Inventing ET Rules to Improve an MI Solver on KR-logic

Tadayuki Yoshida[1], Ekawit Nantajeewarawat[2], Masaharu Munetomo[3] and Kiyoshi Akama[3]
[1]*Tokyo Software Development Laboratory, International Business Machines Corporation, Tokyo, Japan*
[2]*Computer Science Program, Sirindhorn International Institute of Technology Thammasat University, Pathumthani, Thailand*
[3]*Information Initiative Center, Hokkaido University, Sapporo, Japan*

Keywords: Logical Problem Solving Framework, Equivalent Transformation, Knowledge Representation, Computation Rule, Query-answering Problem.

Abstract: We understand that many logical problems cannot be solved by using logic programs. Logic programs have the limited capability of representation. We try to overcome this limitation by adopting KR-logic, an extension to first-order logic. The extension includes function variables. In this paper, we take a problem which is well-described with function variables. We rely on Logical Problem Solving Framework (LPSF) to formalize our problem as a Model-intersection problem. Then we develop a solver for MI problems by adding five new transformation rules concerning function variables. Correctness of each rule is proved.i.e., each rule is an equivalent tranformation (ET) rule. Since each rule is correct, all ET rules can be used together without modification and combinational cost. Thus, the invented rules can be safely reused in other LPSF-based solvers.

1 INTRODUCTION

Logic programs such as Prolog take an approach to rely on SLDNF-resolution (Chang and Lee, 1973; Lloyd, 1987). There are many logical problems which cannot be solved by using conventional logic approach. The Agatha puzzle is one of such problems. We use LPSF (Akama et al., 2019) to formalize the Agatha puzzle as a Model-intersection (Akama and Nantajeewarawat, 2016) problem.

Usual clauses don't have enough expressive power to represent and compute the Agatha puzzle. We take an extended clause in $\mathrm{ECLS_F}$ as a representation space. $\mathrm{ECLS_F}$ is a class of clauses with function variables on KR-logic(Akama et al., 2019). The objective of this paper is to construct a solver for the Agatha puzzle by development of equivalent transformation (ET) rules.

We already have developed several ET rules (Akama and Nantajeewarawat, 2011),(Akama and Nantajeewarawat, 2015), (Akama et al., 2018a), (Akama et al., 2018b) as well as unfolding (Akama and Nantajeewarawat, 2013) for computation of extended clauses. However, it turned out that the solver using these existing rules cannot give an answer to the Agatha puzzle. This difficulty is due to the lack of ET rules to handle function variables. To represent the Agatha puzzle correctly, function variables are essential. Representation spaces without function variables, such as conventional clauses, don't have enough expressive power of existential quantification.

This paper proposes a set of ET rules which transform extended clauses containing function variables. The LPSF theory ensures the correctness of answers to the Agatha puzzle strictly from a combination of ET rules. The heuristic approach cannot guarantee the correctness of the method, while LPSF can do for ET rules. We think ET rules discovered in solving a certain problem are usable for a general purpose. Thus, once we invent an ET rule, such rule is reused independently together with a variety of existing ET rules.

We take the squeeze method to discover ET rules. We also study to prove the correctness of rules discovered. Then we state that a new ET rule is truly invented. In this paper, we successfully invent five new ET rules through iterations of the squeeze method.

The answer to the Agatha puzzle is obtained by expanding a computation sequence through invented rule application. This is our first achievement to construct an MI solver for the Agatha puzzle based on the extended space on $\mathrm{ECLS_F}$. We also demonstrate the fact that new ET rules contribute to an improvement

Yoshida, T., Nantajeewarawat, E., Munetomo, M. and Akama, K.
Inventing ET Rules to Improve an MI Solver on KR-logic.
DOI: 10.5220/0008165702740281
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 274-281
ISBN: 978-989-758-382-7

of the solver. Due to the limited representation capability of conventional clauses with first-order logic, we cannot solve the Agatha puzzle by using any possible computation methods. The LPSF theory provides a firm foundation for stepwise improvement of logical problem solving.

The rest of this paper is organized as follows: Section 2 discusses about how to design an MI solver based on LPSF for the Agatha puzzle. Section 3 explains the rule invention process for a solution of the Agatha puzzle, by iteratively squeezing applicable rules from an interim transformation status as a result of application of exiting known rules. Section 4 proves correctness of the invented five transformation rules in this paper. Section 5 shows how a newly constructed MI solver works and what other LPSF parameters can be taken into considerations. Section 6 gives a discussion about the comparison with existing logical structures, one on the usual clausal logic and the other is the extended clause space with constraints. Section 7 concludes the paper.

