Acta Informatica manuscript No.
(will be inserted by the editor)

Rough notes on Abstract Interpretation and Coinduction Up-to

Filippo Bonchi · Roberto Giacobazzi · Dusko Pavlovic

Received: April 2017 / Accepted: NEVER

Abstract In these notes, we report some of the result obtained during the visit of Roberto Giacobazzi at Dusko's lab in April 2017. We pay particular attention to *proofs* since we expect that, being both abstract interpretation and coinduction up to meta-theory of proofs, the proofs themselves can inspire new results.

Summary of the results. We observed several analogies between Abstract Interpretation and Coinduction up-to, as reported in Table 1. The notion of compatibility of an up-to technique was formerly introduced in the setting of Abstract Interpretation, under the name of Backward completeness. One major technology transfert comes from the modularity of up-to techniques, which has not been so much developed in abstract interpretation, because of the focus on abstract domains (up-closure functions).

Whenever b has a right adjoint b^+, the notions of forward completeness for b and backward completeness for b^+ collapse. So both entail fixed-point completeness for abstract interpretation and soundness of up-to techniques. It is not clear yet, wether also the two problems collapse: this would be a major achievement.

A further direction to explore is the correspondence between the companion, recently developed in coinduction up-to and the most abstract complete domain, important in several application of abstract interpretation.

ToDo 0.1. *Read Samson's paper: https://www.dropbox.com/s/snbbyyrxe5waee9/absint-samson.pdf?dl=0*

ToDo 0.2. *Read papers by Cousot: > http://www.di.ens.fr/ cousot/COUSOTpapers/publications.www/CousotCousot-PacJMath-82-1-1979.pdf <http://www.di.ens.fr/ cousot/COUSOTpapers/publications.www/CousotCousot-PacJMath-82-1-1979.pdf>*

F. Bonchi
LIP, ENS Lyon, 46 Allée d'Italie, 69364 Lyon, FRANCE
E-mail: {filippo.bonchi, damien.pous, jurriaan.rot}@ens-lyon.fr

	Abstract Interpretation	Coinduction up-to
$b\colon C \to C$	Least fixed-point μb	Greatest fixed-point νb
$a\colon C \to C$	abstract domain	up-to techinque
final aim	Completeness	Soundness
sufficient condition	Forward completeness: $ba \sqsubseteq ab$	Compatibility: $ab \sqsubseteq ba$
smart tricks	Most abstract complete domain	Companion

Table 1 Analogies between Abstract Interpretation and Coinduction up-to.

1 Notation and preliminaries

We introduce basic notation, concepts and results from lattice theory.

We use $(L, \sqsubseteq)$, $(L_1, \sqsubseteq_1)$, $(L_2, \sqsubseteq_2)$ to range over complete lattices and x, y, z to range over their elements. We omit the ordering $\sqsubseteq$ whenever unecessary. As usual $\bigsqcup$ and $\bigsqcap$ denote greatest lower bound and least upper bound, $\sqcup$ and $\sqcap$ denote join and meet, $\top$ and $\bot$ top and bottom.

Hereafter we will consider just monotone maps, so we will usually omit to specify that they are monotone. Obviously, the identitiy map $id\colon L \to L$ and the composition $f \circ g\colon L_1 \to L_3$ of two monotone maps $g\colon L_1 \to L_2$ and $f\colon L_2 \to L_3$ are monotone. Given $l\colon L_1 \to L_2$ and $r\colon L_2 \to L_1$, we say that l is the *left adjoint* of r, or equivalently that r is the *right adjoint* of l, written $l \dashv r\colon L_1 \to L_2$, iff $x \sqsubseteq_1 rl(x)$ and $lr(y) \sqsubseteq_2 y$ for all $x \in L_1$ and $y \in L_2$. Observe that if it exists the adjoint is unique. We will often use the fact that a map f is a left adjoint iff it preserves $\bigsqcup$ (in particular $\bot$) and a right adjoint iff it preserves $\bigsqcap$ (in particular $\top$).

Given a monotone map $f\colon L \to L$, x is said to be a *post-fixed point* iff $x \sqsubseteq f(x)$ and a *pre-fixed point* iff $f(x) \sqsubseteq x$. A *fixed point* iff $x = f(x)$. Pre, post and fixed points form complete lattices, denoted by $Pre(f)$, $Post(f)$ and $Fix(f)$: we write μf and νf for the least and greatest fixed-point. Observe that if $l \dashv r$, then $x \sqsubseteq r(x)$ iff $l(x) \sqsubseteq l(r(x))$ iff $l(x) \sqsubseteq x$ and moreover $\mu l = \bot$ and $\nu r = \top$.

For a map $f\colon L \to L$, we inductively define $f^0 = id$ and $f^{n+1} = f \circ f^n$. We fix $f^\uparrow = \bigsqcup_{i\in\omega} f^i$ and $f^\downarrow = \bigsqcap_{i\in\omega} f^i$. A monotone map $f\colon L \to L$ is an *up-closure* operator if $x \sqsubseteq f(x)$ and $ff(x) \sqsubseteq x$. It is a *down-closure* operator if $f(x) \sqsubseteq x$ and $x \sqsubseteq ff(x)$. For any f, $f^\uparrow$ is an up-closure and $f^\downarrow$ is a down-closure. Closure operators and pairs of adjoints are in one to one correspondences: for a pair of adjoint $l \dashv r$, $r \circ l$ is an up-closure operator and $l \circ r$ a down-closure operator. Given an up-closure operator $f\colon L \to L$, the functions $l\colon L \to Pre(f)$, defined as $l(x) = \bigsqcap\{y \mid x \sqsubseteq y \sqsupseteq f(y)\}$ is the left adjoint of $r\colon Pre(f) \to L$, defined as $f(x) = x$.

The Knaster-Tarski fixed-point theorem characterises μf as the least upper bound of all pre-fixed points of f and νf as the greatest lower bound of all its post-fixed points:

$$\mu f = \bigsqcap\{x \mid f(x) \sqsubseteq x\} \qquad \nu f = \bigsqcup\{x \mid x \sqsubseteq f(x)\}. \tag{1}$$

This immediately leads to the *induction* and *coinduction* proof principles.

$$\frac{\exists y,\ f(y) \sqsubseteq y \sqsubseteq x}{\nu f \sqsubseteq x} \qquad \frac{\exists y,\ x \sqsubseteq y \sqsubseteq f(y)}{x \sqsubseteq \nu f} \tag{2}$$

Another fixed-point theorem, usually attributed to Kleene, plays an important role in our exposition. It characterises μf and νf as the least upper bound, respectively

the greatest lower bound, of the chains

$$\bot \sqsubseteq f(\bot) \sqsubseteq ff(\bot) \sqsubseteq fff(\bot) \dots \qquad \top \sqsupseteq f(\top) \sqsupseteq ff(\top) \sqsupseteq fff(\top) \dots \tag{3}$$

In short,

$$\mu f = \bigsqcup_{i \in \omega} f^i(\bot) \qquad \nu f = \bigsqcap_{i \in \omega} f^i(\top) \tag{4}$$

The assumptions are stronger than for Knaster-Tarski: for the leftmost statement, it requires the map f to be *Scott-continuous* (i.e.., it preserves $\bigsqcup$ of directed chains) and, for the rightmost *Scott-cocontinuous* (similar but for $\bigsqcap$). Observe that every left adjoint is Scott-continuous and every right adjoint is Scott-cocontinuous.

Remark 1.1 (Duality) The two fixed-points theorems described above are usually formulated for the least fixed-point. The statements for the greatest fixed-point, on the right of (1) and (4), can be proved simply by *duality*: if a statement holds for an arbitrary complete lattice $(L, \sqsubseteq)$, it holds in particular also for its dual $(L, \sqsupseteq)$.

