Joint NCC & IIS Bull., Comp.Science, 5 (1996), 43–53
© 1996 NCC Publisher

Simple semantic analysis problems for functional programs*

V.K. Sabelfeld, A. Sabelfeld

In the paper two problems of semantic property analysis of recursion schemes are stated and a marking technique for solving these problems is described.

Keywords: *program flow analysis, abstract interpretation, approximation, recursion scheme, strictness analysis, parameter dependence.*

1. Introduction

To check applicability conditions of equivalent program transformations, to detect semantic errors in programs and to verify programs, one have to learn how to determine some semantic properties of programs. Many authors explored program analysis problems on program models. The most significant results were obtained in flow analysis [3] and abstract interpretation [1, 2] approaches. We consider functional, or recursive, programs and investigate the problem of semantic property analysis for a model of functional programs called recursion schemes.

We describe two simple classes of semantic property analysis problems *(forward and backward property analysis problems)* for recursion schemes and use a marking technique for their solution. Our approach is very close to Patrick and Radhia Cousot's abstract interpretation approach [1, 2]. We consider the fixpoint semantics of recursive programs. Namely, we associate a term sequence called *approximation sequence* with a recursion scheme. Given the properties of terms from the approximation sequence we define the program property as the limit of the constructed sequence of properties.

Both forward and backward property analysis problems formulated are proved to be undecidable in the general case. We give sufficient conditions for solving these problems and present marking algorithms that compute program properties under the conditions required. In Sections 5.1 and 5.2 we give examples of forward (*Essentiality of term occurrences*) and backward property analysis problems (*Hopelessness of term occurrences*).

*Partially supported by the Russian Foundation of Fundamental Research under Grants 93-012-576 and by Russian Committee on Higher Education under Grants "Formal methods for program analysis and transformation".

Some state property analysis of recursive programs well described in the flow analysis or abstract interpretation approaches cannot be formulated in a natural way as either forward or backward analysis problems. As an instance, in Section 5.3 we describe the formal parameter dependence analysis of recursion schemes as a flow analysis problem.

2. Recursion scheme definition

Let $\mathcal{X} = \{x, y, z, \ldots\}$ be a set of *variables*, $\mathcal{F}_b = \{\omega, f, g, h, \ldots\}$ be a set of *basic symbols*, $\mathcal{F}_d = \{F, F_1, F_2, \ldots\}$ be a set of *defined symbols*, $\mathcal{F} = \mathcal{F}_b \bigcup \mathcal{F}_d$, and r denote the *rank* function. Every symbol $s \in \mathcal{F}$ of rank $r(s) = n$ is said to have arity n, $r(\omega) = 0$.

For a set X of variables, $X \subset \mathcal{X}$, let $\mathcal{T}(X)$ be the set of all *terms* under variables from X and basic and defined symbols from $\mathcal{F}$.

A *recursion scheme* is a pair $S = \langle e; DEF \rangle$, where e is a term called the *scheme entry* and DEF is a finite set of definitions of symbols in $\mathcal{F}_d$. A *definition* of a symbol $F \in \mathcal{F}_d$ has the form $F(x_1, \ldots, x_n) \Leftarrow t$, where $n = r(F), x_1, \ldots, x_n \in \mathcal{X}$ are different *formal parameters* of F and $t \in \mathcal{T}(x_1, \ldots, x_n)$ is the *body* of the symbol F. We assume each formal parameter being disjoint from all variables appearing in the entry.