2 DESIGN OF MI SOLVER FOR QA PROBLEMS

We discuss a design consideration of MI solver for the famous Agatha Puzzle.

2.1 Agatha Puzzle

Consider the "Dreadsbury Mansion Mystery" problem, which was given by Len Schubelt and can be described as follows: 1) Someone who lives in Dreadsbury Mansion killed Aunt Agatha. 2) Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only people who live therein. 3) A killer always hates his victim, and 4) is never richer than his victim. 5) Charles hates no one that Aunt Agatha hates. 6) Agatha hates everyone except the butler. 7) The butler hates everyone not richer than Aunt Agatha. 8) The butler hates everyone Agatha hates. 9) No one hates everyone. The Agatha QA problem is to find all persons who killed Agatha. The Agatha proof problem is to show that Agatha killed herself.

2.2 Formalization of the Puzzle

We take $\mathrm{ECLS_F}$ as a logical structure. We already know many ET rules on the space of $\mathrm{ECLS_F}$, however, they are not enough for complete solution of the Agatha puzzles.

Assume that *eq* and *neq* are predefined binary constraint predicates. For any ground usual terms t_1 and t_2, (i) $eq(t_1,t_2)$ is true iff $t_1 = t_2$, and (ii) $neq(t_1,t_2)$ is true iff $t_1 \neq t_2$. The background knowledge of this mystery is formalized as the conjunction of the first-order formulas, where (i) the constants A, B, C, and D denote "Agatha," "the butler," "Charles," and "Dreadsbury Mansion," respectively, and (ii) for any terms t_1 and t_2, $live(t_1,t_2)$, $kill(t_1,t_2)$, $hate(t_1,t_2)$, and $richer(t_1,t_2)$ are intended to mean "t_1 lives in t_2," "t_1 killed t_2," "t_1 hates t_2," "t_1 is richer than t_2," respectively.

Let K be the conjunction of the first-order formulas F_1 to F_9 and let $q = kill(x,A)$. The Agatha puzzle is formalized as a QA problem $\langle K,q\rangle$. The answer to this QA problem, denoted by $answer(K,q)$, is the set of all ground instances of q that follows logically from K.

$F_1 = \exists x : (live(x,D) \wedge kill(x,A))$
$F_2 = \forall x : (live(x,D) \leftrightarrow (eq(x,A) \vee eq(x,B) \vee eq(x,C))$
$F_3 = \forall x : \forall y : (kill(x,y) \rightarrow hate(x,y))$
$F_4 = \forall x : \forall y : (kill(x,y) \rightarrow \neg richer(x,y))$
$F_5 = \neg\exists x : (hate(C,x) \wedge hate(A,x) \wedge live(x,D))$
$F_6 = \forall x : ((neq(x,B) \wedge live(x,D)) \rightarrow hate(A,x))$
$F_7 = \forall x : (\neg richer(x,A) \wedge live(x,D)) \rightarrow hate(B,x))$
$F_8 = \forall x : ((hate(A,x) \wedge live(x,D)) \rightarrow hate(B,x))$
$F_9 = \neg\exists x : (live(x,D) \wedge \forall y : (live(y,D) \rightarrow hate(x,y)))$

K is converted into the set Cs consisting of the fourteen extended clauses C_1 to C_{14} by applying meaning-preserving Skolemization (Akama and Nantajeewarawat, 2011) as follows: $\mathrm{MPS}(F_1) = C_1$, C_2, $\mathrm{MPS}(F_2) = C_3$, C_4, C_5, C_6, $\mathrm{MPS}(F_3) = C_7$, $\mathrm{MPS}(F_4) = C_8$, $\mathrm{MPS}(F_5) = C_9$, $\mathrm{MPS}(F_6) = C_{10}$, $\mathrm{MPS}(F_7) = C_{11}$, $\mathrm{MPS}(F_8) = C_{12}$, and $\mathrm{MPS}(F_9) = C_{13}$, C_{14}.

$C_1 : live(x,D) \leftarrow func(f_0,x)$
$C_2 : kill(x,A) \leftarrow func(f_0,x)$
$C_3 : \leftarrow live(x,D), neq(x,A), neq(x,B), neq(x,C)$
$C_4 : live(A,D) \leftarrow$
$C_5 : live(B,D) \leftarrow$
$C_6 : live(C,D) \leftarrow$
$C_7 : hate(x,y) \leftarrow kill(x,y)$
$C_8 : \leftarrow kill(x,y), richer(x,y)$
$C_9 : \leftarrow hate(A,x), hate(C,x), live(x,D)$
$C_{10} : hate(A,x) \leftarrow neq(x,B), live(x,D)$
$C_{11} : richer(x,A), hate(B,x) \leftarrow live(x,D)$
$C_{12} : hate(B,x) \leftarrow hate(A,x), live(x,D)$
$C_{13} : \leftarrow hate(x,y), func(f_1,x,y), live(x,D)$
$C_{14} : live(y,D) \leftarrow live(x,D), func(f_1,x,y)$

Let state S_0 be $\{C_1,...,C_{14}\}$.