Categorical Abstraction 1.2. *Most of the theory developed in these notes can be extended from lattices to categories. In this perspective a lattice can be seen as a category, a monotone map as a functor, an up-closure operator as a monad and a down-closure operator as a comonad. Pre and post fixed point as algebras and coalgebras, least and greatest fixed point as the initial algebra and the final coalgebra.*

2 Coinduction for Determinitic Automata

The two theorems introduced above suggest two different algorithms to check language equivalence of deterministic automata. The Hopcroft algorithm [] can be regarded as a direct incarnation of the Kleene theorem. Knaster and Tarski, instead suggests a more basic algorithm which is an unoptimised version of the Hopcroft and Karp's algorithm [].

A deterministic automaton on the alphabet A is a pair $(X, \langle o, t \rangle)$, where X is a set of states and $\langle o, t \rangle \colon X \to 2 \times X^A$ is a function with two components: o, the output function, determines if a state x is final ($o(x) = 1$) or not ($o(x) = 0$); and t, the transition function, returns for each input letter $a \in A$ the next state.

Every automaton $(X, \langle o, t \rangle)$ induces a function $[\![-]\!] \colon X \to 2^{A^*}$ mapping each state of the automaton to the language that it accepts. Formally this function is defined for all $x \in X$, $a \in A$ and $w \in A^*$ as follows.

$$\begin{aligned} [\![x]\!](\varepsilon) &= o(x) \\ [\![x]\!](aw) &= [\![t(x)(a)]\!](w) \end{aligned}$$

Two states $x, y \in X$ are said to be *language equivalent*, in symbols $x \sim y$, iff $[\![x]\!] = [\![y]\!]$. Alternatively, language equivalence can be defined *coinductively* as the greatest fixed-point of a function b on Rel_X, the lattice of relations over X. For all $R \subseteq X^2$, $b \colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ is defined as

$$b(R) = \{(x, y) \mid o(x) = o(y) \text{ and for all } a \in A,\ (t(x)(a), t(y)(a)) \in R\}. \tag{5}$$

Indeed, one can check that b is monotone and that the greatest fixed-point of b, hereafter denoted by νb, coincides with $\sim$.

Hopcroft

(initialisation) $R_0 := \top$;
(iteration) $R_{i+1} := b(R_i)$;
(termination) **if** $R_{i+1} = R_i$, **then return** R_{i+1};

Fig. 1 Hopcroft algorithm to compute $\sim$ in a deterministic automaton $(X, \langle o, t\rangle)$.

Hopcroft algorithm is recalled in Figure 1. It takes in input a DA $(X, \langle o, t\rangle)$ and gives as output the relation $\sim$. If the set X contains n states, then the algorithm terminates in at most n iterations: the key observation is that all the R_i are *equivalence* relations. The correctness of the algorithm follows immediately from Kleene's theorem: each of the R_i is exactly $b^i(\top)$ and the Kleene chain stabilises after at most n iterations.

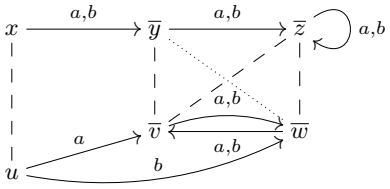
$$\top \supseteq b(\top) \supseteq bb(\top) \supseteq \cdots \supseteq b^i(\top) \supseteq \cdots \supseteq b^{n-1}(\top) \supseteq b^n(\top) = b^{n+1}(\top) = b^{n+2}(\top) \ldots$$

The Knaster-Tarski theorem suggests a different algorithm, based on the coinduction proof principle. For the convenience of the reader, theorem and proof principle are instantiated for b: $Rel_X \to Rel_X$ on the left and on the right below.

$$\nu b = \bigcup \{R \subseteq X^2 \mid R \subseteq b(R)\} \qquad \frac{\exists R,\ S \subseteq R \subseteq B(R)}{S \subseteq \nu B} \tag{6}$$

Hereafter we call a post-fixed point of b a *bisimulation*. The coinduction proof principle allows to prove $x_1 \sim x_2$ by exhibiting a bisimulation R such that $\{(x_1, x_2)\} \subseteq R$.

For an example of a bisimulation, consider the following deterministic automaton, where final states are overlined and the transition function is represented by labeled arrows. The relation consisting of dashed and dotted lines is a bisimulation witnessing, for instance, that $x \sim u$.



(7)

Figure 2 illustrates an algorithm, called **Naive**, that takes in input a DA $(X, \langle o, t\rangle)$ and a pair of states (x_1, x_2). It returns true if $x_1 \sim x_2$ and false otherwise. Note that this solves a rather different problem than the Hopcroft's algorithm which computes $\sim$ over all the states in X. In a nutshell, one can say that **Hopcroft** computes the largest bisimulation, while **Naive** the smallest bisimulation containing $\{(x_1, x_2)\}$.

The soundness of **Naive** can be easily proved by means of coinduction: observe that during the while loop **(3)** the invariant

$$R \subseteq b(R) \cup \mathit{todo}$$

always holds. The algorithm return true only if *todo* is empty and thus $R \subseteq b(R)$. This means that R is a bisimulation containing (x_1, x_2). By coinduction $x_1 \sim x_2$.

Naive (x_1, x_2)

```
(1) R := ∅; todo := ∅
(2) insert (x1, x2) into todo
(3) while todo is not empty do
   (3.1) extract (x1', x2') from todo
   (3.2) if (x1', x2') ∈ R then continue
   (3.3) if o(x1') ≠ o(x2') then return false
   (3.4) for all a ∈ A,
            insert (t_a(x1'), t_a(x2')) into todo
   (3.5) insert (x1', x2') into R
(4) return true
```

Fig. 2 Naive algorithm to check the equivalence of states $x_1, x_2 \in X$ for a deterministic automaton $(X, \langle o, t\rangle)$.

Question 2.1 The proof of *completeness* of **Naive** does not involve coinduction and seems to be somehow related to induction and abstract interpretation. One has to prove that the algorithm returns false only if $x_1 \not\sim x_2$. The key (inductive?) observation is that for every pair (x_1', x_2') extracted from *todo*, there exists a word $w \in A^*$ such that $x_1 \xrightarrow{w} x_1'$ and $x_2 \xrightarrow{w} x_2'$. Now, the algorithm returns false only if $o(x_1') \neq o(x_2')$ which means $[\![x_1]\!](w) \neq [\![x_2]\!](w)$. Is there a nicer way to prove this by induction? Or by means of abstract interpretation?

The algorithm **Naive** requires at most $1 + |A| \times |R|$ iterations, where $|A|$ is the size of the alphabet A and $|R|$ the size of the returned relation R. Therefore, if X has n states, the algorithm has worst case complexity $\mathcal{O}(n^2)$. This is much worst than the Hopcroft's algorithm but, as we will seen in Section 3.1, this can be improved by means of up-to techniques. Most importantly, when it comes to check language equivalence for non-deterministic automata, the combined use of up-to techniques and **Naive** leads to the most efficient algorithm [2].

3 Coinduction up-to

Let $b\colon C \to C$ be a (monotone) map on a complete lattice C. We adopt this notation, since in the abstract interpretation perspective C will represent the concrete domain. The map b, like in the case of automata, defines our object of interest νb. More generally, we are interested in proving that $x \sqsubseteq \nu b$ for a given element x of the lattice.

