« Forward Non-relational Finitary Static Analysis, Part I »

Patrick Cousot

Jerome C. Hunsaker Visiting Professor
Massachusetts Institute of Technology
Department of Aeronautics and Astronautics

cousot@mit.edu
www.mit.edu/~cousot

Course 16.399: "Abstract interpretation"

http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/


THESE

présentée à

Université Scientifique et Médicale de Grenoble
Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES MATHEMATIQUES

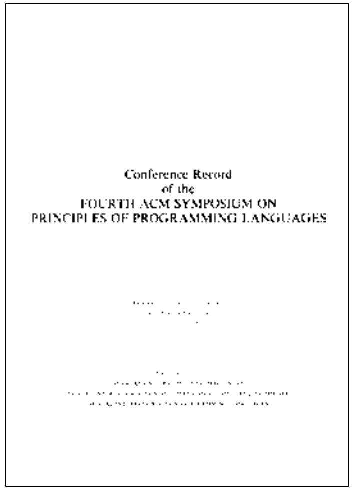
par

Patrick COUSOT

METHODES ITERATIVES DE CONSTRUCTION ET D'APPROXIMATION DE POINTS FIXES D'OPERATEURS MONOTONES SUR UN TREILLIS, ANALYSE SEMANTIQUE DES PROGRAMMES.

Thèse soutenue le 21 mars 1978 devant la Commission d'Examen :

Président : L. BOLLIET
Examinateurs : C. BENZAKEN, Ph. JORRAND, B. LORHO, C. PAIR, F. ROBERT, M. SINTZOFF


Conference Record
of the
FOURTH ACM SYMPOSIUM ON
PRINCIPLES OF PROGRAMMING LANGUAGES

Design of a non-relational abstract interpreter for SIL

Non-relational abstraction

An abstraction $\langle \wp(\mathbb{V} \mapsto \mathcal{V}),\ \subseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle L,\ \sqsubseteq\rangle$ is *non-relational*[1] if and only if for all $X \in \mathbb{V}$ there exists

$$\langle \wp(\mathcal{V}),\ \subseteq\rangle \xleftrightarrow[\alpha_X]{\gamma_X} \langle L,\ \sqsubseteq\rangle$$

such that

$$\alpha(P) = \bigsqcup_{X \in \mathbb{V}} \alpha_X(\{\rho(X) \mid \rho \in P\})$$

[1] Also called "attribute independent" after Muchnick and Jones.

and so, α is the composition of a *cartesian abstraction*

$$\langle \wp(\mathbb{V} \mapsto \mathcal{V}),\ \subseteq \rangle \xleftarrow[\lambda P \cdot \lambda X \cdot \{\rho(X) \mid \rho \in P\}]{\lambda R \cdot \{\rho \mid \forall X \in \mathbb{V} : \rho(X) \in R(X)\}} \langle \prod_{X \in \mathbb{V}} \wp(\mathcal{V}),\ \dot{\subseteq} \rangle$$

and a *componentwise abstraction*

$$\langle \prod_{X \in \mathbb{V}} \wp(\mathcal{V}),\ \dot{\subseteq} \rangle \xleftarrow[\lambda R \cdot \bigsqcup_{X \in \mathbb{V}} \alpha_X(R(X))]{\lambda Q \cdot \lambda X \cdot \gamma_X(Q)} \langle L,\ \sqsubseteq \rangle$$

Proof. – For all $P \in \wp(\mathbb{V} \mapsto \mathcal{V})$ and $R \in \wp(\mathcal{V})$

$$\begin{array}{lll} & \lambda X \cdot \{\rho(X) \mid \rho \in P\} \mathrel{\dot{\subseteq}} R & \\ \iff & \forall X \in \mathbb{V} : \{\rho(X) \mid \rho \in P\} \subseteq R(X) & \text{(pointwise def. of } \dot{\subseteq}\text{)} \end{array}$$

$$\begin{array}{lll} \iff & \forall X \in \mathbb{V} : \forall \rho \in P : \rho(X) \in R(X) & \text{(def. } \subseteq\text{)} \\ \iff & \forall \rho \in P : \forall X \in \mathbb{V} : \rho(X) \in R(X) & \text{(def. } \forall\text{)} \\ \iff & P \subseteq \{\rho \mid \forall X \in \mathbb{V} : \rho(X) \in R(X)\} & \text{(def. } \subseteq\text{)} \end{array}$$

– For all $R \in \prod_{X \in \mathbb{V}} \wp(\mathcal{V})$ and $Q \in L$

$$\begin{array}{lll} & \bigsqcup_{X \in \mathbb{V}} \alpha_X(R(X)) \sqsubseteq Q & \\ \iff & \forall X \in \mathbb{V} : \alpha_X(R(X)) \sqsubseteq Q & \text{(def. lubs)} \\ \iff & \forall X \in \mathbb{V} : R(X) \subseteq \gamma_X(Q) & \text{(def. Galois connection)} \\ \iff & R \mathrel{\dot{\subseteq}} \lambda X \in \mathbb{V} \cdot \gamma_X(Q) & \text{(pointwise def. } \dot{\subseteq}\text{)} \end{array}$$

– The composition is

$$\lambda R \cdot \bigsqcup_{X \in \mathbb{V}} \alpha_X(R(X)) \circ \lambda P' \cdot \lambda X' \cdot \{\rho(X') \mid \rho \in P'\}(P)$$

$$\begin{array}{lll} = & \lambda R \cdot \bigsqcup_{X \in \mathbb{V}} \alpha_X(R(X))(\lambda P' \cdot \lambda X' \cdot \{\rho(X') \mid \rho \in P'\}(P)) & \text{(def. composition } \circ\text{)} \\ = & \lambda R \cdot \bigsqcup_{X \in \mathbb{V}} \alpha_X(R(X))(\lambda X' \cdot \{\rho(X') \mid \rho \in P\}) & \text{(fonction application)} \\ = & \bigsqcup_{X \in \mathbb{V}} \alpha_X(\lambda X' \cdot \{\rho(X') \mid \rho \in P\})(X)) & \text{(fonction application)} \\ = & \bigsqcup_{X \in \mathbb{V}} \alpha_X(\{\rho(X) \mid \rho \in P\}) & \text{(fonction application)} \end{array}$$

$\square$

– Relations between values of variables are lost by α

Example of relational abstraction

- $P = \{\{x \to 1, y \to 2\}, \{x \to 5, y \to 7\}\}$
- $\alpha(P) = \{\{x \to a, y \to b\} \mid 1 \leq a \leq 5 \wedge 2 \leq b \leq 7 \wedge a \leq b\}$

Example of non-relational abstraction

– $P = \{\{x \to 1, y \to 2\}, \{x \to 5, y \to 7\}\}$
– $\alpha(P) = \{\{x \to a, y \to b\} \mid 1 \leq a \leq 5 \wedge 2 \leq b \leq 7\}$
– of the form

$$\begin{array}{rl} P & \Longrightarrow \{x \to \{1,5\}, y \to \{2,7\}\}^{2} \\ & \simeq \{\{x \to a, y \to b\} \mid a \in \{1,5\} \wedge b \in \{2,7\}\} \\ & = \{\{x \to a, y \to b\} \mid a \in \{1,5\}\} \cap \\ & \quad \{\{x \to a, y \to b\} \mid b \in \{2,7\}\} \\ & \Longrightarrow \{\{x \to a, y \to b\} \mid a \in [1,5]\} \cap \\ & \quad \{\{x \to a, y \to b\} \mid b \in [2,7]\}^{3} \\ & = \{\{x \to a, y \to b\} \mid 1 \leq a \leq 5 \wedge 2 \leq b \leq 7\} \end{array}$$

[2] Cartesian abstraction
[3] Interval abstraction

Cost/precision

– Relational abstractions of environment properties are usually precise but complex and costly
– Non-relational abstractions are simpler to program, cheaper but often too imprecise
– In practice a combination of global non-relational abstractions and local relational abstractions is used

Forward non-relational finitary reachability static analysis of SIL

Non-relational abstraction of sets of environments

– By definition, the non-relational abstractions are less precise than the Cartesian abstraction α_r:

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega), \subseteq \rangle \xrightleftharpoons[\alpha_r]{\gamma_r} \langle \mathbb{V} \mapsto \wp(\mathbb{I}_\Omega), \dot{\subseteq} \rangle$$

by defining

$$\begin{aligned} \alpha_r(R) &= \lambda \mathtt{X} \in \mathbb{V} \cdot \{\rho(\mathtt{X}) \mid \rho \in R\}, \\ \gamma_r(r) &= \{\rho \mid \forall \mathtt{X} \in \mathbb{V} : \rho(\mathtt{X}) \in r(\mathtt{X})\} \end{aligned}$$

and the pointwise ordering which is denoted with the dot notation

$$r \mathrel{\dot{\subseteq}} r' \stackrel{\text{def}}{=} \forall \mathtt{X} \in \mathbb{V} : r(\mathtt{X}) \subseteq r'(\mathtt{X}) .$$

– Now any Galois connection (1)

$$\langle \wp(\mathbb{I}_\Omega),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle \tag{1}$$

can be used to approximate the codomain

$$\langle \mathbb{V} \mapsto \wp(\mathbb{I}_\Omega),\ \dot{\subseteq}\rangle \underset{\alpha_c}{\overset{\gamma_c}{\leftrightarrows}} \langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq}\rangle$$

as follows

$$\begin{aligned} r \mathrel{\dot{\sqsubseteq}} r' &\stackrel{\text{def}}{=} \forall \mathtt{X} \in \mathbb{V} : r(\mathtt{X}) \sqsubseteq r'(\mathtt{X}),\\ \alpha_c(R) &\stackrel{\text{def}}{=} \alpha \circ R,\\ \gamma_c(r) &\stackrel{\text{def}}{=} \gamma \circ r, \end{aligned}$$

so that $\langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq},\ \dot{\bot},\ \dot{\top},\ \dot{\sqcup},\ \dot{\sqcap}\rangle$ is a complete lattice for the pointwise ordering $\dot{\sqsubseteq}$.

– We can now use the fact that the composition of Galois connections is a Galois connection, and so the composition of the nonrelational and codomain abstractions is

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega),\ \subseteq\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq}\rangle \tag{2}$$

where

$$\begin{aligned} \dot{\alpha}(R) &\stackrel{\text{def}}{=} \alpha_c \circ \alpha_r(R)\\ &= \lambda \mathtt{X} \in \mathbb{V} \cdot \alpha(\{\rho(\mathtt{X}) \mid \rho \in R\}), && (3)\\ \dot{\gamma}(r) &\stackrel{\text{def}}{=} \gamma_r \circ \gamma_c(r)\\ &= \{\rho \mid \forall \mathtt{X} \in \mathbb{V} : \rho(\mathtt{X}) \in \gamma(r(\mathtt{X}))\}\ . && (4) \end{aligned}$$

– If L has an infimum $\bot$ such that $\gamma(\bot) = \emptyset$, we observe that if $r \in \mathbb{V} \mapsto L$ has $\rho(\mathtt{X}) = \bot$ then $\dot{\gamma}(r) = \emptyset$.

– It follows that the abstract environments with some bottom component all represent the same concrete information ($\emptyset$).

– The abstract lattice can then be reduced to eliminate equivalent abstract environments (i.e. with same meaning):

$$\langle \mathbb{V} \mapsto \mathbb{I}_\Omega,\ \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{V} \stackrel{\perp}{\mapsto} L,\ \dot{\sqsubseteq}\rangle \tag{5}$$

where

$$\mathbb{V} \stackrel{\perp}{\mapsto} L \stackrel{\text{def}}{=} \{\rho \in \mathbb{V} \mapsto L \mid \forall \mathtt{X} \in \mathbb{V} : \rho(\mathtt{X}) \neq \bot\} \cup \{\lambda \mathtt{X} \in \mathbb{V} \cdot \bot\}\ .$$

Forward non-relational finitary reachability static analysis of arithmetic expressions

Definition of the forward collecting semantics of arithmetic expressions (lecture 14)

Recall the *forward/bottom-up collecting semantics* of an arithmetic expression from lecture 8:

$$\mathrm{Faexp} \in \mathrm{Aexp} \mapsto \wp(\mathrm{Env}[\![P]\!]) \xrightarrow{\sqcup} \wp(\mathbb{I}_\Omega),$$

$$\mathrm{Faexp}[\![A]\!]R \stackrel{\text{def}}{=} \{v \mid \exists \rho \in R : \rho \vdash A \Rightarrow v\} . \qquad (6)$$

such that:

$$\mathrm{Faexp}[\![A]\!]\Big(\bigcup_{k\in S} R_k\Big) = \bigcup_{k\in S} (\mathrm{Faexp}[\![A]\!]R_k)$$

$$\mathrm{Faexp}[\![A]\!]\emptyset = \emptyset .$$

Non relational predicate transformer abstraction

– Knowing an abstraction (1) of value properties and (5) of environment properties, we use a functional abstraction of monotonic predicate transformers:

$$\alpha^{\triangleright}(\Phi) \stackrel{\text{def}}{=} \alpha \circ \Phi \circ \dot{\gamma}, \qquad (7)$$

$$\gamma^{\triangleright}(\varphi) \stackrel{\text{def}}{=} \gamma \circ \varphi \circ \dot{\alpha}$$

so that [5]

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega) \xrightarrow{\mathrm{m}} \wp(\mathbb{I}_\Omega),\ \dot{\subseteq}\rangle \underset{\alpha^{\triangleright}}{\overset{\gamma^{\triangleright}}{\longleftrightarrow}} \langle (\mathbb{V} \mapsto L) \xrightarrow{\mathrm{m}} L,\ \dot{\sqsubseteq}\rangle . \qquad (8)$$

[5] The intuition is that for any abstract precondition $p \in L$, or its concrete equivalent $\dot{\gamma}(p) \in \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega)$, the abstract predicate transformer φ should provide an overestimate $\varphi(p)$ of the postcondition $\Phi(\gamma(p))$ defined by the concrete predicate transformer Φ.

Structural specification of the forward collecting semantics of arithmetic expressions (lecture 14)

$$\begin{array}{rcl}
\mathrm{Faexp}[\![\mathtt{n}]\!]R &=& \{\underline{\mathtt{n}}\}^{4}\\
\mathrm{Faexp}[\![\mathtt{X}]\!]R &=& R(\mathtt{X})\\
&& \text{where } R(\mathtt{X}) \stackrel{\text{def}}{=} \{\rho(\mathtt{X}) \mid \rho \in R\}\\
\mathrm{Faexp}[\![?]\!] &=& \mathbb{I}\\
\mathrm{Faexp}[\![\mathtt{u}A']\!]R &=& \underline{\mathtt{u}}^{C}(\mathrm{Faexp}[\![A']\!]R)\\
&& \text{where } \underline{\mathtt{u}}^{C}(V) \stackrel{\text{def}}{=} \{\underline{\mathtt{u}}(v) \mid v \in V\}\\
\mathrm{Faexp}[\![A_1 \,\mathtt{b}\, A_2]\!]R &=& \underline{\mathtt{b}}^{C}(\mathrm{Faexp}[\![A_1]\!], \mathrm{Faexp}[\![A_2]\!])R\\
&& \text{where } \underline{\mathtt{b}}^{C}(F_1, F_2)R \stackrel{\text{def}}{=} \{v_1 \,\underline{\mathtt{b}}\, v_2 \mid \exists \rho \in R : v_1 \in F_1(\{\rho\}) \wedge v_2 \in F_2(\{\rho\})\}
\end{array}$$

[4] For short, the case $\mathrm{Faexp}[\![A]\!]\emptyset = \emptyset$ is not recalled.

PROOF.

$$\begin{array}{ll}
\forall p \in L : \gamma(\varphi(p)) \supseteq \Phi(\dot{\gamma}(p)) & \text{(soundness requirement)}\\
\forall p \in L : \Phi(\dot{\gamma}(p)) \subseteq \gamma(\varphi(p)) & \text{(def. inverse } \supseteq \text{ of } \subseteq\text{)}\\
\forall p \in L : \alpha(\Phi(\dot{\gamma}(p))) \sqsubseteq \varphi(p) & \text{(def. Galois connection)}\\
\alpha \circ \Phi \circ \dot{\gamma} \mathrel{\dot{\sqsubseteq}} \varphi & \text{(def. } \dot{\sqsubseteq}\text{)}\\
\alpha^{\triangleright}(\Phi) \mathrel{\dot{\sqsubseteq}} \varphi & \text{(def. } \alpha^{\triangleright}\text{)} \quad (9)
\end{array}$$

□

– Choosing $\varphi \stackrel{\text{def}}{=} \alpha^{\triangleright}(\Phi)$ is therefore the best of the possible sound choices since it always provides the strongest abstract postcondition, whence, by monotony, the strongest concrete one.

