Program Analysis (470)
Abstract Interpretation

Herbert Wiklicky
herbert@doc.ic.ac.uk

Department of Computing
Imperial College London

Spring 2015

Abstract Multiplication

How can we justify or obtain correct abstract versions of various operations, e.g. multiplication?

$\times^{\#}$	$\bot$	**even**	**odd**	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
even	$\bot$	**even**	**even**	**even**
odd	$\bot$	**even**	**odd**	$\top$
$\top$	$\bot$	**even**	$\top$	$\top$

Abstract Interpretation – invented by Patrick Cousot and Radhia Cousot in 1977 – allows even to compute abstractions which are correct by construction.

Abstract Multiplication

How can we justify or obtain correct abstract versions of various operations, e.g. multiplication?

$\times^{\#}$	$\bot$	**even**	**odd**	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
even	$\bot$	**even**	**even**	**even**
odd	$\bot$	**even**	**odd**	$\top$
$\top$	$\bot$	**even**	$\top$	$\top$

Abstract Interpretation – invented by Patrick Cousot and Radhia Cousot in 1977 – allows even to compute abstractions which are correct by construction.

Abstract Multiplication

How can we justify or obtain correct abstract versions of various operations, e.g. multiplication?

$\times^{\#}$	$\bot$	**even**	**odd**	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
even	$\bot$	**even**	**even**	**even**
odd	$\bot$	**even**	**odd**	$\top$
$\top$	$\bot$	**even**	$\top$	$\top$

Abstract Interpretation – invented by Patrick Cousot and Radhia Cousot in 1977 – allows even to compute abstractions which are correct by construction.

Concrete Semantics $\rightsquigarrow$

We will give a language and semantics independent treatment of correctness.

To set the scene, imagine some programming language, e.g. WHILE. Its semantics identifies some set V of values (like states, closures, double precision reals) and specifies how a program p transforms one value v_1 to another v_2; we may write this as

$$p \vdash v_1 \rightsquigarrow v_2$$

Concrete Semantics $\rightsquigarrow$

We will give a language and semantics independent treatment of correctness.

To set the scene, imagine some programming language, e.g. WHILE. Its semantics identifies some set V of values (like states, closures, double precision reals) and specifies how a program p transforms one value v_1 to another v_2; we may write this as

$$p \vdash v_1 \rightsquigarrow v_2$$

Concrete Semantics $\rightsquigarrow$

We will give a language and semantics independent treatment of correctness.

To set the scene, imagine some programming language, e.g. WHILE. Its semantics identifies some set V of values (like states, closures, double precision reals) and specifies how a program p transforms one value v_1 to another v_2; we may write this as

$$p \vdash v_1 \rightsquigarrow v_2$$

Abstract Analysis $\triangleright$

In a similar way, a program analysis identifies the set L of properties (like shapes of states, abstract closures, lower and upper bounds for reals) and specifies how a program p transforms one property l_1 to another l_2; we may write

$$p \vdash l_1 \triangleright l_2$$

Unlike the semantics, it is customary to require $\triangleright$ to be deterministic and thereby define a function; this will allow us to write

$$f_p(l_1) = l_2 \text{ to mean } p \vdash l_1 \triangleright l_2.$$

Abstract Analysis $\triangleright$

In a similar way, a program analysis identifies the set L of properties (like shapes of states, abstract closures, lower and upper bounds for reals) and specifies how a program p transforms one property l_1 to another l_2; we may write

$$p \vdash l_1 \triangleright l_2$$

Unlike the semantics, it is customary to require $\triangleright$ to be deterministic and thereby define a function; this will allow us to write

$$f_p(l_1) = l_2 \text{ to mean } p \vdash l_1 \triangleright l_2.$$

Situation in While

We have SOS transitions $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ with S and S' programs and $s, s' \in \textbf{State} = \textbf{Var} \to \mathbb{Z}$, e.g.

$$\langle z := 2 \times z, [z \mapsto 2]\rangle \Rightarrow [z \mapsto 4]$$

translates to just an evaluation of the state:

$$z := 2 \times z \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4]$$

The fact that this also holds for the (abstract) parity means:

$$z := 2 \times z \vdash \textbf{even}(z) \rhd \textbf{even}(z)$$

and also $z := 2 \times z \vdash \textbf{odd}(z) \rhd \textbf{even}(z)$.

Situation in While

We have SOS transitions $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ with S and S' programs and $s, s' \in \mathbf{State} = \mathbf{Var} \to \mathbb{Z}$, e.g.

$$\langle \mathtt{z} := 2 \times \mathtt{z}, [z \mapsto 2]\rangle \Rightarrow [z \mapsto 4]$$

translates to just an evaluation of the state:

$$\mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4]$$

The fact that this also holds for the (abstract) parity means:

$$\mathtt{z} := 2 \times \mathtt{z} \vdash \mathbf{even}(z) \rhd \mathbf{even}(z)$$

and also $\mathtt{z} := 2 \times \mathtt{z} \vdash \mathbf{odd}(z) \rhd \mathbf{even}(z)$.

Situation in While

We have SOS transitions $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ with S and S' programs and $s, s' \in \textbf{State} = \textbf{Var} \to \mathbb{Z}$, e.g.

$$\langle \mathtt{z} := 2 \times \mathtt{z}, [z \mapsto 2]\rangle \Rightarrow [z \mapsto 4]$$

translates to just an evaluation of the state:

$$\mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4]$$

The fact that this also holds for the (abstract) parity means:

$$\mathtt{z} := 2 \times \mathtt{z} \vdash \textbf{even}(z) \rhd \textbf{even}(z)$$

and also $\mathtt{z} := 2 \times \mathtt{z} \vdash \textbf{odd}(z) \rhd \textbf{even}(z)$.

Correctness Relations

Every program analysis should be correct with respect to the semantics. For a class of (so-called first-order) program analyses this is established by directly relating properties to values using a correctness relation:

$$R : V \times L \to \{\mathbf{tt}, \mathbf{ff}\} \quad \text{or} \quad R \subseteq V \times L$$

The intention is that $v\ R\ l$ formalises our claim that the value v is described by the property l.

Correctness Relations

Every program analysis should be correct with respect to the semantics. For a class of (so-called first-order) program analyses this is established by directly relating properties to values using a correctness relation:

$$R : V \times L \to \{\mathbf{tt}, \mathbf{ff}\} \quad \text{or} \quad R \subseteq V \times L$$

The intention is that $v\ R\ l$ formalises our claim that the value v is described by the property l.

Preservation of Correctness

To be useful one has to prove that the correctness relation R is preserved under computation: if the relation holds between the initial value and the initial property then it also holds between the final value and the final property.

This may be formulated as the implication

$$v_1 \; R \; l_1 \;\wedge\; p \vdash v_1 \rightsquigarrow v_2 \;\wedge\; p \vdash l_1 \rhd l_2 \;\Rightarrow\; v_2 \; R \; l_2$$

Preservation of Correctness

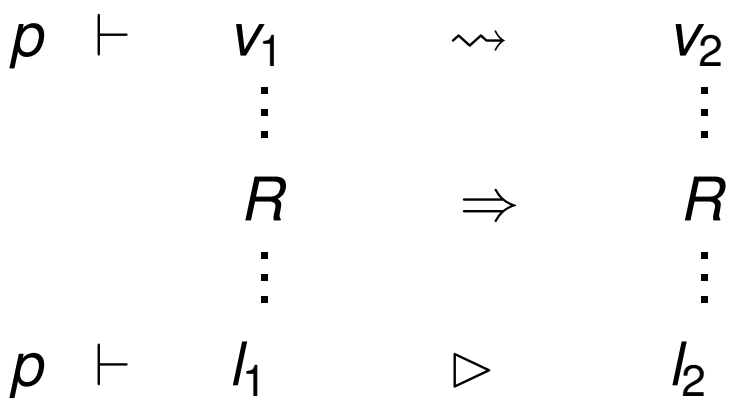
To be useful one has to prove that the correctness relation R is preserved under computation: if the relation holds between the initial value and the initial property then it also holds between the final value and the final property.

This may be formulated as the implication

$$v_1 \; R \; l_1 \;\; \wedge \;\; p \vdash v_1 \rightsquigarrow v_2 \;\; \wedge \;\; p \vdash l_1 \rhd l_2 \;\; \Rightarrow \;\; v_2 \; R \; l_2$$

Preservation of Correctness

This property is also expressed by the following diagram:



Correctness of Parity

$$
\begin{array}{lll|lll}
0 & R & \textbf{even} & 1 & R & \textbf{odd} \\
2 & R & \textbf{even} & 3 & R & \textbf{odd} \\
4 & R & \textbf{even} & 5 & R & \textbf{odd} \\
\multicolumn{3}{c|}{\ldots} & \multicolumn{3}{c}{\ldots}
\end{array}
$$

$$
\begin{array}{l|l}
\mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 1] \rightsquigarrow [z \mapsto 2] & \textbf{odd}(z) \rhd \textbf{even}(z) \\
\mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4] & \textbf{even}(z) \rhd \textbf{even}(z) \\
\mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 3] \rightsquigarrow [z \mapsto 6] & \textbf{odd}(z) \rhd \textbf{even}(z) \\
\ldots & \ldots
\end{array}
$$

$$
\begin{array}{c}
1\ R\ \textbf{odd} \ \wedge\ p \vdash 1 \rightsquigarrow 2 \ \wedge\ p \vdash \textbf{odd} \rhd \textbf{even} \Rightarrow 2\ R\ \textbf{even} \\
2\ R\ \textbf{even} \ \wedge\ p \vdash 2 \rightsquigarrow 4 \ \wedge\ p \vdash \textbf{even} \rhd \textbf{even} \Rightarrow 4\ R\ \textbf{even} \\
3\ R\ \textbf{odd} \ \wedge\ p \vdash 3 \rightsquigarrow 6 \ \wedge\ p \vdash \textbf{odd} \rhd \textbf{even} \Rightarrow 6\ R\ \textbf{even} \\
\ldots
\end{array}
$$

It is therefore correct to say: "$p \equiv \mathtt{z} := 2 \times \mathtt{z}$ always produces an **even** z".