An *up to technique* is a monotone map $a\colon C \to C$. A *bisimulation up to* a is a post-fixed point of ba, that is an x such that $x \sqsubseteq ba(x)$. An up-to technique a is said to be *sound* for b if the following *coinduction up to* principle holds.

$$\frac{\exists y,\ x \sqsubseteq y \sqsubseteq ba(y)}{x \sqsubseteq \nu b} \tag{8}$$

Remark 3.1 (completeness of up-to technique) Observe that, according to the above definition an up-to technique a might not be *complete*: it may exists an x such that $x \sqsubseteq \nu b$ for which there is no y satisfying $x \sqsubseteq y \sqsubseteq ba(y)$. However, if a is an up-closure operator, then $\nu b \sqsubseteq a(\nu b)$ and using monotonicity of b, one obtains that $x \sqsubseteq$

$\nu b = b(\nu b) \sqsubseteq b(a(\nu b))$. The question of completeness for up-to techniques has never been raised because they have always been thought as up-closure operators: e.g., up-to equivalence, up-to congruence, up-to bisimilarity. The only reason for considering arbitrary monotone maps rather than just up-closure operators, comes from the fact that the former allows for more modular proofs of their soundness. This is discussed in more details in Section 7.

ToDo 3.2. *Check whether in the original work of Milner up to techniques are defined as closure operators or monotone maps.*

3.1 Hopcroft and Karp's algorithm

The famous algorithm by Hopcroft and Karp for checking language equivalence [4] relies on coinduction implicitly, long before Milner's pioneering work on bisimulation. Hopcroft and Karp actually use coinduction up to equivalence closure. Consider the function $e\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ mapping every relation $R \subseteq X^2$ to its equivalence closure. A *bisimulation up to* e is a relation R such that

$$R \subseteq b(e(R)).$$

For example, consider the automaton in (7) and the relation R containing only the dashed lines: since $t(x)(b) = y$, $t(u)(b) = w$ and $(y, w) \notin R$, then $(x, u) \notin b(R)$. This means that R is *not* a bisimulation; however it is a bisimulation up to e, since (y, w) belongs to $e(R)$ and (x, u) to $b(e(R))$.

In general, bisimulations up-to can be smaller than plain bisimulation and this feature can have a relevant impact in the performance of algorithms for checking language equivalence. The Hopcroft and Karp's algorithm is defined like `Naive`, but replacing line `(3.2)` with the following.

`(3.2)` `if` $(x_1', x_2') \in e(R)$ `then continue`

This simple optimisation allows to reduce the worst case complexity of `Naive`: the size of the returned relation R cannot be larger than n (the number of states).

The soundness of this algorithm can be proved similarly to the one for `Naive`: during the while loop `(3)`, the invariant

$$R \subseteq be(R) \cup todo$$

always holds. The algorithm returns true only if $R \subseteq be(R)$. This means that R is a bisimulation up to e containing (x_1, x_2). If e is a sound for b then, by (8), one can conclude that $x_1 \sim x_2$.

Proving the soundness of up to techniques might be rather complicated and error prone []. Most of the theory of coinduction up to focuses on finding good methods to prove soundness. In Section 6, we will illustrate the resulting theory and show that is closely related to the problem of completeness for abstract domain. For convenience of the reader, we report in Appendix A some further examples of applications of up to techniques in automata theory.

3.2 Induction (up to) for Deterministic Automata

As already mentioned in Section 2, the `Naive` algorithm builds the smallest bisimulation containing the initial pair of states $\{(x_1, x_2)\}$. Rather than as the computation of a greatest fixed-point, `Naive` can be regarded as the computation of a least fixed-point. Let $b^* \colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ be the function mapping every relation $R \in \mathsf{Rel}_X$ into

$$b^*(R) = \{(x_1', x_2') \mid \exists (x_1, x_2) \in R, a \in A \text{ s.t. } t(x_1)(a) = x_1' \text{ and } t(x_2)(a) = x_2'\} \quad (9)$$

and $reach_{x_1,x_2} \colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ be defined as

$$reach_{x_1,x_2}(R) = b_1^*(R) \cup \{(x_1, x_2)\}.$$

The least fixed-point of $reach_{x_1,x_2}$ is the set of all pairs of states reachable from (x_1, x_2). Let us call f the relation in Rel_X defined as

$$f = \{(x_1, x_2) \mid o(x_1) = o(x_2)\}. \quad (10)$$

Then the problem of language equivalence can be rephrased as a problem of induction:

$$x_1 \sim x_2 \text{ iff } \mu(reach_{x_1,x_2}) \subseteq f.$$

Now the soundness of `Naive` can be proved by means of induction: at any iteration of the while loop `(3)` the invariants

$$reach_{x_1,x_2}(R) \subseteq R \cup todo \qquad \text{and} \qquad R \subseteq f$$

hold. The algorithm returns true only if *todo* is empty, so that $reach_{x_1,x_2}(R) \subseteq R \subseteq f$. By the induction proof principle (on the left of (6)) we have that $\mu(reach_{x_1,x_2}) \subseteq f$.

Question 3.3 Can we elegantly prove completeness using an inductive approach?

The story of induction for determinitic autoamaton ends here. Up-to techniques do not work for induction in this setting. The dual of the coinduction up to principle in (8), the *induction up to principle*, is given below.

$$\frac{\exists y,\ ba(y) \sqsubseteq y \sqsubseteq x}{\mu b \sqsubseteq x} \quad (11)$$

By taking $a = e$ and $b = reach_{x_1,x_2}$, the invariant

$$reach_{x_1,x_2}(e(R)) \subseteq R \cup todo$$

does *not* hold in the Hopcroft and Karp's algorithm. Intuitively, when using induction up to a, one would like a to shrink relations rather than enlarging. Indeed, as explained in the following remark, induction up to makes sense when a is a down-closure.

Remark 3.4 (Completeness of induction up-to) As for coinduction up-to, one may wonder about the completeness of induction up-to. Completeness holds whenever a is a down-closure. Indeed $a(\nu b) \sqsubseteq \nu(b)$ and, by monotonicity of b, one obtains $ba(\nu b) \sqsubseteq b\nu b = \nu b$.

Question 3.5 Researchers interested in coinduction up-to have been wondering for a while about induction up-to. As we will better explain later, coinduction up to a can be seen in a category of coalgebras over the category of Eilenberg-Moore algebras for the monad a. Its dual, induction up to a, can be regarded in a category of algebras over the category of Eilenberg-Moore coalgebras for the comonad a. While the former are common places in computer science (determinised automata, process calculi and so on), the latter does not show off much. What about physics? Can we find an interesting example? More particularly, can we make a nice (and reasonable) proof by means of induction up to?

4 Abstract Interpretation

Like for coinduction up to, the starting data for abstract interpretation are two monotone maps $a, b \colon C \to C$. But now, here, it is important that a is an up-closure. As pointed out in Remark 3.1, this is morally the case also for up-to techniques, but for making soundness proofs modular is often convenient to require a to be just a monotone map.

We denote by $(\alpha \dashv \gamma) \colon C \to A$ the pair of adjoints associated with a. Intuitively, C represents a *concrete domain* of data, A an *abstract domain*, $\alpha \colon C \to A$ the *abstraction* function and $\gamma \colon A \to C$ the *concretisation*. The intuition for the ordering is that if $x \sqsubseteq y$, then y contains less information than x: for $c \in C$ and $a \in A$, $\alpha(c) \sqsubseteq_A a$ (or equivalently $c \sqsubseteq_C \gamma(a)$) means that a is a correct approximation of c.

In a nutshell, the main idea of abstract interpretation is to have a map $\overline{b} \colon A \to A$ representing the program b but on the abstract domain A. *Soundness* of $\overline{b}$ (w.r.t. b) means that

$$\alpha(\mu b) \sqsubseteq_A \mu\overline{b}$$

completeness means that

$$\alpha(\mu b) = \mu\overline{b}. \tag{12}$$

Observe that, while soundness is equivalent to $\mu b \sqsubseteq_A \gamma(\mu\overline{b})$, completeness does *not* coincide with $\mu b = \gamma(\mu\overline{b})$.