A term $F(t_1, \ldots, t_n)$, where $n = r(F)$ and $F \in \mathcal{F}_d$, is said to be a *call* to the symbol F and its subterms $t_1, \ldots, t_n$ are *actual parameters* of this call. We assume each symbol called in a scheme being defined in it. Here is an example of a scheme:

$$S_0 = \left\langle F_1(h,h) \; ; \; \begin{array}{ll} F_1(x,y) & \Leftarrow \mathbf{if}(px, F_2(fx, fy), x) \\ F_2(x,y) & \Leftarrow \mathbf{if}(py, F_1(fy, fx), y) \\ F_3(x) & \Leftarrow \mathbf{if}(px, F_3(fx), x) \end{array} \right\rangle$$

Any occurrence of a subterm in a term or in a scheme can be uniquely identified by its *address* which symbolizes the path from the root of the tree representing the term (or from the scheme entry) to the place of the subterm occurrence. The address of the tree root (or scheme entry) is the empty word Λ (if the tree is unique in the context under consideration), or the entry number in square brackets. The address of the body of a symbol F is the word $[F]$. If an occurrence of a term $f(t_1, \ldots, t_n)$ has an address a, then its subterm occurrences $t_1, \ldots, t_n$ have the addresses $a.1, \ldots, a.n$ respectively. Two addresses are *independent* iff neither is a prefix of the other.

A *substitution* is an arbitrary map $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{X})$ satisfying the condition $\sigma x \neq x$ for only finite number of variables x from $\mathcal{X}$. The substitution σ replacing a variable x_i with a term t_i for $i = 1, \ldots, n$ is denoted by $[t_1/x_1, \ldots, t_n/x_n]$. The notion of a substitution can be extended in a natural

way to arbitrary terms: $\sigma f(t_1,\ldots,t_n) = f(\sigma t_1,\ldots,\sigma t_n)$ for $f \in \mathcal{F}$ and $n = r(f)$. We denote by $t[a \leftarrow \tau]$ the term obtained from the term t by replacing the term occurrence at the address a in t by the term τ. If N is a set of mutually independent addresses of subterm occurrences of a term t, then $t[N \leftarrow \tau]$ denotes the term obtained from the term t by the replacement of all subterm occurrences at addresses from N by the term τ.

Each scheme S defines a map $\pi : \mathcal{T}(\mathcal{X}) \to \mathcal{T}(\mathcal{X})$ which corresponds to the "parallel outermost computation rule":

$$\pi t = \begin{cases} x, & \text{if } t = x, \text{ where } x \in \mathcal{X}, \\ f(\pi t_1,\ldots,\pi t_n), & \text{if } t = f(t_1,\ldots,t_n), \text{ where } n = r(f) \text{ and } f \in \mathcal{F}_b, \\ \sigma\tau, & \text{if } t = F(t_1,\ldots,t_n), \text{ where } F \in \mathcal{F}_d, \\ \cdot & \sigma = [t_1/x_1,\ldots,t_n/x_n], \\ & \text{and } F(x_1,\ldots,x_n) \Leftarrow \tau \text{ is the definition of } F \text{ in } S. \end{cases}$$

3. Interpretation and equivalence

A *domain* is a triple $\langle D, \leq, \perp\rangle$, where D is an arbitrary set, $\perp \in D$ (the "undefined value") and $\leq$ is a partial order on D satisfying the following two conditions:

- $\forall\, d \in D \;\; \perp \leq d$.
- *Completeness condition:* for every chain C in D the *least upper bound* $lub(C)$ belongs to D, i.e. $d \leq lub(C)$ for all d in C, and $lub(C) \leq u$ for all u in D such that $d \leq u$ for all d in C.

The *universal term domain* $\mathcal{U} = \langle T_U, \sqsubseteq, \omega\rangle$ is constructed in the following way. Consider the set T of all terms under the signature $\langle \mathcal{X}, \mathcal{F}_b\rangle$ with the partial order $\sqsubseteq$ introduced by setting $t_1 \sqsubseteq t_2$ iff there exists a set N of mutually independent addresses in t_2, such that $t_1 = t_2[N \leftarrow \omega]$. Let T_U be the completion of the set T by the least upper bounds of all infinite increasing chains of elements from T. The elements from T_U can be thought as (infinite) terms (trees). One can prove that $\mathcal{U}$ is a domain with the partial order $\sqsubseteq$: $t_1 \sqsubseteq t_2$ iff there is an (infinite) set N of mutually independent addresses in t_2 such that $t_1 = t_2[N \leftarrow \omega]$, and bottom ω.