2.3 Answer Mapping

The QA problem $\langle K,q\rangle$ is then reformulated as a model-intersection (MI) problem $\langle Cs,\varphi\rangle$, where φ is a mapping from the power set of $\mathcal{G}_u$ to the power set of $\{A,B,C\}$, defined by $\varphi(G)=\{t\,|\,kill(t,A)\in G\}$ for any $G\subseteq\mathcal{G}_u$. In other words, we have

$answer(K,q)=\varphi(\bigcap Models(Cs))$.

Our plan for solving the Agatha puzzle is to simplify $\varphi(\bigcap Models(Cs))$ mainly by transforming Cs preserving $Models(Cs)$ or $\bigcap Models(Cs)$.

3 RULE INVENTION PROCESS

We introduce a process to invent transformation rules through the squeeze method.

3.1 The Squeeze Method

Many successful research work have proposed general ET rules applicable to a wide range of problem domain. Based on these outcomes, we first apply such existing ET rules to the QA problem introduced in 2.1, and then investigate the interim set of clauses after an application process of ET rules terminates. The iteration of the squeeze method goes as follows: 1) add one or more rules with a proper priority control for them, 2) run an application process of combined ET rule set, 3) verify the interim result if it reaches to the domain of the defined answer mapping. Otherwise, go back to 1).

3.2 Applying Existing ET Rules

By applying ET rules discussed in (Akama and Nantajeewarawat, 2014) and (Akama et al., 2018b), to the original clauses in S_0, these clauses are transformed into simple clauses where atoms like *live* and *hate* are resolved. Then we get the clauses in state S_1 as the first interim result with which our iteration process starts. The process of transformation to these clauses is discussed 2.3 in (Akama and Nantajeewarawat, 2018)

$S_1=\{C_{44},...,C_{55}\}$
$C_{44}:\leftarrow neq(x,B),func(f_1,B,x)$
$C_{45}:\leftarrow func(f_1,B,A)$
$C_{46}:\leftarrow func(f_1,B,C)$
$C_{47}:\leftarrow func(f_0,C)$
$C_{48}:\ kill(x,A)\leftarrow func(f_0,x)$
$C_{49}:\leftarrow func(f_1,B,x),func(f_0,x)$
$C_{50}:\leftarrow func(f_1,x,A),func(f_0,x)$
$C_{51}:\leftarrow neq(x,A),neq(x,B),neq(x,C),func(f_0,x)$
$C_{52}:\leftarrow neq(x,A),neq(x,B),neq(x,C),func(f_1,C,x)$
$C_{53}:\leftarrow func(f_1,A,A)$
$C_{54}:\leftarrow func(f_1,A,C)$
$C_{55}:\leftarrow neq(x,B),func(f_1,A,x)$

3.3 Determining Development Strategy

There are many func-atoms in the body of clauses in S_1. In order to make further transformations, such func-atoms need to get handled correctly. It is observed that two types of func-atoms occurrences in S_1: a) A func-atom alone in a body and no head atoms like C_{45},C_{46},C_{47},C_{53},C_{54}, and b) Combination of a single func-atom variable and a sequence of neq atoms, and no head atoms like C_{44},C_{51},C_{52},C_{55}. In addition to these types of func-atoms, we also need to remove a clause containing a func-atom alone in a body in order to match to the domain of the answer-mapping function defined in 2.3. The answer-mapping function can be applicable when C_{48} is simplified to have no body atoms. So a rule for safe func-atom clause removal is being developed. In the rest of this paper, we invent these types of ET rules on the LPSF and observe if an answer to the problem can be obtained by using such rules from S_0 again.

