Inverting Abstract Unification for Set-Sharing

Xuan Li
Oakland University
Rochester, MI 48309, USA
x2li@oakland.edu

Lunjin Lu
Oakland University
Rochester, MI 48309, USA
l2lu@oakland.edu

ABSTRACT

This paper presents an inversion of the abstract unification operator for forward set-sharing analysis of logic programs. The inverted operator, called a backward abstract unification operator, computes all maximal pre-conditions for a given equation and its post-condition. It is a key operator in a backward analysis. The maximal preconditions are obtained by first calculating a superset of all preconditions and then generating all maximal pre-conditions from the superset. The latter step is transformed to the problem of finding all maximal models of a Boolean formula.

Categories and Subject Descriptors

F.3.2 [**LOGICS AND MEANINGS OF PROGRAMS**]: Semantics of Programming Languages—*Program analysis* ; D.2.8 [**PROGRAMMING LANGUAGES**]: Processors—*Compilers*

General Terms

Algorithms, Languages

Keywords

Backward Analysis, Set-Sharing, Logic Programs, Maximal Models and Boolean Formulas

1. INTRODUCTION

If a programmer wants to call a predicate in a third party program and knows what must hold after the call, it is natural to ask whether there is any condition on the call. This is a point where backward analysis [17, 18] comes to play. Given a post-condition and a goal, backward analysis infers a pre-condition that guarantees satisfaction of the post-condition. Once we know the pre-condition and the post-condition, we can treat the third party library as a black box and use it without inspecting it line by line. Another popular application of backward analysis is software verification [3]. By inferring the pre-condition and comparing it with the programmer's specification, the analyzer can tell whether the program satisfies the specification [10]. Backward analysis is goal independent and derives all information in one application.

A backward analysis can be designed by abstract interpretation of a backward collecting semantics in the same way a forward analysis is. An alternative approach is to obtain a backward analysis by inverting a forward analysis. This will reduce the design effort when there exists a forward analysis for the program property of interest because the correctness of the backward analysis follows from the correctness of the forward analysis [11]. Inverting a forward analysis boils down to inverting its primitive operators. A forward analysis analyzes a primitive program construct C by applying a primitive operator f that abstracts the execution of C. Given a property ϕ that describes the program state before the execution of C, forward analysis computes a property ψ that describes the program state after the execution of C by applying f to ϕ, that is, $\psi = f(\phi)$. The properties ϕ and ψ are called pre and post conditions respectively. [6, 7, 9] In a backward analysis, this computation is inverted. The backward analysis computes all pre-conditions ϕ for C such that a given post-condition ψ is satisfied. The set of those pre-conditions is $\{\phi | f(\phi) \sqsubseteq \psi\}$ where $\sqsubseteq$ is partial order on properties. Let Φ be a set of properties. The set of maximal elements in Φ with respect to $\sqsubseteq$ is denoted by $max(\Phi)$. Due to monotonicity of f, $\{\phi | f(\phi) \sqsubseteq \psi\}$ is down-closed with respect to $\sqsubseteq$ and contains the same information as $max(\{\phi | f(\phi) \sqsubseteq \psi\})$. Thus, given a post-condition ψ, it is only necessary to compute those maximal pre-conditions ϕ_m such that $f(\phi_m) \sqsubseteq \psi$.

In a set-sharing analysis [12, 13] for logic programs, information on aliasing dependencies and groundness dependencies between variables is described by a set of sets of variables where each set of variables is called a group. The most important operator in the analysis is a forward abstract unification operator. This paper presents an inverse of the forward abstract unification operator. The inverted operator, called a backward abstract unification operator, computes $max(\{\phi | amgu_{x=t}(\phi) \sqsubseteq \psi\})$ where $amgu_{x=t}$ is the forward abstract unification operator and $x = t$ is an equation in the program. We assume that x is a variable and t a term such that x does not occur in t.

A brute-force approach to computing the set of pre-conditions for an equation and a post-condition is to apply forward abstract unification to all sharings. The pre-conditions are those sharings on which the output of forward abstract uni-

PPDP'05, July 11–13, 2005, Lisbon, Portugal.

fication is contained in the post-condition. However, this will result in an inefficient algorithm. The proposed backward abstract unification algorithm avoids applying forward abstract unification on all sharings. It first computes a superset of all pre-conditions and then generates maximal pre-conditions by removing sharing groups from the superset. The main contribution of this paper is a backward abstract unification operator for set-sharing that is obtained by inverting the forward abstract unification operator for set-sharing.

The remainder of the paper is organized as follows. Section 2 contains basic concepts and recalls the set-sharing domain. Section 3 informally introduces the algorithm via an example. Sections 4-7 present the backward abstract unification operator in details. Given an equation and its post-condition, the backward abstract unification operator first normalizes the post-condition by filtering out those groups that can not be generated by the forward abstract unification operator. Section 4 presents the normalization step. It also computes two special pre-conditions that are maximal among those pre-conditions that are not relevant to both sides of the equation. The remaining maximal pre-conditions are generated from the union of all the pre-conditions that are relevant to both sides of the equation. Section 5 shows how to compute the union without applying the forward abstract unification operator. Section 6 reduces the problem of finding those maximal pre-conditions to the problem of computing maximal models of a Boolean formula. Section 7 puts together the overall backward abstract unification operator. Section 8 discusses related work and section 9 concludes.

2. PRELIMINARIES

This section recalls some basic concepts that will be used in the following sections.

Assume that there are a set of variables V of interest. Let $var(o)$ be the set of variables in the syntactic object o. An equation over V is a formula of the form $a = b$ with a and b are terms such that $var(a), var(b) \subseteq V$.

Let $B = \{1, 0\}$. A Boolean function is a function $f : B^n \rightarrow B$. It is well known that the domain of Boolean functions is isomorphic to the domain of Boolean formulas. We will not distinguish between Boolean formulas and Boolean functions. A truth assignment Int assigns truth value to variables. We say a truth assignment Int is a model for f, denoted as $Int \models f$, if f evaluates to 1 after each variable x in f is replaced by $Int(x)$. We sometimes write a truth assignment as a set $M = \{X | Int(X) = true\}$. A Boolean function is said to be satisfiable if it has at least one model. Let m and m' be two models, we say $m \leq m'$ if $m(X) = 1$ implies $m'(X) = 1$. We say a model m is a maximal model if there is no other model m' such that $m \leq m'$.

Sharing analysis [13, 12, 24, 2, 19, 4, 16, 5, 23] is an important topic in static analysis of logic programs. There are two widely used abstract domains: set-sharing [12] and pair-sharing [24]. Inverting pair-sharing is presented in [21]. Inverting set-sharing is more complex than inverting pair-sharing because of the complexity of the set-sharing domain and its associated operators.

Set-sharing is due to Dean Jacobs and Anno Langen [12]. The abstract domain $Sharing$ is $\{S \in \wp(\wp(V)) \mid \emptyset \in S\}$. The downward closure of a sharing S is $\downarrow S = \{G \mid \exists G' \in S, G \subseteq G'\}$. The most important operator is the forward abstract unification operator $amgu_{x=t}$. The operator $amgu_{x=t}$ is defined in term of three auxiliary operations. The closure under union of a sharing ϕ is denoted ϕ^*, which is the smallest superset of ϕ satisfying $X \in \phi^* \wedge Y \in \phi^* \Longrightarrow X \cup Y \in \phi^*$. The set of groups in ϕ that are relevant to term t is $rel(t, \phi) = \{G \in \phi \mid var(t) \cap G \neq \emptyset\}$. Let $rel(x = t, \phi) = rel(x, \phi) \cup rel(t, \phi)$. We say that a sharing ϕ is relevant to a term t iff $rel(t, \phi) \neq \emptyset$. The cross union of two sharings defined as ϕ_1 and ϕ_2 is $\phi_1 \uplus \phi_2 = \{G \cup H | G \in \phi_1 \wedge H \in \phi_2\}$.

In this paper, we assume that the equation is of the form $x = t$ where x is a variable and t is a term that does not contains x, because other situations can be reduced to this basic case. The following definition of the forward abstract unification is adapted from [12].

