Math. Struct. in Comp. Science (2015), *vol.* 25, *pp.* 765–804. © Cambridge University Press 2014
doi:10.1017/S0960129513000030 First published online 10 November 2014

Bisimulation proof methods in a path-based specification language for polynomial coalgebras[†]

XIAO-CONG ZHOU[‡], YONG-JI LI[‡], WEN-JUN LI[§], HAI-YAN QIAO[‡] and ZHONG-MEI SHU[‡]

[‡]*School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, P. R. China*
E-mail: isszxc@mail.sysu.edu.cn
[§]*School of Software, Sun Yat-sen University, Guangzhou 510275, P. R. China*

Received 28 April 2011; revised 31 January 2012

What reasoning rules can be used for the deduction of bisimulation formulas in coalgebraic specifications is problematic because those rules used in algebraic specifications possibly cannot be applied to bisimulation formulas. Although some categorical bisimulation proof methods for coalgebras have been proposed, they are not based on specification languages of coalgebras so that they cannot be used as reasoning rules. In this paper, a specification language based on paths of polynomial functors is proposed to specify polynomial coalgebras. Paths of polynomial functors give detailed observations and transitions on the state space of coalgebras so that the techniques used in transition system specifications can be applied to such a path-based language. In particular, because bisimulations can be characterized by paths, the notions of progressions, respectful functions and faithful contexts can be defined based on paths, and then bisimulation up-to proof techniques, including bisimulation up-to bisimilarities and up-to contexts for transition systems can be transformed into reasoning rules in the language. Several examples illustrate how to reason syntactically about bisimulations in the language by using the rules induced by the bisimulation proof techniques.

1. Introduction

During the last 10 years, coalgebraic methods have attracted attentions of many researchers in theoretical computer science. Coalgebras generalize automata, Kripke frames, labelled transition systems and many other mathematical models of state-based systems through providing a uniform and general framework to model such systems and reason about their properties. Recent developments on coalgebraic modal logic (Kupke and Pattinson 2011), coalgebraic automata theory (Kupke and Venema 2008), coalgebras in functional programming and type theory (Capretta 2011), and many other research directions, as witnessed by the proceedings of the CMCS (Coalgebraic Methods in Computer Science) and CALCO (International Conference on Algebra and Coalgebra in

[†] Supported by the National Natural Science Foundation of China under Grant No. 60673050 and the Fundamental Research Funds for the Central Universities of China under Grant No. 11LGPY39.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Computer Science), show that the theory of coalgebra plays more and more important role in theoretical computer science.

Coalgebraic specification languages are developed to specify dynamical state-based systems in the theory of coalgebras. One of the most accepted principles for specifying such systems is that the state space of a system is considered as a black box, and then the properties of the system are specified via its observational behaviours. In this situation, the specified operations of the system may only give limited access to the state space, and certain aspects may be unobservable. As a result, two states with the same observational behaviours need not be equal. When two states s, t are observationally indistinguishable, they are called *bisimilar* and written as $s \sim t$.

In general, specifications give an axiomatic description of mathematical structures in the sense that certain logical rules are needed to deduce more properties about structures from some initial assertions. Since the state space of a coalgebra is a black box, we should only specify that two states are bisimilar rather than equal in coalgebraic specifications, especially for classes in object-oriented programming (Jacobs 2002). And then, certain logical rules are needed to deduce bisimilarity formulas in coalgebraic specifications because equational logic rules possibly cannot be used any more. For example, what rules can be used to deduce $mirror(mirror(x)) \sim x$ from the initial assertions $lchild(mirror(x)) \sim mirror(rchild(x))$ and $rchil(mirror(x)) \sim mirror(lchild(x))$? Here $lchild, rchild, mirror$ are operations on an infinite binary tree, which give the left, right children and swap the children, respectively.

However, as to our knowledge, there is no work so far on the deduction of bisimilarity in coalgebraic specifications. Researchers often concentrate on proving certain states are bisimilar in a concrete coalgebra rather than in a general specification language. The aim of our work is to provide logical rules to reason syntactically about bisimilarity in a coalgebraic specification language. The key insight of our work is that the paths (Jacobs 2000; Rößiger 2001) of polynomial functors define by Rößiger and Jacobs can be used to define a language to specify coalgebras and reason about bisimilarity in a syntactic way.

