



NATO Science for Peace and Security Series
D: Information and Communication Security - Vol. 53

Engineering Secure and Dependable Software Systems


Edited by
Alexander Pretschner
Peter Müller
Patrick Stöckle

IOS
Press



*This publication
is supported by:*

The NATO Science for Peace
and Security Programme



A Formal Introduction to Abstract Interpretation

Patrick COUSOT

Courant Institute of mathematical Sciences, New York University

Abstract. We introduce basic concepts of abstract interpretation using the example of arithmetic and boolean expression semantics, properties, verification, static analysis, and their formal calculational design.

Keywords. Semantics, Property, Verification, Proof method, Static analysis, Calculational design.

1. Introduction

Abstract interpretation [1,2,3] aims at formalizing reasonings on the semantics of programs and automating the inference of properties of such semantics. The very basic concepts of abstract interpretation are illustrated using arithmetic and boolean expressions. We define their syntax, semantics, properties, and formally design static analyses of expressions by calculus.

2. The rule of signs

The Indian mathematician and astronomer **Brahmagupta** (born c. 598, died after 665) was the first to give rules to compute with zero and invented the rule of signs [4, page 151]. Verses 18.30–35 of his *Brāhma-sphuṭ-a-siddhānta* state

[The sum] of two positives is positive, of two negatives negative; of a positive and a negative [the sum] is their difference; if they are equal it is zero. The sum of a negative and zero is negative, [that] of a positive and zero positive, [and that] of two zeros zero.

...

A negative minus zero is negative, a positive [minus zero] positive; zero [minus zero] is zero. When a positive is to be subtracted from a negative or a negative from a positive, then it is to be added.

The product of a negative and a positive is negative, of two negatives positive, and of positives positive; the product of zero and a negative, of zero and a positive, or of two zeros is zero.

A positive divided by a positive or a negative divided by a negative is positive; a zero divided by a zero is zero¹; a positive divided by a negative is negative; a negative divided by a positive is [also] negative.

A negative or a positive divided by zero has that [zero] as its divisor, or zero divided by a negative or a positive [has that negative or positive as its divisor]. The square of a negative or of a positive is positive; [the square] of zero is zero.

Following the pseudo-evaluation idea of Peter Naur in compilation [5,6], Michel Sintzoff [7] postulates the sign analysis in the following way:

“ $a \times a + b \times b$ yields always the object “pos” when a and b are the objects “pos” or “neg”, and when the valuation is defined as follows :

$$\begin{aligned} \text{pos} + \text{pos} &= \text{pos} & \text{pos} \times \text{pos} &= \text{pos} \\ \text{pos} + \text{neg} &= \text{pos, neg} & \text{pos} \times \text{neg} &= \text{neg} \\ \text{neg} + \text{pos} &= \text{pos, neg} & \text{neg} \times \text{pos} &= \text{neg} \\ \text{neg} + \text{neg} &= \text{neg} & \text{neg} \times \text{neg} &= \text{pos} \\ V(p+q) &= V(p)+V(q) & V(p \times q) &= V(p) \times V(q) \\ V(0) &= V(1) = \dots = \text{pos} \\ V(-1) &= V(-2) = \dots = \text{neg} \end{aligned}$$

The valuation of $a \times a + b \times b$ yields “pos” by the following computation :

$$\begin{aligned} V(a) &= \text{pos, neg} & V(b) &= \text{pos, neg} \\ V(a \times a) &= \text{pos} \times \text{pos}, \text{neg} \times \text{neg} & V(b \times b) &= \text{pos} \times \text{pos}, \text{neg} \times \text{neg} \\ &= \text{pos, pos} = \text{pos} & &= \text{pos, pos} = \text{pos} \\ V(a \times a + b \times b) &= V(a \times a) + V(b \times b) & &= \text{pos} + \text{pos} = \text{pos}'' \end{aligned}$$

Observe that $V(0 \times -1) = V(0) \times V(-1) = \text{pos} \times \text{neg} = \text{neg}$ while $V(0 \times -1) = V(0) = \text{pos}$. The error follows from an unsound handling of the abstraction $V(0)$ of 0 into pos. The correct rule should be $\text{neg} \times \text{pos} = \text{neg, pos}$ which is less precise than Brahmagupta’s rule of signs which singles 0 out.

Our objective is to show that such abstract interpretations of the semantics of expressions can be designed formally, without error, by machine-checkable calculational design.

3. Sign analysis of iterative programs

The rule of signs generalizes to programs. For example the sign of x in

$x = 0; \text{ while } (...) \{ x = x+1 \}$

(where the iteration condition $(...)$ is ignored) can be determined as follows:

¹This was Brahmagupta’s only error, $\frac{0}{0}$ is undefined.

- After zero iteration, when entering the loop, if ever, $x = 0$;
- After one iteration, the sign of x is zero, 1 is positive, so the sum $x+1$ of zero and positive is positive;
- For the basis, we have shown that after zero or one iteration, the sign of x is zero (at iteration 0) or positive (at iteration 1) that is positive after at most 1 iteration;
- For the induction step, if after at most $n \geq 0$ iterations, the sign of x is positive, then 1 is positive, so the sum $x+1$ of positive and positive is positive after the next iteration;
- After at most $n+1$ iterations, x is positive (at the previous $n \geq 0$ iterations) or positive (at the $n+1$ -th iteration) then x is positive after at most $n+1$ iterations;
- By recurrence on the number of iterations in the loop, x is positive in the loop.

4. Sign abstraction, informally

The abstraction is that you do not (always) need to know the absolute value of the arguments to know the sign of the result of an operation. This is sometimes precise (for example for the multiplication) but can be imprecise (for example the sign of the sum of a positive and a negative is unknown when ignoring the absolute value of the arguments). This is nevertheless useful in practice if you know what to do when you don't know the sign. For example, a compiler will not suppress the lower bound check when accessing an array with an index not known to be positive. Moreover, it is always possible to refine the abstraction to get more precise results. For example Brahmagupta states [4, page 151]

[If] a smaller [positive] is to be subtracted from a larger positive, [the result] is positive; [if] a smaller negative from a larger negative, [the result] is negative; [if] a larger [negative or positive is to be subtracted] from a smaller [positive or negative, the algebraic sign of] their difference is reversed—negative [becomes] positive and positive negative. ...

Knowing an interval of the possible values is more precise than just knowing the sign. Static interval analysis was introduced in [8,1].

Our objective is to formalize abstract interpretations of arithmetic expressions (like the rule of signs) and to show how the abstraction can be formally calculated out of the semantics of arithmetic expressions.

5. Syntax of expressions

Let us consider the language of expressions.

$x, y, \dots \in \mathbb{V}$	variables (\mathbb{V} not empty)
$A \in \mathbb{A} ::= 1 \mid x \mid A_1 - A_2$	arithmetic expressions
$B \in \mathbb{B} ::= A_1 < A_2 \mid B_1 \text{ nand } B_2$	boolean expressions
$E \in \mathbb{E} ::= A \mid B$	expressions

This context-free grammar [9] specifies sets of program syntactic entities, the set \mathbb{V} of variables, \mathbb{A} of arithmetic expressions, \mathbb{B} of boolean expressions, and \mathbb{E} of either arithmetic or boolean expressions. The mathematical variables $x, y, A, B,$ and E denote arbitrary elements of these sets.

There syntax is defined by grammar rules such as $A ::= 1 \mid x \mid A_1 - A_2$ specifying that an arithmetic expression A is either the constant 1, a variable $x \in \mathbb{V}$, or the difference $A_1 - A_2$ of two arithmetic expressions A_1 and A_2 . The set \mathbb{V} of variables is left unspecified (usually it is an identifier starting with a letter followed by 0 or more letters or digits or special symbols like “_”).

This grammar is ambiguous since $1 - 1 - 1$ can either be understood as $(1 - 1) - 1$ or $1 - (1 - 1)$. We choose the first alternative so the binary operator is left-associative. In boolean expressions, **nand** is left-associative and the arithmetic operators have priority over boolean operators (so $1-1<1-1-1$ is $((1-1)<((1-1)-1))$ *i.e.* false **ff**).

6. Structural definitions

Structural definitions are generalizations of recursive definitions on naturals. Assume that we want to define a total function $f \in \mathbb{E} \rightarrow S$ from the domain \mathbb{E} to the codomain S , where S is a set. A structural definition is a recursive definition of the form

- $f(1)$ and $f(x)$ are defined to be constants (so $f(1) \triangleq c_1$ and $f(x) \triangleq c_x$ where $c_1, c_x \in S$)²;
- $f(A_1 - A_2)$ and $f(A_1 < A_2)$ are functions of $f(A_1)$ and $f(A_2)$ (so $f(A_1 - A_2) \triangleq F_-(f(A_1), f(A_2))$, $f(A_1 < A_2) \triangleq F_<(f(A_1), f(A_2))$);
- $f(B_1 \text{ nand } B_2) \triangleq F_{\text{nand}}(f(B_1), f(B_2))$ where $F_-, F_<, F_{\text{nand}} \in S \times S \rightarrow S$.