Definition 1. (Jacobs and Langen) [13, 12]
$amgu_{x=t}(\phi) = \phi \backslash rel(x, \phi) \backslash rel(t, \phi) \cup (rel(x, \phi)^* \uplus rel(t, \phi)^*)$

3. WORKED EXAMPLE

This section informally introduces the proposed algorithm for backward abstract unification. Given an equation and its post-condition, the brute-force approach tries all possible sharings in order to find all maximal pre-conditions. Suppose that there are k variables of interest. Then there are 2^{2^k-1} possible sharings. The brute-force approach applies for 2^{2^k-1} times the forward abstract unification that is itself exponential [1]. Thus the brute-force approach is impractical since it is prohibitively expensive even for a small k, say 5.

The algorithm proposed in this paper is much more efficient than the brute-force approach. For a given equation and its post-condition, the algorithm first computes two special pre-conditions that are maximal among those pre-conditions that are not relevant to both x and t. Secondly, it computes the union of the pre-conditions that are relevant to both x and t and then generates maximal pre-conditions that are relevant to both x and t from the union. The union, called the cover for the equation and the post-condition can be computed without applying the forward abstract unification operator. We now illustrate this process via an example.

Normalizing the post-condition

The first step in calculating the maximal pre-conditions is to normalize the post-condition by filtering out sharing groups that can never occur in the result produced by forward abstract unification. This is done by making use of the following property: a sharing group in the result of an application of the forward abstract unification operator is relevant to both sides of the equation or to neither side of the equation.

Example 1. Let the equation be $x_1 = f(x_2, x_3)$ and its post-condition $\{\emptyset, \{x_4\}, \{x_2, x_5\}, \{x_1, x_2, x_3\}\}$ denoted ω. Observe that sharing group $\{x_2, x_5\}$ is relevant to $f(x_2, x_3)$ but not to x_1. Therefore, it cannot occur in the result of a forward abstract unification. In other words, letting $\psi = \omega \setminus \{\{x_2, x_5\}\}$, any pre-condition for $x_1 = f(x_2, x_3)$ and ω is also a pre-condition for $x_1 = f(x_2, x_3)$ and ψ and vice versa. So, we can normalize ω into ψ and computing maximal pre-conditions using the smaller sharing ψ as the post-condition instead of the bigger sharing ω. ∎

Computing special pre-conditions

We observe that there are two special pre-conditions:

$$\phi_x = \{\emptyset, \{x_4\}\} \cup \{\{x_1\}, \{x_1, x_4\}, \{x_1, x_5\}, \{x_1, x_4, x_5\}\}$$

and

$$\phi_t = \{\emptyset, \{x_4\}\} \cup \left\{ \begin{array}{c} \{x_2\}, \{x_3\}, \{x_2, x_3\}, \{x_2, x_4\}, \\ \{x_2, x_5\}, \{x_3, x_4\}, \{x_3, x_5\}, \\ \{x_2, x_3, x_4\}, \{x_2, x_3, x_5\}, \{x_2, x_4, x_5\}, \\ \{x_3, x_4, x_5\}, \{x_2, x_3, x_4, x_5\} \end{array} \right\}$$

ϕ_x is only relevant to x_1 but not $f(x_2, x_3)$ and ϕ_t is only relevant to $f(x_2, x_3)$ but not x_1. These two pre-conditions are maximal among those pre-conditions that are not relevant to both x_1 and $f(x_2, x_3)$. All other maximal pre-conditions are relevant to both x_1 and $f(x_2, x_3)$.

Calculating the cover

The cover ϕ_c for an equation $x = t$ and its post-condition ψ is computed without applying the forward abstract unification operator $amgu_{x=t}$. Assume that ψ is normalized. The computation is accomplished by making use of several properties about $amgu_{x=t}$ that will be proved later. Firstly, each maximal pre-condition for $x = t$ and ψ contains all those groups in ψ that are not relevant to either x or t. Secondly, if a group G in a pre-condition for $x = t$ and ψ is relevant to both x and t then G occurs in ψ. Thirdly, if G is relevant to one of x and t but not the both and G is in a pre-condition ϕ that is relevant to both x and t then G is in $\downarrow rel(x = t, \psi)$. So, the cover is union of ψ and the result of removing from $\downarrow rel(x = t, \psi)$ those groups that are relevant to both x and t and those groups that are relevant to neither x nor t.

Example 2. Continue with example 1.

$$\psi = \{\emptyset, \{x_4\}, \{x_1, x_2, x_3\}\}$$

then

$$\begin{array}{lcl} rel(x_1 = f(x_2, x_3), \psi) & = & \{\{x_1, x_2, x_3\}\} \\ \downarrow rel(x_1 = f(x_2, x_3), \psi) & = & \left\{ \begin{array}{c} \emptyset, \{x_1\}, \{x_2\}, \{x_3\}, \\ \{x_1, x_2\}, \{x_1, x_3\}, \\ \{x_2, x_3\}, \{x_1, x_2, x_3\} \end{array} \right\} \end{array}$$

The cover for $x_1 = f(x_2, x_3)$ and ψ is then obtained by removing $\emptyset$, $\{x_1, x_2\}$, $\{x_1, x_3\}$ and $\{x_1, x_2, x_3\}$ from $\downarrow rel(x_1 = f(x_2, x_3), \psi)$ and joining the result with ψ, resulting in

$$\phi_c = \{\emptyset, \{x_4\}, \{x_1\}, \{x_2\}, \{x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$$

Any pre-condition that is relevant to both x and t is a subset of ϕ_c. ∎

Generating a Boolean formula

The remainder of the computation is reduced into the problem of computing maximal models of a Boolean formula that is generated as follows. The forward abstract unification operator $amgu_{x=t}$ is first applied to the cover ϕ_c. The result $amgu_{x=t}(\phi_c)$ may contain groups that are not in the post-condition ψ. Then a sufficient and necessary condition about a pre-condition that is a subset of ϕ_c is generated from these extra groups.

Example 3. Continue with examples 1 and 2. We have

$$\begin{array}{l} amgu_{x_1=f(x_2,x_3)}(\phi_c) \setminus \psi \\ = \{\emptyset, \{x_4\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\} \setminus \psi \\ = \{\{x_1, x_2\}, \{x_1, x_3\}\} \end{array}$$

Maximal pre-conditions are obtained by removing sharing groups from ϕ_c so that the extra sharing groups $\{x_1, x_2\}$ and $\{x_1, x_3\}$ are not generated by the forward abstract unification. In other words, a sharing $\phi \subseteq \phi_c$ is a pre-condition iff $I \notin amgu_{x=t}(\phi)$ holds for each extra group I in $amgu_{x=t}(\phi_c) \setminus \psi$.

Consider the extra group $\{x_1, x_3\}$ first. By the definition of $amgu$, the formula $\{x_1, x_3\} \notin amgu_{x_1=f(x_2,x_3)}(\phi)$ is equivalent to $\{x_1, x_3\} \notin \phi_1 \uplus \phi_2$ since $\{x_1, x_3\}$ is relevant to $x_1 = f(x_2, x_3)$ where $\phi_1 = rel(x_1, \phi)^*$ and $\phi_2 = rel(f(x_2, x_3), \phi)^*$. By the definition of $\uplus$ and the condition $\phi \subseteq \phi_c$, $\{x_1, x_3\} \notin \phi_1 \uplus \phi_2$ is equivalent to that for any pair of non-empty groups $\langle G, H \rangle$ such that G is relevant to x_1, H is relevant to $f(x_2, x_3)$, $G \in \phi_c$, $H \in \phi_c$ and $G \cup H = \{x_1, x_3\}$, either $G \notin \phi_1$ or $H \notin \phi_2$. There are one such pair: $\langle \{x_1\}, \{x_3\} \rangle$. We thus obtain the following formula that is equivalent to $\{x_1, x_3\} \notin amgu_{x_1=f(x_2,x_3)}(\phi)$.