3.4 Squeeze Method Iteration

3.4.1 Side Change of neq-atoms in Body

The rule of side change of *neq*, denoted as `(chSide neq)`, is to move *neq* atoms from body part into head part with altering predicate to *eq*. This rule is applied and C_{61}, C_{62}, C_{63}, and C_{64} are obtained from C_{44}, C_{51}, C_{52} and C_{55} respectively.

these are transformed into:

$C_{61}:\ eq(x,B)\leftarrow func(f_1,B,x)$
$C_{62}:\ eq(x,A),eq(x,B),eq(x,C)\leftarrow func(f_0,x)$
$C_{63}:\ eq(x,A),eq(x,B),eq(x,C)\leftarrow func(f_1,C,x)$
$C_{64}:\ eq(x,B)\leftarrow func(f_1,A,x)$

3.4.2 False func-atom Elimination

The rule of false func-atom elimination, denoted as `elimNegfvr`, is to eliminate a clause containing a false func-atom. According C_{61}, $f_1(B)=B$ is determined. So any other declarations for $f_1(B)$ must be false. Due to this, C_{45} and C_{46} are deleted. Also, $f_1(A)=B$ is determined by C_{64} so any $f_1(A)$ atoms else are false. C_{53} and C_{54} can be eliminated.

3.4.3 Candiate eq-atom Elimination

The rule of candidate eq-atom elimination, denoted as `elimPosfvr`, is to remove *eq* atoms in head that match with negative declaration of the same func-atom. According to C_{47}, $f_0() \neq C$. This means $eq(x,C)$ in the head of C_{62} can be removed and obtain C_{65}.

$C_{65}: eq(x,A), eq(x,B) \leftarrow func(f_0,x)$

3.4.4 True func-atom Elimination

The rule of true func-atom elimination, denoted as `applyPosfvr`, is to eliminate func-atom evaluated as *true* by referring to a clause of the same func-atom. It is applied to C_{49} with C_{61} and replace C_{49} with C_{66}

$C_{66}: \leftarrow func(f_0,B)$

3.4.5 Combinational Rule Application

Again, `elimNegfvr` and `elimPosfvr` several times

$C_{48}: kill(x,A) \leftarrow func(f_0,x)$
$C_{50}: \leftarrow func(f_1,x,A), func(f_0,x)$
$C_{61}: eq(x,B) \leftarrow func(f_1,B,x)$
$C_{63}: eq(x,A), eq(x,B), eq(x,C) \leftarrow func(f_1,C,x)$
$C_{67}: eq(x,A) \leftarrow func(f_0,x)$
$C_{68}: eq(x,B) \leftarrow func(f_1,A,x)$

`applyPosfvr` to C_{50} with C_{67} and replace C_{50} with C_{69}, and `applyPosfvr` to C_{48} with C_{67} and replace C_{48} with C_{70}

$C_{61}: eq(x,B) \leftarrow func(f_1,B,x)$
$C_{62}: eq(x,A) \leftarrow func(f_0,x)$
$C_{63}: eq(x,A), eq(x,B), eq(x,C) \leftarrow func(f_1,C,x)$
$C_{64}: eq(x,B) \leftarrow func(f_1,A,x)$
$C_{69}: \leftarrow func(f_1,A,A)$
$C_{70}: kill(A,A) \leftarrow$

`elimNegfvr` to C_{69}

$C_{61}: eq(x,B) \leftarrow func(f_1,B,x)$
$C_{62}: eq(x,A) \leftarrow func(f_0,x)$
$C_{63}: eq(x,A), eq(x,B), eq(x,C) \leftarrow func(f_1,C,x)$
$C_{64}: eq(x,B) \leftarrow func(f_1,A,x)$
$C_{70}: kill(A,A) \leftarrow$

3.4.6 Isolated func-atom Elimination

The rule of isolated func-atom elimination, denoted as `elimIsoFunc` is to remove a clause containing a func-atom which has no unifiable candidates in body atoms of other clauses. In C_{61}, $f_1(B) = B$ is determined and no unifiable pattern to $f_1(B)$ exist in C_{62},C_{63} and C_{64}. This situation makes the removal of C_{61}.

$C_{62}: eq(x,A) \leftarrow func(f_0,x)$
$C_{63}: eq(x,A), eq(x,B), eq(x,C) \leftarrow func(f_1,C,x)$
$C_{64}: eq(x,B) \leftarrow func(f_1,A,x)$
$C_{70}: kill(A,A) \leftarrow$

In the same way, C_{62}, C_{63} and C_{64} all can be removed.

We run through four iterations of squeezing and finally the clause is simplified enough to apply answer-mapping to get an answer to the Agatha puzzle. The process resulted in finding four new ET rules.

The next chapter, we provide a theoretical basis for each rule so that we can make strict definitions of newly invented rules.

4 CORRECTNESS

We give formal definitions for five new ET rules. Each subsection starts with a formal description of new rule, followed by its brief proof.