The major technical contribution of this paper is to apply the idea of Sangiorgi's bisimulation up-to proof techniques (Sangiorgi 1998) to form logical rules to reason syntactically about bisimilarity in a path-based specification language for polynomial coalgebras. Two basic notions in Sangiorgi's work are *progression* and *sound function*. We use $P \xrightarrow{a} Q$ to denote a state transition of a labelled transition system, where P, Q are processes, and label a is drawn from some alphabet of actions. A binary relation R on processes *progresses* to a relation S, abbreviated $R \rightarrowtail S$, if $(P, Q) \in R$ implies:

i. whenever $P \xrightarrow{a} P'$, there is Q' such that $Q \xrightarrow{a} Q'$ and $(P', Q') \in S$;
ii. whenever $Q \xrightarrow{a} Q'$, there is P' such that $P \xrightarrow{a} P'$ and $(P', Q') \in S$.

A relation R is a *bisimulation* if and only if it is a self-progressing relation, i.e. $R \rightarrowtail R$, and two processes P, Q are bisimilar if there exists a bisimulation R such that $(P, Q) \in R$.

A function ϕ on relations is called a *sound function* if $R \rightarrowtail \phi(R)$ implies $R \subseteq \sim$ for all relations R, where $\sim$ is the largest bisimulation (i.e. the bisimilarity) on processes. By using a sound function ϕ, a bisimulation up-to ϕ proof technique can be used to demonstrate that two processes P and Q are bisimilar, that is, to find a relation R such that $(P, Q) \in R$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

and $R \rightarrowtail \phi(R)$. With the help of ϕ, the relation R can contain less state pairs, and then it is easier to find such a relation than to find a bisimulation directly.

Sangiorgi shows that the class of sound functions contains non-trivial functions and is closure under constructors including composition, union and iteration. And then many bisimulation up-to techniques are recovered as bisimulation up-to sound function techniques. In particular, a powerful class of sound functions provided by Sangiorgi is the class of *contextual functions* which give the closure of a relation under contexts. Sangiorgi shows that, in the case in which the transition relation among processes is defined structurally on the operators of the transition system specification language, certain conditions on the form of the transition rules ensure the soundness of the contextual function induced by the language. The proof technique induced by the contextual functions, which is called bisimulation up-to context, is proved to be a powerful proof method in practice for process algebras like CCS and pi-calculus (Sangiorgi and Walker 2001).

Although the bisimulation up-to techniques are effective for process algebras, they cannot be directly applied to general coalgebras because many coalgebras cannot be specified as labelled transition systems in a natural way. Generally, a coalgebra is a pair of $(X, \alpha : X \rightarrow \boldsymbol{T}X)$, where X is its state space, α is its transition structure, and T is its signature functor. However, α usually has complicated structures so that it cannot be specified by transition rules naturally. For example, it is not easy to specify the coalgebra $(X, \alpha : X \rightarrow A \times (1 + X \times X))$, which is used to model a kind of infinite trees, as a labelled transition system at first glance.

We observe that the paths (Jacobs 2000; Rößiger 2001) of polynomial functors give a decomposition of the transition structure of a polynomial coalgebra so that polynomial coalgebras can be easily fitted to the framework of transition system specifications. For example, the signature functor $\boldsymbol{T}X = A \times (1 + X \times X)$ of the above coalgebra has four paths which can be named as $label, nochild, lchild, rchild$. Using these paths, we can specify its coalgebra as in transition system specifications:

$$\frac{x \xrightarrow{lchild} x'}{mirror(x) \xrightarrow{rchild} mirror(x')} \qquad \frac{x \xrightarrow{rchild} x'}{mirror(x) \xrightarrow{lchild} mirror(x')}.$$

However, there are still major differences between transition system specifications and coalgebraic specifications. First, such coalgebras have many pure observations, such as $label, nochild$ in the above functor, which cannot be specified by transition rules naturally. For example, it is not easy to use transition rules to specify $mirror(x)$ and x have the same labels. Second, as we will explain later (see Example 4.28), the intended meanings of the transition rules in transition system specifications usually are state equations rather than state bisimilarities. For example, the above two transition rules are equivalent to

$$\begin{aligned} lchild(mirror(x)) &= mirror(rchild(x)) \\ rchild(mirror(x)) &= mirror(lchild(x)) \end{aligned}$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

rather than

$$lchild(mirror(x)) \sim mirror(rchild(x))$$
$$rchild(mirror(x)) \sim mirror(lchild(x)).$$

This means that transition system specifications do not regard the state space as a black box, which does not coincide with the principle of coalgebraic specifications.

We propose a path-based equational coalgebraic specification language to specify polynomial coalgebras. In this language, pure observational operations can be easily specified by observational equations, and we can use logical rules as in algebraic specification to reason about the properties of these operations. Moreover, we insist that we can only specify that two states are bisimilar rather than be equal in this language. Therefore, we applied bisimulation up-to techniques to the language to form logical rules for the deduction of bisimilarity. Because bisimulation on polynomial coalgebras can be characterized by paths of polynomial functors (Goldblatt 2001), a notion of progression based on paths of polynomial functors can be defined, and hence the bisimulation-up-to sound function proof techniques can be built for this language. In particular, we study the form of specifications on the coalgebra to ensure the soundness of the contextual functions induced by specifications. We introduce a notion of faithful contexts for the language, and transform the bisimulation up-to context proof technique into a reasoning rule to deduce bisimilarities in a syntactic way.

