Coinductive Proof Techniques for Language Equivalence

Jurriaan Rot[1,2,⋆], Marcello Bonsangue[1,2], and Jan Rutten[2,3]

[1] LIACS – Leiden University, Niels Bohrweg 1, Leiden, Netherlands
[2] Centrum Wiskunde en Informatica, Science Park 123, Amsterdam, Netherlands
[3] ICIS – Radboud University Nijmegen, Heyendaalseweg 135, Nijmegen, Netherlands
`{jrot, marcello}@liacs.nl, janr@cwi.nl.`

Abstract. Language equivalence can be checked coinductively by establishing a bisimulation on suitable deterministic automata. We improve and extend this technique with *bisimulation-up-to*, which is an enhancement of the bisimulation proof method. First, we focus on the regular operations of union, concatenation and Kleene star, and illustrate our method with new proofs of classical results such as Arden's rule. Then we extend our enhanced proof method to incorporate language complement and intersection. Finally we define a general format of behavioural differential equations, in which one can define operations on languages for which bisimulation-up-to is a sound proof technique.

1 Introduction

The set of all languages over a given alphabet can be turned into an (infinite) deterministic automaton. By the *coinduction* principle, any two languages that are *bisimilar* as states in this automaton are in fact equal. The typical way to show that two languages x and y are bisimilar is by exhibiting a *bisimulation*, which in this setting is a relation on languages containing the pair (x, y) and satisfying certain properties. Indeed, this is the basis of a practical coinductive proof method for language equality [21], which has, for example, been applied in effective procedures for checking equivalence of regular languages [12, 21, 13, 6].

In this paper we present *bisimulation up to congruence*, in the context of languages and automata. This is an enhancement of bisimulation originally stemming from process theory [24, 18]. In order to prove bisimilarity of two languages, instead of showing that they are related by a bisimulation, one can show that they are related by a *bisimulation-up-to*, which in many cases yields smaller, easier and more elegant proofs. As such, we introduce a proof method which improves on the more classical coinductive approach based on bisimulations.

At first, we will focus on languages presented by the regular operations of union, concatenation and Kleene star. We will exemplify our coinductive proof method based on bisimulation-up-to by novel proofs of several classical results

⋆ The research of this author has been funded by the Netherlands Organisation for Scientific Research (NWO), CoRE project, dossier number: 612.063.920.

such as Arden's rule and the soundness of the axioms of Kleene algebra. Then we proceed to incorporate language intersection and complement and show the usefulness and versatility of the techniques by giving a full coinductive proof that two context-free languages defined in terms of language equations, that of palindromes and that of non-palindromes, indeed form each others complement. Finally, in order to deal with other language operations, such as shuffle or symmetric difference, we introduce a general format of *behavioural differential equations*, for operations allowing proofs based on bisimulation-up-to.

The soundness of bisimulation-up-to, stating that whenever two languages are related by a bisimulation-up-to they are equal, follows from abstract coalgebraic theory [20, 19]. The main contribution of this paper is the presentation of this proof technique in the context of languages and automata, without explicitly using any coalgebra or category theory. As witnessed by the many examples in this paper, this yields a useful, elegant and efficient method for proving equality of languages, enhancing the existing successful coinductive methods based on establishing bisimulations. Moreover, from the perspective of coalgebraic theory this can be regarded as an extensive concrete exercise in bisimulation-up-to. On the technical side, the general format of operations for which bisimulation-up-to is sound, can be considered a novel contribution.

The outline of this paper is as follows. In Section 2 we recall the notions of bisimulation and coinduction in the context of languages and automata. Then in Section 3 we present bisimulation-up-to for the regular operations. In Section 4 we extend our techniques to deal with complementation and intersection, and in Section 5 we present a format for which bisimulation-up-to is guaranteed to be sound. In Section 6 we place our work in the context of coalgebraic theory and discuss related work, and finally in Section 7 we conclude.

2 Languages, automata, bisimulations and coinduction

Throughout this paper we assume a fixed alphabet A, which is simply a (possibly infinite) set. We denote by A^* the set of *words*, i.e., finite concatenations of elements of A; we denote the empty word by ε, and the concatenation of two words w and v by wv. The set of *languages* over A is given by $\mathcal{P}(A^*)$, and ranged over by x, y, z. We denote the empty language by 0 and the language $\{\varepsilon\}$ by 1. Moreover when no confusion is likely to arise, we write a to denote the language $\{a\}$, for alphabet letters $a \in A$.

A *(deterministic) automaton* over A is a triple (S, o, δ) where S is a set of states, $o\colon S \to \{0,1\}$ is an output function, and $\delta\colon S \times A \to S$ is a transition function. Notice that S is not necessarily finite, and there is no initial state. We say a state $s \in S$ is *final* or *accepting* if $o(s) = 1$. For each automaton (S, o, δ) there is a function $l\colon S \to \mathcal{P}(A^*)$ which assigns to each state $s \in S$ a language, inductively defined as follows: $\varepsilon \in l(s)$ iff $o(s) = 1$ and $aw \in l(s)$ iff $w \in l(\delta(s, a))$.