4.1 Correctness of Computation

The LPSF guarantees a correct computation with a bunch of transformation rules by giving a theoretical foundation to discuss the correctness of each individual rule. This yields a stable growth of an MI solver's capability through adding a set of newly invented transformation rules to existing rule set. We don't need to have any concern about the consistency regarding the correctness of whole computation process. It is always kept valid as long as it is based on the LPSF.

4.2 Side Change of neq-atoms in Body

Let $\langle Cs, \varphi \rangle$ be a MI problem on $\mathrm{ECLS_F}$. Assume that a clause C in Cs such that

$$C = (\leftarrow Neqs(n), func(f, s_1, ..., s_m, x)),$$
$$Neqs(n) = \{neq(x,t_1), ..., neq(x,t_n)\}$$

where x is a usual variable, $t_1,...,t_n$ and $s_1,...,s_m$ are ground terms, f is a function variable, $n > 1$, and $m >= 0$. There exists at least one $func$-atom in body of C and a set of neq-atoms which has the same occurrence of the variable x in the $func$-atom, where all domain terms in $func$-atom have been already grounded $(s_1,...s_m)$.

Under this condition, a new clause C' is constructed from C and all neq-atoms in body of C can be side-changed into head as eq-atoms. The rest of body atoms in C, represented as Bs remain unchanged.

$$C' = (Eqs(n) \leftarrow func(f, s_1, ..., s_m, x))$$
$$Eqs(n) = \{ eq(x,t_1), ..., eq(x,t_n) \}$$
$$Cs' = (Cs - \{ C \}) \cup \{ C' \}$$

The correctness of this rule is shown as follows: Let σ and θ be substitutions for function variables and usual variables, respectively. Assume that $C\sigma\theta$ is true. Since C has no head atom, $Neqs(n)\sigma\theta$ is false, while $func\sigma$ is true.
$Neqs(n)\sigma\theta = \text{false}$
$\Leftrightarrow (x\sigma\theta \neq t_1 \wedge ... \wedge x\sigma\theta \neq t_n) = \text{false}$
$\Leftrightarrow \neg(x\sigma\theta = t_1 \vee ... \vee x\sigma\theta = t_n) = \text{false}$
$\Leftrightarrow (x\sigma\theta = t_1 \vee ... \vee x\sigma\theta = t_n) = \text{true}$
$\Leftrightarrow Eqs(n)\sigma\theta = \text{true}$
If $Eqs(n)\sigma\theta$ is true, then $C'\sigma\theta$ is true and obviously $Models(Cs) = Models(Cs')$ is true.

4.3 Candidate eq-atom Elimination

Let $\langle Cs, \varphi \rangle$ be a MI problem on ECLS_F. Assume that a set of clauses C in Cs such that

$$C = (\leftarrow func(f, s_1, ..., s_m, g))$$

where $s_1,...,s_m,g$ are ground terms, f is a function variable, and $m >= 0$.

If there is a func-atom $func(f,s_1,...,s_m,v)$ in the right-hand side of a clause C_f rather than C in Cs, and the head atoms of C_f only contains a sequence of eq-atoms and each eq atom contains a pair of v and a ground term,

$$C_f = (Eqs(n) \leftarrow func(f, s_1, ..., s_m, v))$$
$$Eqs(n) = \{ eq(v,t_1), ..., eq(v,t_n) \}$$

then Cs' is obtained from Cs as follows:
head atom $eq(v,g)$ can be removed from C_f and C is removed from Cs.

$$C'_f = (Eqs(n,i) \leftarrow func(f, s_1, ..., s_m, v))$$
$$Eqs(n,i) = Eqs(n,i) - \{ eq(v,t_i) \}$$
$$Cs' = (Cs - \{ C, C_f \}) \cup \{ C'_f \}$$

where $t_i = g$.

The correctness of this rule is shown as follows: Basically, C_f constructs a set of possible range values $\{t_1,...t_m\}$ for f. Since the negative clause C declares an impossible candidate g of the range of f, if a range candidate contains t_i $(= g)$, then it can be removed. Also, C itself is no longer needed for defining the models of Cs, so C can be removed from the resulting clauses. Then $Models(Cs) = Models((Cs - \{C, C_f\}) \cup \{C'_f\})$ is true.

4.4 True func-atom Elimination

Let $\langle Cs, \varphi \rangle$ be a MI problem on ECLS_F. Assume that a set of clauses $C \in Cs$ such that

$$C = (eq(x,g) \leftarrow func(f, s_1, ..., s_m, x))$$

where x is a usual variable, $s_1,...,s_m,g$ are ground terms, f is a function variable, and $m >= 0$.