A function $\varphi : D \to D'$ between two domains is *monotone*, iff $\forall\, d, d' \in D$, $d \leq_D d' \Rightarrow \varphi d \leq_{D'} \varphi d'$. A monotone function is *continuous* if it preserves the least upper bounds of non-empty linearly ordered subsets of D, i.e. $\varphi(lub(C)) = lub(\varphi(C))$.

An *interpretation* I fixes a domain D and assigns

- a member $I(x) \in D$ to each variable $x \in \mathcal{X}$,
- a continuous function $I(f) : D^{r(f)} \to D$ to each symbol $f \in \mathcal{F}_b$.

Using the structural induction we can extend the notion of interpretation I on arbitrary finite terms:

$$I(t) = \begin{cases} I(x), & \text{if } t = x, \text{ where } x \in \mathcal{X}; \\ I(f)(I(t_1),\ldots,I(t_n)), & \text{if } t = f(t_1,\ldots,t_n), \text{ where } f \in \mathcal{F}_b; \\ \bot & \text{otherwise.} \end{cases}$$

An important example of an interpretation is the *universal interpretation* J with the domain $\mathcal{U}$, and $J(x) = x$ for variables $x \in \mathcal{X}$; for $f \in \mathcal{F}_b, n = r(f)$ and $t_1,\ldots,t_n \in T_U$ we set

$$J(f)(t_1,\ldots,t_n) = \begin{cases} f(t_1,\ldots,t_n), & \text{if } f \in \mathcal{F}_b, \\ \omega & \text{otherwise.} \end{cases}$$

Let $\{x_1,\ldots,x_k\}$ be the variable set of a term t and let $y_1,\ldots,y_k$ be pairwise distinct variables not appearing in the scheme S. We define the *approximation sequence* of a term t as

$$App(S,t) = \{J([x_1/y_1,\ldots,x_k/y_k]\pi^n[y_1/x_1,\ldots,y_k/x_k]t) | n \geq 0\},$$

where $\pi^0\tau = \tau$ and $\pi^n = \pi^{n-1}\pi$ for $n > 0$. The *determinant* $det(t)$ *of a term* t is the least upper bound of the approximation sequence $App(S,t) = App(S,t)[0]App(S,t)[1]\ldots$, and the *determinant* $det(S)$ *of a scheme* S is the determinant of the entry of the scheme S. For example, the first elements of the approximation sequence of the entry of a scheme $\langle F(u); F(x) \Leftarrow f(x, F(gx))\rangle$ are

$$\omega \sqsubseteq f(u,\omega) \sqsubseteq f(u, f(gu,\omega)) \sqsubseteq f(u, f(gu, f(ggu,\omega))) \sqsubseteq \ldots, \text{ etc.}$$

Two schemes S_1 and S_2 are *equivalent* (in short: $S_1 \sim S_2$) iff $I(t_1) = I(t_2)$ for the entries t_1, t_2 of these schemes and for all interpretations I.

Let us denote by S/a the term at address a in a scheme or a term S, and we denote by $S[a \leftarrow t]$ the scheme obtained from S by replacing the term at address a by a term t.

4. Two problems of semantic analysis

Let $\mathcal{L} = \langle L, \sqcup, \bot\rangle$ be an upper semilattice where $\bot$ is the *bottom* element and $\sqcup$ is a binary operation called *union* satisfying $\forall x,y,z \in L\ x \sqcup y = y \sqcup x, (x \sqcup y) \sqcup z = x \sqcup (y \sqcup z), x \sqcup x = x, x \sqcup \bot = x$. A partial order on L is defined by $x \sqsubseteq y \overset{dfn}{\equiv} x \sqcup y = y$ and $x \sqsubset y \overset{dfn}{\equiv} (x \sqsubseteq y)\ \&\ (x \neq y)$. We will consider only semilattices satisfying the increasing chain condition.