The remainder of this paper is organized as follows. Section 2 gives some basic notions of coalgebra. Section 3 defines the syntax and semantics of a path-based specification language for polynomial coalgebras. Section 4 investigates the bisimulation-up-to proof techniques, especially the notion of faithful context and the bisimulation-up-to context proof technique, and illustrates them with examples. Section 5 discusses related work. Finally, Section 6 concludes the paper and briefly discusses future work.

2. Preliminaries

In this section, we give some basic notions of coalgebra. We refer to Jacobs (2005); Rutten (2000) for more about the theory of coalgebras. Here, we focus on paths of polynomial functors and on how to use paths to characterize bisimulations.

2.1. *Coalgebras and bisimulations*

A functor $\boldsymbol{T} : \mathcal{S}et \to \mathcal{S}et$ on the category of sets is called a *polynomial functor*, if it is constructed from the identity functor and the constant functors by finitely many applications of products, coproducts and constant exponents (Goldblatt 2001):

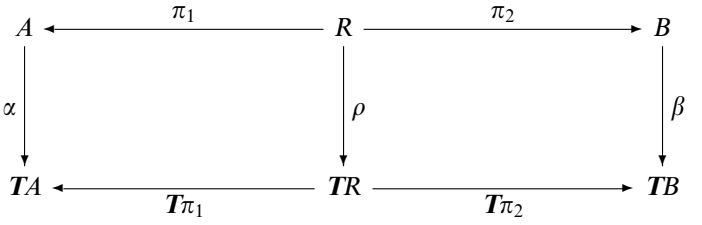
$$\boldsymbol{T} ::= \mathbf{Id} \mid \boldsymbol{K}_D \mid \boldsymbol{T} \times \boldsymbol{T} \mid \boldsymbol{T} + \boldsymbol{T} \mid \boldsymbol{T}^D.$$

Let $\boldsymbol{T} : \mathcal{S}et \to \mathcal{S}et$ be a functor. A $\boldsymbol{T}$-*coalgebra* is a pair (A, α), where A is a set, called the state space of the coalgebra, and $\alpha : A \to \boldsymbol{T}A$ is a function, called the transition structure of the coalgebra. A $\boldsymbol{T}$-coalgebra is called a *polynomial coalgebra* when $\boldsymbol{T}$ is a polynomial

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

functor. A coalgebra *homomorphism* from one coalgebra (A, α) to another coalgebra (B, β) is a function $f : A \to B$ such that $\beta \circ f = \boldsymbol{T}f \circ \alpha$.

If (A, α) and (B, β) are $\boldsymbol{T}$-coalgebras, then a relation $R \subseteq A \times B$ is a *bisimulation (relation)* from (A, α) to (B, β) if there exists a transition structure $\rho : R \to \boldsymbol{T}R$ on R such that the relation projections $\pi_1 : R \to A, \pi_2 : R \to B$ are coalgebra homomorphisms from (R, ρ) to (A, α) and (B, β), respectively, i.e. the following diagram commutes.



The above definition of bisimulation is based on coalgebra homomorphisms. The notion of bisimulation can also be defined by using relation lifting. Let $\boldsymbol{T} : \mathcal{S}et \to \mathcal{S}et$ be a functor. The *relation lifting* of a relation $R \subseteq A \times B$ is a relation $Rel(\boldsymbol{T})(R) \subseteq \boldsymbol{T}A \times \boldsymbol{T}B$, defined as (Hughes and Jacobs 2004)

$$Rel(\boldsymbol{T})(R) \stackrel{\text{def}}{=} \{(u, v) \in \boldsymbol{T}A \times \boldsymbol{T}B \mid \exists w \in \boldsymbol{T}R, \boldsymbol{T}\pi_1(w) = u \wedge \boldsymbol{T}\pi_2(w) = v\}.$$

Let (A, α) and (B, β) be two $\boldsymbol{T}$-coalgebras. A relation $R \subseteq A \times B$ is a *bisimulation* if, for all $x \in A, y \in B$, $(x, y) \in R$ implies $(\alpha(x), \beta(y)) \in Rel(\boldsymbol{T})(R)$ (Jacobs 2005). It is not hard to show that the above two definitions of bisimulation are equivalent for polynomial coalgebras (c.f. Theorem 3.3.2 in Jacobs (2005)).