The classical definition of *bisimulation* [14, 17] applies to labelled transition systems, which, in contrast to deterministic automata, do not feature output and may have a non-deterministic branching behaviour. We will base ourselves on a different notion of bisimulation specific to deterministic automata, which is an instantiation of general coalgebraic theory (see Section 6). A *bisimulation* is a relation $R \subseteq S \times S$ such that for any $(s,t) \in R$: $o(s) = o(t)$ and for all $a \in A$: $(\delta(s,a), \delta(t,a)) \in R$. Given a deterministic automaton, the union of all bisimulations is again a bisimulation, is denoted by $\sim$ and is called *bisimilarity*; if $s \sim t$ for two states s,t then we say these states are *bisimilar*. Notice that in order to show that two states s and t are bisimilar, it suffices to construct a bisimulation R such that $(s,t) \in R$. As it turns out, this gives a proof principle for showing language equivalence of states:

Theorem 1 (Coinduction). *For any two states s,t of a deterministic automaton: if $s \sim t$ then $l(s) = l(t)$.*

This coinduction principle is easily proved by induction. The converse holds as well, i.e., if $l(s) = l(t)$ then s is related to t by some bisimulation R. Such a relation R may well be infinite, but this is not necessarily a problem; in practice one can often give a finite description of such an infinite relation.

In order to proceed we recall the notion of *language derivatives*: the a-derivative of a language x is defined as $x_a = \{w \mid aw \in x\}$. The set $\mathcal{P}(A^*)$ of all languages can be turned into a deterministic automaton by defining the output function and the transition function as follows: $o(x) = 1$ if $\varepsilon \in x$ and $o(x) = 0$ otherwise, and $\delta(x,a) = x_a$ for all $a \in A$. One can check that for any language x, the language accepted by the corresponding state in the automaton is precisely x itself. A relation R on languages is a bisimulation on this automaton if for any $(x,y) \in R$: $o(x) = o(y)$ and for any $a \in A$: $(x_a, y_a) \in R$. By the coinduction principle (Theorem 1), we now have the following method for checking equality of languages x and y: if we can establish a bisimulation containing the pair (x,y), then $x \sim y$, so $x = y$.

We will be interested in the regular operations on languages, defined in a standard way: union $x + y = \{w \mid w \in x \text{ or } w \in y\}$, concatenation $x \cdot y = \{w \mid w = uv \text{ for some } u \in x \text{ and } v \in y\}$ and Kleene star $x^* = \sum_{i \geq 0} x^i$, where $x^0 = 1$ and $x^{i+1} = x \cdot x^i$. We often write xy for $x \cdot y$.

In order to prove equivalence of languages defined using the above operations we may use bisimulations, but for this we need a characterization of the output (acceptance of the empty word) and the derivatives of languages. Such a characterization was given for regular expressions by Brzozowski [3]; we formulate this in terms of languages (e.g., [5, page 41]):

Lemma 2. *For any two languages x,y and for any $a,b \in A$:*

$$\begin{array}{ll}
0_a = 0 & o(0) = 0 \\
1_a = 0 & o(1) = 1 \\
b_a = \begin{cases} 1 & \textit{if } b = a \\ 0 & \textit{otherwise} \end{cases} & o(b) = 0 \\
(x+y)_a = x_a + y_a & o(x+y) = o(x) \vee o(y) \\
(x \cdot y)_a = x_a \cdot y + o(x) \cdot y_a & o(x \cdot y) = o(x) \wedge o(y) \\
(x^*)_a = x_a \cdot x^* & o(x^*) = 1
\end{array}$$

Example 3. Let us prove that $(a+b)^* = (a^*b^*)^*$ for some alphabet letters a, b (for simplicity we assume that the alphabet does not contain any other letters). To this end, we start with the relation $R = \{((a+b)^*, (a^*b^*)^*)\}$ and try to show it is a bisimulation. So we must show that the outputs of $(a+b)^*$ and $(a^*b^*)^*$ coincide, and that their a-derivatives and their b-derivatives are related by R. Using Lemma 2, we see that $o((a+b)^*) = 1 = o((a^*b^*)^*)$. Moreover, again using Lemma 2, we have $((a+b)^*)_a = (a+b)_a(a+b)^* = (1+0)(a+b)^* = (a+b)^*$ and $((a^*b^*)^*)_a = (a^*b^*)_a(a^*b^*)^* = ((a^*)_a b^* + o(a^*)(b^*)_a)(a^*b^*)^* = (a^*b^* + 0)(a^*b^*)^* = (a^*b^*)^*$, so the a-derivatives are again related (notice that apart from Lemma 2, we have used some basic facts about the regular operations). The b-derivative of $(a+b)^*$ is $(a+b)^*$ itself; however, the b-derivative of $(a^*b^*)^*$ is $b^*(a^*b^*)^*$. This means that R is not a bisimulation. However, we can consider instead the relation $R' = R \cup \{((a+b)^*, b^*(a^*b^*)^*)\}$. As it turns out, the pair $((a+b)^*, b^*(a^*b^*)^*)$ satisfies the necessary conditions as well, turning R' into a bisimulation. We leave the details as an exercise for the reader, and conclude $(a+b)^* = (a^*b^*)^*$ by coinduction.

