

The Symbolic Term Abstract Domain

Patrick Cousot

NYU, New York

`pcousot@cs.nyu.edu` `cs.nyu.edu/~pcousot`

Friday, December 11th, 2020

<https://sei.ecnu.edu.cn/tase2020/file/slidesPCousot-TASE-2020.pdf>

Example of Abstract Interpretation

- Brahmagupta, born c. 598 C.E., died after 665:

...

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.

...

$\wp(\mathbb{Z}) \longleftrightarrow \text{signs}$

Example of Abstract Interpretation

- Brahmagupta, born c. 598 C.E., died after 665:

...

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.

...

$\wp(\mathbb{Z}) \longleftrightarrow \text{signs}$

- signs, parity, intervals, octagons, linear equalities, polyhedra, etc. for numerical properties

Properties

- We represent **properties** by sets¹
- $\{0\}$ is “to be zero”
- \mathbb{N} is “to be positive”
- $5 \in \mathbb{N}$ is “5 is positive”
- $5 \notin \{0\}$ is “5 is not zero”
- $\{0\} \subseteq \mathbb{N}$ is “to be zero” implies “to be positive”

¹More general than, e.g., first order logic.

Mastering Complexity of Abstract Interpretation

- Program properties are complex
 - No single kind of property will fit all needs (counterexample: types)
 - It is necessary to decompose complex properties into a combination of simpler ones
- Abstract domain + Combination of abstract domains

²Patrick Cousot, Radhia Cousot, Laurent Mauborgne: Theories, solvers and static analysis by abstract interpretation. J. ACM 59(6): 31:1-31:56 (2012).

Mastering Complexity of Abstract Interpretation

- Program properties are complex
- No single kind of property will fit all needs (counterexample: types)
- It is necessary to decompose complex properties into a combination of simpler ones
→ Abstract domain + Combination of abstract domains
- Examples of abstract domain: negative/positive integers, odd/even integers, ..., theories in SMT solvers
- Examples of combination of abstract domains: reduced product (conjunction, including Nelson-Oppen composition procedure in SMT solvers)²

²Patrick Cousot, Radhia Cousot, Laurent Mauborgne: Theories, solvers and static analysis by abstract interpretation. J. ACM 59(6): 31:1-31:56 (2012).

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³
- The meaning of the abstract properties and operations is defined by a **concretization function** (into more concrete properties)

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³
- The meaning of the abstract properties and operations is defined by a **concretization function** (into more concrete properties)
- The operations include
 - **Logical operations:** \sqsubseteq (implication), \perp (false), \top (true), \sqcup (disjunction), \sqcap (conjunction), ... of abstract properties

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³
- The meaning of the abstract properties and operations is defined by a **concretization function** (into more concrete properties)
- The operations include
 - **Logical operations:** \sqsubseteq (implication), \perp (false), \top (true), \sqcup (disjunction), \sqcap (conjunction), ... of abstract properties
 - **Transformers:** to handle assignment, tests, ...

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³
- The meaning of the abstract properties and operations is defined by a **concretization function** (into more concrete properties)
- The operations include
 - **Logical operations:** \sqsubseteq (implication), \perp (false), \top (true), \sqcup (disjunction), \sqcap (conjunction), ... of abstract properties
 - **Transformers:** to handle assignment, tests, ...
 - **Inductors:** widening for extrapolation, narrowing for interpolation and **co-inductors** dual widening and narrowing, to handle iteration and recursion (co)-inductively

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Domain

- An abstract domain is an **order-theoretic algebraic structure** formalizing abstract properties and operations on these properties³
- The meaning of the abstract properties and operations is defined by a **concretization function** (into more concrete properties)
- The operations include
 - **Logical operations:** \sqsubseteq (implication), \perp (false), \top (true), \sqcup (disjunction), \sqcap (conjunction), ... of abstract properties
 - **Transformers:** to handle assignment, tests, ...
 - **Inductors:** widening for extrapolation, narrowing for interpolation and **co-inductors** dual widening and narrowing, to handle iteration and recursion (co)-inductively
 - **Combinators:** to send or receive information from other abstract domains

³Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282.

Abstract Interpreter

- A **formal abstract semantics** parameterized by an **abstract domain**
- Examples:
 - **collecting semantics**: specifies all possible executions of a program
 - **Astrée**⁴: over 50 abstract domains

⁴<https://www.absint.com/astree/index.htm>

Abstraction

- Specifies a **correspondence between** concrete and abstract **domains**
 - **Concretization** γ : concrete equivalent of an abstract property
Example: $\gamma(\textit{positive}) = \mathbb{N}$

Abstraction α : abstract approximation of a concrete property
Example: $\alpha(\{0, 7, 42\}) = \textit{positive}$
- Induces a **correspondence between** concrete and abstract **semantics**

Soundness

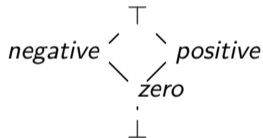
- An abstract property \bar{P} is a **sound abstraction** of a concrete property P if and only if the concrete property P implies the concretization $\gamma(\bar{P})$ of the abstract property \bar{P}
- Example: “to be positive” is a sound abstraction of “to be zero” since “to be zero” implies “to be positive”, formally $\gamma(\mathit{zero}) \subseteq \gamma(\mathit{positive})$, that is,
 $\{0\} \subseteq \mathbb{N} \triangleq \gamma(\mathit{positive})$
- Counter-example: “to be even” is a **unsound** abstraction of “to be positive” since $\gamma(\mathit{positive}) = \mathbb{N} \not\subseteq \{2k+1 \mid k \in \mathbb{Z}\} = \gamma(\mathit{even})$

Galois connections support best abstractions

- A **Galois connection** formalizes the situation when any concrete property has a **best/most precise abstraction**
- $\langle C, \leq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ if and only if
$$\forall x \in C. \forall y \in A. \alpha(x) \sqsubseteq y \Leftrightarrow x \leq \gamma(y)$$

