Inductive Definitions, Semantics and Abstract Interpretation

Patrick Cousot  
LIENS, École Normale Supérieure  
45, rue d'Ulm  
75230 Paris cedex 05 (France)

cousot@dmi.ens.fr

Radhia Cousot  
LIX, École Polytechnique  
91128 Palaiseau cedex (France)

cousot@polytechnique.fr

Abstract

We introduce and illustrate a specification method combining rule-based inductive definitions, well-founded induction principles, fixed-point theory and abstract interpretation for general use in computer science. Finite as well as infinite objects can be specified, always at various levels of details related by abstraction. General proof principles are applicable to prove properties of the specified objects.

The specification method is illustrated by introducing G\(\bowtie\)SOS, a structured operational semantics generalizing Plotkin's [28] structured operational semantics (SOS) so as to describe the finite, as well as the infinite behaviors of programs in a uniform way and by constructively deriving inductive presentations of the other (relational, denotational, predicate transformers, ...) semantics from G\(\bowtie\)SOS by abstract interpretation.

1 Inductive definitions

Inductive definitions which are widely used in mathematical logic to define sets inductively generated by closure conditions, have popularized in computer science over the past few years. Classical or positive inductive definitions, co-inductive, kernel or negative definitions [1] as well as bi-inductive definitions mixing these two kinds of definitions are first captured by a general definition of the notion of rule-based inductive definition which is then generalized to systems of iterated inductive definitions so as to define inductively cartesian products of sets of finite and infinite objects indexed by a well-founded ordering. Examples show that many concepts related to programming come out of such inductive definitions.

1.1 Rule-based inductive definitions

Example 1 (Even numbers) Let the universe \(U\) be the set \(\mathcal{N}\) of natural numbers. The subset \(2\mathcal{N}\) of even numbers can be defined by the following axiom and rule schema:

\[
0 \in \mathcal{N}, \quad \frac{n \in \mathcal{N}}{n + 2 \in \mathcal{N}}
\]

(1)

Instead of considering a formal language for writing such rules and axioms schemata, we reason upon a set \(\Phi\) of rules instances where axioms have empty premises:

\[
\Phi = \{\emptyset\} \cup \left\{ \frac{n}{n + 2} \mid n \in \mathcal{N} \right\}
\]

(2)

The induced operator is:

\[
\bar{\Phi}(X) = \{0\} \cup \{n + 2 \mid \{n\} \subseteq X\}
\]

(3)

Let the basis be \(\bot = \emptyset\) and the join be \(\cup\) which induces the ordering \(\subseteq\). The \(\subseteq\)-smallest set \(\{\Phi\}\) greater than \(\bot\) which satisfies the rules \(\Phi\) is the least fixedpoint of \(\bar{\Phi}\) that is \(2\mathcal{N} = \bar{\Phi}^w = \cup_{n \in \mathcal{N}} \bar{\Phi}^n\) where \(\bar{\Phi}^0 = \bot = \emptyset\) and \(\bar{\Phi}^{n+1} = \bar{\Phi}(\bar{\Phi}^n) = \{0, 2, \ldots, 2n\}\). The closure ordinal of \(\Phi\) is the first infinite limit ordinal \(\omega\).

We now give a general set-theoretic definition of such inductive definitions. We write \(\varphi(S)\) for the powerset of the set \(S\) and \(S \leftrightarrow S'\) (respectively \(S \hookrightarrow S'\)) for the set of total (respectively partial) maps from \(S\) into the set \(S'\).

Definition 2 (Inductive definition) An inductive definition \(\mathcal{D}\) is a quadruple \((U, \Phi, \bot, \cup)\) such that:

- The universe \(U\) is a set,
- \(\Phi\) is a set of rules instances \(P \ni c\) where \(P \subseteq U\) and \(c \in U\),
- \(\bot \subseteq U\) is the basis,
- \(\cup \in \varphi(\varphi(U)) \hookrightarrow \varphi(U)\) is the join and the induced ordering is \(\ni \subseteq \varphi(U)\) is a partial order on \(\varphi(U)\).

\(\Box\)

Definition 3 (Operator induced by an inductive definition) The operator \(\bar{\Phi}\) induced by \(\mathcal{D} = (U, \Phi, \bot, \cup)\) is \(\bar{\Phi} \in \varphi(U) \hookrightarrow \varphi(U)\) such that \(^1\):

\[
\bar{\Phi}(X) = \{c \in U \mid \exists P \subseteq X : \frac{c}{c} \in \Phi\}
\]

(4)

\(\Box\)

\(^1\) More generality can be obtained when replacing \(\subseteq\) by an arbitrary relation.
Definition 1 (Iteration of an operator) The iteration of \( F \in S \mapsto S \) for \( U \subseteq \psi(S) \mapsto S \) is the partial operator \( F^\omega_u \in S \mapsto S \) such that \( F^\omega_u(X) = F^\omega \) where:

- \( F^0 = X \).
- \( F^\lambda = F(\bigcup_{\alpha < \lambda} F^\alpha) \) if \( 0 < \lambda \in \text{Ord} \).
- \( F^\lambda = F^\omega \) if \( \lambda > \omega \).

\( \text{cl}_\omega^u(F) \) is called the closure ordinal of \( F \) for \( X \).

A complete partial order \( (\mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top) \) (cpo for short) is a set partially ordered by \( \sqsubseteq \) such that all increasing chains \( C \) have least upper bounds \( \sqcap C \) and \( \bot \in L \) (\( \bot \) is usually the infimum of \( L \)). It is a complete lattice if and only if all subsets \( X \) have least upper bounds \( \sqcap X \) and therefore greatest lower bounds \( \sqcup X \) (in which case we usually write \( \sqcap \sqcup X \) and \( T = \sqcup L \)). \( F \in L \mapsto L \) is monotonic (written \( F \in L \in \mathbb{L} \to L \)) if and only if for all \( x, y \in \sqcup X \), \( x \leq y \Rightarrow F(x) \leq F(y) \). It is extensive (written \( F \in L \in \mathbb{L} \to L \)) if and only if for all \( x, y \in \sqcup X \), \( x \leq y \Rightarrow F(x) \subseteq F(y) \). It is a complete \( \sqcup \)-morphism (written \( F \in L \in \mathbb{L} \to L \)) if for all families of sets \( X_t \subseteq L \) such that \( \sqcup X_t \) exists in \( L \), we have \( \sqcup F(X_t) = F(\sqcup X_t) \).

The propositions below immediately follow from the constructive proof of Tarski's fixpoint theorem [9]:

Proposition 1 If \( S \subseteq \mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \) is a cpo and \( F \in S \mapsto S \) is monotonic or extensive then \( F^{\infty}_u \in S \mapsto S \) where \( \text{pm}(F) = \{ x \in S \mid x \subseteq F(x) \} \) (so that \( F^{\infty}_u \) is totally defined for pre-fixpoints of \( F \)).

Definition 6 (Well-formed inductive definitions) The inductive definition \( \exists \equiv (\mathbb{U}, \Phi, \sqsubseteq, \sqcup, \top) \) is well-formed if and only if \( \text{cl}_\omega^u(\sqcup) \) exists in which case it is the set \( \text{cl}_\omega^u(\sqcup) \) inductively defined by \( \exists \).

Definition 7 (Inductive definitions on a cpo or a complete lattice) Let \( \exists \equiv (\mathbb{U}, \Phi, \sqsubseteq, \sqcup, \top) \) be an inductive definition. \( \exists \) is on a cpo (respectively a complete lattice) if and only if \( \psi(U) \subseteq \mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \) is a cpo (respectively \( \psi(U) \subseteq \mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \) is a complete lattice).

Definition 8 (Monotonic and extensive inductive definitions) The inductive definition \( \exists \equiv (\mathbb{U}, \Phi, \sqsubseteq, \sqcup, \sqcap, \bot, \top) \) is monotonic (respectively extensive) whenever \( \Phi \) is monotonic and \( \top \subseteq \Phi(\top) \) (respectively \( \Phi \) is extensive).

Corollary 9 A monotonic or extensive inductive definition on a cpo is well-formed.

When it exists, we write \( \text{lfp}_u^\omega F \) for the least fixpoint of \( F \) greater than or equal to \( x : F(F\circ \text{lfp}_u^\omega F) = F\circ \text{lfp}_u^\omega F \), \( x \leq \text{lfp}_u^\omega F \) and \( \forall y : [F(y) = y \land x \leq y] \Rightarrow [y \leq \text{lfp}_u^\omega F \leq y] \). The same way, \( \text{gfp}_u^\omega F \) is the greatest fixpoint of \( F \) less than or equal to \( x \).

Proposition 10 If \( S \subseteq \mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \) is a cpo and \( F \in S \mapsto S \) then \( \text{lfp}_u^\omega S \in S \mapsto S \) if and only if \( \forall S \subseteq \mathbb{L}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \).
The set $T^{<\omega}$ of maximal finite execution traces of this transition system $T$ is defined by the following schema (where free variables $a \in Act$, $s \in S$ and $\sigma \in \Sigma^<$ are universally quantified):

$$
\frac{s \triangleright \sigma}{s \in T^{<\omega}}, \quad \frac{s \triangleright \sigma \land \sigma \in T^{<\omega}}{s \triangleright \sigma \in T^{<\omega}}
$$

which stands for the set $\Phi$ of rule instances:

$$
\frac{t \triangleright \sigma}{s \in \Sigma^<}, \quad \frac{s \triangleright \sigma \land \sigma \in \Sigma^<}{s \triangleright \sigma \in \Sigma^<}
$$

for all $s \in S$ such that $s \triangleright \sigma$ and $s' \in S$, $\alpha \in Act$ and $\sigma \in \Sigma^<$ such that $s' \triangleright \sigma$. The rules are decorated by the $+$ sign to indicate that $\bot = \bot$ and $\uparrow = \uparrow$ followed by the universe $U = \Sigma^{<\omega}$. One or both indications can be left out when clear from the context.