Soundness is usually proved by a stronger property which is much easier to check:

$$\alpha \circ b \sqsubseteq \overline{b} \circ \alpha, \qquad \text{in diagram} \qquad \begin{array}{ccc} A & \xrightarrow{\overline{b}} & A \\ {\scriptstyle\alpha}\uparrow & \swarrow & \uparrow{\scriptstyle\alpha} \\ C & \xrightarrow[b]{} & C \end{array}$$

Once a pair of adjoints $(\alpha \dashv \gamma) \colon C \to A$ is fixed (or equivalently an abstract domain), one can always build a sound $\overline{b}$ as

$$\overline{b} = \alpha \circ b \circ \gamma.$$

Indeed, $\alpha b \sqsubseteq \alpha b \gamma \alpha = \overline{b}\alpha$. This is not the case for completeness: not for all abstract domains and maps b, there exists a complete $\overline{b}$. However, if it exists, this is exactly $\alpha \circ b \circ \gamma$. Hereafter, we assume to have given a and b and we focus on the problem of completeness. We will see that this is intimately related to the problem of soundness for up-to techniques.

Remark 4.1 In Remark 3.1, we observed that given a and b, the problem of completeness for up-to a is trivial whenever a is an up-closure. The above discussion shows that for abstract interpretation, it is the problem of soundness to be trivial.

5 Backward completeness

Proving completeness of an abstract domain is usually rather complicated. In order to simplify these proofs, Cousot introduced in [] the notion of full-completeness, which we will refer hereafter as backward completeness. An up-closure $a\colon C \to C$ is *backward complete* w.r.t. $b\colon C \to C$ iff

$$ab = aba \tag{13}$$

Lemma 5.1 *Let $b\colon C \to C$ be a monotone map and $a\colon C \to C$ an up-colure operator with $(\alpha \dashv \gamma)\colon C \to A$ as associated pair of adjoints. The followings are equivalent*

1. *a is backward complete w.r.t. b: $ab = aba$;*
2. *there exists a Kleisli law: $ba \sqsubseteq ab$;*
3. *there exists a Kleisli extension $\overline{b}\colon A \to A$: $\overline{b}\alpha = \alpha b$.*

$$\begin{array}{ccc} A & \xrightarrow{\overline{b}} & A \\ {\scriptstyle\alpha}\uparrow & & \uparrow{\scriptstyle\alpha} \\ C & \xrightarrow[b]{} & C \end{array}$$

Proof. The proof is trivial. We report it just for the sake of completeness. $(1 \Rightarrow 2)$ $ab = aba \sqsupseteq ba$ since a is an up-closure. $(2 \Rightarrow 3)$ It always holds $(\alpha b\gamma)\alpha \sqsupseteq \alpha b$. For the other inclusion, observe that if $ba \sqsubseteq ab$, then $b\gamma\alpha \sqsubseteq \gamma\alpha b$ and $\alpha b\gamma\alpha \sqsubseteq \alpha\gamma\alpha b \sqsubseteq \alpha b$. $(3 \Rightarrow 1)$ Since $(\alpha b\gamma)\alpha = \alpha b$, then $\gamma(\alpha b\gamma)\alpha = \gamma\alpha b$, that is $aba = ab$. □

Proposition 5.2 (From Cousot POPL 1977 []) *Let $b\colon C \to C$ be a Scott-continuous map and $a\colon C \to C$ an up-closure operator. If a is backward complete, then it is complete.*

Proof. The assumption of Scott-continuity is necessary to characterise

$$\alpha(\mu b) = \alpha(\bigsqcup_n b^n(\bot_C)) \qquad \text{and} \qquad \mu(\alpha b\gamma) = \bigsqcup_n (\alpha b\gamma)^n(\bot_A).$$

Since α is a left adjoint we have that the leftmost is equivalent to $(\bigsqcup_n \alpha b^n(\bot_C)$.

By induction on n, we prove that $\alpha b^n(\bot_C) = (\alpha b\gamma)^n(\bot_A)$.

- For $n = 0$, $\alpha(\bot_C) = \bot_A$;
- For $n + 1$, we have that $\alpha b\gamma(\alpha b\gamma)^n(\bot_A) = \alpha b\gamma\alpha b^n(\bot_C)$ by induction hypothesis. Using the property of Kleisli lifting, the latter is equivalent to $\alpha b b^n(\bot_C) = \alpha b^{n+1}(\bot_C)$.

□

ToDo 5.3. *Carefully compare with the proof of Cousot.*

Question 5.4 Is there a nicer proof which does not use the assumption of Scott continuity?

6 Forward completeness and compatible techniques

Another condition, which is called in [] forward completeness, plays a crucial role for our exposition. An up-closure $a\colon C \to C$ is *backward complete* w.r.t. $b\colon C \to C$ iff

$$ba = aba \tag{14}$$

Lemma 6.1 *Let $b\colon C \to C$ be a monotone map and $a\colon C \to C$ an up-colure operator with $(\alpha \dashv \gamma)\colon C \to A$ as associated pair of adjoints. The followings are equivalent*

1. *a is forward complete w.r.t. b, i.e., $ba = aba$;*
2. *there exists a EM law, i.e., $ab \sqsubseteq ba$;*
3. *there exists a EM lifting $\overline{b}\colon A \to A$, i.e., $\gamma\overline{b} = b\gamma$.*

$$\begin{array}{ccc} A & \xrightarrow{\overline{b}} & A \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\gamma} \\ C & \xrightarrow[b]{} & C \end{array}$$

Proof. The proof is trivial. We report it just for the sake of completeness. ($1 \Rightarrow 2$) Since $b \circ a = a \circ b \circ a$ then $b \circ a = a \circ b \circ a \sqsupseteq a \circ b$. ($2 \Rightarrow 1$) Since $a \circ b \sqsubseteq b \circ a$, then $a \circ b \circ a \sqsubseteq b \circ a \circ a \sqsubseteq b \circ a$. The other inclusion, $b \circ a \subseteq a \circ b \circ a$, holds since a is an up-closure. ($2 \Rightarrow 3$) Observe that $A = Pre(a)$ by construction. For every pre fixed-point $a(x) \sqsubseteq x$, it holds that $ab(x) \sqsubseteq ba(x) \sqsubseteq b(x)$, namely $b(x)$ is a pre fixed-point. One can therefore define $\overline{b}(x) = b(x)$. The fact that $\gamma\overline{b} = b\gamma$ follows immediately by construction of γ. ($3 \Rightarrow 2$) By construction of γ, for every pre fixed point x, $b(x)$ is forced to be a pre fixed-point of a: $ab(x) \sqsubseteq b(x)$. Therefore $ab(x) \sqsubseteq b(x) \sqsubseteq ba(x)$. □

Remark 6.2 From the proof of the above lemma, one can see that the assumption of a being a up-closure operator is necessary only for the first point. The correspondence of the last two points holds also when a is not a closure operator: in this case, A is the lattice of pre fixed-point of a and γ the obvious injection of A in C. In this case, its left adjoint α does not exist.