Correctness of Parity

$$\begin{array}{lll|lll} 0 & R & \textbf{even} & 1 & R & \textbf{odd} \\ 2 & R & \textbf{even} & 3 & R & \textbf{odd} \\ 4 & R & \textbf{even} & 5 & R & \textbf{odd} \\ \multicolumn{3}{c|}{\ldots} & \multicolumn{3}{c}{\ldots} \end{array}$$

$$\begin{array}{l|l} \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 1] \rightsquigarrow [z \mapsto 2] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4] & \textbf{even}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 3] \rightsquigarrow [z \mapsto 6] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \ldots & \ldots \end{array}$$

$$1\ R\ \textbf{odd} \ \wedge\ p \vdash 1 \rightsquigarrow 2 \ \wedge\ p \vdash \textbf{odd} \rhd \textbf{even} \Rightarrow 2\ R\ \textbf{even}$$
$$2\ R\ \textbf{even} \ \wedge\ p \vdash 2 \rightsquigarrow 4 \ \wedge\ p \vdash \textbf{even} \rhd \textbf{even} \Rightarrow 4\ R\ \textbf{even}$$
$$3\ R\ \textbf{odd} \ \wedge\ p \vdash 3 \rightsquigarrow 6 \ \wedge\ p \vdash \textbf{odd} \rhd \textbf{even} \Rightarrow 6\ R\ \textbf{even}$$
$$\ldots$$

It is therefore correct to say: "$p \equiv \mathtt{z} := 2 \times \mathtt{z}$ always produces an **even** z".

Correctness of Parity

$$\begin{array}{lll|lll} 0 & R & \textbf{even} & 1 & R & \textbf{odd} \\ 2 & R & \textbf{even} & 3 & R & \textbf{odd} \\ 4 & R & \textbf{even} & 5 & R & \textbf{odd} \\ \multicolumn{3}{c|}{\ldots} & \multicolumn{3}{c}{\ldots} \end{array}$$

$$\begin{array}{l|l} \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 1] \rightsquigarrow [z \mapsto 2] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4] & \textbf{even}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 3] \rightsquigarrow [z \mapsto 6] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \ldots & \ldots \end{array}$$

$$\begin{array}{rcl} 1\ R\ \textbf{odd} \wedge p \vdash 1 \rightsquigarrow 2 \wedge p \vdash \textbf{odd} \rhd \textbf{even} & \Rightarrow & 2\ R\ \textbf{even} \\ 2\ R\ \textbf{even} \wedge p \vdash 2 \rightsquigarrow 4 \wedge p \vdash \textbf{even} \rhd \textbf{even} & \Rightarrow & 4\ R\ \textbf{even} \\ 3\ R\ \textbf{odd} \wedge p \vdash 3 \rightsquigarrow 6 \wedge p \vdash \textbf{odd} \rhd \textbf{even} & \Rightarrow & 6\ R\ \textbf{even} \\ \multicolumn{3}{c}{\ldots} \end{array}$$

It is therefore correct to say: "$p \equiv \mathtt{z} := 2 \times \mathtt{z}$ always produces an **even** z".

Correctness of Parity

$$\begin{array}{lll|lll} 0 & R & \textbf{even} & 1 & R & \textbf{odd} \\ 2 & R & \textbf{even} & 3 & R & \textbf{odd} \\ 4 & R & \textbf{even} & 5 & R & \textbf{odd} \\ \multicolumn{3}{c|}{\ldots} & \multicolumn{3}{c}{\ldots} \end{array}$$

$$\begin{array}{l|l} \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 1] \rightsquigarrow [z \mapsto 2] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 2] \rightsquigarrow [z \mapsto 4] & \textbf{even}(z) \rhd \textbf{even}(z) \\ \mathtt{z} := 2 \times \mathtt{z} \vdash [z \mapsto 3] \rightsquigarrow [z \mapsto 6] & \textbf{odd}(z) \rhd \textbf{even}(z) \\ \ldots & \ldots \end{array}$$

$$\begin{array}{rcl} 1\ R\ \textbf{odd} \wedge p \vdash 1 \rightsquigarrow 2 \wedge p \vdash \textbf{odd} \rhd \textbf{even} & \Rightarrow & 2\ R\ \textbf{even} \\ 2\ R\ \textbf{even} \wedge p \vdash 2 \rightsquigarrow 4 \wedge p \vdash \textbf{even} \rhd \textbf{even} & \Rightarrow & 4\ R\ \textbf{even} \\ 3\ R\ \textbf{odd} \wedge p \vdash 3 \rightsquigarrow 6 \wedge p \vdash \textbf{odd} \rhd \textbf{even} & \Rightarrow & 6\ R\ \textbf{even} \\ \multicolumn{3}{c}{\ldots} \end{array}$$

It is therefore correct to say: "$p \equiv \mathtt{z} := 2 \times \mathtt{z}$ always produces an **even** z".

Abstract Interpretation and Correctness

The theory of Abstract Interpretation comes to life when we augment the set of properties L with a preorder structure and relate this to the correctness relation R.

The most common scenario is when $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ is a complete lattice with partial ordering $\sqsubseteq$.

We then impose the following relationship between R and L:

$$v \; R \; l_1 \;\wedge\; l_1 \sqsubseteq l_2 \;\Rightarrow\; v \; R \; l_2 \tag{1}$$

$$(\forall l \in L' \subseteq L : v \; R \; l) \;\Rightarrow\; v \; R \left(\bigsqcap L'\right) \tag{2}$$

Abstract Interpretation and Correctness

The theory of Abstract Interpretation comes to life when we augment the set of properties L with a preorder structure and relate this to the correctness relation R.

The most common scenario is when $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ is a complete lattice with partial ordering $\sqsubseteq$.

We then impose the following relationship between R and L:

$$v \; R \; l_1 \;\wedge\; l_1 \sqsubseteq l_2 \;\Rightarrow\; v \; R \; l_2 \tag{1}$$

$$(\forall l \in L' \subseteq L : v \; R \; l) \;\Rightarrow\; v \; R \left(\bigsqcap L'\right) \tag{2}$$

Abstract Interpretation and Correctness

The theory of Abstract Interpretation comes to life when we augment the set of properties L with a preorder structure and relate this to the correctness relation R.

The most common scenario is when $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ is a complete lattice with partial ordering $\sqsubseteq$.

We then impose the following relationship between R and L:

$$v \; R \; l_1 \;\wedge\; l_1 \sqsubseteq l_2 \quad \Rightarrow \quad v \; R \; l_2 \tag{1}$$

$$(\forall l \in L' \subseteq L : v \; R \; l) \quad \Rightarrow \quad v \; R \; \left(\bigsqcap L'\right) \tag{2}$$

Condition (1)

$$v\ R\ l_1 \ \wedge\ l_1 \sqsubseteq l_2 \Rightarrow v\ R\ l_2$$

- The first condition says that the smaller the property is with respect to the partial ordering, the better (i.e. more precise) it is.
- This is an “arbitrary” decision in the sense that we could instead have decided that the larger the property is, the better it is, as is indeed the case in much of the literature on Data Flow Analysis; luckily the principle of duality from lattice theory tells us that this difference is only a cosmetic one.

Condition (1)

$$v\ R\ l_1 \ \wedge\ l_1 \sqsubseteq l_2 \Rightarrow v\ R\ l_2$$

- The first condition says that the smaller the property is with respect to the partial ordering, the better (i.e. more precise) it is.
- This is an “arbitrary” decision in the sense that we could instead have decided that the larger the property is, the better it is, as is indeed the case in much of the literature on Data Flow Analysis; luckily the principle of duality from lattice theory tells us that this difference is only a cosmetic one.

Condition (1)

$$v\ R\ l_1 \ \wedge\ l_1 \sqsubseteq l_2 \Rightarrow v\ R\ l_2$$

- The first condition says that the smaller the property is with respect to the partial ordering, the better (i.e. more precise) it is.

- This is an “arbitrary” decision in the sense that we could instead have decided that the larger the property is, the better it is, as is indeed the case in much of the literature on Data Flow Analysis; luckily the principle of duality from lattice theory tells us that this difference is only a cosmetic one.

Condition (2)

$$(\forall l \in L' \subseteq L : v \; R \; l) \Rightarrow v \; R \; \left(\bigsqcap L'\right)$$

- The second condition says that there is always a best property for describing a value. This is important for having to perform only one analysis (using the best property, i.e. the greatest lower bound of the candidates) instead of several analyses (one for each of the candidates).

- The condition has two immediate consequences:

$$v \; R \; \top$$

$$v \; R \; l_1 \; \wedge \; v \; R \; l_2 \quad \Rightarrow \quad v \; R \; (l_1 \sqcap l_2)$$

Condition (2)

$$(\forall l \in L' \subseteq L : v \; R \; l) \Rightarrow v \; R \; \left(\bigsqcap L'\right)$$

- The second condition says that there is always a best property for describing a value. This is important for having to perform only one analysis (using the best property, i.e. the greatest lower bound of the candidates) instead of several analyses (one for each of the candidates).

- The condition has two immediate consequences:

$$v \; R \; \top$$

$$v \; R \; l_1 \; \wedge \; v \; R \; l_2 \quad \Rightarrow \quad v \; R \; (l_1 \sqcap l_2)$$

Condition (2)

$$(\forall l \in L' \subseteq L : v \; R \; l) \Rightarrow v \; R \; \left(\bigsqcap L'\right)$$

- The second condition says that there is always a best property for describing a value. This is important for having to perform only one analysis (using the best property, i.e. the greatest lower bound of the candidates) instead of several analyses (one for each of the candidates).

- The condition has two immediate consequences:

$$v \; R \; \top$$

$$v \; R \; l_1 \; \wedge \; v \; R \; l_2 \quad \Rightarrow \quad v \; R \; (l_1 \sqcap l_2)$$

A Concrete Example

The abstract properties **even** and **odd** do themselves not form a lattice L, but we can use – as usual: $L = \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$, where $\{\textbf{even}\}$ represents the definitive fact **even** and $\{\textbf{odd}\}$ the precise property **odd**; while the empty set $\bot = \emptyset$ represents an undefined parity and $\top = \{\textbf{even}, \textbf{odd}\}$ stands for any parity.

The conditions imposed on the correctness relation R and the property lattice L mean in this case:

(1) Any parity is always a valid description, e.g.

$$2\ R\ \{\textbf{even}\} \ \wedge\ \{\textbf{even}\} \sqsubseteq \top \Rightarrow 2\ R\ \top$$

(2) The most precise parity is valid, e.g.

$$(2\ R\ \{\textbf{even}\} \wedge 2\ R\ \top) \Rightarrow 2\ R\ (\{\textbf{even}\} \sqcap \top) \text{ i.e. } 2\ R\ \{\textbf{even}\}$$

A Concrete Example

The abstract properties **even** and **odd** do themselves not form a lattice L, but we can use – as usual: $L = \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$, where $\{\textbf{even}\}$ represents the definitive fact **even** and $\{\textbf{odd}\}$ the precise property **odd**; while the empty set $\bot = \emptyset$ represents an undefined parity and $\top = \{\textbf{even}, \textbf{odd}\}$ stands for any parity.