If there is a func-atom $func(f,s_1,...,s_m,v)$ in the right-hand side of a clause C_f rather than C in Cs, then C_f can be specialized by applying a substitution $\theta = \{v/g\}$, and the true func-atom $func(f,s_1,...,s_m,g)$ can be removed from the resulting clause $C_f\theta$.

$$C_f\theta = (Hs\theta \leftarrow Bs\theta, func(f, s_1, ..., s_m, v)\theta)$$
$$C'_f\theta = (Hs\theta \leftarrow Bs\theta)$$
$$Cs' = (Cs - \{ C_f \}) \cup \{ C'_f\theta \}$$

where Hs and Bs are a sequence of atoms.

The correctness of this rule is shown as follows: The above transformation is correct since $func(f,s_1,...,s_m,v)\theta\,(= func(f,s_1,...,s_m,g))$ is true. Then $Models(Cs) = Models((Cs - \{C_f\}) \cup \{C'_f\theta\})$ is true.

4.5 False func-atom Elimination

Let $\langle Cs, \varphi \rangle$ be a MI problem on ECLS_F. Assume that a set of clauses $C \in Cs$ such that

$$C = (Eqs(n) \leftarrow func(f, s_1, ..., s_m, x))$$
$$Eqs(n) = \{ eq(x,t_1), ..., eq(x,t_n) \}$$

where x is a usual variable, $t_1,...,t_n,s_1,...,s_m,g$ are ground terms, f is a function variable, $n >= 1$ and $m >= 0$.

If there is a ground func-atom $func(f,s_1,...,s_m,u)$ in the right-hand side of a clause C_f in Cs and $u \neq \{t_1,...,t_n\}$, then C_f can be removed from Cs.

$$C_f = (Hs \leftarrow Bs, func(f, s_1, ..., s_m, u))$$
$$Cs' = (Cs - \{ C_f \})$$

where Hs and Bs are a sequence of atoms.

The correctness of this rule is shown as follows: The above transformation is correct since $func(f,s_1,...,s_m,u)$ is false. Then $Models(Cs) = Models((Cs - \{C_f\}))$ is true.

4.6 Isolated func-atom Elimination

Let $\langle Cs, \varphi \rangle$ be a MI problem on $\mathrm{ECLS_F}$. Assume that a set of clauses C in Cs such that

$$Cs = Cs_r \cup \{C\},$$
$$C = (Hs \leftarrow Bs, func(f, s_1, ..., s_m, x))$$

where x is a usual variable, $s_1, ..., s_m$ are ground terms, f is a function variable, Hs and Bs are a sequence of built-in atoms, and $m >= 0$.

A func-atom is isolated with respect to Cs_r if there is no func-atom of the form $func(f', s'_1, ..., s'_{m'}, t)$ in the body of any clause in Cs_r such that $f' = f, m' = m$, $s'_1, ...s'_m, t$ are terms, and the lists of terms $[s_1, ...s_m]$ and $[s'_1, ...s'_m]$ are unifiable. If the following conditions are satisfied:

1. $func(f, s_1, ..., s_m, x)$ is isolated with respect to Cs_r, and
2. $(Hs \leftarrow Bs)$ contains no *func* atoms and satisfiable,

then C can be eliminated and $Cs' = Cs_r$.

The correctness of this rule is shown as follows: Assume that g is a ground term such that $(Hs \leftarrow Bs)\{x/g\}$ is satisfiable. Then there exists h such that $func(f, s_1, ..., s_m, g)\{f/h\}$ is true. Thus, $C\{x/g\}\{f/h\}$ is satisfiable and can be eliminated. The instantiation $\{f/h\}$ doesn't affect to Cs_r since $func(f, s_1, ..., s_m, x)$ is isolated with respect to Cs_r. Then the above transformation is correct since $Models(Cs') = Models(Cs_r) = Models(Cs)$.

5 EXPERIMENTS

We show a solution for the example Agatha puzzle by integrating five new rules with existing ones.

5.1 Solution Improvement

We have designed and implemented an improved MI solver with five invented ET rules. In this section we will observe the transformation process of the original clauses C_1 to C_{14} in 2.2.

Before the introduction of five new rules, we use seven existing rules and control of priority shown in Section 3.2. The computation process terminated at 125th steps of transformation where (chSide) is applied 1 time, (dup) 8 times, (erase) 7 times, (neq) 6 times, (specAtom) 1 time, (subsumed) 72 times, and (ud) 30 times. Figure 1 shows the transition of the number of clauses on each transformation step. The number of clauses goes down to 11 at 125th step. However, the answer is not obtained since the resulting clauses are not simple enough to apply the answer mapping function φ.