Generic forward/top-downnon relational abstract semantics of arithmetic expressions

– The generic forward/top-down nonrelational abstract semantics of arithmetic expressions overapproximates the forward collecting semantics:

$$\mathrm{Faexp}^{\triangleright}[\![A]\!] \mathrel{\dot{\sqsupseteq}} \dot{\alpha}^{\triangleright}(\mathrm{Faexp}[\![A]\!]) \ . \tag{10}$$

– We distinguish two cases:

$$\begin{array}{ll} \mathrm{Faexp}^{\triangleright} \in \mathrm{Aexp} \mapsto (\mathbb{V} \longmapsto L) \stackrel{\mathrm{m}}{\longmapsto} L, & \text{when } \gamma(\perp) \neq \emptyset; \\ \mathrm{Faexp}^{\triangleright} \in \mathrm{Aexp} \mapsto (\mathbb{V} \stackrel{\perp}{\mapsto} L) \stackrel{\mathrm{m}}{\longmapsto} L, & \text{when } \gamma(\perp) = \emptyset \end{array}$$

Structural definition of the generic forward/top-down nonrelational abstract semantics of arithmetic expressions

$$\begin{array}{rcll} \mathrm{Faexp}^{\triangleright}[\![A]\!](\lambda \mathrm{Y} \cdot \perp) & \stackrel{\mathrm{def}}{=} & \perp \qquad \text{if } \gamma(\perp) = \emptyset & (11) \\ \mathrm{Faexp}^{\triangleright}[\![\mathrm{n}]\!]r & \stackrel{\mathrm{def}}{=} & \mathrm{n}^{\triangleright} & \\ \mathrm{Faexp}^{\triangleright}[\![\mathrm{X}]\!]r & \stackrel{\mathrm{def}}{=} & r(\mathrm{X}) & \\ \mathrm{Faexp}^{\triangleright}[\![?]\!]r & \stackrel{\mathrm{def}}{=} & ?^{\triangleright} & \\ \mathrm{Faexp}^{\triangleright}[\![\mathrm{u}\ A']\!]r & \stackrel{\mathrm{def}}{=} & \mathrm{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r) & \\ \mathrm{Faexp}^{\triangleright}[\![A_1\ \mathrm{b}\ A_2]\!]r & \stackrel{\mathrm{def}}{=} & \mathrm{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) & \end{array}$$

parameterized by the following forward abstract operations

$$\mathrm{n}^{\triangleright} = \alpha(\{\underline{\mathrm{n}}\}) \tag{12}$$

$$?^{\triangleright} \sqsupseteq \alpha(\mathbb{I}) \tag{13}$$

$$\mathrm{u}^{\triangleright}(p) \sqsupseteq \alpha(\{\underline{\mathrm{u}}\, v \mid v \in \gamma(p)\}) \tag{14}$$

$$\mathrm{b}^{\triangleright}(p_1, p_2) \sqsupseteq \alpha(\{v_1\ \underline{\mathrm{b}}\ v_2 \mid v_1 \in \gamma(p_1) \wedge v_2 \in \gamma(p_2)\}) \tag{15}$$

Calculational design of the structural definition of the generic forward/top-down nonrelational abstract semantics of arithmetic expressions

PROOF. Starting from the formal specification $\dot{\alpha}^{\triangleright}(\mathrm{Faexp}[\![A]\!])$, we derive an algorithm $\mathrm{Faexp}^{\triangleright}[\![A]\!]$ satisfying (10) by calculus

$$\begin{array}{lll} & \dot{\alpha}^{\triangleright}(\mathrm{Faexp}[\![A]\!]) & \\ = & \alpha \circ \mathrm{Faexp}[\![A]\!] \circ \dot{\gamma} & \text{⦃def. (7) of } \dot{\alpha}^{\triangleright}\text{⦄} \\ = & \lambda r \cdot \alpha(\mathrm{Faexp}[\![A]\!](\dot{\gamma}(r))) & \text{⦃def. of composition } \circ\text{⦄} \\ = & \lambda r \cdot \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Rightarrow v\}) & \text{⦃def. (6) of } \mathrm{Faexp}[\![A]\!]\text{⦄} \end{array}$$

If r is the infimum $\lambda \mathrm{Y} \cdot \perp$ where the infimum $\perp$ of L is such that $\gamma(\perp) = \emptyset$, then $\dot{\gamma}(r) = \emptyset$ whence:

$$\dot{\alpha}^{\triangleright}(\mathrm{Faexp}[\![A]\!])(\lambda \mathrm{Y} \cdot \perp)$$

$$= \alpha(\emptyset) \qquad \wr\text{def. (4) of } \dot{\gamma}\int$$

$$= \bot \qquad \wr\text{Galois connection (1) so that } \alpha(\emptyset) = \bot\int$$

When $r \neq \lambda \mathtt{Y} \cdot \bot$ or $\gamma(\bot) \neq \emptyset$, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])r$$

$$= (\lambda r \cdot \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Mapsto v\}))r$$

$$= \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Mapsto v\}) \qquad \wr\text{def. lambda expression}\int$$

and we proceed by induction on the arithmetic expression A.

1 —— When $A = \mathrm{n} \in \mathrm{Nat}$ is a number, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![\mathrm{n}]\!])r$$

$$= \alpha(\{\underline{\mathrm{n}}\}) \qquad \wr\text{def. } \alpha^{\triangleright} \text{ and } \mathrm{Faexp}[\![\mathrm{n}]\!]\int$$

$$= \mathrm{n}^{\triangleright} \qquad \wr\text{by defining } \mathrm{n}^{\triangleright} = \alpha(\{\underline{\mathrm{n}}\})\int$$

$$= \mathrm{Faexp}^{\triangleright}[\![\mathrm{n}]\!]r \qquad \wr\text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathrm{n}]\!]r \stackrel{\text{def}}{=} \mathrm{n}^{\triangleright}\int$$

2 —— When $A = \mathtt{X} \in \mathbb{V}$ is a variable, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![\mathtt{X}]\!])r$$

$$= \alpha(\{\rho(\underline{\mathtt{X}}) \mid \rho \in \dot{\gamma}(r)\}) \qquad \wr\text{def. } \alpha^{\triangleright} \text{ and } \mathrm{Faexp}[\![\mathtt{X}]\!]\int$$

$$= \alpha(\gamma(r(\mathtt{X}))) \qquad \wr\text{def. (4) of } \dot{\gamma}\int$$

$$\sqsubseteq r(\mathtt{X}) \qquad \wr\text{Galois connection (1) so that } \alpha \circ \gamma \text{ is reductive}\int$$

$$= \mathrm{Faexp}^{\triangleright}[\![\mathtt{X}]\!]r \qquad \wr\text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathtt{X}]\!]r \stackrel{\text{def}}{=} r(\mathtt{X})\int$$

3 —— When $A = ?$ is random, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![?]\!])r$$

$$= \alpha(\mathbb{I}) \qquad \wr\text{def. } \alpha^{\triangleright} \text{ and } \mathrm{Faexp}[\![?]\!]\int$$

$$\sqsubseteq ?^{\triangleright} \qquad \wr\text{by defining } ?^{\triangleright} \sqsupseteq \alpha(\mathbb{I})\int$$

$$= \mathrm{Faexp}^{\triangleright}[\![?]\!]r \qquad \wr\text{by defining } \mathrm{Faexp}^{\triangleright}[\![?]\!]r \stackrel{\text{def}}{=} ?^{\triangleright}\int$$

4 —— When $A = \mathrm{u}\, A'$ is a unary operation, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![\mathrm{u}\, A']\!])r$$

$$= \alpha(\mathrm{Faexp}[\![\mathrm{u}\, A']\!](\dot{\gamma}(r))) \qquad \wr\text{def. (7) of } \alpha^{\triangleright}\int$$

$$= \alpha(\underline{\mathrm{u}}^{\mathcal{C}}(\mathrm{Faexp}[\![A']\!](\dot{\gamma}(r))) \qquad \wr\text{structural def. } \mathrm{Faexp}[\![\mathrm{u}\, A']\!]\int$$

$\sqsubseteq$ $\wr$by monotony and (10) so that $\alpha^{\triangleright}(\mathrm{Faexp}[\![A']\!]) \mathrel{\dot\sqsubseteq} \mathrm{Faexp}^{\triangleright}[\![A']\!]$ by induction hypothesis, by def. of Galois connections so that $\mathrm{Faexp}[\![A']\!] \mathrel{\dot\subseteq} \gamma^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!])$, def. $\gamma^{\triangleright}(\varphi) \stackrel{\text{def}}{=} \gamma \circ \varphi \circ \dot{\alpha}$ so that $\mathrm{Faexp}[\![A']\!] \mathrel{\dot\subseteq} \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A']\!] \circ \dot{\alpha}$ and monotony with $\dot{\alpha} \circ \dot{\gamma}$ is reductive by (2) so that $\mathrm{Faexp}[\![A']\!](\dot{\gamma}(r)) \subseteq \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A']\!] \circ \dot{\alpha} \circ \dot{\gamma}(r) \subseteq \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A']\!](r)\int$

$$= \alpha(\underline{\mathrm{u}}^{\mathcal{C}}(\gamma \circ \mathrm{Faexp}^{\triangleright}[\![A']\!](r)))$$

$\sqsubseteq$ $\wr$by defining $\mathrm{u}^{\triangleright}$ such that $\mathrm{u}^{\triangleright}(p) \sqsupseteq \alpha(\{\underline{\mathrm{u}}\, v \mid v \in \gamma(p)\}) = \alpha(\underline{\mathrm{u}}(\gamma(p)))\int$

$$\mathrm{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r)$$

$$= \mathrm{Faexp}^{\triangleright}[\![\mathrm{u}\, A']\!]r \qquad \wr\text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathrm{u}\, A']\!]r \stackrel{\text{def}}{=} \mathrm{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r)\int$$

5 —— When $A = A_1 \mathrel{\mathrm{b}} A_2$ is a binary operation, we have

$$\alpha^{\triangleright}(\mathrm{Faexp}[\![A_1 \mathrel{\mathrm{b}} A_2]\!])r$$

$$= \alpha(\mathrm{Faexp}[\![A_1 \mathrel{\mathrm{b}} A_2]\!](\dot{\gamma}(r))) \qquad \wr\text{by def. } \alpha^{\triangleright}\int$$

$$= \alpha(\underline{\mathrm{b}}^{\mathcal{C}}(\mathrm{Faexp}[\![A_1]\!], \mathrm{Faexp}[\![A_2]\!])(\dot{\gamma}(r))) \qquad \wr\text{by structural def. of } \mathrm{Faexp}[\![A_1 \mathrel{\mathrm{b}} A_2]\!]\int$$

$= \alpha(\{v_1 \mathrel{\underline{\mathrm{b}}} v_2 \mid \exists \rho \in \dot{\gamma}(r) : v_1 \in \mathrm{Faexp}[\![A_1]\!](\{\rho\}) \wedge v_2 \in \mathrm{Faexp}[\![A_2]\!](\{\rho\})\})$ $\wr$by def. $\underline{\mathrm{b}}^{\mathcal{C}}\int$

$\sqsubseteq \alpha(\{v_1 \mathrel{\underline{\mathrm{b}}} v_2 \mid v_1 \in \mathrm{Faexp}[\![A_1]\!](\dot{\gamma}(r)) \wedge v_2 \in \mathrm{Faexp}[\![A_2]\!](\dot{\gamma}(r))\})$ $\wr$by monotony$\int$

$\sqsubseteq$ $\wr$by monotony and $\mathrm{Faexp}[\![A_i]\!](\dot{\gamma}(r)) \subseteq \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_i]\!](r)$, as above$\int$

$$\alpha(\{v_1 \mathrel{\underline{\mathrm{b}}} v_2 \mid v_1 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r) \wedge v_2 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_2]\!]r)\})$$

$\sqsubseteq$ $\wr$by defining $\mathrm{b}^{\triangleright}$ such that $\mathrm{b}^{\triangleright}(p_1, p_2) \sqsupseteq \alpha(\{v_1 \mathrel{\underline{\mathrm{b}}} v_2 \mid v_1 \in \gamma(p_1) \wedge v_2 \in \gamma(p_2)\})\int$

$$\mathrm{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r)$$

$$= \quad \wr \text{by defining } \mathrm{Faexp}^{\triangleright}[\![A_1 \; \mathsf{b} \; A_2]\!]r \stackrel{\mathrm{def}}{=} \mathsf{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \wr$$

$$\mathrm{Faexp}^{\triangleright}[\![A_1 \; \mathsf{b} \; A_2]\!]r \; .$$

In conclusion, we have designed the forward abstract interpretation $\mathrm{Faexp}^{\triangleright}$ of arithmetic expressions in such a way that it satisfies the soundness requirement (10) as summarized in (11). $\square$

Forward non-relational finitary reachability static analysis of boolean expressions

Definition of the forward collecting semantics of boolean expressions (lecture 14)

Recall the *collecting semantics* $\mathrm{Cbexp}[\![B]\!]R$ of a boolean expression B from course 8:

$$\mathrm{Cbexp} \in \mathrm{Bexp} \mapsto \wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\mathrm{Env}[\![P]\!]),$$

$$\mathrm{Cbexp}[\![B]\!]R \stackrel{\mathrm{def}}{=} \{\rho \in R \mid \rho \vdash B \Rightarrow \mathtt{tt}\} \; . \tag{16}$$

such that:

$$\mathrm{Cbexp}[\![B]\!]\left(\bigcup_{k \in S} R_k\right) = \bigcup_{k \in S} (\mathrm{Cbexp}[\![B]\!]R_k)$$

$$\mathrm{Cbexp}[\![B]\!]\emptyset = \emptyset \; .$$

Structural specification of the forward collecting semantics of boolean expressions (lecture 14)

$$\mathrm{Cbexp}[\![\mathtt{true}]\!]R = R$$

$$\mathrm{Cbexp}[\![\mathtt{false}]\!]R = \emptyset$$

$$\mathrm{Cbexp}[\![A_1 \; \mathsf{c} \; A_2]\!] = \underline{\mathsf{c}}^{\mathcal{C}} \; (\mathrm{Faexp}[\![A_1]\!], \mathrm{Faexp}[\![A_2]\!])R$$

where $\underline{\mathsf{c}}^{\mathcal{C}} \; (F, G)R \stackrel{\mathrm{def}}{=} \{\rho \in R \mid \exists v_1 \in F(\{\rho\}) \cap \mathbb{I} : \exists v_2 \in G(\{\rho\}) \cap \mathbb{I} : v_1 \; \underline{\mathsf{c}} \; v_2 = \mathtt{tt}\}$

$$\mathrm{Cbexp}[\![B_1 \;\&\; B_2]\!]R = \mathrm{Cbexp}[\![B_1]\!]R \cap \mathrm{Cbexp}[\![B_2]\!]R$$

$$\begin{aligned}\mathrm{Cbexp}[\![B_1 \mid B_2]\!]R = \; & (\mathrm{Cbexp}[\![B_1]\!]R \cap (\mathrm{Cbexp}[\![B_2]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_2)]\!]R)) \\ & \cup (\mathrm{Cbexp}[\![B_2]\!]R \cap (\mathrm{Cbexp}[\![B_1]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_1)]\!]R))\end{aligned}$$

Non-relational abstraction of the forward collecting semantics of boolean expressions

– Knowing abstractions (1) of value properties and (5) of environment properties, we use a functional abstraction of monotonic predicate transformers:

$$\ddot{\alpha}(\Phi) \stackrel{\text{def}}{=} \dot{\alpha} \circ \Phi \circ \dot{\gamma}, \tag{17}$$

$$\ddot{\gamma}(\varphi) \stackrel{\text{def}}{=} \dot{\gamma} \circ \varphi \circ \dot{\alpha}\ .$$

so that $(\mathbb{R} \stackrel{\text{def}}{=} \mathbb{V} \mapsto \mathbb{I}_{\Omega})$

$$\langle \wp(\mathbb{R}) \stackrel{\sqcup}{\longmapsto} \wp(\mathbb{R}),\ \dot{\subseteq} \rangle \underset{\ddot{\alpha}}{\overset{\ddot{\gamma}}{\leftrightarrows}} \langle (\mathbb{V} \mapsto L) \stackrel{\text{m}}{\longmapsto} (\mathbb{V} \mapsto L),\ \ddot{\sqsubseteq} \rangle \tag{18}$$

$$\Phi \mathrel{\dot{\subseteq}} \Psi \stackrel{\text{def}}{=} \forall R \in \wp(\mathbb{R}) : \Phi(R) \subseteq \Psi(R)$$

$$\varphi \mathrel{\ddot{\sqsubseteq}} \psi \stackrel{\text{def}}{=} \forall r \in \mathbb{V} \mapsto L : \varphi(r) \mathrel{\dot{\sqsubseteq}} \psi(r)$$