Galois connections support best abstractions

- A **Galois connection** formalizes the situation when any concrete property has a **best/most precise abstraction**
- $\langle C, \leq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ if and only if
$$\forall x \in C. \forall y \in A. \alpha(x) \sqsubseteq y \Leftrightarrow x \leq \gamma(y)$$
- Example: “to be zero” $\{0\}$ has a best abstraction (*zero*) in



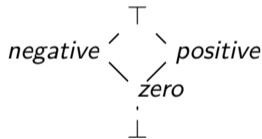
Galois connections support best abstractions

- A **Galois connection** formalizes the situation when any concrete property has a **best/most precise abstraction**

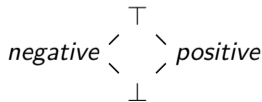
- $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ if and only if

$$\forall x \in C. \forall y \in A. \alpha(x) \sqsubseteq y \Leftrightarrow x \leq \gamma(y)$$

- Example: “to be zero” $\{0\}$ has a best abstraction (*zero*) in



- Counter-example: $\{0\}$ has no best abstraction in (*positive* and *negative* are sound)



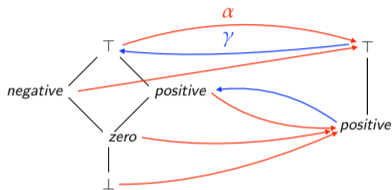
Galois retractions

- A Galois retraction/insertion $\langle C, \leq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ is a Galois connection $\langle C, \leq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ with α surjective (equivalently γ injective, (equivalently $\alpha \circ \gamma$ is the identity))

⁵Hasse diagram.

Galois retractions

- A Galois retraction/insertion $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ is a Galois connection $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ with α surjective (equivalently γ injective, (equivalently $\alpha \circ \gamma$ is the identity))
- Example:⁵

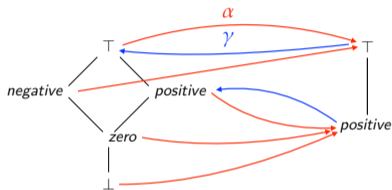


⁵Hasse diagram.

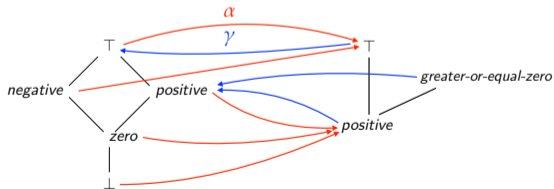
Galois retractions

- A Galois retraction/insertion $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ is a Galois connection $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ with α surjective (equivalently γ injective, (equivalently $\alpha \circ \gamma$ is the identity))

- Example:⁵



- Counter-example:



⁵Hasse diagram.

Order structure preservation

- Galois retractions $\langle C, \leq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$ preserve the order structure
- If $\langle C, \leq, 0, \bigvee \rangle$ is respectively a semi-lattice, lattice, complete partial order, or complete lattice, then $\langle A, \sqsubseteq, \perp, \bigsqcup \rangle$ is respectively a semi-lattice, lattice, complete partial order, or complete lattice
- $\forall x, y \in A . x \sqsubseteq y \Leftrightarrow \gamma(x) \leq \gamma(y)$
- $\forall X \in \wp(A) . \bigsqcup X = \alpha(\bigvee \gamma(X))$ if $\bigvee \gamma(X)$ exists in $\langle C, \leq \rangle$

Objective of the talk and online accompanying paper⁶

- Study an abstract domain (the [symbolic term abstract domain](#))

⁶<https://sei.ecnu.edu.cn/tase2020/file/Cousot-TASE-2020.pdf>

Ground and Symbolic Terms

In mathematical logic, Jacques Herbrand introduced

- *ground terms* [Herbrand, 1930b], Ch. 1⁷ to denote a basic mathematical object (for example, 0) or operation on objects (such as $+(1, 2)$), as well as
- *symbolic terms* that is *terms with variables* (where the variables x are unknowns standing for any ground term [Herbrand, 1930b], Ch. 2) (for example, $+(1, x)$).



Jacques HERBRAND (au centre)
au cours de l'ascension où il trouva la mort.



⁷English translation in [Herbrand, 1930a].

Symbolic Terms in Computer Science

Symbolic terms are of interest in various areas of Computer Science such as

- refutation theorem-proving based on the resolution rule of inference [Robinson, 1965, Robinson, 1979],
- satisfiability modulo theories [Barrett *et al.*, 2009],
- symbolic execution [King, 1976],
- type inference [Milner, 1978, Damas & Milner, 1982],
- logic and constraint programming [Colmerauer, 1985, Kowalski, 1988, Colmerauer & Roussel, 1993, Sterling & Shapiro, 1994, Clark & Åke Tärnlund, 1982], [Barbuti *et al.*, 1993, Hermenegildo *et al.*, 2003, Cousot *et al.*, 2009],
- pointer analysis in imperative [Steensgaard, 1996] or logic languages [Muthukumar & Hermenegildo, 1989],
- and so on.

The Complete Lattice of Symbolic Terms

- Gordon Plotkin [Plotkin, 1970, Plotkin, 1971] and John Reynolds [Reynolds, 1970] proved that the set of symbolic terms form a **complete lattice** with the *less general/subsumption* partial order \preceq^ν on terms.
- For example, $+(1, 2) \preceq^\nu +(1, y) \preceq^\nu +(x, y) \preceq^\nu z$.
- Generalizing this initial point view, our objective is to study the complete lattice of symbolic terms **by abstraction** of the powerset of ground terms.