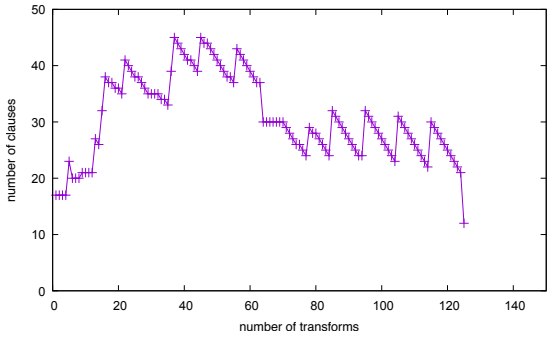

Figure 1: Transition of the number of clauses without new rules.

After integration of five invented rules, we use twelve rules in total. When we modify the rule priority as

```
(neq)>
(elimNegfvr)>(elimPosfvr)>
(applyPosfvr)>(elimIsoFunc)>
(subsumed)>(dup)>(erase)>
(specAtom)>(ud)>
(chSide)>(chSide neq)
```

then the process is stretched with 17 more steps of transformation where (applyPosfvr) is applied 3 times, (chSide neq) 1 time, (elimNegfvr) 7 times, (elimPosfvr) 2 times, and (elimIsoFunc) 4 times. Figure 2 shows the stretched transition of the number of clauses. The number of clauses goes down to 1 at 142th step and it is simple enough to apply the answer mapping φ, then the answer is obtained.

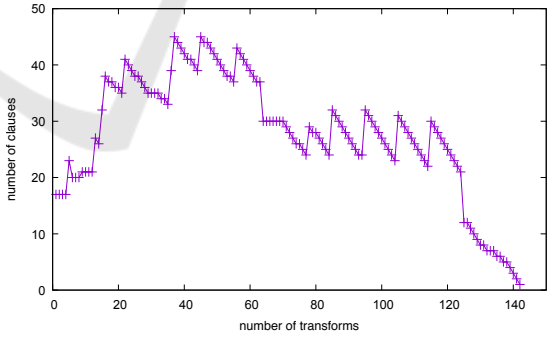

Figure 2: Stretched transition of the number of clauses with new rules.

5.2 Comprehensive Transformation Flow

All 142 steps cannot be listed in this paper. We give a summary of transformation for each stage where atoms are eliminated and simplicity increases.

1. By unfolding using the definition of *kill* (C_2), all body atoms with the predicate *kill* are removed.

By definite-clause removal, the definition of *kill* (C_2) is removed. By unfolding, three body atoms with patterns $hate(A,x)$ or $hate(C,x)$ are removed.

2. By side-change transformation for *richer*, the *richer*-atoms in some clauses are removed and *not_richer*-atoms are obtained in the other side of these clauses.
3. By unfolding, *not_richer*-atoms and *hate*-atoms in clause bodies are removed. The definitions of *not_richer* and *hate* are removed.
4. Unfolding with respect to *live*-atoms has been suspended since one of the definite clauses defining *live* introduces a new *live*-atom with a pure variable as its first argument. To remedy the situation, the atom $live(x,D)$ in the body of a clause is specialized into three clauses, which enable further application of unfolding.
5. By unfolding *live*-atoms and definite-clause removal, all *live*-atoms are removed and the clauses C_{44} to C_{55} are obtained.
6. By side-change transformation for *neq*, *neq*-atoms are removed from body part and *eq*-atoms are obtained in the respective head side in four clauses.
7. By eliminating clauses whose body contain false *func*-atoms, by eliminating *eq*-atoms for a certain *func*-atom which is not consistent with other clauses of the same *func*-atoms in question from a head part of a clause, and by eliminating *func*-atoms which hold true regarding the settled values declared in another clause of the same *func*-atom.
8. By eliminating isolated *func*-atom declarations which don't occur in any other clauses.
9. The ground atom is obtained consistently and the answer A meaning "Agatha" is obtained by applying the answer mapping φ.

6 DISCUSSION

In this section, we discuss the improvement of our approach compared with existing framework.

6.1 Failure in CLS_B (The Usual Clausal Logic)

The conventional formalization consists of the following two steps:

- From the Agatha problem described by sentences (Section 2.1) we obtain K in the first-order formulas.
- To K we apply CSK to have a set of clauses Cs'.

The conventional Skolemization (CSK) is a major hindrance to developing a general solution for proof and QA problems on ECLS_{FC} since it does not generally preserve satisfiability nor logical meanings by Theorem 1 in (Akama and Nantajeewarawat, 2016), and thus does not generally preserve the answers to proof and QA problems.