$$\{x_1\} \notin rel(x_1, \phi)^* \vee \{x_3\} \notin rel(f(x_2, x_3), \phi)^*$$

The above formula is then transformed into an equivalent one that consists of primitive formulas of the form $G \notin \phi$ and logical operations $\wedge$ and $\vee$ where $G \in \phi_c$. For instance, $\{x_1\} \notin rel(x_1, \phi)^*$ is transformed to $\{x_1\} \notin \phi$. The details of this transformation will be described in section 6. In this way, $\{x_1, x_3\} \notin amgu_{x_1=f(x_2,x_3)}(\phi)$ is transformed and simplified into

$$\{x_1\} \notin \phi \vee \{x_3\} \notin \phi$$

Let B be an 1-1 function that maps a group in ϕ_c to a Boolean variable. Assigning a Boolean variable the truth value 1 means the corresponding group is in ϕ. Assigning a Boolean variable the truth value 0 indicates that corresponding group is not in ϕ. The above formula corresponds to the following Boolean formula.

$$\overline{B(\{x_1\})} \vee \overline{B(\{x_3\})} \quad (1)$$

which is equivalent to $\{x_1, x_3\} \notin amgu_{x_1=f(x_2,x_3)}(\phi)$. In the same way, we can show that $\{x_1, x_2\} \notin amgu_{x_1=f(x_2,x_3)}(\phi)$ is equivalent to

$$(\overline{B(\{x_1\})} \vee \overline{B(\{x_2\})} \quad (2)$$

Conjoining (1) and (2), we obtain the following sufficient and necessary condition for a pre-condition ϕ for $x_1 = f(x_2, x_3)$ and ψ such that $\phi \subseteq \phi_c$

$$((\overline{B(\{x_1\})} \vee \overline{B(\{x_3\})})) \quad \wedge \quad ((\overline{B(\{x_1\})} \vee \overline{B(\{x_2\})})) \quad (3)$$

∎

Computing Maximal Models

The Boolean formula generated from a given equation and its post-condition in the previous steps expresses the sufficient and necessary condition on pre-conditions that are contained in ϕ_c. A pre-condition that is contained in ϕ_c corresponds to a model of the Boolean formula. Since ϕ_c contains each and every pre-condition that is relevant to both x and t, the maximal pre-conditions that are relevant to both x and t corresponds to the maximal models of the

Boolean formula. How to generate all maximal models of a Boolean function is a well known problem [15], which can be transformed to the problem of generating all maximal independent sets [14, 25].

Example 4. Continue with example 3. Applying an existing algorithm [15], we get all maximal models of (3) as follows.

$$M_1 = \left\{ \begin{array}{c} B(\{x_1\}) = 0, B(\{x_2\}) = 1, \\ B(\{x_3\}) = 1, B(\{x_4\}) = 1, \\ B(\{x_2, x_3\}) = 1, B(\{x_1, x_2, x_3\}) = 1 \end{array} \right\}$$

$$M_2 = \left\{ \begin{array}{c} B(\{x_1\}) = 1, B(\{x_2\}) = 0, \\ B(\{x_3\}) = 0, B(\{x_4\}) = 1, \\ B(\{x_2, x_3\}) = 1, B(\{x_1, x_2, x_3\}) = 1 \end{array} \right\}$$

Maximal model M_1 and M_2 correspond to maximal pre-conditions μ_1 and μ_2 in that $\mu_i = \{G \in \phi_c | M_i(B(G)) = 1\}$ and

$$\begin{aligned} \mu_1 &= \{\emptyset, \{x_2\}, \{x_3\}, \{x_4\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\} \\ \mu_2 &= \{\emptyset, \{x_1\}, \{x_4\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\} \end{aligned}$$

Observe that $\mu_1 \cup \mu_2$ equals ϕ_c which is not a pre-condition. ∎

All maximal pre-conditions

We now have maximal pre-conditions that are relevant to both x and t. We also have two special pre-conditions ϕ_x and ϕ_t that are maximal among those pre-condition that are not relevant to both x and t. So, the set of all maximal pre-condition is $max(\{\mu_1, \mu_2, \phi_x, \phi_t\})$.

$max(\{\mu_1, \mu_2, \phi_x, \phi_t\}) =$

$$\left\{ \begin{array}{c} \{\emptyset, \{x_2\}, \{x_3\}, \{x_4\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}, \\ \{\emptyset, \{x_1\}, \{x_4\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}, \\ \{\emptyset, \{x_4\}, \{x_1\}, \{x_1, x_4\}, \{x_1, x_5\}, \{x_1, x_4, x_5\}\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_3\}, \{x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \\ \{x_2, x_5\}, \{x_3, x_4\}, \{x_3, x_5\}, \\ \{x_2, x_3, x_4\}, \{x_2, x_3, x_5\}, \{x_2, x_4, x_5\}, \\ \{x_3, x_4, x_5\}, \{x_2, x_3, x_4, x_5\} \end{array} \right\} \end{array} \right\}$$

In summary, given an equation $x = t$ and its post-condition ω, maximal pre-conditions are obtained by normalizing the post-condition, calculating two special pre-conditions, calculating the cover, generating a Boolean formula that expresses the sufficient and necessary for pre-conditions that are contained in ϕ_c and generating maximal models of the Boolean formula. The next sections present these steps in details.

4. NORMALIZING POST-CONDITION

For a given equation $x = t$ and its post-condition ω, backward abstract unification computes $max(\{\phi \mid amgu_{x=t}(\phi) \subseteq \omega\})$. Since the post-condition ω is not necessarily the result of an application of the forward abstract unification operator, it may contain groups that can never be generated by the forward abstract unification. Such groups can be removed from ω without missing any maximal pre-condition. Let $\psi = amgu_{x=t}(\phi)$ for some sharing ϕ. Then each group G in ψ is either related to both x and t or neither of them. Thus, the first step in the backward abstract unification is to filter out groups in ω that is relevant to either x or t but not the both. Let

$$\psi = \omega \setminus rel(x, \omega) \setminus rel(t, \omega) \cup (rel(x, \omega) \cap rel(t, \omega))$$

Then $\{\phi \mid amgu_{x=t}(\phi) \subseteq \psi\}$ is equal to $\{\phi \mid amgu_{x=t}(\phi) \subseteq \omega\}$. The sharing ψ is called a normalized post-condition.

Example 5. Let the equation be $x_2 = x_4$ and its post-condition

$$\omega = \left\{ \begin{array}{c} \emptyset, \{x_5\}, \{x_1, x_2, x_3, x_4\}, \{x_1, x_2, x_3\}, \\ \{x_2, x_4\}, \{x_2, x_3\}, \{x_1, x_3, x_4\}, \{x_2, x_3, x_4\} \end{array} \right\}$$

Then the normalized post-condition is

$$\psi = \{\emptyset, \{x_5\}, \{x_2, x_4\}, \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\}\}$$

Maximal pre-condition for $x = t$ and ψ is maximal pre-condition of $x = t$ and ω and vice verse. ∎

It follows that for a normalized post-condition ψ, $rel(x, \psi) = rel(t, \psi)$. In the sequel, we assume that post-conditions are normalized and thus $rel(x, \psi) = rel(t, \psi)$.

The following lemmas states that each and every maximal pre-conditions contains a fixed part that can be computed from $x = t$ and ψ.

Lemma 1. If ϕ is a pre-condition for $x = t$ and ψ then $\phi \cup (\psi \setminus rel(x, \psi))$ is a pre-condition for $x = t$ and ψ.

PROOF. We have that $amgu_{x=t}(\phi \cup (\psi \setminus rel(x, \psi))) = amgu_{x=t}(\phi) \cup (\psi \setminus rel(x, \psi))$. Since $amgu_{x=t}(\phi) \subseteq \psi$ and $(\psi \setminus rel(x, \psi)) \subseteq \psi$, we have that $amgu_{x=t}(\phi \cup (\psi \setminus rel(x, \psi))) \subseteq \psi$. □

For instance, in example 5, $\psi \setminus rel(x, \psi) = \{\emptyset, \{x_5\}\}$ is in every maximal pre-condition.

A pre-condition for $x = t$ and ψ is either relevant to both x and t or it is not. The following lemma allows us to focus on those pre-conditions that are relevant to both x and t.

