

(2) For the unit clause resolvent C which subsumes its parent clause $C_0 (C\subset C_0)$, construct a transition from an initial state to the part $Pr\times T_1$ of C. This is the case that $C_0=C\bigcup_j C_j$ and $\sim C_j'$ exists for each j such that $C_j=C_j'\,\sigma_j$ for some substitution σ_j.

(3) For the unit clause resolvent C_2 whose part of the term is a function or an inverse function of the part of the term of another unit clause C_1, construct a transition by the function between the parts of the terms of C_1 and C_2 from the part $Pr\times T_1$ of C_1 to the part $Pr\times T_1$ of C_2. This is the case that there is a clause $\sim C_1'\cup C_2\cup_i C_i$ such that (a) C_1' and C_2 contain a variable, (b) $\sim C_i$ exists for each i, and (c) C_1 exists and $C_1'\sigma=C_1$ for some non-empty substitution σ.

In (2) or (3), we will decide whether there are $\sim C'_i$, $\sim C'_j$, and C_1 by detecting the transitions from an initial state to the specified states of the constructed automaton. The construction of 3 is not necessary for the clause which contains at most one literal with a variable.

The detailed construction algorithm is given below. The domain of '*Red*' is extended to $Fn\cup Fn^{-1}\cup V\cup Ct$.

Algorithm 2 (A construction of a finite automaton As from a set S of clauses in ACK_2)

Assume that S is in the form by Definition 4.

(Notation) Let $As=(G\cup G_0\cup G_I, \Gamma, \delta, G_0, G_f)$ be a finite automaton, where G_0 denotes an initial state, $G\subset Pr\times T_1\times\{0, 1\}$ and $G_I\subset Pr\times T_1$ are sets of states, $\Gamma=Fn\cup Fn^{-1}\cup V\cup Ct$ for Fn, V, Ct in Definition 4 and $Fn^{-1}=\{f^{-1}$ (inverse function) $|\ f\in Fn\}$, $\delta\subset(G\cup G_0\cup G_I)\times(\Gamma\cup\{\varepsilon\})\rightarrow 2^{(G\cup G_I)}$ is a mapping and G_f denotes a final state. $T(As, G_f)$ denotes the language accepted by a finite automaton with $\{G_f\}$ as a set of final states. $D_gT(As, G_f)$ for $g\in G\cup G_I$ denotes the language $\{u\backslash v\,|\,u\in T(As, g), v\in T(As, G_f)$ and '\' is a left derivative$\}$. $T_2(L)$ denotes the part T_2 of L for a literal L represented by Definition 4. By $[Pr\times T_1]$ (L) we mean the part $Pr\times T_1$ of a literal L.

(Construction of an automaton)

(1) For each literal L represented by Definition 4 such that $T_2(L)=t_1t_2$ or t_2, construct the following transitions.

(a) In case that t_1 is in Ct (the length of $T_2(L)$ should be one): If $T_2(L)=t_1$, then let $\delta(G_0, t_1)\ni[Pr\times T_1](L)\times\{0\}$. (b) In case that t_2 or t_1 is in V: (i) If $T_2(L)=t_1, t_2$ then let $\delta(G_0, t_2)\ni G_{\mathrm{I}}([Pr\times T_1](L))$ and let $\delta(G_{\mathrm{I}}([Pr\times T_1](L)), t_1)\ni[Pr\times T_1](L)\times\{0\}$ for $G_{\mathrm{I}}([Pr\times T_1](L))\in G_{\mathrm{I}}$. (ii) If $T_2(L)=t_1$, then let $\delta(G_0, t_1)\ni[Pr\times T_1](L)\times\{0\}=G_{\mathrm{I}}([Pr\times T_1](L))$.