Generic forward/top-down non-relational abstract semantics of boolean expressions

We now consider the calculational design of the generic nonrelational abstract semantics of boolean expressions

$$\text{Abexp} \in \text{Bexp} \mapsto (\mathbb{V} \mapsto L) \stackrel{\text{m}}{\longmapsto} (\mathbb{V} \mapsto L)$$

which is a sound overapproximation in that

$$\text{Abexp}[\![B]\!] \mathrel{\ddot{\sqsupseteq}} \ddot{\alpha}(\text{Cbexp}[\![B]\!]) \tag{19}$$

Structural definition of the generic forward/top-down nonrelational abstract semantics of boolean expressions

$$\begin{aligned}
\text{Abexp}[\![B]\!]\lambda \mathtt{Y}\cdot\bot &\stackrel{\text{def}}{=} \lambda \mathtt{Y}\cdot\bot \qquad \text{if } \gamma(\bot) = \emptyset \qquad (20)\\
\text{Abexp}[\![\mathtt{true}]\!]r &\stackrel{\text{def}}{=} r\\
\text{Abexp}[\![\mathtt{false}]\!]r &\stackrel{\text{def}}{=} \lambda \mathtt{Y}\cdot\bot\\
\text{Abexp}[\![A_1 \mathrel{\mathtt{c}} A_2]\!]r &\stackrel{\text{def}}{=} \check{\mathtt{c}}(\text{Faexp}^{\triangleright}[\![A_1]\!]r, \text{Faexp}^{\triangleright}[\![A_2]\!]r)\\
\text{Abexp}[\![B_1 \mathbin{\&} B_2]\!]r &\stackrel{\text{def}}{=} \text{Abexp}[\![B_1]\!]r \mathbin{\dot{\sqcap}} \text{Abexp}[\![B_2]\!]r\\
\text{Abexp}[\![B_1 \mid B_2]\!]r &\stackrel{\text{def}}{=} \text{Abexp}[\![B_1]\!]r \mathbin{\dot{\sqcup}} \text{Abexp}[\![B_2]\!]r
\end{aligned}$$

parameterized by the following abstract comparison operations $\check{\mathtt{c}}$, $\mathtt{c} \in \{<, =\}$ on L

$$\check{\mathtt{c}}(p_1, p_2)r \mathrel{\dot{\sqsupseteq}} (\!|\ \exists v_1 \in \gamma(p_1) : \exists v_2 \in \gamma(p_2) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathtt{c}}} v_2 = \mathtt{tt}\ ?\ r \ \text{¿}\ \dot{\bot} |\!)$$

Calculational design of the structural definition of the generic forward/top-down nonrelational abstract semantics of boolean expressions

PROOF. We derive $\text{Abexp}[\![B]\!]$ as follows

$$\begin{aligned}
&\ddot{\alpha}(\text{Cbexp}[\![B]\!])\\
= {} & \lambda r \in \mathbb{V} \mapsto L \cdot \dot{\alpha}(\text{Cbexp}[\![B]\!]\dot{\gamma}(r)) && \text{(def. (17) of } \ddot{\alpha}\text{)}\\
= {} & \lambda r \in \mathbb{V} \mapsto L \cdot \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash B \Rrightarrow \mathtt{tt}\}) && \text{(def. (16) of Cbexp)}
\end{aligned}$$

If r is the infimum $\lambda \mathtt{Y}\cdot\bot$ and the infimum $\bot$ of L is such that $\gamma(\bot) = \emptyset$ then $\dot{\gamma}(r) = \emptyset$. In this case

$$\begin{aligned}
&\ddot{\alpha}(\text{Cbexp}[\![B]\!]\lambda \mathtt{Y}\cdot\bot)\\
= {} & \dot{\alpha}(\emptyset) && \text{(def. (4) of } \dot{\gamma}\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= \lambda \mathrm{Y} \cdot \bot && \wr\text{def. (3) of } \ddot{\alpha}\int
\end{aligned}
$$

Otherwise $r \neq \lambda \mathrm{Y} \cdot \bot$ or $\gamma(\bot) \neq \emptyset$, and we proceed by induction on the boolean expression B.

6 —— When $B = \mathtt{true}$ is true, we have

$$
\begin{aligned}
&\ddot{\alpha}(\mathrm{Cbexp}[\![\mathtt{true}]\!])r \\
= \;& \dot{\alpha}(\mathrm{Cbexp}[\![\mathtt{true}]\!](\dot{\gamma}(r))) && \wr\text{def. (17) of } \ddot{\alpha}\int \\
= \;& \dot{\alpha}(\dot{\gamma}(r)) && \wr\text{by structural def. of } \mathrm{Cbexp}[\![\mathtt{true}]\!]\int \\
\dot{\sqsubseteq} \;& r && \wr\dot{\alpha}\circ\dot{\gamma} \text{ is reductive (18)}\int \\
= \;& \mathrm{Abexp}[\![\mathtt{true}]\!]r && \wr\text{by defining } \mathrm{Abexp}[\![\mathtt{true}]\!]r \stackrel{\text{def}}{=} r\int
\end{aligned}
$$

7 —— When $B = \mathtt{false}$ is false, we have

$$
\ddot{\alpha}(\mathrm{Cbexp}[\![\mathtt{false}]\!])r
$$

$$
\begin{aligned}
= \;& \dot{\alpha}(\mathrm{Cbexp}[\![\mathtt{false}]\!](\dot{\gamma}(r))) && \wr\text{def. (17) of } \ddot{\alpha}\int \\
= \;& \dot{\alpha}(\emptyset) && \wr\text{def. (2) of } \dot{\alpha}\int \\
= \;& \lambda \mathrm{Y} \cdot \bot && \wr\text{def. (3) of } \dot{\alpha}\int \\
= \;& \mathrm{Abexp}[\![\mathtt{false}]\!]r && \wr\text{by defining } \mathrm{Abexp}[\![\mathtt{false}]\!]r \stackrel{\text{def}}{=} \lambda \mathrm{Y} \cdot \bot\int
\end{aligned}
$$

8 —— When $B = A_1 \mathrel{\mathsf{c}} A_2$ is an arithmetic comparison, we have

$$
\begin{aligned}
&\ddot{\alpha}(\mathrm{Cbexp}[\![A_1 \mathrel{\mathsf{c}} A_2]\!])r \\
= \;& \dot{\alpha}(\mathrm{Cbexp}[\![A_1 \mathrel{\mathsf{c}} A_2]\!](\dot{\gamma}(r))) && \wr\text{def. (17) of } \ddot{\alpha}\int \\
= \;& \dot{\alpha}(\underline{\mathsf{c}}^{C}(\mathrm{Faexp}[\![A_1]\!], \mathrm{Faexp}[\![A_2]\!])(\dot{\gamma}(r))) && \wr\text{structural def. of } \mathrm{Cbexp}[\![A_1 \mathrel{\mathsf{c}} A_2]\!]\int \\
= \;& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \mathrm{Faexp}[\![A_1]\!](\{\rho\}) \cap \mathbb{I} : \exists v_2 \in \mathrm{Faexp}[\![A_2]\!](\{\rho\}) \cap \mathbb{I} : \\
& v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt}\}) && \wr\text{by def. } \underline{\mathsf{c}}^{C}\int \\
\dot{\sqsubseteq} \;& \wr\text{by (10) so that } \alpha^{\triangleright}(\mathrm{Faexp}[\![A_i]\!]) \mathrel{\dot{\sqsubseteq}} \mathrm{Faexp}^{\triangleright}[\![A_i]\!] \text{ by induction hypothesis,} \\
& \text{(8) whence } \mathrm{Faexp}[\![A]\!] \mathrel{\dot{\subseteq}} \gamma^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A]\!]) \text{ and by monotony of } \dot{\alpha}\int \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!])(\{\rho\}) \cap \mathbb{I} : \exists v_2 \in \gamma^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_2]\!])(\{\rho\}) \cap \\
& \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt}\})
\end{aligned}
$$

$$
\begin{aligned}
= \;& \wr\text{def. (7) of } \gamma^{\triangleright} \stackrel{\text{def}}{=} \gamma \circ \varphi \circ \dot{\alpha}\int \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_1]\!] \circ \dot{\alpha}(\{\rho\}) \cap \mathbb{I} : \exists v_2 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_2]\!] \circ \\
& \dot{\alpha}(\{\rho\}) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt}\}) \\
= \;& \wr\text{monotony and } \{\rho\} \subseteq \dot{\gamma}(r)\int \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_1]\!] \circ \dot{\alpha}(\dot{\gamma}(r)) \cap \mathbb{I} : \exists v_2 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_2]\!] \circ \\
& \dot{\alpha}(\dot{\gamma}(r)) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt}\}) \\
\dot{\sqsubseteq} \;& \wr\dot{\alpha}\circ\dot{\gamma} \text{ reductive and monotony}\int \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_1]\!] \circ (r) \cap \mathbb{I} : \exists v_2 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_2]\!](r) \cap \mathbb{I} : \\
& v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt}\}) \\
= \;& (\!|\, \exists v_1 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_1]\!] \circ (r) \cap \mathbb{I} : \exists v_2 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_2]\!](r) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt} \mathrel{?} \\
& \dot{\alpha}(\dot{\gamma}(r)) \mathrel{\vdots} \dot{\alpha}(\emptyset) \,|\!) \\
\dot{\sqsubseteq} \;& \wr\dot{\alpha}\circ\dot{\gamma} \text{ reductive}\int \\
= \;& (\!|\, \exists v_1 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_1]\!] \circ (r) \cap \mathbb{I} : \exists v_2 \in \gamma \circ \mathrm{Faexp}^{\triangleright}[\![A_2]\!](r) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt} \mathrel{?} \\
& r \mathrel{\vdots} \dot{\bot} \,|\!) \\
\dot{\sqsubseteq} \;& \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!], \mathrm{Faexp}^{\triangleright}[\![A_2]\!])r
\end{aligned}
$$

$$
\begin{aligned}
& \wr\text{where } \check{\mathsf{c}}(p_1, p_2)r \mathrel{\dot{=}} (\!|\, \exists v_1 \in \gamma(p_1) : \exists v_2 \in \gamma(p_2) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathtt{tt} \mathrel{?} r \mathrel{\vdots} \\
& \dot{\bot} \,|\!)\int
\end{aligned}
$$

9 —— When $B = B_1 \mathbin{\&} B_2$ is a conjunction, we have

$$
\begin{aligned}
&\ddot{\alpha}(\mathrm{Cbexp}[\![B_1 \mathbin{\&} B_2]\!])r \\
= \;& \dot{\alpha}(\mathrm{Cbexp}[\![B_1 \mathbin{\&} B_2]\!](\dot{\gamma}(r))) && \wr\text{def. (17) of } \ddot{\alpha}\int \\
= \;& \wr\text{def. } \mathrm{Cbexp}[\![B_1 \mathbin{\&} B_2]\!]\int \\
& \dot{\alpha}(\mathrm{Cbexp}[\![B_1]\!](\dot{\gamma}(r)) \cap \mathrm{Cbexp}[\![B_2]\!](\dot{\gamma}(r))) \\
\dot{\sqsubseteq} \;& \wr\text{by monotony}\int \\
= \;& \dot{\alpha}(\mathrm{Cbexp}[\![B_1]\!](\dot{\gamma}(r))) \mathbin{\dot{\sqcap}} \dot{\alpha}(\mathrm{Cbexp}[\![B_2]\!](\dot{\gamma}(r))) \\
= \;& \wr\text{def. (17) of } \ddot{\alpha}\int \\
& \ddot{\alpha}(\mathrm{Cbexp}[\![B_1]\!])r \mathbin{\dot{\sqcap}} \ddot{\alpha}(\mathrm{Cbexp}[\![B_2]\!])r \\
\dot{\sqsubseteq} \;& \wr\text{induction hypothesis (19) and } \dot{\sqcap} \text{ monotone}\int \\
& \mathrm{Abexp}[\![B_1]\!]r \mathbin{\dot{\sqcap}} \mathrm{Abexp}[\![B_2]\!]r
\end{aligned}
$$

$= \quad \wr$ by defining $\mathrm{Abexp}[\![B_1 \,\&\, B_2]\!]r \stackrel{\mathrm{def}}{=} \mathrm{Abexp}[\![B_1]\!]r \;\dot{\sqcap}\; \mathrm{Abexp}[\![B_2]\!]r \wr$

$\mathrm{Abexp}[\![B_1 \,\&\, B_2]\!]r$.

10 —— The case $B = B_1 \mid B_2$ of disjunction is similar by first overapproximating $\mathrm{Cbexp}[\![B_1|B_2]\!]R = (\mathrm{Cbexp}[\![B_1]\!]R \cap (\mathrm{Cbexp}[\![B_2]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_2)]\!]R)) \cup (\mathrm{Cbexp}[\![B_2]\!]R \cap (\mathrm{Cbexp}[\![B_1]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_1)]\!]R))$ by $\mathrm{Cbexp}[\![B_1]\!]R \cup \mathrm{Cbexp}[\![B_2]\!]R$.

In conclusion, we have designed the abstract interpretation Abexp of boolean expressions in such a way that it satisfies the soundness requirement (19) as summarized in (20). □

Definition of the forward reachability collecting semantics of commands (lecture 14)

The forward reachability collecting semantics $\mathrm{Rcom}[\![C]\!]R$ of a command $C \in \mathrm{Com}$ (of a given program P) specifies the set of reachable states during any execution of C starting at its starting point in any of the environments satisfying the precondition R.

$$\mathrm{Rcom} \in \mathrm{Com} \mapsto \wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} (\mathrm{in}_P[\![C]\!] \mapsto \wp(\mathrm{Env}[\![P]\!]))$$

$$\mathrm{Rcom}[\![C]\!]R\ell \stackrel{\mathrm{def}}{=} \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star[\![C]\!]\}$$

Forward non-relational finitary reachability static analysis of commands

Structural specification of the forward reachability collecting semantics of commands (lecture 14)

$\mathrm{Rcom}[\![\mathtt{skip}]\!]R\ell = R$

$\mathrm{Rcom}[\![\mathtt{X} := A]\!]R\ell = \mathsf{match}\ \ell\ \mathsf{with}$
$\mid \mathsf{at}_P[\![\mathtt{X} := A]\!] \rightarrow R$
$\mid \mathsf{after}_P[\![\mathtt{X} := A]\!] \rightarrow \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in (\mathrm{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}$

$\mathrm{Rcom}[\![C]\!]R\ell$ where $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi} =$
$\quad \mathsf{match}\ \ell\ \mathsf{with}$
$\qquad \mid \mathsf{at}_P[\![C]\!] \rightarrow R$
$\qquad \mid \mathsf{in}_P[\![S_t]\!] \rightarrow \mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]R)\ell$
$\qquad \mid \mathsf{in}_P[\![S_f]\!] \rightarrow \mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]R)\ell$
$\qquad \mid \mathsf{after}_P[\![C]\!] \rightarrow \mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]R)(\mathsf{after}_P[\![S_t]\!])$
$\qquad\qquad \cup\, \mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]R)(\mathsf{after}_P[\![S_f]\!])$

$$
\begin{array}{l}
\mathrm{Rcom}[\![C]\!]R\ell \text{ where } C = \texttt{while } B \texttt{ do } S \texttt{ od} = \\
\quad \text{let } I = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot R \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!]) \text{ in} \\
\qquad \text{match } \ell \text{ with} \\
\qquad \mid \mathrm{at}_P[\![C]\!] \to I \\
\qquad \mid \mathrm{in}_P[\![S]\!] \to \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]I)(\ell) \\
\qquad \mid \mathrm{after}_P[\![C]\!] \to \mathrm{Cbexp}[\![T(\neg(B))]\!]R)I \\
\mathrm{Rcom}[\![C\ ;\ S]\!]R\ell = \text{match } \ell \text{ with} \\
\quad \mid \mathrm{in}_P[\![C]\!] \to \mathrm{Rcom}[\![C]\!]R\ell \\
\quad \mid \mathrm{in}_P[\![S]\!] \to \mathrm{Rcom}[\![S]\!](\mathrm{Rcom}[\![C]\!]R(\mathrm{after}_P[\![C]\!]))\ell \\
\mathrm{Rcom}[\![S\ ;;]\!]R\ell = \mathrm{Rcom}[\![S]\!]R\ell
\end{array}
$$