The conditions imposed on the correctness relation R and the property lattice L mean in this case:

(1) Any parity is always a valid description, e.g.

$$2\ R\ \{\textbf{even}\}\ \wedge\ \{\textbf{even}\} \sqsubseteq \top \Rightarrow 2\ R\ \top$$

(2) The most precise parity is valid, e.g.

$$(2\ R\ \{\textbf{even}\} \wedge 2\ R\ \top) \Rightarrow 2\ R\ (\{\textbf{even}\} \sqcap \top) \text{ i.e. } 2\ R\ \{\textbf{even}\}$$

A Concrete Example

The abstract properties **even** and **odd** do themselves not form a lattice L, but we can use – as usual: $L = \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$, where $\{\textbf{even}\}$ represents the definitive fact **even** and $\{\textbf{odd}\}$ the precise property **odd**; while the empty set $\bot = \emptyset$ represents an undefined parity and $\top = \{\textbf{even}, \textbf{odd}\}$ stands for any parity.

The conditions imposed on the correctness relation R and the property lattice L mean in this case:

(1) Any parity is always a valid description, e.g.

$$2\ R\ \{\textbf{even}\}\ \wedge\ \{\textbf{even}\} \sqsubseteq \top \Rightarrow 2\ R\ \top$$

(2) The most precise parity is valid, e.g.

$$(2\ R\ \{\textbf{even}\} \wedge 2\ R\ \top) \Rightarrow 2\ R\ (\{\textbf{even}\} \sqcap \top) \text{ i.e. } 2\ R\ \{\textbf{even}\}$$

A Concrete Example

The abstract properties **even** and **odd** do themselves not form a lattice L, but we can use – as usual: $L = \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$, where $\{\textbf{even}\}$ represents the definitive fact **even** and $\{\textbf{odd}\}$ the precise property **odd**; while the empty set $\bot = \emptyset$ represents an undefined parity and $\top = \{\textbf{even}, \textbf{odd}\}$ stands for any parity.

The conditions imposed on the correctness relation R and the property lattice L mean in this case:

(1) Any parity is always a valid description, e.g.

$$2\ R\ \{\textbf{even}\}\ \wedge\ \{\textbf{even}\} \sqsubseteq \top \Rightarrow 2\ R\ \top$$

(2) The most precise parity is valid, e.g.

$$(2\ R\ \{\textbf{even}\} \wedge 2\ R\ \top) \Rightarrow 2\ R\ (\{\textbf{even}\} \sqcap \top) \text{ i.e. } 2\ R\ \{\textbf{even}\}$$

Representation Functions

An alternative approach to the use of a correctness relation $R : V \times L \to \{\mathbf{tt}, \mathbf{ff}\}$ between values and properties is to use a representation function:

$$\beta : V \to L$$

The idea is that the function β maps a value v to the best property l describing it. The correctness criterion for the analysis will then be formulated as follows:

$$\beta(v_1) \sqsubseteq l_1 \ \wedge \ p \vdash v_1 \rightsquigarrow v_2 \ \wedge \ p \vdash l_1 \rhd l_2 \ \Rightarrow \ \beta(v_2) \sqsubseteq l_2$$

Representation Functions

An alternative approach to the use of a correctness relation $R : V \times L \to \{\mathbf{tt}, \mathbf{ff}\}$ between values and properties is to use a representation function:

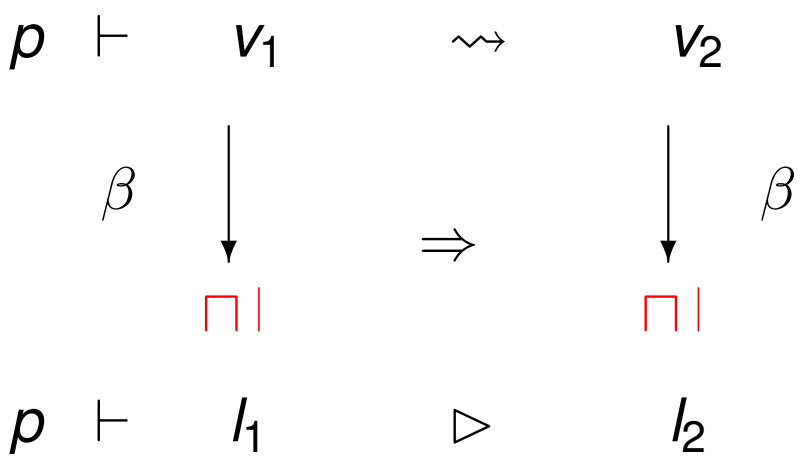
$$\beta : V \to L$$

The idea is that the function β maps a value v to the best property l describing it. The correctness criterion for the analysis will then be formulated as follows:

$$\beta(v_1) \sqsubseteq l_1 \ \wedge \ p \vdash v_1 \rightsquigarrow v_2 \ \wedge \ p \vdash l_1 \rhd l_2 \quad \Rightarrow \quad \beta(v_2) \sqsubseteq l_2$$

Correctness Criterion

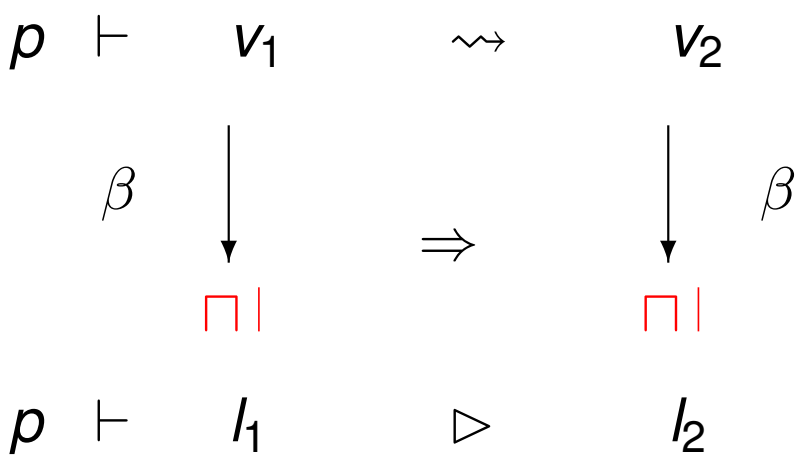
This is also expressed by the following diagram:



Thus the idea is that if the initial value v_1 is safely described by l_1 then the final value v_2 will be safely described by the result l_2 of the analysis.

Correctness Criterion

This is also expressed by the following diagram:



Thus the idea is that if the initial value v_1 is safely described by l_1 then the final value v_2 will be safely described by the result l_2 of the analysis.

Example: Parity

A repesentation function $\beta : \mathbb{Z} \to \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$ is easily defined by:

$$\beta(n) = \begin{cases} \{\textbf{even}\} & \text{if } \exists k \in \mathbb{Z} \text{ s.t. } n = 2k \\ \{\textbf{odd}\} & \text{otherwise} \end{cases}$$

Correctness implies that the abstract properties are dominated by the actual ones, e.g. $\beta(4) = \{\textbf{even}\} \sqsubseteq \top = \{\textbf{even}, \textbf{odd}\}$ is acceptable.

This means that we also could use as a representation function

$$\beta(n) = \top = \{\textbf{even}, \textbf{odd}\}$$

for all $n \in \mathbb{Z}$. Though this would be valid it would also be rather imprecise.

Example: Parity

A repesentation function $\beta : \mathbb{Z} \to \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$ is easily defined by:

$$\beta(n) = \begin{cases} \{\textbf{even}\} & \text{if } \exists k \in \mathbb{Z} \text{ s.t. } n = 2k \\ \{\textbf{odd}\} & \text{otherwise} \end{cases}$$

Correctness implies that the abstract properties are dominated by the actual ones, e.g. $\beta(4) = \{\textbf{even}\} \sqsubseteq \top = \{\textbf{even}, \textbf{odd}\}$ is acceptable.

This means that we also could use as a representation function

$$\beta(n) = \top = \{\textbf{even}, \textbf{odd}\}$$

for all $n \in \mathbb{Z}$. Though this would be valid it would also be rather imprecise.

Example: Parity

A repesentation function $\beta : \mathbb{Z} \to \mathcal{P}(\{\textbf{even}, \textbf{odd}\})$ is easily defined by:

$$\beta(n) = \begin{cases} \{\textbf{even}\} & \text{if } \exists k \in \mathbb{Z} \text{ s.t. } n = 2k \\ \{\textbf{odd}\} & \text{otherwise} \end{cases}$$

Correctness implies that the abstract properties are dominated by the actual ones, e.g. $\beta(4) = \{\textbf{even}\} \sqsubseteq \top = \{\textbf{even}, \textbf{odd}\}$ is acceptable.

This means that we also could use as a representation function

$$\beta(n) = \top = \{\textbf{even}, \textbf{odd}\}$$

for all $n \in \mathbb{Z}$. Though this would be valid it would also be rather imprecise.

Relations vs Functions

A correctness relation R and a representation relation β do correspond to each other.

We can show how to define a correctness relation R_β from a given representation function β:

$$v \; R_\beta \; I \quad \text{iff} \quad \beta(v) \sqsubseteq I$$

We also can show how to define a representation function β_R from a correctness relation R:

$$\beta_R(v) \quad = \quad \bigsqcap \{I \mid v \; R \; I\}$$

Relations vs Functions

A correctness relation R and a representation relation β do correspond to each other.

We can show how to define a correctness relation R_β from a given representation function β:

$$v \; R_\beta \; l \quad \text{iff} \quad \beta(v) \sqsubseteq l$$

We also can show how to define a representation function β_R from a correctness relation R:

$$\beta_R(v) \;=\; \bigsqcap \{l \mid v \; R \; l\}$$

Relations vs Functions

A correctness relation R and a representation relation β do correspond to each other.

We can show how to define a correctness relation R_β from a given representation function β:

$$v \; R_\beta \; l \quad \text{iff} \quad \beta(v) \sqsubseteq l$$

We also can show how to define a representation function β_R from a correctness relation R:

$$\beta_R(v) \quad = \quad \bigsqcap \{l \mid v \; R \; l\}$$

A Modest Generalisation

A program p specifies how one value v_1 is transformed into another value v_2: $p \vdash v_1 \rightsquigarrow v_2$. The analysis f_p of p specifies how a property l_1 is transformed into a property l_2: $p \vdash l_1 \rhd l_2$.

We allow $v_1 \in V_1$ and $v_2 \in V_2$ as well as $l_1 \in L_1$ and $l_2 \in L_2$ and refrain from imposing the restrictions that $V_1 = V_2$ or $L_1 = L_2$.

Correctness of and analysis $f_p = f$ of p now amounts to

$$v_1 \; R_1 \; l_1 \;\wedge\; p \vdash v_1 \rightsquigarrow v_2 \quad\Rightarrow\quad v_2 \; R_2 \; f_p(l_1)$$

for all $v_1 \in V_1$, $v_2 \in V_2$ and $l_1 \in L_1$.

A Modest Generalisation

A program p specifies how one value v_1 is transformed into another value v_2: $p \vdash v_1 \rightsquigarrow v_2$. The analysis f_p of p specifies how a property l_1 is transformed into a property l_2: $p \vdash l_1 \rhd l_2$.