For example $\text{vars} \in \mathbb{E} \rightarrow \wp(\mathbb{V})$, the (possibly empty) set of variables $\text{vars}[\mathbb{E}] \in \wp(\mathbb{V})$ occurring in expression $\mathbb{E} \in \mathbb{E}$, is well-defined as

$$\begin{aligned} \text{vars}[1] &\triangleq \emptyset \\ \text{vars}[x] &\triangleq \{x\} \\ \text{vars}[A_1 - A_2] &\triangleq \text{vars}[A_1] \cup \text{vars}[A_2] \\ \text{vars}[A_1 < A_2] &\triangleq \text{vars}[A_1] \cup \text{vars}[A_2] \\ \text{vars}[B_1 \text{ nand } B_2] &\triangleq \text{vars}[B_1] \cup \text{vars}[B_2] \end{aligned}$$

Structural definitions are the basis of denotational semantics introduced by Dana Scott and Christopher Strachey [10] (and called compositional in this context).

² \triangleq is “is defined as”.

7. Environments

In order to formally define the value of any expression *e.g.* $1 - 1 - 1 = -1$, we need to know the value of variables occurring in expressions *e.g.* $x - 1$ is 2 when $x = 3$, $x - 1$ is 42 when $x = 43$, *etc.* We cannot enumerate the infinitely many cases ..., $x = -1$, $x = 0$, $x = 1$, So we use an environment $\rho \in \mathbb{E}_v$ where $\mathbb{E}_v \triangleq \mathbb{V} \rightarrow \mathbb{Z}$ that is a function ρ mapping a variable x to its value $\rho(x)$ in the set \mathbb{Z} of all mathematical integers. By reasoning on the function ρ we can handle infinitely many cases at once. For example, in environment ρ , the value of $x - 1$ is $\rho(x) - 1$ where $\rho(x)$ is the value of variable x , 1 is the mathematical integer one and $-$ is the mathematical difference.

8. Structural semantics of expressions

Given an environment $\rho \in \mathbb{E}_v \triangleq \mathbb{V} \rightarrow \mathbb{Z}$ mapping variables $x \in \mathbb{V}$ to their value $\rho(x) \in \mathbb{Z}$, the value $\mathcal{A}[\mathbf{A}]\rho \in \mathbb{Z}$ of an arithmetic expression $\mathbf{A} \in \mathbb{A}$ and $\mathcal{B}[\mathbf{B}]\rho \in \mathbb{B}$ of a boolean expression $\mathbf{B} \in \mathbb{B}$ is structurally defined as follows.

$$\begin{aligned}
 \mathcal{A}[1]\rho &\triangleq 1 & (1) \\
 \mathcal{A}[x]\rho &\triangleq \rho(x) \\
 \mathcal{A}[\mathbf{A}_1 - \mathbf{A}_2]\rho &\triangleq \mathcal{A}[\mathbf{A}_1]\rho - \mathcal{A}[\mathbf{A}_2]\rho \\
 \mathcal{B}[\mathbf{A}_1 < \mathbf{A}_2]\rho &\triangleq \mathcal{A}[\mathbf{A}_1]\rho < \mathcal{A}[\mathbf{A}_2]\rho \\
 \mathcal{B}[\mathbf{B}_1 \text{ nand } \mathbf{B}_2]\rho &\triangleq \mathcal{B}[\mathbf{B}_1]\rho \uparrow \mathcal{B}[\mathbf{B}_2]\rho \\
 \mathcal{S}[\mathbf{E}] &\triangleq \mathcal{A}[\mathbf{E}] & \text{when } \mathbf{E} \in \mathbb{A} \\
 \mathcal{S}[\mathbf{E}] &\triangleq \mathcal{B}[\mathbf{E}] & \text{when } \mathbf{E} \in \mathbb{B}
 \end{aligned}$$

1 , x , $-$, $<$, **nand**, \mathbf{A} , and \mathbf{B} are syntactic objects *e.g.* strings of characters. 1 , ρ , $-$, $<$, and \uparrow are mathematical objects. The recursive definition is structural *i.e.* by induction on the syntax of expressions \mathbf{E} (either arithmetic \mathbf{A} or boolean \mathbf{B}). The semantics of complex expressions $\mathcal{A}[\mathbf{A}]$ or $\mathcal{B}[\mathbf{B}]$ is defined in function of the semantics of simpler expressions until reaching basic cases $\mathcal{A}[1]\rho \triangleq 1$ and $\mathcal{A}[x]\rho \triangleq \rho(x)$ for which the value is constant. The “not and” or “nand” boolean operator \uparrow is defined by the following truth table

a	tt	tf	ft	ff
b	tt	ff	tt	ff
$a \uparrow b$	ff	tt	tt	tt

All other logical operators (negation \neg , implication \Rightarrow , conjunction \vee , disjunction \wedge) can be defined in terms of \uparrow .

The functions \mathcal{A} and \mathcal{B} are total functions meaning that they are well-defined for all their arguments *i.e.* $\forall \mathbf{B} \in \mathbb{B} . \mathcal{B}[\mathbf{B}] \in (\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{B}$ and similarly for arithmetic expressions. The well-definedness property is therefore $P = \{\mathbf{B} \in \mathbb{B} \mid \mathcal{B}[\mathbf{B}] \in (\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{B}\}$. Its proof is by structural induction.

9. Proofs by structural induction

Proofs by structural induction are well suited for proving properties of structural definitions.

Proofs by structural induction generalize proofs by recurrence. To prove that a property P holds for all expressions $E \in \mathbb{E}$, we prove that the property holds for the basic cases 1 and x . Then assuming that the property holds for A_1 and A_2 , we prove that it holds for $A_1 - A_2$ and $A_1 < A_2$. Moreover, assuming the property holds for boolean expressions B_1 and B_2 , we prove that it also holds for $B_1 \text{ nand } B_2$. We conclude that $\mathbb{E} \subseteq P$.

10. Properties

10.1. Properties are sets

Properties (e.g. “to be an even integer”, “to be an odd natural”) can be understood as the set of mathematical objects that have this property (e.g. $2\mathbb{Z} \triangleq \{x \in \mathbb{Z} \mid \exists k \in \mathbb{Z} . x = 2k\}$ and $2\mathbb{N} + 1 = \{x \in \mathbb{N} \mid \exists k \in \mathbb{N} . x = 2k + 1\}$). So if P is a property then $x \in P$ means x has property P while $x \notin P$ means x does not have property P . For example $42 \in 2\mathbb{Z}$ but $43 \notin 2\mathbb{Z}$ while the factorial $!$ is well-defined for naturals but not integers so that $! \in \mathbb{N} \rightarrow \mathbb{N}$ and $! \notin \mathbb{Z} \rightarrow \mathbb{Z}$.

10.2. Implication, weaker and stronger properties

When considering properties as sets, logical implication is subset inclusion \subseteq . For example “to be greater than 42 implies to be positive” is $\{x \in \mathbb{Z} \mid x > 42\} \subseteq \{x \in \mathbb{Z} \mid x \geq 0\}$. If $P \subseteq Q$ then P is said to be stronger/more precise than Q and Q is said to be weaker/less precise than P . Stronger/more precise properties are satisfied by less elements while weaker/less precise properties are satisfied by more elements. False ff i.e. \emptyset is the strongest property while true tt i.e. \mathbb{Z} is the weakest property of integers.

11. Semantic properties of expressions

By expression property we might mean a property of the syntax of the expression (such as A has 42 signs – more precisely A belongs to the set of expressions with 42 signs –). This is software metrics and metrology [11], of little interest to us.

Instead an expression property will be understood as a semantic property that is a property of the semantics of expressions.

The semantics $\mathcal{A}[\mathbf{A}]$ of an expression \mathbf{A} maps environments $\rho \in \mathbb{V} \rightarrow \mathbb{Z}$ to a values in \mathbb{Z} , $\mathcal{A}[\mathbf{A}] \in (\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}$. Following Section 10, a semantic property of an expression is a set of possible semantics hence belongs to $\wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$. If $P \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ is a semantic property, then $\mathcal{A}[\mathbf{A}] \in P$ means that “ \mathbf{A} has property P ”.

Example 1 $P = \{b \mid \forall \rho \in \mathbb{V} \rightarrow \mathbb{Z} . b(\rho) = \mathbf{tt}\} \cup \{b \mid \forall \rho \in \mathbb{V} \rightarrow \mathbb{Z} . b(\rho) = \mathbf{ff}\}$ is the semantic property of a boolean expression “to always evaluate to \mathbf{tt} ” or “to always evaluate to \mathbf{ff} ”. For example $\mathbf{x} * \mathbf{x} + 1 > 0$ and $\mathbf{x} * \mathbf{x} < 0$ have this property but not $\mathbf{x} * \mathbf{x} > 0$ since $\mathbf{x} * \mathbf{x} > 0$ is sometimes true (when $|\rho(\mathbf{x})| > 0$) and sometimes false (when $|\rho(\mathbf{x})| = 0$). So $\mathcal{B}[\mathbf{x} * \mathbf{x} + 1 > 0] \in P$ while $\mathcal{B}[\mathbf{x} * \mathbf{x} > 0] \notin P$. \square

Notice that semantic properties P of expressions are just a particular case of property of expressions *i.e.* the property $\{\mathbf{E} \in \mathbb{E} \mid \mathcal{S}[\mathbf{E}] \in P\}$.