The monotone operator $\overline{T} \in \rho(\Sigma^{<\omega})$ is defined by this inductive definition:

$$
\overline{T}(X) = \{ s \triangleright \sigma \mid s \triangleright \sigma \subseteq \Sigma^<, \sigma \in X \}
$$

Let us define $T^{n+1}_a = \overline{T}^n$ and the set $T^{n+1}_a \defeq \{ \sigma \in \Sigma^{<\omega} \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \}$ of execution traces of $T$ of length $n > 0$. Observe, by induction on $n$, that the iterates of the operator $\overline{T}$ are $\overline{\overline{T}} = U^{n+1}_a T^{n+1}_a$, so that $\overline{\overline{T}} = \overline{\overline{T}}$ for all $n \geq 0$. Moreover $\overline{T}^{n+1}_a = \overline{T}$ so that the closure ordinal of $\overline{T}$ is $\omega$. Finally, we have defined $\overline{T} = \overline{T}_a \overline{T} = T^{<\omega} = \{ \sigma \in \Sigma^{<\omega} \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \land |\sigma| = 0 \}$. $\square$

### 1.3 Negative or co-inductive definitions

**Definition 18 (Negative co-inductive definition)** A negative co-inductive definition $\exists = \{ \mathcal{U}, \Phi, \Pi \}$ is $\exists^- = \{ \mathcal{U}, \Phi, \Pi, n \}$. $\square$

**Proposition 19** A negative inductive definition $\exists^+ = \{ \mathcal{U}, \Phi, \Pi \}$ is well-formed and $\{ \exists^+ \} = \overline{\overline{T}}_a(\mathcal{U})$ is the greatest (for $\subseteq$) dense set, i.e. satisfying the rules: dense \_defeq sat_{\exists}(S) = \{ \forall \; \mathcal{E} \in S : \exists \mathcal{P} \subseteq S : \ell \leq \mathcal{E} \}. $\square$

The definition of observation equivalence in [21] and of static typing in [22] are examples of negative co-inductive definitions.

---

Example 20 (Infinite execution traces) The set $T^{<\omega}$ of infinite execution traces of a transition system $T = (S, Act, \{ \triangleright \sigma \mid \sigma \in \Sigma^< \})$ is defined by the following schema (where free variables $a \in Act$, $s' \in S$ and $\sigma \in \Sigma^<$ are universally quantified):

$$
\frac{s \triangleright \sigma \land \sigma \in T^{<\omega}}{s \triangleright \sigma \in T^{<\omega}}
$$

which stands for the set $\Phi$ of rule instances:

$$
\frac{\{ \sigma \}}{s \triangleright \sigma \subseteq \Sigma^<}
$$

for all $s \in S$, $\alpha \in Act$ and $\sigma \in \Sigma^<$ such that $s \triangleright \sigma$. The monotone operator $\overline{T} \in \rho(\Sigma^<)$ is defined by this inductive definition:

$$
\overline{T}(X) = \{ s \triangleright \sigma \mid s \triangleright \sigma \subseteq \Sigma^<, \sigma \in X \}
$$

Let us define $T^{\infty}_a \defeq \Sigma^<$ and the set $T^{\infty}_a \defeq \{ \sigma \in \Sigma^< \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \}$ of infinite traces starting with $n \geq 1$ transitions. Observe, by induction on $n$, that the iterates of the operator $\overline{T}$ are $\overline{T}^n = \overline{T}^n$ for all $n \geq 0$, so that $\overline{T}^n = \overline{T}$ for all $n \geq 0$. Moreover $\overline{T}^{n+1}_a = \overline{T}$ so that $\overline{T}$ is defined. Finally, we have defined $\overline{T} = \overline{T}_a \overline{T} = T^{<\omega} = \{ \sigma \in \Sigma^< \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \land |\sigma| = 0 \}$. $\square$

Example 21 (Maximal execution traces) The set $T^{\omega} \subseteq T^{<\omega} \cup T^\omega$ of execution traces of a transition system $T = (S, Act, \{ \triangleright \sigma \mid \sigma \in \Sigma^< \})$ is defined by the following schema (where free variables $a \in Act$, $s' \in S$ and $\sigma \in \Sigma^<$ are universally quantified):

$$
\frac{s \triangleright \sigma \land \sigma \in T^{\omega}}{s \triangleright \sigma \in T^{\omega}}
$$

which stands for the set $\Phi$ of rule instances:

$$
\frac{t \triangleright \sigma}{s \triangleright \sigma \subseteq \Sigma^<}, \quad \frac{s \triangleright \sigma \land \sigma \in \Sigma^<}{s \triangleright \sigma \in \Sigma^<}
$$

for all $s \in S$, $\alpha \in Act$ and $\sigma \in \Sigma^<$ such that $s \triangleright \sigma$. The monotone operator $\overline{T} \in \rho(\Sigma^<)$ is defined by this inductive definition:

$$
\overline{T}(X) = \{ s \triangleright \sigma \mid s \triangleright \sigma \subseteq \Sigma^<, \sigma \in X \}
$$

Let us define $T^{\infty}_a \defeq \Sigma^<$ and the set $T^{\infty}_a \defeq \{ \sigma \in \Sigma^< \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \}$ of infinite traces starting with $n \geq 1$ transitions. Observe, by induction on $n$, that the iterates of the operator $\overline{T}$ are $\overline{T}^n = \overline{T}^n$ for all $n \geq 0$, so that $\overline{T}^n = \overline{T}$ for all $n \geq 0$. Moreover $\overline{T}^{n+1}_a = \overline{T}$ so that $\overline{T}$ is defined. Finally, we have defined $\overline{T} = \overline{T}_a \overline{T} = T^{<\omega} = \{ \sigma \in \Sigma^< \mid \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \land |\sigma| = 0 \}$. $\square$

Define $T^{\infty} \defeq \{ \sigma \in \Sigma^< \mid |\sigma| = m > 0 \land \forall \; \sigma' \leq \sigma \Rightarrow s' \triangleright \sigma \leq \sigma \}$ to be the set of finite traces of length $m$ starting with $n$ transitions. Observe, by induction on $n$, and using the notations of examples 17 and 20 that the iterates of the operator $\overline{T}$ are $\overline{T}^{n+1}_a = \overline{T}^{n+1}_a$ for all $m \geq n$. Moreover $\overline{T}^{n+1}_a$ stands for $\overline{T}$ for all $n \geq 0$. Recursively $\sigma \in \overline{T}^{n+1}_a$ has either $|\sigma| \leq n$ and $\exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$ or $|\sigma| > n$ and $\exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$. Also $|\sigma| \leq n$ implies $\sigma \in \overline{T}^{n+1}_a$ for all $n \geq 0$. Recursively $\sigma \in \overline{T}^{n+1}_a$ has either $|\sigma| = 0 \land \exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$ or $|\sigma| = \ell \land \exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$ and $\sigma \in \overline{T}^{n+1}_a$. Also $|\sigma| \leq n$ implies $\sigma \in \overline{T}^{n+1}_a$ for all $n \geq 0$. Recursively $\sigma \in \overline{T}^{n+1}_a$ has either $|\sigma| \leq n$ and $\exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$ or $|\sigma| > n$ and $\exists \mathcal{E} \in S : \exists \mathcal{P} \subseteq \Sigma^<, \ell \leq \mathcal{E}$.
the closure ordinal of $\mathbb{F}^{-}$ is $\omega$. Finally, we have defined $\mathbb{F}^{-} = \operatorname{lfp}_{\leq \omega} \mathbb{F}^{-} = \mathbb{T}^{\omega}$. However this definition is not quite satisfactory because of the term $\bigcup_{n \geq 0} T^{nm}$ in $\mathbb{F}^{-}$, which disappears when passing to the limit $\mathbb{F}^{-}$ (since $\bigcup_{n \geq 0} T^{nm} = \emptyset$). A definition of $T^{\omega}$ such that $\mathbb{F}^{-}$ is a conservative extension of the $\mathbb{F}^{-}$ is proposed in example 28.

If $F \in B \mapsto B$ is an operator on a complete boolean lattice $B = \{\bot, \top, \lor, \land, \neg\}$, its dual $\bar{F} \in B \mapsto B$ is defined by $\forall x \in B : \bar{F}(x) = \neg F(\neg x)$. If $F$ is monotonic, so is $\bar{F}$. By duality, a negative inductive definition can always be given an indirect equivalent positive definition. However, as shown by the above examples, most of the time, using a direct co-inductive definition is much more clear.

**Proposition 22 (Duality of positive and negative inductive definitions)** If $\mathbb{F} = \{U, \Phi\}$ is an inductive definition then $\operatorname{sat}_{\Phi}(S) = \operatorname{sat}_{\Phi}(\neg S), [\Phi] = \operatorname{lfp}_{\leq \omega} \Phi^{-} = -\operatorname{lfp}_{\geq \omega} \Phi^{-}$ and $[\Phi] = -[\Phi]^{-}$. □

### 1.4 Bi-inductive definitions

Assume the universe $U$ is covered by two subsets $U^{+}$ and $U^{-}$, that is $U = U^{+} \cup U^{-}$. For example $U^{+}$ may be the set of finite objects of $U$ while $U^{-}$ is the set of infinite objects of $U$. To define a subset $[S]$ of $U$, we could define separately the finite objects $[S] \cap U^{+}$ using a positive inductive definition and the infinite objects $[S] \cap U^{-}$ using a negative inductive definition. In practice, this must often be done simultaneously. For example in denotational semantics the terminating and non-terminating behaviors of programs are defined at the same time using a single fixpoint operator. Since, for clarity, we insist upon using rule-based inductive definitions (preferred to fixpoint definitions [5]), we propose a method for combining inductive and co-inductive definitions which allows for the simultaneous use of positive and negative axioms and induction rules. We then give equivalent (but maybe less intuitive) fixpoint characterizations.