We allow $v_1 \in V_1$ and $v_2 \in V_2$ as well as $l_1 \in L_1$ and $l_2 \in L_2$ and refrain from imposing the restrictions that $V_1 = V_2$ or $L_1 = L_2$.

Correctness of and analysis $f_p = f$ of p now amounts to

$$v_1 \; R_1 \; l_1 \;\wedge\; p \vdash v_1 \rightsquigarrow v_2 \quad\Rightarrow\quad v_2 \; R_2 \; f_p(l_1)$$

for all $v_1 \in V_1$, $v_2 \in V_2$ and $l_1 \in L_1$.

A Modest Generalisation

A program p specifies how one value v_1 is transformed into another value v_2: $p \vdash v_1 \rightsquigarrow v_2$. The analysis f_p of p specifies how a property l_1 is transformed into a property l_2: $p \vdash l_1 \rhd l_2$.

We allow $v_1 \in V_1$ and $v_2 \in V_2$ as well as $l_1 \in L_1$ and $l_2 \in L_2$ and refrain from imposing the restrictions that $V_1 = V_2$ or $L_1 = L_2$.

Correctness of and analysis $f_p = f$ of p now amounts to

$$v_1 \; R_1 \; l_1 \;\; \wedge \;\; p \vdash v_1 \rightsquigarrow v_2 \quad \Rightarrow \quad v_2 \; R_2 \; f_p(l_1)$$

for all $v_1 \in V_1$, $v_2 \in V_2$ and $l_1 \in L_1$.

Example: Integer Addition

Consider the program `plus` with the semantics given by

$$\texttt{plus} \vdash (z_1, z_2) \rightsquigarrow z_1 + z_2$$

where $z_1, z_2 \in \mathbf{Z}$.

A very precise analysis might use the complete lattices $L_1 = (\mathcal{P}(\mathbf{Z}), \subseteq)$ and $L_2 = (\mathcal{P}(\mathbf{Z} \times \mathbf{Z}), \subseteq)$ as follows:

$$f_{\texttt{plus}}(ZZ) = \{z_1 + z_2 \mid (z_1, z_2) \in ZZ\}$$

where $ZZ \subseteq \mathbf{Z} \times \mathbf{Z}$.

Example: Integer Addition

Consider the program `plus` with the semantics given by

$$\texttt{plus} \vdash (z_1, z_2) \rightsquigarrow z_1 + z_2$$

where $z_1, z_2 \in \mathbf{Z}$.

A very precise analysis might use the complete lattices $L_1 = (\mathcal{P}(\mathbf{Z}), \subseteq)$ and $L_2 = (\mathcal{P}(\mathbf{Z} \times \mathbf{Z}), \subseteq)$ as follows:

$$f_{\texttt{plus}}(ZZ) = \{z_1 + z_2 \mid (z_1, z_2) \in ZZ\}$$

where $ZZ \subseteq \mathbf{Z} \times \mathbf{Z}$.

Correctness: Addition

Consider now the correctness relations $R_{\mathbb{Z}}$ and $R_{\mathbb{Z}\times\mathbb{Z}}$ generated by the representation functions:

$$\begin{aligned} \beta_{\mathbb{Z}}(z) &= \{z\} \\ \beta_{\mathbb{Z}\times\mathbb{Z}}(z_1, z_2) &= \{(z_1, z_2)\} \end{aligned}$$

The correctness of the analysis of `plus` can be expressed by

$$\begin{aligned} \forall z_1, z_2, z, ZZ : \texttt{plus} \vdash (z_1, z_2) \rightsquigarrow z \\ \wedge\ (z_1, z_2)\ R_{\mathbb{Z}\times\mathbb{Z}}\ ZZ \Rightarrow\ z\ R_{\mathbb{Z}}\ f_{\texttt{plus}}(ZZ) \end{aligned}$$

Correctness: Addition

Consider now the correctness relations $R_{\mathbb{Z}}$ and $R_{\mathbb{Z}\times\mathbb{Z}}$ generated by the representation functions:

$$\begin{aligned} \beta_{\mathbb{Z}}(z) &= \{z\} \\ \beta_{\mathbb{Z}\times\mathbb{Z}}(z_1, z_2) &= \{(z_1, z_2)\} \end{aligned}$$

The correctness of the analysis of `plus` can be expressed by

$$\begin{aligned} \forall z_1, z_2, z, ZZ : \texttt{plus} \vdash (z_1, z_2) \rightsquigarrow z \\ \wedge\ (z_1, z_2)\ R_{\mathbb{Z}\times\mathbb{Z}}\ ZZ \Rightarrow\ z\ R_{\mathbb{Z}}\ f_{\texttt{plus}}(ZZ) \end{aligned}$$

Concrete Values vs Abstract Properties

The above example illustrates how the abstract concepts can give a more succinct formulation of the correctness of the analysis.

In the following we shall see several cases where we move freely between what we may call a “concrete” formulation of a property and an “abstract” formulation of the same property.

And we shall see that the latter often will allow us to reuse general results so that we do not have to redevelop parts of the theory for each application considered.

Concrete Values vs Abstract Properties

The above example illustrates how the abstract concepts can give a more succinct formulation of the correctness of the analysis.

In the following we shall see several cases where we move freely between what we may call a “concrete” formulation of a property and an “abstract” formulation of the same property.

And we shall see that the latter often will allow us to reuse general results so that we do not have to redevelop parts of the theory for each application considered.

Concrete Values vs Abstract Properties

The above example illustrates how the abstract concepts can give a more succinct formulation of the correctness of the analysis.

In the following we shall see several cases where we move freely between what we may call a “concrete” formulation of a property and an “abstract” formulation of the same property.

And we shall see that the latter often will allow us to reuse general results so that we do not have to redevelop parts of the theory for each application considered.

Aproximating Fixed Points

We shall now present a complete lattice that may be used for Array Bound Analysis, i.e. for determining if an array index is always within the bounds of the array – if this is the case then a number of run-time checks can be eliminated.

The lattice $(\mathbf{Interval}, \sqsubseteq)$ of intervals over $\mathbf{Z}$ may be described as follows. The elements are

$$\begin{aligned} \mathbf{Interval} &= \{\bot\} \\ &\cup \{[z_1, z_2] \mid z_1 \leq z_2, z_1 \in \mathbf{Z} \cup \{-\infty\}, z_2 \in \mathbf{Z} \cup \{\infty\}\} \end{aligned}$$

Aproximating Fixed Points

We shall now present a complete lattice that may be used for Array Bound Analysis, i.e. for determining if an array index is always within the bounds of the array – if this is the case then a number of run-time checks can be eliminated.

The lattice $(\mathbf{Interval}, \sqsubseteq)$ of intervals over $\mathbf{Z}$ may be described as follows. The elements are

$$\begin{aligned} \mathbf{Interval} &= \{\bot\} \\ &\cup \{[z_1, z_2] \mid z_1 \leq z_2, z_1 \in \mathbf{Z} \cup \{-\infty\}, z_2 \in \mathbf{Z} \cup \{\infty\}\} \end{aligned}$$

Interval Lattice

- The ordering $\leq$ on $\mathbf{Z}$ is extended to an ordering on $\mathbf{Z}' = \mathbf{Z} \cup \{-\infty, \infty\}$ by setting $-\infty \leq z$, $z \leq \infty$, and $-\infty \leq \infty$ (for all $z \in \mathbf{Z}$).
- Intuitively, $\bot$ denotes the empty interval and $[z_1, z_2]$ is the interval from z_1 to z_2 including end points if they are in $\mathbf{Z}$.
- We shall use *int* to range over elements of **Interval**.
- Intuitively, $int_1 \sqsubseteq int_2$ means $\{z \mid z \in int_1\} \subseteq \{z \mid z \in int_2\}$.

Interval Lattice

- The ordering $\leq$ on $\mathbf{Z}$ is extended to an ordering on $\mathbf{Z}' = \mathbf{Z} \cup \{-\infty, \infty\}$ by setting $-\infty \leq z$, $z \leq \infty$, and $-\infty \leq \infty$ (for all $z \in \mathbf{Z}$).
- Intuitively, $\bot$ denotes the empty interval and $[z_1, z_2]$ is the interval from z_1 to z_2 including end points if they are in $\mathbf{Z}$.
- We shall use *int* to range over elements of **Interval**.
- Intuitively, $int_1 \sqsubseteq int_2$ means $\{z \mid z \in int_1\} \subseteq \{z \mid z \in int_2\}$.