The *image set* $Image(a)$ of address a of a scheme S is defined as such a minimal subset of addresses of subterms occurring in $det(S)$ which satisfies the following conditions:

- $[\Lambda] \in Image([\Lambda])$.
- If $b \in Image(a), S/a = f(t_1,\ldots,t_n)$ and $f \in \mathcal{F}_b$, then $b.i \in Image(a.i)$ for all $i = 1,\ldots,n$.

– If $b \in Image(a)$ and $S/a = F(t_1, \ldots, t_n)$ for a symbol F defined in S, then $b \in Image([F])$.

– If $Image(a) \neq \emptyset, S/a = F(t_1, \ldots, t_n)$ for a symbol F defined in S, c is the address of the occurrence of the i-th formal parameter in the body of F, and $b \in Image(c)$, then $b \in Image(a.i)$.

We have an instance of a *forward semantic analysis problem* for a recursion scheme S, if the following is given:

– A property semilattice $\mathcal{L}$ satisfying the chain condition.

– An initial property $h_0 \in L$.

– A *semantic heritage function* $Sem^{\downarrow}$ assigning a monotone property transformer $Sem^{\downarrow}(f, i) : L \to L$ to each symbol $f \in \mathcal{F}_b$, and natural i $(1 \leq i \leq r(f))$ which defines the property of the i-th subterm of a term $f(\ldots)$ with given property $s \in L$ as $Sem^{\downarrow}(f, i)(s)$.

The property $h^{\downarrow}(a)$ of an address a in $det(S)$ is defined by

$$h^{\downarrow}(a) = \begin{cases} h_0, & \text{if } a = \Lambda, \\ Sem^{\downarrow}(f, i)(h^{\downarrow}(b)), & \text{if } f \in \mathcal{F}_b \ \& \ \exists i \ a = b.i \ \& \ t/b = f(t_1, \ldots, t_n). \end{cases}$$

The forward semantic analysis problem for a scheme S consists in finding the property

$$H^{\downarrow}(a) = \bigsqcup_{b \in Image(a)} h^{\downarrow}(b)$$

for addresses a in S.

We have an instance of a *backward semantic analysis problem* for a recursion scheme S, if the following is given:

– A property semilattice $\mathcal{L}$ satisfying the chain condition.

– A *deductive semantic function* $Sem^{\uparrow}$ assigning a monotone property transformer $Sem^{\uparrow}(f) : L^n \to L, n = r(f)$ to each symbol $f \in \mathcal{F}_b$ which defines the property of a term $f(t_1, \ldots, t_n)$ using the properties $s_1, \ldots, s_n$ of its subterms $t_1, \ldots, t_n \in L$ as $Sem^{\uparrow}(f)(s_1, \ldots, s_n)$.

For an address a in a scheme S, $Var(a)$ denotes the formal parameter set of a symbol F, if a is the address of a subterm of the body of F, or the set of all variables occurring in the scheme entry, if a is the address of a subterm of the entry. We denote by Pr_i the *projection function* $Pr_i \in (L^n \to L), Pr_i(s_1, \ldots, s_n) = s_i$. The deductive semantic function can be extended on arbitrary terms t without calls by setting

$$Sem^*(t) = \begin{cases} Pr_i, & \text{if } t = x_i, \\ Sem^{\uparrow}(f)(Sem^*(t_1), \ldots, Sem^*(t_n)), & \text{if } f \in \mathcal{F}_b \ \& \ t = f(t_1, \ldots, t_n). \end{cases}$$

For an address a in a scheme S we define $d(a, n) = Sem^*(App(S, S/a)_n)$ as an element of the semilattice $L^{|Var(a)|} \to L$.

The backward semantic analysis problem for the scheme S consists in finding the property

$$D(a) = \bigsqcup_{n=0}^{\infty} d(a,n)$$

for addresses a of the scheme S.