Ground terms

- The *signature* \mathbf{F} defines a set of function symbols $f \setminus n$ (f for brevity), each one with an *arity* n , that is, a fixed number of parameters (0 for constants). The round parentheses “(”, “)” and comma “,” do not belong to \mathbf{F} .

$$\begin{aligned} f \setminus n, g \setminus n, h \setminus n &\in \mathbf{F} \setminus n && \text{signature } n \geq 0 \\ f, g, h &\in \mathbf{F} = \bigcup_{n \in \mathbb{N}} \mathbf{F} \setminus n \end{aligned}$$

- We assume that the signature \mathbf{F} has at least two different function symbols.
- Ground terms denote uninterpreted functional expressions.

$$\begin{aligned} \mathbf{t} \in \mathbf{T} &::= && \text{ground terms} \\ & \quad f \setminus 0 && \text{constants of arity } 0 \\ & \quad | \quad f \setminus n(\mathbf{t}_1, \dots, \mathbf{t}_n) && \text{term of arity } n \in \mathbb{N}^+ \end{aligned}$$

The Herbrand Universe

- The set \mathbf{T} of all ground terms is called the *Herbrand universe* with signature \mathbf{F} .
- Sets of ground terms form a complete lattice partially ordered by inclusion

$$\langle \wp(\mathbf{T}), \subseteq, \emptyset, \mathbf{T}, \cup, \cap \rangle \quad \text{sets of ground terms} \quad (1)$$

Terms with variables

- *Term with variables* (also called *symbolic term*)

$$\begin{array}{ll} \alpha, \beta, \gamma \in \mathbb{V}_t & \text{term variables} \\ \tau \in \mathbf{T}^\nu ::= & \text{terms with variables} \\ & f \setminus 0 \quad \text{constants} \\ & | \quad f \setminus n(\tau_1, \dots, \tau_n) \quad \text{term of arity } n \in \mathbb{N} \\ & | \quad \alpha \quad \text{term variable} \end{array} \quad (2)$$

- The round parentheses “(”, “)”, comma “,”, and variables $\alpha \in \mathbb{V}_t$ do not belong to **F**.

Term abstraction

- A *term with variables* (also called *symbolic term*) abstracts a set of terms.
- For example the set of ground terms $\{+(0, 1), +(0, +(1, 1)), +(0, +(1, +(1, 1))), +(0, +(1, +(1, +(1, 1))))), \dots\}$ can be abstracted by the term $+(0, \alpha)$ with variable α .
- The abstraction can be very imprecise.
- For example $\{0, +(0, 1)\}$ would be abstracted by variable α which concretization is the set of all ground terms.
- So the abstraction is precise enough only for set of terms with adequate regularity properties.

Terms with variables

- We write $\text{vars}[\tau]$ for the free variables of a term τ .

$$\begin{aligned}\text{vars}[\alpha] &\triangleq \{\alpha\} && \alpha \in V_t \\ \text{vars}[f(\tau_1, \dots, \tau_n)] &\triangleq \bigcup_{i=1}^n \text{vars}[\tau_i]\end{aligned}\tag{3}$$

Syntactic Replacement

- The *syntactic replacement*/substitution $\tau[\alpha \leftarrow \tau']$ on terms with variables τ replaces all instances of a variable α in the term τ by another term with variables τ' .

$$\begin{aligned} \alpha[\alpha \leftarrow \tau'] &\triangleq \tau' && (4) \\ \beta[\tau' \leftarrow \alpha] &\triangleq \beta && \text{when } \beta \neq \alpha \\ f(\tau_1, \dots, \tau_n)[\alpha \leftarrow \tau'] &\triangleq f(\tau_1[\alpha \leftarrow \tau'], \dots, \tau_n[\alpha \leftarrow \tau']) \end{aligned}$$

- This is a *syntactic notion* (similar to text editing by replacement)
- Examples
 - $+(x, y)[x \leftarrow y] = +(y, y)$
 - $+(x, x)[x \leftarrow -(y, z)] = +(-(y, z), -(y, z))$

Term Assignments

- An **assignment** maps variables to ground terms.

$$\rho \in \mathbf{P}^\nu \triangleq \mathbb{V}_t \rightarrow \mathbf{T} \quad \text{assignment} \quad (5)$$

- An assignment can be homomorphically extended to a term with variables, as follows:

$$\rho(f(\tau_1, \dots, \tau_n)) \triangleq f(\rho(\tau_1), \dots, \rho(\tau_n)) \quad (6)$$

- Example: if $\rho(x) = 1$ and $\rho(y) = 2$ then $\rho(+ (x, y)) = +(\rho(x), \rho(y)) = +(1, 2)$
- The intuition is that $\rho \in \mathbf{T}^\nu \rightarrow \mathbf{T}$ is the evaluation $\rho(\tau)$ of term τ by replacing variables α of τ by their value $\rho(\alpha)$ which is a ground term.
- This is a **semantic notion** (similar to the evaluation of expressions)

Variable assignments

- Variable assignment $\rho[\alpha \leftarrow t]$ can be used to change the value of a variable α to t

$$\rho[x \leftarrow v](x) \triangleq v \quad (7)$$

$$\rho[x \leftarrow v](y) \triangleq \rho(y) \quad \text{when } x \neq y$$

- Example: $\rho[x \leftarrow 1][y \leftarrow 2](+(x, y)) = +(1, 2)$

Syntactic Replacement versus Term Assignments

- We use the same notation for syntactic replacement (4) and variable assignment (7) because of the following lemma 1 showing that instantiation of syntactic replacement and environment assignment commute.
- Example:

$$\begin{aligned} & \rho(+ (x, 1) [x \leftarrow y]) \\ = & \rho(+ (y, 1)) \\ = & + (\rho(y), 1) \\ = & \rho[x \leftarrow \rho(y)](+ (x, 1)) \end{aligned}$$

Syntactic Replacement versus Term Assignments

- We use the same notation for syntactic replacement (4) and variable assignment (7) because of the following lemma 1 showing that instantiation of syntactic replacement and environment assignment commute.
- Example:

$$\begin{aligned} & \rho(+ (x, 1) [x \leftarrow y]) \\ = & \rho(+ (y, 1)) \\ = & + (\rho(y), 1) \\ = & \rho[x \leftarrow \rho(y)](+ (x, 1)) \end{aligned}$$

Lemma (1)

$$\rho(\tau[\alpha \leftarrow \tau']) = \rho[\alpha \leftarrow \rho(\tau')](\tau).$$


Proof of lemma 1.

By structural induction on τ . (see the paper). \square

Occur-check

- Let us call $\rho(\tau)$ the *ground instance* of τ for the assignment ρ .
- Unless it is reduced to a variable, a term with variables cannot have the same instance as any one of its variables (this is known as *occur-check*).