Nonrelational abstraction of the forward reachability collecting semantics of commands

$$
\langle \wp(\mathbb{R}) \stackrel{\sqcup}{\longmapsto} (\mathrm{in}_P[\![C]\!] \mapsto \wp(\mathbb{R})),\ \ddot{\subseteq}\rangle \underset{\alpha[\![C]\!]}{\overset{\gamma[\![C]\!]}{\leftrightarrows}} \langle (\mathbb{V} \mapsto L) \stackrel{\mathrm{m}}{\longmapsto} (\mathrm{in}_P[\![C]\!] \mapsto (\mathbb{V} \mapsto L)),\ \ddot{\dot{\sqsubseteq}}\rangle
$$

where $\mathbb{R} \stackrel{\text{def}}{=} \mathbb{V} \mapsto \mathbb{I}_\Omega$ and

$$
\alpha[\![C]\!]\varphi \stackrel{\text{def}}{=} \lambda r \cdot \lambda \ell \cdot \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))
$$

$$
\gamma[\![C]\!]\psi \stackrel{\text{def}}{=} \lambda R \cdot \lambda \ell \cdot \dot{\gamma}(\psi(\dot{\alpha}(R))(\ell))
$$

Proof.

$$
\alpha[\![C]\!]\varphi\ \ddot{\dot{\sqsubseteq}}\ \psi
$$

$$
\begin{array}{ll}
\Longleftrightarrow & \lambda r \cdot \lambda \ell \cdot \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))\ \ddot{\sqsubseteq}\ \psi \\
\Longleftrightarrow & \forall r : \forall \ell : \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))\ \dot{\sqsubseteq}\ \psi(r)(\ell) \\
\Longleftrightarrow & \forall r : \forall \ell : \varphi(\dot{\gamma}(r))(\ell) \subseteq \dot{\gamma}(\psi(r)(\ell)) \\
\Longrightarrow & \forall R : \forall \ell : \varphi(\dot{\gamma}(\dot{\alpha}(R)))(\ell) \subseteq \dot{\gamma}(\psi(\dot{\alpha}(R))(\ell)) \\
\Longrightarrow & \qquad \wr \dot{\gamma} \circ \dot{\alpha} \text{ is extensive, } \varphi \text{ is monotone} \int \\
 & \forall R : \forall \ell : \varphi(R)(\ell) \subseteq \dot{\gamma}(\psi(\dot{\alpha}(R))(\ell)) \\
\Longleftrightarrow & \varphi\ \ddot{\subseteq}\ \lambda R \cdot \lambda \ell \cdot \dot{\gamma}(\psi(\dot{\alpha}(R))(\ell)) \\
\Longleftrightarrow & \varphi\ \ddot{\subseteq}\ \gamma[\![C]\!](\psi) \\
\Longrightarrow & \forall R : \forall \ell : \varphi(R)(\ell) \subseteq \dot{\gamma}(\psi(\dot{\alpha}(R))(\ell)) \\
\Longrightarrow & \forall r : \forall \ell : \varphi(\dot{\gamma}(r))(\ell) \subseteq \dot{\gamma}(\psi(\dot{\alpha}(\dot{\gamma}(r)))(\ell)) \\
\Longrightarrow & \forall r : \forall \ell : \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))\ \dot{\sqsubseteq}\ \psi(\dot{\alpha}(\dot{\gamma}(r)))(\ell) \\
\Longrightarrow & \qquad \wr \dot{\alpha} \circ \dot{\gamma} \text{ is reductive and } \psi \text{ monotone} \int \\
 & \forall r : \forall \ell : \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))\ \dot{\sqsubseteq}\ \psi(r)(\ell)
\end{array}
$$

$$
\begin{array}{ll}
\Longrightarrow & \forall r : \forall \ell : \alpha[\![C]\!](\varphi)(r)(\ell)\ \dot{\sqsubseteq}\ \psi(r)(\ell) \\
\Longrightarrow & \alpha[\![C]\!](\varphi)\ \ddot{\dot{\sqsubseteq}}\ \psi
\end{array}
$$

$\square$

Generic non-relational forward reachability abstract semantics of commands

We can now define the generic[6] non-relational forward reachability abstract semantics of commands C by

$$\mathsf{Acom}[\![C]\!] \ \dot{\dot{\sqsupseteq}}\ \alpha[\![C]\!](\mathrm{Rcom}[\![C]\!]) \qquad (21)$$

[6] i.e. parameterized by $\langle \wp(\mathbb{V} \mapsto \mathcal{V}), \subseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle L, \sqsubseteq\rangle$.

Structural definition of the generic non-relational forward reachability abstract semantics of commands

$\mathrm{Rcom}[\![\mathtt{skip}]\!]R\ell \ = \ R$

$\mathsf{Acom}[\![\mathtt{X} := A]\!]R\ell \ = \ \mathsf{match}\ \ell\ \mathsf{with}$
$\mid \mathsf{at}_P[\![\mathtt{X} := A]\!] \to R$
$\mid \mathsf{after}_P[\![\mathtt{X} := A]\!] \to R[\mathtt{X} := \mathsf{Faexp}[\![A]\!](R) \sqcap ?^{\triangleright}]$

$\mathsf{Acom}[\![C]\!]R\ell$ where $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi} =$
$\quad \mathsf{match}\ \ell\ \mathsf{with}$
$\quad\quad \mid \mathsf{at}_P[\![C]\!] \to R$
$\quad\quad \mid \mathsf{in}_P[\![S_t]\!] \to \mathsf{Acom}[\![S_t]\!](\mathsf{Abexp}[\![B]\!]R)\ell$
$\quad\quad \mid \mathsf{in}_P[\![S_f]\!] \to \mathsf{Acom}[\![S_f]\!](\mathsf{Abexp}[\![T(\neg(B))]\!]R)\ell$
$\quad\quad \mid \mathsf{after}_P[\![C]\!] \to \mathsf{Acom}[\![S_t]\!](\mathsf{Abexp}[\![B]\!]R)(\mathsf{after}_P[\![S_t]\!])$
$\quad\quad\quad \dot{\sqcup}\ \mathsf{Acom}[\![S_f]\!](\mathsf{Abexp}[\![T(\neg(B))]\!]R)(\mathsf{after}_P[\![S_f]\!])$

$\mathsf{Acom}[\![C]\!]R\ell$ where $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od} =$
$\quad \mathsf{let}\ I \ \dot{\sqsupseteq}\ \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\bot}}\ \lambda X \cdot R \,\dot{\sqcup}\, \mathsf{Acom}[\![S]\!](\mathsf{Abexp}[\![B]\!]X)(\mathsf{after}_P[\![S]\!])\ \mathsf{in}$
$\quad\quad \mathsf{match}\ \ell\ \mathsf{with}$
$\quad\quad \mid \mathsf{at}_P[\![C]\!] \to I$
$\quad\quad \mid \mathsf{in}_P[\![S]\!] \to \mathsf{Acom}[\![S]\!](\mathsf{Abexp}[\![B]\!]I)\ell$
$\quad\quad \mid \mathsf{after}_P[\![C]\!] \to \mathsf{Abexp}[\![T(\neg(B))]\!]I$

$\mathsf{Acom}[\![C\ ;\ S]\!]R\ell \ = \ \mathsf{match}\ \ell\ \mathsf{with}$
$\quad \mid \mathsf{in}_P[\![C]\!] \to \mathsf{Acom}[\![C]\!]R\ell$
$\quad \mid \mathsf{in}_P[\![S]\!] \to \mathsf{Acom}[\![S]\!](\mathsf{Acom}[\![C]\!]R(\mathsf{after}_P[\![C]\!]))\ell$

$\mathsf{Acom}[\![S\ ;;]\!]R\ell \ = \ \mathsf{Acom}[\![S]\!]R\ell$

Calculational design of the structural definition of the generic nonrelational forward reachability abstract semantics of commands

PROOF.

— $\alpha[\![\mathtt{skip}]\!](\mathrm{Rcom}[\![\mathtt{skip}]\!])R\ell$ ⟨def. $\mathsf{Acom}[\![\mathtt{skip}]\!]$⟩

$=$ $\dot{\alpha}(\mathrm{Rcom}[\![\mathtt{skip}]\!](\dot{\gamma}(R))(\ell))$ ⟨def. $\alpha[\![\mathtt{skip}]\!]$⟩

$=$ $\mathsf{match}\ \ell\ \mathsf{with}$
$\quad \mid \mathsf{at}_P[\![\mathtt{skip}]\!] \to \dot{\alpha}(\dot{\gamma}(R))$
$\quad \mid \mathsf{after}_P[\![\mathtt{skip}]\!] \to \dot{\alpha}(\dot{\gamma}(R))$ ⟨def. $\mathrm{Rcom}[\![\mathtt{skip}]\!]$⟩

$\dot{\sqsubseteq}$ $\mathsf{match}\ \ell\ \mathsf{with}$
$\quad \mid \mathsf{at}_P[\![\mathtt{skip}]\!] \to R$
$\quad \mid \mathsf{after}_P[\![\mathtt{skip}]\!] \to R$ ⟨$\dot{\alpha} \circ \dot{\gamma}$ is reductive⟩

$\stackrel{\text{def}}{=}$ $\mathsf{Acom}[\![\mathtt{skip}]\!]$ ⟨Q.E.D.⟩

$$
\begin{aligned}
&\text{---}\ \alpha[\![\texttt{X} := A]\!](\mathrm{Rcom}[\![\texttt{X} := A]\!])R\ell && \wr\text{def. } \mathrm{Acom}[\![\texttt{X} := A]\!]\wr\\
&= \dot\alpha(\mathrm{Rcom}[\![\texttt{X} := A]\!](\dot\gamma(R))(\ell)) && \wr\text{def. } \alpha[\![\texttt{X} := A]\!]\wr\\
&= \textsf{match } \ell \textsf{ with}\\
&\quad\mid \mathsf{at}_P[\![\texttt{X} := A]\!] \rightarrow \dot\alpha(\dot\gamma(R))\\
&\quad\mid \mathsf{after}_P[\![\texttt{X} := A]\!] \rightarrow \dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in (\mathrm{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}) && \wr\text{def. } \mathrm{Rcom}[\![\texttt{X} := A]\!]\wr\\
&\mathrel{\dot\sqsubseteq} \textsf{match } \ell \textsf{ with}\\
&\quad\mid \mathsf{at}_P[\![\texttt{X} := A]\!] \rightarrow R\\
&\quad\mid \mathsf{after}_P[\![\texttt{X} := A]\!] \rightarrow \dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in (\mathrm{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}) && \wr\dot\alpha \circ \dot\gamma \text{ reductive}\wr
\end{aligned}
$$

We consider the second term

$$
\begin{aligned}
&\dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in (\mathrm{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\})\\
&\mathrel{\dot\sqsubseteq} \dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in (\gamma \circ \alpha(\mathrm{Faexp}[\![A]\!](\dot\gamma(R)))) \cap \mathbb{I}\}) && \wr\gamma \circ \alpha \text{ is extensive and monotony with } \{\rho\} \subseteq \dot\gamma(R)\wr\\
&= \dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in \gamma(\alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])R) \cap \mathbb{I}\}) && \wr\text{def. (7) of } \alpha^{\triangleright}(\Phi) \stackrel{\text{def}}{=} \alpha \circ \Phi \circ \dot\gamma\wr
\end{aligned}
$$

$$
\begin{aligned}
&\mathrel{\dot\sqsubseteq} \dot\alpha(\{\rho[\texttt{X} := i] \mid \rho \in \dot\gamma(R) \wedge i \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R) \cap \mathbb{I}\}) && \wr\text{def. (10) of } \mathrm{Faexp}^{\triangleright}[\![A]\!] \sqsupseteq \alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!]) \text{ and monotony}\wr\\
&= \lambda \texttt{Y} \cdot \alpha(\{\rho[\texttt{X} := i](\texttt{Y}) \mid \rho \in \dot\gamma(R) \wedge i \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R) \cap \mathbb{I}\}) && \wr\text{pointwise def. } \dot\alpha\wr\\
&= \lambda \texttt{Y} \cdot (\!|\, \texttt{Y} \neq \texttt{X} \mathrel{?} \alpha(\{\rho(\texttt{Y}) \mid \rho \in \dot\gamma(R)\}) \mathrel{\text{\textexclamdown}} \alpha(\{i \mid i \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R) \cap \mathbb{I}\})\,|\!) && \wr\text{def. assignment}\wr\\
&= \lambda \texttt{Y} \cdot (\!|\, \texttt{Y} \neq \texttt{X} \mathrel{?} \alpha(\{\rho(\texttt{Y}) \mid \forall \texttt{Z} \in \mathrm{Var}[\![P]\!] : \rho(\texttt{Z}) \in \gamma(R(\texttt{Z}))\}) \mathrel{\text{\textexclamdown}} \alpha(\gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R) \cap \mathbb{I})\,|\!) && \wr\text{pointwise def. } \dot\gamma\wr\\
&= \lambda \texttt{Y} \cdot (\!|\, \texttt{Y} \neq \texttt{X} \mathrel{?} \alpha(\{\rho(\texttt{Y}) \mid \rho(\texttt{Y}) \in \gamma(R(\texttt{Y}))\}) \mathrel{\text{\textexclamdown}} \alpha(\gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R) \cap \mathbb{I})\,|\!) && \wr\text{no constraints on } \rho(\texttt{Z}),\ \texttt{Z} \neq \texttt{Y}\wr\\
&= \lambda \texttt{Y} \cdot (\!|\, \texttt{Y} \neq \texttt{X} \mathrel{?} \alpha(\gamma(R(\texttt{Y}))) \mathrel{\text{\textexclamdown}} \alpha(\gamma(\mathrm{Faexp}^{\triangleright}[\![A]\!]R)) \sqcap \alpha(\mathbb{I})\,|\!) && \wr\alpha \text{ monotone}\wr\\
&\mathrel{\dot\sqsubseteq} \lambda \texttt{Y} \cdot (\!|\, \texttt{Y} \neq \texttt{X} \mathrel{?} R(\texttt{Y}) \mathrel{\text{\textexclamdown}} \mathrm{Faexp}^{\triangleright}[\![A]\!]R \sqcap ?^{\triangleright}\,|\!) && \wr\alpha \circ \gamma \text{ reductive, (13) so that } ?^{\triangleright} \sqsupseteq \alpha(\mathbb{I}), \text{ monotony of } \sqcap\wr\\
&= R[\texttt{X} := \mathrm{Faexp}[\![A]\!](R) \sqcap ?^{\triangleright}] && \wr\text{def. assignment}\wr
\end{aligned}
$$

Grouping the two cases, we get

$$
\begin{aligned}
&\quad\textsf{match } \ell \textsf{ with}\\
&\quad\mid \mathsf{at}_P[\![\texttt{X} := A]\!] \rightarrow R\\
&\quad\mid \mathsf{after}_P[\![\texttt{X} := A]\!] \rightarrow R[\texttt{X} := \mathrm{Faexp}[\![A]\!](R) \sqcap ?^{\triangleright}] && \wr\wr\\
&\stackrel{\text{def}}{=} \mathrm{Acom}[\![\texttt{X} := A]\!] && \wr\text{Q.E.D.}\wr\\
&\text{---}\ \alpha[\![C]\!](\mathrm{Rcom}[\![C]\!])R\ell \text{ where } C = \texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi} && \wr\text{def. } \mathrm{Acom}[\![C]\!]\wr\\
&= \dot\alpha(\mathrm{Rcom}[\![C]\!](\dot\gamma(R))(\ell)) && \wr\text{def. } \alpha[\![C]\!]\wr\\
&= \textsf{match } \ell \textsf{ with}\\
&\quad\mid \mathsf{at}_P[\![C]\!] \rightarrow \dot\alpha(\dot\gamma(R))\\
&\quad\mid \mathsf{in}_P[\![S_t]\!] \rightarrow \dot\alpha(\mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]\dot\gamma(R))\ell)\\
&\quad\mid \mathsf{in}_P[\![S_f]\!] \rightarrow \dot\alpha(\mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]\dot\gamma(R))\ell)\\
&\quad\mid \mathsf{after}_P[\![C]\!] \rightarrow \dot\alpha(\mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]\dot\gamma(R))(\mathsf{after}_P[\![S_t]\!])\\
&\quad\cup \mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]\dot\gamma(R))(\mathsf{after}_P[\![S_f]\!])) && \wr\text{def. } \mathrm{Rcom}[\![C]\!]\wr
\end{aligned}
$$