12. Collecting semantics of expressions

The collecting semantics of expressions is the strongest property of an expression.

$$\mathcal{C}[\mathbf{A}] \triangleq \{\mathcal{A}[\mathbf{A}]\} \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}) \quad (2)$$

Arithmetic expression \mathbf{A} is said to have semantic property $P \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ if and only if $\mathcal{A}[\mathbf{A}] \in P$ or equivalently $\mathcal{C}[\mathbf{A}] \subseteq P$ so that $\mathcal{C}[\mathbf{A}]$ is the strongest property of \mathbf{A} . The idea of collecting semantics was introduced in [1] (under the qualifier “static semantics”) as a basis for proving the soundness of static analyzes.

The fact that $(\mathcal{A}[\mathbf{A}] \in P) \Leftrightarrow (\mathcal{C}[\mathbf{A}] \subseteq P)$ may suggest that the concept of collecting semantics is of poor interest. However, $x \in S \Leftrightarrow \{x\} \subseteq S$ is the basic idea for abstracting set theory into order/lattice theory [12] (which has the equivalent of \subseteq but not of \in).

Similarly, the collecting semantics of boolean expressions is

$$\mathcal{C}[\mathbf{B}] \triangleq \{\mathcal{B}[\mathbf{B}]\} \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{B})$$

Again the collecting semantics $\mathcal{C}[\mathbf{E}]$ of expressions \mathbf{E} is just a particular case of property of expressions *i.e.* the property $\{\mathbf{E}' \in \mathbb{E} \mid \mathcal{S}[\mathbf{E}'] \in \mathcal{C}[\mathbf{E}]\}$ *i.e.* all expressions \mathbf{E}' that have the same semantics as \mathbf{E} .

13. Proving semantic properties of expressions by structural induction

Semantic properties can be proved by structural induction on expressions. For basic cases the proof is $\mathcal{C}[\mathbf{1}] \subseteq P$ and $\mathcal{C}[\mathbf{x}] \subseteq P$. Assuming $\mathcal{C}[\mathbf{A}_1] \subseteq P$ and $\mathcal{C}[\mathbf{A}_2] \subseteq P$, we prove $\mathcal{C}[\mathbf{A}_1 - \mathbf{A}_2] \subseteq P$ and $\mathcal{C}[\mathbf{A}_1 < \mathbf{A}_2] \subseteq P$. Assuming $\mathcal{C}[\mathbf{B}_1] \subseteq P$ and $\mathcal{C}[\mathbf{B}_2] \subseteq P$, we prove that for $\mathcal{C}[\mathbf{B}_1 \text{ \texttt{and} } \mathbf{B}_2] \subseteq P$. By structural induction, we conclude that $\mathbb{E} \subseteq \{\mathbf{E} \in \mathbb{E} \mid \mathcal{C}[\mathbf{E}] \subseteq P\}$ *i.e.* $\forall \mathbf{E} \in \mathbb{E} . \mathcal{C}[\mathbf{E}] \subseteq P$.

By structural induction on expressions, we have (we use Church’s lambda notation $\lambda x \cdot e$ for the anonymous function mapping x to the value of expression e for x [13] and $\lambda x \in S \cdot e$ to mean that the parameter x must belong to the set S)

$$\begin{aligned} \mathcal{C}[\mathbf{1}] &= \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot 1\} \\ \mathcal{C}[\mathbf{x}] &= \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot \rho(\mathbf{x})\} \end{aligned}$$

$$\begin{aligned}
\mathcal{C}[\mathbf{A}_1 - \mathbf{A}_2] &= \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot f_1(\rho) - f_2(\rho) \mid f_1 \in \mathcal{C}[\mathbf{A}_1] \wedge f_2 \in \mathcal{C}[\mathbf{A}_2]\} \\
\mathcal{C}[\mathbf{A}_1 < \mathbf{A}_2] &= \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot f_1(\rho) < f_2(\rho) \mid f_1 \in \mathcal{C}[\mathbf{A}_1] \wedge f_2 \in \mathcal{C}[\mathbf{A}_2]\} \\
\mathcal{C}[\mathbf{B}_1 \text{ nand } \mathbf{B}_2] &= \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot f_1(\rho) \uparrow f_2(\rho) \mid f_1 \in \mathcal{C}[\mathbf{B}_1] \wedge f_2 \in \mathcal{C}[\mathbf{B}_2]\} \quad \square
\end{aligned}$$

For example $\mathcal{C}[\mathbf{x} - \mathbf{x}] = \{\lambda \rho \in (\mathbb{V} \rightarrow \mathbb{Z}) \cdot 0\}$.

14. Abstract sign properties

We let $\mathbb{P}^\pm \triangleq \{\perp_\pm, <0, =0, >0, \leq 0, \neq 0, \geq 0, \top_\pm\}$ be the set of signs where <0 is “strictly negative”, ≥ 0 is “positive or zero”, *etc.*, $=0$ is “equal to zero”, $\neq 0$ is “different from zero” (*i.e.* “strictly negative or strictly positive”). \top_\pm (top) is “unknown sign” (*i.e.* \top that is “negative, zero, or positive”), \perp_\pm (bottom) is “no sign” (*i.e.* ff that is “neither negative, zero, nor positive”) be the abstract properties of the sign abstract domain \mathbb{P}^\pm . For example, the sign of \mathbf{x} at point ℓ of the conditional `if (0==1) ℓx=1;` is \perp_\pm since that point is unreachable.

The sign minus operation $\neg_\pm \in \mathbb{P}^\pm \times \mathbb{P}^\pm \rightarrow \mathbb{P}^\pm$ defines the sign $s_1 \neg_\pm s_2$ of $\mathbf{x} - \mathbf{y}$ when \mathbf{x} has sign s_1 and \mathbf{y} has sign s_2 .

		s_2							
		\perp_\pm	<0	$=0$	>0	≤ 0	$\neq 0$	≥ 0	\top_\pm
s_1	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm	\perp_\pm
	<0	\perp_\pm	\top_\pm	<0	<0	\top_\pm	\top_\pm	<0	\top_\pm
	$=0$	\perp_\pm	>0	$=0$	<0	≥ 0	$\neq 0$	≤ 0	\top_\pm
	>0	\perp_\pm	>0	>0	\top_\pm	>0	\top_\pm	\top_\pm	\top_\pm
	≤ 0	\perp_\pm	\top_\pm	≤ 0	<0	\top_\pm	\top_\pm	≤ 0	\top_\pm
	$\neq 0$	\perp_\pm	\top_\pm	$\neq 0$	\top_\pm	\top_\pm	\top_\pm	\top_\pm	\top_\pm
	≥ 0	\perp_\pm	>0	≥ 0	\top_\pm	≥ 0	\top_\pm	\top_\pm	\top_\pm
	\top_\pm	\perp_\pm	\top_\pm	\top_\pm	\top_\pm	\top_\pm	\top_\pm	\top_\pm	\top_\pm

The sign operator \neg_\pm is imprecise for difference ($-$). In contrast, the sign operator for multiplication of mathematical integers (\times) is exact *i.e.* the sign of the result is exactly known from the sign of the parameters. The above sign minus operation \neg_\pm is incorrect with machine integers because of overflows as found *e.g.* in the `int abs(int x) { return (x<0) ? -x : x; }` method in JavaTM returning a wrong value for `Integer.Min_VALUE`.

15. Structural sign semantics of expressions

The sign of an expression depends upon the sign of its free variables. We represent the sign of variables by a sign environment $\vec{\rho} \in \mathbb{V} \rightarrow \mathbb{P}^\pm$ such that $\vec{\rho}(\mathbf{x})$ is the sign of variable \mathbf{x} .

The sign semantics $\mathcal{S}^\pm[\mathbf{A}]\overset{\ddagger}{\rho}$ of an arithmetic expression \mathbf{A} is the sign of the expression value when evaluated with variables which sign is given by the sign environment $\overset{\ddagger}{\rho}$. For example, if $\overset{\ddagger}{\rho}(\mathbf{x}) = >0$ and $\overset{\ddagger}{\rho}(\mathbf{y}) = \leq 0$ then $\mathcal{S}^\pm[\mathbf{x} - \mathbf{y}]\overset{\ddagger}{\rho} = >0$.