If x and $f(x)$ have the same instance then



Occur-check

- Let us call $\rho(\tau)$ the *ground instance* of τ for the assignment ρ .
- Unless it is reduced to a variable, a term with variables cannot have the same instance as any one of its variables (this is known as *occur-check*).

If x and $f(x)$ have the same instance then



Lemma (2)

For all variables $\alpha \in \text{vars}[\tau]$ of a term with variables $\tau \in \mathbf{P}^\nu \setminus V_t$, there is no assignment $\rho \in \mathbf{P}^\nu \triangleq V_t \rightarrow \mathbf{T}$ such that $\rho(\alpha) = \rho(\tau)$.

Proof of lemma 2.

See the paper. \square

The Symbolic Abstraction

- The symbolic abstraction abstracts a set of ground terms into a term with variables.
- The symbolic abstraction is easily defined by its concretization, that is, it's set of ground instances.

$$\begin{aligned} \mathit{ground}(\tau) &\triangleq \{\rho(\tau) \mid \rho \in \mathbf{P}^\nu\} \\ \mathit{ground}(\bar{\emptyset}^\nu) &\triangleq \emptyset \end{aligned} \tag{8}$$

- Since all terms with variables $\tau \in \mathbf{P}^\nu$ have a nonempty concretization $\mathit{ground}(\tau)$, we add the empty term $\bar{\emptyset}^\nu \notin \mathbf{P}^\nu$ to denote the empty set \emptyset with $\rho(\bar{\emptyset}^\nu) = \emptyset$.

The Subsumption Preorder

- We define the preorder \preceq^ν on terms with variables, called *subsumption*, as the inclusion of sets of their ground instances.⁸

$$(\tau \preceq^\nu \tau') \triangleq (\text{ground}(\tau) \subseteq \text{ground}(\tau')) \quad (9)$$

- This is a preorder $\langle \mathbf{T}^\nu \cup \{\bar{\emptyset}^\nu\}, \preceq^\nu \rangle$ with infimum $\bar{\emptyset}^\nu$.
- For example $f(a, b) \preceq^\nu f(\alpha, b) \preceq^\nu f(\alpha, \beta) \preceq^\nu \gamma$.
- The corresponding equivalence relation is \simeq^ν .
- The quotient is a partial order $\langle \mathcal{P}^H, \preceq_{\simeq^\nu} \rangle$

⁸This is different from Plotkin/Reynolds classical definition, but will be shown to be equivalent in theorem 13.

Semantic characterization of the subsumption partial order

Lemma (3)

Observe that for all terms with variables $\tau, \tau' \in \mathbf{T}^\nu$, we have $\tau \preceq^\nu \tau'$ if and only if $\forall \varrho \in \mathbf{P}^\nu . \exists \varrho' \in \mathbf{P}^\nu . \varrho(\tau) = \varrho'(\tau')$.

Proof of lemma 3.

The case of \emptyset is trivial. Otherwise,

$$\tau \preceq^\nu \tau'$$

$$\Leftrightarrow \text{ground}(\tau) \subseteq \text{ground}(\tau') \quad \{\text{def. (9) of } \preceq^\nu \}$$

$$\Leftrightarrow \{\varrho(\tau) \mid \varrho \in \mathbf{P}^\nu\} \subseteq \{\varrho'(\tau') \mid \varrho' \in \mathbf{P}^\nu\} \quad \{\text{def. (8) of } \text{ground}\}$$

$$\Leftrightarrow \forall \varrho \in \mathbf{P}^\nu . \exists \varrho' \in \mathbf{P}^\nu . \varrho(\tau) = \varrho'(\tau') \quad \{\text{def. } \subseteq \} \quad \square$$

The Subsumption Partial Order

- The subsumption preorder \preceq^ν on terms with variables (9) is $(\tau \preceq^\nu \tau') \triangleq (\text{ground}(\tau) \subseteq \text{ground}(\tau'))$
- The corresponding equivalence relation is \simeq^ν .
- The quotient is a partial order $\langle \mathcal{P}^H, \preceq_{\simeq^\nu} \rangle$ where

$$\begin{aligned}
 \tau \simeq^\nu \tau' &\triangleq \tau \preceq^\nu \tau' \wedge \tau' \preceq^\nu \tau & (10) \\
 \mathcal{P}^H &\triangleq (\mathbf{T}^\nu \cup \{\bar{\emptyset}^\nu\}) / \simeq^\nu \\
 &\triangleq \{[\tau]_{\simeq^\nu} \mid \tau \in \mathbf{T}^\nu \cup \{\bar{\emptyset}^\nu\}\} \\
 [\tau]_{\simeq^\nu} &\triangleq \{\tau' \mid \tau \simeq^\nu \tau'\} \\
 [\tau]_{\simeq^\nu} \preceq_{\simeq^\nu} [\tau']_{\simeq^\nu} &\triangleq \exists \bar{\tau} \in [\tau]_{\simeq^\nu}, \bar{\tau}' \in [\tau']_{\simeq^\nu} . \bar{\tau} \preceq^\nu \bar{\tau}'
 \end{aligned}$$

- For example $f(\alpha, \alpha) \simeq^\nu f(\beta, \beta)$ and $[f(\alpha, \alpha)]_{\simeq^\nu} = \{f(\gamma, \gamma) \mid \gamma \in \mathbb{V}_t\}$.

Syntactic Characterization of Term Equivalence

- A **renaming** is an assignment $\rho \in \mathbb{V}_t \rightsquigarrow \mathbb{V}_t$ between variables extended to terms with variables by (6), that is $\rho(f(\tau_1, \dots, \tau_n)) = f(\rho(\tau_1), \dots, \rho(\tau_n))$.
- Example: if $\rho(x) = y$ and $\rho(y) = z$ then $\rho(+ (x, y)) = + (y, z)$
- Equivalent terms are equal up to variable renaming.

