Sharing and Groundness Dependencies in Logic Programs

MICHAEL CODISH, HARALD SØNDERGAARD, and PETER J. STUCKEY
The University of Melbourne

We investigate Jacobs and Langen's Sharing domain, introduced for the analysis of variable sharing in logic programs, and show that it is isomorphic to Marriott and Søndergaard's Pos domain, introduced for the analysis of groundness dependencies. Our key idea is to view the sets of variables in a Sharing domain element as the models of a corresponding Boolean function. This leads to a recasting of sharing analysis in terms of the property of "not being affected by the binding of a *single* variable." Such an "unaffectedness dependency" analysis has close connections with groundness dependency analysis using positive Boolean functions. This new view improves our understanding of sharing analysis, and leads to an elegant expression of its combination with groundness dependency analysis based on the reduced product of Sharing and Pos. It also opens up new avenues for the efficient implementation of sharing analysis, for example using reduced order binary decision diagrams, as well as efficient implementation of the reduced product, using domain factorizations.

Categories and Subject Descriptors: D.1.6 [**Programming Languages**]: Logic Programming; D.3.4 [**Programming Languages**]: Processors—*compilers*; *optimization*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*assertions*; *invariants*; *logics of programs*

General Terms: Languages, Theory

Additional Key Words and Phrases: Abstract interpretation, Boolean functions, dataflow analysis, global analysis, groundness analysis, program analysis, sharing analysis

1. INTRODUCTION

Of the abstract domains used in abstract interpretation of logic programs, two that have received considerable attention are Pos and Sharing. The former, originally introduced by Marriott and Søndergaard [1989] consists of the class of positive Boolean functions [Armstrong et al. 1998; Cortesi et al. 1991; 1996; Marriott and Søndergaard 1993]. The Pos domain is most commonly applied to the analysis of groundness dependencies for logic programs. The Sharing domain, introduced by Jacobs and Langen [1989; 1992] and Langen [1991], consists of sets of sets of variables and is applied to the analysis of possible *set-sharing*, that is, sharing amongst sets of program variables. The two abstract domains seem very different in nature. While Pos is considered a clean, intelligible abstract domain, many theoreticians have grappled with the subtleties of Sharing, and implementors have found that analysis based on Sharing is cumbersome and inefficient.

We show that Sharing and Pos are isomorphic, even though the interpretation of

M. Codish has been on leave from Ben-Gurion University of the Negev. H. Søndergaard and P. Stuckey have been supported by the Australian Research Council.
Authors' addresses: M. Codish, Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel; email: mcodish@cs.bgu.ac.il; H. Søndergaard and P. Stuckey, Department of Computer Science, The University of Melbourne, Parkville Victoria 3052, Australia; email: harald@cs.mu.oz.au, pjs@cs.mu.oz.au.

the Boolean functions (and hence some of the associated abstract operations) differ from one to the other. We show, that, while the two analyses are different and yield different information in general, they are surprisingly similar for many common programs. An aim of this article is to clarify the relation between set-sharing and groundness dependencies.

The insight that emerges is that Sharing gives a "truth-table-like" representation of certain Boolean functions, whereas Pos usually is implemented using much more compact, symbolic representations of Boolean functions, such as reduced order binary decision diagrams (ROBDDs) [Bryant 1992]. These representations also enable simple implementations of the essential abstract operations. One ramification of a "logical" view is, therefore, that set-sharing analysis may become more amenable to efficient implementation, by adapting representations and widening techniques developed for fast Pos-based groundness analyzers. Another is that combined sharing/groundness analysis is better understood and can be implemented more efficiently. We address these points in this article, building on results by Codish and Søndergaard [1998].

The extra complexity of Sharing and its operations can be attributed to the fact that set-sharing is a polyadic property ("it takes more than one to share"). In contrast, Pos is concerned with dependencies related to groundness (or instantiation), a monadic property. The intuition behind our result is to view Jacobs and Langen's set-sharing analysis as "dual negative"—a term that will become clear shortly—to groundness analysis. We effectively recast the variable-sharing problem as one of "unaffectedness dependencies" (or noninstantiation dependencies). We claim that there are both practical and conceptual advantages of viewing the polyadic "sharing" property in terms of dependencies related to the monadic "unaffectedness" property. We demonstrate how this new view leads to interesting results regarding the relation between variable sharing and groundness analyses and in particular to a specification of their reduced product.

A Sharing description is a set of variable sets which always contains the empty set of variables as one of its elements. The key idea applied in this article is to view the sets of variables in an element of the Sharing domain as the models of a Boolean function. Because the elements of the Sharing domain always contain the empty set, we are concerned with the class of Boolean functions which are satisfied by assigning the value *false* to all of the variables. We call this class of functions $\overline{\mathsf{coPos}}$, as these are precisely the set of dual negated Pos functions (or equivalently, the negated dual Pos functions); see Section 3.5. The order isomorphism between Sharing and $\overline{\mathsf{coPos}}$ and, hence, also between Sharing and Pos is immediate and yet intriguing. It enables us to recast sharing analysis in the Pos domain by interpreting Boolean functions as specifying unaffectedness dependencies. Considering the well-known subset Def $\subseteq$ Pos, we show that groundness and sharing analyses using Def coincide exactly, thus extending an important result by Cortesi et al. [1992].

2. A MOTIVATING EXAMPLE

A Sharing set conveys sophisticated information about possible sharing amongst program variables. Consider, for example, the substitution

$$\theta = \{x \mapsto f(u_1, u_2), y \mapsto g(u_2, u_3, u_4), z \mapsto h(u_1, u_2, u_3, u_4)\}.$$

Assume that the variables of interest are $\{w, x, y, z\}$ and that $\{u_1, u_2, u_3, u_4\}$ are other variables. Jacobs and Langen [1989; 1992] and Langen [1991] capture the sharing information present in θ with the description

$$S = \{\emptyset, \{w\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}.$$

Here the presence of, say, $\{x, z\}$ indicates that it is possible to further instantiate x and z *simultaneously, by changing a single variable*, in this case u_1, while at the same time instantiating no other variable of interest. Similarly, it is possible to further instantiate x, y, and z simultaneously, namely by binding the variable u_2 further. Note that w is independent from the other variables: it is not possible to change it simultaneously with any other variable by instantiating a single variable.

This brief introduction to Jacobs and Langen's domain is sufficient to let us study a simple example analysis. Consider the program below, each program point annotated with set-sharing information. The variables of interest are those occurring in the program. The details of the analysis are given by Jacobs and Langen [1992], and we shall later be more precise about these details. In the example, we use upper case for the program variables and lower case for their sharing descriptions.

`append(U,V,W) :-`	ⓐ	$\{\emptyset, \{u\}, \{v\}, \{w\}\}$
`U = [],`	ⓑ	$\{\emptyset, \{v\}, \{w\}\}$
`V = W.`	ⓒ	$\{\emptyset, \{v, w\}\}$
`append(U,V,W) :-`	ⓓ	$\{\emptyset, \{u\}, \{v\}, \{w\}, \{x\}, \{y\}, \{z\}\}$
`U = [X|Y],`	ⓔ	$\{\emptyset, \{v\}, \{w\}, \{z\}, \{u, x\}, \{u, y\}, \{u, x, y\}\}$
`W = [X|Z],`	ⓕ	$\{\emptyset, \{v\}, \{u, y\}, \{w, z\}, \{u, w, z\},$ $\{u, w, x, y\}, \{u, w, x, z\}, \{u, w, x, y, z\}\}$
`append(Y,V,Z).`	ⓖ	$\{\emptyset, \{u, w\}, \{v, w\}, \{u, v, w\}\}$

The annotations are justified as follows. From the beginning, all variables in a clause are assumed to be independent (program points ⓐ and ⓓ). At point ⓑ, `U` has become ground, and so cannot share with any other variable. At point ⓒ, `V` and `W` share, hence the presence of $\{v, w\}$. In fact, `V` and `W` can no longer be instantiated independently, hence the absence of $\{v\}$ and $\{w\}$.

The analysis of the second clause is similar. At point ⓔ the sharing sets $\{u, x\}$, $\{u, y\}$, and $\{u, x, y\}$ have been created because of the unification `U = [X|Y]`. Indeed, from that point it would be possible to change `U` and `X` simultaneously (without any other effects) by further instantiating `X`. Similarly we may instantiate `U` and `Y` simultaneously by further instantiating `Y`. Although for this concrete example there is no possibility of simultaneously changing all three variables `U`, `X` and `Y`, because the analysis does not record the linearity of `U`, the element $\{u, x, y\}$ must be added for safety (imagine if `U = [S|S]` held before the unification). At the last program point, ⓖ, we have projected the result onto the set of variables of interest, namely those that appear in the head of the clause.

Consider now a groundness dependency analysis of the same program using the Pos domain. The annotations that we obtain are

`append(U,V,W) :-`	ⓐ	*true*
`U = [],`	ⓑ	u
`V = W.`	ⓒ	$u \wedge (v \leftrightarrow w)$
`append(U,V,W) :-`	ⓓ	...
`U = [X|Y],`	ⓔ	...
`W = [X|Z],`	ⓕ	$(u \leftrightarrow (x \wedge y)) \wedge (w \leftrightarrow (x \wedge z))$
`append(Y,V,Z).`	ⓖ	$w \leftrightarrow (u \wedge v)$

For example, the annotation at point ⓑ states that `U` is ground at this point, and the annotation at ⓒ says, furthermore, that `V` will become ground when `W` does, and vice versa. For details, see for example Marriott and Søndergaard [1993].

At first sight there is nothing much in common between the two analyses. But consider what happens if we read the sharing annotations as sets of models and ask which Boolean functions possess exactly those sets of models. The following annotations simply rephrase the sharing annotations given previously. For example, using $\overline{e}$ to denote the negation of e, we translate the set of sets $\{\emptyset, \{u\}, \{v\}, \{w\}\}$ to $(\overline{u} \vee \overline{v}) \wedge (\overline{u} \vee \overline{w}) \wedge (\overline{v} \vee \overline{w})$, since that formula is true for truth assignments that make at most one of u, v, and w true, and for no other truth assignments.

`append(U,V,W) :-`	ⓐ	$(\overline{u} \vee \overline{v}) \wedge (\overline{u} \vee \overline{w}) \wedge (\overline{v} \vee \overline{w})$
`    U = [],`	ⓑ	$\overline{u} \wedge (\overline{v} \vee \overline{w})$
`    V = W.`	ⓒ	$\overline{u} \wedge (\overline{v} \leftrightarrow \overline{w})$
`append(U,V,W) :-`	ⓓ	...
`    U = [X|Y],`	ⓔ	...
`    W = [X|Z],`	ⓕ	$(\overline{u} \leftrightarrow (\overline{x} \wedge \overline{y})) \wedge (\overline{w} \leftrightarrow (\overline{x} \wedge \overline{z}))$
`    append(Y,V,Z).`	ⓖ	$\overline{w} \leftrightarrow (\overline{u} \wedge \overline{v})$

The similarity with the groundness annotations is striking. For both clauses, once all variables have been involved in a unification, the resulting Boolean functions are identical, provided we systematically change the sign of variables. It is, therefore, natural to ask what the exact relation between the analyses might be.

Note, for the program point ⓒ, that we can read the sharing annotation as saying that the instantiation of any one variable will not affect `U`, and `V` is unaffected if and only if `W` is. A bit of reflection will convince the reader that this assertion contains exactly the same information as the groundness annotation at ⓒ, which says (1) that `U` is ground and (2) that any further instantiation that grounds `V` will ground `W` too, and vice versa. In Section 5 we prove that the two assertions are equivalent.

However, the assertions obtained at points ⓐ and ⓑ differ from one analysis to the other. Note that the formula $\overline{u} \vee \overline{v}$ can be read "the instantiation of some single variable will leave `U` unaffected or it will leave `V` unaffected, or both." In other words, the two variables are independent. The sharing information at ⓐ is, therefore, that the three variables are independent, whereas the groundness information is vacuous. At ⓑ the two analyses both find that `U` is ground, but the sharing analysis gives the additional information that `V` and `W` are independent.

Will the sharing analysis always give at least as much information as the groundness analysis, for each program point, as the example indicates? The answer is no. Consider the following program where `a` is a constant and `U` and `V` are variables.

`q(U,V) :-`	ⓐ	$\overline{u} \vee \overline{v}$
`    r(U,V),`	ⓑ	$\overline{u} \vee \overline{v}$
`    U = V.`	ⓒ	$\overline{u} \leftrightarrow \overline{v}$
`r(U,a).`	ⓓ	$\overline{v}$
`r(a,V).`	ⓔ	$\overline{u}$

We have shown the result of a sharing analysis. The final result for `q` is that `U` and `V` may share. Namely, one is unaffected by further bindings if and only if the other is. However, `U` and `V` will in fact be independent—no further binding can affect the variables, as they are ground at ⓒ. A groundness analysis will establish this:

`q(U,V) :-`	ⓐ	*true*
`    r(U,V),`	ⓑ	$u \vee v$
`    U = V.`	ⓒ	$u \wedge v$
`r(U,a).`	ⓓ	v
`r(a,V).`	ⓔ	u

A comparison of the analyses' progress for the first clause is illuminating. At point ⓐ, the sharing analysis has the stronger result, that `U` and `V` are independent. At ⓑ, the Boolean functions generated by the two analyses are identical, apart from the signs of the variables. However, as we shall later see, a disjunctive groundness assertion such as $u \vee y$ is stronger than the corresponding disjunctive sharing assertion $\overline{u} \vee \overline{v}$. The former says that at least one of `U` and `V` is ground. The latter says that `U` and `V` are independent. Clearly the groundness assertion implies that the two variables are independent. We can read it as having a stronger modality; namely, `U` and `V` will *remain* independent in the current computation branch, i.e., they are *necessarily* independent, whereas the sharing assertion is that `U` and `V` are independent at this point (but they may later be made to share).