Theorem 1. Let $\phi_x = \psi \setminus rel(x, \psi) \cup \{G \in \wp(V) | G \cap var(t) = \emptyset \wedge x \in G\}$ and $\phi_t = \psi \setminus rel(x, \psi) \cup \{G \in \wp(V) | G \cap var(t) \neq \emptyset \wedge x \notin G\}$. Then,

(a) $amgu_{x=t}(\phi_x) \subseteq \psi$ and if $rel(t, \phi) = \emptyset$ and $amgu_{x=t}(\phi) \subseteq \psi$ then $\phi \subseteq \phi_x$; and

(b) $amgu_{x=t}(\phi_t) \subseteq \psi$ and if $rel(x, \phi) = \emptyset$ and $amgu_{x=t}(\phi) \subseteq \psi$ then $\phi \subseteq \phi_t$.

PROOF. We only prove (a) since (b) is symmetric to (a). Let $\eta = \{G \in \wp(V) | G \cap var(t) = \emptyset \wedge x \in G\}$. Then $\phi_x = (\psi \setminus rel(x, \psi)) \cup \eta$. We first prove that $amgu_{x=t}(\phi_x) \subseteq \psi$. We have that $rel(x, \phi_x) = \eta$ and that $rel(t, \phi_x) = \emptyset$. So, $amgu_{x=t}(\phi_x) = \phi_x \setminus \eta = \psi \setminus rel(x, \psi) \subseteq \psi$.

The rest of the proof is done by contradiction. Assume that $rel(t, \phi) = \emptyset$, $amgu_{x=t}(\phi) \subseteq \psi$ and $\phi \not\subseteq \phi_x$. Then there is a $G \in \phi$ and $G \notin \phi_x$. Since $G \in \phi$, $G \cap var(t) = \emptyset$. Since $G \notin \phi_x$, $G \notin \eta$ that together with $G \cap var(t) = \emptyset$ implies $x \notin G$. So, $G \in amgu_{x=t}(\phi)$, implying $G \in \psi$. Since $G \notin \phi_x$, $G \notin (\psi \setminus rel(x, \psi))$ together with $G \in \psi$ implies that $x \in G$. Both $x \in G$ and $x \notin G$, which is a contradiction. □

Theorem 1 implies that ϕ_x and ϕ_t are pre-conditions for $x = t$ and ψ. Furthermore, any pre-condition for $x = t$ and ψ that is not relevant to both x and t is contained in either ϕ_x or ϕ_x. In the case where a pre-condition ϕ for $x = t$ and ψ is not relevant to either x or t, ϕ is contained in both ϕ_x and ϕ_t.

Example 6. Continue with example 5. Let

$$V = \{x_1, x_2, x_3, x_4, x_5\}$$

Then

$$\phi_x = \{\emptyset, \{x_1, x_5\}\} \cup \left\{ \begin{array}{c} \{x_2\}, \{x_1, x_2\}, \{x_2, x_3\}, \{x_2, x_5\}, \\ \{x_1, x_2, x_3\}, \{x_1, x_2, x_5\}, \\ \{x_2, x_3, x_5\}, \{x_1, x_2, x_3, x_5\} \end{array} \right\}$$

$$\phi_t = \{\emptyset, \{x_5\}\} \cup \left\{ \begin{array}{c} \{x_4\}, \{x_1, x_4\}, \{x_3, x_4\}, \\ \{x_4, x_5\}, \{x_1, x_3, x_4, \}, \{x_1, x_4, x_5\}, \\ \{x_3, x_4, x_5\}, \{x_1, x_3, x_4, x_5\} \end{array} \right\}$$

∎

Lemma 2. If $rel(x, \psi) = \emptyset$ then $max(\{\phi \mid amgu_{x=t}(\phi) \subseteq \psi\}) = max(\{\phi_x, \phi_t\})$, where ϕ_x, ϕ_t are defined in theorem 1.

PROOF. Since $rel(x, \psi) = rel(t, \psi) = \emptyset$ by assumption, we have $rel(t, \phi_x) = \emptyset$ and $rel(x, \phi_x) = \{G \in \wp(V) | G \cap var(t) = \emptyset \wedge x \in G\}$. Therefore, $amgu_{x=t}(\phi_x) = \psi$. Similarly, we can prove that $amgu_{x=t}(\phi_t) = \psi$. It remains to prove that ϕ_x and ϕ_t are the only maximal pre-conditions for $x = t$ and ψ. By way of contradiction, assume that $\phi \not\subseteq \phi_x$ and $\phi \not\subseteq \phi_t$ and $amgu_{x=t}(\phi) \subseteq \psi$. Since $\phi \not\subseteq \phi_x$ and $\phi \not\subseteq \phi_t$, there is a group G in ϕ such that either (i) $x \in G$ and $G \cap vars(t) \neq \emptyset$ or (ii) $G \notin \psi$ and $x \notin G$ and $G \cap vars(t) = \emptyset$. Consider the case (i) first. We have that $G \in rel(x, \phi)$ and $G \in rel(t, \phi)$ and hence, by the definition of $amgu_{x=t}$, we have that $G \in amgu_{x=t}(\phi)$. This implies that $G \in \psi$ and $rel(x, \psi) \neq \emptyset$ that is a contradiction to that $rel(x, \psi) = \emptyset$. In the case (ii), $G \notin rel(x, \phi)$ and $G \notin rel(t, \phi)$. We have that $G \in \phi \setminus rel(x, \phi) \setminus rel(x, \phi)$ and hence $G \in amgu_{x=t}(\phi)$. This implies $G \in \psi$ that contradicts that $G \notin \psi$. □

Observe that if $\psi = \{\emptyset\}$ then $rel(x, \psi) = \emptyset$ and the above lemma applies.

Example 7. Let $V = \{x_1, x_2, x_5\}$, equation be $x_1 = x_5$ and $\psi = \{\emptyset, \{x_2\}\}$. Then

$$\phi_x = \{\emptyset, \{x_2\}, \{x_1\}, \{x_1, x_2\}\}$$

and

$$\phi_t = \{\emptyset, \{x_2\}, \{x_5\}, \{x_5, x_2\}\}$$

∎

ϕ_t is contained in ϕ_x if $var(t) = \emptyset$ as shown in following example. This explains why max is used in lemma 2.

Example 8. Consider the equation $x_1 = a$ and its post-condition be $\psi = \{\emptyset, \{x_2\}\}$ where a is a constant. According to lemma 2, we have $max(\{\phi \mid amgu_{x_1=a}(\phi) \subseteq \psi\})$ equals $\{\emptyset, \{x_2\}\} \cup \{G \in \wp(V) | x_1 \in G\}$ ∎

5. CALCULATING COVER

The remaining task in backward abstract unification is to find those maximal pre-conditions for $x = t$ and ψ that are relevant to both x and t. The following lemma states that any group in such a pre-condition is either in the downward closure of $rel(x, \psi)$ or it is not relevant to $x = t$.

Lemma 3. Let ϕ be a sharing such that $rel(x, \phi) \neq \emptyset \wedge rel(t, \phi) \neq \emptyset$. If $amgu_{x=t}(\phi) \subseteq \psi$ then G is in $\downarrow rel(x, \psi)$ for each G in $rel(x = t, \phi)$.

PROOF. Let $G \in rel(x = t, \phi)$. Then $G \in \phi$ and either $x \in G$ or $G \cap var(t) \neq \emptyset$. Without loss of generality, we assume that $x \in G$. By the definition of $amgu_{x=t}$ and the condition that $rel(t, \phi) \neq \emptyset$, there is an $H \in rel(t, \phi)$ such that $G \cup H \in rel(x, \psi)$. So, $G \in \downarrow rel(x, \psi)$. □

Maximal pre-conditions for $x = t$ and ψ that are relevant to both x and t are generated from a sharing that is their union. The sharing is called the cover for $x = t$ and ψ and defined as

$$Cov(x = t, \psi) = \bigcup \left\{ \phi \;\middle|\; \begin{array}{c} (amgu_{x=t}(\phi) \subseteq \psi) \\ \wedge \\ (rel(x, \phi) \neq \emptyset) \wedge (rel(t, \phi) \neq \emptyset) \end{array} \right\}$$

The following theorem gives a method that computes cover $Cov(x = t, \psi)$ without applying the forward abstract unification operator $amgu_{x=t}$.