Theorem 1. *Both semantic analysis problems described are undecidable in the general case.*

Theorem 2. *If all property transformers are distributive, the forward analysis problem can be solved using the following marking algorithm.*

An *h-marking* of a scheme S is an arbitrary map μ assigning a property (called *h-mark*) $\mu a \in L$ to each address a in S. The initial h-marking μ_0 assigns the h-mark h_0 to the entry address and the h-mark $\perp$ to all the other addresses in S. The initial h-marking and all h-markings which can be obtained from it by application of *h-marking rule* are called *reachable*. The application of the h-marking rule to an address a consists in the following:

1. If $S/a = f(t_1,\ldots,t_n)$ where $f \in \mathcal{F}_b$, then for all i $(1 \le i \le n)$ the h-mark $\mu a.i$ of the address $a.i$ is replaced by the h-mark $\mu a.i \sqcup Sem^{\downarrow}(f,i)(\mu a)$.
2. If $S/a = F(t_1,\ldots,t_n)$, where $F \in \mathcal{F}_d$, then the h-mark $\mu[F]$ of the address $[F]$ is replaced by the h-mark $\mu[F] \sqcup \mu a$.
3. If a is an address of some occurrence of the i-th formal parameter of F, then for each address b of a call to F satisfying $\mu b \neq \perp$, the h-mark $\mu b.i$ of the address $b.i$ is replaced by the h-mark $\mu b.i \sqcup \mu a$.

A *stationary h-marking* is a reachable h-marking not changed by any application of the h-marking rule. This stationary marking gives the precise solution of the forward analysis problem.

Theorem 3. *If the semilattice L is finite and all property transformers are distributive in all arguments, the backward analysis problem can be solved using the following marking algorithm.*

A *d-marking* of a scheme S is an arbitrary map μ assigning a map (*d-mark*) $\mu a \in (L^n \to L)$, $n = |Var(a)|$ to each address a in S. The initial d-marking μ_0 assigns the d-mark Pr_i to all the addresses of occurences of variable x_i and the d-mark $\perp\!\!\!\perp$ $(\perp\!\!\!\perp(x_1,\ldots,x_n) = \perp)$ to all the other addresses in S. The initial d-marking and all d-markings which can be obtained from it by application of *d-marking rule* are called *reachable*. The application of the d-marking rule to an address a consists in the following:

Let $S/a = f(t_1,\ldots,t_n)$ where $f \in \mathcal{F}, n = r(f) \ge 0$. We replace the d-mark μa of the address a by the d-mark $\mu a \sqcup Q$, where the map Q is defined by

$$Q = \begin{cases} Sem^{\uparrow}(f)(\mu a.1,\ldots,\mu a.n), & \text{if } f \in \mathcal{F}_b, \\ \mu[f](\mu a.1,\ldots,\mu a.n), & \text{if } f \in \mathcal{F}_d. \end{cases}$$

A *stationary d-marking* is a reachable d-marking not changed by any application of the d-marking rule. This stationary marking gives the precise solution of the backward analysis problem.

5. Examples of forward and backward analysis problems

5.1. Unessential addresses

Let us call an address a in a scheme S *unessential* iff $Image(a) = \emptyset$. For an unessential address a, $S \sim S[a \leftarrow t]$ holds for arbitrary terms $t \in T(Var(a))$. We can formulate forward analysis problem for finding all unessential addresses of a scheme S.

The property semilattice: $L = \{\mathbf{O}, \mathbb{1}\}, \mathbf{O}$ for "unessential" and $\mathbb{1}$ for "essential", $p \sqcap p' = \mathbb{1} \Leftrightarrow p = \mathbb{1}$ & $p' = \mathbb{1}$.
Initial property: $\mathbb{1}$.
The semantic heritage function $Sem^{\downarrow}$ assigns the distributive property transformer $Sem^{\downarrow}(f, i) = id_L$ to each basic symbol $f \in \mathcal{F}_b$ and natural i $(1 \leq i \leq r(f))$. Then

$$H^{\downarrow}_{ess}(a) = \bigsqcup_{b \in Image(a)} h^{\downarrow}(b) = \mathbb{1} \Leftrightarrow Image(a) = \emptyset.$$