We now prove a lemma for $S \in \{S_t, S_f\}$ and more generally for any immediate subcomponent of the command C so that $\alpha[\![C]\!](\mathrm{Rcom}[\![S]\!])(\mathrm{Abexp}[\![B]\!]R)\ell \mathrel{\dot\sqsubseteq} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]R)\ell$ by induction hypothesis. We have

$$
\begin{aligned}
&\dot\alpha(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]\dot\gamma(R))\ell)\\
&\mathrel{\dot\sqsubseteq} \dot\alpha(\mathrm{Rcom}[\![S]\!](\dot\gamma(\dot\alpha(\mathrm{Cbexp}[\![B]\!]\dot\gamma(R))))\ell) && \wr\dot\gamma \circ \dot\alpha \text{ extensive and monotony}\wr\\
&= \dot\alpha(\mathrm{Rcom}[\![S]\!](\dot\gamma(\ddot\alpha(\mathrm{Cbexp}[\![B]\!])R))\ell) && \wr\text{def. (17) of } \ddot\alpha(\Phi) \stackrel{\text{def}}{=} \dot\alpha \circ \Phi \circ \dot\gamma\wr\\
&\mathrel{\dot\sqsubseteq} \dot\alpha(\mathrm{Rcom}[\![S]\!](\dot\gamma(\mathrm{Abexp}[\![B]\!]R))\ell) && \wr\text{def. (19) of } \mathrm{Abexp}[\![B]\!] \mathrel{\ddot\sqsupseteq} \ddot\alpha(\mathrm{Cbexp}[\![B]\!]) \text{ and monotony}\wr\\
&= \alpha[\![C]\!](\mathrm{Rcom}[\![S]\!])(\mathrm{Abexp}[\![B]\!]R)\ell && \wr\text{def. } \alpha[\![C]\!]\varphi \stackrel{\text{def}}{=} \lambda r \cdot \lambda \ell \cdot \dot\alpha(\varphi(\dot\gamma(r))(\ell))\wr
\end{aligned}
$$

$\dot{\sqsubseteq}$ $\mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]R)\ell$ $\qquad\wr$ by induction hypothesis $\int$

We have $\dot{\alpha}(\dot{\gamma}(R)) \mathrel{\dot{\sqsubseteq}} R$. Applying the lemma to S_t and S_f and observing that the last case is a combination of the two using the fact that $\dot{\alpha}$ is a complete join morphism, we get

$$
\begin{array}{ll}
\dot{\sqsubseteq} & \text{match } \ell \text{ with} \\
& \mid \mathrm{at}_P[\![C]\!] \to R \\
& \mid \mathrm{in}_P[\![S_t]\!] \to \mathrm{Acom}[\![S_t]\!](\mathrm{Abexp}[\![B]\!]R)\ell \\
& \mid \mathrm{in}_P[\![S_f]\!] \to \mathrm{Acom}[\![S_f]\!](\mathrm{Abexp}[\![T(\neg(B))]\!]R)\ell \\
& \mid \mathrm{after}_P[\![C]\!] \to \mathrm{Acom}[\![S_t]\!](\mathrm{Abexp}[\![B]\!]R)(\mathrm{after}_P[\![S_t]\!]) \\
& \dot{\sqcup}\ \mathrm{Acom}[\![S_f]\!](\mathrm{Abexp}[\![T(\neg(B))]\!]R)(\mathrm{after}_P[\![S_f]\!]) \\
\stackrel{\text{def}}{=} & \mathrm{Acom}[\![C]\!] \qquad \wr \text{Q.E.D.} \int \\
\text{—} & \alpha[\![C]\!](\mathrm{Rcom}[\![C]\!])R\ell \text{ where } C = \texttt{while } B \texttt{ do } S \texttt{ od} \qquad \wr \text{def. Acom}[\![C]\!] \int \\
= & \dot{\alpha}(\mathrm{Rcom}[\![C]\!](\dot{\gamma}(R))(\ell)) \qquad \wr \text{def. } \alpha[\![C]\!] \int
\end{array}
$$

$$
\begin{array}{ll}
= & \text{let } I = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot \dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!]) \text{ in} \\
& \text{match } \ell \text{ with} \\
& \mid \mathrm{at}_P[\![C]\!] \to \dot{\alpha}(I) \\
& \mid \mathrm{in}_P[\![S]\!] \to \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]I)(\ell)) \\
& \mid \mathrm{after}_P[\![C]\!] \to \dot{\alpha}(\mathrm{Cbexp}[\![T(\neg(B))]\!]R)I) \qquad \wr \text{def. Rcom}[\![C]\!] \int \\
\dot{\sqsubseteq} & \text{let } \overline{I} \mathrel{\dot{\sqsupseteq}} \dot{\alpha}(\mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot \dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \text{ in} \\
& \text{match } \ell \text{ with} \\
& \mid \mathrm{at}_P[\![C]\!] \to \dot{\alpha}(\dot{\gamma}(\overline{I})) \\
& \mid \mathrm{in}_P[\![S]\!] \to \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]\gamma(\overline{I}))(\ell)) \\
& \mid \mathrm{after}_P[\![C]\!] \to \dot{\alpha}(\mathrm{Cbexp}[\![T(\neg(B))]\!]R)\gamma(\overline{I})) \qquad \wr \text{by monotony} \int \\
\dot{\sqsubseteq} & \text{let } \overline{I} \mathrel{\dot{\sqsupseteq}} \dot{\alpha}(\mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot \dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \text{ in} \\
& \text{match } \ell \text{ with} \\
& \mid \mathrm{at}_P[\![C]\!] \to \overline{I} \\
& \mid \mathrm{in}_P[\![S]\!] \to \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]\gamma(\overline{I}))(\ell)) \\
& \mid \mathrm{after}_P[\![C]\!] \to \dot{\alpha}(\mathrm{Cbexp}[\![T(\neg(B))]\!]R)\gamma(\overline{I})) \qquad \wr \dot{\alpha} \circ \gamma \text{ is reductive} \int
\end{array}
$$

To overestimate $\dot{\alpha}(\mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot \dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!]))$ in fixpoint form, we use the least fixpoint abstraction theorem of course 15. To compute the abstract transformer, we compute:

$$
\begin{array}{ll}
& \dot{\alpha}(\dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \\
= & \dot{\alpha}(\dot{\gamma}(R)) \mathrel{\dot{\sqcup}} \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \qquad \wr \dot{\alpha} \text{ is a complete join morphism} \int \\
\dot{\sqsubseteq} & R \mathrel{\dot{\sqcup}} \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \qquad \wr \dot{\alpha} \circ \dot{\gamma} \text{ is reductive and } \dot{\sqcup} \text{ monotone} \int \\
\dot{\sqsubseteq} & R \mathrel{\dot{\sqcup}} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]X(\mathrm{after}_P[\![S]\!])) \qquad \wr \text{by the previous lemma with } \ell = \mathrm{after}_P[\![S]\!] \int
\end{array}
$$

and so

$$
\begin{array}{ll}
& I \\
\dot{\sqsubseteq} & \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\alpha}(\emptyset)}\ \lambda X \cdot R \mathrel{\dot{\sqcup}} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]X(\mathrm{after}_P[\![S]\!])) \qquad \wr \text{by fixpoint approximation} \int
\end{array}
$$

$$
= \ \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\bot}}\ \lambda X \cdot R \mathrel{\dot{\sqcup}} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]X(\mathrm{after}_P[\![S]\!])) \qquad \wr \text{by def. (3)}
$$

of $\dot{\alpha}(R) \stackrel{\text{def}}{=} \alpha_c \circ \alpha_r(R) = \lambda \mathtt{X} \in \mathbb{V} \cdot \alpha(\{\rho(\mathtt{X}) \mid \rho \in R\}$ so that $\dot{\alpha}(R) = \lambda \mathtt{X} \cdot \alpha(\emptyset)$. But $\emptyset \subseteq \gamma(\bot)$ implies by (1) that $\alpha(\emptyset) \sqsubseteq \bot$ whence $\alpha(\emptyset) = \bot$ by def. infimum and so pointwise $\dot{\alpha}(\emptyset) = \dot{\bot}$ $\int$

If we let $\overline{I}$ be an overapproximation of the abstract fixpoint, that is

$$
\mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\bot}}\ \lambda X \cdot \dot{\gamma}(R) \mathrel{\dot{\sqcup}} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!]) \mathrel{\dot{\sqsubseteq}} \overline{I}
$$

then $\dot{\alpha}(\mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \cdot \dot{\gamma}(R) \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!])) \mathrel{\dot{\sqsubseteq}} \overline{I}$ by transitivity and so

$$
\begin{array}{ll}
& \dot{\alpha}(\mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]\dot{\gamma}(R))\ell) \\
\dot{\sqsubseteq} & \text{let } \overline{I} \mathrel{\dot{\sqsupseteq}} \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\bot}}\ \lambda X \cdot \dot{\gamma}(R) \mathrel{\dot{\sqcup}} \mathrm{Acom}[\![S]\!](\mathrm{Abexp}[\![B]\!]X)(\mathrm{after}_P[\![S]\!]) \text{ in} \\
& \text{match } \ell \text{ with} \\
& \mid \mathrm{at}_P[\![C]\!] \to \overline{I} \\
& \mid \mathrm{in}_P[\![S]\!] \to \dot{\alpha}(\mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]\gamma(\overline{I}))(\ell)) \\
& \mid \mathrm{after}_P[\![C]\!] \to \dot{\alpha}(\mathrm{Cbexp}[\![T(\neg(B))]\!]R)\gamma(\overline{I}))
\end{array}
$$

$=$ let $\overline{I} \mathrel{\dot{\sqsupseteq}} \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{\bot}}\ \lambda X \cdot \dot{\gamma}(R) \mathrel{\dot{\sqcup}} \text{Acom}[\![S]\!](\text{Abexp}[\![B]\!]X)(\text{after}_P[\![S]\!])$ in
match ℓ with
$\mid \text{at}_P[\![C]\!] \rightarrow \overline{I}$
$\mid \text{in}_P[\![S]\!] \rightarrow \text{Acom}[\![S]\!](\text{Abexp}[\![B]\!]\overline{I})\ell$
$\mid \text{after}_P[\![C]\!] \rightarrow \text{Abexp}[\![T(\neg(B))]\!]\overline{I}$ $\quad \wr$by the lemma and hyp. (19) on $\text{Abexp}[\![B]\!] \mathrel{\ddot{\sqsupseteq}} \ddot{\alpha}(\text{Cbexp}[\![B]\!])\wr$

$\stackrel{\text{def}}{=} \text{Acom}[\![C]\!]$ $\quad \wr$Q.E.D.$\wr$

— $\alpha[\![C\ ;\ S]\!](\text{Rcom}[\![C\ ;\ S]\!])R\ell$ $\quad \wr$def. $\text{Acom}[\![C\ ;\ S]\!]\wr$

$= \dot{\alpha}(\text{Rcom}[\![C\ ;\ S]\!](\dot{\gamma}(R))(\ell))$ $\quad \wr$def. $\alpha[\![C\ ;\ S]\!]\wr$

$=$ match ℓ with
$\mid \text{in}_P[\![C]\!] \rightarrow \dot{\alpha}(\text{Rcom}[\![C]\!]\dot{\gamma}(R)\ell)$
$\mid \text{in}_P[\![S]\!] \rightarrow \dot{\alpha}(\text{Rcom}[\![S]\!](\text{Rcom}[\![C]\!]\dot{\gamma}(R)(\text{after}_P[\![C]\!]))\ell)$ $\quad \wr$def. $\text{Rcom}[\![C\ ;\ S]\!]\wr$

$\dot{\sqsubseteq}$ match ℓ with
$\mid \text{in}_P[\![C]\!] \rightarrow \dot{\alpha}(\text{Rcom}[\![C]\!]\dot{\gamma}(R)\ell)$
$\mid \text{in}_P[\![S]\!] \rightarrow \dot{\alpha}(\text{Rcom}[\![S]\!](\dot{\gamma}(\dot{\alpha}(\text{Rcom}[\![C]\!]\dot{\gamma}(R)(\text{after}_P[\![C]\!]))\ell)))$ $\quad \wr\dot{\gamma} \circ \dot{\alpha}$ extensive and monotony$\wr$

$\dot{\sqsubseteq}$ match ℓ with
$\mid \text{in}_P[\![C]\!] \rightarrow \alpha[\![C]\!](\text{Rcom}[\![C]\!])(R)\ell)$
$\mid \text{in}_P[\![S]\!] \rightarrow \alpha[\![C]\!](\text{Rcom}[\![S]\!])(\alpha[\![C]\!](\text{Rcom}[\![C]\!])(R)(\text{after}_P[\![C]\!]))\ell))$ $\quad \wr$def. (21) of $\alpha[\![C]\!]\varphi \stackrel{\text{def}}{=} \lambda r \cdot \lambda \ell \cdot \dot{\alpha}(\varphi(\dot{\gamma}(r))(\ell))\wr$

$\dot{\sqsubseteq}$ match ℓ with
$\mid \text{in}_P[\![C]\!] \rightarrow \text{Acom}[\![C]\!]R\ell$
$\mid \text{in}_P[\![S]\!] \rightarrow \alpha[\![C]\!](\text{Rcom}[\![S]\!])(\text{Acom}[\![C]\!]R(\text{after}_P[\![C]\!]))\ell))$ $\quad \wr$ind. hyp. and monotony$\wr$

$=$ match ℓ with
$\mid \text{in}_P[\![C]\!] \rightarrow \text{Acom}[\![C]\!]R\ell$
$\mid \text{in}_P[\![S]\!] \rightarrow \text{Acom}[\![S]\!](\text{Acom}[\![C]\!]R(\text{after}_P[\![C]\!]))\ell$ $\quad \wr$ind. hyp.$\wr$

$\stackrel{\text{def}}{=} \text{Acom}[\![C\ ;\ S]\!]$ $\quad \wr$Q.E.D.$\wr$

— $\alpha[\![C]\!](\text{Rcom}[\![S\ ;;]\!])R\ell$ $\quad \wr$def. $\text{Acom}[\![S\ ;;]\!]\wr$

$= \dot{\alpha}(\text{Rcom}[\![S\ ;;]\!](\dot{\gamma}(R))(\ell))$ $\quad \wr$def. $\alpha[\![S\ ;;]\!]\wr$

$= \dot{\alpha}(\text{Rcom}[\![S]\!]\dot{\gamma}(R)\ell)$ $\quad \wr$def. $\text{Rcom}[\![S\ ;;]\!]\wr$

$\dot{\sqsubseteq} \text{Acom}[\![S]\!]R\ell$ $\quad \wr$by induction hypothesis$\wr$

$\stackrel{\text{def}}{=} \text{Acom}[\![S\ ;;]\!]$ $\quad \wr$Q.E.D.$\wr$

$\square$

Example: initialization and simple sign analysis

Peter Naur

Michel Sintzoff

Ben Wegbreit

References

[1] Naur P.. "Checking of operand types in ALGOL compilers". BIT 5 (1965), 151-163.