(2) For each clause S_m represented by Definition 4, construct the following transitions. (a) In case that $S_m=\bigcup_{k=1}^{2} P_k(f_kx)\cup\{\bigcup_j\bigcup_i R_j^i(a_j^i)\}$ for R_k and R_j^i in $Pr\times T_1$, f_k in $Fn\cup\{\varepsilon\}$, a_j^i in Ct and x in V. (i) For each k execute the next steps. $(i-1)$ If $xf_k\in Red(T(As, \sim P_k\times\{0\}))\xi_k$ ($k'\neq k$ and ξ_k is some substitution) and $(\forall_j)$ $(\exists_i)\{a_j^i\in Red(T(As, \sim R_j^i\times\{0\})\eta_j^i\}$ (η_j^i is some substitution), then let $\delta(G_0, x)\ni G_{\mathrm{I}}(P_k)$ and $\delta(G_{\mathrm{I}}(P_k), f_k)\ni P_k\times\{0\}$ for $f_k\neq\varepsilon$, or let $\delta(G_0, x)\ni G_{\mathrm{I}}(P_k)=P_k\times\{0\}$ for $f_k=\varepsilon$. $(i-2)$ $Red(T(As, \sim P_1\times\{0\}))/f_1\cap Red(T(As, \sim P_2\times\{0\}))/f_2 \bigcup_{g\in G\cup G_{\mathrm{I}}}\{Red(D_{G_{\mathrm{I}}(\sim P_1)}T(As, \sim P_1\times\{0\}))/f_1\cap Red(D_gT(As,\sim P_2\times\{0\}))/f_2\cup Red(D_{G_{\mathrm{I}}(\sim P_2)}T(As, \sim P_2\times\{0\}))/f_2\cap Red(D_gT(As, \sim P_1\times\{0\}))/f_1\}\neq\phi$ and $(\exists_{i0})(\forall_j)(\exists_i)\{(j\neq j_0)\cap a_j^i\in Red(T(As, \sim R_j^i\times\{0\}))\eta_j^i\}$ (η_j^i is some substitution), then let $\delta(G_0, a_{j0}^i)\ni R_{j0}^i\times\{0\}$ for each i.

(ii) If $(\forall_j)(\exists_i)\{a_j^i\in Red(T(As, \sim R_j^i\times\{0\}))\eta_j^i\}$ (η_j^i is some substitution) and $Red(T(As, \sim P_{k'}\times\{0\}))-\{x, xf_{k'}\}\xi_{k'}\neq\phi$ ($k'\neq k$ and $\xi_{k'}$ is some substitution), then let $\delta(\sim P_{k'}\times\{0\}, f_{k'}^{-1})\ni P_k\times\{1\}$ and $\delta(P_k\times\{1\}, f_k)\ni P_k\times\{0\}$ for $f_{k'}\neq\varepsilon$ or let $\delta(\sim P_{k'}\times\{0\}, f_k)\ni P_k\times\{0\}$ for $f_{k'}=\varepsilon$.

(b) In case that $S_m=P(fx)\cup\{\bigcup_j\bigcup_i R_j^i(a_j^i)\}$ or $\bigcup_j\{\bigcup_i R_j^i(a_j^i)\}$ for P and R_j^i in $Pr\times T_1$, f in $Fn\cup\{\varepsilon\}$, a_j^i in Ct and x in V.

(i) If $(\forall_i)(\exists_j)\{a_j^i\in Red(T(As, \sim R_j^i\times\{0\}))\eta_j^i\}$ (η_j^i is some substitution), the let $\delta(G_0, x)\ni G_{\mathrm{I}}(P\times\{0\})$ and $\delta(G_{\mathrm{I}}(P\times\{0\}), f)\ni P\times\{0\}$ for $f\neq\varepsilon$, or let $\delta(G_0, x)\ni G_{\mathrm{I}}(P\times\{0\})=P\times\{0\}$ for $f=\varepsilon$.

(ii) If $Red(T(As, \sim P\times\{0\}))\xi=xf$ for some substitution ξ, or $\bigcup_{g\in G\cup G_{\mathrm{I}}} Red(D_gT(As, \sim P\times\{0\}))\cap f\neq\phi$ and $(\exists_{j0})(\forall_i)(\exists_i)(j\neq j_0)\{Red(T(As, R_j^i\times\{0\}))\eta_j^i\ni a_j^i\}$ (η_j^i is some substitution), then let $\delta(G_0, a_j^i)\ni R_{j0}^i\times\{0\}$ for each i.

Repeap the step (2) until any new transition cannot be defined.

The following properties hold for a finite automaton As constructed by Algorithm 2.