Proposition 6.3 *Let $a, b\colon C \to C$ be two monotone maps so that a is backward complete w.r.t. b. Let $\overline{b}\colon A \to A$ be the corresponding EM lifting. Then $\gamma(\nu\overline{b}) = \nu b$.*

Proof. Recall that $A = Pre(a)$. By the Knaster-Tarski fixed-point theorem $\gamma(\nu\overline{b}) = \bigsqcup\{x \mid a(x) \sqsubseteq x \sqsubseteq b(x)\}$ and $\nu b = \bigsqcup\{x \mid x \sqsubseteq b(x)\}$. Since $\{x \mid a(x) \sqsubseteq x \sqsubseteq b(x)\} \subseteq \{x \mid x \sqsubseteq b(x)\}$, then $\gamma(\nu\overline{b}) \sqsubseteq \nu b$. Observe that $a(\nu b) = ab(\nu b) \sqsubseteq ba(\nu b)$, namely $a(\nu b)$ is a post fixed-point of b. Therefore $a(\nu b) \sqsubseteq \nu b \sqsubseteq b(\nu b)$, which means $\nu b \in \{x \mid a(x) \sqsubseteq x \sqsubseteq b(x)\}$. This proves that $\nu b \sqsubseteq \gamma(\nu\overline{b})$. □

Exactly the same condition of backward completeness has been introduced also for up-to techniques, under the name of compatibility. An up-to technique $a\colon C \to C$ is said to be *compatible* w.r.t. b iff

$$ab \sqsubseteq ba$$

Observe that this is just the definition of EM-law and, therefore, when a is an up-closure coincides exactly with backward completeness.

Like forward completeness entails completeness for abstract interpretation, compatibility entails soundness for up-to techniques.

Proposition 6.4 *Let a be compatible w.r.t. b. Then a is sound w.r.t. b.*

Proof. For each $y \sqsubseteq ba(y)$, $a(y) \sqsubseteq aba(y) \sqsubseteq baa(y) \sqsubseteq ba(y)$. Therefore $a(y) \sqsubseteq \nu b$. If $x \sqsubseteq y$, then $x \sqsubseteq a(y) \sqsubseteq \nu b$. □

Backward completeness instead does not entail (fixpoint) completeness for abstract interpretation. We will discuss this further in Section **??**.

Categorical Abstraction 6.5. *Observe that we have called the conditions $ba \sqsubseteq ab$ and $ab \sqsubseteq ba$ Kleisli law and EM law. Indeed, one can think to the problem of completeness of abstract interpretation and soundness of up-to techniques as the problem of* extending *and* lifting *the functor $b\colon C \to C$ to some functor $\overline{b}$ either on the Kleisli category $Kl(a)$ or to the Eilenberg Moore category $EM(a)$ of algebras for the monad $a\colon C \to C$. In this case, since C is a lattice, one has that $Kl(a) = EM(a) = A$. In this perspective, completeness of full abstraction means that there is a functor $\alpha\colon Alg(b) \to Alg(\overline{b})$ preserving initial algebra (this is entailed by requiring α to be a left adjoint). Similarly, soundness of up-to techniques means that there is a functor $\gamma\colon Coalg(\overline{b}) \to Coalg(b)$ that preserves the final coalgebra (this is entailed by requiring γ to be a right adjoint). The latter is rather well-known problem (see e.g., nondeterministic automata as coalgebra, and GSOS specification). The latter is far less understood: see the attached scribble by Dusko.*

Abstract Interpretation	*Coinduction up-to*
Kleisli law $ba \sqsubseteq ab$	*EM-law* $ab \sqsubseteq ba$
Kleisli Extension $\overline{b}\colon Kl(a) \to Kl(a)$	*EM lifting* $\overline{b}\colon EM(a) \to EM(a)$
$\alpha\colon Alg(b) \to Alg(\overline{b})$ *is a left adjoint*	$\gamma\colon Coalg(\overline{b}) \to Coalg(b)$ *is a right adjoint*

7 Modularity

Before focusing on our (so far) main achievement, we make a short detour, explaining one of the side result of our comparison.

We have seen that up-to techniques are morally up-closure, but they are usually defined as monotone maps. The reason for this definition is that one can define operations on functions, in a much straightforward way, e.g., both $f \sqcup g$ and $f \sqcap g$ are defined pointwise for all maps $f, g\colon C \to C$, while the pointwise joint of two up-closures is not necessarily an up-closure. The following proposition allows to make modular proof of compatibility or, in term of abstract interpretation, backward completeness.

Proposition 7.1 (modularity) *Let $a, b, a_1, a_2, b_1, b_2\colon C \to C$ be monotone maps on some complete lattice C. Then:*

1. *$id \circ b \sqsubseteq b \circ id$;*
2. *if $a_1 \circ b \sqsubseteq b \circ a_1$ and $a_2 \circ b \sqsubseteq b \circ a_2$, then $(a_1 \circ a_2) \circ b \sqsubseteq b \circ (a_1 \circ a_2)$.*

Moreover:

3. *if $a_1 \circ b \sqsubseteq b \circ a_1$ and $a_2 \circ b \sqsubseteq b \circ a_2$, then $(a_1 \sqcup a_2) \circ b \sqsubseteq b \circ (a_1 \sqcup a_2)$;*
4. *if $a \circ b \sqsubseteq b \circ a$, then $a^{\uparrow} \circ b \sqsubseteq b \circ a^{\uparrow}$.*

Dually:

5. *if $a \circ b_1 \sqsubseteq b_1 \circ a$ and $a \circ b_2 \sqsubseteq b_2 \circ a$, then $a \circ (b_1 \sqcap b_2) \sqsubseteq (b_1 \sqcap b_2) \circ a$;*

6. if $a \circ b \sqsubseteq b \circ a$, then $a \circ b^{\downarrow} \sqsubseteq b^{\downarrow} \circ a$.

ToDo 7.2. *The proof is simple but should be written down.*

This proposition has been exploited extensively for up-to techniques. We illustrates its importance below.

7.1 Back to Hopcroft and Karp: proving soundness of equivalence closure.

Recall the monotone map $b\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ defined in (5) and the up-closure $e\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ introduced in Section 3.1. In order to prove that the Hopcroft and Karp algorithm is sound one has to relies on the fact that e is sound w.r.t. b. Thanks to Proposition 6.4, one can prove soundness by showing that e is compatible w.r.t. b. The proof of compatibility can be simplified by mean of Proposition 7.1.

For every relation $R \subseteq X \times X$, define $r, s, t\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ as follows.

$$r(R) = \{(x,x) \mid x \in X\} \qquad s(R) = \{(y,x) \mid (x,y) \in R\}$$

$$t(R) = \{(x,z) \mid \exists y \text{ s.t. } (x,y) \in R \text{ and } (y,z) \in R\}$$

Observe that r is a constant function, mapping every relation into the identity relation. Intuitively $r \sqcup id$ is the reflexive closure. Similarly s maps a relation R into ints inverse R^{-1}: again $s \sqcup id$ is the symmetric closure. The function t is called the squaring function: to obtain the transitive closure one has to take $t^{\uparrow}$. Now, the equivalence closure can be decomposed as

$$e = (id \sqcup r \sqcup s \sqcup t)^{\uparrow}$$

In order to prove compatibility of e, it is enough to prove, thank to Proposition 7.1, compatibility of the much simpler functions r, s, t. We leave this simple task to the reader.

Remark 7.3 (Modularity for forward completeness) As the notions of forward completeness and compatibility coincide, one can use Proposition 7.1 also for making modular proofs of forward completeness. Observe that, by point 4. of Proposition 7.1, one can always obtain an up-closure operator. For instance suppose to have two up-closure operator a_1 and a_2 that are both forward complete. Then $a_1 \sqcup a_2$ is still forward complete, but it is not an up-closure. To have an up-closure, one takes $(a_1 \sqcup a_2)^{\uparrow}$ which is ensured to be forward complete by points 3. and 4. of Proposition 7.1.