2.2. *Using paths to characterize bisimulations*

In this paper, we focus on polynomial coalgebras because they are simple but enough to capture many of the interesting examples in computer science. For example, both automata and classes in object-oriented programming languages are polynomial coalgebras. In addition, polynomial functors allow us to define the notion of path, which is the cornerstone of this paper. The notion of path of polynomial functors is first studied by Rößiger in (Rößiger 1999, 2001), and explored by Jacobs in details (Jacobs 1999, 2000).

A *path* from one polynomial functor $\boldsymbol{T}$ to another polynomial functor $\boldsymbol{S}$, denoted as $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$, is a finite list of symbols $\pi_1, \pi_2, \varepsilon_1, \varepsilon_2, ev(d)$, for elements $d \in D$ of sets D occurring as exponent in T. Let $\boldsymbol{T}$ and $\boldsymbol{S}$ be polynomial functors. The set of paths from $\boldsymbol{T}$ to $\boldsymbol{S}$ is the least set generated by the following clauses.

1. $\langle\rangle : \boldsymbol{T} \rightsquigarrow \boldsymbol{T}$, where $\langle\rangle$ is the empty list.
2. $\pi_i \cdot p : \boldsymbol{T}_1 \times \boldsymbol{T}_2 \rightsquigarrow \boldsymbol{S}$ for $p : \boldsymbol{T}_i \rightsquigarrow \boldsymbol{S}$, where $i = 1, 2$.
3. $\varepsilon_i \cdot p : \boldsymbol{T}_1 + \boldsymbol{T}_2 \rightsquigarrow \boldsymbol{S}$ for $p : \boldsymbol{T}_i \rightsquigarrow \boldsymbol{S}$, where $i = 1, 2$.
4. $ev(d) \cdot p : \boldsymbol{T}_1^D \rightsquigarrow \boldsymbol{S}$ for all $d \in D$ and $p : \boldsymbol{T}_1 \rightsquigarrow \boldsymbol{S}$.

A path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$ is a *state path* if $\boldsymbol{S} = \mathbf{Id}$, and an *observation path* if $\boldsymbol{S} = \boldsymbol{K}_D$ for some set D. The set of all state paths from $\boldsymbol{T}$ is denoted as $\texttt{SPATH}(\boldsymbol{T})$, the set of all observation paths from $\boldsymbol{T}$ is denoted as $\texttt{OPATH}(\boldsymbol{T})$, and their union is denoted as $\texttt{OSPATH}(\boldsymbol{T})$.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

For each set A, a partial function from $\boldsymbol{T}A$ to $\boldsymbol{S}A$ is defined inductively in Goldblatt (2001) for a path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$. However, here we lift such partial functions to (total) functions because we need (total) functions to give semantics of the typed terms in the sequel.

We assume that, for each set A, there exists an element $\perp$ which is not in A, and then A can be lifted to $A_\perp = A \cup \{\perp\}$. A partial function $f : A \rightharpoonup B$ can be lifted to a (total) function $f_\perp : A_\perp \to B_\perp$ in a straightforward way, i.e. for any $x \in A_\perp$, if $x \in \mathbf{dom}(f)$ then $f_\perp(x) = f(x)$, otherwise $f_\perp(x) = \perp$. Moreover, a (total) function $f : A \to B$ can also be lifted to $f_\perp : A_\perp \to B_\perp$, which is defined as, $f_\perp(\perp) = \perp$, and $f_\perp(x) = f(x)$ for any $x \in A$. It is not hard to show that, for two partial or total functions $f : A \to B, g : B \to C$, $(g \circ f)_\perp = g_\perp \circ f_\perp$. For any function $f : A_\perp \to B_\perp$, we define $\mathbf{wdom}(f) = \{x \in A_\perp \mid f(x) \neq \perp\}$, called the *well-defined domain* of f. Similarly, for any relation $R \subseteq A \times B$, we define its lifting as $R_\perp = R \cup \{(\perp, \perp)\}$. Obviously, if $R \subseteq A \times A$ is an equivalence relation, then $R_\perp$ is an equivalence relation on $A_\perp$.

We can now lift the partial functions for paths in Goldblatt (2001) to (total) functions. A path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$ induces a function $\widehat{p}_A : (\boldsymbol{T}A)_\perp \to (\boldsymbol{S}A)_\perp$ for each set A, defined by induction on the length of p as follows.