Syntactic Characterization of Term Equivalence

- A **renaming** is an assignment $\rho \in \mathbb{V}_t \mapsto \mathbb{V}_t$ between variables extended to terms with variables by (6), that is $\rho(f(\tau_1, \dots, \tau_n)) = f(\rho(\tau_1), \dots, \rho(\tau_n))$.
- Example: if $\rho(x) = y$ and $\rho(y) = z$ then $\rho(+ (x, y)) = + (y, z)$
- Equivalent terms are equal up to variable renaming.

Lemma (4)

Equivalent terms have a bijective renaming of their variables and reciprocally, that is, $\forall \tau, \tau' \in \mathbf{T}^\nu . (\tau \simeq^\nu \tau') \Leftrightarrow (\exists \rho \in \text{vars}[\tau] \mapsto \text{vars}[\tau'] . \rho(\tau) = \tau')$.

Proof of lemma 4.

See the paper. \square

Comparison of Equivalence Classes

- The comparison of equivalence classes is equivalent to the comparison of the representatives of these classes.

Lemma (5)

$$[\tau_1]_{\simeq^\nu} \preceq_{\simeq^\nu} [\tau_2]_{\simeq^\nu} \Leftrightarrow \tau_1 \preceq^\nu \tau_2.$$

Proof of lemma 5.

$$\begin{aligned} & [\tau_1]_{\simeq^\nu} \preceq_{\simeq^\nu} [\tau_2]_{\simeq^\nu} \\ \Leftrightarrow & \exists \tau'_1 \in [\tau_1]_{\simeq^\nu}, \tau'_2 \in [\tau_2]_{\simeq^\nu} . \tau'_1 \preceq^\nu \tau'_2 && \text{\{def. (10) of } \preceq_{\simeq^\nu} \text{\}} \\ \Leftrightarrow & \exists \tau'_1, \tau'_2 . \tau'_1 \simeq^\nu \tau_1 \wedge \tau'_2 \simeq^\nu \tau_2 \wedge \tau'_1 \preceq^\nu \tau'_2 && \text{\{def. } [\tau]_{\simeq^\nu} \text{\}} \\ \Leftrightarrow & \exists \tau'_1, \tau'_2 . \tau_1 \preceq^\nu \tau'_1 \wedge \tau'_1 \preceq^\nu \tau'_2 \wedge \tau'_2 \preceq^\nu \tau_2 \wedge \tau'_1 \preceq^\nu \tau_1 \wedge \tau_2 \preceq^\nu \tau'_2 && \text{\{def. } \simeq^\nu \text{\}} \\ \Leftrightarrow & \tau_1 \preceq^\nu \tau_2 \\ & \begin{aligned} & \text{\{(\Rightarrow) transitivity} \\ & \text{\{(\Leftarrow) choosing } \tau'_1 = \tau_1, \tau'_2 = \tau_2, \text{ and reflexivity}\}} \end{aligned} \end{aligned}$$

Naming Scheme in the Symbolic Abstraction Function I

- The abstraction of $\{f(a, a), f(b, b), f(c, c)\}$ is $f(\alpha, \alpha)$ since the parameters of f are equal
- $\{f(a, b), f(b, a), f(a, a)\}$ is $f(\beta, \gamma)$ since the parameters of f are not always related.
- The abstraction function must select variables so as to identify equal parameters on all instances of f .
- For this purpose, we encode sets as families, for example, sequences $\langle f(a, a), f(b, b), f(c, c) \rangle$ and $\langle f(a, b), f(b, a), f(a, a) \rangle$.
- In the first case, the subterms all yield $\langle a, b, c \rangle$ which is abstracted by a variable say α .
- In the second case we get $\langle a, b, a \rangle$ encoded by β and $\langle b, a, a \rangle$ which is different so is encoded by a different variable γ .

Naming Scheme in the Symbolic Abstraction Function II

- Notice that the variable name does not matter and that the order in the sequences does not matter either
- So sets of ground terms encoded differently as index families will have the same abstraction, up to variable renaming via a bijection between variables; see lemma 6).
- We arbitrarily define a scheme to name sets of ground terms by a unique variable thanks to an injective function

$$\nu \in (\Delta \rightarrow \mathbf{T}) \mapsto \mathbb{V}_{\mathbf{t}} \quad (\text{naming scheme}) \quad (11)$$

assigning a variable $\nu(\{\mathbf{t}_i \mid i \in \Delta\})$ to any arbitrary family of ground terms $\{\mathbf{t}_i \mid i \in \Delta\}$.

- Injectivity ensures uniqueness, that is, different families of terms are abstracted by different variables.

The Symbolic Abstraction Function

- The abstraction is called the *least common generalization* (*lcg*).

$$lcg[\nu](\emptyset) \triangleq \bar{\emptyset}^\nu \tag{12}$$

$$\begin{aligned} lcg[\nu](\{f_i(\mathbf{t}_i^1, \dots, \mathbf{t}_i^{n_i}) \mid i \in \Delta\}) &\triangleq \\ &\text{if } \forall i, j \in \Delta . f_i = f_j = f \wedge n_i = n_j = n \text{ then} \\ &\quad \text{let } T^k = lcg[\nu](\{\mathbf{t}_i^k \mid i \in \Delta\}), k = 1, \dots, n \text{ in} \\ &\quad f(T^1, \dots, T^n) \\ &\text{else } \nu(\{f_i(\mathbf{t}_i^1, \dots, \mathbf{t}_i^{n_i}) \mid i \in \Delta\}) \end{aligned}$$

- If all the terms in the family have the same structure then the abstraction proceeds recursively else the family is abstracted by a variable.
- Equalities between all subterms of the family are preserved by the abstraction since the families of these subterms are abstracted by the same variable when they have different structures.