For program point ©, we can capitalize on the extra information provided by the groundness analysis. Unifying `U` and `V` in a state where at least one is ground means that both will come out ground. The sharing analysis is not able to make the stronger, valid inference $\overline{u} \wedge \overline{v}$. This example motivates our interest in combining the two analyses, a goal we pursue in Section 6.

3. PRELIMINARIES AND BACKGROUND

We assume familiarity with the standard definitions and notation for logic programs [Lloyd 1987] and abstract interpretation [Cousot and Cousot 1977; 1992].

We denote the set of all (logic) variables by *Vars* but usually restrict attention to a fixed finite set *VI* of variables of interest. Object language variables are typically chosen from the set $\{u, v, w, x, y, z\}$ (possibly with subscripts), while we use a as an object language constant, $\{f, g, h\}$ for function symbols and $\{s, t\}$ for terms. For a syntactic object O, $vars(O)$ denotes the set of variables in O.

A substitution θ is a mapping from variables to terms which has a finite support. Namely, the set $dom(\theta) = \{v \in \mathit{Vars} \mid \theta(v) \neq v\}$ (referred to as the *domain* of θ) is finite. The *range* of a substitution θ is $range(\theta) = \bigcup_{w \in dom(\theta)} vars(\theta(w))$. The identity substitution is denoted ε. We use Sub to denote the set of (idempotent) substitutions. An (idempotent) substitution $\{v \mapsto t\}$ (with $t \neq v$) is referred to as a *binding*. The set of all bindings is denoted by $Bind$.

3.1 Abstract Domains and Galois Insertions

We briefly recall the essential concepts of abstract interpretation, mainly to fix our notation. The notion of one semantics "correctly approximating" another semantics is made precise with the use of *Galois connections*. A Galois connection is a quadruple $\langle D, C, \gamma, \alpha\rangle$ where D and C are posets, $\gamma : D \to C$ and $\alpha : C \to D$ are monotonic functions, and $\alpha(c) \leq d$ if and only if $c \leq \gamma(d)$ for all $d \in D, c \in C$.

In abstract interpretation, D and C (which typically will be complete lattices) are referred to as the *abstract* and the *concrete* domains, respectively. Loosely speaking, we state that the abstract domain D represents a set of designated elements of C, with the property that each $c \in C$ has a unique best "approximation" in D. The functions γ and α are the *concretization* and *abstraction* functions, respectively.

Approximation is understood as follows. An element $d \in D$ approximates (or describes) $c \in C$ if and only if $c \leq \gamma(d)$. There are standard ways of extending this notion to composite domains, such as Cartesian products and function spaces [Marriott et al. 1994].

If α is surjective (or equivalently, if γ is injective), as is usually the case in abstract interpretation of logic programming languages, then the Galois connection is referred to as a *Galois insertion* (or sometimes as a "Galois surjection" [Cousot and Cousot 1992]).

Any Galois connection induces closure operators $\alpha \circ \gamma$ and $\gamma \circ \alpha$. Namely, both are monotonic and idempotent functions, and the former is reductive ($\alpha(\gamma(d)) \leq d$) and so a lower closure operator, while the latter is extensive ($c \leq \gamma(\alpha(c))$) and so an upper closure operator.

3.2 Set-Sharing a la Jacobs and Langen

Let us be more precise about Sharing. This abstract domain is defined as

$$\mathsf{Sharing} = \big(\{\ S \subseteq \mathcal{P}(Vars) \,\big|\, \emptyset \in S\ \},\ \subseteq\big).$$

We say that a set s of variables *occurs* in a substitution θ *through* the variable v, if s is (exactly) the set of variables (of interest) which are mapped to terms containing v. If s occurs in θ through some variable v then we say that s is a set of variables which *share* under θ. Note, that, if a set s of variables share under a substitution θ then there exists a single variable v such that further instantiation of v will "affect" exactly those variables in s. Namely, for any binding $\mu = \{v \mapsto t\} \in \mathit{Bind}$ the variables in s are exactly those which will change when v is further instantiated:

$$s = \{\ x \,\big|\, \mu(\theta(x)) \neq \theta(x)\ \}$$

Jacobs and Langen [1992] provide the following definition:

$$\begin{aligned}&\mathit{occs} : \mathit{Sub} \times \mathit{VI} \to \mathcal{P}(\mathit{VI})\\ &\mathit{occs}(\theta, v) = \{\ x \in \mathit{VI} \,\big|\, v \in \mathit{vars}(\theta(x))\ \}\end{aligned}$$

The abstraction function $\alpha_{\mathsf{JL}} : 2^{Sub} \to \mathsf{Sharing}$ for set-sharing analysis is then defined as

$$\alpha_{\mathsf{JL}}(\Theta) \;=\; \{\ \mathit{occs}(\theta, v) \,\big|\, \theta \in \Theta,\ v \in \mathit{Vars}\ \}.$$

Observe that a variable $x \in \mathit{VI}$ is absent from each set in $\alpha_{\mathsf{JL}}(\Theta)$ if and only if every $\theta \in \Theta$ maps x to a ground term. Also note that the presence of $\emptyset$ in every element of Sharing is natural: the empty set can be seen as the contribution from the variables in $\mathit{Vars} \setminus \mathit{VI}$. Namely, because both VI and the domain of a substitution θ are always finite, there will always be a variable $v \notin \mathit{VI}$ for which $\theta(v) = v$ and hence also $\mathit{occs}(\theta, v) = \emptyset$. Jacobs and Langen [1989] prove that Sharing enjoys a Galois insertion into the domain of concrete substitutions, and in particular, a concretization function $\gamma_{\mathsf{JL}} : \mathsf{Sharing} \to 2^{Sub}$ exists.

Example 3.1. Let $\mathit{VI} = \{w, x, y, z\}$; recall that ε denotes the identity substitution; and let $\theta_1 = \{x \mapsto f(u_1, u_2), y \mapsto g(u_2, f(u_3, u_4)), z \mapsto h(u_4, u_1, c, g(u_2, u_3), u_4)\}$ and $\theta_2 = \{w \mapsto a, x \mapsto f(y, z)\}$. Then

$$\begin{aligned}\alpha_{\mathsf{JL}}(\{\varepsilon\}) \;&=\; \{\ \emptyset, \{w\}, \{x\}, \{y\}, \{z\}\ \}\\ \alpha_{\mathsf{JL}}(\{\theta_1\}) \;&=\; \{\ \emptyset, \{w\}, \{x, z\}, \{y, z\}, \{x, y, z\}\ \}\\ \alpha_{\mathsf{JL}}(\{\theta_2\}) \;&=\; \{\ \emptyset, \{x, y\}, \{x, z\}\ \}\\ \alpha_{\mathsf{JL}}(\{\theta_1, \theta_2\}) \;&=\; \{\ \emptyset, \{w\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\ \}.\end{aligned}$$

3.3 Boolean Functions

Let $B = \{\mathit{true}, \mathit{false}\}$. A Boolean function (on $\mathit{VI} = \{x_1, \ldots, x_n\}$) is a function $\varphi : B^n \to B$. An *interpretation* $\mu : \mathit{VI} \to B$ is an assignment of truth values to the variables in VI. An interpretation μ is a *model* for φ, denoted $\mu \models \varphi$, if $\varphi(\mu(x_1), \ldots, \mu(x_n)) = \mathit{true}$. We will often write an interpretation as the set of variables which are assigned the value *true*. The set of models of φ is thus viewed as a set of sets of variables defined by

$$[\![\varphi]\!]_{\mathit{VI}} = \{\ \{x \in \mathit{VI} \mid \mu(x) = \mathit{true}\} \,\big|\, \mu \models \varphi\ \}.$$

Much of the time we will omit the subscript VI as it will be clear from the context.

Throughout the article we will switch freely amongst the views of φ as a function, as a formula, and as a set of models $[\![\varphi]\!]$, depending on what seems to aid understanding. Moreover, we may leave out the square brackets, even when we use the set-of-models view, so that notation does not get cluttered, especially in proofs.

The reader should have no problem with the implicit conversions. As is well known, $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$, $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$, and $[\![\varphi]\!] \subseteq [\![\psi]\!]$ if and only if $\varphi \models \psi$.

Of course these conventions may lead to ambiguity, and we shall try to be precise wherever confusion could arise. We take the basic semantic object to be the Boolean *function*. In general, writing a function as a formula or a set of models is ambiguous because an expression such as $x \vee y$ could denote a function of x and y, or a function of x, y, and z, say. Church's lambda notation can be used to disambiguate formulas. However, we prefer not to complicate the notation, instead relying on an understanding that, at any time, there is a set of variables of interest, say, $VI = \{x, y, z\}$ so that a formula φ denotes the Boolean function $\lambda xyz : \varphi$. It is worth noting that our sets-of-models notation, perhaps unexpectedly, will be unambiguous. Observation 4.2 will explain why.

Let φ be a Boolean function on VI. We say that φ is *positive* if $VI \in [\![\varphi]\!]_{VI}$, that is, $\varphi(\mathit{true}, \ldots, \mathit{true}) = \mathit{true}$. The abstract domain Pos consists of the positive Boolean functions ordered by logical consequence.

We say that a Boolean function φ is *down-closed* if $[\![\varphi]\!]$ is closed under intersection. We denote by $\varphi{\downarrow}$ the strongest logical consequence of φ which is down-closed. In other words, $[\![\varphi{\downarrow}]\!] = [\![\varphi]\!]{\downarrow}$ is the smallest set that contains $[\![\varphi]\!]$ and is closed under intersection. It is clear that $\varphi{\downarrow}$ is well defined, as the function *true* is both down-closed and a logical consequence of φ, so $\varphi{\downarrow}$ can be characterized as the conjunction of all down-closed logical consequences of φ.

Example 3.2. Consider $\varphi = (x \to y) \to z$. This function's set of models

$$\{\ \{x\}, \{z\}, \{x, z\}, \{y, z\}, \{x, y, z\}\ \}$$

is not down-closed, since it does not contain $\emptyset = \{x\} \cap \{z\}$. Adding $\emptyset$, the result is down-closed, and we get $\varphi{\downarrow} = y \to z$.

A positive Boolean function is *definite* if it is down-closed. We denote the class of definite Boolean functions by Def. A definite function can be written as a conjunction of clauses of the form $a_0 \leftarrow a_1, \ldots, a_n$. Clearly Def is not closed under disjunction. Armstrong et al. [1998] discuss the abstract operations for Def and show how the join operation for Def ($(\varphi \vee \psi){\downarrow}$) can be calculated while maintaining functions in a canonical clausal form.

Example 3.3. Let $VI = \{w, x, y, z\}$. The function $w \wedge (x \leftrightarrow (y \wedge z))$ is closed under intersection of models and hence is an element of Def:

$$[\![w \wedge (x \leftrightarrow (y \wedge z))]\!] = \{\ \{w\}, \{w, y\}, \{w, z\}, \{w, x, y, z\}\ \}$$

It is not hard to see that the operation $\downarrow$ is an upper closure operator on Pos and that it, thus, induces a Galois insertion between Pos and Def. This Galois insertion is $\langle \mathsf{Def}, \mathsf{Pos}, id, \downarrow \rangle$.

3.4 Groundness Dependencies

In the analysis of groundness dependencies using the domain Pos, a function such as $x \wedge (y \to z)$ is used to describe a program state in which x is definitely bound to a ground term and in which there exists an instantiation dependency such that whenever y becomes bound to a ground term then so does z.

The abstraction and concretization functions $\alpha_{\mathsf{Gr}} : \mathcal{P}(Sub) \to \mathsf{Pos}$ and $\gamma_{\mathsf{Gr}} : \mathsf{Pos} \to \mathcal{P}(Sub)$ for groundness dependencies are defined (in terms of sets of models) as follows. For all substitutions θ, let $\mathit{ground}(\theta)$ be the set of variables of interest mapped to ground terms by θ, that is,

$$\mathit{ground}(\theta) = \{\ v \in VI \mid vars(\theta(v)) = \emptyset\ \}.$$

Then

$$\begin{aligned} \alpha_{\mathsf{Gr}}(\Theta) &= \{\ \mathit{ground}(\mu \circ \theta) \mid \theta \in \Theta, \mu \in \mathit{Sub}\ \} \\ \gamma_{\mathsf{Gr}}(\psi) &= \{\ \theta \mid \alpha_{\mathsf{Gr}}(\{\theta\}) \subseteq \psi\ \}\,. \end{aligned}$$

The underlying intuition is that ψ describes a substitution θ if any set of variables that might become ground by further instantiating θ is a model of ψ. Each model m of ψ represents the possibility to simultaneously ground the variables of m by further instantiation for a substitution represented by ψ. For example, the logical reading of the function $\psi = x \wedge (y \rightarrow z)$ for groundness analysis is: on application of any substitution, x is ground, and if y becomes ground then also z does. The models of ψ are $\{\{x\},\{x,z\},\{x,y,z\}\}$. Notice that a substitution such as $\theta = \{x \mapsto a\}$ is not described by this set of models. Namely, we can further instantiate θ to $\{y \mapsto a\} \circ \theta$, with the effect that x and y are ground while z is not, and $\{x,y\}$ is not a model of ψ. Cortesi et al. [1991] show that α_{Gr} and γ_{Gr} form a Galois insertion.