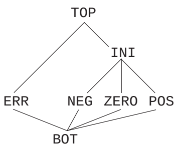
[2] M. Sintzoff. "Calculating properties of programs by valuations on specific models". Proceedings of ACM conference on Proving assertions about programs, Las Cruces, New Mexico, USA, pp. 203–207, 1972

[3] Ben Wegbreit. "Property Extraction in Well-Founded Property Sets". IEEE Trans. Software Eng. 1(3): 270-285 (1975)

Initialization and simple sign abstraction

– Abstraction: record initialization and sign only

– Abstract lattice:



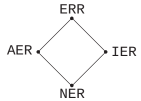
– Concretization:

$$\begin{array}{llll}
\gamma(\texttt{BOT}) \stackrel{\text{def}}{=} \{\Omega_{\text{a}}\} & & \gamma(\texttt{INI}) \stackrel{\text{def}}{=} \mathbb{I} \cup \{\Omega_{\text{a}}\}, & \\
\gamma(\texttt{NEG}) \stackrel{\text{def}}{=} [\texttt{min_int}, -1] \cup \{\Omega_{\text{a}}\} & & \gamma(\texttt{ERR}) \stackrel{\text{def}}{=} \{\Omega_{\text{i}}, \Omega_{\text{a}}\} & (22)\\
\gamma(\texttt{ZERO}) \stackrel{\text{def}}{=} \{0, \Omega_{\text{a}}\} & & \gamma(\texttt{TOP}) \stackrel{\text{def}}{=} \mathbb{I}_\Omega & \\
\gamma(\texttt{POS}) \stackrel{\text{def}}{=} [1, \texttt{max_int}] \cup \{\Omega_{\text{a}}\} & & &
\end{array}$$

– NEGZ, NZERO and POSZ such that $\gamma(\texttt{NEGZ}) \stackrel{\text{def}}{=} [\texttt{min\_int}, 0] \cup \{\Omega_{\text{a}}\}$, $\gamma(\texttt{NZERO}) \stackrel{\text{def}}{=} [\texttt{min\_int}, -1] \cup [1, \texttt{max\_int}] \cup \{\Omega_{\text{a}}\}$ and $\gamma(\texttt{POSZ}) \stackrel{\text{def}}{=} [0, \texttt{max\_int}] \cup \{\Omega_{\text{a}}\}$ not included to illustrate losses of information

– if we had defined $\gamma(\texttt{ERR}) \stackrel{\text{def}}{=} \{\Omega_{\text{i}}\}$ then γ would not be monotone (hence we would not have a Galois connection)

– Another abstract value would be needed to discriminate the initialization and arithmetic errors:



$$\begin{aligned}
\gamma(\texttt{NER}) &\stackrel{\text{def}}{=} \mathbb{I}\\
\gamma(\texttt{IER}) &\stackrel{\text{def}}{=} \mathbb{I} \cup \{\Omega_{\text{i}}\}\\
\gamma(\texttt{AER}) &\stackrel{\text{def}}{=} \mathbb{I} \cup \{\Omega_{\text{a}}\}\\
\gamma(\texttt{ERR}) &\stackrel{\text{def}}{=} \mathbb{I} \cup \{\Omega_{\text{a}}, \Omega_{\text{i}}\}
\end{aligned}$$

– Another possible definition of γ would have been (22) but with $\gamma(\texttt{BOT}) \stackrel{\text{def}}{=} \emptyset$.

Then γ would not preserve meets (since e.g. $\gamma(\texttt{NEG} \sqcap \texttt{POS}) = \gamma(\texttt{BOT}) = \emptyset \neq \{\Omega_{\text{a}}\} = \gamma(\texttt{NEG}) \sqcap \gamma(\texttt{POS})$). It would then follow that $\langle\alpha, \gamma\rangle$ is not a Galois connection since best approximations may not exist. For example $\{\Omega_{\text{a}}\}$ would be upper approximable by the minimal ERR, NEG, ZERO or POS, none of which being more precise than the others in all contexts.

– Another completely different choice of γ would be

$$\begin{array}{ll}
\gamma(\texttt{BOT}) \stackrel{\text{def}}{=} \emptyset, & \gamma(\texttt{INI}) \stackrel{\text{def}}{=} \mathbb{I},\\
\gamma(\texttt{NEG}) \stackrel{\text{def}}{=} [\texttt{min_int}, -1], & \gamma(\texttt{ERR}) \stackrel{\text{def}}{=} \{\Omega_{\text{i}}, \Omega_{\text{a}}\},\\
\gamma(\texttt{ZERO}) \stackrel{\text{def}}{=} \{0\}, & \gamma(\texttt{TOP}) \stackrel{\text{def}}{=} \mathbb{I}_\Omega\ .\\
\gamma(\texttt{POS}) \stackrel{\text{def}}{=} [1, \texttt{max_int}], &
\end{array}$$

- With such a definition of γ for a program analysis taking arithmetic overflows into account, the usual rule of signs $\texttt{POS} + \texttt{POS} = \texttt{POS}$ would <u>not</u> hold since the sums of large positive machine integers may yield an arithmetic error Ω_a such that $\Omega_\mathrm{a} \notin \gamma(\texttt{POS})$. The correct version of the rule of sign would be $\texttt{POS} + \texttt{POS} = \texttt{TOP}$, which is too imprecise.
- A similar error ($\gamma(\texttt{NEG}) \stackrel{\text{def}}{=} \{z \in \mathbb{N} \mid z < 0\}$ and $\gamma(\texttt{POS}) \stackrel{\text{def}}{=} \{z \in \mathbb{N} \mid z \geq 0\}$) is done in [2], the first attempt to apply the rule of signs to programs [7] ($\perp$, $\top$, the lattice structure of abstract values [3], fixpoints and soundness criteria were also missing, indeed the sign analysis of [2] is erroneous)

[7] following the pionner work of [1]

Initialization and simple sign abstraction

The abstraction of $P \in \wp(\mathbb{I}_\Omega)$ is

$$\begin{array}{rl} \alpha(P) \stackrel{\text{def}}{=} & (\!\!\!\; P \subseteq \{\Omega_\mathrm{a}\} \;?\; \texttt{BOT} \\ & \| \; P \subseteq [\texttt{min\_int}, -1] \cup \{\Omega_\mathrm{a}\} \;?\; \texttt{NEG} \\ & \| \; P \subseteq \{0, \Omega_\mathrm{a}\} \;?\; \texttt{ZERO} \\ & \| \; P \subseteq [1, \texttt{max\_int}] \cup \{\Omega_\mathrm{a}\} \;?\; \texttt{POS} \\ & \| \; P \subseteq \mathbb{I} \cup \{\Omega_\mathrm{a}\} \;?\; \texttt{INI} \\ & \| \; P \subseteq \{\Omega_\mathrm{i}, \Omega_\mathrm{a}\} \;?\; \texttt{ERR} \\ & ¿ \; \texttt{TOP})\!\!\!\; . \end{array} \tag{23}$$

so that (1) holds: $\langle \wp(\mathbb{I}_\Omega), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L, \sqsubseteq \rangle$.

Initialization and simple sign abstract forward arithmetic operations

Considering the initialization and simple sign abstraction (23), the calculational design of the forward abstract operations proceeds as follows

$$\begin{array}{ll} & \alpha(\{\underline{\mathrm{n}}\}) \\ = & \begin{array}{lll} \texttt{NEG} & \text{if} & \underline{\mathrm{n}} \in [\texttt{min\_int}, -1] \\ \texttt{ZERO} & \text{if} & \underline{\mathrm{n}} = 0 \\ \texttt{POS} & \text{if} & \underline{\mathrm{n}} \in [1, \texttt{max\_int}] \\ \texttt{BOT} & \text{if} & \underline{\mathrm{n}} < \texttt{min\_int} \;\text{ or }\; \underline{\mathrm{n}} > \texttt{max\_int} \end{array} \qquad \wr (23) \text{ and case analysis} \int \\ \stackrel{\text{def}}{=} & \mathrm{n}^{\triangleright} \;. \end{array}$$

$$\begin{array}{ll} & \alpha(\mathbb{I}) \\ = & \texttt{INI} \qquad \wr (\int 23) \\ \stackrel{\text{def}}{=} & ?^{\triangleright} \;. \end{array}$$

We design $-^{\triangleright}(p) \stackrel{\text{def}}{=} \alpha(\{-v \mid v \in \gamma(p)\})$ by case analysis. Recall the definition of bounded machine integers.

$$\begin{array}{ll} \texttt{max\_int} > 9, & \text{greatest machine integer;} \\ \texttt{min\_int} \stackrel{\text{def}}{=} -\texttt{max\_int} - 1, & \text{smallest machine integer;} \\ z \in \mathbb{Z}, & \text{mathematical integers;} \\ i \in \mathbb{I} \stackrel{\text{def}}{=} [\texttt{min\_int}, \texttt{max\_int}], & \text{bounded machine integers.} \end{array} \tag{24}$$

We have

$$
\begin{array}{lll}
-^{\triangleright}(\mathtt{BOT}) & = \alpha(\{-\, v \mid v \in \gamma(\mathtt{BOT})\}) & \wr\text{def. (14) of } -^{\triangleright}\int \\
& = \alpha(\{\underline{-}\, v \mid v \in \{\Omega_{\mathrm{a}}\}\}) & \wr\text{def. (22) of } \gamma\int \\
& = \alpha(\{\Omega_{\mathrm{a}}\}) & \wr\text{def. } \underline{-}\int \\
& = \mathtt{BOT} & \wr\text{def. (23) of } \alpha\int \\
-^{\triangleright}(\mathtt{POS}) & = \alpha(\{-\, v \mid v \in \gamma(\mathtt{POS})\}) & \wr\text{def. (14) of } -^{\triangleright}\int \\
& = \alpha(\{\underline{-}\, v \mid v \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathrm{a}}\}\}) & \wr\text{def. (22) of } \gamma\int \\
& = \alpha([-\mathtt{max\_int}, -1] \cup \{\Omega_{\mathrm{a}}\}) & \wr\text{def. } \underline{-} \text{ and (24)}\int \\
& = \mathtt{NEG} & \wr\text{def. (23) of } \alpha\int \\
-^{\triangleright}(\mathtt{ERR}) & = \alpha(\{-\, v \mid v \in \gamma(\mathtt{ERR})\}) & \wr\text{def. (14) of } -^{\triangleright}\int \\
& = \alpha(\{\underline{-}\, v \mid v \in \{\Omega_{\mathrm{i}}, \Omega_{\mathrm{a}}\}\}) & \wr\text{def. (22) of } \gamma\int \\
& = \alpha(\{\Omega_{\mathrm{i}}, \Omega_{\mathrm{a}}\}) & \wr\text{def. of } \underline{-}\int \\
& = \mathtt{ERR} & \wr\text{def. (23) of } \alpha\int
\end{array}
$$

The calculational design for the other cases of $-^{\triangleright}$ and that of $+^{\triangleright}$ is similar and we get

p	BOT	NEG	ZERO	POS	INI	ERR	TOP
$+^{\triangleright}(p)$	BOT	NEG	ZERO	POS	INI	ERR	TOP
$-^{\triangleright}(p)$	BOT	POS	ZERO	NEG	INI	ERR	TOP

The calculational design of the abstract binary operators is also similar and will not be fully detailed. For division, we get

$/^{\triangleright}(p, q)$		q: BOT	NEG	ZERO	POS	INI	ERR	TOP
p	BOT	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	NEG	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	ZERO	BOT	BOT	BOT	ZERO	POS	ERR	TOP
	POS	BOT	BOT	BOT	INI	INI	ERR	TOP
	INI	BOT	BOT	BOT	INI	INI	ERR	TOP
	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
	TOP	ERR	ERR	ERR	TOP	TOP	ERR	TOP

Let us consider a few typical cases. First division by a negative number always leads to an arithmetic error

$$
\begin{array}{lll}
/^{\triangleright}(\mathtt{POS}, \mathtt{NEG}) & = \alpha(\{v_1 \underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{POS}) \wedge v_2 \in \gamma(\mathtt{NEG})\}) & \wr\text{def. (15) of } /^{\triangleright}\int \\
& = \alpha(\{v_1 \underline{/}\, v_2 \mid v_1 \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathrm{a}}\} \wedge & \wr\text{def. (22) of } \gamma\int \\
& \qquad v_2 \in [\mathtt{min\_int}, -1] \cup \{\Omega_{\mathrm{a}}\}\}) & \\
& = \alpha(\{\Omega_{\mathrm{a}}\}) & \wr\text{def. } \underline{/}\int \\
& = \mathtt{BOT} & \wr\text{def. (23) of } \alpha\int
\end{array}
$$

No abstract property exactly represents non-negative numbers which yields imprecise results

$$
\begin{array}{lll}
/^{\triangleright}(\mathtt{POS}, \mathtt{POS}) & = \alpha(\{v_1 \underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{POS}) \wedge v_2 \in \gamma(\mathtt{POS})\}) & \wr\text{def. (15) of } /^{\triangleright}\int \\
& = \alpha(\{v_1 \underline{/}\, v_2 \mid v_1 \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathrm{a}}\} \wedge & \wr\text{def. (22) of } \gamma\int \\
& \qquad v_2 \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathrm{a}}\}\}) & \\
& = \alpha([0, \mathtt{max\_int}] \cup \{\Omega_{\mathrm{a}}\}) & \wr\text{def. } \underline{/}\int \\
& = \mathtt{INI} & \wr\text{def. (23) of } \alpha\int
\end{array}
$$

Because of left to right evaluation, left errors are propagated first

$$
/^{\triangleright}(\mathtt{BOT}, \mathtt{ERR}) = \alpha(\{v_1 \underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{BOT}) \wedge v_2 \in \gamma(\mathtt{ERR})\}) \quad \wr\text{def. (15) of } /^{\triangleright}\int
$$

$$
\begin{array}{rcll}
 & = & \alpha(\{v_1 \mathbin{\underline{/}} v_2 \mid v_1 \in \{\Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \langle \text{def. (22) of } \gamma \rangle \\
 & = & \alpha(\{\Omega_{\mathtt{a}}\}) & \langle \text{def. } \underline{/} \rangle \\
 & = & \mathtt{BOT} & \langle \text{def. (23) of } \alpha \rangle \\
\underline{/}^{\triangleright}(\mathtt{ERR}, \mathtt{BOT}) & = & \alpha(\{v_1 \mathbin{\underline{/}} v_2 \mid v_1 \in \gamma(\mathtt{ERR}) \wedge v_2 \in \gamma(\mathtt{BOT})\}) & \langle \text{def. (15) of } \underline{/}^{\triangleright} \rangle \\
 & = & \alpha(\{v_1 \mathbin{\underline{/}} v_2 \mid v_1 \in \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{a}}\}) & \langle \text{def. (22) of } \gamma \rangle \\
 & = & \alpha(\{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \langle \text{def. } \underline{/} \rangle \\
 & = & \mathtt{ERR} & \langle \text{def. (23) of } \alpha \rangle \\
\underline{/}^{\triangleright}(\mathtt{TOP}, \mathtt{BOT}) & = & \alpha(\{v_1 \mathbin{\underline{/}} v_2 \mid v_1 \in \gamma(\mathtt{TOP}) \wedge v_2 \in \gamma(\mathtt{BOT})\}) & \langle \text{def. (15) of } \underline{/}^{\triangleright} \rangle \\
 & = & \alpha(\{v_1 \mathbin{\underline{/}} v_2 \mid v_1 \in [\mathtt{min\_int}, \mathtt{max\_int}] \cup & \langle \text{def. (22) of } \gamma \rangle \\
 & & \quad \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{a}}\}) & \\
 & = & \alpha(\{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \langle \text{def. } \underline{/} \rangle
\end{array}
$$

$$
\begin{array}{rcll}
 & = & \mathtt{ERR} & \langle \text{def. (23) of } \alpha \rangle
\end{array}
$$

The other forward abstract binary arithmetic operators for initialization and simple sign analysis are as follows

$+^{\triangleright}(p,q)$		q: BOT	NEG	ZERO	POS	INI	ERR	TOP
	BOT	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	NEG	BOT	NEG	NEG	INI	INI	ERR	TOP
	ZERO	BOT	NEG	ZERO	POS	INI	ERR	TOP
p	POS	BOT	INI	POS	POS	INI	ERR	TOP
	INI	BOT	INI	INI	INI	INI	ERR	TOP
	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
	TOP	ERR	TOP	TOP	TOP	TOP	ERR	TOP

$-^{\triangleright}(p,q)$		q: BOT	NEG	ZERO	POS	INI	ERR	TOP
	BOT	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	NEG	BOT	INI	NEG	NEG	INI	ERR	TOP
	ZERO	BOT	POS	ZERO	NEG	INI	ERR	TOP
p	POS	BOT	POS	POS	INI	INI	ERR	TOP
	INI	BOT	INI	INI	INI	INI	ERR	TOP
	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
	TOP	ERR	TOP	TOP	TOP	TOP	ERR	TOP

$*^{\triangleright}(p,q)$		q: BOT	NEG	ZERO	POS	INI	ERR	TOP
	BOT	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	NEG	BOT	POS	ZERO	NEG	INI	ERR	TOP
	ZERO	BOT	ZERO	ZERO	ZERO	ZERO	ERR	TOP
p	POS	BOT	NEG	ZERO	POS	INI	ERR	TOP
	INI	BOT	INI	ZERO	INI	INI	ERR	TOP
	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
	TOP	ERR	TOP	TOP	TOP	TOP	ERR	TOP

$\mathtt{mod}^{\triangleright}(p,q)$		q: BOT	NEG	ZERO	POS	INI	ERR	TOP
	BOT	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	NEG	BOT	BOT	BOT	BOT	BOT	BOT	BOT
	ZERO	BOT	BOT	BOT	ZERO	ZERO	ERR	TOP
p	POS	BOT	BOT	BOT	INI	INI	ERR	TOP
	INI	BOT	BOT	BOT	INI	INI	ERR	TOP
	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
	TOP	ERR	ERR	ERR	TOP	TOP	ERR	TOP

Initialization and simple sign abstract forward arithmetic comparison operations

The abstract strict comparison for initialization and simple sign analysis is as follows

$\check{<}(p_1,p_2)R$		p_2: BOT, ERR	NEG	ZERO	POS	INI, TOP
	BOT, ERR	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$
	NEG	$\dot{\mathtt{BOT}}$	R	R	R	R
p_1	ZERO	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	R	R
	POS	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	$\dot{\mathtt{BOT}}$	R	R
	INI, TOP	$\dot{\mathtt{BOT}}$	R	R	R	R

Proof. – If $p_i \in \{\mathtt{BOT}, \mathtt{ERR}\}$, $i = 1$ or $i = 2$ then $\gamma(p_i) \subseteq \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}$ so that $\gamma(p_i) \cap \mathbb{I} = \emptyset$ and we get $\check{<}(p_1,p_2)R = \lambda \mathtt{X} \cdot \mathtt{BOT} \stackrel{\text{def}}{=} \dot{\mathtt{BOT}}$.