Example

- Assume that $\nu(\langle a, b \rangle) = \alpha$ and $\nu(\langle b, a \rangle) = \beta$, then

$$lcg[\nu](\langle f(g(a, a), h(b, b), a, b), f(g(b, b), h(a, a), b, a) \rangle)$$

$$= f(lcg[\nu](\langle g(a, a), g(b, b) \rangle), lcg[\nu](\langle h(b, b), h(a, a) \rangle), lcg[\nu](\langle a, b \rangle), lcg[\nu](\langle b, a \rangle))$$

$$= f(g(lcg[\nu](\langle a, b \rangle), lcg[\nu](\langle a, b \rangle)), h(lcg[\nu](\langle b, a \rangle), lcg[\nu](\langle b, a \rangle)), lcg[\nu](\langle a, b \rangle), lcg[\nu](\langle b, a \rangle))$$

$$= f(g(\nu(\langle a, b \rangle), \nu(\langle a, b \rangle)), h(\nu(\langle b, a \rangle), \nu(\langle b, a \rangle)), \nu(\langle a, b \rangle), \nu(\langle b, a \rangle))$$

$$= f(g(\alpha, \alpha), h(\beta, \beta), \alpha, \beta)$$

□

Independence from the Naming Scheme

Lemma (6)

The definition (12) of the symbolic abstraction $lcg[\nu]$ is independent of the naming scheme ν . If $\nu, \nu' \in (\Delta \rightarrow \mathbf{T}) \rightarrow \mathbb{V}_t$ then $\forall T \in \wp(\mathbf{T}) . lcg[\nu](T) \simeq^\nu lcg[\nu'](T)$.

Proof of lemma 6.

See the paper. \square

Galois Connection, Prolegomena I

- We now want to identify a Galois connection with abstraction $lcg[\nu]$ and concretization *ground*.
- Several preliminary results are needed.
- First, the symbolic abstraction $lcg[\nu]$ is \preceq^ν -increasing.

Galois Connection, Prolegomena I

- We now want to identify a Galois connection with abstraction $lcg[\nu]$ and concretization *ground*.
- Several preliminary results are needed.
- First, the symbolic abstraction $lcg[\nu]$ is \preceq^ν -increasing.

Lemma (7)

Let $\Delta \subseteq \Delta'$ be index sets and $\mathbf{t} \in \Delta' \rightarrow \mathbf{T}$ (and therefore $\{\mathbf{t}_i \mid i \in \Delta\} \subseteq \{\mathbf{t}_i \mid i \in \Delta'\}$). Then $lcg[\nu](\{\mathbf{t}_i \mid i \in \Delta\}) \preceq^\nu lcg[\nu](\{\mathbf{t}_i \mid i \in \Delta'\})$.

Proof of lemma 7.

See the paper. \square

Galois Connection, Prolegomena II

- The abstraction of a set of terms overapproximates any term of the set.

Lemma (8)

Let Δ be a nonempty set and $\mathbf{t} \in \Delta \rightarrow \mathbf{T}$ be a family of terms. Then

$$\forall j \in \Delta . \mathbf{t}_j \preceq^{\nu} \text{lcg}[\nu](\{\mathbf{t}_i \mid i \in \Delta\})$$

that is

$$\forall j \in \Delta . \exists \mathbf{e}' \in \mathbf{P}^{\nu} . \mathbf{t}_j = \mathbf{e}'(\text{lcg}[\nu](\{\mathbf{t}_i \mid i \in \Delta\})).$$

Proof of lemma 8.

See the paper. \square

Galois Connection, Prolegomena III

- The symbolic abstraction is an over approximation of properties of ground terms, that is, $ground \circ lcg[\nu]()$ is extensive.

Corollary (9)

If Δ is a nonempty set and $\{t_i \mid i \in \Delta\} \in \Delta \rightarrow \mathbf{T}$, then

$$\{t_i \mid i \in \Delta\} \subseteq ground(lcg[\nu](\{t_i \mid i \in \Delta\})).$$

Proof of corollary 9.

See the paper. \square

Galois Connection, Prolegomena IV

- The abstraction of the concretization of a term with variables loses no information.

Corollary (10)

For all $\tau \in \mathbf{T}^\nu$. $ground \circ lcg[\nu] \circ ground(\tau) = ground(\tau)$.

Proof of corollary 10.

See the paper. \square

Galois Connection, Prolegomena V

- In order to take into account the equivalence of terms with variables up to variable renaming (see lemma 4), we reason on the quotient partial order of terms $\langle \mathcal{P}^H, \preceq_{\simeq\nu} \rangle$.
- We extend the concretization (8) and the abstraction (12) to equivalence classes as follows.

$$\begin{aligned} lcg_{\simeq\nu}[\nu](\{\mathbf{t}_i \mid i \in \Delta\}) &\triangleq [lcg[\nu](\{\mathbf{t}_i \mid i \in \Delta\})]_{\simeq\nu} \\ ground_{\simeq\nu}([\tau]_{\simeq\nu}) &\triangleq ground(\tau). \end{aligned} \tag{13}$$

The Symbolic Term Galois Connection

Theorem (11)

For any naming scheme $\nu \in (\Delta \rightarrow \mathbf{T}) \rightarrow \mathbb{V}_{\mathbb{t}}$,

$$\langle \wp(\mathbf{T}), \subseteq \rangle \begin{array}{c} \xleftarrow{\text{ground}_{\sim\nu}} \\ \xrightarrow{\text{lcg}_{\sim\nu}[\nu]} \end{array} \langle \mathcal{P}^{\mathbf{H}}, \preceq_{\sim\nu} \rangle \quad (14)$$

This definition of the Galois retraction is independent of the choice of the naming scheme ν . □

Proof of theorem 11.

By def. of a Galois connection, we must prove that for all families of terms $\{\mathbf{t}_i \mid i \in \Delta\} \in \wp(\mathbf{T})$ and term with variables $\tau' \in \mathbf{T}^\nu \cup \{\bar{\emptyset}^\nu\}$,

$$lcg_{\simeq\nu}[\nu](\{\mathbf{t}_i \mid i \in \Delta\}) \preceq_{\simeq\nu} [\tau']_{\simeq\nu} \Leftrightarrow \{\mathbf{t}_i \mid i \in \Delta\} \subseteq ground_{\simeq\nu}([\tau']_{\simeq\nu}) \quad (15)$$

Moreover $ground_{\simeq\nu}$ is injective so (14) is a Galois retraction (also called Galois insertion).

By lemma 6, this definition of the Galois retraction (14) is independent of the choice of the naming scheme ν .

See details in the paper.