5.2. Extended strictness analysis: detection of hopeless addresses

This analysis generalizes the Mycroft's strictness analysis [4, 5]. Let us suppose that some subset $Strict(f)$ of the set $2^{\{1,\ldots,r(f)\}}$ is associated with each basic symbol f; we call $Strict(f)$ the set of *strict parameter collections* of the symbol f. An interpretation I *strict*, if the condition

$$(\forall \Delta \in Strict(f)\ \exists i \in \Delta\ d_i = \bot) \Rightarrow I(f)(d_1, \ldots, d_n) = \bot$$

holds for each basic symbol $f \in \mathcal{F}_b$. For example, the natural way to bound the interpretations of a ternary symbol **if** is to settle $Strict(\mathbf{if}) = \{\{1,2\},\{1,3\}\}$. This means that we restrict all possible interpretations of the symbol **if** to functions *cond* for which the condition

$$\forall\, d, d' \in D\ (cond(\bot, d, d') = \bot)\ \&\ (cond(d, \bot, \bot) = \bot)$$

holds. Another example is the vote function Γ^3_2 with the set $Strict(\Gamma^3_2) = \{\{1,2\},\{1,3\},\{2,3\}\}$ of strict parameter collection, so that

$$\forall\, d \in D\ I(\Gamma^3_2)(\bot, \bot, d) = \bot\ \&\ I(\Gamma^3_2)(\bot, d, \bot) = \bot\ \&\ I(\Gamma^3_2)(d, \bot, \bot) = \bot$$

for all strict interpretations I. An address a of a scheme S is called *hopeless*, iff $I(S/a) = \perp$ for all strict interpretations I. Now we formulate the backward analysis problem for finding all hopeless addresses of a scheme.

The property semilattices: $\mathcal{L}_n = \langle 2^{\{1,\ldots,n\}}, \cup, \emptyset \rangle$ for $n \geq 0$, the set of all subsets of the set $\{1,\ldots,n\}$ with the set union $\cup$ and the empty set as the bottom element.

The deductive semantic function $Sem^{\uparrow}_{hope}$ assigns a monotone property transformer

$$Sem^{\uparrow}_{hope}(f)(s_1,\ldots,s_n) = \bigcup_{\Delta \in Strict(f)} \prod_{i \in \Delta} s_i$$

to each symbol $f \in \mathcal{F}_b$. Then the address a of a scheme S is hopeless iff $D_{hope}(a) = \emptyset$.

According to the above-described main algorithm for the solution of the backward analysis problem, we obtain the following procedure.

The initial marking: $\mu_0 a = Pr_i$ for addresses a of the occurrences of a variable x_i, and $\mu_0 a = \perp$ for all the other addresses a in S.

The marking rule: if $S/a = f(t_1,\ldots,t_n)$, where $f \in \mathcal{F}, n = r(f) \geq 0$, then the mark μa of an address a is replaced by the mark $\mu' a$:

$$\mu' a := \mu a \cup \begin{cases} \bigcup\limits_{\Delta \in Strict(f)} \prod\limits_{i \in \Delta} \mu a.i, & \text{if } f \in \mathcal{F}_b, \\ \bigcup\limits_{\Delta \in \mu[f]} \prod\limits_{i \in \Delta} \mu a.i, & \text{if } f \in \mathcal{F}_d. \end{cases}$$

5.3. Formal parameter dependence

Let us describe and solve the analysis problem for formal parameter dependence as a flow analysis problem. As mentioned in Introduction, this problem can be formulated as neither forward nor backward analysis problem since any term in the approximation sequence does not contain formal parameters and, therefore, no formal parameter properties can be taken into account.

We formulate a data flow analysis problem for a graph $Graph(S)$ obtained from a scheme S in the following way. The nodes of this graph will be the entry address and the addresses of the bodies of the defined symbols of the scheme S. The arcs will be the call addresses. We draw an arc a from an address b to an address $[F]$, if a is an address of a call to F occurring in the term at the address b.