1. $\widehat{\langle\rangle}_A : (\boldsymbol{T}A)_\perp \to (\boldsymbol{T}A)_\perp$ is the identity function $id_{(\boldsymbol{T}A)_\perp}$.
2. $\widehat{(\pi_j \cdot p)}_A = \widehat{p}_A \circ (\pi_j)_\perp (j = 1, 2)$, where $(\pi_j)_\perp : (\boldsymbol{T}_1 A \times \boldsymbol{T}_2 A)_\perp \to (\boldsymbol{T}_j A)_\perp$ is the lifting of the product projection.
3. $\widehat{(\varepsilon_j \cdot p)}_A = \widehat{p}_A \circ (\varepsilon_j)_\perp (j = 1, 2)$, where $(\varepsilon_j)_\perp : (\boldsymbol{T}_1 A + \boldsymbol{T}_2 A)_\perp \to (\boldsymbol{T}_j A)_\perp$ is the lifting of the coproduct extraction defined in Goldblatt (2001), i.e. if $y = \kappa_j\ x$ for some $x \in \boldsymbol{T}A_j$ then $(\varepsilon_j)_\perp(y) = x$, otherwise $(\varepsilon_j)_\perp(y) = \perp$, where κ_j is the coproduct injections.
4. $\widehat{(ev(d) \cdot p)}_A = \widehat{p}_A \circ (ev_d)_\perp$, where $(ev_d)_\perp : ((\boldsymbol{T}A)^D)_\perp \to (\boldsymbol{T}A)_\perp$ is defined as, $(ev_d)_\perp(\perp) = \perp$, and $(ev_d)_\perp(f) = f(d)$ for each $f \in (\boldsymbol{T}A)^D$.

Note that only when ε_1 or ε_2 occur in a path, the function induced by the path may yield the value $\perp$. Indeed, a state path p does a transition on the state space A of the coalgebra, i.e. from a state x to $\widehat{p}_A(\alpha(x))$, while an observation path p does a pure observation on the state space, i.e. giving an observational value $\widehat{p}_A(\alpha(x))$. Given $\boldsymbol{T}A = \boldsymbol{T}_1 A + \boldsymbol{T}_2 A$, a path with ε_1 (or ε_2) as an ingredient is not a valid transition or observation on a state x with $\alpha(x) \in \boldsymbol{T}_2 A$ ($\alpha(x) \in \boldsymbol{T}_1 A$). This is why, we have to use $\perp$ to lift the functions induced by paths.

Lemma 2.1. Let $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$ be a path. The functions induced by p are natural, that is, for any function $f : A \to B$, the following diagram commutes.

$$\begin{array}{ccc} (\boldsymbol{T}A)_\perp & \xrightarrow{\widehat{p}_A} & (\boldsymbol{S}A)_\perp \\ \downarrow{\scriptstyle (\boldsymbol{T}f)_\perp} & & \downarrow{\scriptstyle (\boldsymbol{S}f)_\perp} \\ (\boldsymbol{T}B)_\perp & \xrightarrow[\widehat{p}_B]{} & (\boldsymbol{S}B)_\perp \end{array}$$

Proof. By induction on paths. □

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Jacobs observed that a coalgebra can be decomposed and assembled according to observational and state paths (Jacobs 2000). Let $\boldsymbol{T}$ be a polynomial functor. A $\boldsymbol{T}$-decomposition of a set X is a collection of subsets $(X_p)_{p\in\mathtt{OSPATH}(\boldsymbol{T})}$, satisfying:

1. X is covered by $(X_p)_{p\in\mathtt{OSPATH}(\boldsymbol{T})}$:
$$\bigcup_{p\in\mathtt{OSPATH}(\boldsymbol{T})} X_p = X.$$
2. For each path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{T}_1 \times \boldsymbol{T}_2$,
$$\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1\times\boldsymbol{T}_2)} X_{p\cdot q} = \bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1)} X_{p\cdot\pi_1\cdot q}$$
$$\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1\times\boldsymbol{T}_2)} X_{p\cdot q} = \bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_2)} X_{p\cdot\pi_2\cdot q}.$$
3. For each path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{T}_1 + \boldsymbol{T}_2$,
$$\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1+\boldsymbol{T}_2)} X_{p\cdot q} = \bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1)} X_{p\cdot\varepsilon_1\cdot q} \oplus \bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_2)} X_{p\cdot\varepsilon_2\cdot q},$$
where $\oplus$ is the operator of set symmetric differences.
4. For each path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{T}_1^D$, and $d \in D$,
$$\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1^D)} X_{p\cdot q} = \bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{T}_1)} X_{p\cdot ev(d)\cdot q}.$$

Jacobs showed that a coalgebra can be assembled by a $\boldsymbol{T}$-decomposition:

Proposition 2.2. Let $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{S}$ be a path, and $(X_q)_{q\in\mathtt{OSPATH}(\boldsymbol{T})}$ be a $\boldsymbol{T}$-decomposition. Assume that there exists a function $c_q : X_q \to X$ for each state path $q : \boldsymbol{T} \rightsquigarrow \mathbf{Id}$, and there exists a function $c_q : X_q \to D$ for each observation path $q : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$. Then,

1. A function $c_p : \left(\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{S})} X_{p\cdot q}\right) \to \boldsymbol{S}X$ can be assembled from these data.
2. In particular, for $p = \langle\rangle$ and thus $\boldsymbol{T} = \boldsymbol{S}$, a coalgebra $c_{\langle\rangle} : X \to \boldsymbol{T}X$ results.