Theorem 3 Let As be a finite automaton constructed by Algorithm 2 for a set of clauses in ACK_2. For any γ in $Red(T(As, P))$ such that P is in $Pr\times T_1\times\{0\}$, $P\circ\gamma^{Rev}$ can be generated from S, where '*Rev*' denotes the reverse string. $P\circ\gamma^{Rev}$ denotes a literal consiting of a predicate with or without negation sign corresponding to the part Pr of P, and a term composed of the part T_1 of P and γ^{Rev}.

Proof The theorem evidently holds for γ in $Red(T(As, P))$ with P in $Pr\times T_1\times\{0\}$

defined in (1) of Algorithm 2. For γ in $Red(T(As, P))$ with P in $Pr\times T_1\times\{0\}$ defined in (2) (a) (i) or (2) (b) of Algorithm 2, a unit clause whose term is the same as that in its parent clause corresponds to $P\circ\gamma^{Rev}$. If some γ_0 in $Red(T(As, P))$ and $\delta(P, \beta)\ni Q$ is defined for P and Q in $Pr\times T_1\times\{0\}$, then there is a resolvent $\sim P(fx)\cup Q(hx)$ for a variable x and function symbols f and h such that $h\circ f^{-1}=\beta$. Thus $Q\circ Red(\beta\gamma_1)^{Rev})$ can be generated by unit resolution for any γ_1 in $Red(T(As, P))$. Therefore we can claim that the theorem holds for γ in $Red(T(As, P))$ including the transitions defined by the procedure (2) (a) (ii). Q. E. D.

Theorem 4 Let As be a finite automaton constructed by Algorithm 2 for a set S of clauses in ACK_2. For any unit clause in S or generated from S, there is $P=[Pr\times T_1](L)$ in $Pr\times T_1\times\{0\}$ and γ in Fn^* such that $P\circ\gamma^{Rev}=L$.

Proof If there is a clause C in S such that $L\subset C$, then L can be represented in such a way as above, because the procedures (1), (2) (a) (i) and (2) (b) are defined in Algorithm 2. If there is a unit clause L_1 such that $T_2(L)$ can be obtained from $T_2(L_1)$ and L_1 can be represented in such a way as above, then L can be represented by means of the mapping defined in (1) (a) (ii) of Algorithm 2. Thus the theorem holds. Q. E. D.

There is a procedure in Algorithm 2 to decide the emptiness of (intersections of) languages operated by '*Red*'. To avoid using '*Red*', we provide another construction of an automaton in which equivalent mappings are defined one after another for the mappings by $Fn^{-1}\cup\{\varepsilon\}$ in As.

Algorithm 3 (A construction of a finite automaton Bs from a set S of clauses in ACK_2)

(Notation) Let $Bs=(G\cup G_0\cup G_I,\ \Gamma s,\ \delta s,\ G_0,\ G_f)$ be a finite automaton, where G_0 denotes an initial state, G and $G_I\subset Pr\times T_1$ being sets of states, G_f denotes a final state, $\Gamma s=Fn\cup V\cup Ct$, and $\delta s\subset(G\cup G_0\cup G_I)\times\Gamma s\rightarrow 2^{(G\cup G_0)}$(a mapping). $T(Bs, G_f)$ denotes the language accepted by Bs with $\{G_f\}$ as a set of final states. $D_gT(Bs, G_f)$ for $g\in G\cup G_I$ denotes the language $\{u\backslash v\,|\,u\in T(Bs, g),\ v\in T(Bs, G_f)$ and '\' is a left derivatives$\}$.

(Construction of an automaton)

(1) For each literal L represented by Definition 4 such that $T_2(L)=t_1\ t_2$ or t_2, construct the following transitions.

(a) In case that t_1 is in Ct: Let $\delta s(G_0,\ t_1)\ni[Pr\times T_1](L)$.

(b) In case that t_1 or t_2 is in V. (i) If $T_2(L)=t_1t_2$, then let $\delta s(G_0,\ t_2)\ni G_I([Pr\times T_1](L))$ and $\delta s(G_I([Pr\times T_1](L)),\ t_1)\ni[Pr\times T_1](L)$ for $G_I([Pr\times T_1](L))\in G_I$. (ii) If $T_2(L)=t_1$, then let $\delta s(G_0,\ t_1)\ni[Pr\times T_1](L)=G_I([Pr\times T_1](L))$.