Let X be a finite set of variables, $X \subset \mathcal{X}$. We use below the semilattice $\mathcal{L}(X)$ of context free grammars G describing finite languages $L(G)$ of term equalities. These grammars have the terminal set

$$\Sigma = \mathcal{F}_b \cup X \cup \{\equiv, (,), ,\}$$

and rules of the following three forms

1. $S \to x \equiv A$ with nonterminals S, A; S is the initial nonterminal of the grammar; $x \in X$;
2. $A \to x$ with nonterminal A; $x \in X$;
3. $A \to f(A_1, \ldots, A_n)$ with nonterminals $A, A_1, \ldots, A_n$ and terminal $f \in \mathcal{F}_b$.

Such a grammar G can be reduced in linear time to a ***reduced*** grammar $Red(G) = \langle N, \Sigma, S, P \rangle$ satisfying the following conditions:

- $L(G) = L(Red(G))$ is a finite language,
- $\forall A, B \in N\ L(A) \cap L(B) = \emptyset$,
- $\forall A \in N\ L(A) \neq \emptyset$,
- $\forall A \in N\ \exists \alpha, \beta \in \{N \cup \Sigma\}^*\ S \xrightarrow{*} \alpha A \beta$.

The *meet* operation $\sqcap$ on reduced grammars corresponds to the language intersection and can be defined in the following way. Let $G_i = \langle N_i, \Sigma, S_i, P_i \rangle$ for $i = 1,2$ be two reduced grammars. We define $G = \langle N, \Sigma, S, P \rangle$, where $N = N_1 \times N_2$, $S = \langle S_1, S_2 \rangle$, $P = \{S \to x \equiv \langle A_1, A_2 \rangle | (S_i \to x \equiv A_i) \in P_i, i = 1,2\} \cup \{\langle A_1, A_2 \rangle \to f(\langle B_1^1, B_1^2 \rangle, \ldots, \langle B_n^1, B_n^2 \rangle) | (A_i \to f(B_1^i, \ldots, B_n^i)) \in P_i, i = 1,2\} \cup \{\langle A_1, A_2 \rangle \to x | (A_i \to x) \in P_i, i = 1,2, x \in X\}$. Finally, we define $G_1 \sqcap G_2 \stackrel{def}{=} Red(G)$.

Lemma 1. $L(G_1 \sqcap G_2) = L(G_1) \cap L(G_2)$.

We denote by $\mathcal{L}(X)$ the set of all reduced grammars augmented by a new distinguished element $\mathbb{1}$ satisfying $\mathbb{1} \sqcap G = G \sqcap \mathbb{1} = G$. The partial order $\sqsubseteq$ on $\mathcal{L}(X)$ is introduced by the definition $G_1 \sqsubseteq G_2 \stackrel{def}{=} G_1 \sqcap G_2 = G_1$. The grammar $\{S \to x \equiv A_x, A_x \to x \mid x \in X\}$ will be denoted by $\mathbf{O}$, $L(\mathbf{O}) = \{x \equiv x \mid x \in X\}, \forall\ G \in \mathcal{L}(X)\ \mathbf{O} \sqsubseteq G$. We say that a nonterminal A of a grammar *knows* a term t, if $A \xrightarrow{*} t$ holds. For a grammar G and a term t, the grammar $Add(G,t)$ which has a nonterminal A knowing t can be built in the following way.

If the grammar G already has a nonterminal knowing t, or $G = \mathbb{1}$, then $Add(G,t) \stackrel{def}{=} G$. Otherwise, if t is a variable x, then add a new nonterminal A and a rule $A \to x$ to the grammar G; if $t = f(t_1, \ldots, t_n)$, build the grammar $G' = Add(Add(\ldots Add(G, t_1), \ldots), t_n)$ and add a new nonterminal A and a rule $A \to f(A_1, \ldots, A_n)$ to the grammar G', where for $i = 1, \ldots, n$ A_i is the nonterminal in G' which knows the term t_i.