Lemma 4.
$Cov(x = t, \psi) =$
$\psi \cup \{G \in \downarrow rel(x = t, \psi) \mid (x \in G) \oplus (var(t) \cap G \neq \emptyset)\}$
where $\oplus$ is exclusive disjunction.

PROOF. ($\subseteq$) We first assume that $amgu_{x=t}(\phi) \subseteq \psi$ and $rel(x, \phi) \neq \emptyset$ and $rel(t, \phi) \neq \emptyset$. Let $H \in \phi$. If $H \notin rel(x = t, \phi)$ then $H \in amgu_{x=t}(\phi)$ and hence $H \in \psi$. If H is only relevant to x or only relevant to t then according to lemma 3, $H \in \{G \in \downarrow rel(x = t, \psi) | (x \in G) \oplus (var(t) \cap G \neq \emptyset)\}$. Otherwise, H is relevant to both x and t and we have that $H \in amgu_{x=t}(\phi)$ and hence $H \in \psi$.

($\supseteq$) We first prove that $Cov(x = t, \psi) \supseteq \psi$. Let H be an arbitrary group in ψ. If $H \in \psi \setminus rel(x, \psi)$ then $H \in Cov(x = t, \psi)$ by lemma 1 that states that every maximal pre-condition contains $\psi \setminus rel(x, \psi)$. Otherwise, $H \in rel(x, \psi)$. We have $amgu_{x=t}(\{H\}) = \{H\} \subseteq \psi$ since $rel(x, \{H\}) = rel(t, \{H\}) = \{H\} \neq \emptyset$. So, $H \in Cov(x = t, \psi)$. Therefore, $Cov(x = t, \psi) \supseteq \psi$.

Let H be an arbitrary group in $\{G \in \downarrow rel(x = t, \psi) | (x \in G) \oplus (var(t) \cap G \neq \emptyset)\}$. There is an G in $\downarrow rel(x = t, \psi)$ such that $H \subseteq G$. Also H is relevant to one and only one of x and t. Without loss of generality, assume that H is relevant to only x. We then have $rel(x, \{H, G\}) = \{H, G\}$ and $rel(t, \{H, G\}) = \{G\}$. It follows that $amgu_{x=t}(\{H, G\}) = (\{H, G\} \setminus \{H, G\} \setminus \{G\}) \cup \{H, G\}^* \uplus \{G\}^* = \{G\} \subseteq \psi$. Recall that G is relevant to both x and t. Thus, $H \in Cov(x = t, \psi)$.

□

Example 9. Continue with example 5. We have

$$\psi = \{\emptyset, \{x_5\}, \{x_2, x_4\}, \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\}\}$$

$$\{G \in \downarrow rel(x = t, \psi) \mid (x \in G) \oplus (var(t) \cap G \neq \emptyset)\}$$
$$= \left\{ \begin{array}{c} \{x_2\}, \{x_4\}, \{x_1, x_2\}, \{x_1, x_4\}, \\ \{x_2, x_3\}, \{x_3, x_4\}, \\ \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\} \end{array} \right\}$$

Hence

$$Cov(x = t, \psi) = \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_4\}, \{x_5\}, \{x_2, x_4\}, \\ \{x_1, x_2\}, \{x_1, x_4\}, \{x_2, x_3\}, \\ \{x_3, x_4\}, \{x_1, x_2, x_3\}, \\ \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\} \end{array} \right\}$$

which is denoted by ϕ_c in subsequence examples. ∎

6. GENERATING BOOLEAN FORMULA

Let $\phi_c = Cov(x = t, \psi)$. We now present a method for computing those maximal pre-conditions for $x = t$ and ψ that are relevant to both x and t. If $amgu_{x=t}(\phi_c) \subseteq \psi$ then the cover ϕ_c is the only maximal pre-condition that is relevant to both x and t. If $amgu_{x=t}(\phi_c) \not\subseteq \psi$, we generate a Boolean formula whose models corresponds to each and every pre-condition that is contained in ϕ_c and generate all maximal models of the boolean formula.

Let $\phi \subseteq \phi_c$. If $amgu_{x=t}(\phi) \subseteq \psi$ then $I \notin amgu_{x=t}(\phi)$ for each I in the set $amgu_{x=t}(\phi_c) \setminus \psi$ since $amgu_{x=t}$ is monotonic. In the other direction, if $I \notin amgu_{x=t}(\phi)$ for each I in the set $amgu_{x=t}(\phi_c) \setminus \psi$ then $amgu_{x=t}(\phi) \subseteq \psi$, for otherwise, there is a contradiction as follows. That $amgu_{x=t}(\phi) \not\subseteq \psi$ implies that there is a group I' such that $I' \in amgu_{x=t}(\phi)$ and $I' \notin \psi$. By the monotonicity of $amgu_{x=t}$, $I' \in amgu_{x=t}(\phi_c)$ which together with $I' \notin \psi$ implies $I' \notin amgu_{x=t}(\phi)$. Therefore, a sharing ϕ satisfying $\phi \subseteq \phi_c$ is pre-condition for $x = t$ and ψ iff $I \notin amgu_{x=t}(\phi)$ for all $I \in amgu_{x=t}(\phi_c) \setminus \psi$. The groups in $amgu_{x=t}(\phi_c) \setminus \psi$ are used to generate a Boolean formula whose maximal models corresponds to maximal pre-conditions that are relevant to both x and t. Each group I in $amgu_{x=t}(\phi_c) \setminus \psi$ is the union of one group G in $rel(x, \phi_c)^*$ and another group H in $rel(t, \phi_c)^*$. Let

$$\Pi(I) = \left\{ (G,H) \;\middle|\; \begin{array}{c} (G \in rel(x,\phi_c)^*) \\ \wedge \\ (H \in rel(t,\phi_c)^*) \wedge (I = G \cup H) \\ \wedge \\ (G \subset I) \wedge (H \subset I) \end{array} \right\}$$

Example 10. Continue with example 9.

$$amgu_{x=t}(\phi_c) = \left\{ \begin{array}{c} \emptyset, \{x_5\}, \{x_2,x_4\}, \{x_1,x_2,x_4\}, \\ \{x_2,x_3,x_4\}, \{x_1,x_2,x_3,x_4\} \end{array} \right\}$$

$$amgu_{x=t}(\phi_c) \setminus \psi = \{\{x_1,x_2,x_4\}\}$$

$$\Pi(\{x_1,x_2,x_4\}) = \left\{ \begin{array}{c} (\{x_2\},\{x_1,x_4\}), \\ (\{x_1,x_2\},\{x_4\}), \\ (\{x_1,x_2\},\{x_1,x_4\}), \\ (\{x_1,x_2\},\{x_2,x_4\}), \\ (\{x_2,x_4\},\{x_1,x_4\}) \end{array} \right\}$$

∎

The lemma below states that the following condition is both sufficient and necessary for any subset ϕ of ϕ_c to be a pre-condition for $x = t$ and ψ.

$$\bigwedge_{I \in amgu_{x=t}(\phi_c) \setminus \psi} \left(\bigwedge_{(G,H) \in \Pi(I)} \left(\begin{array}{c} G \notin rel(x,\phi)^* \\ \vee \\ H \notin rel(t,\phi)^* \end{array} \right) \right) \quad (4)$$

Lemma 5. Let $\phi \subseteq \phi_c$. Then $amgu_{x=t}(\phi) \subseteq \psi$ iff condition (4) holds.