Bisimulation proofs in general will follow the above pattern of using Lemma 2 to compute outputs and to expand the derivatives, and then using some reasoning to show that the outputs are equal and the derivatives related. In the sequel we will sometimes use Lemma 2 without further reference to it. We note that the above coinductive proof method applies to general languages, not only to regular ones. However if one restricts to regular languages, then this technique give rise to an effective algorithm for checking equivalence.

The axioms of *Kleene algebra (KA)* [11] constitute a complete axiomatisation of language equivalence of regular expressions. We recall them here for the following two reasons. First, they provide a number of interesting examples for our methods. Second, and more importantly, these axioms are often quite useful for relating derivatives. We state the axioms in terms of languages and our concrete operations:

$$x + (y+z) = (x+y) + z \quad (1) \qquad (y+z)x = yx + zx \quad (8)$$
$$x + y = y + x \quad (2) \qquad x(y+z) = xy + xz \quad (9)$$
$$x + x = x \quad (3) \qquad x^*x + 1 = x^* \quad (10)$$
$$x + 0 = x \quad (4) \qquad xx^* + 1 = x^* \quad (11)$$
$$x(yz) = (xy)z \quad (5) \qquad xy \subseteq x \to xy^* \subseteq x \quad (12)$$
$$x \cdot 1 = 1 \cdot x = x \quad (6) \qquad yx \subseteq x \to y^*x \subseteq x \quad (13)$$
$$x \cdot 0 = 0 \cdot x = 0 \quad (7)$$

Notice that $x \subseteq y$ iff $x + y = y$. All of (1) through (9) follow easily from the definition of the operations. The remaining axioms will be treated below.

3 Bisimulation-up-to

In this section we will introduce an enhancement of the bisimulation proof method. We first illustrate the need for such an enhancement with an example. Consider the Kleene algebra axiom (11) (see Section 2). In order to prove this identity coinductively, consider the relation $R = \{(xx^*+1, x^*) \mid x \in \mathcal{P}(A^*)\}$; let us see if this is a bisimulation. Using Lemma 2, it is easy to show that for any language x, the outputs of $xx^* + 1$ and x^* are equal. For any $a \in A$:

$$(xx^* + 1)_a = x_a x^* + o(x)x_a x^* + 0 = x_a x^* = (x^*)_a$$

where the leftmost and rightmost equality are by Lemma 2, and in the second step we use some of the KA identities which we know to hold. Now we have shown that the derivatives are *equal*; this does not allow us to conclude that R is a bisimulation, since for that, the derivatives need to be *related by* R. Instead we can consider the relation $R' = R \cup \{(y, y) \mid y \in \mathcal{P}(A^*)\}$. Then the derivatives of $xx^* + 1$ and x^* are related by R'; moreover, the diagonal is easily seen to satisfy the properties of a bisimulation as well. This solves the problem, but is obviously somewhat inconvenient.

Now consider the relation $R = \{(x^*x + 1, x^*) \mid x \in \mathcal{P}(A^*)\}$ which we may try to use to prove (10) by coinduction. The derivatives are (using Lemma 2):

$$(x^*x + 1)_a = x_a x^* x + x_a + 0 = x_a(x^*x + 1) \qquad \text{and} \qquad (x^*)_a = x_a x^*$$

Clearly $x_a x^*$ can be obtained from $x_a(x^*x + 1)$ by substituting $x^*x + 1$ for x^*, and indeed the latter two languages are related by R. However, unfortunately these derivatives are not related *directly* by R, and so R is not a bisimulation. Extending R to a bisimulation is indeed a non-trivial task.

When proving identities over languages coinductively, situations such as in the above examples occur very often. To deal with such cases in a better way, we will introduce *bisimulation-up-to*. We need the notion of congruence closure.

Definition 4. *For a relation $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$, define the* congruence closure *of R as the least relation $\equiv$ satisfying the following rules:*

$$\frac{xRy}{x \equiv y} \qquad \frac{}{x \equiv x} \qquad \frac{x \equiv y}{y \equiv x} \qquad \frac{x \equiv y \qquad y \equiv z}{x \equiv z}$$

$$\frac{x_1 \equiv y_1 \qquad x_2 \equiv y_2}{x_1 + x_2 \equiv y_1 + y_2} \qquad \frac{x_1 \equiv y_1 \qquad x_2 \equiv y_2}{x_1 \cdot x_2 \equiv y_1 \cdot y_2} \qquad \frac{x \equiv y}{x^* \equiv y^*}$$

In the sequel we denote the congruence closure of a given relation R by $\equiv_R$.