**Definition 23 (Bi-inductive definition)** A bi-inductive definition $\mathbb{F} = \{U, \pi^{+}, \pi^{-}, \Phi\}$, where $\pi^{+}, \pi^{-} \in \mathfrak{p}(U) \mapsto \mathfrak{p}(U)$ is $\mathbb{F}^+ = \{U^+ / \equiv \} \cup \Phi^+ \cup \bot^+ \cup \bot^+$, with, for all $S, T, S_i \subseteq U$:

- $U^+ \overset{\text{def}}{=} \pi^{+}(U) \cup \pi^{-}(U)$
- $S \equiv_{T} \overset{\text{def}}{=} \{ \pi^{+}(S) = \pi^{+}(T) \} \lor \{ \pi^{-}(S) = \pi^{-}(T) \}$
- $\Phi^+ \overset{\text{def}}{=} \{ \pi^{+}(P) \mid \exists \phi \in \Phi \land \phi \in \pi^{+}([c]) \}$
- $\Phi^- \overset{\text{def}}{=} \{ \pi^{-}(P) \mid \exists \phi \in \Phi \land \phi \in \pi^{-}([c]) \}$
- $\Phi^+ \overset{\text{def}}{=} \Phi^+ \cup \Phi^-$
- $\bot^+ \overset{\text{def}}{=} \pi^{+}(U)$ and $\bot^-$
- $\bigcup_i S_i \overset{\text{def}}{=} \bigcup_i \pi^{+}(S_i) \cup \bigcap_i \pi^{-}(S_i)$

□

We often use bi-inductive definitions to simultaneously define subsets of the universe which are separated in the following sense:

**Definition 24 (Separated bi-inductive definition)** A bi-inductive definition $\mathbb{F} = \{U, \pi^{+}, \pi^{-}, \Phi\}$ is separated if and only if $\pi^{+}$ and $\pi^{-}$ are monotonic for $\subseteq$, idempotent and satisfy the separation property: $\forall P \subseteq \pi^{+}(U) : \forall M \subseteq \pi^{-}(U) : \pi^{+}(P \cup M) = \pi^{+}(P)$ and $\pi^{-}(P \cup M) = \pi^{-}(M)$.

**Proposition 25** Let $\mathbb{F} = \{U, \pi^{+}, \pi^{-}, \Phi\}$ be a separated bi-inductive definition. Let us define, for all $S, T, S_i \subseteq U$:

- $S \equiv^+ T \overset{\text{def}}{=} \{ \pi^{+}(S) \subseteq \pi^{+}(T) \} \land \{ \pi^{-}(S) \supseteq \pi^{-}(T) \}$
- $\mathbb{T}^+ \overset{\text{def}}{=} \pi^{+}(U)$ and $\mathbb{T}^- \overset{\text{def}}{=} \pi^{-}(U)$
- $\bigcap S_i \overset{\text{def}}{=} \bigcap_i \pi^{+}(S_i) \cup \bigcup_i \pi^{-}(S_i)$

Then:

1. $\forall P(U) / \equiv^+ (\mathbb{T}^+, \mathbb{T}^-, \mathbb{T}^+, \mathbb{T}^-)$ is a complete lattice.
2. $\mathbb{F}^+$ is monotonic for $\subseteq^+$.
3. $\mathbb{F}^+$ is well-formed.

□

Disjointed bi-inductive definitions are used to simultaneously define subsets of disjointed universes:

**Definition 26 (Disjointed bi-inductive definition)** A disjointed bi-inductive definition $\mathbb{F} = \{U^+, U^-, \Phi\}$ where $U^+ \cap U^- = \emptyset$ is $\mathbb{F}^+ = \{U^+ \cup U^-, \pi^+, \pi^-\}$ with $\pi^+ \overset{\text{def}}{=} \pi^{+}(X) = X \cap U^+$ and $\pi^- \overset{\text{def}}{=} \pi^{-}(X) = X \cap U^-$. □

**Proposition 27** A disjointed bi-inductive definition $\mathbb{F}^+ = \{U^+, U^-, \Phi\}$ is separated, hence well-formed and such that $\equiv^+$ is equality and $[\Phi^+] = \operatorname{lfp}_{\leq \omega} \Phi^+ = -\operatorname{lfp}_{\geq \omega} \Phi^+$. □

**Example 28 (Maximal execution traces II)** The set $T^{\omega} \overset{\text{def}}{=} T^{\omega} \cup T^{w}$ of execution traces of a transition system $T = (S, Act, \{T^{\omega} / a \in Act\}$, init) can be defined by a disjointed bi-inductive definition $\mathbb{F}^+ = \{U^+, U^-, \pi^+, \pi^-, \Phi\}$ where $U^+ = \Sigma^{\omega}$ is the set of finite traces, $U^- = \Sigma^{\omega}$ is the set of infinite traces over $S$ and $Act$ and $\Phi$ is specified by the following rule schemata:

$$
\begin{align*}
\frac{s \in T^{\omega}}{s \in T^{\omega}} + \\
\frac{s \in T^{\omega}}{s \in T^{\omega}} + \\
\end{align*}
$$

(14)

$$
\begin{align*}
\frac{s \in T^{\omega}}{s \in T^{\omega}} - \\
\frac{s \in T^{\omega}}{s \in T^{\omega}} - \\
\end{align*}
$$

(15)

For simplicity we write $\pi^{+}(T^{\omega})$ instead of $\pi^{+}(T^{\omega})$ (that is $\pi^{+} \in T^{\omega}$) in positive rule schemata and the same way $T^{\omega}$ stands for $\pi^{-}(T^{\omega})$ that is $T^{\omega}$
in the negative ones. For short, the rule schemata (14) and (15) can be written as follows:
\[
\frac{s \mapsto \sigma}{s \in T \Rightarrow \sigma} + \frac{s \mapsto \sigma \land \sigma \in T}{s \in T \Rightarrow \sigma}
\] (16)

The monotone operator $\overline{T^*} \in \rho(\Sigma^\omega)$ induced by this inductive definition is:
\[
\overline{T^*}(X) = \begin{cases} 
    \{ s \in S \mid s \mapsto \} \cup \\
    \{ s \mapsto \sigma \cup s \mapsto \sigma \land \sigma \in X \} 
\end{cases}
\] (17)

Observe, by induction on $n$, and using the notations of examples 17 and 20 that the iterates of the operator $\overline{T^*}$ are $\overline{T^{n+1}} = (U_{\omega+1}^{n+1} \cap (R_{\omega+1}^{n+1} \cap T^\omega))$ so that $\overline{T^\omega} = (U_{\omega+2}^{\omega+2} \cap (R_{\omega+2}^{\omega+2} \cap T^\omega)) = T^\omega \cup T^\omega = T^\omega$. Moreover $\overline{T^\omega} = \overline{T^\omega}$ so that the closure ordinal of $\overline{T^*}$ is $\omega$. Finally, we have defined $\overline{T^*} = \text{lfp}_{\omega}^\exists \overline{T^*} = T^\omega$.

Our approach for defining program behaviors, and more generally a subset of a space with finite and infinite computable objects, is characterized by the following remarks:
1. The finite objects are constructed from their finite components using positive rules;
2. The infinite objects are not obtained as limits of finite ones (this may be done once for all in the domain theoretical definition of the universe $U$) but selected among all the possible ones by successive inspections of finite parts using negative rules;
3. By combining the two methods, one can define other fixpoints, in addition to the usual least (for $\omega$) and greatest fixpoints.

The usefulness of such non-extremal fixpoints is illustrated by the example below:

**Example 29 (Infinitary languages)** Let $A = \{a, b, c\}$ be an alphabet, $U = U^+ \cup U^-$ where $U^+ = A^*$ and $U^- = A^0$ are respectively the sets of finite and infinite words written on $A$. The infinitary language $L = (a \cup b)^*c \cup b^\omega$ is defined by the following axioms and rule schemata (which are disjoined since $U^+ \cap U^- = \emptyset$):
\[
\begin{align*}
    c & \in L^+ \\
    \sigma & \in L \\
    a\sigma & \in L \\
    b\sigma & \in L \\
    a & \in L^- \\
    b & \in L^- \\
    c & \in L^+ \\
    a & \in L^- \\
    b & \in L^- \\
\end{align*}
\] (18)

The operator associated with the instances of rules schemata (18) is $\overline{T} = \{c\} \cup \{a\sigma \mid \sigma \in X \cap A^*\} \cup \{b\sigma \mid \sigma \in X\}$. It is a monotone operator on $\rho(\Sigma^\omega)$ for $\leq$, $\leq^*$, $\leq^+$, $\leq^-$, $\leq^\omega$, $\leq^\omega$, and $\text{lfp}_{\omega}^\exists \overline{T} = (a \cup b)^*c \cup b^\omega = L$.

Observe however that $L$ is neither the least fixpoint $(a \cup b)^c$ nor the greatest fixpoint $(a \cup b)^c \cup (a \cup b)^a$ of the context-free grammar $X ::= c \mid aX \mid bX$ (these fixpoints being extremal for $\leq$).