Example 3.4. Let $VI = \{x,y,z\}$, $\theta_1 = \{x \mapsto a\}$, and $\theta_2 = \{x \mapsto u, y \mapsto a, z \mapsto f(u,u)\}$. Under θ_1 it is possible to simultaneously ground any subset of VI which contains x (x is already ground and, hence, must be present in every model). Hence,

$$\alpha_{\mathsf{Gr}}(\{\theta_1\}) = \{\ \{x\},\{x,y\},\{x,z\},\{x,y,z\}\ \}\,.$$

Any instance of θ_2 will ground y and will affect both x and z in the same way—both will either be ground or nonground. Hence,

$$\alpha_{\mathsf{Gr}}(\{\theta_2\}) = \{\ \{y\},\{x,y,z\}\ \}\,.$$

When abstracting sets of substitutions a model is part of the abstraction if its variables become ground in an instance of one of the substitutions. So, for example,

$$\alpha_{\mathsf{Gr}}(\{\theta_1,\theta_2\}) = \{\ \{x\},\{y\},\{x,y\},\{x,z\},\{x,y,z\}\ \}\,.$$

Note that $\alpha_{\mathsf{Gr}}(\{\theta_1\})$,$\alpha_{\mathsf{Gr}}(\{\theta_2\})$ and $\alpha_{\mathsf{Gr}}(\{\theta_1,\theta_2\})$ correspond to the Boolean functions x, $y \wedge (x \leftrightarrow z)$, and $x \vee (y \wedge (x \leftrightarrow z))$, respectively.

Example 3.5. Let $VI = \{x,y,z\}$, and consider $\psi = (x \vee y) \wedge (x \vee z) \wedge (y \vee z)$. That is,

$$[\![\psi]\!] = \{\ \{x,z\},\{x,y\},\{y,z\},\{x,y,z\}\ \}\,.$$

Then $\{x \mapsto u, y \mapsto a, z \mapsto a\} \in \gamma_{\mathsf{Gr}}(\psi)$, but $\{x \mapsto u_1, y \mapsto u_2, z \mapsto u_3\} \notin \gamma_{\mathsf{Gr}}(\psi)$. This is easily seen, since $\{y,z\} \in [\![\psi]\!]$, while $\emptyset \notin [\![\psi]\!]$.

3.5 Duality and Negation

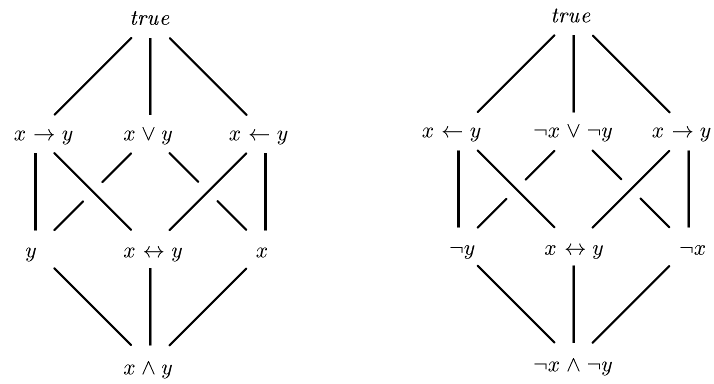
We say that a Boolean function φ is *negative* if and only if $\varphi(\mathit{true},\ldots,\mathit{true}) = \mathit{false}$, that is, if and only if φ is not positive. Let $\overline{\mathsf{Pos}}$ denote the class of negative Boolean functions.

The *dual* of a Boolean function is the function that results when the roles of *false* and *true* are interchanged [Schröder 1877]. Given a propositional formula, a formula for the dual function is easily obtained: interchange, wherever they occur, the constants *false* and *true*, the connectives $\vee$ and $\wedge$, the connectives $\leftrightarrow$ and $\not\leftrightarrow$ (exclusive or), $\rightarrow$ and $\not\leftarrow$, $\leftarrow$ and $\not\rightarrow$. Negation is self-dual, so $\neg$ is left unchanged, as are variables. It follows that existential quantification ($\exists$) and universal quantification ($\forall$) over propositional variables are dual. Table I summarizes the truth tables for some binary propositional connectives and their duals.

For any Boolean function φ we denote by $\mathit{coneg}(\varphi)$ the dual of the negation of φ (or, equivalently, the negation of its dual). For a set of Boolean functions Φ we define $\mathit{coneg}(\Phi) = \{\mathit{coneg}(\varphi) \mid \varphi \in \Phi\}$. It is easy to verify, that, given a formula for φ, a formula for $\mathit{coneg}(\varphi)$ is obtained by replacing each variable occurrence v

Table I. Truth Tables for Some Binary Connectives and Their Duals

x	y	$x \wedge y$	$x \vee y$	$x \leftrightarrow y$	$x \not\leftrightarrow y$	$x \rightarrow y$	$x \not\leftarrow y$	$x \leftarrow y$	$x \not\rightarrow y$
false	*false*	*false*	*false*	*true*	*false*	*true*	*false*	*true*	*false*
false	*true*	*false*	*true*	*false*	*true*	*true*	*true*	*false*	*false*
true	*false*	*false*	*true*	*false*	*true*	*false*	*false*	*true*	*true*
true	*true*	*true*	*true*	*true*	*false*	*true*	*false*	*true*	*false*



a. Pos   b. $\overline{\textsf{coPos}} = \mathit{coneg}(\textsf{Pos})$

Fig. 1. Two classes of Boolean functions for $VI = \{x, y\}$. (a) Pos: the positive functions. (b) $\overline{\textsf{coPos}}$: the dual negated positive functions.

by its negation $\overline{v}$. As an example, $\mathit{coneg}(x \leftrightarrow y) = \overline{x} \leftrightarrow \overline{y} = x \leftrightarrow y$, so $x \leftrightarrow y$ is a fixed point for *coneg*.

We let $\overline{\textsf{coPos}} = \mathit{coneg}(\textsf{Pos})$ denote the dual negative functions. Figure 1 illustrates Pos and $\overline{\textsf{coPos}}$ for $VI = \{x, y\}$. It is easy to see that $\overline{\textsf{coPos}}$ shares many properties with Pos. For example, $\overline{\textsf{coPos}}$ is closed under conjunction, disjunction, and existential quantification, but not under negation. Clearly, *coneg* is an order isomorphism between Pos and $\overline{\textsf{coPos}}$. Observe that *coneg* is its own inverse.

Observation 3.1. In terms of the models of a function φ we have that

$$[\![\mathit{coneg}(\varphi)]\!]_{VI} = \{\, VI \setminus m \mid m \in [\![\varphi]\!] \}.$$

In other words, we have the following *pointwise complementation principle*: to move from $\{m_1, \ldots, m_n\}$ to its dual negated counterpart, replace each element by its complement, to get $\{\overline{m_1}, \ldots, \overline{m_n}\}$.

4. THE LOGIC OF SHARING

It is now easy to see that Sharing and Pos are order isomorphic, as the isomorphism is given by *coneg*. Note that it is natural, and common, to also allow $\emptyset$ as an additional, least, element of Sharing, and to allow *false* as an element of $\overline{\textsf{coPos}}$ and of Pos. In either case the element corresponds to the empty set of substitutions. The isomorphism is easily extended so that these elements correspond.

In the following we shall use Pos for sharing analysis as well as for groundness analysis. Of course, in general, a positive Boolean function used in the role of a sharing description may have a different meaning to the same function used as a groundness dependency description. Thus, to remain unambiguous about the role of Boolean functions, we introduce names for the three Galois insertions involved:

$$\begin{aligned}\mathsf{JL} &= \langle \mathsf{Sharing}, \mathcal{P}(Sub), \gamma_{\mathsf{JL}}, \alpha_{\mathsf{JL}} \rangle \\ \mathsf{Sh} &= \langle \mathsf{Pos}, \mathcal{P}(Sub), \gamma_{\mathsf{Sh}}, \alpha_{\mathsf{Sh}} \rangle \\ \mathsf{Gr} &= \langle \mathsf{Pos}, \mathcal{P}(Sub), \gamma_{\mathsf{Gr}}, \alpha_{\mathsf{Gr}} \rangle\end{aligned}$$

We may then refer to Sh as the abstract domain used in the second Galois insertion, that is, as the incarnation of Pos whose semantics is given by γ_{Sh} (α_{Sh} and γ_{Sh} are given in Section 4.1), while Gr may be used to denote the abstract domain given by the third Galois insertion.

4.1 Set Sharing Expressed in Pos

The (logical) interpretation of a Pos function for set-sharing is clarified by the corresponding abstraction and concretization functions $\alpha_{\mathsf{Sh}} : \mathcal{P}(Sub) \to \mathsf{Pos}$ and $\gamma_{\mathsf{Sh}} : \mathsf{Pos} \to \mathcal{P}(Sub)$ for set-sharing which are defined as follows.

Recall that $Bind \subseteq Sub$ is the set of single-variable (nontrivial) *bindings* $\{v \mapsto t\}$. For $\theta \in Sub$ and $\mu \in Bind$ define

$$\mathit{unaffected}(\theta, \mu) = \{\, v \in \mathit{VI} \mid \mu(\theta(v)) = \theta(v) \,\}\,.$$

We say that v is *unaffected* in θ for the application of binding μ if $v \in \mathit{unaffected}(\theta, \mu)$. We can now define α_{Sh} and γ_{Sh}:

$$\begin{aligned}\alpha_{\mathsf{Sh}}(\Theta) &= \left\{\, \mathit{unaffected}(\theta, \mu) \,\middle|\, \theta \in \Theta,\ \mu \in \mathit{Bind} \,\right\} \\ \gamma_{\mathsf{Sh}}(\varphi) &= \left\{\, \theta \,\middle|\, \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi \,\right\}\end{aligned}$$

The underlying intuition is that φ describes a substitution θ if any set of variables that might remain unaffected when further instantiating θ by a single variable binding constitutes a model of φ. Each model m of φ, thus, represents the possibility of the variables in m to remain unaffected under some single binding for a substitution represented by φ. For example, the logical reading of the function $x \wedge (y \to z)$ for sharing analysis is: on application of any *binding*, x is unaffected, and if y is unaffected then z also is unaffected.

The concretization function in a Galois insertion is determined uniquely by the abstraction function, and usually it would be defined as $\bigcup\{\Theta \mid \alpha_{\mathsf{Sh}}(\Theta) \subseteq \varphi\}$. The form that we use instead in this definition (as well as in the definition of γ_{Gr} in Section 3.4) will help simplify proofs. The two are equivalent, since α_{Sh} is additive, that is, $\alpha_{\mathsf{Sh}}(\Theta \cup \Theta') = \alpha_{\mathsf{Sh}}(\Theta) \cup \alpha_{\mathsf{Sh}}(\Theta')$.

The relation with Jacobs and Langen's definition is clarified by the following:

Observation 4.1. For a substitution θ and binding $\{v \mapsto t\}$,

$$\mathit{unaffected}(\theta, \{v \mapsto t\}) = \mathit{VI} \setminus \mathit{occs}(\theta, v).$$

Hence, for any set of substitutions Θ,

$$m \in \alpha_{\mathsf{Sh}}(\Theta) \text{ iff } (\mathit{VI} \setminus m) \in \alpha_{\mathsf{JL}}(\Theta).$$

Example 4.1. Let $\mathit{VI} = \{w, x, y, z\}$. Then

$$\begin{aligned}\alpha_{\mathsf{JL}}(\{\ \{w \mapsto a, x \mapsto f(u_1, u_2), y \mapsto u_1, z \mapsto u_2\}\ \}) &= \{\ \emptyset, \{x, y\}, \{x, z\}\ \} \\ \alpha_{\mathsf{Sh}}(\{\ \{w \mapsto a, x \mapsto f(u_1, u_2), y \mapsto u_1, z \mapsto u_2\}\ \}) &= \{\ \{w, x, y, z\}, \{w, z\}, \{w, y\}\ \}\,.\end{aligned}$$

Notice that the sets in the Sh (Pos) abstraction are the complements of the sets in the JL (Sharing) abstraction. In the Sh abstraction, the set $\{w, x, y, z\}$ indicates the possibility of all variables in VI to remain unaffected—for example, when applying the binding $\{v \mapsto a\}$ (because $v \notin \mathit{VI}$). In the JL abstraction, the element $\emptyset$ plays the same role. Similarly the set $\{w, z\}$ in the Sh abstraction indicates that w and z may both remain unaffected, for example, when applying the binding

$\{u_1 \mapsto a\}$. The corresponding set $\{x, y\}$ in the JL abstraction says that x and y may be instantiated simultaneously by some binding, e.g., $\{u_1 \mapsto a\}$.