Proof. Since $amgu_{x=t}$ is monotonic and $\phi \subseteq \phi_c$, we have $amgu_{x=t}(\phi) \subseteq amgu_{x=t}(\phi_c)$. If condition (4) holds, $I \notin amgu_{x=t}(\phi)$ for any $I \in amgu_{x=t}(\phi_c) \setminus \psi$ and hence $(amgu_{x=t}(\phi_c) \setminus \psi) \cap amgu_{x=t}(\phi) = \emptyset$, implying $amgu_{x=t}(\phi) \subseteq \psi$. In the other direction, if $amgu_{x=t}(\phi) \subseteq \psi$ then

$$(amgu_{x=t}(\phi_c) \setminus \psi) \cap amgu_{x=t}(\phi) = \emptyset$$

Thus, for each $I \in amgu_{x=t}(\phi_c) \setminus \psi$, $I \notin amgu_{x=t}(\phi)$. So condition (4) is true. □

Example 11. Continue with example 10. The condition (4) is:

$$\left(\begin{array}{c} (\{x_2\} \notin rel(x_2,\phi)^* \vee \{x_1,x_4\} \notin rel(x_4,\phi)^*) \\ \wedge \\ (\{x_1,x_2\} \notin rel(x_2,\phi)^* \vee \{x_4\} \notin rel(x_4,\phi)^*) \\ \wedge \\ (\{x_1,x_2\} \notin rel(x_2,\phi)^* \vee \{x_1,x_4\} \notin rel(x_4,\phi)^*) \\ \wedge \\ (\{x_1,x_2\} \notin rel(x_2,\phi)^* \vee \{x_2,x_4\} \notin rel(x_4,\phi)^*) \\ \wedge \\ (\{x_2,x_4\} \notin rel(x_2,\phi)^* \vee \{x_1,x_4\} \notin rel(x_4,\phi)^*) \end{array} \right)$$

∎

The condition (4) is not in a form that can be directly used to find maximal pre-conditions for $x = t$ and ψ that are relevant to both x and t. It has subformulas of the form $H \notin rel(t,\phi)^*$ in which H is not necessary in ϕ_c. We transform each subformula of the form $H \notin rel(t,\phi)^*$ into a formula that consists of primitive formulas of the form $G \notin \phi$ and logical operators $\vee$ and $\wedge$ where G is in ϕ_c. We first introduce an operation that decomposes a group H relevant to t into pairs of groups (H_1, H_2) such that $H = H_1 \cup H_2$ and both H_1 and H_2 are relevant to t.

$$\cup^{-1}(H,t) = \left\{ (H_1,H_2) \;\middle|\; \begin{array}{c} (H = H_1 \cup H_2) \\ \wedge \\ (H_1 \cap var(t) \neq \emptyset) \\ \wedge \\ (H_2 \cap var(t) \neq \emptyset) \\ \wedge \\ (H_1 \subset H) \wedge (H_2 \subset H) \end{array} \right\}$$

For instance, let H be $\{x_2, x_3\}$ and $t = f(x_2, x_3)$. Then $\cup^{-1}(\{x_2,x_3\}, t) = \{(\{x_2\},\{x_3\})\}$. The following function transforms the formulas of the form $H \notin rel(t,\phi)^*$ where t is either a variable or a term.

$\Gamma(H \notin rel(t,\phi)^*) =$

$$\left\{ \begin{array}{ll} (H \notin \phi) \wedge \wedge_{(H_1,H_2)\in\pi} \sigma(H_1,H_2) & if\ H \in \phi_c \\ \\ \wedge_{(H_1,H_2)\in\pi} \sigma(H_1,H_2) & if\ H \notin \phi_c \end{array} \right.$$

where $\pi = \cup^{-1}(H,t)$, $\sigma(H_1,H_2) = \left(\begin{array}{c} \Gamma(H_1 \notin rel(t,\phi)^*) \\ \vee \\ \Gamma(H_2 \notin rel(t,\phi)^*) \end{array} \right)$

Example 12. Continue with examples 9 and 11.
Let consider $\{x_2,x_4\} \notin rel(x_4,\phi)^*$. Since $\{x_2,x_4\} \in \phi_c$ and $\cup^{-1}(\{x_2,x_4\},x_4) = \emptyset$, So $\Gamma(\{x_2,x_4\} \notin rel(x_4,\phi)^*) = \{x_2,x_4\} \notin \phi$ ∎

We give another example about function Γ.

Example 13. Let $V = \{x_1,x_2,x_3,x_4\}$, $\phi_c = \wp(V)$, $t = f(x_1,x_2)$ and $H = \{x_1,x_2,x_3\}$. We have

$$\cup^{-1}(\{x_1,x_2,x_3\},t) = \left\{ \begin{array}{c} (\{x_1\},\{x_2,x_3\}), \\ (\{x_1,x_2\},\{x_1,x_3\}), \\ (\{x_1,x_2\},\{x_2,x_3\}), \\ (\{x_1,x_3\},\{x_2,x_3\}), \\ (\{x_1,x_3\},\{x_2\}) \end{array} \right\}$$

and

$$\cup^{-1}(\{x_1,x_2\},t) = \{(\{x_1\},\{x_2\})\}$$

Then
$\Gamma(\{x_1,x_2,x_3\} \notin rel(f(x_1,x_2),\phi)^*) =$

$$\left(\begin{array}{c} \{x_1,x_2,x_3\} \notin \phi \\ \wedge \\ (\{x_1\} \notin \phi \vee \{x_2,x_3\} \notin \phi) \\ \wedge \\ (((\{x_1,x_2\} \notin \phi \wedge (\{x_1\} \notin \phi \vee \{x_2\} \notin \phi)) \vee \{x_1,x_3\} \notin \phi) \\ \wedge \\ (((\{x_1,x_2\} \notin \phi \wedge (\{x_1\} \notin \phi \vee \{x_2\} \notin \phi)) \vee \{x_2,x_3\} \notin \phi) \\ \wedge \\ (\{x_1,x_3\} \notin \phi \vee \{x_2,x_3\} \notin \phi) \\ \wedge \\ (\{x_1,x_3\} \notin \phi \vee \{x_2\} \notin \phi) \end{array}\right)$$

■

Lemma 6. Let $\phi \subseteq \phi_c$ and $H \cap var(t) \neq \emptyset$. Then $H \notin rel(t,\phi)^*$ is true iff $\Gamma(H \notin rel(t,\phi)^*)$ is.

PROOF. The proof is done by induction on the number of elements in H.

Basis. H contains only one element. There are two cases to consider: (i) $H \in \phi_c$ or (ii) $H \notin \phi_c$. Consider case (i). That $H \notin rel(t,\phi)^*$ is equivalent to that $H \notin \phi$ since $H \cap var(t) \neq \emptyset$. We have that $\cup^{-1}(H,t) = \emptyset$ and $\Gamma(H \notin rel(t,\phi)^*)$ reduces to $H \notin \phi$. So, the lemma holds in the case (i). Similarly, it can be verified that the lemma holds in the case (ii).

Now assume that H contains n elements and that $H' \notin rel(t,\phi)^* \Leftrightarrow \Gamma(H' \notin rel(t,\phi)^*)$ holds for any H' that has less than n elements. There are two cases to consider: (i) $H \in \phi_c$ or (ii) $H \notin \phi_c$. Consider case (i). $H \notin rel(t,\phi)^*$ is true iff that $H \notin rel(t,\phi)$ is true and for all $H_1 \subset H$ and $H_2 \subset H$ that $H = H_1 \cup H_2$, $(H_1 \notin rel(t,\phi)^* \vee H_1 \notin rel(t,\phi)^*)$ holds. Observe that if $H_i \cap var(t) = \emptyset$ then $H_1 \notin rel(t,\phi)^*$ holds. Note that $H \notin rel(t,\phi)$ is equivalent to $H \notin \phi$ since $H \cap var(t) \neq \emptyset$. So, That $H \notin rel(t,\phi)^*$ is true iff $(H \notin \phi) \wedge \wedge_{(H_1,H_2)\in\cup^{-1}(H,t)}(H_1 \notin rel(t,\phi)^* \vee H_2 \notin rel(t,\phi)^*)$ holds. By the induction hypothesis, $H_i \notin rel(t,\phi)^*$ is equivalent to $\Gamma(H_i \notin rel(t,\phi)^*)$ for $i = 1,2$. Therefore, That $H \notin rel(t,\phi)^*$ is true iff $(H \notin \phi) \wedge \wedge_{(H_1,H_2)\in\cup^{-1}(H,t)}(\Gamma(H_1 \notin rel(t,\phi)^*) \vee \Gamma(H_2 \notin rel(t,\phi)^*))$ is true. By the definition of Γ, $H \notin rel(t,\phi)^*$ is true iff $\Gamma(H \notin rel(t,\phi)^*)$ is true. The case (ii) is similar. □