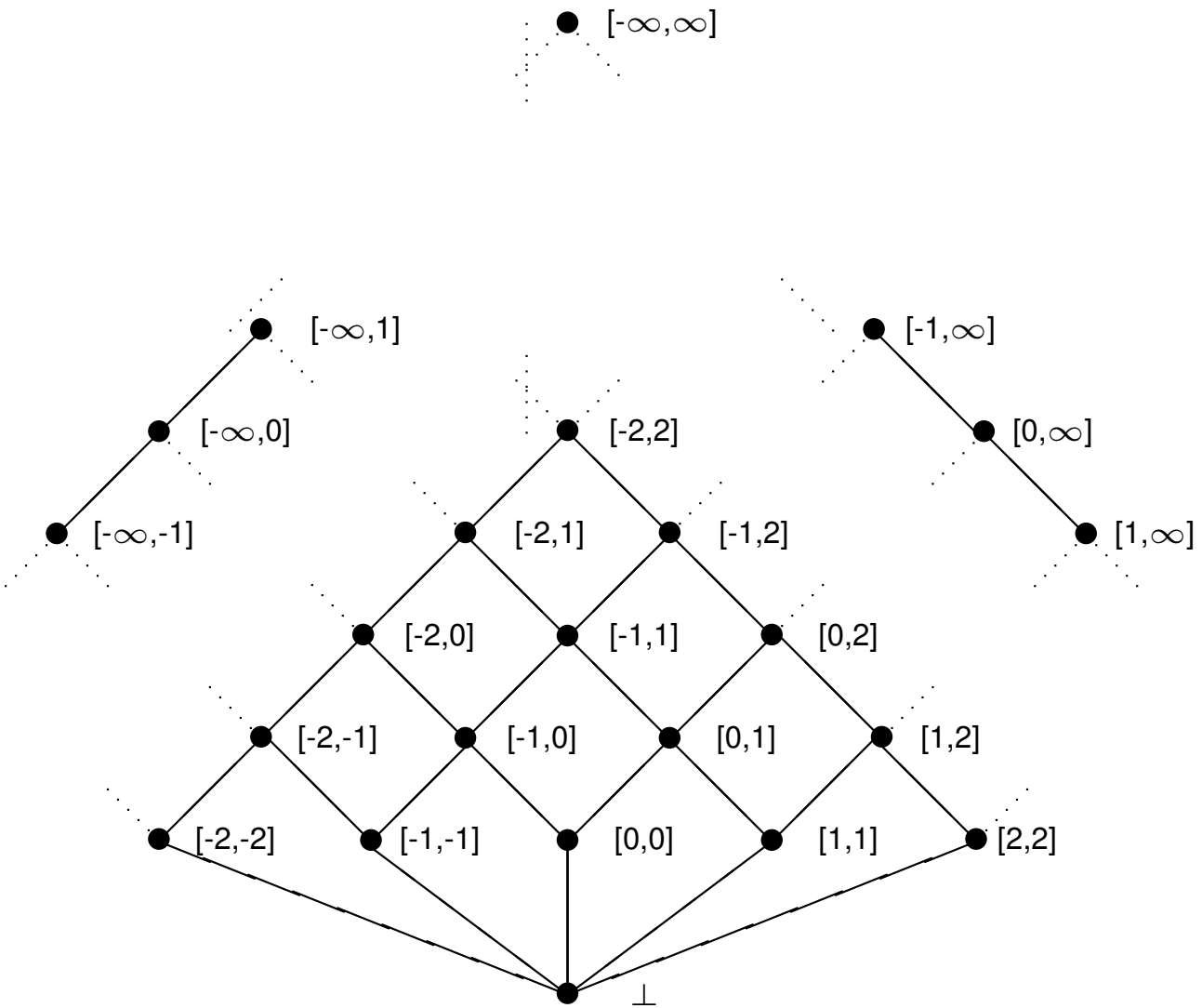
Interval Lattice

- The ordering $\leq$ on $\mathbf{Z}$ is extended to an ordering on $\mathbf{Z}' = \mathbf{Z} \cup \{-\infty, \infty\}$ by setting $-\infty \leq z$, $z \leq \infty$, and $-\infty \leq \infty$ (for all $z \in \mathbf{Z}$).
- Intuitively, $\bot$ denotes the empty interval and $[z_1, z_2]$ is the interval from z_1 to z_2 including end points if they are in $\mathbf{Z}$.
- We shall use *int* to range over elements of **Interval**.
- Intuitively, $int_1 \sqsubseteq int_2$ means $\{z \mid z \in int_1\} \subseteq \{z \mid z \in int_2\}$.

Interval Lattice

- The ordering $\leq$ on $\mathbf{Z}$ is extended to an ordering on $\mathbf{Z}' = \mathbf{Z} \cup \{-\infty, \infty\}$ by setting $-\infty \leq z$, $z \leq \infty$, and $-\infty \leq \infty$ (for all $z \in \mathbf{Z}$).
- Intuitively, $\bot$ denotes the empty interval and $[z_1, z_2]$ is the interval from z_1 to z_2 including end points if they are in $\mathbf{Z}$.
- We shall use *int* to range over elements of **Interval**.
- Intuitively, $int_1 \sqsubseteq int_2$ means $\{z \mid z \in int_1\} \subseteq \{z \mid z \in int_2\}$.

A Sketchy Picture of **Interval**



Lattice Structure

To give a succinct definition of the partial ordering we define the infimum and supremum operations on intervals as follows:

$$\inf(int) = \begin{cases} \infty & \text{if } int = \bot \\ z_1 & \text{if } int = [z_1, z_2] \end{cases}$$

$$\sup(int) = \begin{cases} -\infty & \text{if } int = \bot \\ z_2 & \text{if } int = [z_1, z_2] \end{cases}$$

This allows us to define a partial order:

$$int_1 \sqsubseteq int_2 \quad \text{iff} \quad \inf(int_2) \leq \inf(int_1) \wedge \sup(int_1) \leq \sup(int_2)$$

Lattice Structure

To give a succinct definition of the partial ordering we define the infimum and supremum operations on intervals as follows:

$$\inf(int) = \begin{cases} \infty & \text{if } int = \bot \\ z_1 & \text{if } int = [z_1, z_2] \end{cases}$$

$$\sup(int) = \begin{cases} -\infty & \text{if } int = \bot \\ z_2 & \text{if } int = [z_1, z_2] \end{cases}$$

This allows us to define a partial order:

$$int_1 \sqsubseteq int_2 \quad \text{iff} \quad \inf(int_2) \leq \inf(int_1) \wedge \sup(int_1) \leq \sup(int_2)$$

Analysis

- Given a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ the effect of a program, p, in transforming one property, l_1, into another, l_2, i.e. $p \vdash l_1 \rhd l_2$, is normally given by an equation

$$f_p(l_1) = f(l_1) = l_2$$

  for a monotone function $f : L \to L$ dependent on the program p.

- Note that the demand that $f = f_p$ is monotone is very natural for program analysis; it merely says that if l_1' describes at least the values that l_1 does then also $f(l_1')$ describes at least the values that $f(l_1)$ does.

Analysis

- Given a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ the effect of a program, p, in transforming one property, l_1, into another, l_2, i.e. $p \vdash l_1 \rhd l_2$, is normally given by an equation

$$f_p(l_1) = f(l_1) = l_2$$

  for a monotone function $f : L \to L$ dependent on the program p.

- Note that the demand that $f = f_p$ is monotone is very natural for program analysis; it merely says that if l_1' describes at least the values that l_1 does then also $f(l_1')$ describes at least the values that $f(l_1)$ does.

Approximating Solutions

- For recursive or iterative program constructs we ideally want to obtain the least fixed point, $lfp(f)$, as the result of a finite iterative process.
- However, the iterative sequence $(f^n(\bot))_n$ needs not to eventually stabilise nor need its least upper bound necessarily equal $lfp(f)$ (no ACC/DCC).
- This might suggest considering the iterative sequence $(f^n(\top))_n$ and, even when it does not eventually stabilise, we can always terminate the iteration at an arbitrary point in time.
  This is safe but grossly imprecise in practice.

Approximating Solutions

- For recursive or iterative program constructs we ideally want to obtain the least fixed point, $lfp(f)$, as the result of a finite iterative process.
- However, the iterative sequence $(f^n(\bot))_n$ needs not to eventually stabilise nor need its least upper bound necessarily equal $lfp(f)$ (no ACC/DCC).
- This might suggest considering the iterative sequence $(f^n(\top))_n$ and, even when it does not eventually stabilise, we can always terminate the iteration at an arbitrary point in time.
  This is safe but grossly imprecise in practice.

Approximating Solutions

- For recursive or iterative program constructs we ideally want to obtain the least fixed point, $lfp(f)$, as the result of a finite iterative process.
- However, the iterative sequence $(f^n(\bot))_n$ needs not to eventually stabilise nor need its least upper bound necessarily equal $lfp(f)$ (no ACC/DCC).
- This might suggest considering the iterative sequence $(f^n(\top))_n$ and, even when it does not eventually stabilise, we can always terminate the iteration at an arbitrary point in time.
  This is safe but grossly imprecise in practice.

Iterations and Convergence

In Denotational Semantics it is customary to iterate to the least fixed point by taking the least upper bound of $(f^n(\bot))_n$.

However, we have not imposed any continuity requirements on f – e.g. that $f(\bigsqcup_n l_n) = \bigsqcup_n (f(l_n))$ for all ascending chains $(l_n)_n$ – and consequently we cannot be sure to actually reach the fixed point. In a similar way one could consider the greatest lower bound of the sequence $(f^n(\top))_n$. One can show that

$$f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \sqsubseteq lfp(f) \sqsubseteq gfp(f) \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top)$$

Indeed all inequalities (i.e. $\sqsubseteq$) can be strict (i.e. $\neq$).

We need techniques to guarantee convergence to a solution.

Iterations and Convergence

In Denotational Semantics it is customary to iterate to the least fixed point by taking the least upper bound of $(f^n(\bot))_n$.

However, we have not imposed any continuity requirements on f – e.g. that $f(\bigsqcup_n l_n) = \bigsqcup_n (f(l_n))$ for all ascending chains $(l_n)_n$ – and consequently we cannot be sure to actually reach the fixed point. In a similar way one could consider the greatest lower bound of the sequence $(f^n(\top))_n$. One can show that

$$f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \sqsubseteq lfp(f) \sqsubseteq gfp(f) \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top)$$

Indeed all inequalities (i.e. $\sqsubseteq$) can be strict (i.e. $\neq$).

We need techniques to guarantee convergence to a solution.

Iterations and Convergence

In Denotational Semantics it is customary to iterate to the least fixed point by taking the least upper bound of $(f^n(\bot))_n$.

However, we have not imposed any continuity requirements on f – e.g. that $f(\bigsqcup_n l_n) = \bigsqcup_n (f(l_n))$ for all ascending chains $(l_n)_n$ – and consequently we cannot be sure to actually reach the fixed point. In a similar way one could consider the greatest lower bound of the sequence $(f^n(\top))_n$. One can show that

$$f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \sqsubseteq lfp(f) \sqsubseteq gfp(f) \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top)$$

Indeed all inequalities (i.e. $\sqsubseteq$) can be strict (i.e. $\neq$).

We need techniques to guarantee convergence to a solution.

Iterations and Convergence

In Denotational Semantics it is customary to iterate to the least fixed point by taking the least upper bound of $(f^n(\bot))_n$.

However, we have not imposed any continuity requirements on f – e.g. that $f(\bigsqcup_n l_n) = \bigsqcup_n (f(l_n))$ for all ascending chains $(l_n)_n$ – and consequently we cannot be sure to actually reach the fixed point. In a similar way one could consider the greatest lower bound of the sequence $(f^n(\top))_n$. One can show that

$$f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \sqsubseteq \mathit{lfp}(f) \sqsubseteq \mathit{gfp}(f) \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top)$$

Indeed all inequalities (i.e. $\sqsubseteq$) can be strict (i.e. $\neq$).

We need techniques to guarantee convergence to a solution.

Widenings to Gurantee Stabilisation

Since we cannot guarantee that the iterative sequence $(f^n(\bot))_n$ eventually stabilises nor that its least upper bound necessarily equals $lfp(f)$, we must consider a way of approximating $lfp(f)$.

The idea is now to replace it by a new sequence $(f^n_\nabla)_n$ that is known to eventually stabilise and to do so with a value that is a safe (upper) approximation of the least fixed point.

The construction of the new sequence is parameterised on the operator ∇, called a widening operator; the precision of the approximated fixed point as well as the cost of computing it depends on the actual choice of widening operator.

Widenings to Gurantee Stabilisation

Since we cannot guarantee that the iterative sequence $(f^n(\bot))_n$ eventually stabilises nor that its least upper bound necessarily equals $lfp(f)$, we must consider a way of approximating $lfp(f)$.

The idea is now to replace it by a new sequence $(f^n_\nabla)_n$ that is known to eventually stabilise and to do so with a value that is a safe (upper) approximation of the least fixed point.

The construction of the new sequence is parameterised on the operator ∇, called a widening operator; the precision of the approximated fixed point as well as the cost of computing it depends on the actual choice of widening operator.