– If $p_1 = \mathtt{POS}$ and $p_2 \in \{\mathtt{NEG}, \mathtt{ZERO}\}$ then $\forall v_1 \in \gamma(p_1) \cap \mathbb{I} = \{x \in \mathbb{I} \mid x > 0\}$: $\forall v_2 \in \gamma(p_2) \cap \mathbb{I} = \{x \in \mathbb{I} \mid x < 0\}$ or $\{0\}$: $\neg(v_1 < v_2)$ and so $\check{<}(p_1,p_2)R = \dot{\mathtt{BOT}}$.

– If $p_1, p_2 = \mathtt{POS}$ then $\exists v_1 \in \gamma(p_1) \cap \mathbb{I} = \{x \in \mathbb{I} \mid x > 0\}$: $\exists v_2 \in \gamma(p_2) \cap \mathbb{I} = \{x \in \mathbb{I} \mid x > 0\}$: $(v_1 < v_2)$ (for example $v_1 = 1$ and $v_2 = 2$). So, by (21), $\check{c}(p_1,p_2)R \mathrel{\dot{\sqsupseteq}} (\exists v_1 \in \gamma(p_1) : \exists v_2 \in \gamma(p_2) \cap \mathbb{I} : v_1 \mathrel{\underline{c}} v_2 = \mathrm{tt} \mathrel{?} R \mathrel{¿} \perp) = R$

– The other cases are handled in a similar way.

$\square$

Similarly,

$\check{\doteq}(p_1, p_2)R$		p_2: BOT, ERR	NEG	ZERO	POS	INI, TOP
p_1	BOT, ERR	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$
	NEG	$\mathtt{B\dot{O}T}$	R	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	R
	ZERO	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	R	$\mathtt{B\dot{O}T}$	R
	POS	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	$\mathtt{B\dot{O}T}$	R	R
	INI, TOP	$\mathtt{B\dot{O}T}$	R	R	R	R

Implementation of the abstract interpreter

Implementation: abstract domain of sets of values

```
(* avalues.mli *)
(* abstraction of sets of machine integers by initialization *)
(* and simple sign                                         f*)
type t
val bot : unit -> t
val isbotempty : unit -> bool
val initerr : unit -> t
val top : unit -> t
val join : t -> t -> t
val meet : t -> t -> t
val leq : t -> t -> bool
val eq : t -> t -> bool
val in_errors : t -> bool
```



```
val print : t -> unit
(* forward abstract semantics of arithmetic expressions *)
val f_NAT : string -> t
val f_RANDOM : unit -> t
val f_UMINUS : t -> t
val f_UPLUS : t -> t
val f_PLUS : t -> t -> t
val f_MINUS : t -> t -> t
val f_TIMES : t -> t -> t
val f_DIV : t -> t -> t
val f_MOD : t -> t -> t
(* forward abstract semantics of boolean expressions *)
val f_EQ : t -> t -> bool
val f_LT : t -> t -> bool
```

```
(* avalues.ml *)
open Values
(* abstraction of sets of machine integers by initialization *)
(* and  simple sign *)
(* complete lattice *)
type t = BOT | NEG | ZERO | POS | INI | ERR | TOP
(* \gamma(BOT)  = {_O_(a)}                                   *)
(* \gamma(NEG)  = [min_int,-1] U {_O_(a)}                    *)
(* \gamma(POS)  = [1,max_int] U {_O_(a)}                     *)
(* \gamma(ZERO) = {0, _O_(a)}                                *)
(* \gamma(INI)  = [min_int,max_int] U {_O_(a)}               *)
(* \gamma(ERR)  = {_O_(i),_O_(a)}                            *)
(* \gamma(TOP)  = [min_int,max_int] U {_O_(a),_O_(i)}        *)
(* infimum *)
let bot () = BOT
(* bottom is emptyset? *)
let isbotempty () = false (* \gamma(BOT) = {_O_(a)} <> \emptyset *)
```



```
(* uninitialization *)
let initerr () = ERR
(* supremum *)
let top () = TOP
(* least upper bound *)
let nat_of_lat u =
      match u with
      | BOT  -> 0
      | NEG  -> 1
      | ZERO -> 2
      | POS  -> 3
      | INI  -> 4
      | ERR  -> 5
      | TOP  -> 6
let select t u v = t.(nat_of_lat u).(nat_of_lat v)
let join_table =
(*          BOT    NEG   ZERO   POS   INI   ERR   TOP  *)
(*BOT*)[|[| BOT ;  NEG ; ZERO ; POS ; INI ; ERR ; TOP |];
```



```
(*NEG*)  [| NEG ;  NEG ; INI  ; INI ; INI ; TOP ; TOP |];
(*ZERO*) [| ZERO ; INI ; ZERO ; INI ; INI ; TOP ; TOP |];
(*POS*)  [| POS ;  INI ; INI  ; POS ; INI ; TOP ; TOP |];
(*INI*)  [| INI ;  INI ; INI  ; INI ; INI ; TOP ; TOP |];
(*ERR*)  [| ERR ;  TOP ; TOP  ; TOP ; TOP ; ERR ; TOP |];
(*TOP*)  [| TOP ;  TOP ; TOP  ; TOP ; TOP ; TOP ; TOP |]|]
let join u v = select join_table u v
(* greatest lower bound *)
let meet_table =
(*          BOT   NEG   ZERO   POS   INI   ERR   TOP   *)
(*BOT*)[|[| BOT ; BOT ; BOT  ; BOT ; BOT ;  BOT ; BOT  |];
(*NEG*)  [| BOT ; NEG ; BOT  ; BOT ; NEG ;  BOT ; NEG  |];
(*ZERO*) [| BOT ; BOT ; ZERO ; BOT ; ZERO ; BOT ; ZERO |];
(*POS*)  [| BOT ; BOT ; BOT  ; POS ; POS ;  BOT ; POS  |];
(*INI*)  [| BOT ; NEG ; ZERO ; POS ; INI ;  BOT ; INI  |];
(*ERR*)  [| BOT ; BOT ; BOT  ; BOT ; BOT ;  ERR ; ERR  |];
(*TOP*)  [| BOT ; NEG ; ZERO ; POS ; INI ;  ERR ; TOP  |]|]
let meet u v = select meet_table u v
```



```
(* approximation ordering *)
let leq_table =
(*          BOT     NEG     ZERO    POS     INI     ERR     TOP  *)
(*BOT*)[|[| true ;  true ;  true ;  true ;  true ;  true ;  true |];
(*NEG*)  [| false ; true ;  false ; false ; true ;  false ; true |];
(*ZERO*) [| false ; false ; true ;  false ; true ;  false ; true |];
(*POS*)  [| false ; false ; false ; true ;  true ;  false ; true |];
(*INI*)  [| false ; false ; false ; false ; true ;  false ; true |];
(*ERR*)  [| false ; false ; false ; false ; false ; true ;  true |];
(*TOP*)  [| false ; false ; false ; false ; false ; false ; true |]|]
let leq u v = select leq_table u v
(* equality *)
let eq u v = (u = v)
(* included in errors? *)
let in_errors v = (leq v ERR)
(* printing *)
let print u =
  match u with
```

```
    | BOT  -> (print_string "BOT")
    | NEG  -> (print_string "NEG")
    | ZERO -> (print_string "ZERO")
    | POS  -> (print_string "POS")
    | INI  -> (print_string "INI")
    | ERR  -> (print_string "ERR")
    | TOP  -> (print_string "TOP")
(* forward abstract semantics of arithmetic expressions *)
let f_NAT s =
   match (machine_int_of_string s) with
   | (ERROR_NAT INITIALIZATION) -> ERR
   | (ERROR_NAT ARITHMETIC) -> BOT
   | (NAT i) -> if i = 0 then ZERO else if i > 0 then POS else NEG
let f_RANDOM () = INI
let f_UMINUS a =
   match a with
   | BOT  -> BOT
   | NEG  -> POS
```



```
   | ZERO -> ZERO
   | POS  -> NEG
   | INI  -> INI
   | ERR  -> ERR
   | TOP  -> TOP
let f_UPLUS a = a
let f_PLUS_table =
(*     +    BOT    NEG    ZERO    POS    INI    ERR    TOP  *)
(*BOT*)[|[| BOT ; BOT ; BOT  ; BOT ; BOT ; BOT ; BOT |];
(*NEG*)  [| BOT ; NEG ; NEG  ; INI ; INI ; ERR ; TOP |];
(*ZERO*) [| BOT ; NEG ; ZERO ; POS ; INI ; ERR ; TOP |];
(*POS*)  [| BOT ; INI ; POS  ; POS ; INI ; ERR ; TOP |];
(*INI*)  [| BOT ; INI ; INI  ; INI ; INI ; ERR ; TOP |];
(*ERR*)  [| ERR ; ERR ; ERR  ; ERR ; ERR ; ERR ; ERR |];
(*TOP*)  [| ERR ; TOP ; TOP  ; TOP ; TOP ; ERR ; TOP |]|]
let f_PLUS  u v = select f_PLUS_table u v
let f_MINUS_table =
(*     -    BOT    NEG    ZERO    POS    INI    ERR    TOP  *)
```



```
(*BOT*)[|[| BOT ; BOT ; BOT  ; BOT ; BOT ; BOT ; BOT |];
(*NEG*)  [| BOT ; INI ; NEG  ; NEG ; INI ; ERR ; TOP |];
(*ZERO*) [| BOT ; POS ; ZERO ; NEG ; INI ; ERR ; TOP |];
(*POS*)  [| BOT ; POS ; POS  ; INI ; INI ; ERR ; TOP |];
(*INI*)  [| BOT ; INI ; INI  ; INI ; INI ; ERR ; TOP |];
(*ERR*)  [| ERR ; ERR ; ERR  ; ERR ; ERR ; ERR ; ERR |];
(*TOP*)  [| ERR ; TOP ; TOP  ; TOP ; TOP ; ERR ; TOP |]|]
let f_MINUS u v = select f_MINUS_table u v
let f_TIMES_table =
(*     *    BOT    NEG    ZERO    POS    INI    ERR    TOP  *)
(*BOT*)[|[| BOT ; BOT ;  BOT  ; BOT ;  BOT ;  BOT ; BOT |];
(*NEG*)  [| BOT ; POS ;  ZERO ; NEG ;  INI ;  ERR ; TOP |];
(*ZERO*) [| BOT ; ZERO ; ZERO ; ZERO ; ZERO ; ERR ; TOP |];
(*POS*)  [| BOT ; NEG ;  ZERO ; POS ;  INI ;  ERR ; TOP |];
(*INI*)  [| BOT ; INI ;  ZERO ; INI ;  INI ;  ERR ; TOP |];
(*ERR*)  [| ERR ; ERR ;  ERR  ; ERR ;  ERR ;  ERR ; ERR |];
(*TOP*)  [| ERR ; TOP ;  TOP  ; TOP ;  TOP ;  ERR ; TOP |]|]
let f_TIMES u v = select f_TIMES_table u v
```



```
let f_DIV_table =
(*     /    BOT    NEG    ZERO  POS    INI    ERR    TOP  *)
(*BOT*)[|[| BOT ; BOT ; BOT ; BOT ;  BOT ; BOT ; BOT |];
(*NEG*)  [| BOT ; BOT ; BOT ; BOT ;  BOT ; BOT ; BOT |];
(*ZERO*) [| BOT ; BOT ; BOT ; ZERO ; POS ; ERR ; TOP |];
(*POS*)  [| BOT ; BOT ; BOT ; INI ;  INI ; ERR ; TOP |];
(*INI*)  [| BOT ; BOT ; BOT ; INI ;  INI ; ERR ; TOP |];
(*ERR*)  [| ERR ; ERR ; ERR ; ERR ;  ERR ; ERR ; ERR |];
(*TOP*)  [| ERR ; ERR ; ERR ; TOP ;  TOP ; ERR ; TOP |]|]
let f_DIV   u v = select f_DIV_table u v
let f_MOD_table =
(*    mod   BOT    NEG    ZERO  POS    INI    ERR    TOP  *)
(*BOT*)[|[| BOT ; BOT ; BOT ; BOT  ; BOT  ; BOT ; BOT |];
(*NEG*)  [| BOT ; BOT ; BOT ; BOT  ; BOT  ; BOT ; BOT |];
(*ZERO*) [| BOT ; BOT ; BOT ; ZERO ; ZERO ; ERR ; TOP |];
(*POS*)  [| BOT ; BOT ; BOT ; INI  ; INI  ; ERR ; TOP |];
(*INI*)  [| BOT ; BOT ; BOT ; INI  ; INI  ; ERR ; TOP |];
(*ERR*)  [| ERR ; ERR ; ERR ; ERR  ; ERR  ; ERR ; ERR |];
```

```
(*TOP*)  [| ERR ; ERR ; ERR ; TOP  ; TOP  ; ERR ; TOP |]|]
let f_MOD   u v = select f_MOD_table u v
(* forward abstract semantics of boolean expressions *)
let f_EQ_table =
(*    mod   BOT     NEG     ZERO    POS     INI     ERR     TOP    *)
(*BOT*)[|[| false ; false ; false ; false ; false ; false ; false |];
(*NEG*)  [| false ; true  ; false ; false ; true  ; false ; true  |];
(*ZERO*) [| false ; false ; true  ; false ; true  ; false ; true  |];
(*POS*)  [| false ; false ; false ; true  ; true  ; false ; true  |];
(*INI*)  [| false ; true  ; true  ; true  ; true  ; false ; true  |];
(*ERR*)  [| false ; false ; false ; false ; false ; false ; false |];
(*TOP*)  [| false ; true  ; true  ; true  ; true  ; false ; true  |]|]
(* Are there integer values in gamma(u) equal to values in gamma(v)? *)
let f_EQ u v = select f_EQ_table u v
let f_LT_table =
(*    mod   BOT     NEG     ZERO    POS     INI     ERR     TOP    *)
(*BOT*)[|[| false ; false ; false ; false ; false ; false ; false |];
(*NEG*)  [| false ; true  ; true  ; true  ; true  ; false ; true  |];
```



```
(*ZERO*) [| false ; false ; true  ; true  ; true  ; false ; true  |];
(*POS*)  [| false ; false ; false ; true  ; true  ; false ; true  |];
(*INI*)  [| false ; true  ; true  ; true  ; true  ; false ; true  |];
(*ERR*)  [| false ; false ; false ; false ; false ; false ; false |];
(*TOP*)  [| false ; true  ; true  ; true  ; true  ; false ; true  |]|]
(* Are there integer values in gamma(u) less than or equal to (<=) *)
(* integer values in gamma(v)?                                     *)
let f_LT u v = select f_LT_table u v
```