ToDo 7.4. *Modularity for forward completeness for abstract domains has been studied quite a lot. We should check whether this new perspective really brings something interesting.*

8 Relating Abstract Interpretation and Coinduction up-to by adjointness

To relate abstract interpretation and coinduction up-to we need to assume that the monotone map $b\colon C \to C$ is part of an adjunction. Hereafter we denote the adjunction by $b^* \dashv b_*\colon C \to C$. Our idea is based on the observation in [] relating backward and forward completeness:

$$ab^* = ab^*a \quad \text{iff} \quad b_*a = ab_*a.$$

We find convenient to rephrase this as

$$b^* a \sqsubseteq a b^* \quad \text{iff} \quad a b_* \sqsubseteq b_* a. \tag{15}$$

Note however that since b^* is a left adjoint and b_* is a right adjoint one always has that $\mu b^* = \bot$ and $\nu b_* = \top$, meaning that the abstract interpretation and coinduction up-to are always rather trivial.

We then consider some *intial* and *final* conditions $i, f \in C$. With an abuse of notation, we will use this letter also to denote the constant functions $i \colon C \to C$ and $f \colon C \to C$ mapping all elements to, respectively, i and f. We consider the problem of completeness of a w.r.t. $b^* \sqcup i$ and the problem of soundness w.r.t. $b_* \sqcap f$.

The former is entailed by backward completeness

$$(b^* \sqcup i)a \sqsubseteq a(b^* \sqcup i)$$

which, by Proposition 7.1.3 is entailed by

$$b^* a \sqsubseteq a b^* \quad \text{and} \quad i \sqsubseteq a i. \tag{16}$$

Soundness is entailed by forward completeness (or compatibility)

$$a(b_* \sqcap f) \sqsubseteq (b_* \sqcap f)a.$$

which, by Proposition 7.1.5 is entailed by

$$a b_* \sqsubseteq b_* a \quad \text{and } a f \sqsubseteq f. \tag{17}$$

Observe that there is a slight asymmetry in (16) and (17): the condition $i \sqsubseteq ai$ is guaranteed for any $i \in C$, since a is an up-closure. This is not the case for $af \sqsubseteq f$.

Therefore the data that allows to link abstract interpretation and coinduction up-to are:

- $b^* \dashv b_* \colon C \to C$,
- $a \colon C \to C$,
- $i \in C$,
- $f \in C$ such that $af \sqsubseteq f$.

In this setting, by mean of (15), a is forward complete w.r.t. b^* iff a is backward complete w.r.t. b_*. These entails both that a is forward complete w.r.t. $b^* \sqcup i$ and a is backward complete w.r.t. $b_* \sqcup f$.

Remark 8.1 Apparently, we do not have an iff between the conditions of a is forward complete w.r.t. $b^* \sqcup i$ and a is backward complete w.r.t. $b_* \sqcup f$.

ToDo 8.2. *Please check carefully the iff in the remark.*

8.1 Hopcroft and Karp as complete abstract interpretation

We now show how the Hopcroft and Karp algorithm can be seen as an instance of complete abstract interpretation, using the technology developed above.

First of all observe that $b\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ in (5) can be decomposed as

$$b = b_* \sqcap f$$

where $b_*\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ is defined for all relations R as

$$b_*(R) = \{(x,y) \mid \text{ for all } a \in A,\ (t(x)(a), t(y)(a)) \in R\} \tag{18}$$

and $f\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ maps everything to the relation f defined in (10). Now recall the up-closure $e\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ mapping each relation in its equivalence closure. It is immediate to see that $f \in \mathsf{Rel}_X$ is an equivalence relation, that is $e(f) \subseteq f$.

The function b_* has a left adjoint, which is exactly b^* as defined in (9). We take i to be the singleton relation containing the pair of initial states (x_1, x_2).

These are all the data needed to link coinduction up-to and abstract interpretation. Indeed, the problem of language equivalence for x_1 and x_2 can be regarded both in an inductive and a coinductive way:

$$\mu(b^* \cup i) \subseteq f \qquad \text{iff} \qquad i \subseteq \nu(b_* \cap f).$$

We have shown that the righmost inclusion can be checked using coinduction up-to e. To prove soundness of this technique we needed to prove that e is compatible w.r.t. $b = (b_* \cap f)$. One can easily check that e is also compatible w.r.t. b_*.

Remark 8.3 Unfortunately here, we cannot say that the fact that e is compatible w.r.t. $b = (b_* \cap f)$ entails that e is compatible w.r.t. b_*. the other implication hold, but not this one. Indeed:

$$e(b_* \cap f) \subseteq (b_* \cap f)e \text{ iff } eb_* \cap ef \subseteq b_*e \cap f \text{ iff } eb_* \cap ef \subseteq b_*e \text{ and } eb_* \cap ef \subseteq f.$$

ToDo 8.4. *Understand this remark better.*

ToDo 8.5. *The problem in the remark above can be done more smooth by using a different proof strategy. We decompose $b = b = (b_* \cap f)$ already in Section 7.1 and we use modularity w.r.t. $\cap$ to prove compatibility of e. This would look more natural in a final version of the paper.*

Now, since e is compatible w.r.t. b_*, we have by (15) that it is forward complete w.r.t. b^* and, by the fact that $i \subseteq e(i)$ and Proposition 7.1.5, also w.r.t. $b^* \cup i$. By Proposition 5.2, e is a complete domain for abstract interpretation.

This provides a novel perspective on the Hopcroft and Karp's algorithm. It can be though as the computation of the least fixed-point of the function $b^* \cup i$ abstracted to the lattice of equivalence relations $ERel_X$. We denote this function by $\overline{b^* \cup i}\colon ERel_X \to ERel_X$ and $\alpha \dashv \gamma\colon Rel_X \to ERel_X$ the pair of adjoint associated to e. The map α assign to every relation its equivalence closure; γ is just the obvious injection. The function $\overline{b^* \cup i}$ is just $\alpha \circ (b^* \cup i) \circ \gamma$: it basically takes an equivalence relation, computes $b^* \cup i$ and then makes its equivalence closure. The algorithm returns true iff $\mu(\overline{b^* \cup i}) \subseteq f$. Soundness is trivial: if $\mu(\overline{b^* \cup i}) \subseteq f$, then

$$\mu(b^* \cup i) \subseteq \gamma\alpha\mu(b^* \cup i) \subseteq \gamma\mu(\overline{b^* \cup i}) \subseteq \gamma f = f.$$

Completeness informs us that $\mu(\overline{b^* \cup i}) = \alpha\mu(b^* \cup i)$: if $\mu(b^* \cup i) \subseteq f$, then

$$\mu(\overline{b^* \cup i}) = \alpha\mu((b^* \cup i)) \subseteq \alpha(f) = f.$$

8.2 Fixed-point completeness

So far, we have established a link between the sufficient conditions (see Table 1), namely between forward completeness of an abstract domain and compatibility of an up-to technique. Is it the case that one can establish a link also the final aims, that is completeness of an abstract domain and soundness of an up-to technique?

Hereafter, we reformulate the question in purely lattice-theoretic terms.

Given,

- an adjunction of monotone maps $b^* \dashv b_* \colon C \to C$,
- an up-closure $a \colon C \to C$, with corresponding pair of adjoints $\alpha \dashv \gamma$,
- an element $i \in C$,
- a pre-fixpoint $f \in C$, i.e., an element such that $af \sqsubseteq f$.

Does it hold that

$$\alpha(\mu(b^* \sqcup i)) = \mu(\alpha \circ (b^* \sqcup i) \circ \gamma) \qquad \text{iff} \qquad \nu((b \sqcap f)a) = \nu(b \sqcap f) \qquad ?$$

We have explored this in a rather extensive way without finding either a proof or a counterexample. Some hints could come by assuming the lattice C to be boolean. Some partial but interesting results are in the attached notes scanned by Roberto.