Proof. See Proposition 7.4 in Jacobs (2000). □

More interestingly, given a $\boldsymbol{T}$-coalgebra (X, α), we can define a $\boldsymbol{T}$-decomposition, from which the coalgebra itself can be assembled.

Proposition 2.3. Let (X, α) be a $\boldsymbol{T}$-coalgebra. We define:
$$(X_p)_{p\in\mathtt{OSPATH}(\boldsymbol{T})} \stackrel{\text{def}}{=} \{\mathbf{wdom}(\widehat{p}_X \circ \alpha_\perp) \mid p \in \mathtt{OSPATH}(\boldsymbol{T})\}.$$
Then, $(X_p)_{p\in\mathtt{OSPATH}(\boldsymbol{T})}$ is a $\boldsymbol{T}$-decomposition of X. Moreover, for each state path $q : \boldsymbol{T} \rightsquigarrow \mathbf{Id}$, we have a function $\widehat{q}_X \circ \alpha_\perp : \mathbf{wdom}(\widehat{q}_X \circ \alpha_\perp) \to X$ and for each observation path $q : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$, we have a function $\widehat{q}_X \circ \alpha_\perp : \mathbf{wdom}(\widehat{q}_X \circ \alpha_\perp) \to D$. From these data, the function $c_p : (\bigcup_{q\in\mathtt{OSPATH}(\boldsymbol{S})} \mathbf{wdom}((\widehat{p\cdot q})_X \circ \alpha_\perp)) \to \boldsymbol{S}X$ assembled as in Proposition 2.2 is $\widehat{p}_X \circ \alpha_\perp$. In particular, for $p = \langle\rangle$, $c_p = \alpha$.

Proof. See Lemmas 6.4, 7.1 and 7.5 in Jacobs (2000). Although the notations used here is different from those used in (Jacobs 2000), it is easy to prove this proposition by

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

rewriting the proofs in Jacobs (2000) if only we notice that $\neg[\![p]\!](\bot)$ in Jacobs (2000) is equivalent to $\mathbf{wdom}(\widehat{p}_X \circ \alpha_\bot)$ here. The function $\widehat{p}_X \circ \alpha_\bot$ will be very important for this paper, so that we use a shorter notation $p^\sharp_\alpha$ to represent it. □

The next result given in Jacobs (2000) describes how to define a homomorphism between two assembled coalgebras.

Proposition 2.4. Suppose, we have a pair of sets X, Y, with $\boldsymbol{T}$-decompositions $(X_p)_{p\in \mathtt{OSPATH}(\boldsymbol{T})}$ and $(Y_p)_{p\in \mathtt{OSPATH}(\boldsymbol{T})}$ and functions $c_p : X_p \to X, d_p : Y_p \to Y$ for each state path $p : \boldsymbol{T} \rightsquigarrow \mathbf{Id}$, and $c_p : X_p \to D, d_p : Y_p \to D$ for each observation path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$. If a function $f : X \to Y$ restricts to $X_p \to Y_p$ and satisfies

$$\begin{array}{ccc} X_p & \xrightarrow{f} & Y_p \\ {\scriptstyle c_p}\downarrow & & \downarrow{\scriptstyle d_p} \\ X & \xrightarrow[f]{} & Y \end{array} \qquad \begin{array}{ccc} X_p & \xrightarrow{f} & Y_p \\ {\scriptstyle c_p}\downarrow & & \downarrow{\scriptstyle d_p} \\ D & \xrightarrow[id_D]{} & D \end{array}$$

for each $p \in \mathtt{OSPATH}(\boldsymbol{T})$, then f is a homomorphism from $c_{\langle\rangle} : X \to \boldsymbol{T}X$ to $d_{\langle\rangle} : Y \to \boldsymbol{T}Y$ between the assembled coalgebras.

Proof. See Lemma 7.6 in (Jacobs 2000). □

From these results, we obtain a useful way to define a colagebra structure on a set.