The upper left rule ensures $R \subseteq \equiv_R$. The three rules on the right in the first row ensure that $\equiv_R$ is an equivalence relation. Notice that the reflexivity rule has as a consequence that languages which are equal, are also related by $\equiv_R$. The transitivity rule allows to relate languages in multiple "proof steps". Finally the three rules on the second row ensure that $\equiv_R$ is a congruence, which in particular means that $\equiv_R$ relates languages which are obtained by (syntactic) substitution of languages related by R. For example, if $x^*x+1 \; R \; x^*$, then we can derive from the above rules that $x_a(x^*x + 1) \equiv_R x_a x^*$.

Definition 5. *Let $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$ be a relation on languages. We say R is a* bisimulation up to congruence*, or simply a* bisimulation-up-to*, if for any pair $(x, y) \in R$: $o(x) = o(y)$ and for all $a \in A$: $x_a \equiv_R y_a$.*

The idea of bisimulation-up-to is that now the derivatives are easier to relate, since they can be related by the *congruence* $\equiv_R$ instead of only the relation R itself. Indeed, to prove that R is a bisimulation-up-to, the derivatives can be related by equational reasoning. Of course such a bisimulation-up-to R is in general not a bisimulation. As it turns out, showing that R is a bisimulation-up-to is enough to ensure that $\equiv_R$ is itself a bisimulation (this result is based on general coalgebraic theory which we will shortly discuss in Section 6). This means that in that case for any $(x, y) \in \equiv_R$ we have $x = y$ by coinduction. Since $R \subseteq \equiv_R$ this holds in particular for all pairs related by the bisimulation-up-to R. So we have the following proof principle:

Theorem 6 (Coinduction-up-to). *If R is a bisimulation-up-to then for any $(x, y) \in R$: $x = y$.*

Any bisimulation is also a bisimulation-up-to, so this generalizes Theorem 1 in the case of languages. Consequently, the converse of the above principle holds as well. We proceed with a number of proofs based on bisimulation-up-to.

Example 7. Recall the relation $R = \{(x^*x + 1, x^*) \mid x \in \mathcal{P}(A^*)\}$ from the beginning of this section. As we have seen, the a-derivatives are $x_a(x^*x + 1)$ and $x_a x^*$, which are not related by R; however they *are* related by $\equiv_R$. So R is a bisimulation-up-to, and consequently $x^*x + 1 = x^*$. Moreover, the relation $\{(xx^* + 1, x^*) \mid x \in \mathcal{P}(A^*)\}$ from the beginning of this section is a bisimulation-up-to as well; there, the derivatives are equal and thus related by $\equiv_R$.

We have covered coinductively the soundness of most of the axioms of Kleene algebra; the only remaining ones are right and left star induction, i.e., (12) and (13). Instead of proving these we will consider *Arden's rule* below; the coinductive proof of (12) and (13) is very similar (notice that we can treat an inclusion $x \subseteq y$ simply by showing $x + y = y$). Finally, notice that instead of proving the KA axioms one by one, we could consider the equivalence relation R induced by all the KA axioms together, and show that this is a bisimulation-up-to.

Example 8. Arden's rule states that if $x = yx + z$ for some languages x, y and z, and y does not contain the empty word, then $x = y^*z$. In order to prove its validity coinductively, let x, y, z be languages such that $\varepsilon \notin y$ and $x = yx + z$, and let $R = \{(x, y^*z)\}$. Then $o(y) = 0$, so $o(x) = o(yx + z) = (o(y) \wedge o(x)) \vee o(z) = (0 \wedge o(x)) \vee o(z) = o(z) = 1 \wedge o(z) = o(y^*) \wedge o(z) = o(y^*z)$ and for any $a \in A$: $x_a = (yx + z)_a = y_a x + o(y) \cdot x_a + z_a = y_a x + z_a \equiv_R y_a y^* z + z_a = (y^*z)_a$. So R is a bisimulation-up-to, proving Arden's rule.

While Arden's rule is not extremely difficult to prove without coinduction either, the textbook proofs are significantly longer and arguably more involved than the above proof, which is not much more than taking derivatives combined with a tiny bit of algebraic reasoning. Nevertheless this coinductive proof is completely precise. Giving a formal proof without our methods is not trivial at all; see, e.g., [7] for the discussion of a proof within the theorem prover Isabelle.

In fact, [21] already contains a coinductive proof of Arden's rule; however, this is based on a bisimulation (in contrast to our proof which is based on a bisimulation-up-to). Indeed, in [21] the infinite relation $\{(ux+v, uy^*z+v) \mid u, v \in \mathcal{P}(A^*)\}$ is used, requiring more work in checking the bisimulation conditions. In that case one essentially closes the relation under (certain) contexts manually; the coinduction-up-to principle does this in a general and systematic fashion.