Widenings to Gurantee Stabilisation

Since we cannot guarantee that the iterative sequence $(f^n(\bot))_n$ eventually stabilises nor that its least upper bound necessarily equals $lfp(f)$, we must consider a way of approximating $lfp(f)$.

The idea is now to replace it by a new sequence $(f^n_\nabla)_n$ that is known to eventually stabilise and to do so with a value that is a safe (upper) approximation of the least fixed point.

The construction of the new sequence is parameterised on the operator ∇, called a widening operator; the precision of the approximated fixed point as well as the cost of computing it depends on the actual choice of widening operator.

Upper Bound Operators

In preparation for the development, an operator $\check{\sqcup} : L \times L \to L$ on a complete lattice $L = (L, \sqsubseteq)$ is called an upper bound operator if

$$l_1 \sqsubseteq (l_1 \check{\sqcup} l_2) \sqsupseteq l_2$$

for all $l_1, l_2 \in L$, i.e. it always returns an element larger than both its arguments. Note that we do *not* require $\check{\sqcup}$ to be monotone, commutative, associative, nor absorptive (i.e. that $l \check{\sqcup} l = l$).

Widening Operators

We can now introduce a special class of upper bound operators that will help us to approximate the least fixed points: An operator $\nabla : L \times L \to L$ is a widening operator if and only if:

- it is an upper bound operator, and
- for all ascending chains $(l_n)_n$ the associated ascending chain $(l_n^{\nabla})_n$ eventually stabilises.

The idea is as follows: Given a monotone function $f : L \to L$ on a complete lattice L and given a widening operator ∇ on L, we shall calculate the sequence $(f_{\nabla}^n)_n$ defined by

$$f_{\nabla}^n = \begin{cases} \bot & \text{if } n = 0 \\ f_{\nabla}^{n-1} & \text{if } n > 0 \ \wedge \ f(f_{\nabla}^{n-1}) \sqsubseteq f_{\nabla}^{n-1} \\ f_{\nabla}^{n-1} \ \nabla \ f(f_{\nabla}^{n-1}) & \text{otherwise} \end{cases}$$

Widening Operators

We can now introduce a special class of upper bound operators that will help us to approximate the least fixed points: An operator $\nabla : L \times L \to L$ is a widening operator if and only if:

- it is an upper bound operator, and
- for all ascending chains $(l_n)_n$ the associated ascending chain $(l_n^{\nabla})_n$ eventually stabilises.

The idea is as follows: Given a monotone function $f : L \to L$ on a complete lattice L and given a widening operator ∇ on L, we shall calculate the sequence $(f_{\nabla}^n)_n$ defined by

$$f_{\nabla}^n = \begin{cases} \bot & \text{if } n = 0 \\ f_{\nabla}^{n-1} & \text{if } n > 0 \ \wedge \ f(f_{\nabla}^{n-1}) \sqsubseteq f_{\nabla}^{n-1} \\ f_{\nabla}^{n-1} \ \nabla \ f(f_{\nabla}^{n-1}) & \text{otherwise} \end{cases}$$

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Stabilisation Using Widening

One can show that for any upper bound operator ∇ we have:

(i) the sequence $(f^n_\nabla)_n$ is an ascending chain;

(ii) if $f(f^m_\nabla) \sqsubseteq f^m_\nabla$ for some m then $(f^n_\nabla)_n$ eventually stabilises and $\forall n > m : f^n_\nabla = f^m_\nabla$ and $\bigsqcup_n f^n_\nabla = f^m_\nabla$;

(iii) if $(f^n_\nabla)_n$ eventually stabilises then there exists an m such that $f(f^m_\nabla) \sqsubseteq f^m_\nabla$; and

(iv) if $(f^n_\nabla)_n$ eventually stabilises then $\bigsqcup_n f^n_\nabla \sqsupseteq \mathit{lfp}(f)$.

Moreover, if ∇ is a widening operator then the ascending chain $(f^n_\nabla)_n$ eventually stabilises.

Example: Widening on **Interval**

Consider the complete lattice (**Interval**, $\sqsubseteq$). Let K be a *finite* set of integers, e.g. the set of integers explicitly mentioned in a given program.

We shall now define a widening operator ∇_K based on K. The idea is that $[z_1, z_2] \nabla_K [z_3, z_4]$ is something like

$$[\ \mathrm{LB}(z_1, z_3)\ ,\ \mathrm{UB}(z_2, z_4)\ ]$$

where $\mathrm{LB}(z_1, z_3) \in \{z_1\} \cup K \cup \{-\infty\}$ is the best possible lower bound and $\mathrm{UB}(z_2, z_4) \in \{z_2\} \cup K \cup \{\infty\}$ is the best possible upper bound.

In this way a change in any of the bounds of the interval $[z_1, z_2]$ can only take place in a finite number of steps (corresponding to the elements of K).

Example: Widening on **Interval**

Consider the complete lattice (**Interval**, $\sqsubseteq$). Let K be a *finite* set of integers, e.g. the set of integers explicitly mentioned in a given program.

We shall now define a widening operator ∇_K based on K. The idea is that $[z_1, z_2] \nabla_K [z_3, z_4]$ is something like

$$[\ \mathrm{LB}(z_1, z_3)\ ,\ \mathrm{UB}(z_2, z_4)\ ]$$

where $\mathrm{LB}(z_1, z_3) \in \{z_1\} \cup K \cup \{-\infty\}$ is the best possible lower bound and $\mathrm{UB}(z_2, z_4) \in \{z_2\} \cup K \cup \{\infty\}$ is the best possible upper bound.

In this way a change in any of the bounds of the interval $[z_1, z_2]$ can only take place in a finite number of steps (corresponding to the elements of K).

Example: Widening on **Interval**

Consider the complete lattice (**Interval**, $\sqsubseteq$). Let K be a *finite* set of integers, e.g. the set of integers explicitly mentioned in a given program.

We shall now define a widening operator ∇_K based on K. The idea is that $[z_1, z_2] \nabla_K [z_3, z_4]$ is something like

$$[\ \mathrm{LB}(z_1, z_3)\ ,\ \mathrm{UB}(z_2, z_4)\ ]$$

where $\mathrm{LB}(z_1, z_3) \in \{z_1\} \cup K \cup \{-\infty\}$ is the best possible lower bound and $\mathrm{UB}(z_2, z_4) \in \{z_2\} \cup K \cup \{\infty\}$ is the best possible upper bound.

In this way a change in any of the bounds of the interval $[z_1, z_2]$ can only take place in a finite number of steps (corresponding to the elements of K).

Example: Upper and Lower Bounds

For the precise definition we let $z_i \in \mathbf{Z} \cup \{-\infty, \infty\}$ and write:

$$\mathrm{LB}_K(z_1, z_3) = \begin{cases} z_1 & \text{if } z_1 \leq z_3 \\ k & \text{if } z_3 < z_1 \ \wedge \ k = \max\{k \in K \mid k \leq z_3\} \\ -\infty & \text{if } z_3 < z_1 \ \wedge \ \forall k \in K : z_3 < k \end{cases}$$

$$\mathrm{UB}_K(z_2, z_4) = \begin{cases} z_2 & \text{if } z_4 \leq z_2 \\ k & \text{if } z_2 < z_4 \ \wedge \ k = \min\{k \in K \mid z_4 \leq k\} \\ \infty & \text{if } z_2 < z_4 \ \wedge \ \forall k \in K : k < z_4 \end{cases}$$

Example: Upper and Lower Bounds

For the precise definition we let $z_i \in \mathbf{Z} \cup \{-\infty, \infty\}$ and write:

$$\mathrm{LB}_K(z_1, z_3) = \begin{cases} z_1 & \text{if } z_1 \leq z_3 \\ k & \text{if } z_3 < z_1 \ \wedge \ k = \max\{k \in K \mid k \leq z_3\} \\ -\infty & \text{if } z_3 < z_1 \ \wedge \ \forall k \in K : z_3 < k \end{cases}$$

$$\mathrm{UB}_K(z_2, z_4) = \begin{cases} z_2 & \text{if } z_4 \leq z_2 \\ k & \text{if } z_2 < z_4 \ \wedge \ k = \min\{k \in K \mid z_4 \leq k\} \\ \infty & \text{if } z_2 < z_4 \ \wedge \ \forall k \in K : k < z_4 \end{cases}$$

Example: Widening on **Interval**

We can now define $\nabla = \nabla_K$ by:

$$int_1 \nabla int_2 = \begin{cases} \bot & \text{if } int_1 = int_2 = \bot \\ [\ \mathrm{LB}_K(\inf(int_1), \inf(int_2)), \mathrm{UB}_K(\sup(int_1), \sup(int_2))\ ] & \\ & \text{otherwise} \end{cases}$$

Example: As an example consider the ascending chain $(int_n)_n$:

$$[0,1], [0,2], [0,3], [0,4], [0,5], [0,6], [0,7], \cdots$$

and assume that $K = \{3, 5\}$. Then $(int_n^{\nabla})_n$ will be the chain

$$[0,1], [0,3], [0,3], [0,5], [0,5], [0,\infty], [0,\infty], \cdots$$

Example: Widening on **Interval**

We can now define $\nabla = \nabla_K$ by:

$$int_1 \nabla int_2 = \begin{cases} \bot & \text{if } int_1 = int_2 = \bot \\ [\ \mathrm{LB}_K(\inf(int_1), \inf(int_2)), \mathrm{UB}_K(\sup(int_1), \sup(int_2))\ ] & \\ & \text{otherwise} \end{cases}$$

Example: As an example consider the ascending chain $(int_n)_n$:

$$[0,1], [0,2], [0,3], [0,4], [0,5], [0,6], [0,7], \cdots$$

and assume that $K = \{3, 5\}$. Then $(int_n^{\nabla})_n$ will be the chain

$$[0,1], [0,3], [0,3], [0,5], [0,5], [0,\infty], [0,\infty], \cdots$$

Abstractions and Concretisations

Sometimes calculations on a complete lattice L may be too costly or even uncomputable and this may motivate replacing L by a simpler lattice M.

An example is when L is the powerset of integers and M is a lattice of intervals, or even just the parity lattice.

To express the relationship between L and M it is customary to use an abstraction function

$$\alpha : L \to M$$

and a concretisation function

$$\gamma : M \to L$$

Abstractions and Concretisations

Sometimes calculations on a complete lattice L may be too costly or even uncomputable and this may motivate replacing L by a simpler lattice M.

An example is when L is the powerset of integers and M is a lattice of intervals, or even just the parity lattice.

To express the relationship between L and M it is customary to use an abstraction function

$$\alpha : L \to M$$

and a concretisation function

$$\gamma : M \to L$$

Abstractions and Concretisations

Sometimes calculations on a complete lattice L may be too costly or even uncomputable and this may motivate replacing L by a simpler lattice M.

An example is when L is the powerset of integers and M is a lattice of intervals, or even just the parity lattice.