Note that the presence of w in all the Sh models indicates that w is necessarily unaffected by any further bindings. This means that w must be ground. In the domain of Jacobs and Langen, w is *absent* from all sharing sets.

Observation 4.2. Using Pos to represent set-sharing information has an important advantage, compared to Jacobs and Langen's representation, even when we consider only the sets-of-models view. For Jacobs and Langen, every description must be accompanied by a specification of the variables of interest (see for example Bagnara et al. [1997] or Cortesi et al. [1998]). Otherwise it is unclear whether a variable that is absent from all sharing sets is ground or simply not of interest. For example, the set $\{\emptyset, \{x\}\}$ may indicate that y is ground (if $y \in VI$), or it may say nothing about y (if $y \notin VI$).

In the dual negated presentation of Sharing, a set of variable sets is *explicit* about the set VI of variables of interest, since the defining property of a function $\varphi \in$ Pos is that $VI \in [\![\varphi]\!]$. Hence, for a sharing description expressed in Pos, such as $\{\{y\}, \{x, y\}\}$, there can be no ambiguity: thus VI can be read directly off the description; it is its largest set. In the example, $VI = \{x, y\}$.

4.2 The Sharing Operations Expressed in Pos

We now turn to the abstract operations on Sharing. Our aim is to give these in a logical form, by considering sharing information to be expressed in Pos. We need to define the abstract operations *init* (to describe a computation's "initial state"), *projection* (or *variable elimination*), *join*, and *aunify* (for *abstract unification*).

The description of a computation's initial state, that is, the abstraction of the identity substitution ε, can be considered an abstract operation *init*. This operation depends on VI. The (set of models of the) Pos function which best describes the identity substitution for a sharing analysis is

$$\alpha_{\mathsf{Sh}}(\{\varepsilon\}) = \left\{ \ VI \setminus \{v\} \bigm| v \in \mathit{Vars} \ \right\} \cup \{VI\}.$$

The intuition is that by binding a single variable, at most one variable in VI will be affected. In terms of unaffectedness dependencies, this means that if a single variable (of interest) is instantiated (by a single binding) then all other variables are unaffected. Note that the corresponding Sharing description consists of the empty set together with the set of singletons for each element of VI.

Definition 4.1. The sharing set $S \in$ Sharing and the Boolean function $\varphi \in$ Pos *correspond* for a set of variables V, written $S \sim_V \varphi$, if and only if

$$S = [\![\mathit{coneg}(\varphi)]\!]_V.$$

When the set of variables V is clear from context, we may leave out the subscript.

We now show that projection corresponds to existential quantification. The projection operation of Jacobs and Langen, which projects a sharing description S onto variables $V \subseteq VI$, is defined as

$$\mathit{project}_{\mathsf{JL}}(S, V) = \left\{ \ m \cap V \mid m \in S \ \right\}.$$

In terms of Pos operations this is equivalent to

$$\mathit{project}_{\mathsf{Sh}}(\varphi, V) = \exists w_1 \cdots \exists w_k \varphi \text{ where } \{w_1, \ldots, w_k\} = VI \setminus V.$$

LEMMA 4.1. *Let $V \subseteq VI$. For any set-sharing description S (which can be considered a $\overline{\mathsf{coPos}}$ function) and Pos function φ, if $S \sim_{VI} \varphi$ then $\mathit{project}_{\mathsf{JL}}(S, V) \sim_V \mathit{project}_{\mathsf{Sh}}(\varphi, V)$.*

PROOF. We show that each model $m \in \mathit{project}_{\mathsf{JL}}(S,V)$ has a corresponding model $V \setminus m \in \mathit{project}_{\mathsf{Sh}}(\varphi,V)$. The one nontrivial step uses the fact that the models of $\exists w_1 \cdots \exists w_n \varphi$ are the models of φ intersected with V.

$$\begin{array}{ll} & m \in \mathit{project}_{\mathsf{JL}}(S,V) \\ \text{iff} & \exists m' \text{ where } m = m' \cap V \text{ and } m' \in S \\ \text{iff} & \exists m' \text{ where } m = m' \cap V \text{ and } (VI \setminus m') \in \varphi \\ \text{iff} & \exists m' \text{ where } m = m' \cap V \text{ and } V \cap (VI \setminus m') \in \mathit{project}_{\mathsf{Sh}}(\varphi,V) \\ \text{iff} & \exists m' \text{ where } m = m' \cap V \text{ and } (V \cap VI) \setminus (V \cap m') \in \mathit{project}_{\mathsf{Sh}}(\varphi,V) \\ \text{iff} & (V \setminus m) \in \mathit{project}_{\mathsf{Sh}}(\varphi,V). \end{array}$$

□

The join operation of Jacobs and Langen is defined as union, while in the Pos formulation it is disjunction. The set of models for $\varphi_1 \vee \varphi_2$ is the union of the sets of models for φ_1 and φ_2, and hence the operations correspond because of the isomorphism between models and elements of the sharing description.

Abstract unification is (as expected) the only tricky operation. Let us first consider a unification $x = t$ (where x is a variable and t a term such that $x \notin \mathit{vars}(t)$) where no other information on the state is present. The Pos description of the unification for sharing is then

$$\sigma = x \leftrightarrow \bigwedge \mathit{vars}(t)$$

which is exactly the same as the corresponding groundness description. We can read this as saying that (under a single binding), x remains unaffected if and only if all of the variables in t remain unaffected.

Jacobs and Langen [1992] define abstract unification using the following auxiliary functions.

—The *up-closure* of $S \in$ Sharing, denoted S^* is the smallest superset of S such that $A \cup B \in S^*$ whenever $A, B \in S^*$.

—The set of components of $S \in$ Sharing which are *relevant* to a set of variables V is defined as $\mathit{rel}(S,V) = \{ A \in S \mid A \cap V \neq \emptyset \}$.

—The *cross union* of $S_1, S_2 \subseteq \mathcal{P}(VI)$ is defined: $S_1 \uplus S_2 = \{A \cup B \mid A \in S_1, B \in S_2\}$.

Let $S \in$ Sharing be given, together with a program constraint $x = t$ (where x is a variable and t a term such that $x \notin \mathit{vars}(t)$), and let $A = \mathit{rel}(S, \{x\})$ and $B = \mathit{rel}(S, \mathit{vars}(t))$. Then the abstract unification operation is defined by

$$\mathit{aunify}_{\mathsf{JL}}(S,x,t) = (S \setminus (A \cup B)) \cup (A \uplus B)^*. \tag{1}$$

Abstract unification for a pair of atoms a and b in a sharing element S is then defined inductively on the primitive constraints in $\mathit{mgu}(a,b)$.

For the logical formulation of abstract unification, $\mathit{aunify}_{\mathsf{Sh}}$, let $\varphi \in$ Pos be the function corresponding to S, namely $S = [\![\mathit{coneg}(\varphi)]\!]$, and let d be the conjunction of variables in $\mathit{vars}(t)$, that is,

$$d = \bigwedge \mathit{vars}(t).$$

Further, let $d' = x \wedge d$. Then

$$\mathit{aunify}_{\mathsf{Sh}}(\varphi,x,t) \quad = \quad (x \leftrightarrow d) \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow).$$

Before we show that this is equivalent to Jacobs and Langen's definition of abstract unification, we need some lemmas.

LEMMA 4.2. *Let φ and ψ be Boolean functions. If ψ is down-closed then*

$$(\varphi \wedge \psi)\!\downarrow\, \models \psi.$$

PROOF. We have $\varphi \wedge \psi \models \psi$. Since $\downarrow$ is monotonic, $(\varphi \wedge \psi)\!\downarrow \models \psi\!\downarrow = \psi$. $\square$

LEMMA 4.3. *Let $S \in$ Sharing, x be a variable, and t a term such that $x \notin vars(t)$, and let $A = rel(S, \{x\})$ and $B = rel(S, vars(t))$. If $S \sim \varphi$, then we have*

(1) $S^* \sim \varphi\!\downarrow$,

(2) $B \sim (\varphi \wedge \neg d)$,

(3) $S \setminus (A \cup B) \sim \varphi \wedge d'$, and

(4) $(A \uplus B)^* \sim \neg x \wedge \neg d \wedge (\varphi \wedge \neg d')\!\downarrow$.

PROOF. The items (1), (2), and (3) are straightforward. Item (4) follows, since $[\![coneg((\varphi \wedge \neg d')\!\downarrow)]\!] = (A \cup B)^*$ which contains $(A \uplus B)^*$. But the conjunction with $\neg x$ and with $\neg d$ corresponds to restricting $(A \cup B)^*$ to those sets relevant both to x and to $vars(t)$. This is precisely $(A \uplus B)^*$. $\square$

THEOREM 4.1. *If $S \sim \varphi$ then $aunify_{\mathsf{JL}}(S, x, t) \sim aunify_{\mathsf{Sh}}(\varphi, x, t)$.*

PROOF. We have

$$
\begin{array}{ll}
\multicolumn{2}{l}{aunify_{\mathsf{Sh}}(\varphi, x, t)} \\
= & (x \leftrightarrow d) \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow) \\
\stackrel{(1)}{=} & ((\neg x \wedge \neg d) \vee d') \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow) \\
\stackrel{(2)}{=} & (\neg x \wedge \neg d \wedge \varphi) \\
& \quad \vee (\neg x \wedge \neg d \wedge (\varphi \wedge \neg d')\!\downarrow) \\
& \quad \vee (\varphi \wedge d') \\
& \quad \vee (d' \wedge (\varphi \wedge \neg d')\!\downarrow) \\
\stackrel{(3)}{=} & (\neg x \wedge \neg d \wedge \varphi) \\
& \quad \vee (\neg x \wedge \neg d \wedge (\varphi \wedge \neg d')\!\downarrow) \\
& \quad \vee (\varphi \wedge d') \\
\stackrel{(4)}{=} & \underbrace{(\varphi \wedge d')}_{S \setminus (A \cup B)} \vee \underbrace{(\neg x \wedge \neg d \wedge (\varphi \wedge \neg d')\!\downarrow)}_{(A \uplus B)^*}
\end{array}
$$

The labeled steps in the proof are justified as follows:

(1) $x \leftrightarrow d \;\equiv\; (\neg x \wedge \neg d) \vee d'$;

(2) $\wedge$ distributes over $\vee$;

(3) $d' \wedge (\varphi \wedge \neg d')\!\downarrow \;\equiv$ *false* by Lemma 4.2, since $\neg d'$ is down-closed, and so $(\varphi \wedge \neg d')\!\downarrow \;\models \neg d'$;

(4) $\neg d' = \neg x \vee \neg d$, so we have $\neg x \wedge \neg d \models \neg d'$. It follows that $\neg x \wedge \neg d \wedge \varphi \models \varphi \wedge \neg d'$. Since the down-closure of any function is weaker than the function itself, we have $\varphi \wedge \neg d' \models (\varphi \wedge \neg d')\!\downarrow$ and hence $\neg x \wedge \neg d \wedge \varphi \models (\varphi \wedge \neg d')\!\downarrow$. So $\neg x \wedge \neg d \wedge \varphi \models \neg x \wedge \neg d \wedge (\varphi \wedge \neg d')\!\downarrow$.

The theorem then follows from Lemma 4.3 as indicated in the last of the displayed formulae. $\square$

Example 4.2. Let $VI = \{x, y, z\}$, and consider the analysis of $x = f(y, z)$. The initial approximation $\varphi = init_{\mathsf{Sh}}(VI)$ is, in terms of models,

$$[\![\varphi]\!] = \{\{x, y, z\}, \{x, y\}, \{x, z\}, \{y, z\}\}.$$

In the notation from above, $\varphi \wedge \neg d'$ is

$$\{\{x, y\}, \{x, z\}, \{y, z\}\} \quad \text{or} \quad (x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z).$$

The down-closure of this is

$$\{\{x, y\}, \{x, z\}, \{y, z\}, \{x\}, \{y\}, \{z\}, \emptyset\} \quad \text{or} \quad \neg x \vee \neg y \vee \neg z.$$

Taking the disjunction with φ yields

$$\{\{x,y,z\},\{x,y\},\{x,z\},\{y,z\},\{x\},\{y\},\{z\},\emptyset\} \quad \text{or} \quad \mathit{true}.$$

Hence the result is

$$\{\{x,y,z\},\{y\},\{z\},\emptyset\} \quad \text{or} \quad x \leftrightarrow (y \wedge z),$$

just as it would be for a groundness dependency analysis. The result expressed with Sharing sets is obtained by taking complements: $\{\emptyset,\{x,z\},\{x,y\},\{x,y,z\}\}$.

Notice that the processing of a unification such as $x = f(y,z)$ yields a result which is down-closed if $VI = \{x,y,z\}$.