Example 9. Let us prove that for any language x, we have $xx = 1 \to x = 1$. Assume $xx = 1$ and let $R = \{(x, 1)\}$. Since $o(xx) = o(1) = 1$ also $o(x) = 1 = o(1)$. We show that the derivatives of x and 1 are equal, turning R into a bisimulation-up-to. For any $a \in A$: $x_a x + x_a = x_a x + o(x)x_a = (xx)_a = 1_a = 0$. One easily proves that this implies $x_a = 0$ (for example by showing that $\{(y, 0) \mid x + y = 0\}$ is a bisimulation). Thus $x_a = 0 = 1_a$, so $x_a \equiv_R 1_a$.

Example 10. We prove the soundness of the axiom $xx = x \to x^* = 1 + x$, by establishing a bisimulation-up-to. This axiom was used by Boffa in his complete axiomatisation of equivalence of regular expressions. Let x be a language for which $xx = x$ and consider the relation $R = \{(x^*, 1 + x)\}$. Indeed, $o(x^*) = 1 = o(1 + x)$, and for any $a \in A$: $(x^*)_a = x_a x^* \equiv_R x_a(1 + x) = x_a + x_a x = x_a + o(x)x_a + x_a x = x_a + (xx)_a = x_a + x_a = x_a$.

In fact the above axiom can also easily be proved by induction. In practice, one wants to combine inductive and coinductive methods.

4 Language equations and Boolean operators

Context-free languages can be expressed in terms of certain types of language equations [8]. For example, the language $\{a^n b^n \mid n \in \mathbb{N}\}$ is the unique language x such that $x = axb + 1$. Our coinductive techniques can directly be applied to languages defined in such a way, and so we are able to reason about (equivalence of) context-free languages in a novel manner.

Example 11. Let x, y, z be languages such that $x = ayzb + 1$, $y = azxb + 1$ and $z = axyb + 1$. Without thinking of what possible concrete descriptions of x, y and z can be, let us show, by coinduction, that $x = y = z$. We use the relation $R = \{(x, y), (y, z)\}$. Obviously $o(x) = o(y)$ and $o(y) = o(z)$. Moreover for any alphabet letter b other than a, we have $x_b = 0 = y_b$ and $y_b = 0 = z_b$. For the a-derivatives we have $x_a = yzb \equiv_R zyb \equiv_R zxb = y_a$ and similarly for (y, z); so R is a bisimulation-up-to, proving that $x = y = z$.

So far we have considered the regular operations of union, concatenation and Kleene star. We proceed to incorporate complement and intersection, defined as $\overline{x} = \{w \mid w \notin x\}$ and $x \wedge y = \{w \mid w \in x \text{ and } w \in y\}$ respectively. Language equations including these additional operators can be used to give semantics to *conjunctive*- and *Boolean grammars* [16]. Complement and intersection have a known characterization in terms of outputs and derivatives as well [3]:

Lemma 12. *For any two languages x, y and for any $a \in A$:*

$$\begin{array}{ll} o(\overline{x}) = \neg o(x) & \overline{x}_a = \overline{x_a} \\ o(x \wedge y) = o(x) \wedge o(y) & (x \wedge y)_a = x_a \wedge y_a \end{array}$$

As a consequence, we have bisimulation and coinduction to our disposal to show equivalence of languages defined in terms of systems of equations involving these additional operators. In order to allow for proofs based on bisimulation-up-to, we first need to extend the congruence closure to deal with intersection and complement. The *boolean congruence closure* $\equiv_R^B$ of a relation R is defined as the least relation $\equiv$ which satisfies the rules of the congruence closure (Definition 4) plus the following two rules:

$$\frac{x_1 \equiv y_1 \qquad x_2 \equiv y_2}{x_1 \wedge x_2 \equiv y_1 \wedge y_2} \qquad \frac{x \equiv y}{\overline{x} \equiv \overline{y}}$$

The associated definition of bisimulation-up-to is as expected. Fortunately, it is also sound as a proof technique for language equality. We do not formally state this here, but in Section 5 we will introduce a general coinduction-up-to theorem which also applies to this case. In the sequel we will simply write $\equiv_R$ for $\equiv_R^B$.

We have already seen that $(\mathcal{P}(A^*), 0, 1, +, \cdot, ^*)$ is a Kleene algebra; it is useful to know that $(\mathcal{P}(A^*), 0, A^*, \overline{}, +, \wedge)$ is a Boolean algebra. Moreover, below we will need the following property, which holds for any language x and $a \in A$:

$$\overline{xa} = \overline{x}a + \sum_{b \in A \setminus \{a\}} A^*b + 1 \tag{14}$$

For lack of space we do not include a proof; it can very well be shown coinductively.