### 1.5 Proof methods

It is interesting to notice that numerous mathematical tools come of such inductive definitions. In particular, methods for proving properties of fixpoints (such as Park's fixpoint induction) can be transcribed to prove properties of inductively defined sets.

**Proposition 30 (Fixpoint induction)** Let $L(\Sigma, \Gamma, \Delta)$ be a cpo and $F \in L(\Sigma, \Gamma, \Delta)$. Then $[\text{lfp}^\exists_F \subset \Gamma] \Rightarrow [\Sigma \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \exists L \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \exists \sigma \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \sigma \in \text{lfp}^\exists_F \subset \Gamma]$. If $\Gamma$ is well-defined then $[\Sigma \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \exists L \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \exists \sigma \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \sigma \in \text{lfp}^\exists_F \subset \Gamma]$. □

**Corollary 31** If $G \equiv (\Sigma, \Phi, \Gamma, \Delta)$ is a monotonic inductive definition on a cpo then $[\Sigma \in L(\Sigma, \Gamma, \Delta) \\
\Rightarrow \text{lfp}^\exists_F \subset \Gamma]$ is a strict well-founded partial ordering so that transfinite induction can be used to prove the inverse inequality:

**Definition 32 (Well-foundedness)** A relation $\leq$ on a class $W$ is well-founded if and only if $\text{Wellfounded}(W, \leq) \equiv \forall F \in W : [\emptyset \neq \emptyset \Rightarrow \exists y \in E : (-\exists z \in E : z < y)]$ holds. □

**Proposition 33 (Iteration induction)** Let $L(\Sigma, \Gamma, \Delta)$ be a cpo with supremum $\top = \emptyset$, $F \in L(\Sigma, \Gamma, \Delta)$ and $Q \in L$. We have: $[Q \equiv \text{lfp}F] \Rightarrow [\exists W \subseteq \Sigma : \forall x \in W \\
\Rightarrow \forall x \in W : [\forall x \in W : l(x) \in F(\cup \{x \mid x \in W \})] \\
\Rightarrow Q \subseteq \bigcup_{x \in W} l(x)]$. □

Another useful method for proving properties of fixpoints is computational induction. We let $\text{Ord}$ be the class of ordinals and $\text{Limit} = \{a \in \text{Ord} : \text{Ord} \ni \sigma = a \neq 0\}$ be the class of limit ordinals.

**Proposition 34 (Computational induction)** Let $L(\Sigma, \Gamma, \Delta)$ be a cpo, $F \in L(\Sigma, \Gamma, \Delta)$ and $P \subseteq \rho(\Sigma)$.

We have: $[\text{lfp}^\exists_F F \in P] \Leftrightarrow [\exists I \in \text{Ord} \Rightarrow I : [\exists l \in I] \\
\Rightarrow \exists X \in \text{Ord} \cap \text{Ord} \ni F(X) \in I(\alpha) \\
\Rightarrow [\forall \alpha \in \text{Ord} : [X(\beta) \ni F(X) \in I(\alpha) \\
\Rightarrow [\forall \alpha \in \text{Ord} : [X(\alpha) \ni F(X) \in I(\alpha)]]] \Rightarrow [\bigcup_{\alpha \in \text{Ord} X(\alpha) \in I(\alpha)]] \\
\Rightarrow [\bigcup_{\alpha \in \text{Ord} X(\alpha) \in I(\alpha)]]]$. □

An interesting particular case (which amounts to Scott induction when the function is upper continuous and the property is admissible) consists in choosing the invariant $I(\alpha)$ as $P$, but this is not a semantically complete proof method.

**Proposition 35 (Stepwise computational induction)** Let $L(\Sigma, \Gamma, \Delta)$ be a cpo, $F \in L(\Sigma, \Gamma, \Delta)$ and $P \subseteq \rho(\Sigma)$.

We have: $[\exists I \in \text{Ord} \Rightarrow [\forall Y \in P : F(Y) \ni X \in P] \\
\Rightarrow [\forall \alpha \in \text{Ord} : [X(\alpha) \ni F(X) \in P] \\
\Rightarrow [\forall X \in \text{Ord} \ni F(X) \in P]]]$. □
A last example is the inductive definitions of functions \( f \in [\mathbb{I}] \mapsto D \).

**Proposition 36 (Inductive function definition)** Let \( \mathcal{G} = \langle U, \Phi, \bot, \cup \rangle \) be a well-formed inductive definition, \( D \) be a set and \( \gamma_i \in \wp(P_i \times D) \mapsto D \) for all \( P_i \in \Phi \). There exists a unique total function \( f : f(c_i) = \gamma_i(\langle x, f(x) \rangle \mid x \in P_i) \).

When specialized to positive inductive definitions, propositions (30) and (33) amount to (for short, similar corollaries holding for the other varieties of inductive definitions are not stated):

**Corollary 37 (Fixpoint induction for positive inductive definitions)** Let \( \mathcal{G}^+ = \langle U, \Phi^+ \rangle \) be a positive inductive definition and \( R \subseteq U \). Then \([\mathcal{G}^+] \cap Q \subseteq R \implies [\mathcal{G} \in U \cup Q \mapsto \{\mathcal{T}, \mathcal{F}\}] : (\forall i \in \Phi : [\forall x \in P : I(x) \Rightarrow I(c) \wedge (\forall x \in Q : I(x) \Rightarrow x \in R)] \).

**Corollary 38 (Iteration induction for positive inductive definitions)** Let \( \mathcal{G}^+ = \langle U, \Phi^+ \rangle \) be a positive inductive definition and \( Q \subseteq U \). We have: \([Q \subseteq [\mathcal{G}^+] \implies [\exists W : \exists \prec \subseteq W \times W : \text{Well-founded}(W, \prec) \wedge \exists \mathcal{W} \subseteq W \mapsto \mathcal{W}(U) : [\forall x \in W : x \in I(x)] \cap [\forall x \in W : \forall c \in I(x) : \exists s^w \subseteq \Phi : \forall c \in P : \exists s^w < x : c \in I(s^w)]] \).

**Example 39 (Well-founded part of a relation)** Let \( S \) be a set of states and \( \mathcal{T} \subseteq S \times S \) be a relation on \( S \). We write \( s \prec s' \) for \((s, s') \in \mathcal{T} \). The well-founded part \( Wf(t) \) of \( t \) is the set of \( \sigma_0 \in S \) such that there is no infinite sequence \( \sigma_0 \prec \sigma_1 \prec \ldots \).

\[
Wf(t) = \{s \in S \mid \neg(\exists s_0 \in \omega \mapsto S : s = \sigma_0 \wedge (\forall i \in \omega : \sigma_i \prec \sigma_{i+1})) \}
\]

The well-founded part \( Wf(t) \) of \( t \) is specified by the inductive definition \( \mathcal{G}^+ = \langle S, \Phi^+ \rangle \) where \( \Phi \) consists of the rules instances [1]:

\[
\{s' \in S \mid s \prec s' \} (s) \quad (20)
\]

for all \( s \in S \).

- To show that \([\mathcal{G}^+] \subseteq Wf(t)\), we use proposition 37 with \( I(x) = \{x \in Wf(t)\} \) so that \((\forall x \in S : I(x) \Rightarrow x \in Wf(t))\) is obvious. We must also show that \( \forall s \in S : [\forall s' \in S : s \prec s' \Rightarrow I(s')] \Rightarrow I(s) \) which holds since \( [\forall s \in \omega \mapsto S : s = \sigma_0 \wedge (\forall i \in \omega : \sigma_i \prec \sigma_{i+1})] \) implies \( [\exists s' \in S : s \prec s' \wedge (\exists \sigma_0 \in \omega \mapsto S : s = \sigma_0 \wedge (\forall i \in \omega : \sigma_i \prec \sigma_{i+1})] = [\exists s \in S : s \prec s' \wedge \neg I]'s \)

by choosing \( s' = \sigma_0 \) and \( \sigma_i = \sigma_{i+1} \) for all \( i \in \omega \).

- To show that \( Wf(t) \subseteq [\mathcal{G}^+] \), we use proposition 38 with \( W = Wf(t) \) and \( s \prec s \Rightarrow s = s' \) so that obviously \( \text{WellFounded}(W, \prec) \) holds and \( I(x) = \{x\} \) so that \([\forall u \in Wf(t) : \exists x \in W : x \in I(u)] \) is true when choosing \( x = u \). Moreover \( s \prec s' \) implies \( s' \prec s \) so that \( [\forall x \in W : \forall c \in I(x) : \exists s^w \subseteq \Phi : \forall c \in P : \exists s^w < x : c \in I(s^w)] \).

**Example 40 (Floyd's partial correctness and termination proof methods)** Let \( T = \langle S, \mathcal{A}, \{\omega \mid \omega \in \mathcal{A}, \text{Init}\} \rangle \) be a labelled transition system specified by a program \( P \). Let \( \Psi \in S \times S \mapsto \{tt, ff\} \) be an input-output specification of program \( P \). \( P \) is partially correct with respect to \( \Psi \) if and only if \( \forall s, s' \in S : s \leftrightarrow s' \Rightarrow \Psi(s, s') \) (where \( \leftrightarrow \) is the reflexive transitive closure of \( \Rightarrow \)). According to proposition 37, the partial correctness proof can be done by discovering an invariant \( I \subseteq U \times S \times S \) satisfying the following verification conditions (as given in [11]):

\[
\forall s \in S : (s, s) \in I 
\]

\[
\forall s, s', s'' \in S : \forall x \in \mathcal{A} : [(s, s') \in I \wedge s' \prec s'' \Rightarrow (s, s'') \in I] 
\]

\[
\forall s, s' \in S : [(s, s') \in I] \Rightarrow \Psi(s, s') 
\]

Let \( \varepsilon \in S \mapsto \{tt, ff\} \) be a specification of the initial states of program \( P \). Program \( P \) terminates if and only if there is no infinite execution trace \( \sigma_0 \prec \sigma_1 \prec \sigma_2 \ldots \) such that \( \varepsilon(\sigma_0) \). According to example 39, we must show that: \( \forall s \in S : \varepsilon(s) \Rightarrow s \in Wf(\varepsilon) \).