Example 4.3. Again let $VI = \{x,y,z\}$, but now consider the analysis of $x = y, y = z$. First consider $x = y$. In the notation from above, $d' = x \wedge y$, and so $\varphi \wedge \neg d'$ is

$$\{\{x,z\},\{y,z\}\} \quad \text{or} \quad (x \not\leftrightarrow y) \wedge z.$$

The down-closure of this is

$$\{\{x,z\},\{y,z\},\{z\}\} \quad \text{or} \quad (\neg x \vee \neg y) \wedge z.$$

Taking the disjunction with φ, we get

$$\{\{x,y,z\},\{x,y\},\{x,z\},\{y,z\},\{z\}\} \quad \text{or} \quad (x \wedge y) \vee z.$$

Conjoining this with $x \leftrightarrow y$ we arrive at

$$\{\{x,y,z\},\{x,y\},\{z\}\} \quad \text{or} \quad (x \leftrightarrow y) \wedge (x \vee z) \wedge (y \vee z).$$

This result, call it φ', is the sharing description that applies at the point between the two constraints. Notice that the result differs from what a groundness analysis yields here, namely $x \leftrightarrow y$.

Now $\varphi' \wedge \neg(y \wedge z)$ is

$$\{\{x,y\},\{z\}\} \quad \text{or} \quad (x \wedge y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z),$$

the down-closure of which is

$$\{\{x,y\},\{z\},\emptyset\} \quad \text{or} \quad (x \wedge y \wedge \neg z) \vee (\neg x \wedge \neg y).$$

Taking the disjunction with φ' yields

$$\{\{x,y,z\},\{x,y\},\{z\},\emptyset\} \quad \text{or} \quad x \leftrightarrow y.$$

So the result after both constraints is $\{\{x,y,z\},\emptyset\}$, that is, $(x \leftrightarrow y) \wedge (y \leftrightarrow z)$.

5. THE RELATION BETWEEN SHARING AND GROUNDNESS

With both set-sharing and groundness dependencies specified as abstract interpretations of Pos, it is worth having a second look at the relation between these two analyses. We first note the following relation between the abstraction functions.

LEMMA 5.1. *For any substitution $\theta \in Sub$, $\alpha_{Sh}(\{\theta\})\!\downarrow\ = \alpha_{Gr}(\{\theta\})$.*

PROOF. Consider a substitution $\theta \in Sub$ and assume without loss of generality that $dom(\theta) = VI$. For any variable v, define the binding μ_v to be $\{v \mapsto a\}$. A variable $x \in VI$ is unaffected in θ by μ_v if and only if $v \notin vars(\theta(x))$. Let $m_v = \{x \in VI \mid v \notin vars(\theta(x))\}$. Then $\alpha_{\mathsf{Sh}}(\{\theta\}) = \{\, m_v \mid v \in Vars \,\}$. Note that for $v \notin range(\theta)$, $m_v = VI$.

By definition, $\alpha_{\mathsf{Gr}}(\{\theta\}) = \{\ \{x \mid \mu(\theta(x)) \text{ is ground}\} \bigm| \mu \in \mathit{Sub}\ \}$. Let V be a set of variables, and let $\mu_V \in \mathit{Sub}$ ground exactly the variables in $vars(\theta) \setminus V$. Then

$$\begin{aligned}\{x \mid \mu_V(\theta(x)) \text{ is ground}\} &= \{x \in \mathit{VI} \mid V \cap vars(\theta(x)) = \emptyset\}\\ &= \bigcap_{v \in V} \{x \in \mathit{VI} \mid v \notin vars(\theta(x))\}\\ &= \bigcap_{v \in V} m_v.\end{aligned}$$

Hence $m = \{x \mid \mu_V(\theta(x)) \text{ is ground}\} \in \alpha_{\mathsf{Gr}}(\{\theta\})$ iff $m = \bigcap_{v \in V} m_v \in \alpha_{\mathsf{Sh}}(\{\theta\})\!\downarrow$. $\square$

An immediate consequence of Lemma 5.1 is, that, whenever φ describes the groundness dependencies of a single substitution, then it also correctly describes its set-sharing. We state that as a theorem:

THEOREM 5.1. *For any $\varphi \in \mathsf{Pos}$ and $\theta \in \mathit{Sub}$, if $\theta \in \gamma_{\mathsf{Gr}}(\varphi)$ then $\theta \in \gamma_{\mathsf{Sh}}(\varphi)$.*

The converse does not necessarily hold for $\varphi \in \mathsf{Pos}$. However, it does hold when $\varphi \in \mathsf{Def}$, as we now show. Because of the unified logical view of set-sharing, the meaning of a function φ in Def is the same, whether interpreted as a sharing statement or as a groundness statement.

THEOREM 5.2. *For all $\varphi \in \mathsf{Def}$, $\gamma_{\mathsf{Sh}}(\varphi) = \gamma_{\mathsf{Gr}}(\varphi)$.*

PROOF. Take $\theta \in \gamma_{\mathsf{Sh}}(\varphi)$. As $\{\theta\} \subseteq \gamma_{\mathsf{Sh}}(\varphi)$, we have $\alpha_{\mathsf{Sh}}(\{\theta\}) \models \varphi$. By monotonicity, $\alpha_{\mathsf{Sh}}(\{\theta\})\!\downarrow \models \varphi\!\downarrow$. By Lemma 5.1, $\alpha_{\mathsf{Gr}}(\{\theta\}) \models \varphi\!\downarrow = \varphi$, since $\varphi \in \mathsf{Def}$. It follows that we have $\theta \in \gamma_{\mathsf{Gr}}(\varphi)$, and so $\gamma_{\mathsf{Sh}}(\varphi) \subseteq \gamma_{\mathsf{Gr}}(\varphi)$. The converse holds by Theorem 5.1. $\square$

It is interesting to note, that, if we consider only the subset Def of Pos, then *coneg* is homomorphic with respect to all of the interesting (abstract) operations, including $\wedge$, $\vee$, and $\exists$. It was already known that Sharing properly contains Def. Cortesi et al. [1992] show that the groundness component of Sharing is equivalent to a domain Cov used to express "covering." Cov is defined syntactically, and in hindsight it is exactly Def [Cortesi et al. 1995]. A mapping from Sharing to Cov is given as follows [Cortesi et al. 1995]:

$$\mathcal{C}(S) = \bigwedge\{\bigwedge V_1 \to \bigwedge V_2 \mid V_1, V_2 \subseteq \mathit{VI}, \forall m \in S : (m \cap V_2 \neq \emptyset) \to (m \cap V_1 \neq \emptyset)\}$$

An alternative was suggested by Cortesi et al. [1997], where $S \in \mathsf{Sharing}$ is mapped to

$$\{\mathit{VI} \setminus \bigcup B \mid B \neq \emptyset \text{ and } B \subseteq S\}.$$

However, the relation between Sharing and Def is made simpler here, because the translation via *coneg* turns elements of Sharing into positive Boolean functions, of which elements of Def are just special cases. This leads to stronger results and simplified proofs.

Casting Sharing in terms of Pos immediately gives the notion that Sharing properly contains Def, and we can establish that a sharing analysis which uses Def rather than Pos is exactly equivalent to a groundness analysis that uses Def.

LEMMA 5.2. *Let $\varphi \in \mathsf{Def}$, and let x be a variable and t a term such that $x \notin vars(t)$. Then*

$$\mathit{aunify}_{\mathsf{Gr}}(\varphi, x, t) = \mathit{aunify}_{\mathsf{Sh}}(\varphi, x, t) = \varphi \wedge (x \leftrightarrow \bigwedge vars(t)).$$

PROOF. Let $d = \bigwedge vars(t)$ and $d' = x \wedge d$. Recall that $\mathit{aunify}_{\mathsf{Sh}}(\varphi, x, t) = (x \leftrightarrow d) \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow)$. If $\varphi \in \mathsf{Def}$ then $\varphi \wedge \neg d'$ is downward closed, so $(\varphi \wedge \neg d')\!\downarrow = \varphi \wedge \neg d'$. It follows then that $\mathit{aunify}_{\mathsf{Sh}}(\varphi, x, t) = (x \leftrightarrow d) \wedge \varphi$, that is, the result from $\mathit{aunify}_{\mathsf{Gr}}(\varphi, x, t)$. $\square$

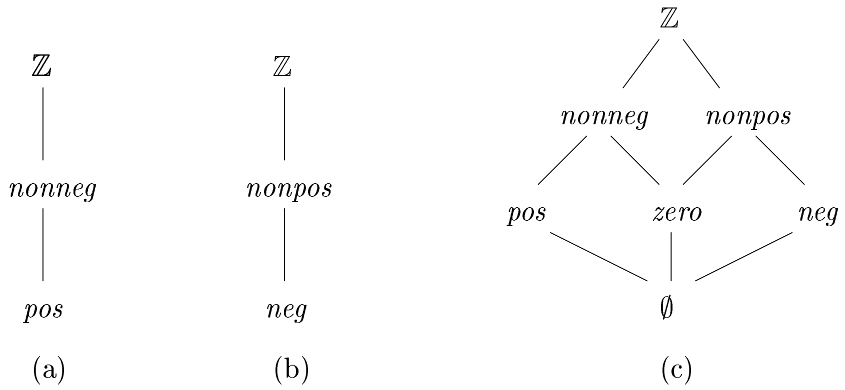


Fig. 2. The lattices (a) Sign^{+}, (b) Sign^{-}, and (c) their reduced product $\mathsf{Sign}^{+} \otimes \mathsf{Sign}^{-}$

It would be possible to use Def instead of Pos for sharing analysis, although the result would be weaker. A sharing analysis in Def is the same as a sharing analysis in Pos, except that in order to ensure that every description is in Def, the result after every operation needs to be down-closed. For example, the Pos function which best describes the identity substitution for a sharing analysis, $init_{\mathsf{Sh}}(VI)$, is not down-closed, and hence the initial description for a sharing analysis in Def is $init_{\mathsf{Sh}}(VI)\!\downarrow$ or *true*. In practice, given the theorem above, the only possibility of creating a function in $\mathsf{Pos} \setminus \mathsf{Def}$ is after a join operation. So the only differences in abstract operations for sharing in Pos and Def occur with the operations *init* and *join*.

It is easily checked that $init_{\mathsf{Sh}}(VI)\!\downarrow = init_{\mathsf{Gr}}(VI)$. Moreover, the join and projection operations for both analyses are respectively defined by $\sigma \sqcup \sigma' = (\sigma \vee \sigma')\!\downarrow$ and existential quantification. So, it follows from Lemma 5.2 that a sharing analysis and a groundness analysis using Def are identical.

6. SHARING AND GROUNDNESS: THE REDUCED PRODUCT

With both sharing and groundness analyses formalized as abstract interpretations of Pos and having investigated the relationship between the two, we now focus on a combined analysis. Clearly, two copies of Pos will enable us to perform both analyses simultaneously. However, it is also clear that two copies of Pos contain a large degree of redundancy.

In abstract interpretation, the construction that eliminates redundancy in the combination of several abstract domains is known as their *reduced product* [Cousot and Cousot 1979]. For a simple example, consider the two abstract domains Sign^{+} and Sign^{-} shown in Figure 2(a) and (b). Their elements describe sets of integers. The meaning of the elements smaller than $\mathbb{Z}$ is

$$\begin{array}{ll} pos: & \{n \in \mathbb{Z} \mid n > 0\} \\ nonneg: & \{n \in \mathbb{Z} \mid n \geq 0\} \\ zero: & \{0\} \\ neg: & \{n \in \mathbb{Z} \mid n < 0\} \\ nonpos: & \{n \in \mathbb{Z} \mid n \leq 0\}. \end{array}$$

A pair (a, b) in the Cartesian product $\mathsf{Sign}^{+} \times \mathsf{Sign}^{-}$ denotes the intersection of the sets of elements represented by a and by b. However, the Cartesian product contains redundant elements. For example, (pos, neg) and $(pos, nonpos)$ both denote the empty set of integers and are thus considered equivalent. Rather than having $3 \cdot 3 = 9$ elements, the reduced product, shown in Figure 2(c), contains only the 6 elements of the Cartesian product that correspond to distinct sets.

Consider two Galois connections $\langle A, C, \gamma_A, \alpha_A \rangle$ and $\langle B, C, \gamma_B, \alpha_B \rangle$ based on the same concrete domain C. Their reduced product

$$A \otimes B = (A \times B)_{\equiv}$$

is the quotient of the Cartesian product with respect to the equivalence relation $\equiv$ defined by

$$(a, b) \equiv (a', b') \text{ iff } \gamma_A(a) \sqcap_C \gamma_B(b) =_C \gamma_A(a') \sqcap_C \gamma_B(b').$$

The elements of the reduced product can be represented by the pairs

$$\langle \alpha_A(\gamma_A(a) \sqcap_C \gamma_B(b)), \alpha_B(\gamma_A(a) \sqcap_C \gamma_B(b)) \rangle \in A \times B$$

where $a \in A$ and $b \in B$. The abstraction and concretization functions for $A \otimes B$ can be inferred from those of A and B. This in turn determines the optimal abstract operations on the reduced product $A \otimes B$. In this article we do not investigate the design of optimal operations but take the approach of Codish et al. [1995], in which reduced product operations are designed from the corresponding operations on the individual domains. Each operation on a pair $\langle a, b \rangle$ involves the application of the corresponding operations from A and from B followed by a reduction step in which redundancies are removed from the resulting pair $\langle a', b' \rangle$.