Example 13. There are unique languages x and y such that

$$x = axa + bxb + a + b + 1 \qquad\qquad y = aya + byb + aA^*b + bA^*a$$

x is the language of *palindromes*, i.e., words which are equal to their own reverse. We claim that y must be the language of all *non*-palindromes, i.e., $y = \overline{x}$. We proceed to prove this formally by showing that the relation $R = \{(\overline{x}, y)\}$ is a bisimulation-up-to. The outputs are easily seen to be equal: $o(\overline{x}) = \neg o(x) = \neg o(1) = 0 = o(y)$. We consider the a-derivatives; the b-derivatives are of course similar. In the fourth step we use (14).

$$\begin{aligned}\overline{x}_a &= \overline{x_a} = \overline{xa+1} = \overline{xa} \wedge \overline{1} = (\overline{x}a + A^*b + 1) \wedge \overline{1} \\ &\equiv_R (ya + A^*b + 1) \wedge \overline{1} = ya \wedge \overline{1} + A^*b \wedge \overline{1} + 1 \wedge \overline{1} = ya + A^*b = y_a\end{aligned}$$

So R is a bisimulation-up-to, proving that y indeed is the complement of x.

5 Bisimulation-up-to for other operations

So far we have focused on the regular operations in Section 3 and added complement and intersection in Section 4. One may be interested in other operations on languages as well, such as shuffle or symmetric difference. In this section we provide general conditions under which bisimulation-up-to can be applied.

A *signature* Σ is a collection of operator names $\sigma \in \Sigma$ with associated arities $|\sigma| \in \mathbb{N}$. We define a general congruence closure with respect to a signature:

Definition 14. *For a relation $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$, define the Σ-*congruence closure *$\equiv_R^\Sigma$ of R as the least relation $\equiv$ satisfying the following rules:*

$$\frac{xRy}{x \equiv y} \qquad \frac{}{x \equiv x} \qquad \frac{x \equiv y}{y \equiv x} \qquad \frac{x \equiv y \quad y \equiv z}{x \equiv z}$$

$$\frac{x_1 \equiv y_1 \quad \dots \quad x_n \equiv y_n}{\sigma(x_1, \dots, x_n) \equiv \sigma(y_1, \dots, y_n)} \qquad \textit{for each } \sigma \in \Sigma, n = |\sigma|$$

The congruence closure for the regular operators (Definition 4) and the Boolean congruence closure are special cases of the above definition. Given this generalized congruence closure, we define bisimulation-up-to with respect to a given signature, generalizing Definition 5: a relation $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$ is a *bisimulation-up-to w.r.t. Σ*, if for any $(x, y) \in R$: $o(x) = o(y)$ and for any $a \in A$: $x_a \equiv_R^\Sigma y_a$.

We will give a general condition on the operations involved under which we have an associated coinduction-up-to principle. In order to proceed we define the set of *terms* $T_\Sigma V$ over a signature Σ and a set of variables V by the grammar $t ::= v \mid \sigma(t_1, \dots, t_n)$ where v ranges over V, σ ranges over Σ and $n = |\sigma|$.

Definition 15. *Assume given a signature Σ and for each $\sigma \in \Sigma$ an interpretation $\hat{\sigma} \colon \mathcal{P}(A^*)^n \to \mathcal{P}(A^*)$, where $n = |\sigma|$. We say the semantics of Σ can be given by* behavioural differential equations *if for all $\sigma \in \Sigma$ there is:*

- *a function $f \colon 2^n \to 2$ (n is the arity of σ),*
- *a term $t(v_1, \dots, v_m) \in TV$ for some set of variables V and some $m \in \mathbb{N}$,*
- *for every $a \in A$, a function $g_a \colon \mathcal{P}(A^*)^n \to \mathcal{P}(A^*)^m$ such that for each $i \leq m$ there is $j \leq n$ s.t. either $g_a(x_1, \dots, x_n)(i) = x_j$, or $g_a(x_1, \dots, x_n)(i) = (x_j)_b$ for some $b \in A$, or $g_a(x_1, \dots, x_n)(i) = o(x_j)$,*

such that $o(\hat{\sigma}(x_1, \dots, x_n)) = f(o(x_1), \dots, o(x_n))$ and

$$\hat{\sigma}(x_1, \dots, x_n)_a = t(g_a(\widehat{x_1, \dots}, x_n)) \qquad \textit{for each } a \in A$$

where $\hat{\cdot}$ is the extension of the interpretation of the operators to terms.

Indeed Lemma 2 witnesses that the regular operations can be given by behavioural differential equations, and Lemma 12 shows this additionally for complement and intersection (and see, e.g., [21] for a definition of the shuffle operator by behavioural differential equations). So the following theorem generalizes the coinduction-up-to principle of Theorem 6:

Theorem 16 (Coinduction-up-to). *If the semantics of the operators of a signature Σ can be given by behavioural differential equations, then for any relation R which is a bisimulation-up-to w.r.t Σ: if $(x, y) \in R$ then $x = y$.*

In fact, the coinduction-up-to principle holds even if, in Definition 15, one allows for a different term t_a for every alphabet symbol $a \in A$, instead of a single term t. We have chosen not to include this in the definition for readability reasons.