The structural sign semantics $\mathcal{S}^\pm[\mathbf{A}] \in (\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm$ may be defined as follows.

$$\begin{aligned}\mathcal{S}^\pm[\mathbf{1}]\overset{\ddagger}{\rho} &= >0 \\ \mathcal{S}^\pm[\mathbf{x}]\overset{\ddagger}{\rho} &= \overset{\ddagger}{\rho}(\mathbf{x}) \\ \mathcal{S}^\pm[\mathbf{A}_1 - \mathbf{A}_2]\overset{\ddagger}{\rho} &= (\mathcal{S}^\pm[\mathbf{A}_1]\overset{\ddagger}{\rho}) \text{ } \neg_\pm \text{ } (\mathcal{S}^\pm[\mathbf{A}_2]\overset{\ddagger}{\rho})\end{aligned}$$

To be more precise, if any of the variables has sign \perp_\pm , meaning “the expression is never evaluated” then the result is \perp_\pm , meaning “no result is ever returned”. We say that signs are \perp_\pm -strict and define \downarrow^\pm to enforce it³.

$$\begin{aligned}\downarrow^\pm[\overset{\ddagger}{\rho}]s &\triangleq (\exists \mathbf{y} \in \mathbb{V} . \overset{\ddagger}{\rho}(\mathbf{y}) = \perp_\pm \text{ } ? \text{ } \perp_\pm \text{ } s) \\ \mathcal{S}^\pm[\mathbf{1}]\overset{\ddagger}{\rho} &= \downarrow^\pm[\overset{\ddagger}{\rho}](>0) \\ \mathcal{S}^\pm[\mathbf{x}]\overset{\ddagger}{\rho} &= \downarrow^\pm[\overset{\ddagger}{\rho}] (\overset{\ddagger}{\rho}(\mathbf{x})) \\ \mathcal{S}^\pm[\mathbf{A}_1 - \mathbf{A}_2]\overset{\ddagger}{\rho} &= (\mathcal{S}^\pm[\mathbf{A}_1]\overset{\ddagger}{\rho}) \text{ } \neg_\pm \text{ } (\mathcal{S}^\pm[\mathbf{A}_2]\overset{\ddagger}{\rho})\end{aligned}\tag{3}$$

By structural induction on \mathbf{A} , if $\exists \mathbf{x} \in \mathbb{V} . \overset{\ddagger}{\rho}(\mathbf{x}) = \perp_\pm$ then $\mathcal{S}^\pm[\mathbf{A}]\overset{\ddagger}{\rho} = \perp_\pm$.

16. Soundness

We would like to prove that the sign semantics $\mathcal{S}^\pm[\mathbf{A}]$ of an arithmetic expression \mathbf{A} is a weaker property than the collecting semantics $\mathcal{C}[\mathbf{A}]$. But $\mathcal{S}^\pm[\mathbf{A}] \in (\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm$ while $\mathcal{C}[\mathbf{A}] \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ and the concrete semantic properties in $\wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ are hardly comparable to the abstract sign properties in $(\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm$.

The solution is to express abstract properties in $(\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm$ as a concrete property in $\wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$. For that purpose we will define a concretization function $\check{\gamma}_\pm \in ((\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm) \rightarrow (\wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}))$ mapping an abstract property to an “equivalent” concrete property.

Then the concrete semantics implies the abstract semantics up to concretization in that for all arithmetic expressions \mathbf{A} ,

$$\mathcal{C}[\mathbf{A}] \subseteq \check{\gamma}_\pm(\mathcal{S}^\pm[\mathbf{A}]).$$

17. Sign concretization

We define the sign concretization function $\check{\gamma}_\pm$ in several steps.

³The conditional expression is $(\mathbf{tt} \text{ } ? \text{ } a \text{ } s \text{ } b) = a$ and $(\mathbf{ff} \text{ } ? \text{ } a \text{ } s \text{ } b) = b$.

1. First we consider signs (in \mathbb{P}^\pm) as properties of integers (in $\wp(\mathbb{Z})$).

$$\begin{aligned}
\gamma_\pm(\perp_\pm) &\triangleq \emptyset & \gamma_\pm(\leq 0) &\triangleq \{z \in \mathbb{Z} \mid z \leq 0\} \\
\gamma_\pm(< 0) &\triangleq \{z \in \mathbb{Z} \mid z < 0\} & \gamma_\pm(\neq 0) &\triangleq \{z \in \mathbb{Z} \mid z \neq 0\} \\
\gamma_\pm(= 0) &\triangleq \{0\} & \gamma_\pm(\geq 0) &\triangleq \{z \in \mathbb{Z} \mid z \geq 0\} \\
\gamma_\pm(> 0) &\triangleq \{z \in \mathbb{Z} \mid z > 0\} & \gamma_\pm(\top_\pm) &\triangleq \mathbb{Z}
\end{aligned} \tag{4}$$

2. Then we consider sign environments $\overset{\pm}{\rho} \in \mathbb{V} \rightarrow \mathbb{P}^\pm$ as properties of environments (in $\wp(\mathbb{V} \rightarrow \mathbb{Z})$). $\overset{\pm}{\rho}$ is the abstract property of all concrete environments ρ such that for all variables \mathbf{x} , the sign of $\rho(\mathbf{x})$ is $\overset{\pm}{\rho}(\mathbf{x})$.

$$\dot{\gamma}_\pm(\overset{\pm}{\rho}) \triangleq \{\rho \in \mathbb{V} \rightarrow \mathbb{Z} \mid \forall \mathbf{x} \in \mathbb{V} . \rho(\mathbf{x}) \in \gamma_\pm(\overset{\pm}{\rho}(\mathbf{x}))\} \tag{5}$$

Observe that if $\overset{\pm}{\rho}(\mathbf{x}) = \perp_\pm$ for some $\mathbf{x} \in \mathbb{V}$ then $\gamma_\pm(\overset{\pm}{\rho}(\mathbf{x})) = \emptyset$ so $\forall \mathbf{x} \in \mathbb{V} . \rho(\mathbf{x}) \in \gamma_\pm(\overset{\pm}{\rho}(\mathbf{x}))$ is false proving that $\dot{\gamma}_\pm(\overset{\pm}{\rho}) = \emptyset$. So the abstraction of false ($\emptyset \in \wp(\mathbb{V} \rightarrow \mathbb{Z})$) is any abstract environment $\overset{\pm}{\rho}$ with at least one variable \mathbf{x} such that $\overset{\pm}{\rho}(\mathbf{x}) = \perp_\pm$.

3. Finally the concretization of abstract properties $\overline{P} \in (\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm$ is the concrete property $\dot{\gamma}_\pm(\overline{P}) \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ defined as

$$\dot{\gamma}_\pm(\overline{P}) \triangleq \{\mathcal{S} \in (\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z} \mid \forall \overset{\pm}{\rho} \in \mathbb{V} \rightarrow \mathbb{P}^\pm . \forall \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho}) . \mathcal{S}(\rho) \in \gamma_\pm(\overline{P}(\overset{\pm}{\rho}))\} \tag{6}$$

i.e. \mathbf{A} has abstract property \overline{P} , that is $\mathcal{A}[\mathbf{A}] \in \dot{\gamma}_\pm(\overline{P})$, if and only if for all environments ρ with signs $\overset{\pm}{\rho}$, the value $\mathcal{A}[\mathbf{A}]\rho$ of arithmetic expression \mathbf{A} has sign $\overline{P}(\overset{\pm}{\rho})$.

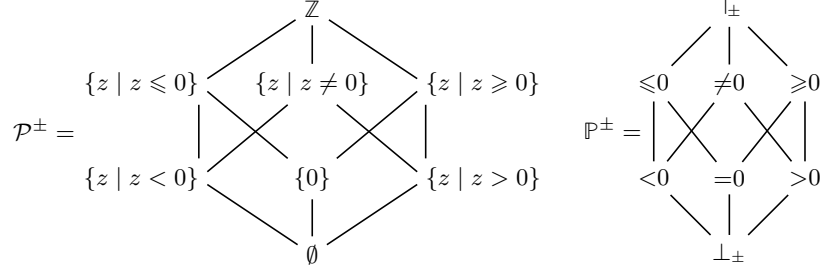
This is sound in that for all $\mathbf{A} \in \mathbb{A}$, $\mathcal{C}[\mathbf{A}] \subseteq \dot{\gamma}_\pm(\mathcal{S}^\pm[\mathbf{A}])$.

Observe that in Section 2, the concretization of signs is defined as $\gamma(\text{pos}) = \{z \in \mathbb{Z} \mid z \geq 0\}$ and $\gamma(\text{neg}) = \{z \in \mathbb{Z} \mid z < 0\}$. A sound definition of the rule of signs for multiplication \times with this interpretation of the rule of sign would have been $\text{pos} \times \text{neg} = \text{pos}$, $\text{neg} \times \text{neg} = \text{neg}$ *i.e.* \top_\pm .

18. Sign lattice

Sign properties $\mathcal{P}^\pm \triangleq \{\gamma_\pm(s) \mid s \in \mathbb{P}^\pm\}$ of integers can be partially ordered by \subseteq (*i.e.* implication) as represented by the Hasse diagram below where the nodes are the elements of \mathcal{P}^\pm and there is a bottom-up arrow from $P \in \mathcal{P}^\pm$ to $P' \in \mathcal{P}^\pm$ when $P \subsetneq P'$ and no $Q \in \mathcal{P}^\pm$ such that $P \subsetneq Q \subsetneq P'$. So $P \subseteq Q$ if and only if there is a path from P to Q in the Hasse diagram.

The abstract signs \mathbb{P}^\pm are an isomorphic representation of \mathcal{P}^\pm as shown on the right, where the isomorphism is $\gamma_\pm \in \mathbb{P}^\pm \rightarrow \mathcal{P}^\pm$.



Therefore, the abstract sign properties are partially ordered by \sqsubseteq_{\pm} defined by $s \sqsubseteq_{\pm} s'$ if and only if $\gamma_{\pm}(s) \subseteq \gamma_{\pm}(s')$. An algorithm for the inclusion \sqsubseteq_{\pm} on \mathbb{P}^{\pm} easily follows from this formal definition by case analysis.