To express the relationship between L and M it is customary to use an abstraction function

$$\alpha : L \to M$$

and a concretisation function

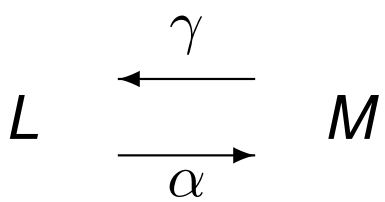
$$\gamma : M \to L$$

Galois Connections

We shall write

$$(L, \alpha, \gamma, M)$$

or



We define (L, α, γ, M) to be a Galois connection between the complete lattices $(L, \sqsubseteq)$ and $(M, \sqsubseteq)$ if and only if

$\alpha : L \to M$ and $\gamma : M \to L$ are monotone functions

that satisfy:

$$\gamma \circ \alpha \sqsupseteq id_L$$

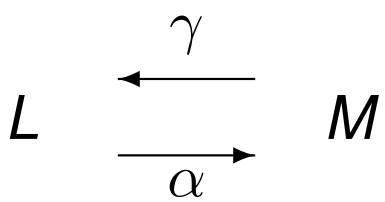
$$\alpha \circ \gamma \sqsubseteq id_M$$

Galois Connections

We shall write

$$(L, \alpha, \gamma, M)$$

or



We define (L, α, γ, M) to be a Galois connection between the complete lattices $(L, \sqsubseteq)$ and $(M, \sqsubseteq)$ if and only if

$\alpha : L \to M$ and $\gamma : M \to L$ are monotone functions

that satisfy:

$$\gamma \circ \alpha \sqsupseteq id_L$$

$$\alpha \circ \gamma \sqsubseteq id_M$$

Adjunctions

We define (L, α, γ, M) to be an adjunction between complete lattices $L = (L, \sqsubseteq)$ and $M = (M, \sqsubseteq)$ if and only if

$$\alpha : L \to M \text{ and } \gamma : M \to L \text{ are total functions}$$

that satisfy

$$\alpha(l) \sqsubseteq m \quad \Leftrightarrow \quad l \sqsubseteq \gamma(m)$$

for all $l \in L$ and $m \in M$.

Representation Functions and Galois Connections

Representation functions can be used to define Galois connections.

So consider once again the representation function $\beta : V \to L$ mapping the values of V to the properties of the complete lattice L.

It gives rise to a Galois connection

$$(\mathcal{P}(V), \alpha, \gamma, L)$$

between $\mathcal{P}(V)$ and L where the abstraction and concretisation functions are defined by

$$\begin{aligned} \alpha(V') &= \bigsqcup\{\beta(v) \mid v \in V'\} \\ \gamma(l) &= \{v \in V \mid \beta(v) \sqsubseteq l\} \end{aligned}$$

for $V' \subseteq V$ and $l \in L$.

Representation Functions and Galois Connections

Representation functions can be used to define Galois connections.
So consider once again the representation function $\beta : V \to L$ mapping the values of V to the properties of the complete lattice L.
It gives rise to a Galois connection

$$(\mathcal{P}(V), \alpha, \gamma, L)$$

between $\mathcal{P}(V)$ and L where the abstraction and concretisation functions are defined by

$$\begin{aligned} \alpha(V') &= \bigsqcup\{\beta(v) \mid v \in V'\} \\ \gamma(l) &= \{v \in V \mid \beta(v) \sqsubseteq l\} \end{aligned}$$

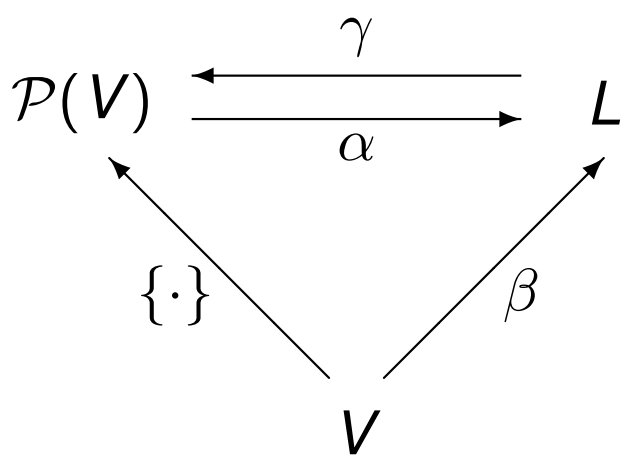
for $V' \subseteq V$ and $l \in L$.

The Corresponding Adjunction

Let us pause for a minute to see that this indeed defines an adjunction:

$$\begin{aligned} \alpha(V') \sqsubseteq l &\iff \bigsqcup\{\beta(v) \mid v \in V'\} \sqsubseteq l \\ &\iff \forall v \in V' : \beta(v) \sqsubseteq l \\ &\iff V' \subseteq \gamma(l) \end{aligned}$$

It is also immediate that $\alpha(\{v\}) = \beta(v)$ as illustrated by the diagram:



Extraction Functions and Galois Connections

A special case of the above construction that is frequently useful is when $L = (\mathcal{P}(D), \subseteq)$ for some set D and we have an extraction function: $\eta : V \to D$.

We will then define the representation function $\beta_\eta : V \to \mathcal{P}(D)$ by $\beta_\eta(v) = \{\eta(v)\}$ and the Galois connection between $\mathcal{P}(V)$ and $\mathcal{P}(D)$ will now be written

$$(\mathcal{P}(V), \alpha_\eta, \gamma_\eta, \mathcal{P}(D))$$

where

$$\begin{array}{rcccl} \alpha_\eta(V') & = & \bigcup\{\beta_\eta(v) \mid v \in V'\} & = & \{\eta(v) \mid v \in V'\} \\ \gamma_\eta(D') & = & \{v \in V \mid \beta_\eta(v) \subseteq D'\} & = & \{v \mid \eta(v) \in D'\} \end{array}$$

Extraction Functions and Galois Connections

A special case of the above construction that is frequently useful is when $L = (\mathcal{P}(D), \subseteq)$ for some set D and we have an extraction function: $\eta : V \to D$.

We will then define the representation function $\beta_\eta : V \to \mathcal{P}(D)$ by $\beta_\eta(v) = \{\eta(v)\}$ and the Galois connection between $\mathcal{P}(V)$ and $\mathcal{P}(D)$ will now be written

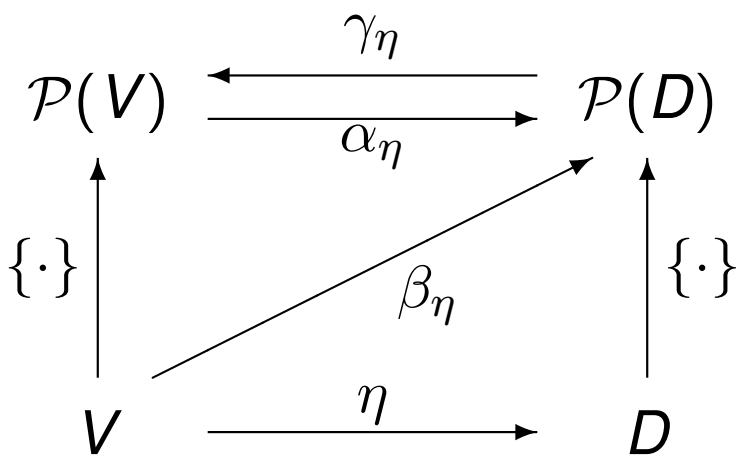
$$(\mathcal{P}(V), \alpha_\eta, \gamma_\eta, \mathcal{P}(D))$$

where

$$\begin{array}{rcccl} \alpha_\eta(V') & = & \bigcup\{\beta_\eta(v) \mid v \in V'\} & = & \{\eta(v) \mid v \in V'\} \\ \gamma_\eta(D') & = & \{v \in V \mid \beta_\eta(v) \subseteq D'\} & = & \{v \mid \eta(v) \in D'\} \end{array}$$

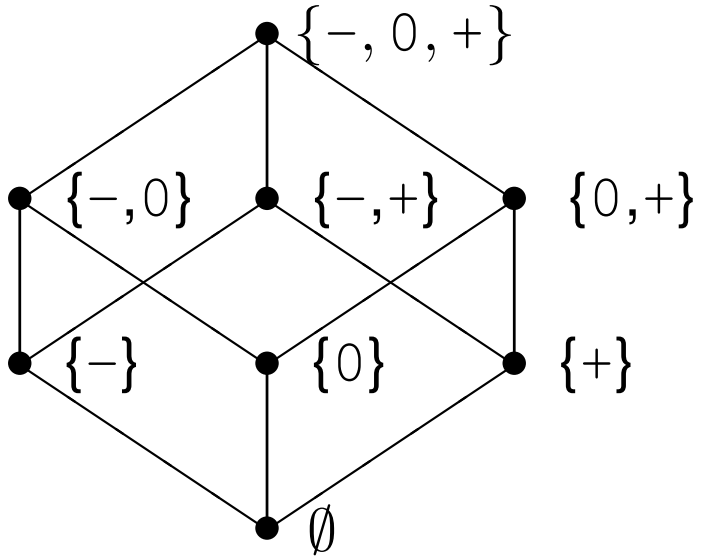
Representation and Abstraction

The relationship between η, β_η, α_η and γ_η is illustrated by the diagram:



Example: Sign Lattice

Let us consider the two complete lattices $(\mathcal{P}(\mathbf{Z}), \subseteq)$ and $(\mathcal{P}(\mathbf{Sign}), \subseteq)$ where $\mathbf{Sign} = \{-, 0, +\}$.



Example: Sign Extraction

The extraction function

$$\text{sign} : \mathbf{Z} \rightarrow \mathbf{Sign}$$

simply defines the signs of the integers and is specified by:

$$\text{sign}(z) = \begin{cases} - & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ + & \text{if } z > 0 \end{cases}$$

Example: Sign Abstraction

The above construction then gives us a Galois connection

$$(\mathcal{P}(\mathbf{Z}), \alpha_{\text{sign}}, \gamma_{\text{sign}}, \mathcal{P}(\mathbf{Sign}))$$

with

$$\begin{aligned} \alpha_{\text{sign}}(Z) &= \{\text{sign}(z) \mid z \in Z\} \\ \gamma_{\text{sign}}(S) &= \{z \in \mathbf{Z} \mid \text{sign}(z) \in S\} \end{aligned}$$

where $Z \subseteq \mathbf{Z}$ and $S \subseteq \mathbf{Sign}$.

Properties of Galois Connections

If (L, α, γ, M) is a Galois connection then:

(i) α uniquely determines γ by $\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ and γ uniquely determines α by $\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$.

(ii) α is completely additive and γ is completely multiplicative.

In particular $\alpha(\bot) = \bot$ and $\gamma(\top) = \top$.