(2) For each clause S_m represented by Definition 4, construct the following transitions.

(a) In case that $S_m=\bigcup_{k=1}^{2}P_k(f_kx)\cup\{\bigcup_j\bigcup_i R_j^i(a_j^i)\}$ for P_k and R_j^i in $Pr\times T_1$, f_k in $Fn\cup\{\varepsilon\}$,

a_j^i and x in V: (i) For each k excecute the next steps. $(i-1)$ If $xf_{k'}\in T(Bs, \sim P_{k'})$ $\eta_{k'}(k'\neq k$ and η_k is some substituion) and $(\forall_j)(\exists_i)\{a_j^i\in T(Bs, \sim R_j^i)\eta_j^i\}$ (η_j^i is some substituion), then let $\delta s(G_0, x)\ni G_{\mathrm{I}}(P_k)$ and $\delta s(G_{\mathrm{I}}(P_k), f_k)\ni P_k$ for $f_k\neq\varepsilon$, or let $\delta s(G_0, x)\ni G_{\mathrm{I}}(P_k)=P_k$ for $f_k=\varepsilon$. $(i-2)$ If $T(Bs, \sim P_1)/f_1\cap T(Bs, \sim P_2)/f_2 \bigcup_{g\in G\cup G_{\mathrm{I}}} \{D_{G_{\mathrm{I}}(\sim P_1)} T(Bs, \sim P_1)/f_1\cap D_g T(Bs, \sim P_2)/f_2\cup D_g T(Bs, \sim P_1)/f_1\cap D_{G_{\mathrm{I}}(\sim P_2)}T(Bs, \sim P_2)/f_2\}\neq\phi$ and $(\exists_{j0})(\forall_j)(\exists_i)\{(j\neq j_0)\cap a_j^i\in T(Bs, \sim R_j^i)\eta_j^i\}$ (η_j^i is some substitution), then let $\delta s(G_0, a_j^i)\ni R_{j0}^i$.

(ii) $(\forall_j)(\exists_i)\{a_j^i\in T(Bs, \sim R_i^j)\eta_j^i\}$ (η_j^i is some substitution) and $T(Bs, \sim P_{k'})-\{x, xf_{k'}\}$ $\eta_{k'}\neq\phi$ $(k'\neq k$ and $\eta_{k'}$ is some substitution), then construct the following transitions. (ii$-$1) Let $\delta s\ g, f_k)\ni P_k$ for $f_k\neq\varepsilon$ and $f_{k'}\neq\varepsilon$, where g is in $G\cup G_0\cup G_{\mathrm{I}}$ such that $\delta s(g, f_{k'})\ni\sim P_{k'}$. (ii$-$2) Let $\delta s(\sim P_{k'}, f_k)\ni P_k$ for $f_k\neq\varepsilon$ and $f_{k'}=\varepsilon$. (ii$-$3) Let $\delta s(g_1, f)$ $\ni P_k$ for $f_k=\varepsilon$ and $f_{k'}\neq\varepsilon$, where g_1 is in $G\cup G_0\cup G_1$ such that $\delta s(g_1, f)\ni g$ and $\delta s(g, f_{k'})\ni\sim P_{k'}$ for f in Fn. (ii$-$4) Let $\delta s(g, f)\ni P_k$ for $f_{k'}=\varepsilon$ and $f_k=\varepsilon$, where g is in $G\cup G_0\cup G_{\mathrm{I}}$ such that $\delta s(g, f)\ni\sim P_{k'}$ for f in Fn.

(b) In case that $S_m=P(fx)\bigcup_j\{\bigcup_i R_j^i(a_j^i)\}$ or $\bigcup_j\{\bigcup_j R_j^i(a_j^i)\}$ for P and R_j^i in $Pr\times T_1$, f in $Fn\cup\{\varepsilon\}$, a_j^i in Ct, and x in V: Construct the same transition as in (2) (b) of Algorithm 2, where the indication $\{0\}$ is dropped out.