When considering the reduced product of groundness and sharing, the accuracy in both domains may potentially be improved in view of the overlapping information in the two domains. We have already seen in Section 2 that a set-sharing analysis in general provides less groundness information than a corresponding groundness analysis using Pos. In Section 5 we explored the observation [Cortesi et al. 1992] that the groundness information present in a sharing description is an element of the subdomain Def of Pos. However, the interaction between sharing and groundness information is more subtle than that. Given a pair of Pos descriptions for sharing and for groundness dependencies, the information in each can be refined in view of the other.

Example 6.1. Let $VI = \{x, y, z\}$, and let

$$\varphi = init_{\mathsf{Sh}}(VI) = \{\{x, y, z\}, \{x, y\}, \{y, z\}, \{z, x\}\}$$

be a sharing description in Pos. Note that φ states that all of the variables in VI are independent. Suppose the groundness description in Pos is $\psi = x \leftrightarrow y$, which states that x and y are "dependent"—either both ground or both nonground. Consequently, both x and y must be ground (otherwise φ and ψ would make contradicting statements), and hence we can replace φ with $\{\{x, y, z\}, \{x, y\}\}$ and ψ with $\{\{x, y, z\}, \{x, y\}\}$.

The exact way in which sharing and groundness information affect each other is expressed in Theorem 6.1 below. Before we can state and prove it, we need two lemmas. The first characterizes the effect of groundness information in reducing a sharing description and captures the sharing information implicit in a groundness description.

LEMMA 6.1. *For all $\varphi, \psi \in$ Pos, we have*

$$\alpha_{\mathsf{Sh}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)) = \varphi \wedge \psi.$$

PROOF. We prove the two inclusions separately. We have

$$\begin{aligned} & \alpha_{\mathsf{Sh}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)) \\ = \; & \alpha_{\mathsf{Sh}}(\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi\} \cap \{\theta \mid \alpha_{\mathsf{Gr}}(\{\theta\}) \subseteq \psi\}) \\ = \; & \alpha_{\mathsf{Sh}}(\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi\} \cap \{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\})\!\downarrow \; \subseteq \psi\}) \end{aligned}$$

by Lemma 5.1. Moreover, since $\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \psi\} \supseteq \{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\})\!\downarrow\, \subseteq \psi\}$, we have

$$\begin{aligned} & \alpha_{\mathsf{Sh}}(\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi\} \cap \{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\})\!\downarrow\; \subseteq \psi\}) \\ \subseteq\ & \alpha_{\mathsf{Sh}}(\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi\} \cap \{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \psi\}) \\ =\ & \alpha_{\mathsf{Sh}}(\{\theta \mid \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi \cap \psi\}) \\ =\ & \alpha_{\mathsf{Sh}}(\gamma_{\mathsf{Sh}}(\varphi \wedge \psi)) \\ =\ & \varphi \wedge \psi. \end{aligned}$$

Next we show the opposite inclusion. Take a model m of $\varphi \wedge \psi$. Let a be a constant, and let $\theta_m = \{x \mapsto a \mid x \in m\} \cup \{x \mapsto u \mid x \notin m\}$ for some variable $u \notin VI$. Then $\theta_m \in \gamma_{\mathsf{Sh}}(\varphi)$ and $\theta_m \in \gamma_{\mathsf{Gr}}(\psi)$, and hence θ_m is also in the intersection of the two. Now $\alpha_{\mathsf{Sh}}(\{\theta_m\}) = \{m, VI\}$, and hence $m \in \alpha_{\mathsf{Sh}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi))$. Since the result holds for all models of $\varphi \wedge \psi$ we have that $\alpha_{\mathsf{Sh}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)) \supseteq \varphi \wedge \psi$. □

COROLLARY 6.1. *The sharing information in a groundness description* ψ *in* Pos *is* ψ.

PROOF. Take $\varphi = true$ in Lemma 6.1. □

We now turn to the effect of a sharing description on a groundness description. From the previous discussion it might appear obvious that the groundness information in a sharing description φ is its Def component $\varphi\!\downarrow$. Indeed, in some sense this is true, but there is more groundness information than this in a sharing description if we consider its effect on a groundness description. The next example investigates this.

Example 6.2. Let $VI = \{x, y, z\}$, and consider the sharing description

$$\varphi = init_{\mathsf{Sh}}(VI) = \{\ \{x, y, z\}, \{x, y\}, \{x, z\}, \{y, z\}\ \}$$

and the groundness description

$$\psi = \quad (x \rightarrow (y \vee z)) \quad \wedge \quad (y \rightarrow (x \vee z)) \quad \wedge \quad (z \rightarrow (x \vee y))$$

both in Pos. The sharing description φ states that all variables are independent (which includes the possibility that any of the variables are ground). This may appear to contain no groundness information, as $\varphi\!\downarrow = true$. The groundness description ψ has the set of models $\{\{x, y, z\}, \{x, y\}, \{x, z\}, \{y, z\}, \emptyset\}$ and states, that, if any one variable is ground then at least one other variable is ground as well. But ψ also imposes dependencies on the variables in a substitution (even if no variables are ground). To resolve any seemingly conflicting information, the reduced product will omit the $\emptyset$ element from the groundness description, making explicit the fact that at least two of the variables of interest are ground in any substitution described by both φ and ψ.

Example 6.3. Consider the sharing description

$$\varphi = \{\ \{x, y, z\}, \{x, y\}, \{x, z\}\ \}$$

and the groundness description

$$\psi = \{\ \{x, y, z\}, \{x, y\}, \{x, z\}, \{x\}\ \}.$$

The question is whether the sharing description φ provides additional groundness information. In particular, is the element $\{x\}$ supportable given the sharing information? It is because we can choose two models $\{x, y\}$ and $\{x, z\}$ in the intersection of φ and ψ, so as to build a substitution $\theta = \{x \mapsto f(v_1, v_2), y \mapsto v_1, z \mapsto v_2\}$ so that $\alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi$ by definition and

$$\alpha_{\mathsf{Gr}}(\{\theta\}) = \{\ \{x, y, z\}, \{x, y\}, \{x, z\}, \{x\}\ \}.$$

So θ is in $\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)$, and hence $\{x\}$ cannot be removed when the result is expressed in the reduced product.

The next lemma formalizes the effect of a sharing description in removing redundant information from a groundness description. It also captures the groundness information implicit in a sharing description. For convenience we first define a new connective.

Definition 6.1. Let φ and ψ be elements of Pos. Then

$$\varphi \downarrow \psi \quad = \quad \bigcup\{S{\downarrow} \mid S \subseteq \varphi, S{\downarrow} \subseteq \psi\}.$$

Clearly $\downarrow$ is monotonic in both its argument positions. Notice also that $\varphi \downarrow \psi \models \psi$.

LEMMA 6.2. *For all $\varphi, \psi \in Pos$ we have*

$$\alpha_{Gr}(\gamma_{Sh}(\varphi) \cap \gamma_{Gr}(\psi)) = \varphi \downarrow \psi.$$

PROOF. We prove this in terms of the models of the two Boolean functions. First we show that $\alpha_{\mathsf{Gr}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)) \supseteq \varphi \downarrow \psi$. Take $S \subseteq \varphi$ where $S{\downarrow} \subseteq \psi$. Take a substitution θ such that $\alpha_{\mathsf{Sh}}(\{\theta\}) = S$ (note that such a substitution always exists). Then $\alpha_{\mathsf{Gr}}(\{\theta\}) = S{\downarrow}$ by Lemma 5.1, and hence $\alpha_{\mathsf{Gr}}(\{\theta\}) \subseteq \psi$ which implies that $\theta \in \gamma_{\mathsf{Gr}}(\psi)$. By definition, $\theta \in \gamma_{\mathsf{Sh}}(\varphi)$, so we have $\theta \in \gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)$. Hence $S \subseteq S{\downarrow} \subseteq \alpha_{\mathsf{Gr}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi))$ because $\alpha_{\mathsf{Gr}}(\{\theta\}) = S{\downarrow}$ and $\theta \in \gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)$.

We now prove $\alpha_{\mathsf{Gr}}(\gamma_{\mathsf{Sh}}(\varphi)\cap\gamma_{\mathsf{Gr}}(\psi)) \subseteq \varphi \downarrow \psi$. Take $m \in \alpha_{\mathsf{Gr}}(\gamma_{\mathsf{Sh}}(\varphi)\cap\gamma_{\mathsf{Gr}}(\psi))$. Since m is in $\alpha_{\mathsf{Gr}}(\gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi))$, there exists $\theta \in \gamma_{\mathsf{Sh}}(\varphi) \cap \gamma_{\mathsf{Gr}}(\psi)$ where $m \in \alpha_{\mathsf{Gr}}(\{\theta\})$. By Lemma 5.1, $m \in \alpha_{\mathsf{Gr}}(\{\theta\}) = \alpha_{\mathsf{Sh}}(\{\theta\}){\downarrow}$, so there exists $S \subseteq \alpha_{\mathsf{Sh}}(\{\theta\})$ such that $m = \cap S$. Now, $m \in \alpha_{\mathsf{Gr}}(\{\theta\}) \subseteq \psi$ (because $\theta \in \gamma_{\mathsf{Gr}}(\psi)$), $S \subseteq \alpha_{\mathsf{Sh}}(\{\theta\}) \subseteq \varphi$ (because $\theta \in \gamma_{\mathsf{Sh}}(\varphi)$), and $m \in S{\downarrow}$. Now $S \subseteq \alpha_{\mathsf{Sh}}(\{\theta\})$ implies that $S{\downarrow} \subseteq \alpha_{\mathsf{Sh}}(\{\theta\}){\downarrow}$ $= \alpha_{\mathsf{Gr}}(\{\theta\}) \subseteq \psi$. Hence $m \in \varphi \downarrow \psi$. $\square$

COROLLARY 6.2. *The groundness information in a sharing description φ in Pos is $\varphi{\downarrow}$.*

PROOF. Take $\psi = \mathit{true}$ in Lemma 6.2 and observe that $\varphi \downarrow \mathit{true} = \varphi{\downarrow}$. $\square$

Together Lemma 6.1 and Lemma 6.2 entail the following theorem.

THEOREM 6.1. *Let $\langle\varphi, \psi\rangle$ be a pair of Pos functions representing sharing and groundness, respectively. Then $\langle\varphi \wedge \psi, \varphi \downarrow \psi\rangle \in Sh \otimes Gr$ is that pair's (canonical) representation in the reduced product.*

The reduced product $\mathsf{Sh} \otimes \mathsf{Gr}$ for $\mathit{VI} = \{x, y\}$ is shown in Figure 3. The difference compared with Pos is small in this case of two variables (there is only one "extra" element), but in general, the reduced product will have many additional elements. It is interesting to note, that, whenever φ (for sharing) and ψ (for groundness) are both in Def then their representation in the reduced product, $\langle\varphi\wedge\psi, \varphi \downarrow \psi\rangle$, satisfies $\varphi \wedge \psi \;=\; \varphi \downarrow \psi$. We postpone a proof of this to the next section (Corollary 7.1). Meanwhile we show that Def is closed under the binary $\downarrow$.

LEMMA 6.3. *Let φ and ψ be sharing and groundness descriptions respectively which are elements of Def. Then both $\varphi \wedge \psi$ and $\varphi \downarrow \psi$ are also elements of Def.*

PROOF. The first claim is trivial. For the second we assume that $m_1, m_2 \in \varphi \downarrow \psi$ and show that $m_1 \cap m_2 \in \varphi \downarrow \psi$. Since $m_1, m_2 \in \varphi \downarrow \psi$, there are sets $S_1, S_2 \subseteq \varphi$ such that $m_1 = \bigcap S_1$, $m_2 = \bigcap S_2$, $S_1{\downarrow}\subseteq \psi$, and $S_2{\downarrow}\subseteq \psi$. Taking $S = S_1 \cup S_2$ it is easy to check that $S \subseteq \varphi$, $S \subseteq \psi$ and since $\psi \in \mathsf{Def}$ also $S{\downarrow}\subseteq \psi$, and $m_1 \cap m_2 = \bigcap S$ which implies that $m_1 \cap m_2 \in \varphi \downarrow \psi$. $\square$

Let us reflect on the statement of Theorem 6.1. We know that the analysis of groundness dependencies with Pos is optimal for groundness [Cortesi et al. 1996], and we know that the groundness information in a sharing analysis corresponds to a Def element, i.e., it may be weaker than that obtained using Pos. Hence, there is nothing surprising in the observation that the groundness component removes

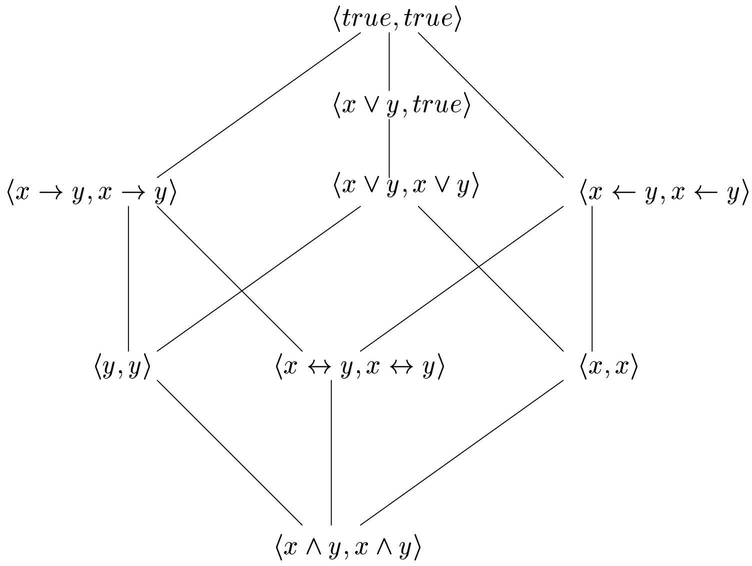


Fig. 3. The reduced product $\mathsf{Sh} \otimes \mathsf{Gr}$ for $VI = \{x, y\}$

redundancies from the sharing component in an analysis based on the reduced product.