Remark 1 Observe that \neg_{\pm} is increasing in each of its parameters i.e. if $s_1 \sqsubseteq_{\pm} s'_1$ then $s_1 \neg_{\pm} s_2 \sqsubseteq_{\pm} s'_1 \neg_{\pm} s_2$ and $s_2 \sqsubseteq_{\pm} s'_2$ then $s_1 \neg_{\pm} s_2 \sqsubseteq_{\pm} s_1 \neg_{\pm} s'_2$ so that if $s_1 \sqsubseteq_{\pm} s'_1$ and $s_2 \sqsubseteq_{\pm} s'_2$ then $s_1 \neg_{\pm} s_2 \sqsubseteq_{\pm} s'_1 \neg_{\pm} s'_2$. \square

19. Sign abstraction, formally

An integer property like $2\mathbb{N} + 1$ (odd naturals) can be over-approximated in \mathbb{P}^{\pm} by sign properties $\{z \in \mathbb{Z} \mid z > 0\}$, $\{z \in \mathbb{Z} \mid z \geq 0\}$, and \mathbb{Z} . The best over-approximation of $2\mathbb{N} + 1$ in \mathbb{P}^{\pm} is $\{z \in \mathbb{Z} \mid z > 0\}$ since it is sound (in that $2\mathbb{N} + 1 \subseteq \{z \in \mathbb{Z} \mid z > 0\}$) and the most precise/strongest (in that $\{z \in \mathbb{Z} \mid z > 0\} \subseteq \{z \in \mathbb{Z} \mid z \geq 0\} \subseteq \mathbb{Z}$).

More generally, the best over-approximation of any integer property $P \in \wp(\mathbb{Z})$ in \mathbb{P}^{\pm} is given by the abstraction function

$$\alpha_{\pm}(P) \triangleq (P \subseteq \emptyset \ ? \ \perp_{\pm} \quad (7)$$

$$\begin{aligned} & \parallel P \subseteq \{z \mid z < 0\} \ ? \ <0 \\ & \parallel P \subseteq \{0\} \ ? \ =0 \\ & \parallel P \subseteq \{z \mid z > 0\} \ ? \ >0 \\ & \parallel P \subseteq \{z \mid z \leq 0\} \ ? \ \leq 0 \\ & \parallel P \subseteq \{z \mid z \neq 0\} \ ? \ \neq 0 \\ & \parallel P \subseteq \{z \mid z \geq 0\} \ ? \ \geq 0 \\ & \text{: } \top_{\pm} \text{) } \end{aligned}$$

$\alpha_{\pm}(P)$ is the best over-approximation of $P \in \wp(\mathbb{Z})$ in \mathbb{P}^{\pm} since

- $P \subseteq \gamma_{\pm}(\alpha_{\pm}(P))$ i.e. $\alpha_{\pm}(P)$ is an over-approximation/sound abstraction of P ;
- if $\bar{P} \in \mathbb{P}^{\pm}$ and $P \subseteq \gamma_{\pm}(\bar{P})$ then $\alpha_{\pm}(P) \sqsubseteq_{\pm} \bar{P}$ i.e. $\alpha_{\pm}(P)$ is more precise than any other over-approximation/sound abstraction of P .

We have

$$s_1 \dashv\vdash s_2 = \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(s_1) \wedge y \in \gamma_{\pm}(s_2)\}). \quad (8)$$

We can use the soundness requirement (8) as a definition of $s_1 \dashv\vdash s_2 \triangleq \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(s_1) \wedge y \in \gamma_{\pm}(s_2)\})$ to design $\dashv\vdash$ by calculus. We have to consider all possible cases for s_1 and s_2 . We show three cases $\top_{\pm} \dashv\vdash \perp_{\pm} = \perp_{\pm}$, $<0 \dashv\vdash \geq 0 = <0$, and $\geq 0 \dashv\vdash \geq 0 = \top_{\pm}$.

$$\begin{aligned} & - \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(\top_{\pm}) \wedge y \in \gamma_{\pm}(\perp_{\pm})\}) \\ &= \alpha_{\pm}(\{x - y \mid x \in \mathbb{Z} \wedge y \in \emptyset\}) && \text{\textit{\textless def. } \gamma_{\pm}\textit{\textless}} \\ &= \alpha_{\pm}(\emptyset) = \perp_{\pm} && \text{\textit{\textless def. } \alpha_{\pm}\textit{\textless}} \\ & - \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(<0) \wedge y \in \gamma_{\pm}(\geq 0)\}) \\ &= \alpha_{\pm}(\{x - y \mid x < 0 \wedge y \geq 0\}) && \text{\textit{\textless def. } \gamma_{\pm}\textit{\textless}} \\ &= \alpha_{\pm}(\{x \mid x < 0\}) = <0 && \text{\textit{\textless def. } \alpha_{\pm}\textit{\textless}} \\ & - \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(\geq 0) \wedge y \in \gamma_{\pm}(\geq 0)\}) \\ &= \alpha_{\pm}(\{x - y \mid x \geq 0 \wedge y \geq 0\}) && \text{\textit{\textless def. } \gamma_{\pm}\textit{\textless}} \\ &= \alpha_{\pm}(\mathbb{Z}) = \top_{\pm} && \text{\textit{\textless def. } \alpha_{\pm}\textit{\textless}} \end{aligned}$$

The calculations can be formally certified by a proof verifier [14,15].

One can also consider all cases $s \in \mathbb{P}^{\pm}$ for $s_1 \dashv\vdash s_2$ for given s_1, s_2 when needed, using a theorem prover to make the proof that $\{x - y \mid x \in \gamma_{\pm}(s_1) \wedge y \in \gamma_{\pm}(s_2)\} \subseteq \gamma_{\pm}(s)$, and returning \top_{\pm} when the proof fails (*e.g.* times out). Among all possible answers s for which the theorem prover could make the proof, the \sqsubseteq_{\pm} -minimal one is chosen, if any. Otherwise, an arbitrary \sqsubseteq_{\pm} -minimal one has to be selected. This is called predicate abstraction [16].

The finite join \sqcup_{\pm} on \mathbb{P}^{\pm} is defined such that $\sqcup_{\pm}\{s_i \mid i \in \Delta\} \triangleq \alpha_{\pm}(\bigcup\{\gamma_{\pm}(s_i) \mid i \in \Delta\})$. It follows that $s \sqcup_{\pm} s' = \{a \mid a \in \{<0, =0, >0\} \wedge (a \sqsubseteq_{\pm} s \vee a \sqsubseteq_{\pm} s')\}$ which directly yields an algorithm for computing \sqcup_{\pm} on \mathbb{P}^{\pm} .

19.1. Abstraction of environment properties

The best abstraction of an environment property $P \in \wp(\mathbb{V} \rightarrow \mathbb{Z})$ is

$$\hat{\alpha}_{\pm}(P) \triangleq \lambda \mathbf{x} \in \mathbb{V} \cdot \alpha_{\pm}(\{\rho(\mathbf{x}) \mid \rho \in P\}) \quad (9)$$

i.e. for each variable \mathbf{x} it is the sign of the set of values $\rho(\mathbf{x})$ in all environments ρ satisfying P .

Observe that $\hat{\alpha}_{\pm}(P) \triangleq \hat{\perp}_{\pm} \triangleq \lambda \mathbf{x} \in \mathbb{V} \cdot \perp_{\pm}$ while if $\hat{\rho}(\mathbf{x}) = \perp_{\pm}$ then $\hat{\gamma}_{\pm}(\hat{\rho}) = \emptyset$ so $\emptyset \in \wp(\mathbb{V} \rightarrow \mathbb{Z})$ has several possible abstractions in \mathbb{P}^{\pm} but $\hat{\perp}_{\pm}$ is the pointwise \sqsubseteq_{\pm} -smallest of them.

19.2. Abstraction of semantic properties

The best abstraction of a semantic property $P \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ is

$$\hat{\alpha}_{\pm}(P) \triangleq \lambda \hat{\rho} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} \cdot \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in P \wedge \rho \in \hat{\gamma}_{\pm}(\hat{\rho})\}) \quad (10)$$

i.e. given a sign environment $\overset{\pm}{\rho}$, $\overset{\pm}{\alpha}_{\pm}(P)\overset{\pm}{\rho}$ is the sign of the possible results $\mathcal{S}(\rho)$ of the semantics $\mathcal{S} \in P$ with property P for all environments ρ with sign $\overset{\pm}{\rho}$.