We conclude this section with an example, adapted from [18], showing that bisimulation-up-to is in general not sound (for operations not given by behavioural differential equations), and as such illustrating the need for a restricted format. Assume, for simplicity, a singleton alphabet $\{a\}$. Consider the constants $\underline{0}$ and $\underline{a}$, and a unary function h, with the following interpretation in languages: $\underline{0} = 0$, $\underline{a} = \{a\}$, $h(0) = 0$ and $h(x) = 1$ for all languages x s.t. $x \neq 0$. The outputs of these two constants and this function are as follows: $o(\underline{0}) = o(\underline{a}) = 0$ and $o(h(x)) = 0$ iff $x = 0$. For the derivatives we have $\underline{a}_a = h(\underline{a})$, $\underline{0}_a = h(\underline{0})$ and $h(x)_a = 0$. Now the relation $R = \{(\underline{0}, \underline{a})\}$ is a bisimulation-up-to, while $\underline{0} \neq \underline{a}$; so for these operations, there is no coinduction-up-to principle. Indeed, h can not be given by behavioural differential equations, because of the case distinction on its argument x rather than that it is based only on the output $o(x)$.

6 Discussion and related work

The theory of bisimulation and bisimulation-up-to developed in this paper are instantiations of the general theory of *coalgebras.* Coalgebra [22] is a general mathematical theory for the uniform study of state-based systems including labelled transition systems but also stream systems, various kinds of (weighted or probabilistic) automata, etc. Indeed, *deterministic automata*, as presented in Section 2, are also a certain type of coalgebras. *Bisimulation* is the canonical notion of equivalence of coalgebras, which, for labelled transition systems, coincides with the classical notion introduced by Milner and Park [14, 17]. In the case of deterministic automata, the associated instance of bisimulation is precisely the one presented in Section 2. The material of that section is from [21], which contains an extensive investigation of automata and languages as coalgebras.

Bisimulation-up-to classically is a family of enhancements of bisimulation for labelled transition systems [24, 18]; it is a rich theory which drastically improves the bisimulation proof method for, e.g., CCS processes. One interesting result based on bisimulation-up-to is the decidability result of [4], on equivalence of context-free processes. Note that these notions of bisimulation-up-to do not directly apply to our case, since our notion of bisimulation for *automata* is different from the classical one for labelled transition systems. Recently the theory of bisimulation-up-to was generalized from labelled transition systems to a large class of coalgebras [20, 19], yielding enhancements of the bisimulation proof method for many different kinds of state-based systems. In the present paper we have instantiated this general theory to deterministic automata and certain operations on languages. All operations defined in this paper adhere to specifications in terms of behavioural differential equations [23] (Definition 15). These can easily be represented as so-called *abstract GSOS specifications*, which are a way of defining operational semantics [25]. As a consequence, by the theory of [20, 19] bisimulation up to congruence for all of these cases is sound, meaning that any bisimulation-up-to can be extended to a bisimulation.

While we have introduced techniques which are much more widely applicable (as we have shown) than only to regular languages, we proceed to recall some of the related work on checking equivalence of regular expressions. There is a wide range of different tools and techniques tailored towards doing this; we only recall the ones most relevant to our work. CIRC [13] is a general coinductive theorem prover, which can deal with regular expressions. Recently, various algorithms based on Brzozowski derivatives and bisimulations have been implemented in Isabelle [12] and formalized in type theory, yielding an implementation in Coq [6] (while [6] does not mention bisimulations explicitly, their method is based on constructing a bisimulation). An efficient algorithm for deciding equivalence in Kleene algebra, based on automata but not on derivatives and bisimulations, was recently implemented in Coq as well [2]. Of course, one can reason about regular expressions in Kleene algebra; this is however a fundamentally different approach than the coinductive techniques of the present paper. In [9] a proof system for equivalence of regular expressions is presented, based on bisimulations but not on bisimulation-up-to. In [10] a general coinductive axiomatization of regular expression containment is given, based on an interpretation of regular expressions as types. The authors of [10] instantiate their axiomatization with the main coinductive rule from [9]. The focus of [10] is on constructive proofs based on parse trees of regular expressions; instead, we base ourselves on bisimulations between languages. Finally, the recent [1] introduces an efficient algorithm for checking equivalence of non-deterministic automata, based on a different notion of bisimulation up to congruence. One difference with the approach of [1], is that we can deal with (quasi)-equations over arbitrary languages.