By proposition 38 applied to the rule-based inductive definition:

\[
\{s' \in S \mid s \prec s' \} (s) \forall s \in S 
\]

of \( Wf(\varepsilon) \), this can be done by finding a well-founded set \( W, \prec \) and an invariant \( I \in W \mapsto \wp(S) \) satisfying the following verification conditions (as given in [12]):

\[
\forall s \in W : \forall s' \in P : s \prec s' \Rightarrow [\exists \varepsilon \in \mathcal{A} : s \prec s' \Rightarrow \Psi(s, s') 
\]

\[
\forall s \in S : \varepsilon(s) \Rightarrow [\exists \varepsilon \in W : s \in P] 
\]

**1.6 Well-founded system of inductive definitions**

To get more expressive power, we introduce systems of inductive definitions in order to define subsets of a cartesian product \( \prod_{i \in \Delta} U[\bar{i}] \) of sets \( U[\bar{i}], i \in \Delta \).

We mix this notion with transfinite iterated definitions by induction on a well-founded set \( W, \prec \).

Such well-founded (also called iterated) inductive definitions were first introduced by G. Kreisel and then further developed by S. Feferman [15].

**Definition 41 (Well-founded system of inductive definitions)** A well-founded system of inductive definitions \( \mathcal{G} \) is a tuple \( \langle W, \prec, \Delta, U, \Phi, \bot, \cup \rangle \) such that:

- \( \prec \) is a well-founded binary relation on the set \( W \).

- The index \( \Delta \) is such that for all \( w \in W, \Delta[w] \) is a set.

- For all \( w \in W \) and all \( i \in \Delta[w] \), the universe \( U[\bar{i}] \) is a set.
• $\Phi$ is a set of rules instances $P^{\pi \bullet P}$ where $w \in W, i \in \Delta[w], P \subseteq \Pi_{i \in \Delta[w]} \Phi(U[I][w][i])$, $\pi, \nu \subseteq \{w', S\}$ s.t. $w' \approx-c w \land S \subseteq \Pi_{i \in \Delta[w]} \Phi(U[I][w][i])$, and $c \in U[I][w][i]$. 

For all $w \in W$, $\|w\| \in \Pi_{i \in \Delta[w]} \Phi(U[I][w][i])$ is the basis.

• For all $w \in W$, $v[I][w][i] \in \Pi_{i \in \Delta[w]} \Phi(U[I][w][i]) \Rightarrow \Pi_{i \in \Delta[w]} \Phi(U[I][w][i])$ is the join.

\[\square\]

Definition 42 (Operator induced by a well-founded system of inductive definitions) For each $w \in W$, the operator $\Phi[I][w][i]$ induced by $\Theta = (W, \rightarrow, \Delta, U, \Phi, \bot, \cup)$ is well-formed if and only if $\Phi[I][w][i]$ is defined by:

$$\Phi[I][w][i] = \Pi_{i \in \Delta[w]} \{ c \in U[I][w][i] | \exists P \subseteq X : \exists \pi \subseteq \{(w', S) | w' \approx-c w \land S \subseteq \|\Phi[I][w][i]\|) : \exists \nu \subseteq \{(w', S) | w' \approx-c w \land S \subseteq \|\Phi[I][w][i]\|) : P \Phi[I][w][i] \subset \Xi\} \}
$$

where $\forall i \in \Delta[w]$, $\|\Phi[I][w][i]\|$ is defined by $\Xi(I)[w][i]$. 

\[\square\]

Definition 43 (Well-formed well-founded system of inductive definitions) The well-founded system of inductive definitions $\Theta = (W, \rightarrow, \Delta, U, \Phi, \bot, \cup)$ is well-formed if and only if $\Xi[I][w][i]$ is defined by \(\Xi(I)[w][i]\) inductively defined by $\Theta$, $\square$.

If each $\Phi(U[I][w][i])$ is a cpo (or a complete lattice) then propositions 9, 11, 13 and the above proof methods are easily extended to well-founded systems of inductive definitions by componentwise induction on the well-founded relation $(W, \rightarrow)$.

When defining subsets of a cartesian product of sets, the well-founded set $(W, \rightarrow)$ can be omitted since it is reduced to a singleton. This would be the case for example when understanding a context-free grammar as a system of inductive definitions. The next example is a well-founded inductive definition (the index set $\Delta$ being omitted since it is reduced to a singleton).

Example 44 (Weak fairness) Let $A$ be an alphabet and $X$ be a finite subset of $A$. The set $F[X]$ of finite words containing at least one occurrence of each $x \in X$ is defined by:

$$y \in A \land x \in X \Rightarrow \frac{y \in F[W] +}{x \in F[X]}$$

$$y \in A \land X \subseteq Y \land \sigma \in F[Y] \Rightarrow \frac{y \in F[Y]}{x \sigma \in F[Y \cup \{x\}]}$$

The set $I[X]$ of infinite words with infinitely many occurrences of each $x \in X$ is defined by:

$$\sigma \in F[X] \land \exists \zeta \in I[X]$$

This is a very simple example with $W = \{F[Y] | Y \subseteq X\} \cup \{I[X]\}$, and $F[Y] \rightarrow F[Z]$ when $Y \subseteq Z \subseteq X$ and $F[Y] \rightarrow I[X]$ when $Y \subseteq X$. \(\square\)

Negation can be used in the premises of the rules of iterated definitions: a set $X[\lambda]$ can be inductively defined in terms of $X[\lambda]$ as well as $X[\alpha]$ and $\neg X[\alpha]$ for $\alpha < \lambda$. In particular this generalizes J. Groote’s transition system specifications with negative premises [16] without resorting to 3-valued minimal model or stable model approaches [6] originating from logic programming, for which the logical meaning is not always simple and clear [19].

Combining systems and well-founded inductive definitions, we get well-founded systems of inductive definitions as illustrated by the following example:

Example 45 (Maximal execution traces III) Let $T = (S, \mathbb{A}, \{\lambda \vdash \alpha \in \mathbb{A}\}, \mathbb{A}[I])$ be a labelled transition system.

The set $T[\lambda] (T[\lambda] \land T[\leq \lambda])$, $\lambda \leq \omega$, of maximal execution traces of length $\lambda$ (respectively strictly less than $\lambda$ and less than or equal to $\lambda$) can be defined as follows ($s, s', s'' \in S$, $0 \leq n < \omega$, $\lambda \leq \omega$):

$$\begin{align*}
s &\approx \bot  \\
\sigma \in T[\omega][s] &\land \sigma \in \mathbb{A}[I][\omega][s'] &\land \sigma \in \mathbb{A}[I][\omega][s'']
\end{align*}$$

The partial ordering $\prec$ such that for all $0 \leq n < \lambda \leq \omega$, $n \rightarrow \rightarrow n + 1 \rightarrow n \rightarrow n$, $\omega \prec \omega$, $n \prec \lambda \rightarrow \lambda \prec \lambda \rightarrow \lambda \prec \leq \lambda$, is well-founded. \(\square\)

Such iterated inductive definitions are very powerful since they have an expressive power greater than D. Park’s $\mu$-calculus [26] or E. Emerson’s CTL* [14].

These iterated inductive definitions are used in $\mu$SOS to define the semantics of programming languages by induction on the syntax of programs. In this case, $W$ is the set of all program components while $\rightarrow$ means “is a sub-component of”.  

90
2 Abstract interpretation of inductive definitions

The quest for a unique general-purpose semantics for programming languages has failed. A better approach is to establish correspondences between various semantics at different levels of abstraction. As noticed by E. Astesiano [4] and G. Reggio [29], rule-based inductive definitions should form a unifying framework for expressing these semantics. A first step toward this goal is to describe finite and infinite program behaviors in the same way. A second step consists in adopting an abstract interpretation approach [8] [10] in order to relate inductive definitions. Abstract interpretation can be used, as follows, to justify and even to formally construct abstract rules in terms of a concrete ground semantics (named static in [8] and renamed collecting in [24]) (since it could be used in e.g. execution traces as it is the case in our examples):

Definition 46 (Operator abstraction) for standing either for = or \text{?}, we write:

\[ [S \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [S' \langle \varepsilon', \bot', \downarrow' \rangle, F'] \]  (38)

to mean that:

- \( S \langle \varepsilon, \bot, \downarrow \rangle \) is a cpo such that \( \downarrow \in S \),
- \( S' \langle \varepsilon', \bot', \downarrow' \rangle \) is a partial order such that \( \downarrow' \in S' \),
- \( F \in S \overset{\text{def}}{=} S', F' \in S' \overset{\text{def}}{=} S' \\
- \( \sigma \in S \overset{\text{def}}{=} \downarrow' \) and
- \( \forall \lambda \leq \text{clo}_\Delta \alpha(F) : \alpha(F^\lambda) \prec F^\lambda \) (iteration from \( F^0 = \downarrow \) and \( F^0 = \downarrow' \)).