20. Characteristic property of abstraction/concretization

The abstraction/concretization functions $\langle \alpha_{\pm}, \gamma_{\pm} \rangle$ are closely related in that for all $P \in \wp(\mathbb{Z})$ and $\overline{P} \in \mathbb{P}^{\pm}$, they satisfy

$$\alpha_{\pm}(P) \sqsubseteq_{\pm} \overline{P} \Leftrightarrow P \subseteq \gamma_{\pm}(\overline{P})$$

Proof 1 By definition (4) of γ_{\pm} and (7) of α_{\pm} , we observe that

- γ_{\pm} is increasing i.e. if $s \sqsubseteq_{\pm} s'$ then $\gamma_{\pm}(s) \subseteq \gamma_{\pm}(s')$;
- α_{\pm} is increasing i.e. if $P \subseteq P'$ then $\alpha_{\pm}(P) \sqsubseteq_{\pm} \alpha_{\pm}(P')$; (11)
- if $\alpha_{\pm}(P) = s$ then $P \subseteq \gamma_{\pm}(s)$ so $\gamma_{\pm} \circ \alpha_{\pm}$ is extensive i.e. $P \subseteq \gamma_{\pm} \circ \alpha_{\pm}(P)$; (12)
- by case analysis, if $P = \gamma_{\pm}(s)$ then $\alpha_{\pm}(P) = s$ so $\alpha_{\pm} \circ \gamma_{\pm}$ is the identity hence reductive i.e. $\alpha_{\pm} \circ \gamma_{\pm}(s) \sqsubseteq_{\pm} s$ since \sqsubseteq_{\pm} is reflexive. (13)

It follows that

$$\begin{aligned} & \alpha_{\pm}(P) \sqsubseteq_{\pm} \overline{P} \\ \Rightarrow & \gamma_{\pm} \circ \alpha_{\pm}(P) \subseteq \gamma_{\pm}(\overline{P}) && \{\gamma_{\pm} \text{ is increasing and def. function composition } \circ\} \\ \Rightarrow & P \subseteq \gamma_{\pm}(\overline{P}) && \{\gamma_{\pm} \circ \alpha_{\pm} \text{ is extensive and } \subseteq \text{ transitive}\} \\ \Rightarrow & \alpha_{\pm}(P) \sqsubseteq_{\pm} \alpha_{\pm} \circ \gamma_{\pm}(\overline{P}) && \{\alpha_{\pm} \text{ is increasing and def. function composition } \circ\} \\ \Rightarrow & \alpha_{\pm}(P) \sqsubseteq_{\pm} \overline{P} && \{\alpha_{\pm} \circ \gamma_{\pm} \text{ is reductive and def. function composition } \circ\} \quad \square \end{aligned}$$

Similar results hold for $\langle \hat{\alpha}_{\pm}, \hat{\gamma}_{\pm} \rangle$, and $\langle \check{\alpha}_{\pm}, \check{\gamma}_{\pm} \rangle$, see (14).

21. Galois connection

The abstraction/concretization functions $\langle \alpha_{\pm}, \gamma_{\pm} \rangle$ satisfy $\forall P \in \wp(\mathbb{Z}) . \forall \overline{P} \in \mathbb{P}^{\pm} . \alpha_{\pm}(P) \sqsubseteq_{\pm} \overline{P} \Leftrightarrow P \subseteq \gamma_{\pm}(\overline{P})$, which is the definition of a Galois connection, which we write $\langle \wp(\mathbb{Z}), \subseteq \rangle \xleftrightarrow[\alpha_{\pm}]{\gamma_{\pm}} \langle \mathbb{P}^{\pm}, \sqsubseteq_{\pm} \rangle$.

More generally,

Definition 1 (Galois connection) a Galois connection $\langle \mathbb{P}, \subseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \overline{\mathbb{P}}, \overline{\subseteq} \rangle$ is such that the concrete domain $\langle \mathbb{P}, \subseteq \rangle$ and the abstract domain $\langle \overline{\mathbb{P}}, \overline{\subseteq} \rangle$ are partial orders, $\alpha \in \mathbb{P} \rightarrow \overline{\mathbb{P}}$ is the abstraction function, $\gamma \in \overline{\mathbb{P}} \rightarrow \mathbb{P}$ is the concretization function, and $\forall P \in \mathbb{P} . \forall \overline{P} \in \overline{\mathbb{P}} . \alpha(P) \overline{\subseteq} \overline{P} \Leftrightarrow P \subseteq \gamma(\overline{P})$. \square

For example

$$\begin{aligned} \langle \wp(\mathbb{V} \rightarrow \mathbb{Z}), \sqsubseteq \rangle &\xleftrightarrow[\dot{\alpha}_{\pm}]{\dot{\gamma}_{\pm}} \langle \mathbb{V} \rightarrow \mathbb{P}^{\pm}, \dot{\sqsubseteq}_{\pm} \rangle & (14) \\ \langle \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}), \sqsubseteq \rangle &\xleftrightarrow[\ddot{\alpha}_{\pm}]{\ddot{\gamma}_{\pm}} \langle (\mathbb{V} \rightarrow \mathbb{P}^{\pm}) \rightarrow \mathbb{P}^{\pm}, \dot{\sqsubseteq}_{\pm} \rangle. \end{aligned}$$

Proof 2 For all $P \in \wp(\mathbb{V} \rightarrow \mathbb{Z})$ and $\dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm}$, we have

$$\begin{aligned} &\dot{\alpha}_{\pm}(P) \dot{\sqsubseteq}_{\pm} \dot{\bar{\rho}} \\ \Leftrightarrow &\forall \mathbf{x} \in \mathbb{V} . \dot{\alpha}_{\pm}(P)\mathbf{x} \dot{\sqsubseteq}_{\pm} \dot{\bar{\rho}}(\mathbf{x}) && \{ \text{pointwise def. of } \dot{\sqsubseteq}_{\pm} \} \\ \Leftrightarrow &\forall \mathbf{x} \in \mathbb{V} . \alpha_{\pm}(\{\rho(\mathbf{x}) \mid \rho \in P\}) \dot{\sqsubseteq}_{\pm} \dot{\bar{\rho}}(\mathbf{x}) && \{ \text{def. (9) of } \dot{\alpha}_{\pm} \} \\ \Leftrightarrow &\forall \mathbf{x} \in \mathbb{V} . \{\rho(\mathbf{x}) \mid \rho \in P\} \subseteq \gamma_{\pm}(\dot{\bar{\rho}}(\mathbf{x})) && \{ \langle \wp(\mathbb{Z}), \sqsubseteq \rangle \xleftrightarrow[\alpha_{\pm}]{\gamma_{\pm}} \langle \mathbb{P}^{\pm}, \sqsubseteq_{\pm} \rangle \} \\ \Leftrightarrow &\forall \mathbf{x} \in \mathbb{V} . \forall \rho \in P . \rho(\mathbf{x}) \in \gamma_{\pm}(\dot{\bar{\rho}}(\mathbf{x})) && \{ \text{def. } \in \} \\ \Leftrightarrow &\forall \rho \in P . \forall \mathbf{x} \in \mathbb{V} . \rho(\mathbf{x}) \in \gamma_{\pm}(\dot{\bar{\rho}}(\mathbf{x})) && \{ \text{def. } \forall \} \\ \Leftrightarrow &P \subseteq \{\rho \in \mathbb{V} \rightarrow \mathbb{Z} \mid \forall \mathbf{x} \in \mathbb{V} . \rho(\mathbf{x}) \in \gamma_{\pm}(\dot{\bar{\rho}}(\mathbf{x}))\} && \{ \text{def. } \subseteq \} \\ \Leftrightarrow &P \subseteq \dot{\gamma}_{\pm}(\dot{\bar{\rho}}) && \{ \text{def. (5) of } \dot{\gamma}_{\pm}, \text{ proving } \langle \wp(\mathbb{V} \rightarrow \mathbb{Z}), \sqsubseteq \rangle \xleftrightarrow[\dot{\alpha}_{\pm}]{\dot{\gamma}_{\pm}} \langle \mathbb{V} \rightarrow \mathbb{P}^{\pm}, \dot{\sqsubseteq}_{\pm} \rangle \} \end{aligned}$$

For all $P \in \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ and $\bar{P} \in (\mathbb{V} \rightarrow \mathbb{P}^{\pm}) \rightarrow \mathbb{P}^{\pm}$, we have

$$\begin{aligned} &\ddot{\alpha}_{\pm}(P) \dot{\sqsubseteq}_{\pm} \bar{P} \\ \Leftrightarrow &\forall \dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} . \ddot{\alpha}_{\pm}(P)\dot{\bar{\rho}} \dot{\sqsubseteq}_{\pm} \bar{P}(\dot{\bar{\rho}}) && \{ \text{pointwise def. of } \dot{\sqsubseteq}_{\pm} \} \\ \Leftrightarrow &\forall \dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} . \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in P \wedge \rho \in \dot{\gamma}_{\pm}(\dot{\bar{\rho}})\}) \dot{\sqsubseteq}_{\pm} \bar{P}(\dot{\bar{\rho}}) && \{ \text{def. (10) of } \ddot{\alpha}_{\pm} \} \\ \Leftrightarrow &\forall \dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} . \{\mathcal{S}(\rho) \mid \mathcal{S} \in P \wedge \rho \in \dot{\gamma}_{\pm}(\dot{\bar{\rho}})\} \subseteq \gamma_{\pm}(\bar{P}(\dot{\bar{\rho}})) && \{ \langle \wp(\mathbb{Z}), \sqsubseteq \rangle \xleftrightarrow[\alpha_{\pm}]{\gamma_{\pm}} \langle \mathbb{P}^{\pm}, \sqsubseteq_{\pm} \rangle \} \\ \Leftrightarrow &\forall \dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} . \forall \mathcal{S} \in P . \forall \rho \in \dot{\gamma}_{\pm}(\dot{\bar{\rho}}) . \mathcal{S}(\rho) \in \gamma_{\pm}(\bar{P}(\dot{\bar{\rho}})) && \{ \text{def. } \subseteq \} \\ \Leftrightarrow &\forall \mathcal{S} \in P . \forall \dot{\bar{\rho}} \in \mathbb{V} \rightarrow \mathbb{P}^{\pm} . \forall \rho \in \dot{\gamma}_{\pm}(\dot{\bar{\rho}}) . \mathcal{S}(\rho) \in \gamma_{\pm}(\bar{P}(\dot{\bar{\rho}})) && \{ \text{def. } \forall \} \\ \Leftrightarrow &\forall \mathcal{S} \in P . \mathcal{S} \in \dot{\gamma}_{\pm}(\bar{P}) && \{ \text{def. } \in \text{ and (6) of } \dot{\gamma}_{\pm} \} \\ \Leftrightarrow &P \subseteq \dot{\gamma}_{\pm}(\bar{P}) \\ &\{ \text{def. } \subseteq, \text{ proving } \langle \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}), \sqsubseteq \rangle \xleftrightarrow[\ddot{\alpha}_{\pm}]{\ddot{\gamma}_{\pm}} \langle (\mathbb{V} \rightarrow \mathbb{P}^{\pm}) \rightarrow \mathbb{P}^{\pm}, \dot{\sqsubseteq}_{\pm} \rangle. \} \quad \square \end{aligned}$$