The theorem also indicates, more surprisingly, that the sharing component can improve the groundness information in a combined analysis. However, as we shall now show, a standard (optimal) groundness analysis using Pos will *not* contain redundant sharing information for an analysis based on the (sharing) operations given above.

The abstract operations used by Jacobs and Langen with their Sharing domain have been shown to be optimal by Cortesi and Filé [1999], and this optimality result applies also when the sharing analysis is based on Pos—the issue is just one of representation. The nonredundancy of the groundness information in a combined analysis does not follow automatically from the optimality of the Pos analysis for groundness dependencies. Rather, it is a consequence of the following lemma and theorem which state, that, at every point in a combined sharing and groundness analysis described by φ and ψ for sharing and groundness respectively, $\psi = \varphi \downarrow \psi$.

LEMMA 6.4. *Let x be a variable, t a term such that $x \notin vars(t)$, and let φ, φ', ψ, and ψ' be Pos functions such that $\psi = \varphi \downarrow \psi$ and $\psi' = \varphi' \downarrow \psi'$. Then*

(1) $init_{\mathsf{Gr}} = init_{\mathsf{Sh}} \downarrow init_{\mathsf{Gr}}$

(2) $\psi \vee \psi' = \varphi \vee \varphi' \downarrow \psi \vee \psi'$

(3) $\exists v : \psi = \exists v : \varphi \downarrow \exists v : \psi$

(4) $aunify_{\mathsf{Gr}}(\psi, x, t) = aunify_{\mathsf{Sh}}(\varphi, x, t) \downarrow aunify_{\mathsf{Gr}}(\psi, x, t)$.

PROOF. Item (1) holds trivially, as $init_{\mathsf{Gr}} = true$. For the remaining items note, that, by the definition of $\downarrow$, $\psi = \varphi \downarrow \psi$ if and only if $\psi \subseteq \varphi \downarrow \psi$.

For item (2), consider any model m of ψ. This m is "supported" by a set of models $S \subseteq \varphi$ in the sense that $m = \bigcap S$ and $S\!\downarrow \ \subseteq \psi$. As $\downarrow$ is monotonic, the same support set S ensures that $m \in (\varphi \vee \varphi') \downarrow (\psi \vee \psi')$.

Item (3) is proved in essentially the same manner. For any model m' of $\exists v : \psi$ there is a model $m \in \psi$ which is supported by a set $S \subseteq \varphi$ such that $S\!\downarrow \ \subseteq \psi$. The set $S' = \exists v : S$ is a subset of $\exists v : \varphi$ and $S'\!\downarrow \ \subseteq (\exists v : \psi)$. Hence $m' \in (\exists v : \varphi) \downarrow (\exists v : \psi)$.

For item (4), let

$$\begin{array}{rcl}\varphi' & = & \mathit{aunify}_{\mathsf{Sh}}(\varphi, x, t) \;=\; (x \leftrightarrow d) \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow) \\ \psi' & = & \mathit{aunify}_{\mathsf{Gr}}(\psi, x, t) \;=\; (x \leftrightarrow d) \wedge \psi.\end{array}$$

We need to show that

$$\psi' \subseteq \varphi' \downarrow \psi'.$$

Take a model $m \in \psi'$. Then $m \in \psi$ and $m \in x \leftrightarrow d$. Since $m \in \psi$ there exists a support set $S \subseteq \varphi$ such that $m = \bigcap S$ and $S\!\downarrow \subseteq \psi$. Note that

$$S\!\downarrow \cap\, (x \leftrightarrow d) \subseteq \psi'. \tag{2}$$

Let $S_1 = S \cap (x \wedge d)$ and $S_2 = S \cap (\neg x \vee \neg d)$. Note that

$$m = \bigcap S_1 \cap \bigcap S_2. \tag{3}$$

Define $S_3 = S_2\!\downarrow \cap\, (x \leftrightarrow d)$. Then $S_1 \subseteq \varphi'$ and $S_3 \subseteq \varphi'$, and hence $S_1 \cup S_3 \subseteq \varphi'$. By Eq. (2), we have $(S_1 \cup S_3)\!\downarrow \subseteq \psi'$, since $S_1 \cup S_3 \subseteq S\!\downarrow$ and $S_1 \cup S_3 \subseteq (x \leftrightarrow d)$. Hence, we need to show that

$$m \in (S_1 \cup S_3)\!\downarrow, \tag{4}$$

from which item (4) follows.

We prove Eq. (4) by cases. Since $m \in (x \leftrightarrow d)$ either $m \models x \wedge d$ or $m \models \neg x \wedge \neg d$. In the first case, where $m \models x \wedge d$, we have $S_1 = S$ and $S_2 = S_3 = \emptyset$, and Eq. (4) follows immediately. In the second case $m \models \neg x \wedge \neg d$. Let $m' = \bigcap S_2$. By Eq. (3), since $\bigcap S_1 \models x \wedge d$ it must be the case that $m' \models \neg x \wedge \neg d$, since otherwise m could not model $\neg x \wedge \neg d$. But $m' \in S_2\!\downarrow$. Since also $m' \models \neg x \wedge \neg d$ it must be that $m' \in S_3$. Thus $m = (\cap S_1) \cap m' \in (S_1 \cup S_3)\!\downarrow$, and hence Eq. (4) is satisfied. □

We sum up the consequence of this result in the following theorem.

THEOREM 6.2. *Let $\langle \varphi, \psi \rangle \in Sh \times Gr$ be the result obtained at some program point during a simultaneous sharing/groundness analysis using the operations above. Then $\varphi \downarrow \psi = \psi$.*

The significance of this result becomes clear in light of Theorem 6.1. For a practical combined analysis, the (canonical) element of the reduced product $\mathsf{Sh} \otimes \mathsf{Gr}$ is obtained very easily: if $\langle \varphi, \psi \rangle \in \mathsf{Sh} \times \mathsf{Gr}$ are the independently calculated sharing and groundness descriptions, then $\langle \varphi \wedge \psi, \psi \rangle \in \mathsf{Sh} \otimes \mathsf{Gr}$ is the result in the reduced product.

We conclude this section with a practical note. The popularity of Sharing as a domain for sharing analysis is primarily due to the fact that it also tracks groundness information "almost as well" as Pos does for groundness. It is for this reason that it is more accurate than previous proposals for sharing analysis domains. Nevertheless, Pos is still more accurate than Sharing for groundness information, and as Pos analyses are very fast in practice, researchers have long since considered a combined groundness/sharing analysis as the best bet for deriving accurate sharing information.

It is interesting to note, that, in the context of such a combined analysis (where reduction is applied after each step involving corresponding operations from the individual domains), the abstract unification operation for sharing analysis can be further simplified. Consider its definition in Pos,

$$\mathit{aunify}_{\mathsf{Sh}}(\varphi, x, t) = (x \leftrightarrow d) \wedge (\varphi \vee (\varphi \wedge \neg d')\!\downarrow),$$

where $d = \bigwedge \mathit{vars}(t)$. The conjunct $(x \leftrightarrow d)$ is redundant! It pertains to the propagation of groundness information within the sharing analysis. In a combined analysis, if $\langle\varphi,\psi\rangle$ is in $\mathsf{Sh}\otimes\mathsf{Gr}$ we can apply a simpler operation. Namely,

$$aunify_{\mathsf{simpleSh}}(\varphi,x,t)=\varphi\vee(\varphi\wedge\neg d')\!\downarrow .$$

In the subsequent reduction operation based on Theorem 6.1, the result will be the same, since the groundness analysis contributes $x\leftrightarrow d$. This is worth stating as a theorem:

THEOREM 6.3. *Assume, in a combined sharing/groundness analysis based on* $\mathsf{Sh}\otimes\mathsf{Gr}$*, that the current description is* $\langle\varphi,\psi\rangle$*. Let the equation* $x=t$ *be given; let* $d=\bigwedge vars(t)$*; and let* $d'=x\wedge d$*. Moreover, let* $\psi'=aunify_{\mathsf{Gr}}(\psi,x,t)=(x\leftrightarrow d)\wedge\psi$*. Then* $aunify_{\mathsf{Sh}}(\varphi,x,t)\wedge\psi'=aunify_{\mathsf{simpleSh}}(\varphi,x,t)\wedge\psi'$*.*

In other words, whether $aunify_{\mathsf{Sh}}$ or $aunify_{\mathsf{simpleSh}}$ is used in the combined analysis makes no difference.

It is also worth recasting this result in the terminology of Jacobs and Langen. If $S\in\mathsf{Sharing}$, x is a variable, and t a term such that $x\notin vars(t)$, then in a combined sharing/groundness analysis we can redefine the abstract unification operation for the sharing component as follows, with no loss of precision:

$$aunify_{\mathsf{simpleJL}}(S,x,t)=S\cup(A\cup B)^*,$$

where $A=rel(S,\{x\})$ and $B=rel(S,vars(t))$. Compared with Eq. (1) this is significantly simpler. Among other simplifications, the "cross union" $A\uplus B$ has been replaced by $A\cup B$. This results in a very simple view of set-sharing analysis in which abstract unification simply adds all possible sharing among the variables involved. This simplification is a consequence of separating the groundness and sharing analyses.

Example 6.4. This example is cast in terms of Jacobs and Langen's sharing sets. Consider the combined $\mathsf{JL}\otimes Gr$ analysis of the second clause of the append program from Section 2. The sharing description at ⓓ is

$$S=\{\ \emptyset,\{u\},\{v\},\{w\},\{x\},\{y\},\{z\}\ \}.$$

We can use the weaker operation $aunify_{\mathsf{simpleJL}}$ to obtain the sharing at ⓔ, as groundness analysis will improve the result anyway. The less precise sharing now calculated for ⓔ is

$$\{\ \emptyset,\{u\},\{v\},\{w\},\{x\},\{y\},\{z\},\{u,x\},\{u,y\},\{u,x,y\}\ \}.$$

However, the groundness description $u\leftrightarrow(x\wedge y)$ eliminates the sharing sets $\{u\}$, $\{x\}$, and $\{y\}$ to give the result that would have been obtained using the more complex $aunify_{\mathsf{JL}}$.

7. REPRESENTING THE REDUCED PRODUCT

The previous section illustrates that a combined sharing and groundness analysis using two copies of Pos contains redundant information which is eliminated through their reduced product. This section points out, that, in practice we do not even need two Pos functions (in the reduced product) for the combined analysis. The key observation is that any Pos function can be partitioned into a conjunction of two functions—one in Def and the other in $(\mathsf{Pos}\setminus\mathsf{Def})\cup\{true\}$. The reason is, that, as we have seen, if $\langle\varphi,\psi\rangle\in\mathsf{Sh}\otimes\mathsf{Gr}$ constitute the sharing and groundness information in a combined (reduced product) analysis then their Def components are identical.

We first illustrate how to partition a Pos function. For a Pos function φ and set of variables of interest VI, let us denote $\varphi_\vee=\varphi\vee init_{\mathsf{Sh}}(VI)$. Recall that $\varphi\!\downarrow$ is the Def component of φ. Also note that $\varphi_\vee$ is an element of $(\mathsf{Pos}\setminus\mathsf{Def})\cup\{true\}$. This is because for any $\varphi\in\mathsf{Pos}$, $(\varphi_\vee)\!\downarrow\ =\ true$.

LEMMA 7.1. *Let* φ *be a Pos function. Then* $\varphi=\varphi\!\downarrow\wedge\,\varphi_\vee$*.*

PROOF. Clearly $\varphi \subseteq \varphi\!\downarrow \wedge \varphi_\vee$. We need to show that $\varphi \supseteq \varphi\!\downarrow \wedge\, \varphi_\vee$. Take $m \in \varphi_\vee$. If $m \in \varphi$ we are done. Otherwise $m \notin \varphi$, and m must consists of all variables in VI except one, by the definition of $\varphi_\vee$. Now $m \notin \varphi\!\downarrow$ since the only sets S of models which combine to create m with only one variable missing are $S = \{m\}$ or $S = \{m, VI\}$. Clearly $S \not\subseteq \varphi$. □

The following theorem implies that Pos is the reduced product of its two disjoint (except for the element *true*) subsets Def and $(\mathsf{Pos} \setminus \mathsf{Def}) \cup \{true\}$. This is the Pos formalization of a corresponding theorem for the *complement* $\mathsf{Sharing}^+$ of Sharing with respect to Def, due to Cortesi et al. [1995].