The formula $\Gamma(H \notin rel(t,\phi)^*)$ does not contain any application of the functions $rel(\cdot)$ and $(\cdot)^*$. Lemma 6 allows us to transform condition (4) into

$$\bigwedge_{I\in amgu_{x=t}(\phi_c)\setminus\psi} \left(\bigwedge_{(G,H)\in\Pi(I)} \left(\begin{array}{c} \Gamma(G \notin rel(x,\phi)^*) \\ \vee \\ \Gamma(H \notin rel(t,\phi)^*) \end{array} \right) \right) \quad (5)$$

that consists of logical operators $\vee$ and $\wedge$ and primitive formulas of the form $G \notin \phi$ where G is in ϕ_c. The above condition is then translated into a Boolean formula as follows. Each group G in ϕ_c is encoded by a Boolean variable denoted $B(G)$. Assigning the truth value 1 to $B(G)$ indicates that G is in ϕ; and assigning the truth value 0 to $B(G)$ means that G is not in ϕ. A sharing ϕ corresponds to a truth assignment μ in that ϕ contains a group in ϕ_c iff $\mu(B(G)) = 1$. The sharing that corresponds to a truth assignment μ is denoted $[\mu]$. The condition (5) can be transformed into the following Boolean formula by a mapping β that replaces $G \notin \phi$ with $\overline{B(G)}$ where $\overline{\cdot}$ is the negation operator.

$$\beta\left(\bigwedge_{I\in\phi_1} \left(\bigwedge_{(G,H)\in\Pi(I)} \left(\begin{array}{c} \Gamma(G \notin rel(x,\phi)^*) \\ \vee \\ \Gamma(H \notin rel(t,\phi)^*) \end{array} \right) \right) \right) \quad (6)$$

where $\phi_1 = amgu_{x=t}(\phi_c) \setminus \psi$

Example 14. Continue with example 11. The formula (5) is

$$\left(\begin{array}{c} (\{x_2\} \notin \phi \vee \{x_1,x_4\} \notin \phi) \\ \wedge \\ (\{x_1,x_2\} \notin \phi \vee \{x_4\} \notin \phi) \\ \wedge \\ (\{x_1,x_2\} \notin \phi \vee \{x_1,x_4\} \notin \phi) \\ \wedge \\ (\{x_1,x_2\} \notin \phi \vee \{x_2,x_4\} \notin \phi) \\ \wedge \\ (\{x_2,x_4\} \notin \phi \vee \{x_1,x_4\} \notin \phi) \end{array}\right)$$

and the formula (6) is

$$\left(\begin{array}{c} (\overline{B(\{x_2\})} \vee \overline{B(\{x_1,x_4\})}) \\ \wedge \\ (\overline{B(\{x_1,x_2\})} \vee \overline{B(\{x_4\})}) \\ \wedge \\ (\overline{B(\{x_1,x_2\})} \vee \overline{B(\{x_1,x_4\})}) \\ \wedge \\ (\overline{B(\{x_1,x_2\})} \vee \overline{B(\{x_2,x_4\})}) \\ \wedge \\ (\overline{B(\{x_2,x_4\})} \vee \overline{B(\{x_1,x_4\})}) \end{array}\right)$$

■

The Boolean formula (6) is equivalent to the logical formula (5) in that a sharing is a model of (5) iff its corresponding true assignment is a model of (6). Maximal models of (6) corresponds to maximal models of (5).

The following theorem gives a method for computing maximal pre-conditions for $x = t$ and ψ.

Theorem 2. Let $\mathcal{M}$ be the set of maximal models of (6). Then the set of maximal pre-conditions for $x = t$ and ψ is $max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$ where ϕ_x and ϕ_t are defined in theorem 1.

PROOF. Let ϕ be a maximal pre-condition for $x = t$ and ψ. If ϕ is not related to both x and t, by theorem 1, $\phi \subseteq \phi_x$ or $\phi \subseteq \phi_t$. Assume that $\phi \subseteq \phi_x$ without loss of generality. Since ϕ is a maximal pre-condition for $x = t$ and ψ, $\phi = \phi_x$ and $\phi_x \in max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$. So, $\phi \in max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$. Otherwise, ϕ is relevant to both x and t. By lemmas 4, 5 and 6, ϕ corresponds to a maximal model of (6) and hence $\phi \in max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$.

In the other direction, assume that $\phi \in max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$. If $\phi = \phi_{si}$ for any $i = 1,2$ then by the definition of max and theorem 1, ϕ is a maximal pre-condition for $x = t$ and ψ. Otherwise, ϕ corresponds to a maximal model of (6) that is not contained in either ϕ_x or ϕ_t. Then ϕ is relevant to both x and t, for otherwise, ϕ is contained in ϕ_x or ϕ_t by theorem 1. By lemmas 4, 5 and 6, ϕ is a maximal pre-condition for $x = t$ and ψ. □

Example 15. Continue with example 14.
The set of maximal models of formula (6) in this example corresponds to following set of sharings

$$\left\{ \begin{array}{c} \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_4\}, \{x_5\}, \\ \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}, \\ \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\}, \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_4\}, \{x_5\}, \{x_1, x_4\}, \{x_2, x_3\}, \\ \{x_3, x_4\}, \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\} \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_5\}, \{x_1, x_2\}, \{x_2, x_3\}, \\ \{x_3, x_4\}, \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\} \end{array} \right\} \end{array} \right\}$$

How to generate all maximal models of a Boolean formula is well known [15]. The Boolean formula (6) can be put into a purely negative CNF (conjunctive normal form). Finding all maximal models of purely negative CNF can be transferred to the problem of generating all maximal independent sets [14, 25]. Applying an algorithm in [15], we can generate all maximal models for the Boolean formula (6) and obtain all maximal pre-conditions for $x = t$ and ψ using theorem 2.

7. BACKWARD ABSTRACT UNIFICATION

We are now ready to put together the results in previous sections and present the backward abstract unification operator. Given an equation $x = t$ and its post-condition ω, the backward abstract unification function first normalizes the post-condition ω to ψ and then computes the maximal pre-conditions for $x = t$ and ψ applying results in theorems 1 and 2.

Definition 2. (backward abstract unification)

$$amgu^{-1}_{x=t}(\psi) = max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\})$$

where $\psi = \omega \setminus rel(x, \omega) \setminus rel(t, \omega) \cup (rel(x, \omega) \cap rel(t, \omega))$, ω is an arbitrary post-condition, $\mathcal{M}$ is the set of maximal models of (6), ϕ_x and ϕ_t are given in theorem 1.

Example 16. Continue with example 15 and 6.
$max(\{[\mu] \mid \mu \in \mathcal{M}\} \cup \{\phi_x, \phi_t\}) =$

$$\left\{ \begin{array}{c} \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_4\}, \{x_5\}, \\ \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}, \\ \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\}, \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_4\}, \{x_5\}, \{x_1, x_4\}, \{x_2, x_3\}, \\ \{x_3, x_4\}, \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\} \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_5\}, \{x_1, x_2\}, \{x_2, x_3\}, \\ \{x_3, x_4\}, \{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}, \\ \{x_2, x_3, x_4\}, \{x_1, x_2, x_3, x_4\} \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_2\}, \{x_5\}, \{x_1, x_2\}, \{x_2, x_3\}, \{x_2, x_5\}, \\ \{x_1, x_2, x_3\}, \{x_1, x_2, x_5\}, \\ \{x_2, x_3, x_5\}, \{x_1, x_2, x_3, x_5\} \end{array} \right\}, \\ \left\{ \begin{array}{c} \emptyset, \{x_4\}, \{x_5\}, \{x_1, x_4\}, \{x_3, x_4\}, \\ \{x_4, x_5\}, \{x_1, x_3, x_4, \}, \{x_1, x_4, x_5\}, \\ \{x_3, x_4, x_5\}, \{x_1, x_3, x_4, x_5\} \end{array} \right\} \end{array} \right\}$$

∎

8. RELATED WORK

Backward analysis propagates properties against control flow. It infers a pre-condition that satisfies assertions attached to the source code. Forward analysis manipulates the properties in the direction of the control flow. It checks whether those assertions could be violated. Both forward and backward analysis are useful in software verification.