Implementation: abstract domain of sets of environments

The remaining files are (essentially[8]) unchanged

```
(* aenv.mli *)
open Abstract_Syntax
open Avalues
(* set of environments *)
type t
(* infimum *)
val bot : unit -> t
(* check for infimum *)
val is_bot : t -> bool
(* uninitialization *)
val initerr : unit -> t
(* supremum *)
```

[8] apart from trivial module renaming


```
val top : unit -> t
(* copy *)
val copy : t -> t
(* least upper bound *)
val join : t -> t -> t
(* greatest lower bound *)
val meet : t -> t -> t
(* approximation ordering *)
val leq : t -> t -> bool
(* equality *)
val eq : t -> t -> bool
(* printing *)
val print : t -> unit
(* r(X) = {e(X) | X in r} *)
val get : t -> variable -> Avalues.t
(* r[X <- v] = {e[X <- i] | e in r /\ i in v}                    *)
val set : t -> variable -> Avalues.t -> t
(* collecting semantics of assignment                            *)
```

```
(* f_ASSIGN x f r =  {e[x <- i] | e in r /\ i in f({e}) cap I }      *)
val f_ASSIGN : variable -> (t -> Avalues.t) -> t -> t
(* collecting semantics of boolean expressions                       *)
(* f_EQ f g r =                                                      *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) cap I: *)
(*             v1 = v2 }                                             *)
val f_EQ : (t -> Avalues.t) -> (t -> Avalues.t) -> t -> t
(* f_LT f g r =                                                      *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) cap I: *)
(*             v1 < v2 }                                             *)
val f_LT : (t -> Avalues.t) -> (t -> Avalues.t) -> t -> t
```



```
(* aenv.ml *)
open Variables
open Avalues
open Array
type t = Avalues.t array
(* infimum *)
let bot () = Array.create (number_of_variables ()) (Avalues.bot ())
(* check for infimum *)
let is_bot r =
   let f x v = x or (Avalues.eq v (Avalues.bot ())) in
      Array.fold_left f false r
(* uninitialization *)
let initerr () =
  Array.create (number_of_variables ()) (Avalues.initerr ())
(* supremum *)
let top () = Array.create (number_of_variables ()) (Avalues.top ())
(* copy *)
```



```
let copy = copy (* implementation without side-effects *)
(* least upper bound *)
let join r1 r2 =
   let f i v = (Avalues.join (get r1 i) v) in
      mapi f r2
(* greatest lower bound *)
let meet r1 r2 =
    let f i v = (Avalues.meet (get r1 i) v) in
        mapi f r2
(* approximation ordering *)
exception NotTrue
let leq r1 r2 =
   try
      let f x =
         if not (Avalues.leq (get r1 x) (get r2 x)) then
            raise NotTrue
         in for_all_variables f;
         true
```



```
   with
      NotTrue -> false
(* equality *)
let eq r1 r2 =
   try
      let f i =
         if not (Avalues.eq (get r1 i) (get r2 i)) then
            raise NotTrue
      in for_all_variables f;
      true
   with
      NotTrue -> false
(* printing *)
let print r =
   let p v = Avalues.print (get r v) in
      print_map_variables p
(* r(X) = {e(X) | eo in r}                    *)
(* val get : t -> variable -> Cvalues.t       *)
```

```
let get r x = (get r x)
(* r[X <- v] = {e[X <- i] | e in r /\ i in v} *)
(* val set : t -> variable -> Cvalues.t -> t  *)
let set r x v =
   if (Avalues.eq v (Avalues.bot ())) & (Avalues.isbotempty ()) then
      (bot ()) (* reduce *)
   else
      (let r' = copy r in (set r' x v; r'))
(* f_ASSIGN x f r =  {e[x <- i] | e in r /\ i in f({e}) cap I }       *)
let f_ASSIGN x f r = set r x (Avalues.meet (f r) (Avalues.f_RANDOM ()))
(* f_EQ f g r =                                                        *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) cap I: *)
(*              v1 = v2 }                                              *)
let f_EQ f g r = if (Avalues.f_EQ (f r) (g r)) then r else (bot ())
(* f_LT f g r =                                                        *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) cap I: *)
(*              v1 < v2 }                                              *)
let f_LT f g r = if (Avalues.f_LT (f r) (g r)) then r else (bot ())
```


Implementation: abstract interpretation of arithmetic operations

```
(* aaexp.mli *)
open Abstract_Syntax
open Avalues
open Aenv
(* evaluation of arithmetic operations *)
val a_aexp : aexp -> Aenv.t -> Avalues.t
```



```
(* aaexp.ml *)
open Abstract_Syntax
(* Abstract interpretation of arithmetic operations *)
let rec a_aexp a r =
 if (Aenv.is_bot r) && (Avalues.isbotempty()) then Avalues.bot() else
  match a with
  | (Abstract_Syntax.NAT i) -> (Avalues.f_NAT i)
  | (VAR v)       -> (Aenv.get r v)
  | RANDOM        -> Avalues.f_RANDOM ()
  | (UPLUS a1)    -> (Avalues.f_UPLUS (a_aexp a1 r))
  | (UMINUS a1)   -> (Avalues.f_UMINUS (a_aexp a1 r))
  | (PLUS (a1, a2))  -> (Avalues.f_PLUS (a_aexp a1 r) (a_aexp a2 r))
  | (MINUS (a1, a2)) -> (Avalues.f_MINUS (a_aexp a1 r) (a_aexp a2 r))
  | (TIMES (a1, a2)) -> (Avalues.f_TIMES (a_aexp a1 r) (a_aexp a2 r))
  | (DIV (a1, a2))   -> (Avalues.f_DIV (a_aexp a1 r) (a_aexp a2 r))
  | (MOD (a1, a2))   -> (Avalues.f_MOD (a_aexp a1 r) (a_aexp a2 r))
```


Implementation: abstract interpretation of boolean operations

```
(* abexp.mli *)
open Abstract_Syntax
open Avalues
open Aenv
(* abstract interpretation of boolean operations *)
val a_bexp : bexp -> Aenv.t -> Aenv.t
```

```
(* abexp.ml *)
open Abstract_Syntax
open Avalues
open Aenv
open Aaexp
(* abstract interpretation of boolean operations *)
let rec a_bexp b r =
  match b with
  | TRUE           -> r
  | FALSE          -> (Aenv.bot ())
  | (EQ (a1, a2))  -> f_EQ (a_aexp a1) (a_aexp a2) r
  | (LT (a1, a2))  -> f_LT (a_aexp a1) (a_aexp a2) r
  | (AND (b1, b2)) -> Aenv.meet (a_bexp b1 r) (a_bexp b2 r)
  | (OR (b1, b2))  -> Aenv.join (a_bexp b1 r) (a_bexp b2 r)
```


Implementation: abstract interpretation of commands

```
(* acom.mli *)
open Abstract_Syntax
open Labels
open Aenv
(* forward abstract interpretation of commands *)
val acom : com -> Aenv.t -> label -> Aenv.t
```



```
(* acom.ml *)
open Abstract_Syntax
open Labels
open Aenv
open Aaexp
open Abexp
open Fixpoint
(* forward abstract semantics of commands *)

exception Error of string
let rec acom c r l =
 match c with
 | (SKIP (l', l'')) ->
     if (l = l') then r
     else if (l = l'') then r
     else (raise (Error "SKIP incoherence"))
 | (ASSIGN (l',x,a,l'')) ->
```



```
     if (l = l') then r
     else if (l = l'') then
                f_ASSIGN x (a_aexp a) r
     else (raise (Error "ASSIGN incoherence"))
 | (SEQ (l', s, l'')) ->
     (acomseq s r l)
 | (IF (l', b, nb, t, f, l'')) ->
     (if (l = l') then r
     else if (incom l t) then
        (acom t (a_bexp b r) l)
     else if (incom l f) then
        (acom f (a_bexp nb r) l)
     else if (l = l'') then
        (join (acom t (a_bexp b r) (after t))
         (acom f (a_bexp nb r) (after f)))
     else (raise (Error "IF incoherence")))
 | (WHILE (l', b, nb, c', l'')) ->
   let f x = join r (acom c' (a_bexp b x) (after c'))
```

```
    in let i = lfp (bot ()) leq f in
      (if (l = l') then i
       else if (incom l c') then (acom c' (a_bexp b i) l)
       else if (l = l'') then (a_bexp nb i)
       else (raise (Error "WHILE incoherence")))
and acomseq s r l = match s with
 | []   -> raise (Error "empty SEQ incoherence")
 | [c]  -> if (incom l c) then (acom c r l)
           else (raise (Error "SEQ incoherence"))
 | h::t -> if (incom l h) then (acom h r l)
           else (acomseq t (acom h r (after h)) l)
```


Implementation: abstract interpreter

```
(* main.ml *)
open Program_To_Abstract_Syntax
open Labels
open Pretty_Print
open Aenv
open Acom
let _ =
  let arg = if (Array.length  Sys.argv) = 1 then ""
            else Sys.argv.(1) in
     Random.self_init ();
    let p = (abstract_syntax_of_program arg) in
      (print (initerr ());
       pretty_print p;
       print (acom p (initerr ()) (after p));
       print_newline ())
```


Implementation: `makefile`

```
# makefile

SOURCES = \
symbol_Table.mli \
symbol_Table.ml \
variables.mli \
variables.ml \
abstract_Syntax.ml \
concrete_To_Abstract_Syntax.mli \
concrete_To_Abstract_Syntax.ml \
labels.mli \
labels.ml \
parser.mli \
parser.ml \
```



```
lexer.ml \
program_To_Abstract_Syntax.mli \
program_To_Abstract_Syntax.ml \
pretty_Print.mli \
pretty_Print.ml \
values.mli \
values.ml \
avalues.mli \
avalues.ml \
aenv.mli \
aenv.ml \
aenv.mli \
aenv.ml \
aaexp.mli \
aaexp.ml \
abexp.mli \
abexp.ml \
fixpoint.mli \
```

```
fixpoint.ml \
acom.mli \
acom.ml \
main.ml

.PHONY : help
help :
   @echo ""
   @echo "make help        : this help"
   @echo "make trace       : trace fixpoint iterates"
   @echo "make untrace     : don't trace fixpoint iterates"
   @echo "make compile     : compile"
   @echo "./a.out filename : execute"
   @echo "make examples    : execute the examples"
   @echo "make errors      : execute the examples with runtime errors"
   @echo "make clean       : remove auxilairy files"
   @echo ""

```



```
.PHONY : trace preparetrace
trace: preparetrace compile
   @echo "fixpoint tracing mode"
preparetrace:
   @/bin/rm -f fixpoint.ml
   @ln -s fixpoint_printing_iterates.ml fixpoint.ml
   @/bin/rm -f aaexp.ml
   @ln -s aaexp-trace.ml aaexp.ml
   @/bin/rm -f abexp.ml
   @ln -s abexp-trace.ml abexp.ml

.PHONY : untrace prepareuntrace
untrace: prepareuntrace compile
   @echo "no fixpoint tracing, recompile!"
prepareuntrace:
   @/bin/rm -f fixpoint.ml
   @ln -s fixpoint_no_printing.ml fixpoint.ml
   @/bin/rm -f aaexp.ml
```



```
   @ln -s aaexp-untrace.ml aaexp.ml
   @/bin/rm -f abexp.ml
   @ln -s abexp-untrace.ml abexp.ml

.PHONY : compile
compile:
   ocamlyacc parser.mly
   ocamllex lexer.mll
#   ocamlc -i $(SOURCES) # to print types
   ocamlc $(SOURCES)

.PHONY : examples
examples :
   ./a.out ../Examples/example00.sil
   ./a.out ../Examples/example01.sil
   ./a.out ../Examples/example02.sil
   ./a.out ../Examples/example03.sil
   ./a.out ../Examples/example04.sil
```



```
   ./a.out ../Examples/example05.sil
   ./a.out ../Examples/example07.sil

.PHONY : errors
errors :
   ./a.out ../Examples/example06.sil
   ./a.out ../Examples/example08.sil
   ./a.out ../Examples/example09.sil
   ./a.out ../Examples/example10.sil
   ./a.out ../Examples/example11.sil

.PHONY :
clean :
   /bin/rm -f *.cmi *.cmo *~ a.out lexer.ml parser.mli parser.ml
```

Implementation: examples without runtime errors

```
Script started on Fri Apr 15 16:49:58 2005
Initialization-Simple-Sign % make clean
...
Initialization-Simple-Sign % make untrace
...
Initialization-Simple-Sign % make compile
...
Initialization-Simple-Sign % make examples

./a.out ../Examples/example0.sil
{}
0:
  skip
```



```
1:

{}
./a.out ../Examples/example1.sil
{ x:ERR }
0:
  x := 1;
1:
  while (x < 100) do
    2:
      x := (x + 1)
    3:
  od {((100 < x) | (x = 100))}
4:

{ x:POS }
./a.out ../Examples/example2.sil
{ x:ERR; y:ERR }
```



```
0:
  x := (-1073741823 - 1);
1:
  y := (x - 1)
2:

{ x:NEG; y:NEG }
./a.out ../Examples/example3.sil
{ x:ERR; y:ERR }
0:
  x := 0;
1:
  y := 1
2:

{ x:ZERO; y:POS }
./a.out ../Examples/example4.sil
{ x:ERR }
```



```
0:
  if true then
    1:
      x := 1
    2:
  else {false}
    3:
      x := 0
    4:
  fi
5:

{ x:INI }
./a.out ../Examples/example5.sil
{ x:ERR }
0:
  if false then
    1:
```

```
      x := 1
    2:
  else {true}
    3:
      x := 0
    4:
  fi
5:

{ x:INI }
./a.out ../Examples/example7.sil
{ x:ERR }
0:
  x := 1;
1:
  while ((x < 10) | (x = 10)) do
    2:
      x := (x + 1)
```



```
    3:
  od {(10 < x)}
4:

{ x:POS }
Initialization-Simple-Sign % ^Dexit
Script done on Fri Apr 15 16:50:42 2005
```


Implementation: examples with runtime errors

```
Script started on Fri Apr 15 16:51:19 2005
Initialization-Simple-Sign % make errors
./a.out ../Examples/example6.sil
{ x:ERR }
0:
  x := -1073741824
1:

{ x:BOT }
./a.out ../Examples/example8.sil
{ x:ERR }
0:
  x := 1073741823
```



```
1:

{ x:POS }
./a.out ../Examples/example9.sil
{ x:ERR; y:ERR; z:ERR; t:ERR }
0:
  x := (-536870912 * 2);
1:
  y := (536870912 * 2);
2:
  z := ((-1073741823 - 1) * 1);
3:
  t := ((-1073741823 - 1) * 1073741823)
4:

{ x:NEG; y:POS; z:NEG; t:NEG }
./a.out ../Examples/example10.sil
{ x:ERR }
```

```
0:
  x := ?;
1:
  if (x < (-1073741823 - 1)) then
    2:
      x := 1
    3:
  else {(((-1073741823 - 1) < x) | (x = (-1073741823 - 1)))}
    4:
      x := 0
    5:
  fi
6:

{ x:INI }
./a.out ../Examples/example11.sil
{ x:ERR }
0:
```



```
  x := 1;
1:
  while (0 < 1073741824) do
    2:
      x := (x + 1)
    3:
  od {((1073741824 < 0) | (1073741824 = 0))}
4:

{ x:BOT }
Initialization-Simple-Sign % ^Dexit
Script done on Fri Apr 15 16:51:42 2005
```

On `example10.sil`, observe that the test yields {x : `BOT`}, but since γ(`BOT`) $\neq \emptyset$, the assignments respectively yield {x : `POS`} and {x : `ZERO`} whence the join {x : `INI`} on exit of the test.

Personal project: homework 1

- Change file `avalues.ml` to implement a finitary analysis of your choice (e.g. initialization analysis, Killdall's constant propagation, parity analysis, enriched sign analysis, etc.).
- Provide the abstraction and concretization functions for sets of concrete values.
- The main design criteria are originality and <u>soundness</u>

Bibliography

[4] P. Cousot. "The Calculational Design of a Generic Abstract Interpreter". In M. Broy and R. Steinbrüggen (eds.): *Calculational System Design*. NATO ASI Series F. Amsterdam: IOS Press, 1999.

[5] P. Cousot. "The Marktoberdorf'98 Generic Abstract Interpreter". `http://www.di.ens.fr/~cousot/Marktoberdorf98.shtml`.

THE END

My MIT web site is http://www.mit.edu/~cousot/

The course web site is http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/.