\( \square \)

The above condition \( \forall \lambda \leq \text{clo}_\Delta \alpha(F) : \alpha(F^\lambda) \prec F^\lambda \) is implied by the following ones:

- \( \alpha(\bot) \prec \downarrow' \),
- \( \forall \lambda \leq \text{clo}_\Delta \alpha(F) : \sigma \circ F(\alpha) \prec F' \circ \alpha(\alpha) \) and
- for all limit ordinals \( \lambda \in \text{Limit} \) such that \( \lambda \leq \text{clo}_\Delta \alpha(F) \), we have \( \langle \cup_{\lambda \leq \lambda} F^\lambda \rangle \prec \cup_{\lambda \leq \lambda} \alpha(F^\lambda) \).

and by the strongest ones:

- \( \alpha(\bot) \prec \downarrow' \),
- \( \sigma \circ F \prec F' \circ \alpha \)
- \( \alpha(\cup_{\lambda \leq \lambda} x^\lambda) \prec \cup_{\lambda \leq \lambda} \alpha(x^\lambda) \) for all increasing chains \( \langle x^\delta \mid \delta \leq \text{clo}_\Delta \alpha(F) \rangle \).

This last condition is implied by the fact that \( \alpha \) is a complete join morphism, which is the case when it is the upper adjoint of a Galois connection:

Definition 47 (Galois connection) We write \( P(\leq) = \overset{\text{def}}{=} Q(\leq) \) to mean that:

- \( P(\leq) \) and \( Q(\leq) \) are posets,
- \( \sigma \in P \mapsto Q \),
- \( \gamma \in Q \mapsto P \) and
- \( \forall x \in P : \forall y \in Q : [\alpha(x) \leq y] \Leftrightarrow [x \leq \gamma(y)] \).

\( \square \)

In this case, we write \( [S \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [S' \langle \varepsilon', \bot', \downarrow' \rangle, F'] \)

Proposition 48 If \( [S \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [S' \langle \varepsilon', \bot', \downarrow' \rangle, F'] \) then \( \alpha(F) \sqsubseteq \langle \upharpoonright F \rangle \).

Abstractions are usually specified by successive compositions:

Proposition 49 If \( [S \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [S' \langle \varepsilon', \bot', \downarrow' \rangle, F'] \) and \( [S' \langle \varepsilon', \bot', \downarrow' \rangle, F'] \overset{\text{ab}}{\mapsto} [S'' \langle \varepsilon'', \bot'', \downarrow'' \rangle, F''] \) then \( [S \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [S'' \langle \varepsilon'', \bot'', \downarrow'' \rangle, F''] \) where \( \prec_1 \) (respectively \( \prec_2 \)) stands either for = (resp. \( \subseteq \)) or \( \sqsubseteq \) (resp. \( \sqsubseteq \)).

Definition 50 (Inductive definition abstraction) If \( \exists = (U, F, \bot, \downarrow) \) and \( \exists' = (U', F', \bot', \downarrow') \) then we write \( \exists \overset{\text{ab}}{\mapsto} \exists' \) for \( [p(U) \langle \varepsilon, \bot, \downarrow \rangle, F] \overset{\text{ab}}{\mapsto} [p(U') \langle \varepsilon', \bot', \downarrow' \rangle, F'] \) where \( \exists \) and \( \exists' \) are the partial orderings respectively induced by \( U \) and \( U' \) and \( \neg \) stands either for = or \( \subseteq \).

If moreover \( p(U) \langle \varepsilon, \bot, \downarrow \rangle \overset{\text{ab}}{\mapsto} p(U') \langle \varepsilon', \bot, \downarrow' \rangle \) then we write \( \exists \overset{\text{ab}}{\mapsto} \exists' \).

\( \square \)

Corollary 51 If \( \exists \overset{\text{ab}}{\mapsto} \exists' \) then \( \alpha(\exists) \overset{\text{ab}}{\mapsto} \alpha(\exists') \)

Example 52 (Disjoint bi-inductive definition) A positive inductive definition \( \exists^+ = (U^+, F^+, \bot^+ \downarrow^+) \) and a negative one \( \exists^- = (U^-, F^- \downarrow^+) \) such that \( U^+ \cap U^- = \emptyset \) can be combined into a system of inductive definitions \( \exists = (\Delta, U, F, \bot, \downarrow) \) where \( \Delta = \{1, 2\}, U = U^+ \times U^-, F = \{[F_1, F_2] \mid F_1 \in F^+, F_2 \in F^-\} \cup \{[u \in U] \mid u \in \bot\} \), \( \downarrow = (\emptyset, U^-) \) and \( U_{\in S} X_u = \{u \in U \mid u \in S \} \). They can also be combined into a disjoint bi-inductive definition \( \exists^\perp = (U^+, U^-, \bot^-, F^- \bot^+) \). The bijection is established by \( \alpha(X, Y) = X \cup Y \) and \( \gamma(X) = (X \cap U^+, X \cap U^-) \), so that by propositions 27 and 51, we have \( [\exists^+ \cup \exists^-] = \exists^\perp = \alpha(\exists) \).

A simple way to abstract inductive definitions is to use an abstraction of subsets of the universe:

Proposition 53 Let \( \exists = (U, F, \bot, \downarrow) \) be a monotonic inductive definition on a cpo \( p(U) \langle \varepsilon, \bot, \downarrow \rangle \), \( p(U') \langle \varepsilon', \bot', \downarrow' \rangle \) be a partial order and \( \alpha \in p(U) \overset{\text{ab}}{\mapsto} p(U') \) be a complete \( U \)-morphism (for all sets \( \{F_i\} \beta < \lambda, \in \text{Ord} \)) and a complete \( U \)-morphism such that:

\[ \forall x \in \Phi : \forall x \in U : \alpha(P) \subseteq \alpha(X) \Rightarrow \exists \overset{\text{ab}}{\mapsto} \exists' \in \Phi : F' \subseteq X \wedge \alpha(Q) \subseteq \alpha(\{c\}) \]  (39)

Define \( \Phi \overset{\text{def}}{=} \{ \alpha(P) \mid \alpha \in \Phi \wedge \forall \alpha(Q) \subseteq \alpha(\{c\}) \} \) and \( \downarrow \overset{\text{def}}{=} \alpha(\bot) \). Then \( \exists \overset{\text{ab}}{\mapsto} \alpha(U), \Phi', \bot', \downarrow' \) is a well formed inductive definition such that \( S \overset{\text{ab}}{\mapsto} \exists' \). If moreover \( p(U) \langle \varepsilon, \bot, \downarrow, \uparrow, \bot \rangle \) is a complete lattice then \( \exists \overset{\text{ab}}{\mapsto} \exists' \) where \( \gamma \in p(U') \mapsto p(U) \) is \( \gamma(Y) = \{X \in U \mid \alpha(X) \subseteq \exists' \} \).

\( \square \)
This notion of abstraction can be used to show the equivalence of inductive definitions.

**Example 54 (Maximal execution traces IV)** In example 21 we have characterized the execution traces $T_{\Sigma^w}$ of a transition system $T = \langle S, Act, \{\alpha \mid \alpha \in Act\}$, 

which by a negative inductive definition $\Sigma^{-} = \langle \Sigma_{\Sigma^w}, \Phi^{-}, \Sigma_{\Sigma^w}, \emptyset \rangle$ where $\Phi^{-}$ is specified by schema (12) and $\rho(\Sigma_{\Sigma^w}) = \langle \Sigma_{\Sigma^w}, \emptyset \rangle$ is the induced cpo. In example 28 we have characterized $T_{\Sigma^w}$ by a disjoined inductive definition $\Delta^+ = \langle \Sigma_{\Sigma^w}, \Sigma^w, \Phi^+ \rangle$ where $\Phi^+$ is specified by schema (14) and (15) and $\rho(\Sigma_{\Sigma^w}) = \langle \Sigma_{\Sigma^w}, \bot^+, \bot^+ \rangle$ is the induced cpo.

In order to relate them, we observe that there are abstractions of a common ground semantics:

\[ \Delta_{\Sigma^w} \times (\omega + 1) \langle \emptyset, \bot, \bot \rangle, \rho \] (40)

such that $\langle X, \lambda \rangle \subseteq \langle X', \lambda' \rangle$ def $\langle X \subseteq X', \lambda \leq \lambda' \rangle$, $\bot^+ \subseteq \{\bot^+, \emptyset\}$, $\rho(U) = \langle U, \bot^+ \rangle$ where $\bot^+ = \{\bot^+, \emptyset\}$ and $\rho(\Sigma_{\Sigma^w}) = \langle \Sigma_{\Sigma^w}, \bot^+, \bot^+ \rangle$.

By proposition 48, it follows that $\alpha^{-}(U_{\Sigma^w} \phi) = \rho(U)^{\rho^{-1}} \emptyset^\gamma = \emptyset$. Abstraction can also be used to relate definitions of the semantics of languages described at different levels of details.

**Example 55 (Erratic, demoniac and angelic relational semantics of transition systems)**

- The **erratic relational semantics** is obtained by abstraction of $T_{\Sigma^w}$ as characterized by the disjoined inductive definition $\Delta^+ = \langle \Sigma_{\Sigma^w}, \Sigma^w, \Phi^+ \rangle$ def of example 28, where $\Phi^+$ is specified by schema (14) and (15) and $\rho(U) = \langle U, \bot^+, \bot^+ \rangle$ where $\rho(U) = \langle U, \bot^+, \bot^+ \rangle$ is the induced cpo.

- The **demoniac relational semantics** is obtained by abstraction of $T_{\Sigma^w}$ as characterized by the negative inductive definition $\Sigma^{-} = \langle \Sigma_{\Sigma^w}, \Phi^{-}, \Sigma_{\Sigma^w}, \emptyset \rangle$ of example 21 where $\Phi^{-}$ is specified by schema (12) and $\rho(\Sigma_{\Sigma^w}) = \langle \Sigma_{\Sigma^w}, \emptyset \rangle$ is the induced cpo.