22. Computational design of the sign semantics of expressions

The soundness requirement in Section 17 is that $\forall \mathbf{A} \in \mathbb{A} . \mathcal{C}[\mathbf{A}] \subseteq \dot{\gamma}_{\pm}(\mathcal{S}^{\pm}[\mathbf{A}])$. By the Galois connection of (14), this is equivalent to $\ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}]) \dot{\sqsubseteq}_{\pm} \mathcal{S}^{\pm}[\mathbf{A}]$. Therefore the sign semantics is a sign abstraction of the collecting semantics. It follows that we can design $\mathcal{S}^{\pm}[\mathbf{A}]$ by calculus, calculating $\ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}])$ using $\dot{\sqsubseteq}_{\pm}$ -over-approximation to avoid all computations made in the concrete domain.

- We first consider the case when $\exists \mathbf{x} \in \mathbb{V} . \dot{\bar{\rho}}(\mathbf{x}) = \perp_{\pm}$ so that $\dot{\gamma}_{\pm}(\dot{\bar{\rho}}) = \emptyset$.

$$\begin{aligned}
& - \ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}])\overset{\pm}{\rho} \\
& = \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{A}] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (10) of } \ddot{\alpha}_{\pm} \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\mathcal{A}[\mathbf{A}](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (2) of } \mathcal{C}[\mathbf{A}] \text{\textcircled{?}} \\
& = \alpha_{\pm}(\emptyset) \quad \text{\textcircled{?} } \exists \mathbf{x} \in \mathbb{V} . \overset{\pm}{\rho}(\mathbf{x}) = \perp_{\pm} \text{ so that } \dot{\gamma}_{\pm}(\overset{\pm}{\rho}) = \emptyset \text{\textcircled{?}} \\
& = \perp_{\pm} \quad \text{\textcircled{?} def. (7) of } \alpha_{\pm} \text{\textcircled{?}} \\
& \triangleq \mathcal{S}^{\pm}[\mathbf{A}]\overset{\pm}{\rho} \\
& \quad \text{\textcircled{?} in accordance with (3) such that, } \exists \mathbf{x} \in \mathbb{V} . \overset{\pm}{\rho}(\mathbf{x}) = \perp_{\pm} \text{ implies} \\
& \quad \mathcal{S}^{\pm}[\mathbf{A}]\overset{\pm}{\rho} = \perp_{\pm} \text{\textcircled{?}}
\end{aligned}$$

- Then we consider the case when $\forall \mathbf{x} \in \mathbb{V} . \overset{\pm}{\rho}(\mathbf{x}) \neq \perp_{\pm}$ so that $\dot{\gamma}_{\pm}(\overset{\pm}{\rho}) \neq \emptyset$.

We proceed by structural induction on \mathbf{A} .

- For the basic case of a constant 1, we just apply the definitions.

$$\begin{aligned}
& \ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{1}])\overset{\pm}{\rho} \\
& = \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{1}] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (10) of } \ddot{\alpha}_{\pm} \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\mathcal{A}[\mathbf{1}](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (2) of } \mathcal{C}[\mathbf{1}] \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{1\}) \quad \text{\textcircled{?} } \dot{\gamma}_{\pm}(\overset{\pm}{\rho}) \text{ is not empty and def. (1) of } \mathcal{A}[\mathbf{1}] \text{\textcircled{?}} \\
& = >0 \quad \text{\textcircled{?} def. (7) of } \alpha_{\pm} \text{\textcircled{?}} \\
& \triangleq \mathcal{S}^{\pm}[\mathbf{1}]\overset{\pm}{\rho} \quad \text{\textcircled{?} in accordance with (3) when } \forall \mathbf{y} \in \mathbb{V} . \overset{\pm}{\rho}(\mathbf{y}) \neq \perp_{\pm} \text{\textcircled{?}}
\end{aligned}$$

- For the basic case of a variable \mathbf{x} , we apply the definitions and then simplify.

$$\begin{aligned}
& \ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{x}])\overset{\pm}{\rho} \\
& = \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{x}] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (10) of } \ddot{\alpha}_{\pm} \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\mathcal{A}[\mathbf{x}](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (2) of } \mathcal{C}[\mathbf{x}] \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\rho(\mathbf{x}) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \quad \text{\textcircled{?} def. (1) of } \mathcal{A}[\mathbf{x}] \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\rho(\mathbf{x}) \mid \forall \mathbf{y} \in \mathbb{V} . \rho(\mathbf{y}) \in \gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{y}))\}) \quad \text{\textcircled{?} def. (5) of } \dot{\gamma}_{\pm} \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{\rho(\mathbf{x}) \mid \rho(\mathbf{x}) \in \gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{x}))\}) \\
& \quad \text{\textcircled{?} since } \gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{y})) \text{ is not empty so for } \mathbf{y} \neq \mathbf{x}, \rho(\mathbf{y}) \text{ can be chosen} \\
& \quad \text{arbitrarily to satisfy } \rho(\mathbf{y}) \in \gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{y})) \text{\textcircled{?}} \\
& = \alpha_{\pm}(\{x \mid x \in \gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{x}))\}) \quad \text{\textcircled{?} letting } x = \rho(\mathbf{x}) \text{\textcircled{?}} \\
& = \alpha_{\pm}(\gamma_{\pm}(\overset{\pm}{\rho}(\mathbf{x}))) \quad \text{\textcircled{?} since } S = \{x \mid z \in S\} \text{ for any set } S \text{\textcircled{?}} \\
& = \overset{\pm}{\rho}(\mathbf{x}) \quad \text{\textcircled{?} by (13), } \alpha_{\pm} \circ \gamma_{\pm} \text{ is the identity \texttextcircled{?}} \\
& \triangleq \mathcal{S}^{\pm}[\mathbf{x}]\overset{\pm}{\rho} \quad \text{\textcircled{?} in accordance with (3) when } \forall \mathbf{y} \in \mathbb{V} . \overset{\pm}{\rho}(\mathbf{y}) \neq \perp_{\pm} \text{\textcircled{?}}
\end{aligned}$$

- For the inductive case of $\mathbf{A}_1 - \mathbf{A}_2$, we assume, by structural induction hypothesis, that $\ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_1]) \dot{\sqsubseteq}_{\pm} \mathcal{S}^{\pm}[\mathbf{A}_1]$ and $\ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_2]) \dot{\sqsubseteq}_{\pm} \mathcal{S}^{\pm}[\mathbf{A}_2]$

$$\ddot{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_1 - \mathbf{A}_2])\overset{\pm}{\rho}$$