9 Most abstract Complete domain and companion

ToDo 9.1. *This is a completely unexplored field where I can we can say a lot. We should look at really carefully.*

10 Side tracks

We conclude these notes with some ideas that emerged during the discussion, but that does not seem to play a pivotal role in the actual development.

10.1 Up-to techniques for induction and coinduction

Hereafter we compare soundness and completeness of induction up-to (on the left) and coinduction up-to (on the right) for two monotone maps $a, b \colon C \to C$.

$$\frac{\exists y,\ ba(y) \sqsubseteq y \sqsubseteq x}{\mu b \sqsubseteq x} \qquad\qquad \frac{\exists y,\ x \sqsubseteq y \sqsubseteq ba(y)}{x \sqsubseteq \nu b} \tag{19}$$

We distintiguish two important cases, when a is an up-closure and when it is a down-closure. Apart from the cases marked with ?, for which we have not found yet a simple condition (and probably we will never find), all the proofs are given in the lemmas below. The missing cases are obtained by duality.

		Induction	Coinduction
a up-closure	Soundness	Trivial	$ab \sqsubseteq ba$
	Completeness	?	Trivial
a down-closure	Soundness	$ba \sqsubseteq ab$	Trivial
	Completeness	Trivial	?

Lemma 10.1 *Let a be an up-closure. Then coinduction up-to a is complete.*

Proof. Take $y = \nu b$. Since, a is an up-closure $\nu b \sqsubseteq a\nu b$. Then $x \sqsubseteq \nu b = b\nu b \sqsubseteq ba\nu b$. □

Lemma 10.2 *Let a be an up-closure. If $ab \sqsubseteq ba$, then coinduction up-to a is sound.*

Proof. Note that the statement is exactly the same of Proposition 6.4. We copied here once more for the convenience of the reader. □

Lemma 10.3 *Let a be an up-closure. Then induction up-to a is sound.*

Proof. Since, a is an up-closure, then $b(y) \sqsubseteq ba(y) \sqsubseteq y$. Therefore $\mu b \sqsubseteq y \sqsubseteq x$. □

10.2 A principle for abstract intepretation

The following principle, where $x', y' \in A$, coincides with abstract interpretation.

$$\frac{\exists y',\ \alpha b\gamma(y') \sqsubseteq y' \sqsubseteq x'}{\alpha(\mu b) \sqsubseteq x'} \tag{20}$$

We first prove that soundness of this principle holds iff $\alpha(\mu b) \sqsubseteq \mu(\alpha b\gamma)$. We have already seen that $\alpha(\mu b) \sqsubseteq \mu(\alpha b\gamma)$ is always true. We prove that from it, it follows the soundness of the proof principle observe that if the premises are true then $\alpha(\mu b) \sqsubseteq \mu(\alpha b\gamma) \sqsubseteq y' \sqsubseteq x'$.

The completeness is the interesting part. Assume that the rule is complete and take $x' = \alpha(\mu b)$. Then there exists $y' \sqsubseteq \alpha(\mu b)$ which is a prefix-point of $\alpha b\gamma$. Therefore $\mu(\alpha b\gamma)y' \sqsubseteq \alpha(\mu b)$. The other inclusion is always true. Now assume that $\mu(\alpha b\gamma)y' = \alpha(\mu b)$ and that the conclusion of the rule holds. Therefore $\alpha b\gamma(\mu(\alpha b\gamma)) = \mu(\alpha b\gamma) = \alpha(\mu b)x'$.

References

1. Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. In: POPL, pp. 229–239. ACM (1988). DOI 10.1145/73560.73580. URL http://doi.acm.org/10.1145/73560.73580
2. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: POPL, pp. 457–468. ACM (2013). URL http://doi.acm.org/10.1145/2429069.2429124
3. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM **11**(4), 481–494 (1964)
4. Hopcroft, J.E., Karp, R.M.: A linear algorithm for testing equivalence of finite automata. Tech. Rep. 114, Cornell Univ. (1971). URL http://techreports.library.cornell.edu:8081/Dienst/UI/1.0/Display/cul.cs/TR71-114
5. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. In: Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991, pp. 214–225 (1991). DOI 10.1109/LICS.1991.151646. URL http://dx.doi.org/10.1109/LICS.1991.151646
6. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
7. Rot, J.: Enhanced coinduction. Ph.D. thesis, Leiden University (2015)

A More examples of up-to techniques

For the convenience of the reader, we report below two further examples of coinduction up-to published by the first author somewhere else.

A.1 Regular Expressions and Kleene Algebra

Beyond algorithms, up-to techniques are useful to prove different sorts of properties of systems specified by a given syntax. Indeed, this was the original motivation for the introduction of up-to techniques in Milner's work on CCS [6]. To keep the presentation simpler and, at the same time, to show to the reader the large spectrum of applications of up-to techniques, we consider *regular expressions* and we provide coinductive proofs for some of the axioms of Kleene Algebra [5] with respect to the regular language interpretation.

First, recall that regular expressions are generated by the following grammar

$$e ::= 0 \mid 1 \mid a \mid e + e \mid e \cdot e \mid e^\star$$

where a ranges over symbols of the alphabet A. To make the notation lighter we will often avoid to write $\cdot$, so that ef stands for $e \cdot f$.

We will prove language equivalence of regular expressions by considering bisimulations on an automaton having as state space the set RE of regular expressions. This automaton is constructed using Brzozowski derivatives [3]. The following inference rules

$$\frac{-}{1\!\downarrow} \qquad \frac{e\!\downarrow}{(e+f)\!\downarrow} \qquad \frac{f\!\downarrow}{(e+f)\!\downarrow} \qquad \frac{e\!\downarrow\ f\!\downarrow}{(e\cdot f)\!\downarrow} \qquad \frac{-}{e^\star\!\downarrow}$$

define the output function $o\colon RE \to 2$ as $o(e) = 1$ iff $e\!\downarrow$. The following inference rules

$$\frac{-}{0 \xrightarrow{a} 0} \qquad \frac{-}{1 \xrightarrow{a} 0} \qquad \frac{-}{a \xrightarrow{a} 1} \qquad \frac{b \neq a}{a \xrightarrow{b} 0} \qquad \frac{e \xrightarrow{a} e' \quad f \xrightarrow{a} f'}{e + f \xrightarrow{a} e' + f'}$$

$$\frac{e \xrightarrow{a} e' \quad f \xrightarrow{a} f'}{e \cdot f \xrightarrow{a} e' \cdot f + o(e) \cdot f'} \qquad \frac{e \xrightarrow{a} e'}{e^\star \xrightarrow{a} e' \cdot e^\star}$$

define the transition function $t\colon RE \to RE^A$ as $t(e)(a) = e'$ iff $e \xrightarrow{a} e'$. The above presentation of Brzozowski derivatives by means of inference rules is unusual, but it is convenient here to stress the similarity with GSOS specifications [1] that will be pivotal for our development in Section **??**.