Finite traces are approximated by the pair of their initial and final states while infinite ones are simply ignored. This can be formalized by the erratic abstraction $\alpha^1(\rho(U)) = \rho(U)$ such that:

\[ \alpha^1(X) = \{\{\sigma_0, \sigma_{\rho^{-1}}(s)\mid \sigma \in X \cap \Sigma_{\Sigma^w} \} \cup \{\sigma_0, x\mid \sigma \in X \cap \Sigma_{\Sigma^w} \} \} \subseteq X \cap \Sigma_{\Sigma^w} \cup X \cap \Sigma_{\Sigma^w} \] (41)

\[ s \in T^1 \rightarrow \alpha^1(s') \wedge \langle s', x \rangle \in T^1 \] (42)

\[ s \in T^1 \rightarrow \alpha^1(s') \wedge \langle s', x \rangle \in T^1 \] (43)

\[ \bot^1 \equiv S \times \{\bot\} \] (44)

\[ \cup_{\bot^1 \bot} \equiv \cup \pi^1+r(\bot^1) \cup r \pi^1=\bot(\bot^1) \] (45)

The approximation can be formalized by the erratic abstraction $\alpha^1(\rho(U)) = \rho(U)$ such that:

\[ \alpha^1(X) = \{\{\sigma_0, \sigma_{\rho^{-1}}(s)\mid \sigma \in X \cap \Sigma_{\Sigma^w} \} \cup \{\sigma_0, x\mid \sigma \in X \cap \Sigma_{\Sigma^w} \} \} \subseteq X \cap \Sigma_{\Sigma^w} \cup X \cap \Sigma_{\Sigma^w} \] (41)

\[ s \in T^1 \rightarrow \alpha^1(s') \wedge \langle s', x \rangle \in T^1 \] (42)

\[ \bot^1 \equiv S \times \{\bot\} \] (44)

\[ \cup_{\bot^1 \bot} \equiv \cup \pi^1+r(\bot^1) \cup r \pi^1=\bot(\bot^1) \] (45)

Then proposition 53 implies $\exists^+ \rho(\Sigma_{\Sigma^w}) \exists^1$ since $\alpha^1(\rho(U)) = \rho(U)$ and $\alpha^1(\bot^1) = \bot^1 = \emptyset$. By proposition 49, we conclude $\exists^+ \alpha^1 \rho(\Sigma_{\Sigma^w}) \exists^1$.

- The **demoniac relational semantics** is obtained by abstraction of $T_{\Sigma^w}$ as characterized by the negative inductive definition $\Sigma^{-} = \langle \Sigma_{\Sigma^w}, \Phi^{-}, \Sigma_{\Sigma^w}, \emptyset \rangle$ of example 21 where $\Phi^{-}$ is specified by schema (12) and $\rho(\Sigma_{\Sigma^w}) = \langle \Sigma_{\Sigma^w}, \emptyset \rangle$ is the induced cpo.

Finite traces are approximated by the pair of their initial and final states and the infinite ones by their initial state together with $\bot$ denoting non-termination. Let us define $U_{\bot^1} \equiv S \times S, U_{\bot^1} \equiv S \times \{\bot\}$ where $\bot \notin S$ and $U_{\bot^1} \equiv U_{\bot^1} \lor U_{\bot^1}$. By propositions 21.1 and 27, $\rho(U_1) = \langle S, \bot^1, \bot^1, \bot^1 \rangle$ is a complete lattice, where:

\[ \pi^1+(r) \equiv r \cap (S \times S) \] (41)

\[ \pi^1=\bot(r) \equiv r \cap (S \times S) \] (42)

\[ \tau \bot^1 \equiv \tau \cap S \times S \] (43)

\[ \bot^1 \equiv S \times \{\bot\} \] (44)

\[ \cup_{\bot^1 \bot} \equiv \cup \pi^1+r(\bot^1) \cup r \pi^1=\bot(\bot^1) \] (45)

Then proposition 53 implies $\exists^+ \rho(\Sigma_{\Sigma^w}) \exists^1$ since $\alpha$ is a complete $\cup$-morphism, property (39) holds, $\alpha^1(\bot^1) = \bot^1$ and $\alpha^1(\bot^1) = \bot^1 = \emptyset$. By proposition 49, we conclude $\exists^+ \alpha^1 \rho(\Sigma_{\Sigma^w}) \exists^1$.

Then proposition 53 implies $\exists^+ \rho(\Sigma_{\Sigma^w}) \exists^1$ since $\alpha$ is a complete $\cup$-morphism, property (39) holds, $\alpha^1(\bot^1) = \bot^1$ and $\alpha^1(\bot^1) = \bot^1 = \emptyset$. By proposition 49, we conclude $\exists^+ \alpha^1 \rho(\Sigma_{\Sigma^w}) \exists^1$.
Let \( \mathcal{I} = \langle U^1, \Phi^1, U^1, \cap \rangle \) be the negative inductive definition such that the abstract rules instances \( \Phi^1 \) are given by the following schemata (where \( s, s', s'' \in S \) and \( x \in S \cup \{ \bot \} \)):

\[
\begin{align*}
\frac{s \leftarrow s', s'' \in T^1}{\langle s, s' \rangle \in T^1} & \quad (53) \\
\frac{s \leftarrow s', \forall y \in S \cup \{ \bot \} : \langle s', y \rangle \in T^1}{\langle s, x \rangle \in T^1} & \quad (54)
\end{align*}
\]

Then proposition 53 implies \( \exists s \in T^1 \). Then rule schema (54) can be simplified into:

\[
\frac{s \leftarrow s', \bot \in T^1}{\langle s, x \rangle \in T^1} 
\]

since \( \bot \in T^1 \) implies \( \forall y \in S \cup \{ \bot \} : y \in T^1 \).

By further abstractions one can derive order domains based state transformation semantics [3].

\[\text{3 G}\mathbf{\times} \text{SOS} \]

We now introduce \( \text{G}\mathbf{\times} \text{SOS} \), a generalization of SOS (G. Plotkin’s structured operational semantics [28]) using the above rule-based systems of transfinitely iterated inductive definitions. \( \text{G}\mathbf{\times} \text{SOS} \) enables us to describe the finite, as well as the infinite executions of programs. The nature of the finitary or infinitary objects representing terminating and non-terminating program executions is not fixed and depends upon the considered language. For example we have defined the \( \text{G}\mathbf{\times} \text{SOS} \) semantics of \( \text{\Lambda}\)-calculi [27] using judgements \( \rho \vdash E \sim \nu \) (expression \( E \) evaluates to \( \nu \) in environment \( \rho \)) and \( \rho \vdash E \sim \bot \) (evaluation of expression \( E \) does not terminate in environment \( \rho \)); K. Apt and G. Plotkin’s nondeterministic language [3] using maximal finite and infinite execution traces and R. Milner’s CCS using infinite synchronizing trees [21] and partial orders [13] with infinite chains.

**Example 56 (Trace semantics of while loops)**

In order to define the operational semantics of an imperative language, let \( \rho \in \Gamma \) be an environment (recording the values of identifiers), \( c \in C \) be a command, \( s \in S = (\Gamma \times C) \cup \Gamma \) be a state written \( \rho \vdash c \) or \( \rho \), \( U^+ = \Sigma^{\leq w} \) the set of finite traces, \( U^- = \Sigma^w \) the set of infinite traces and \( U = \Sigma^{\leq w} = \Sigma^w \cup \Sigma^w \) be the set of traces over states \( S \) and actions \( C \). For short assume that the evaluation of a boolean expression \( b \in B \) in environment \( \rho \) always terminates without error and yields a boolean value \( \mathcal{B}[b]([\rho]) \). If \( \sigma \in \Sigma^{\leq w} \) and \( c \in C \), define \( \sigma \upbrace{c} \) by \( \sigma \upbrace{c} = \rho \vdash c \), \( \rho \vdash c \) \( \upbrace{c} \) \( \upbrace{c} \) \( \upbrace{c} \) \( \upbrace{c} \) \( \upbrace{c} \) \( \upbrace{c} \)

\[
\begin{align*}
\frac{-\mathcal{B}[b]([\rho])}{\rho \vdash c \sim \sigma \vdash \mathcal{B}[b]([\rho])} & \quad (56) \\
\text{A terminating or non-terminating execution of the loop starts with a first step } \rho \vdash w \vdash \rho \vdash c \vdash \text{ to evaluate the test } b \text{ to true followed by an execution } \rho \vdash c \sim \sigma \vdash \mathcal{B}[b]([\rho]) \vdash c \vdash \text{ of the loop body } c \text{ (where control states memorize the fact that further iterations of } w \text{ may be necessary: } (\rho \vdash c \sim \sigma \vdash \mathcal{B}[b]([\rho]) \vdash c \vdash \text{ followed by the remaining iterations } \rho \vdash w \vdash c \vdash w \vdash (c \upbrace{c}) \vdash \text{ where } c \vdash (c \upbrace{c}) \vdash \text{ or } c \vdash (c \upbrace{c}) \vdash \text{ is inferred. It is a sound principle just because lengths of finite executions ordered by } < \text{ are well-founded. This argument is no longer valid in the instance of rule (57) for finite execution traces since non-terminating executions of } w \text{ in environments } \rho \text{ take strictly less steps than its execution in environment } \rho. \text{ This can also be understood as action induction [21], an induction upon the depth of the inference by which } \rho \vdash w \vdash \rho \vdash c \vdash w \vdash (c \upbrace{c}) \vdash \text{ is inferred. It is a sound principle just because lengths of finite executions ordered by } < \text{ are well-founded. This argument is no longer valid in the instance of rule (57) for finite execution traces since non-terminating executions of } w \text{ in environments } \rho \text{ and } \rho \text{ both take infinitely many steps. This shows that negative rules are useful to provide a direct description of non-termination.} \]

Denotational semantics [23] has several important advantages over traditional operational semantics:

1. The semantics of programs is given in terms of mathematical models (domain theory [17]).
2. Denotations are defined by induction on the abstract syntax of programs.
3. Finite and infinite program behaviors are handled in the same way (using fixpoints).