Repeat the step (2) until any nuw transition cannot be defined.

The properties similar to Theorem 3 and Theorem 4 hold for the finite automaton Bs.

2.3 Algorithm to Decide Unit Resolution Refutability

We can obtain the following algorithm, by means of constructions of finite automata in the previous section.

Algorithm 4 (A decision algorithm of unit resolution refutability for ACK_2)

(1) Obtain a representation of a given set of clauses in ACK_2 by Definition 4, and denote it by S.

(2) Obtain As or Bs by Algorithm 2 or Algorithm 3.

(3) Decide whether there is a complementary pair of unit clauses by means of the following decision of emptiness of intersections of regular languages

Decide whether $Red(T(As, g))\cap Red(T(As, g'))\bigcup_{g_1\in G\cup G_{\mathrm{I}}, g_2\in G_{\mathrm{I}}}\{Red(D_{g_1}T(As, g))\cap Red(D_{g_2}T(As, g'))\cup Red(D_{g_2}T(As, g))\cap Red(D_{g_1}(T(As, g'))\}=\phi$ for g and g' such that g and g' are in $Pr\times T_1\times\{0\}$, and $[Pr\times T_1](g)=\sim[P\times T_1](g')$. *Or* decide whether $T(Bs, g)$ $\cap T(Bs, g')\bigcup_{g_1\in G\cup G_{\mathrm{I}}, g_2\in G_{\mathrm{I}}}\{D_{g_1}T(Bs, g)\cap D_{g_2}T(Bs, g')\cup D_{g_2}T(Bs, g(\cap D_{g_1}T(Bs, g')\}=\phi$ for g and g' such that g and g' are in $Pr\times T_1$, and $[Pr\times T_1](g)=\sim[Pr\times T_1]g')$. If some intersection is not empty, then there is a unit resolution refutation.

Otherwise there is no unit resolution refutation.

Example 2 Let $\{P(a), \sim P(x)\cup Q(f(x)), \sim Q(y)\cup R(g(y)), \sim R(g(z))\cup Q(f(z)), \sim Q(a), Q(u)\cup T(f(u)), \sim T(f(s))\cup U(f(s)), \sim U(w)\cup V(f(w)), \sim V(t)\cup\sim Q(f(t))\}$ be a set of clauses in ACK_2, where P, Q, R, T, U and V denote predicate symbols, f and g denote function symbols, x, y, z, s, t, u, and w denote variables, and a denotes a constant. It can be represented by Definition 4 as $\{\{P(\varepsilon)(a)\}$, $\sim P(\varepsilon)(x) \cup Q(\varepsilon)(fx)$, $\sim Q(\varepsilon)(y) \cup R(\varepsilon)(gy)$, $\sim R(\varepsilon)(gz) \cup Q(\varepsilon)(fz)$, $\{Q(\varepsilon)(a)\}$, $Q(\varepsilon)(u) \cup T(\varepsilon)(fu)$, $\sim T(\varepsilon)$

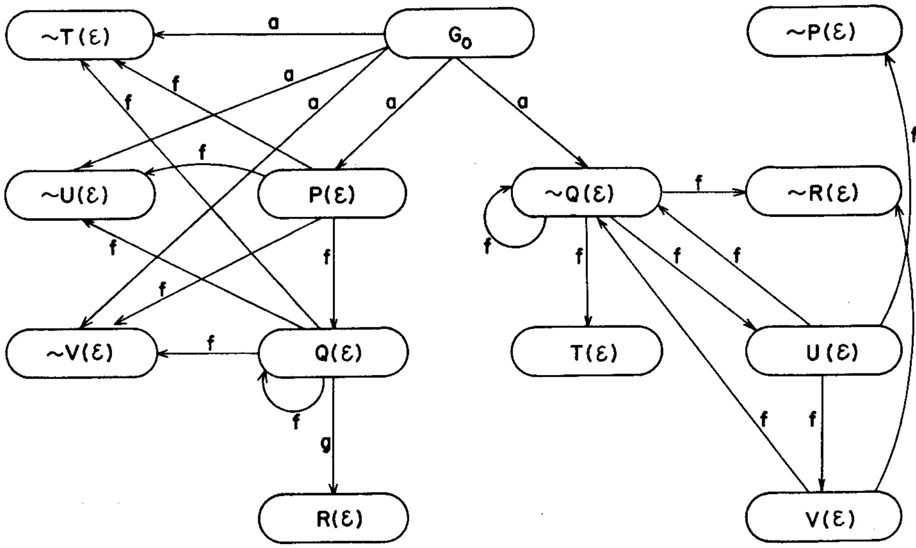

Fig. 1. A finite automaton As for Example 2.