For an address a of a call $F(t_1, \ldots, t_n)$ in a scheme we define the grammar transformer

$$[\![call(a)]\!] : \mathcal{L}(Var(a) \cup Var(\Lambda)) \to \mathcal{L}(Var([F]) \cup Var(\Lambda)).$$

If $G = \mathbb{1}$, then $[\![call(a)]\!]G = G$.

Otherwise, let $G' = Add(\ldots Add(G, t_1)\ldots t_n)$, let A_i be the nonterminal of G' knowing the term t_i, and let B_i be the nonterminal of G' knowing the i-th formal parameter x_i of the symbol $F (i = 1, \ldots, n)$. For all $i = 1, \ldots, n$, if $A_i \neq B_i$, then delete the rules $S \to x_i \equiv B_i$ and $B_i \to x_i$ from the grammar G' and add the new rules $S \to x_i \equiv A_i$ and $A_i \to x_i$. Finally, define $[\![call(a)]\!]G \stackrel{def}{=} Red(G')$.

Lemma 2. *For all addresses a of a scheme, $[\![call(a)]\!]$ is a distributive grammar transformer, i.e.,*

$$[\![call(a)]\!](G_1 \sqcap G_2) = [\![call(a)]\!]G_1 \sqcap [\![call(a)]\!]G_2.$$

Now we formulate a data flow analysis problem for $Graph(S)$ by stating

- an initial marking μ_0 associating the grammar $\mathbf{O}$ with the entry node Λ and the grammar $\mathbb{1}$ with all other nodes of $Graph(S)$,
- a semantic function which associates a distributive grammar transformer $[\![call(a)]\!]$ with any arc a of $Graph(S)$.

Let w be a path in $Graph(S)$, i.e., a sequence of adjacent arcs, and let

$$\Phi_w(G) = \begin{cases} G, & \text{if } w \text{ does not contain any arc,} \\ [\![call(a)]\!]\Phi_{w'}(G), & \text{if } w = w'a, \text{ where } a \text{ is an address of a call.} \end{cases}$$

Our data flow analysis problem consists in finding the 'meet over all path solution'

$$mop([F]) = \prod_{w \in W} \Phi_w(\mathbf{O})$$

for all nodes $[F]$ in $Graph(S)$, where W is the set of all paths in $Graph(S)$ from the entry node to the node $[F]$.

It is well known [3] that a stationary marking μ of $Graph(S)$ can be effectively constructed such that $\mu[F] = mop([F])$ for all nodes $[F]$, and hence, if the condition $(x \equiv t) \in L(\mu[F])$ is true for a node $[F]$ and for a formal parameter x of F, then $\mathcal{G} \sim \mathcal{G}[a \leftarrow t]$ holds for all addresses a of the occurrences of x in the body of the symbol F.

Example: for the stationary marking μ of the scheme

$$\left\langle \begin{array}{l} F(h, h); \\ F(x, y) \Leftarrow \mathbf{if}(px, F(fx, fy), gx) \end{array} \right\rangle$$

$(x \equiv y) \in L(\mu[F])$ holds; it means that the actual parameter values coincide in all calls to F.

References

[1] P. Cousot, R. Cousot, *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, Conference Records of the 4^{th} Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Los Angeles, California, 1977, 238–252.

[2] P. Cousot, R. Cousot, *Abstract interpretation frameworks*, Rapport de Recherche, Ecole Polytechnique, Laboratoire d'Informatique, 1992.

[3] J.B. Kam, J.D. Ullman, *Monotone data flow analysis frameworks*, Acta Informatica, **7**, No. 3, 1977, 305–318.

[4] A. Mycroft, *The theory and practice of transforming call-by-need into call-by-name*, Proceedings of the Fourth International Symposium on Programming, Paris, 22–24 April 1980, Lecture Notes of Computer Science, **83**, 1980, 270–281.

[5] A. Mycroft, *Abstract Interpretation and Optimizing Transformations for Applicative Programs*, Ph.D. Dissertation, CST-15-81, Department of Computer Science, University of Edinburgh, December 1981.