Termination inference [8] proposed by Genaim and Codish generalizes traditional termination analysis. It infers modes for which a logic program is guaranteed to terminate. One of the key components in [8] is the backward groundness analysis proposed in [17].

In [17], King et al. show how a backward analysis can infer the pre-condition that ensure satisfaction of the post-condition. The analysis is composed of a least fixpoint component and a greatest fixpoint component. Least fixpoint component calculates success patterns and greatest fixpoint component uses these success patterns to infer the pre-condition. They use a pseudo-complement operator to infer mode requirement from right to left. The pseudo-complement operator, however, requires the domain to be condensing.

In a type inference presented in [20], the post-condition is a set of type signatures for selected predicates such as built-ins. Then the analysis infers all valid type signatures for other predicates that guarantee satisfaction of the type signatures for the selected type signatures.

Determinacy inference for logic program [22] infers determinacy conditions on a call that ensures it computes at most one answer and that answer is generated only once. One principal component in [22] generates post-condition first and then backward analysis computes pre-conditions.

Hughes and Launchbury [11] shows how to invert a function in abstract domain. Although their object language is functional language, they argue the reversal of a semantic of function should not refer to the concrete semantics. This paper shows this can be done for set-sharing analysis of logic programs.

In [21], a backward pair-sharing analysis is obtained by inverting abstract operators for a corresponding forward pair-sharing analysis. The domain and operators for forward pair-sharing analysis are much simpler than those for forward set-sharing analysis. There are also properties of pair-sharing that are not possessed by set-sharing. For instance, if t is not ground, then for pair-sharing, $amgu_{x=t}(S) \subseteq T$ implies $S \subseteq T$. Thus, a pre-condition is a subset of a given post-condition and the maximal pre-conditions can be generated by removing sharing pairs from the post-condition. The forward set-sharing analysis has a more complex domain and more complex abstract operators; making it difficult to invert.

There have been much research in sharing analysis of logic programs [12, 24, 2, 19, 4, 16, 5, 23]; Set-sharing analysis is one of the well studied analysis for logic programs. Set-sharing captures variable alias and groundness. Rather than designing a backward set-sharing analysis from scratch, this paper shows how an abstract operator for backward set-sharing analysis can be obtained by inverting a corresponding abstract operator for forward set-sharing analysis.

9. CONCLUSIONS

We have presented a backward abstract unification operator for set-sharing by inverting forward abstract unification. The inversion is realized by transforming the problem of finding all maximal pre-conditions into the problem of finding all maximal models of a Boolean formula. The backward abstract unification operator can be plugged into a backward

analysis engine to perform backward set-sharing analysis.

Acknowledgements

This work was supported by the National Science Foundation under grants CCR-0131862 and INT-0327760. We would like to thank Enea Zaffanella and other referees of a previous version of this draft for their insightful comments. We would also like to thank Andy King for helpful discussions on backward analysis.

10. REFERENCES

[1] R. Bagnara, P. M. Hill, and E. Zaffanella. Sharing revisited. In M. Falaschi, M. Navarro, and A. Policriti, editors, *APPIA-GULP-PRODE'97: Proceedings of the 1997 Joint Conference on Declarative Programming*, pages 69–80, Grado, Italy, June 1997.

[2] R. Bagnara, E. Zaffanella, and P. M. Hill. Enhanced sharing analysis techniques: A comprehensive evaluation. *Theory and Practice of Logic Programming*, 5(1&2):1–43, 2005.

[3] Francisco Bueno, Pierre Deransart, Wlodzimierz Drabent, Gérard Ferrand, Manuel V. Hermenegildo, Jan Maluszynski, and German Puebla. On the role of semantic approximations on validation and diagnosis of contraint logic programs. In Mariam Kamkar, editor, *AADEBUG: Proceeding of the 3rd Automated and Algorithmic Debugging*, pages 155–169, 1997.

[4] Michael Codish, Harald Søndergaard, and Peter J. Stuckey. Sharing and groundness dependencies in logic programs. *ACM Transactions on Programming Languages and Systems*, 21(5):948–976, 1999.

[5] Agostino Cortesi and Gilberto Filé. Sharing is optimal. *Journal of Logic Programming*, 38(3):371–386, 1999.

[6] P. Cousot and R. Cousot. Induction principles for proving invariance properties of programs. In D. Néel, editor, *Tools and Notions for Program Construction: an Advanced Course*, pages 75–119. Cambridge University Press, Cambridge, UK, 1982.

[7] Robert W. Floyd. Assigning meaning to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science, volume 19 of Proceedings of Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, 1967.

[8] Samir Genaim and Michael Codish. Inferring termination conditions for logic programs using backwards analysis. *Theory and Practice of Logic Programming*, 5(1&2):75–91, 2005.

[9] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[10] Jacob M. Howe, Andy King, and Lunjin Lu. Analysing logic programs by reasoning backwards. In Maurice Bruynooghe and Kung-Kiu Lau, editors, *Program Development in Computational Logic*, volume 3049 of *Lecture Notes in Computer Science*, pages 152–188. Springer, 2004.

[11] John Hughes and John Launchbury. Reversing abstract interpretations. *Science of Computer Programming*, 22(3):307–326, 1994.

[12] Dean Jacobs and Anno Langen. Accurate and efficient approximation of variable aliasing in logic programs. In Ross A. Overbeek Ewing L. Lusk, editor, *NACLP: Proceedings of the North American Conference on Logic Programming*, pages 154–165. MIT Press, 1989.

[13] Dean Jacobs and Anno Langen. Static analysis of logic programs for independent and-parallelism. *Journal of Logic Programming*, 13(2&3):291–314, 1992.

[14] David S. Johnson and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[15] Dimitris J. Kavvadias, Martha Sideri, and Elias C. Stavropoulos. Generating all maximal models of a boolean expression. *Information Processing Letters*, 74(3-4):157–162, 2000.

[16] Andy King. Pair-sharing over rational trees. *Journal of Logic Programming*, 46(1-2):139–155, 2000.

[17] Andy King and Lunjin Lu. A backward analysis for constraint logic programs. *Theory and Practice of Logic Programming*, 2(4-5):517–547, 2002.

[18] Andy King and Lunjin Lu. Forward versus backward verification of logic programs. In Catuscia Palamidessi, editor, *Proceedings of 19th International Conference on Logic Programming*, volume 2916 of *Lecture Notes in Computer Science*, pages 315–330. Springer, 2003.

[19] Vitaly Lagoon and Peter J. Stuckey. Precise pair-sharing analysis of logic programs. In *PPDP '02: Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 99–108. ACM Press, 2002.

[20] Lunjin Lu and Andy King. Backward type inference generalises type checking. In Manuel V. Hermenegildo and German Puebla, editors, *Proceedings of 9th Static Analysis Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 85–101. Springer, 2002.

[21] Lunjin Lu and Andy King. Backward pair sharing analysis. In Yukiyoshi Kameyama and Peter J. Stuckey, editors, *Proceedings of 7th Fuji International Symposium on Functional and Logic Programming*, volume 2998 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2004.

[22] Lunjin Lu and Andy King. Determinacy Inference for Logic Programs. In Mooly Sagiv, editor, *Proceedings of the 14th European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 108–123. Springer-Verlag, 2005.

[23] Kalyan Muthukumar and Manuel V. Hermenegildo. Combined determination of sharing and freeness of program variables through abstract interpretation. In Koichi Furukawa, editor, *Proceedings of the 8th International Conference on Logic Programming*, pages 49–63, 1991.

[24] Harald Søndergaard. An application of abstract interpretation of logic programs: Occur check reduction. In Bernard Robinet and Reinhard Wilhelm, editors, *Proceedings of the 1st European Symposium on Programming*, volume 213 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 1986.

[25] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.