When we transform the Agatha background knowledge K into a set Cs' of usual clauses using the conventional Skolemization, we know that $Models(K) \neq \emptyset$ and $Models(Cs') = \emptyset$, hence neither satisfiability nor logical meanings is preserved in this case, and further computation starting from Cs' to find answers is thus useless.

6.2 Success in $\mathrm{ECLS}_{\mathrm{F}}$ (The Extended Clausal Logic)

In this paper, we are successful to create an MI solver based on the LPSF with a general logical structure as much as possible. We already have achieved another approach to introduce constraint is one of key considerations on building such a general one (Akama and Nantajeewarawat, 2018). However, rules which work independently from constraints would benefit MI solvers to focus on creating rules which are more specific and effective to a certain problem.

Based on LPSF, transformation rule is a small building block to construct an algorithm for problems which requires logical structure for representation. We proved the correctness of each individual transformation rule. This means we can reuse safely newly invented rules in other scenario where same knowledge representation may fit to MI-solver. When integrating invented rules, we don't need additional effort to prove overall correctness as long as the rule-by-rule correctness is given.

One of KR-logic unique approach is the existence of *func*-atom using function variable and a substitution σ. As shown in the computation process of Agatha puzzle, we have rich computation algorithm using ET rules to handling *func*-atom.

7 CONCLUSIONS

The conventional clauses don't have enough expressive power of existential quantification that is required to represent and compute the Agatha puzzle. Function variables are essentially used for transformation of existential quantification. We take extended clauses in $\mathrm{ECLS}_{\mathrm{F}}$ that have function variables. Hence, we need

a new logic and logical computation for the extended space.

To solve the Agatha puzzle, we discover ET rules to handle function variables. We obtained five general ET rules as follows: (1) side-change of *neq* body-atoms to *eq* head-atoms, (2) elimination of candidate *eq* head-atom, (3) elimination of *func* atom by enforced specialization, (4) elimination of a clause by false *func* atom, and (5) elimination of a clause containing an isolated *func* atom.

These invented rules are discovered through the squeeze method. We also prove the correctness of each rule strictly. Thus, we construct an MI solver based on the extended space on $\mathrm{ECLS_F}$. The solver is applied to not only the Agatha puzzle, but also any other problems represented similarly in extended clauses. It is shown that the proposed new ET rules contribute to an improvement of a solver. We can make a constant progress by adopting a suitable representation space for the problem that cannot be solved in the conventional clause space. Increasing the capability by accumulating properly validated ET rules is a core research methodology of logical problem solving framework.

REFERENCES

Akama, K. and Nantajeewarawat, E. (2011). Meaning-preserving skolemization. In *Proc. 3rd international conference on Knowledge Engineering and Ontology Development (KEOD)*, pages 322–327, Paris, France.

Akama, K. and Nantajeewarawat, E. (2013). Unfolding-based simplification of query-answering problems in an extended clause space. *International Journal of Innovative Computing*, Information and Control 9:3515–3526.

Akama, K. and Nantajeewarawat, E. (2014). Equivalent transformation in an extended space for solving query-answering problems. In *Proc. 6th Asian Conference on Intelligent Information and Database Systems*, pages 232–241, Bangkok, Thailand.

Akama, K. and Nantajeewarawat, E. (2015). Function-variable elimination and its limitations. In *Proc. 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 2, pages 212–222, Lisbon, Portugal.

Akama, K. and Nantajeewarawat, E. (2016). Model-intersection problems with existentially quantified function variables: Formalization and a solution schema. In *Proc. 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 2, pages 52–63, Porto, Portugal.

Akama, K. and Nantajeewarawat, E. (2018). Solving query-answering problems with constraints for function variables. *Springer International Publishing AG, part of Springer Nature 2018, N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10751*, pages 36–47.

Akama, K., Nantajeewarawat, E., and Akama, T. (2018a). Computation control by prioritized et rules. In *Proc. 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD)*, volume 2, pages 84–95, Seville, Spain.

Akama, K., Nantajeewarawat, E., and Akama, T. (2018b). Side-change transformation. In *Proc. 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD)*, volume 2, pages 237–246, Seville, Spain.

Akama, K., Nantajeewarawat, E., and Akama, T. (2019). Logical problem solving framework. *Springer International Publishing AG, part of Springer Nature 2019, N. T. Nguyen et al. (Eds.): ACIIDS 2019, LNAI 11431*, pages 28–40.

Chang, C.-L. and Lee, R. C.-T. (1973). Academic Press.

Lloyd, J. W. (1987). Springer-Verlag, second edition.