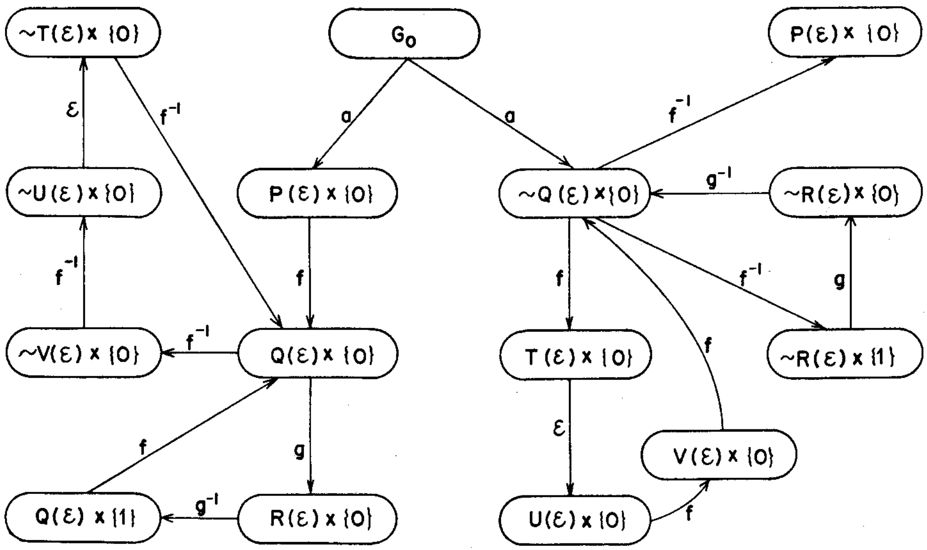

Fig. 2. A finite automaton Bs for Example 2.

$(fs) \cup U(\varepsilon)(fs)$, $\sim U(\varepsilon)(w) \cup V(\varepsilon)(fw)$, $\sim V(\varepsilon)(t) \cup \sim Q(\varepsilon)(ft)\}$, which we denote by S ('$\circ$' is omitted in S).

An automaton As by Algorithm 2 can be constructed as shown in Fig. 1. An automaton Bs by Algorithm 3 can be constructed as shown in Fig. 2.

We can conclude that $Red(T(As, Q(\varepsilon) \times \{0\})) = T(Bs, Q(\varepsilon)) = af^*$ and $Red(T(As, \sim Q(\varepsilon) \times \{0\})) = T(Bs, \sim Q(\varepsilon)) = af^*$. Thus $Q(f^*a)$ and $\sim Q(f^*a)$ can be generated by unit resolution. Therefore there is a unit resolution refutation.

3. Computational Complexity of Deciding Unit Resolution Refutablility for ACK$_2$

In this section we discuss computational complexity (time complexity and space complexity) of deciding a unit resolution refutability for a set of clauses in ACK_2. Computational complexity in this section is defined by means of multitape Turing machines.

Lemma 5 Algorithm 3 is of $O(n^9)$ time complexity and of $O(n^3)$ space compleity for the input length n.

Proof The size of the constructed algorithm is at most $O(n^3)$, since the size of states is at most $O(n)$ and the number of transition symbols is at most $O(n)$. Evidently Step (1) can be excuted in $O(n^2)$ time and in $O(n)$ space. Step (2)(a)(i) is a procedure to represent a unit clause subsuming its parent input clause. Step (i-1) can be excecuted in $O(n^6)$ time and in $O(n^3)$ space for a mapping to be detected, since in this step it is decided whether there is a transition from an initial state to a specified state at most $O(n^3)$ times. Step (i-2) can be excecuted in $O(n^6)$ time and in $O(n^3)$ space for a mapping to be detected, since in this step it is decided whether there is a common transition between sequences or subsequences from an initial state to two specified states at most $O(n^3)$ times and it is decided whether there is a transition from an initial state to a specified state at most $O(n^3)$ times.