$$\begin{aligned}
&= \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{A}_1 - \mathbf{A}_2] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) && \text{\textcircled{def. (10) of } \check{\alpha}_{\pm}} \\
&= \alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_1 - \mathbf{A}_2](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) && \text{\textcircled{def. (2) of } \mathcal{C}[\mathbf{A}_1 - \mathbf{A}_2]} \\
&= \alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_1](\rho) - \mathcal{A}[\mathbf{A}_2](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) && \text{\textcircled{def. (1) of } \mathcal{A}} \\
\sqsubseteq_{\pm} \alpha_{\pm}(\{x - y \mid x \in \{\mathcal{A}[\mathbf{A}_1](\rho') \mid \rho' \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\} \wedge y \in \{\mathcal{A}[\mathbf{A}_2](\rho'') \mid \rho'' \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}\}) \\
&\quad \{\{f(\rho) - g(\rho) \mid \rho \in R\} \subseteq \{x - y \mid x \in \{f(\rho') \mid \rho' \in R\} \wedge y \in \{g(\rho'') \mid \rho'' \in R\}\} \text{ and } \alpha_{\pm} \text{ is increasing by (11).}\} \\
&\quad \text{\textcircled{This over-approximation allows for } \mathbf{A}_1 \text{ and } \mathbf{A}_2 \text{ to be evaluated in the concrete with different environments } \rho' \text{ and } \rho'' \text{ with the same sign of variables but possibly different values of variables. This accounts for the fact that the rule of signs does not take relationships between values of variables into account. For example the sign of } \mathbf{x} - \mathbf{x} \text{ is not } =0 \text{ in general.}} \\
\sqsubseteq_{\pm} \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(\alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_1](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\})) \wedge y \in \gamma_{\pm}(\alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_2](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}))\}) \\
&\quad \{\{x - y \mid x \in P \wedge y \in Q\} \subseteq \{x - y \mid x \in \gamma_{\pm}(\alpha_{\pm}(P)) \wedge y \in \gamma_{\pm}(\alpha_{\pm}(Q))\} \\
&\quad \text{since } \gamma_{\pm} \circ \alpha_{\pm} \text{ is extensive by (12) and } \alpha_{\pm} \text{ is increasing by (11).}\} \\
&\quad \text{\textcircled{This over-approximation allows for the evaluation of the sign to be performed in the abstract with } \mp_{\pm} \text{ instead of the concrete.}} \\
&= \alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_1](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \mp_{\pm} \alpha_{\pm}(\{\mathcal{A}[\mathbf{A}_2](\rho) \mid \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \\
&\quad \{\{s_1 \mp_{\pm} s_2 = \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}(s_1) \wedge y \in \gamma_{\pm}(s_2)\})\} \text{ by (8)} \\
&= \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{A}_1] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \mp_{\pm} \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{C}[\mathbf{A}_2] \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) && \text{\textcircled{def. (2) of } \mathcal{C}[\mathbf{A}]} \\
&= \check{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_1]) \overset{\pm}{\rho} \mp_{\pm} \check{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_2]) \overset{\pm}{\rho} && \text{\textcircled{def. (10) of } \check{\alpha}_{\pm}} \\
&= \check{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_1]) \overset{\pm}{\rho} \mp_{\pm} \check{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}_2]) \overset{\pm}{\rho} && \text{\textcircled{def. (10) of } \check{\alpha}_{\pm}} \\
\sqsubseteq_{\pm} (\mathcal{S}^{\pm}[\mathbf{A}_1]) \overset{\pm}{\rho} \mp_{\pm} (\mathcal{S}^{\pm}[\mathbf{A}_2]) \overset{\pm}{\rho} \\
&\quad \text{\textcircled{induction hypothesis and } \mp_{\pm} \text{ is increasing in both parameters by Remark 1}} \\
\triangleq \mathcal{S}^{\pm}[\mathbf{A}_1 - \mathbf{A}_2] \overset{\pm}{\rho} && \text{\textcircled{in accordance with (3) when } \forall y \in \mathbb{V} . \overset{\pm}{\rho}(y) \neq \perp_{\pm}} \quad \square
\end{aligned}$$

23. Calculational design of abstract interpretations

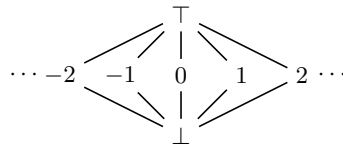
This concludes our formal design of the rule of signs for arithmetic expressions.

- We first define the semantics $\mathcal{A}[\mathbf{A}]$ of arithmetic expressions \mathbf{A} in (1);
- The strongest property of the semantics of arithmetic expressions \mathbf{A} is their collecting semantics $\mathcal{C}[\mathbf{A}]$ in (2);
- Among the semantic properties $\wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z})$ of arithmetic expressions, we select a subset of properties of interest *i.e.* the sign properties and choose a computer representation, as defined by the abstraction function $\check{\alpha}_{\pm}$ in (10), which is the lower adjoint of the Galois connection (14);
- The rule of sign $\mathcal{S}^{\pm}[\mathbf{A}]$ is then formally derived by calculational design in Section 22 by over-approximating the best abstraction $\check{\alpha}_{\pm}(\mathcal{C}[\mathbf{A}])$ of the collecting semantics $\mathcal{C}[\mathbf{A}]$.

It follows that $\mathcal{S}^\pm[\mathbb{A}]$ is sound by construction.

24. Classical finitary abstractions

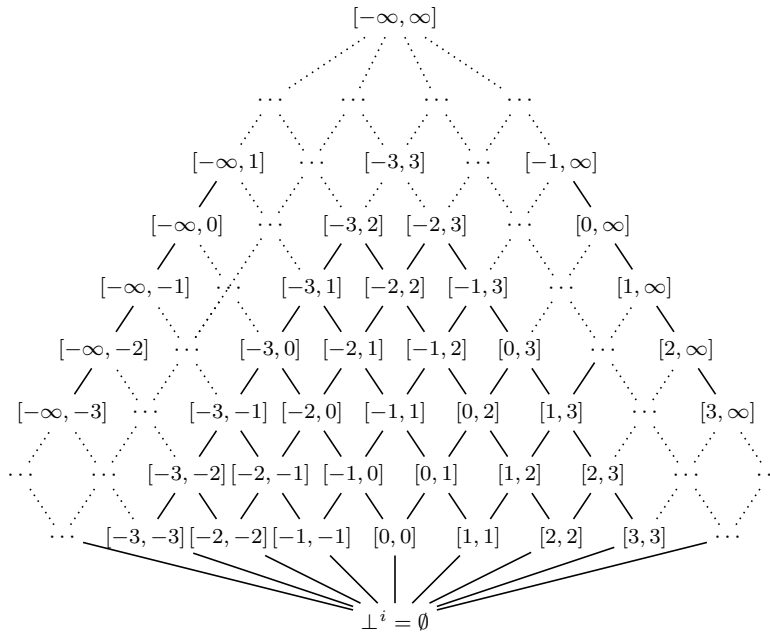
Other elementary examples are the parity analysis (which is correct with machine integers), [17] constancy analysis based on the lattice



such that $\gamma(\perp) = \emptyset$, $\gamma(i) = \{i\}$, $i \in \mathbb{Z}$, and $\gamma(\top) = \mathbb{Z}$.

25. Classical infinitary abstractions

As noticed by [Brahmagupta](#), the sign analysis is not expressive enough to exactly determine the sign of expressions knowing the sign of its free variables. As shown by [18], interval analysis [8,1] will provide the desired answer. Interval analysis is based in the following lattice.



Because the lattice has infinite strictly increasing chains, the induction illustrated in Section 3 must be mechanized. This is the objective of widening and narrowing operators [8,1,19], see [20,21] for an introduction.

26. Conclusion

We have illustrated the basics of abstract interpretation by defining the semantics of expressions, their properties, a proof method, and a sign analysis.

Instead of designing the rule of sign empirically and then proving its soundness, we used the soundness requirement as a guideline for designing the abstract sign semantics by calculus.

This sign analysis discovers an abstract property of an arithmetic expression by computing in the abstract only. This may involve some loss of precision, which was the case for the sign analysis.

The sign semantics is finite so it is an easily implementable static analysis. For infinite abstract domains, widening and narrowing operators are necessary.

References

- [1] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252. ACM, 1977.
- [2] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282. ACM Press, 1979.
- [3] Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *J. Log. Comput.*, 2(4):511–547, 1992.
- [4] Kim Plofker. *Mathematics in India*. Princeton University Press, 2007.
- [5] Peter Naur. The design of the GIER ALGOL compiler. *BIT Numerical Mathematics*, 3:124–140 and 145–166, June 1963.
- [6] Peter Naur. Checking of operand types in ALGOL compilers. *BIT Numerical Mathematics*, 5:151–163, September 1965.
- [7] Michel Sintzoff. Calculating properties of programs by valuations on specific models. In *Proceedings of ACM Conference on Proving Assertions About Programs*, pages 203–207. ACM, 1972.
- [8] Patrick Cousot and Radhia Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.
- [9] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.
- [10] Dana S. Scott and Christopher Strachey. Towards a mathematical semantics for computer languages. Technical report PRG-6, Oxford University Computer Laboratory, August 1971.
- [11] International Organization for Standardization. Iso/iec 19761: Software engineering – cosmic: a functional size measurement method. March 2011.
- [12] Garrett Birkhoff. *Lattice Theory*. American Mathematical Society, Colloquium publications, Volume XXV, third edition edition, 1973.
- [13] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics. Series 2.*, 33(2):346–366, 1932.
- [14] David Cachera and David Pichardie. Programmation d’un interpréteur abstrait certifié en logique constructive. *Technique et Science Informatiques*, 30(4):381–408, 2011.
- [15] Jacques-Henri Jourdan, Vincent Laporte, Sandrine Blazy, Xavier Leroy, and David Pichardie. A formally-verified C static analyzer. In *POPL*, pages 247–259. ACM, 2015.
- [16] Susanne Graf and Hassen Saïdi. Verifying invariants using theorem proving. In *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 1996.
- [17] Gary A. Kildall. A unified approach to global program optimization. In *POPL*, pages 194–206. ACM Press, 1973.

- [18] Roberto Giacobazzi and Francesco Ranzato. Completeness in abstract interpretation: A domain perspective. In *AMAST*, volume 1349 of *Lecture Notes in Computer Science*, pages 231–245. Springer, 1997.
- [19] Patrick Cousot. Abstracting induction by extrapolation and interpolation. In *VMCAI*, volume 8931 of *Lecture Notes in Computer Science*, pages 19–42. Springer, 2015.
- [20] P. Cousot and R. Cousot. *A gentle introduction to formal verification of computer systems by abstract interpretation*, pages 1–29. NATO Science Series III: Computer and Systems Sciences. IOS Press, 2010.
- [21] Patrick Cousot. The calculational design of a generic abstract interpreter. In M. Broy and R. Steinbrüggen, editors, *Calculational System Design*. NATO ASI Series F. IOS Press, Amsterdam, 1999.

The author(s) of this publication is/are solely responsible for its content. This publication does not reflect the opinion of the publisher. The publisher cannot be held liable for any loss or damage that may occur because of this publication.