Proposition 2.5. Let $\boldsymbol{T}$ be a polynomial functor, and (Y, β) be a $\boldsymbol{T}$-coalgebra. If there exists a function $f : X \to Y$ that satisfies: (1) for each state path $q : \boldsymbol{T} \rightsquigarrow \mathbf{Id} \in \mathtt{SPATH}(\boldsymbol{T})$, there exists a function $\varphi_q : X_\bot \to X_\bot$ such that $q^\sharp_\beta \circ f_\bot = f_\bot \circ \varphi_q$; (2) for each observation path $q : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D \in \mathtt{SPATH}(\boldsymbol{T})$, there exists a function $\varphi_q : X_\bot \to D_\bot$ such that $q^\sharp_\beta \circ f_\bot = \varphi_q$. Then, a coalgebra structure $\alpha : X \to \boldsymbol{T}X$ can be defined on X, and the function $f : X \to Y$ is a homomorphism from (X, α) to (Y, β).

Proof. Because the collection of subsets $(\mathbf{wdom}(\varphi_q))_{q\in \mathtt{OSPATH}(\boldsymbol{T})}$ is a $\boldsymbol{T}$-decomposition, and the function $f : X \to Y$ can be restricted to $f : \mathbf{wdom}(\varphi_q) \to \mathbf{wdom}(q^\sharp_\alpha)$ and satisfies the diagrams in Proposition 2.4. □

Using Proposition 2.5, we can prove the following important result, which is first given by Goldblatt (c.f. Theorem 5.5 in Goldblatt (2001)). It shows that a bisimulation can be characterized as a relation that is 'preserved' by the functions induced by state and observation paths.

Proposition 2.6. Let (X, α) and (Y, β) be two $\boldsymbol{T}$-coalgebras. Then a relation $R \subseteq X \times Y$ is a bisimulation if and only if, for any $x \in X$ and $y \in Y$, $x\ R\ y$ implies

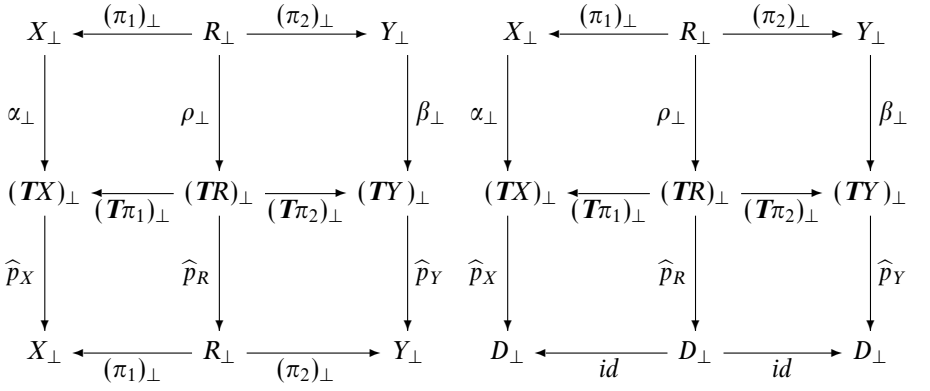
1. for each path $p \in \mathtt{OSPATH}(\boldsymbol{T})$, $x \in \mathbf{wdom}(p^\sharp_\alpha)$ iff $y \in \mathbf{wdom}(p^\sharp_\beta)$;
2. for each observation path $p \in \mathtt{OPATH}(\boldsymbol{T})$, $p^\sharp_\alpha(x) = p^\sharp_\beta(y)$;

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

3. for each state path $p \in \mathtt{SPATH}(\boldsymbol{T})$, if $x \in \mathbf{wdom}(p^\sharp_\alpha)$ and $y \in \mathbf{wdom}(p^\sharp_\beta)$, then $\langle p^\sharp_\alpha(x), p^\sharp_\beta(y)\rangle \in R$.

Proof. Goldblatt's proof in Goldblatt (2001) is based on the relation lifting definition of bisimulation. Here, we use the technique of decomposition and assembly of polynomial coalgebras as shown above to give a proof based on the homomorphism definition of bisimulation.

($\Longrightarrow$). If R is a bisimulation, then there exists a coalgebra structure $\rho : R \to \boldsymbol{T}R$. From the fact that the functions induced by paths are natural (c.f. Lemma 2.1), for each state path $p : \boldsymbol{T} \rightsquigarrow \mathbf{Id}$, the left diagram below commutes, and for each observation path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$, the right diagram below commutes



From these diagrams, it is easy to show (1), (2) and (3) in the proposition.

($\Longleftarrow$). We define a collection of functions $(\varphi_p)_{p\in\mathtt{OSPATH}(\boldsymbol{T})}$ on R as follows.