If one works with syntactic *terms*, such as regular expressions, rather than with languages, the notion of *bisimulation up to bisimilarity* becomes relevant. In the corresponding proof method, one can relate derivatives, which are then terms, modulo bisimilarity. Since we work directly with languages, in our case this is not necessary; but for dealing with terms our techniques can easily be combined with up-to-bisimilarity (see [20, 19]). Bisimulation up to bisimilarity (alone, without context and equivalence closure) was originally introduced in [15], and in the context of automata and languages in [21].

7 Conclusions

We presented a proof method for language equivalence, based on coinduction and bisimulation-up-to. In particular we have considered (quasi-)equations over languages presented by the regular operations of union, concatenation and Kleene star, and additionally by complement and intersection. We have exemplified our approach with novel proofs of classical results.

The presented proof technique is very general, and it applies to undecidable problems such as language equivalence of context-free grammars. Indeed, automation is not the aim of the present paper. Nevertheless, the present techniques can be seen as a foundation for novel interactive theorem provers, and extensions of fully automated tools such as [12, 13, 6].

References

1. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proc. POPL (2013), to appear.

2. Braibant, T., Pous, D.: Deciding Kleene algebras in Coq. Logical Methods in Computer Science 8(1) (2012)
3. Brzozowski, J.: Derivatives of regular expressions. J. ACM 11(4), 481–494 (1964)
4. Christensen, S., Hüttel, H., Stirling, C.: Bisimulation equivalence is decidable for all context-free processes. In: Cleaveland, R. (ed.) CONCUR. LNCS, vol. 630, pp. 138–147. Springer (1992)
5. Conway, J.: Regular Algebra and Finite Machines. Chapman and Hall (1971)
6. Coquand, T., Siles, V.: A decision procedure for regular expression equivalence in type theory. In: Jouannaud, J.P., Shao, Z. (eds.) CPP. LNCS, vol. 7086, pp. 119–134. Springer (2011)
7. Foster, S., Struth, G.: Automated analysis of regular algebra. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR. LNCS, vol. 7364, pp. 271–285. Springer (2012)
8. Ginsburg, S., Rice, H.: Two families of languages related to ALGOL. J. ACM 9(3), 350–371 (1962)
9. Grabmayer, C.: Using proofs by coinduction to find "traditional" proofs. In: Fiadeiro, J., Harman, N., Roggenbach, M., Rutten, J. (eds.) CALCO. LNCS, vol. 3629, pp. 175–193. Springer (2005)
10. Henglein, F., Nielsen, L.: Regular expression containment: coinductive axiomatization and computational interpretation. In: Ball, T., Sagiv, M. (eds.) POPL. pp. 385–398. ACM (2011)
11. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. In: LICS. pp. 214–225. IEEE Computer Society (1991)
12. Krauss, A., Nipkow, T.: Proof pearl: Regular expression equivalence and relation algebra. J. Automated Reasoning 49, 95–106 (2012), published online March 2011
13. Lucanu, D., Goriac, E., Caltais, G., Rosu, G.: CIRC: A behavioral verification tool based on circular coinduction. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO. LNCS, vol. 5728, pp. 433–442. Springer (2009)
14. Milner, R.: A Calculus of Communicating Systems, LNCS, vol. 92. Springer (1980)
15. Milner, R.: Calculi for synchrony and asynchrony. TCS 25, 267–310 (1983)
16. Okhotin, A.: Conjunctive and boolean grammars: the true general case of the context-free grammars, `http://users.utu.fi/aleokh/papers/boolean_survey.pdf`
17. Park, D.M.R.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) Theoretical Computer Science. LNCS, vol. 104, pp. 167–183. Springer (1981)
18. Pous, D., Sangiorgi, D.: Enhancements of the bisimulation proof method. In: Advanced Topics in Bisimulation and Coinduction, pp. 233–289. Cambridge University Press (2012)
19. Rot, J., Bonchi, F., Bonsangue, M., Pous, D., Rutten, J., Silva, A.: Enhanced coalgebraic bisimulation, `http://www.liacs.nl/~jrot/up-to.pdf`
20. Rot, J., Bonsangue, M., Rutten, J.: Coalgebraic bisimulation-up-to. In: SOFSEM. LNCS, Springer (2013), to appear.
21. Rutten, J.: Automata and coinduction (an exercise in coalgebra). In: CONCUR. LNCS, vol. 1466, pp. 194–218. Springer (1998)
22. Rutten, J.: Universal coalgebra: a theory of systems. Theor. Comp. Sci. 249(1), 3–80 (2000)
23. Rutten, J.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theor. Comput. Sci. 308(1-3), 1–53 (2003)
24. Sangiorgi, D.: On the bisimulation proof method. Math. Struct. in Comp. Sci. 8(5), 447–479 (Oct 1998), `http://dx.doi.org/10.1017/S0960129598002527`
25. Turi, D., Plotkin, G.: Towards a mathematical operational semantics. In: LICS. pp. 280–291. IEEE Computer Society (1997)