SOS copes with point (2) but not (3). One can define a big-steps SOS semantics where only error-free, terminating behaviors of programs are described. Alternatively, one can define a small-steps semantics where

\[\text{2These examples, as others, are omitted for lack of space.} \]
non-terminating executions have to be described using another formalism such as execution traces (see [3]). Both approaches are incomplete since the first describes the “good” cases and leaves out the “bad” ones while the second provides a microscopic view of a macroscopic process. Using both approaches simultaneously implies a lot of work to relate them, and this has to be done again and again for each language [4]. \(G^{\infty}\) SOS can cope with terminating and non-terminating executions whence the prefix \(G^{\infty}\) added to SOS indicating the generalization to infinite behaviors. The correct handling of non-termination is necessary, for example to define fair executions of parallel processes or to serve as a ground semantics for the inference of liveness properties of programs by abstract interpretation such as strictness analysis. This abstraction process can also be used to define more abstract semantics as shown by the following:

**Example 57 (Relational semantics of while loops)** To obtain the big-step semantics of commands, we define an abstraction \(\alpha \in \rho(\Sigma^{\infty}) \rightarrow \rho(\Gamma \times \Gamma_{\perp})\) by \(\alpha(\{\sigma_{i} \mid i \in \Delta\}) \stackrel{\text{def}}{=} \{\alpha(\sigma_{i}) \mid i \in \Delta\}\) where \(\Gamma_{\perp} \stackrel{\text{def}}{=} \Gamma \cup \{\bot\}\), \(\alpha \in \Sigma^{\infty} \rightarrow \Gamma \times \Gamma_{\perp}\) is given by \(\alpha(\rho \perp c \perp \sigma \perp \rho') \stackrel{\text{def}}{=} \langle \rho, \rho' \rangle\) for finite traces and by \(\alpha(\rho \perp c \perp \sigma) \stackrel{\text{def}}{=} \langle \rho, \bot\rangle\) for infinite traces. The relational semantics of \(c \in C\) is then \(\mathcal{R}[c] = \alpha(\{\mathcal{T}[c][\rho] \mid \rho \in \Gamma\})\). We write \(\rho \perp c \perp \rho'\) for \(\langle \rho, \rho' \rangle \in \mathcal{R}[c]\). The natural semantics [18] of commands is \(\mathcal{R}[c] \cap \Sigma^{\infty}\) since it only deals with terminating executions. By proposition 53 applied to (56), (57), (58), it is defined for the while loop \(w \equiv \textbf{while } b \textbf{ do } c\) by the following rule schemata:

\[
\begin{align*}
&\begin{array}{c}
\textbf{null}\[\rho] \land \\
\rho \perp w \perp \rho' \land \\
\rho \perp w \perp \rho' \\
\end{array} \\
&\begin{array}{c}
\textbf{true}\[\rho] \land \\
\rho \perp c \perp \rho' \land \\
\rho \perp c \perp \rho' \land \\
\rho \perp w \perp \rho' \\
\end{array} \\
&\begin{array}{c}
\textbf{false}\[\rho] \land \\
\rho \perp w \perp \bot \\
\rho \perp c \perp \bot \land \\
\rho \perp c \perp \bot \land \\
\rho \perp w \perp \bot \\
\end{array}
\end{align*}
\]

Using \(G^{\infty}\) SOS, non-termination can be expressed directly:

\[
\begin{align*}
&\textbf{null}\[\rho] \land \\
&\textbf{true}\[\rho] \land \\
&\textbf{false}\[\rho] \land \\
&\rho \perp w \perp \bot \\
&\rho \perp c \perp \bot \land \\
&\rho \perp c \perp \bot \\
&\rho \perp w \perp \bot \\
\end{align*}
\]

\(\Box\)

**Example 58 (\(G^{\infty}\) SOS semantics of the non-deterministic choice)** To illustrate fairness, let us consider the simple case of the non-deterministic choice operator \([c_{1} ! c_{2}]\).

- With Plotkin’s erotic semantics, termination or non-termination of \([c_{1} ! c_{2}]\) is possible whenever that of \(c_{1}\) or \(c_{2}\) is possible:

\[
\begin{align*}
&\rho \perp c_{1} \perp c' \land \\
&\rho \perp [c_{1} ! c_{2}] \perp c' \\
&\rho \perp c_{1} \perp \bot \\
\end{align*}
\]

\(\Box\)

- With Hoare’s fair angelic semantics, termination is possible when execution of \(c_{1}\) or that of \(c_{2}\) may terminate whereas non-termination of \([c_{1} ! c_{2}]\) requires that both \(c_{1}\) and \(c_{2}\) cannot terminate:

\[
\begin{align*}
\rho \perp c_{1} \perp c' & \land \rho \perp [c_{1} ![c_{2}] \perp c' \\
& \rho \perp c_{2} \perp \bot \\
\end{align*}
\]

(As pointed out by M. Brob, if \(\rho \perp c_{1} \perp \bot, \rho \perp c_{2} \perp 2\) and \(\rho \perp c_{1} \perp \bot, \rho \perp [c_{1} ![c_{2}] \perp 1\), \(\rho \perp [c_{1} ![c_{2}] \perp 2\) but \(\rho \perp [c_{1} ![c_{2}] \perp \bot\) is not true, a miracle since \([c_{1} ![c_{2}]\) must be able to avoid non-termination in the erratic behavior of both \(c_{1}\) and \(c_{2}\))

- With McCarthy’s fair parallel semantics, non-termination is possible when both executions of \(c_{1}\) and \(c_{2}\) may not terminate:

\[
\begin{align*}
&\rho \perp c_{1} \perp c' \land \rho \perp [c_{1} ![c_{2}] \perp c' \\
&\rho \perp c_{2} \perp \bot \\
\end{align*}
\]

\(\Box\)

- With Smyth’s unfair demonic semantics, termination of \([c_{1} ![c_{2}]\) is possible only when both that of \(c_{1}\) and \(c_{2}\) are guaranteed whereas non-termination is possible whenever that of \(c_{1}\) or \(c_{2}\) is possible:

\[
\begin{align*}
&\rho \perp c_{1} \perp c' \land \rho \perp [c_{1} ![c_{2}] \perp c' \\
&\rho \perp c_{2} \perp \bot \\
\end{align*}
\]

(166)

\[
\begin{align*}
&\rho \perp c_{1} \perp \bot \\
&\rho \perp c_{2} \perp \bot \\
\end{align*}
\]

\(\Box\)

- With the unfair, à la Prolog, left to right semantics, termination of \([c_{1} ![c_{2}]\) is possible when that of \(c_{1}\) is possible or when \(c_{1}\) cannot diverge and \(c_{2}\) may terminate. Non-termination of \([c_{1} ![c_{2}]\) is possible when that of \(c_{1}\) is possible or when \(c_{1}\) cannot diverge but \(c_{2}\) can:

\[
\begin{align*}
&\rho \perp c_{1} \perp c' \land \rho \perp [c_{1} ![c_{2}] \perp c' \\
&\rho \perp c_{2} \perp \bot \\
\end{align*}
\]

\(\Box\)

This last example illustrates the **semantical compositionality** property: we can change the semantics of the non-deterministic choice command without having to change the semantics of other (while, if, ...)
(sub)commands. In denotational semantics, one will have to use Plotkin, Hoare, Smyth, ... powerdomains [17] to describe the behavior of the choice command. By doing so one will have to change the ordering used for computing fixpoints, hence potentially the semantics of other commands (such as while loops or recursion). More generally, semantical compositionality requires that the specification of the whole should be done without interfering with the specification of the components.

As a last abstraction, let us consider predicate transformers:

**Example 59 (Predicate transformers)** The predicate transformer of command $c \in C$ is $\wp[c] = \alpha(R[c])$ where the abstraction $\alpha \in \wp(\Gamma \times \Gamma_l) \to (\wp(\Gamma) \to \wp(\Gamma))$ is defined by $\alpha(R) = \lambda P. \{ \rho \mid (3\rho' \in P : (\rho', \rho)^t \in R \land (3\rho' : (\rho', \rho)^t \in R \Rightarrow \rho' \in P) \land (\rho, \bot) \notin R \}$. Since $R[c] \neq \emptyset$, proposition 55 applied to (59) and (60) yields the following definition of $\wp[u]$ where $u \equiv \emptyset$ while $h$ do e:

$$
\frac{\beta[c]\rho \land \rho \in \wp[u]}{\wp[u](P)} \quad \frac{\beta[Q]\rho \land \rho \in \wp[u]}{\wp[u](P)} - (71)
$$

Further abstractions would perfect the lattice of abstract interpretations considered in [10]. Abstract interpretation was first introduced using transition systems [7] that is an operational semantics. Mycroft [24], followed by Nielson [25], advocated using denotational instead of operational base semantics. We think that G^\infty SO5 is better suited for designing ground semantics from which other, more abstract or approximate, semantics can be derived. In particular denotational semantics, which are abstract interpretations of operational behaviors, can be understood as intermediate steps in the approximation process.

**Acknowledgements** We would like to thank the members of IFIP WG 2.3 present at Santa Catalina and Pouilly en Auxois meetings for discussions.

**References**