□

The Symbolic Abstract Domain is a Complete Lattice I

- The image $\alpha(C)$ of a complete lattice $\langle C, \leq, 0, 1, \vee, \wedge \rangle$ by a Galois retraction $\langle C, \leq \rangle \xleftarrow{\gamma} \langle A, \sqsubseteq \rangle \xrightarrow{\alpha}$ is a complete lattice $\langle A, \sqsubseteq, \perp, \top, \sqcup, \sqcap \rangle$ where
 - $\perp = \alpha(0)$
 - $\top = \alpha(1)$
 - $x \sqsubseteq y \Leftrightarrow \gamma(x) \leq \gamma(y)$
 - $\sqcup X = \alpha(\vee(\gamma(X)))$
 - $\sqcap X = \alpha(\wedge(\gamma(X)))$
- Therefore, the terms with variables form a complete lattice since they are the image of the complete lattice of properties of ground terms by the Galois retraction (14).

The Symbolic Abstract Domain is a Complete Lattice II

Corollary (12, symbolic abstract domain)

For any naming scheme $\nu \in (\Delta \rightarrow \mathbf{T}) \mapsto \mathbb{V}_{\mathbb{t}}$, $\langle \mathcal{P}^H, \preceq_{\sim\nu}, [\bar{\emptyset}^\nu]_{\sim\nu}, [\alpha]_{\sim\nu}, LCG_{\sim\nu}, GCI_{\sim\nu} \rangle$ is a complete lattice where

- $\alpha \in \mathbb{V}_{\mathbb{t}}$,
- the least upper bound is $LCG_{\sim\nu}(S) \triangleq lcg_{\sim\nu}[\nu](\bigcup \text{ground}_{\sim\nu}(S))$ (binary lcg for symbolic terms and $lcg_{\sim\nu}$ for term classes), and
- the greatest lower bound is $GCI_{\sim\nu}(S) \triangleq lcg_{\sim\nu}[\nu](\bigcap \text{ground}_{\sim\nu}(S))$ (binary gci and $gci_{\sim\nu}$).

This characterization of the lattice operations is independent of the naming scheme ν which is used.

On the Symbolic Abstraction

- Observe that ground terms $[\mathbf{t}]_{\simeq\nu} \in \mathcal{P}^H$ belongs to the abstract domain and abstract the concrete property $\{\mathbf{t}\}$ of being that ground term. Then $lcg_{\simeq\nu}[\nu](\{\mathbf{t}\}) = LCG_{\simeq\nu}(\{[\mathbf{t}]_{\simeq\nu}\})$, because, by (14), we have

$$\begin{aligned} & LCG_{\simeq\nu}(\{[\mathbf{t}]_{\simeq\nu}\}) \\ \triangleq & lcg_{\simeq\nu}[\nu](\bigcup \text{ground}_{\simeq\nu}(\{[\mathbf{t}]_{\simeq\nu}\})) \\ = & lcg_{\simeq\nu}[\nu](\bigcup \text{ground}_{\simeq\nu}(\{\mathbf{t}\})) \\ = & lcg_{\simeq\nu}[\nu](\bigcup \{\mathbf{t}\}) \\ = & lcg_{\simeq\nu}[\nu](\{\mathbf{t}\}). \end{aligned}$$

- This explains why the abstraction and the lub in the complete lattice have been given the same name.

The classical definition of the subsumption partial order using substitutions

Syntactic Subsumption Preorder

- The subsumption preorder \preceq^ν is classically defined syntactically, using substitutions [Robinson, 1979, pp. 180–188] (instead of (9)) [Plotkin, 1970, Plotkin, 1971, Reynolds, 1970].
- This classical syntactic definition is equivalent to the semantic definition (9) based on the interpretation of terms with variables as properties of ground terms.

Substitutions

- Assignments (5) record ground values of variables
- Substitutions can record symbolic values of some variables
- Substitutions are partial functions

$$\vartheta \in \Sigma \triangleq \mathcal{V}_t \rightharpoonup \mathbf{T}^\nu \quad (16)$$

mapping variables α in its domain $\text{dom}(\vartheta)$ to terms with variables $\vartheta(\alpha)$.

- Substitutions are extended to a total function $\vartheta \in \mathcal{V}_t \rightarrow \mathbf{T}^\nu$ and homomorphically to terms with variables, as follows

$$\begin{aligned} \vartheta(\alpha) &\triangleq \alpha && \text{when } \alpha \notin \text{dom}(\vartheta) \\ \vartheta(f(\tau_1, \dots, \tau_n)) &\triangleq f(\vartheta(\tau_1), \dots, \vartheta(\tau_n)) \end{aligned} \quad (17)$$

- Observe that the substitution is carried out simultaneously on all variable occurrences.

The classical characterization of the subsumption preorder using substitutions I

- The following theorem 13 shows that the syntactic and semantic definitions of subsumption are equivalent.

Theorem

$$\forall \tau_1, \tau_2 \in \mathbf{T}^\nu . [\tau_1]_{\approx^\nu} \preceq_{\approx^\nu} [\tau_2]_{\approx^\nu} \Leftrightarrow \exists \vartheta \in \Sigma . \vartheta(\tau_2) = \tau_1.$$

Proof of theorem 13.

See the paper. \square

- It follows that the subsumption lattice of [Reynolds, 1970, Plotkin, 1970, Plotkin, 1971, Huet, 1980] is the complete lattice considered in corollary 12 since the partial order is the same (although defined differently).

Results

- Concrete program property represented by a set of ground terms have a best abstraction by a term with variables
- The abstract is a Galois retraction
- The image of the powerset of ground terms by this Galois retraction is the complete lattice of symbolic terms by Gordon Plotkin [Plotkin, 1970, Plotkin, 1971] and John Reynolds [Reynolds, 1970]
- This approach yields algorithms together with their soundness proof by abstraction preservation [Cousot, 2021], section 48.8.