If (L, α, γ, M) is a Galois connection then $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

Properties of Galois Connections

If (L, α, γ, M) is a Galois connection then:

(i) α uniquely determines γ by $\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ and γ uniquely determines α by $\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$.

(ii) α is completely additive and γ is completely multiplicative.

In particular $\alpha(\bot) = \bot$ and $\gamma(\top) = \top$.

If (L, α, γ, M) is a Galois connection then $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

Properties of Galois Connections

If (L, α, γ, M) is a Galois connection then:

(i) α uniquely determines γ by $\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ and γ uniquely determines α by $\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$.

(ii) α is completely additive and γ is completely multiplicative.

In particular $\alpha(\bot) = \bot$ and $\gamma(\top) = \top$.

If (L, α, γ, M) is a Galois connection then $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

Properties of Galois Connections

If (L, α, γ, M) is a Galois connection then:

(i) α uniquely determines γ by $\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ and γ uniquely determines α by $\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$.

(ii) α is completely additive and γ is completely multiplicative.

In particular $\alpha(\bot) = \bot$ and $\gamma(\top) = \top$.

If (L, α, γ, M) is a Galois connection then $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

Properties of Galois Connections

If (L, α, γ, M) is a Galois connection then:

(i) α uniquely determines γ by $\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ and γ uniquely determines α by $\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$.

(ii) α is completely additive and γ is completely multiplicative.

In particular $\alpha(\bot) = \bot$ and $\gamma(\top) = \top$.

If (L, α, γ, M) is a Galois connection then $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

Galois Connections and Correctness

If there is a Galois connection (L, α, γ, M) between L and M then we can construct a correctness relation between V and M and a representation function from V to M.

Let $R : V \times L \to \{\textbf{true}, \textbf{false}\}$ be a correctness relation.

Furthermore, let (L, α, γ, M) be a Galois connection between the complete lattices L and M.

It is then natural to define $S : V \times M \to \{\textbf{true}, \textbf{false}\}$ by

$$v \; S \; m \;\text{ iff }\; v \; R \; (\gamma(m))$$

Galois Connections and Correctness

If there is a Galois connection (L, α, γ, M) between L and M then we can construct a correctness relation between V and M and a representation function from V to M.

Let $R : V \times L \to \{\textbf{true}, \textbf{false}\}$ be a correctness relation.

Furthermore, let (L, α, γ, M) be a Galois connection between the complete lattices L and M.

It is then natural to define $S : V \times M \to \{\textbf{true}, \textbf{false}\}$ by

$$v \; S \; m \;\text{ iff }\; v \; R \; (\gamma(m))$$

Galois Connections and Correctness

If there is a Galois connection (L, α, γ, M) between L and M then we can construct a correctness relation between V and M and a representation function from V to M.

Let $R : V \times L \to \{\textbf{true}, \textbf{false}\}$ be a correctness relation.

Furthermore, let (L, α, γ, M) be a Galois connection between the complete lattices L and M.

It is then natural to define $S : V \times M \to \{\textbf{true}, \textbf{false}\}$ by

$$v \; S \; m \;\text{ iff }\; v \; R \; (\gamma(m))$$

Galois Connections and Correctness

If there is a Galois connection (L, α, γ, M) between L and M then we can construct a correctness relation between V and M and a representation function from V to M.

Let $R : V \times L \to \{\textbf{true}, \textbf{false}\}$ be a correctness relation.

Furthermore, let (L, α, γ, M) be a Galois connection between the complete lattices L and M.

It is then natural to define $S : V \times M \to \{\textbf{true}, \textbf{false}\}$ by

$$v \; S \; m \;\; \text{iff} \;\; v \; R \; (\gamma(m))$$

Representation and Correctness

Continuing the above line of reasoning assume now that R is generated the representation function $\beta : V \to L$,
i.e. $v \; R \; l \;\Leftrightarrow\; \beta(v) \sqsubseteq l$.

Since (L, α, γ, M) is a Galois connection and hence an adjunction we may calculate

$$\begin{aligned} v \; S \; m &\Leftrightarrow v \; R \; (\gamma(m)) \\ &\Leftrightarrow \beta(v) \sqsubseteq \gamma(m) \\ &\Leftrightarrow (\alpha \circ \beta)(v) \sqsubseteq m \end{aligned}$$

showing that S is *generated by* $\alpha \circ \beta : V \to M$.

Representation and Correctness

Continuing the above line of reasoning assume now that R is generated the representation function $\beta : V \to L$,
i.e. $v \; R \; l \;\Leftrightarrow\; \beta(v) \sqsubseteq l$.

Since (L, α, γ, M) is a Galois connection and hence an adjunction we may calculate

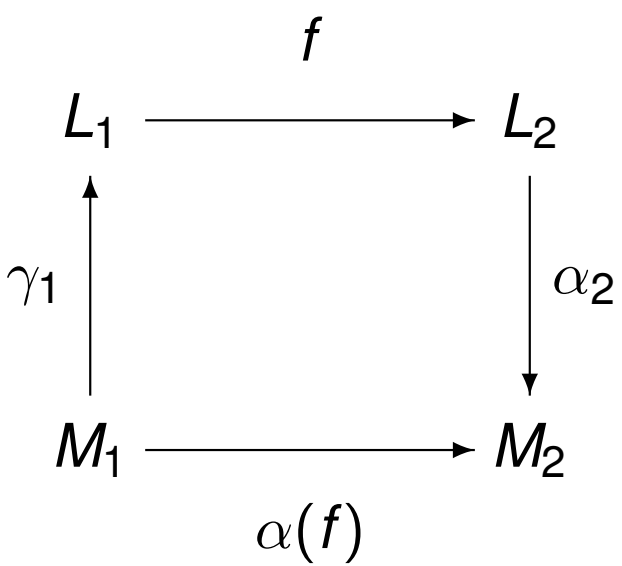
$$\begin{aligned} v \; S \; m &\Leftrightarrow v \; R \; (\gamma(m)) \\ &\Leftrightarrow \beta(v) \sqsubseteq \gamma(m) \\ &\Leftrightarrow (\alpha \circ \beta)(v) \sqsubseteq m \end{aligned}$$

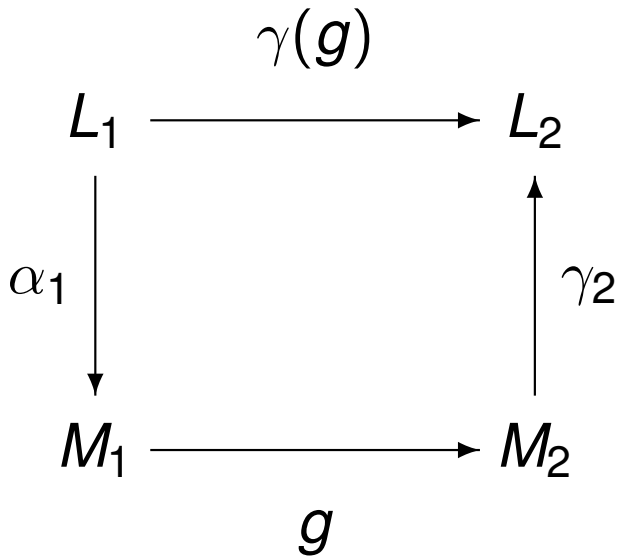
showing that S is *generated by* $\alpha \circ \beta : V \to M$.

Monotone Function Space.

Let $(L_1, \alpha_1, \gamma_1, M_1)$ and $(L_2, \alpha_2, \gamma_2, M_2)$ be Galois connections. Then we obtain the Galois connection $(L_1 \to L_2, \alpha, \gamma, M_1 \to M_2)$ by taking

$$\begin{aligned} \alpha(f) &= \alpha_2 \circ f \circ \gamma_1 \\ \gamma(g) &= \gamma_2 \circ g \circ \alpha_1 \end{aligned}$$





Induced Operations

Suppose that we have Galois connections $(L_i, \alpha_i, \gamma_i, M_i)$ such that each M_i is a more approximate version of L_i (for $i = 1, 2$). To improve an existing (analysis) $f_p : L_1 \to L_2$ with a new and more approximate analysis $g_p : M_1 \to M_2$ take

$$\alpha_2 \circ f_p \circ \gamma_1 \text{ is a candidate for } g_p$$

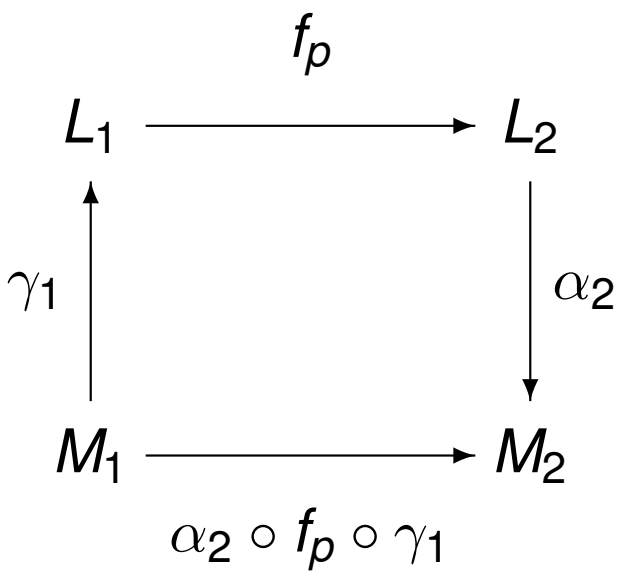
Then $\alpha_2 \circ f_p \circ \gamma_1$ is said to be induced by f_p and the two Galois connections.

Induced Operations

Suppose that we have Galois connections $(L_i, \alpha_i, \gamma_i, M_i)$ such that each M_i is a more approximate version of L_i (for $i = 1, 2$). To improve an existing (analysis) $f_p : L_1 \to L_2$ with a new and more approximate analysis $g_p : M_1 \to M_2$ take

$$\alpha_2 \circ f_p \circ \gamma_1 \text{ is a candidate for } g_p$$

Then $\alpha_2 \circ f_p \circ \gamma_1$ is said to be induced by f_p and the two Galois connections.



Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.

It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.

Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.
It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.

Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.
It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.

Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.
It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.

Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.
It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.

Galois Connections and Widenings

Suppose that we have a Galois connection (L, α, γ, M) between complete lattices L and M.
It is often possible to consider a widening operator on M, i.e. $\nabla_M : M \times M \to M$, and use it to define $\nabla_L : L \times L \to L$ by:

$$l_1 \nabla_L l_2 = \gamma(\alpha(l_1) \nabla_M \alpha(l_2))$$

Let ∇_M be an upper bound operator on M then ∇_L defines an upper bound operator on L. It defines a widening operator if one of the following two conditions are fulfilled:

(i) M satisfies the Ascending Chain Condition, or

(ii) (L, α, γ, M) is a Galois insertion and $\nabla_M : M \times M \to M$ is a widening operator.