Step (2)(a)(ii) can be executed in $O(n^6)$ time and in $O(n^3)$ space for a mapping to be detected, since in this step it is decided whether there is a transition from an initial state to a specified state at most $O(n^3)$ times. Step (2)(b) is similar to Step (2)(a)(i-1). A transformation from a mapping with inverse function symbols or ε to a mapping without inverse function symbols and ε can be executed in $O(n^5)$ time and in $O(n^3)$ space for each mapping, since transitions of length at most 2 are examined in constucting the automaton.

$O(n^3)$ mappings are constructed by Step (2).

The above considerations lead us to the conclusion. *Q. E.* D.

Theorem 6 It can be decided in $O(n^9)$ time and in $O(n^3)$ space whether there is a unit resolution refutaion from a set of clauses of length n in ACK_2.

Proof We consider computational complexity of decidaility on the basis of Algorithm 4.

Step (1) can be exceuted in $O(n^2)$ time and in $O(n)$ space.

Step (2) to obtain Bs can be executed in $O(n^9)$ time and in $O(n)$ space (Lemma 5). The length of Bs is $O(n^3)$.

Step (3) can be executed in $O(n^6)$ time and in $O(n^3)$ space, since it can be decided in $O(n^6)$ time and in $O(n^3)$ space whether the intersection of two regular languages accepted by two finite automata of length $O(n^3)$ is empty. Thus the theorem holds.

Q. E. D.

4. Concuding Remarks

In this paper we provided an algorithm to decide a unit resolution refutability for a subclass ACK_2 of the Ackermann class by constructing finite automata.

We showed that the algorithm is of $O(n^9)$ time complexity and of $O(n^3)$ space complexity for an input set of length n on multitape Turing machines.

The algorithm works in deterministic polynomial time and therefore it is applicable to the preprocessing of the unsatisfiability problem for a subclass ACK_2 of the Ackermann class.

Utilizing [8], we can conclude that the decidability problem of a unit resolution refutability for the Ackermann class is P-space hard. That is to say all the problems in polynomial space can be transformed to the decidability problem of a unit resolution refutability for the Ackermann class.

We discussed the decidability problem of a unit resolution refutability for ACK_2. For what subclasses of the Ackermann class, other than ACK_2, the decidability problem of a unit resolution refutability can be solved in determinitic polynomial time, is left for a future study.

It is not of advantage to apply unit resolution to the Gödel class, since a unit resolution refutability for this class is not decidable.

References

1) Aho, A. V., Hopcroft, J. E. and Ullman, J. D., "The Design and Analysis of Computer Algorithms", Addison-Wesley Publishing Company, (1974).
2) Aspvall B. et al., "A linear time algorithm for testing the truth of certain quantified boolean formulas", Information Processing Letters, 8, 3, (1979), p. 121-123.
3) Brzozowski, J. A., „Regular-like expressions for some irregular languages", IEEE Conference Record of the 9th Annual Symposium on Switching and Automata Theory, (1968), p. 278-286.
4) Chang, C. L. and Lee, R. C., "Symbolic Logic and Mechanical Theorem Proving", Academic Press, (1974).
5) Hopcroft, J. E. and Ullman, J. D., "Formal Languages and Their Relation to Automata",

Addison-Wesley Publishing Company, (1969).

6) Jones, N. D. and Laaser, W. T., "Complete problem for deterministic polynomial time", Proc. of the 6th Annual ACM Symposium on Theory of Computing, (1974), p. 40-46.

7) Joyner, W. H., Jr., "Resolution strategies as decision procedures", J. of ACM, 23, 3, (1976), p. 398-417.

8) Lewis, H. R., "Complexity of solvable cases of the decision problems for the predicate calcululus", IEEE Conference Record of the 19th Annual Symposium on Foundations of Computer Science, (1978), p. 35-47.