Methodology

This approach is typical of abstract interpretation:

- Define a **concrete semantics** $Programs \rightarrow Semantic\ domain$
- The **concrete properties** are $Programs \rightarrow \wp(Semantic\ domain)$
- Define an **abstraction** $\alpha \in \wp(Semantic\ domain) \rightarrow Abstract\ domain$ (for example, by a Galois connection)
- This induces a **sound abstract semantics** $Programs \rightarrow Abstract\ domain$ preserving some (if not all) properties of the concrete semantics
- Application to the formal design of **semantics**, **verification** method, **typing**, and **static analyzers**, see [Cousot, 2021]

Bibliography I

AÏT-KACI, HASSAN. 1984.

A lattice theoretic approach to computation based on a calculus of partially ordered type structures (property inheritance, nets, graph unification).

Phd thesis, Computer and Information Science Dept., University of Pennsylvania.

AÏT-KACI, HASSAN, PODELSKI, ANDREAS, & GOLDSTEIN, SETH COPEN. 1997.

Order sorted feature theory unification.

J. log. program., **30**(2), 99–124.

BARBUTI, ROBERTO, GIACOBAZZI, ROBERTO, & LEVI, GIORGIO. 1993.

A general framework for semantics-based bottom-up abstract interpretation of logic programs.

ACM trans. program. lang. syst., **15**(1), 133–181.

BARRETT, CLARK W., SEBASTIANI, ROBERTO, SESHIA, SANJIT A., & TINELLI, CESARE. 2009.

Satisfiability modulo theories.

Pages 825–885 of: Handbook of satisfiability.

Frontiers in Artificial Intelligence and Applications, vol. 185.

IOS Press.

CHURCH, ALONZO. 1940.

A formulation of the simple theory of types.

J. symb. log., **5**(2), 56–68.

CLARK, KEITH L., & ÅKE TÄRNLUND, STEN. 1982.

Logic programming.

Academic Press, New York, NY, US.

Bibliography II

- COLMERAUER, ALAIN. 1984.
Equations and inequations on finite and infinite trees.
Pages 85–99 of: FGCS.
OHMSHA Ltd. Tokyo and North-Holland.
- COLMERAUER, ALAIN. 1985.
Prolog in 10 figures.
Commun. ACM, 28(12), 1296–1310.
- COLMERAUER, ALAIN, & ROUSSEL, PHILIPPE. 1993.
The birth of prolog.
Pages 37–52 of: HOPL preprints.
ACM.
- COUSOT, PATRICK. 1997.
Types as abstract interpretations.
Pages 316–331 of: POPL.
ACM Press.
- COUSOT, PATRICK. 2021.
Principle of abstract interpretation.
MIT Press.
- COUSOT, PATRICK, COUSOT, RADHIA, & GIACOBAZZI, ROBERTO. 2009.
Abstract interpretation of resolution-based semantics.
Theor. comput. sci., 410(46), 4724–4746.

Bibliography III

DAMAS, LUÍS, & MILNER, ROBIN. 1982.

Principal type-schemes for functional programs.

Pages 207–212 of: *POPL*.

ACM Press.

HERBRAND, JACQUES. 1930a.

Investigations in proof theory.

Thesis, Université de Paris.

Ch. V of “Logical Writings”, Warren D. Goldfarb (Ed.), Springer Netherlands, 1971, pp. 44–202, English translation of [Herbrand, 1930b].

HERBRAND, JACQUES. 1930b.

Recherches sur la théorie de la démonstration.

Thèse, Université de Paris.

Ch. V of “Écrits logiques”, Jean Van Heijenoort (Ed.), Presses Universitaires de France, 1968, pp. 35–143.

HERMENEGILDO, MANUEL V., PUEBLA, GERMÁN, BUENO, FRANCISCO, & LÓPEZ-GARCÍA, PEDRO. 2003.

Program development using abstract interpretation (and the ciao system preprocessor).

Pages 127–152 of: *SAS*.

Lecture Notes in Computer Science, vol. 2694.

Springer.

HINDLEY, J. ROGER. 2008.

Basic simple type theory.

Cambridge Tracts in Theoretical Computer Science.

Cambridge University Press.

Bibliography IV

HUET, GÉRARD P. 1980.

Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems.

J. ACM, 27(4), 797–821.

KING, JAMES C. 1976.

Symbolic execution and program testing.

Commun. ACM, 19(7), 385–394.

KOWALSKI, ROBERT A. 1988.

The early years of logic programming.

Commun. ACM, 31(1), 38–43.

LERoy, XAVIER, DOLIGeZ, DAMIEN, FRISCH, ALAIN, GARRIGUE, JACQUES, RÉMY, DIDIER, & VOUILLOn, JÉRÔME. 2020.

The OCaml system, release 4.10, Documentation and user's manual.

Institut National de Recherche en Informatique et en Automatique.

MILNER, ROBIN. 1978.

A theory of type polymorphism in programming.

J. comput. syst. sci., 17(3), 348–375.

MUTHUKUMAR, KALYAN, & HERMENEGILDO, MANUEL V. 1989.

Determination of variable dependence information through abstract interpretation.

Pages 166–185 of: NAACP.

MIT Press.

Bibliography V

PLOTKIN, GORDON D. 1970.

A note on inductive generalization.

Pages 153—163 of: MELTZER, B.; MICHIE, D. (ed), *Machine intelligence*, vol. 5.
Edinburgh University Press.

PLOTKIN, GORDON D. 1971.

A further note on inductive generalization.

Pages 101—124 of: MELTZER, B.; MICHIE, D. (ed), *Machine intelligence*, vol. 6.
Edinburgh University Press.

REYNOLDS, JOHN C. 1970.

Transformational systems and the algebraic structure of atomic formulas.

Pages 135—151 of: MELTZER, B.; MICHIE, D. (ed), *Machine intelligence*, vol. 5.
Edinburgh University Press.

ROBINSON, JOHN ALAN. 1965.

A machine-oriented logic based on the resolution principle.

J. ACM, 12(1), 23–41.

ROBINSON, JOHN ALAN. 1979.

Logic: Form and function – the mechanization of deductive reasoning.

Artificial Intelligence.

Elsevier North-Holland.

STEENSGAARD, BJARNE. 1996.

Points-to analysis in almost linear time.

Pages 32–41 of: *POPL*.

ACM Press.