The deterministic automaton $(RE, \langle o, t\rangle)$ uniquely defines the map $[\![-]\!]\colon RE \to 2^{A^\star}$ and Kleene Algebra provides a sound and complete axiomatisation for $\sim$. The soundness of these axioms can be now proved by means of coinduction. For instance, commutativity of $+$,

$$e + f \sim f + e$$

is simply proved by checking that the relation $R = \{(e+f, f+e) \mid e, f \in RE\}$ is a bisimulation. Indeed $(e+f)\!\downarrow \Leftrightarrow e\!\downarrow \vee f\!\downarrow \Leftrightarrow (f+e)\!\downarrow$ and for all $a \in A$,

$$\begin{array}{ccc} e+f & & f+e \\ {\scriptstyle a}\!\downarrow & & \downarrow\!{\scriptstyle a} \\ e'+f' & R & f'+e' \end{array}$$

In a similar way, one can prove that $(RE, +, 0)$ is a monoid, but things get trickier for distributivity, for instance on the right:

$$e(f+g) \sim ef + eg.$$

Indeed, let us check whether the relation $R = \{(e(f+g), ef+eg) \mid e, f, g \in RE\}$ is a bisimulation. It is immediate to check that $e(f+g)\downarrow \Leftrightarrow (ef+eg)\downarrow$. However, the arriving states after a transition are not related by R, hence R is not a bisimulation.

$$\begin{array}{ccc} e(f+g) & & ef+eg \\ {\scriptstyle a}\big\downarrow & & \big\downarrow{\scriptstyle a} \\ e'(f+g)+o(e)(f'+g') & \not R & (e'f+o(e)f')+(e'g+o(e)g') \end{array} \tag{21}$$

However, as we will see below, the relation R is a *bisimulation up-to* for a particular composite up-to technique. Its components are the function $Bhv\colon \mathsf{Rel}_{RE} \to \mathsf{Rel}_{RE}$ defined for all relations $R \subseteq RE^2$ as

$$Bhv(R) = \{(e, f) \mid \exists e', f' \text{ s.t. } e \sim e' R f' \sim f\}$$

and the function $Ctx\colon \mathsf{Rel}_{RE} \to \mathsf{Rel}_{RE}$ mapping every relation R to its contextual closure $Ctx(R)$. The latter is defined inductively by the following rules.

$$\frac{e\ R\ f}{e\ Ctx(R)\ f} \qquad \frac{-}{0\ Ctx(R)\ 0} \qquad \frac{-}{1\ Ctx(R)\ 1}$$

$$\frac{e\ Ctx(R)\ e' \qquad f\ Ctx(R)\ f'}{e+f\ Ctx(R)\ e'+f'} \qquad \frac{e\ Ctx(R)\ e' \qquad f\ Ctx(R)\ f'}{ef\ Ctx(R)\ e'f'} \qquad \frac{e\ Ctx(R)\ f}{e^\star\ Ctx(R)\ f^\star}$$

Now, it is easy to see that the relation $R = \{(e(f+g), ef+eg) \mid e, f, g \in RE\}$ is a *bisimulation up to* $Bhv \circ Ctx$, meaning that $R \subseteq B(Bhv(Ctx(R)))$. Indeed (21) is proved to hold by observing that

$$e'(f+g)+o(e)(f'+g')\ Ctx(R)\ (e'f+e'g)+(o(e)f'+o(e)g')$$

and that $(e'f+e'g)+(o(e)f'+o(e)g') \sim (e'f+o(e)f')+(e'g+o(e)g')$ since, as shown above, $+$ is associative and commutative. Hence, the arriving states in (21) are related by $Bhv \circ Ctx(R)$.

A.2 Arden's rule

As the last example of this section, we provide a coinductive proof of Arden's rule. This is usually formulated for arbitrary languages, but we rephrase it here in terms of regular expressions so to reuse the notation introduced so far. The coinductive proof for arbitrary languages is completely analogous, see [7].

Arden's rule states that, given two expressions k and m, the "behavioural" equation

$$e \sim ke + m$$

has $e = k^\star m$ as *solution*, i.e., $k^\star m \sim kk^\star m + m$. Furthermore,

(a) it is the *smallest solution* (up to $\sim$), namely if $f \sim kf + m$ then $k^\star m \precsim f$;
(b) if $k\not\downarrow$, then it is the *unique solution* (up to $\sim$), namely if $f \sim kf + m$ then $k^\star m \sim f$.

Here $\precsim$ denotes *language inclusion*: $e \precsim f$ iff $[\![e]\!] \subseteq [\![f]\!]$. In order to proceed with a coinductive proof of Arden's rule, we characterise $\precsim$ as $\nu B'$, the greatest fixed-point of the monotone function $B'\colon \mathsf{Rel}_{RE} \to \mathsf{Rel}_{RE}$ mapping $R \subseteq RE^2$ to

$$B'(R) = \{(e, f) \mid o(e) \leq o(f) \text{ and for all } a \in A,\ (t(e)(a), t(f)(a)) \in R\}.$$

One can apply the Knaster-Tarski fixed point theorem to B' so to obtain the analogue of (6) which allows to prove $e \precsim f$ by showing a relation R such that $\{(e, f)\} \subseteq R$ and R is a *simulation*, i.e., $R \subseteq B'(R)$.

The proof proceeds as follows. First observe that $k^\star m$ is indeed a solution since $k^\star m \sim (kk^\star + 1)m \sim kk^\star m + m$. For (a), we prove that $S = \{(k^\star m, f)\}$ is a *simulation up-to*. For the

outputs, $k^\star m{\downarrow} \Rightarrow m{\downarrow} \Rightarrow (kf+m){\downarrow} \Rightarrow f{\downarrow}$ where the last implication follows from $f \sim kf+m$. For every $a \in A$, we have

$$\begin{array}{ccccccccccc} k^\star m & & & & & & & & & & f \\ \big\downarrow{\scriptstyle a} & & & & & & & & & & {\scriptstyle a}\big\downarrow \\ (k'k^\star)m + 1 \cdot m' & \sim & k'(k^\star m) + m' & Ctx(S) & k'f + m' & \precsim & (k'f + o(k)f') + m' & \sim & f' \end{array} \tag{22}$$

where the leftmost transition is derived as on the left below and $(k'f + o(k)f') + m' \sim f'$ follows from $kf + m \sim f$ and the transition derived on the right below.

$$\frac{\dfrac{k \xrightarrow{a} k'}{k^\star \xrightarrow{a} k'k^\star} \qquad m \xrightarrow{a} m'}{k^\star m \xrightarrow{a} k'k^\star m + 1 \cdot m'} \qquad\qquad \frac{\dfrac{k \xrightarrow{a} k' \qquad f \xrightarrow{a} f'}{kf \xrightarrow{a} k'f + o(k)f'} \qquad m \xrightarrow{a} m'}{kf + m \xrightarrow{a} (k'f + o(k)f') + m'}$$

Observe that S is not a simulation up to $Bhv \circ Ctx$, since in (22) it is necessary to use $\precsim$. We have to use a further up-to technique $Slf : \mathsf{Rel}_{RE} \to \mathsf{Rel}_{RE}$ defined for all R as

$$Slf(R) = \{(e, f) \mid \exists e', f' \text{ s.t. } e \precsim e' \; R \; f' \precsim f\}.$$

Since $k'f + m' \precsim (k'f + o(k)f') + m' \sim f'$, then $k'f + m' \precsim f'$ and therefore S is a *simulation up to* $Slf \circ Ctx$, i.e., $S \subseteq B'(Slf(Ctx(S)))$.

For (b), we assume $k{\not\downarrow}$ and $f \sim kf+m$, and we show that $R = \{(k^\star m, f)\}$ is a bisimulation up to $Bhv \circ Ctx$. For the outputs, since $k^\star{\downarrow}$, $k{\not\downarrow}$ and $f \sim kf + m$, we have $k^\star m{\downarrow} \Leftrightarrow m{\downarrow} \Leftrightarrow (kf + m){\downarrow} \Leftrightarrow f{\downarrow}$. For every $a \in A$, the transitions are the same as in (22), and the proof that the arriving states are related by $Bhv \circ Ctx(S)$ is similar. The only difference is that the step $k'f + m' \precsim (k'f + o(k)f') + m'$ is replaced by $k'f + m' \sim (k'f + o(k)f') + m'$, which is valid since $k{\not\downarrow}$ by assumption.