THEOREM 7.1. *Let $\varphi \in$ Sh and $\psi \in$ Gr be given. Then* $(\varphi \wedge \psi)\!\downarrow \;=\; (\varphi \downarrow \psi)\!\downarrow$.

PROOF. We consider the two inclusions in turn. The "$\subseteq$" direction holds because $\varphi \wedge \psi \subseteq \varphi \downarrow \psi$. To see this consider, that, if $m \in \varphi$ and $m \in \psi$ then $m \in \varphi \downarrow \psi$.

For the "$\supseteq$" direction, note, that, for $m \in (\varphi \downarrow \psi)\!\downarrow$ there exist $m_1, \ldots, m_k \in \varphi \downarrow \psi$ such that $m = \bigcap\{m_1, \ldots, m_k\}$. This implies that there exist $S_1, \ldots, S_k \subseteq \varphi$ such that $m_i = \bigcap S_i$ and $S_i\!\downarrow \subseteq \psi$ for $1 \le i \le k$. Consequently, $S_i \subseteq \psi$, and hence $S_i \subseteq \varphi \wedge \psi$ which implies that $m_i = \bigcap S_i \in (\varphi \wedge \psi)\!\downarrow$ for $1 \le i \le k$. Now we have $m_1, \ldots, m_k \in (\varphi \wedge \psi)\!\downarrow$ which implies that $m = \bigcap\{m_1, \ldots, m_k\} \in (\varphi \wedge \psi)\!\downarrow$. □

The following corollary implies that the reduced product of Def functions always has the form $\langle \xi, \xi \rangle$ for some Def function ξ (see also Figure 3).

COROLLARY 7.1. *Let $\varphi \in$ Sh and $\psi \in$ Gr be given. If $\varphi, \psi \in$ Def, then* $\varphi \wedge \psi = \varphi \downarrow \psi$.

PROOF. By Lemma 6.3, both components of $\langle \varphi \wedge \psi, \varphi \downarrow \psi \rangle$ are in Def. By Theorem 7.1, both contain the same Def information. □

While the factorization given above is appealing from a theoretical point of view, it is not clear how it can be put to a practical use in a $\mathsf{Sh} \otimes \mathsf{Gr}$ analysis. Bagnara and Schachte [1998] represent a groundness description $\psi \in$ Pos by a triplet $\langle G, E, R \rangle$. Here G is the set of variables that are deemed ground by ψ (that is, $\psi \models \bigwedge G$); E is the set of equivalences deemed to hold by ψ (that is, $\psi \models \bigwedge\{x \leftrightarrow y \mid (x, y) \in E\}$); and R is the residual Boolean function, that is, ψ simplified by specializing according to G and E. The reason why this factorization is successful is that it is relatively easy to perform the simplifications and to maintain a canonical form. Bagnara and Schachte's experimental evaluation for groundness analysis indicates that this factorization improves the efficiency of analyses.

The Bagnara-Schachte factorization seems particularly well suited for the combined sharing/groundness analysis given by the reduced product. Namely, the Boolean functions that capture the information in G and E are in Def, and so G and E are interpreted identically by the sharing analysis and the groundness analysis. A combined sharing/groundness analysis, therefore, needs only one copy of G, one of E, and two separate copies R_{Sh} and R_{Gr} for the residual information.

Interestingly, if φ is broken into parts G, E, and R_{Sh} then the function $\varphi\!\downarrow$ can be computed by simply taking the down-closure of the residual part R_{Sh}. Hence the (relatively expensive) down-closure operation need only be performed on a function that usually will have lower complexity, as it contains fewer variables.

8. CONCLUSION

We have studied the relation between Pos used for "a la Jacobs and Langen" sharing analysis (Sh) and Pos used for "a la Marriott and Søndergaard" groundness analysis (Gr), as well as their combination. In our setting, both are incarnations of Pos, the set of positive Boolean functions. From this point of view, sharing analysis and groundness analysis appear surprisingly similar.

Jacobs and Langen's domain can be thought of as a "truth-table-like" representation of the dual negated positive Boolean functions, $\overline{\mathsf{coPos}}$. Since that set of functions is isomorphic to Pos, the set of positive Boolean functions, we may as well use Pos for sharing descriptions. This "logical" view of sharing has advantages, not only because it helps our understanding of sharing analysis and its combination with groundness analysis, but also because it points toward efficient implementation of set-sharing analysis. It has already been established that Pos-based groundness analysis can be highly efficient [Bagnara and Schachte 1998; Cortesi et al. 1995; Fecht 1997; Schachte 1999]. To a large extent this has been due to the fact that Boolean functions are well understood, and efficient representations, such as ROBDDs [Bryant 1992], are available. It seems reasonable to expect that taking the "logical" view and using similar representations can lead to faster sharing analyses. For example, "a la Jacobs and Langen" sharing analysis suffers badly when sharing sets become large, whereas ROBDDs for Boolean functions with many models are often concise. While the down arrow operation is not logical, it can be implemented as an ROBDD operation, and together with the more compact representation, we expect this to lead to more efficient implementations of set-sharing analyses.

We observe, that, if Def is used for set-sharing analysis, then the result is an analysis that is identical to a groundness analysis based on Def. This sharpens the observation by Cortesi et al. [1992] that Sharing properly subsumes Def.

Perhaps the main contribution of this article is its study of the reduced product $\mathsf{Sh} \otimes \mathsf{Gr}$. We have shown, that, if the sharing/groundness descriptions obtained at some program point by independent analyses is $\langle \varphi, \psi \rangle$, then the stronger result of a combined analysis is obtained simply as $\langle \varphi \wedge \psi, \psi \rangle$. This is a particularly pleasing way of viewing a combined analysis, made possible by the logical view of sharing. We have also shown, for a combined analysis, that abstract unification can be simplified considerably, compared to Jacobs and Langen's original operation.

We therefore believe that the logic-based approach suggested here can lead to faster implementations of sharing/groundness analysis. The factorization suggested by Bagnara and Schachte [1998] seems particularly useful for a combined sharing/groundness analysis. Moreover, Bagnara et al. [1997] have shown, that, if the ultimate goal of a sharing analysis is to determine *pair-sharing*, that is, whether a given pair of variables are guaranteed to be independent, then a set-sharing analysis can be simplified further. It remains to be investigated whether this result can also help in a logic-based implementation.

Considerable imprecision is present in a sharing analysis that uses Sharing because the linearity of variables is not tracked. It is for this reason that the domains Asub [Søndergaard 1986] and Esharing [Langen 1991] incorporate linearity information with sharing information. A clean, logical incorporation of linearity information into the sharing and groundness analysis discussed here would be of great theoretical *and* practical importance.

ACKNOWLEDGMENTS

We would like to thank Roberto Bagnara, Andy King, Grigory Mashevitzky, Lee Naish, Peter Schachte, and Enea Zaffanella for stimulating discussions. We would also like to thank the referees for their careful reviews which undoubtedly improved the article.

REFERENCES

ARMSTRONG, T., MARRIOTT, K., SCHACHTE, P., AND SØNDERGAARD, H. 1998. Two classes of Boolean functions for dependency analysis. *Sci. Comput. Program. 31,* 1, 3–45.

BAGNARA, R. AND SCHACHTE, P. 1998. Factorizing equivalent variable pairs in ROBDD-based implementations of *Pos*. In *Proc. 7th Int. Conf. Algebraic Methodology and Software Technology (AMAST '98)*, A. Haeberer, Ed. Lecture Notes in Computer Science, vol. 1548. Springer, Berlin, 471–485.

BAGNARA, R., HILL, P., AND ZAFFANELLA, E. 1997. Set-sharing is redundant for pair-sharing. In *Static Analysis: Proc. 4th Int. Symp.*, P. V. Hentenryck, Ed. Lecture Notes in Computer Science, vol. 1302. Springer, Berlin, 53–67. Extended version to appear in *Theoretical Computer Science*.

BRYANT, R. 1992. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv. 24,* 3, 293–318.

CODISH, M. AND SØNDERGAARD, H. 1998. The Boolean logic of set sharing analysis. In *Principles of Declarative Programming*, C. Palamidessi, H. Glaser, and K. Meinke, Eds. Lecture Notes in Computer Science, vol. 1490. Springer, Berlin, 89–101.

CODISH, M., MULKERS, A., BRUYNOOGHE, M., DE LA BANDA, M. G., AND HERMENEGILDO, M. 1995. Improving abstract interpretations by combining domains. *ACM Trans. Program. Lang. Syst. 17,* 1, 28–44.

CORTESI, A. AND FILÉ, G. 1999. Sharing is optimal. *J. Logic Program. 38,* 3, 371–386.

CORTESI, A., FILÉ, G., GIACOBAZZI, R., PALAMIDESSI, C., AND RANZATO, F. 1995. Complementation in abstract interpretation. In *Static Analysis: Proc. 2nd Int. Symp.*, A. Mycroft, Ed. Lecture Notes in Computer Science, vol. 983. Springer, Berlin, 100–117.

CORTESI, A., FILÉ, G., GIACOBAZZI, R., PALAMIDESSI, C., AND RANZATO, F. 1997. Complementation in abstract interpretation. *ACM Trans. Program. Lang. Syst. 19,* 1, 7–47.

CORTESI, A., FILÉ, G., AND WINSBOROUGH, W. 1991. *Prop* revisited: Propositional formula as abstract domain for groundness analysis. In *Proc. 6th Annual IEEE Symp. Logic in Computer Science*. IEEE Computer Society Press, Los Alamitos, California, 322–327.

CORTESI, A., FILÉ, G., AND WINSBOROUGH, W. 1992. Comparison of abstract interpretations. In *Automata, Languages and Programming: Proc. 19th Int. Coll.*, W. Kuich, Ed. Lecture Notes in Computer Science, vol. 623. Springer-Verlag, Berlin, 521–532.

CORTESI, A., FILÉ, G., AND WINSBOROUGH, W. 1996. Optimal groundness analysis using propositional logic. *J. Logic Program. 27,* 2, 137–167.

CORTESI, A., FILÉ, G., AND WINSBOROUGH, W. 1998. The quotient of an abstract interpretation. *Theor. Comput. Sci. 202,* 1–2, 163–192.

CORTESI, A., LE CHARLIER, B., AND VAN HENTENRYCK, P. 1995. Evaluation of the domain Prop. *J. Logic Program. 23,* 3, 237–278.

COUSOT, P. AND COUSOT, R. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symp. Principles of Programming Languages*. ACM Press, New York, 238–252.

COUSOT, P. AND COUSOT, R. 1979. Systematic design of program analysis frameworks. In *Proc. 6th ACM Symp. Principles of Programming Languages*. ACM Press, New York, 269–282.

COUSOT, P. AND COUSOT, R. 1992. Abstract interpretation and application to logic programs. *J. Logic Program. 13,* 2–3, 103–179.

FECHT, C. 1997. Abstrakte interpretation logischer programme: Theorie, implementierung, generierung. Ph.D. thesis, Universität des Saarlandes, Germany.

JACOBS, D. AND LANGEN, A. 1989. Accurate and efficient approximation of variable aliasing in logic programs. In *Logic Programming: Proc. North American Conf. 1989*, E. L. Lusk and R. A. Overbeek, Eds. MIT Press, Cleveland, Ohio, 154–165.

JACOBS, D. AND LANGEN, A. 1992. Static analysis of logic programs for independent and parallelism. *J. Logic Program. 13,* 2 & 3, 291–314.

LANGEN, A. 1991. Advanced techniques for approximating variable aliasing in logic programs. Ph.D. thesis, University of Southern California, California.

LLOYD, J. W. 1987. *Foundations of Logic Programming*, second ed. Springer-Verlag, Berlin.

MARRIOTT, K. AND SØNDERGAARD, H. 1989. Notes for a tutorial on abstract interpretation of logic programs. Presented at the 1989 North American Conf. Logic Programming.

MARRIOTT, K. AND SØNDERGAARD, H. 1993. Precise and efficient groundness analysis for logic programs. *Lett. Program. Lang. Syst. 2,* 1–4, 181–196.

MARRIOTT, K., SØNDERGAARD, H., AND JONES, N. D. 1994. Denotational abstract interpretation of logic programs. *ACM Trans. Program. Lang. Syst. 16,* 3, 607–648.

SCHACHTE, P. 1999. Precise and efficient static analysis of logic programs. Ph.D. thesis, The University of Melbourne, Australia.

SCHRÖDER, E. 1877. *Der Operationskreis des Logikkalküls.* B. G. Teubner, Leibzig, Germany.

SØNDERGAARD, H. 1986. An application of abstract interpretation of logic programs: Occur check reduction. In *Proc. ESOP 86*, B. Robinet and R. Wilhelm, Eds. Lecture Notes in Computer Science, vol. 213. Springer-Verlag, Berlin, 327–338.

Received November 1998; accepted April 1999