1. For each state path $p \in \mathtt{SPATH}(\boldsymbol{T})$, for any $\langle x, y\rangle \in R$,

$$\varphi_p(\langle x, y\rangle) \;=\; \begin{cases} \langle p^\sharp_\alpha(x), p^\sharp_\beta(y)\rangle & \text{if } x \in \mathbf{wdom}(p^\sharp_\alpha) \text{ and } y \in \mathbf{wdom}(p^\sharp_\beta) \\ \perp & \text{otherwise.} \end{cases}$$

   It is easy to show that φ_p is a well-defined function by using the conditions (1) and (3), and it satisfies $p^\sharp_\alpha \circ (\pi_1)_\perp = (\pi_1)_\perp \circ \varphi_p$ and $p^\sharp_\beta \circ (\pi_2)_\perp = (\pi_2)_\perp \circ \varphi_p$.

2. For each observation path $p : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$, for any $\langle x, y\rangle \in R$,

$$\varphi_p(\langle x, y\rangle) \;=\; \begin{cases} p^\sharp_\alpha(x) & \text{if } x \in \mathbf{wdom}(p^\sharp_\alpha) \\ \perp & \text{otherwise.} \end{cases}$$

   It is also easy to show that φ_p is a well-defined function by using the condition (2), and it satisfies $p^\sharp_\alpha \circ (\pi_1)_\perp = \varphi_p$ and $p^\sharp_\beta \circ (\pi_2)_\perp = \varphi_p$.

Then, by Proposition 2.5, we can define a coalgebra structure ρ on R, and it makes π_1 and π_2 two homomorphisms. □

The notions of lifting can simplify our notations. For example, the above conditions (1), (2) and (3) can be simplified to the following two conditions: (i) for each observation path p, $p^\sharp_\alpha(x) = p^\sharp_\beta(y)$, and (ii) for each state path p, $(p^\sharp_\alpha(x), p^\sharp_\beta(y)) \in R_\perp$.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

3. A path-based language for polynomial coalgebras

In this section, we present a specification language for specifying polynomial coalgebras. Like algebraic specification languages, a signature over a polynomial functor is used to construct terms of the language, and then the terms are used to form formulas according to an equational approach. A distinctive feature of the language is that paths are used as function symbols to construct terms. Observation paths are used to specify pure observations, and state paths are used to specify state transitions on the state space of polynomial coalgebras.

3.1. *Syntax and semantics of terms*

For a polynomial functor $\boldsymbol{T}$, let $\mathbb{O}_{\boldsymbol{T}}$ be the set of constant sets occurring in the observation paths of $\boldsymbol{T}$, i.e. $\mathbb{O}_{\boldsymbol{T}} = \{D \mid \exists p : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D \in \mathtt{OPATH}(\boldsymbol{T})\}$. A *signature* over a polynomial functor $\boldsymbol{T}$ is a pair $\mathsf{Sg}_{\boldsymbol{T}} = (\mathsf{Sort}_{\boldsymbol{T}}, \mathsf{Op}_{\boldsymbol{T}})$, where

1. $\mathsf{Sort}_{\boldsymbol{T}}$, called *types* of the signature, satisfies that (i) $\mathsf{State} \in \mathsf{Sort}_{\boldsymbol{T}}$, i.e. there exists a type State; (ii) for each $D \in \mathbb{O}_{\boldsymbol{T}}$, there exists a type $\sigma_D \in \mathsf{Sort}_{\boldsymbol{T}}$.
2. $\mathsf{Op}_{\boldsymbol{T}}$ is a set of *function symbols*, each with a *signature* $f : \sigma_1, \sigma_2, \ldots, \sigma_n \rightarrow \sigma$, where $\sigma_i, \sigma \in \mathsf{Sort}_{\boldsymbol{T}}$, and σ is called the *result type* of f. n is the arity of f, and f is called *constant symbol* when $n = 0$.

We often omit the subscribe $\boldsymbol{T}$ for convenience. The type State is called *state type*, and all other types are called *observation types*.

To define *terms* we fix a denumerable set **Var** of *variables*. The 'raw' terms (i.e. untyped terms) are defined by the following abstract syntax:

$$M ::= x \mid k \mid f(M_1, \ldots, M_n) \mid p^{\sharp}(M),$$

where $x \in \mathbf{Var}$ is a variable, k is a constant symbol, f is a n-ary function symbols, and p is a state or observation path.

Given a raw term M, the set of variables occurring in M is denoted as $fv(M)$. Given a variable x and a raw term N, the substitution of N for a variable x is denoted as $M[N/x]$, which can be defined easily by induction. In general, we use $M[N_1/x_1, \ldots, M_n/x_n]$, or shortly $M[\overrightarrow{N}/\overrightarrow{x}]$, to denote the simultaneous substitution of $N_1, \ldots, N_n$ for variables $x_1, \ldots, x_n$.

To assign a type to a raw term, we define a *context* to be a finite (possible empty) list $\Gamma = [x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n]$ of assignments of types σ_i to variable x_i, with the proviso that $x_1, x_2, \ldots, x_n$ are all distinct. For convenience, we use $\mathbf{dom}(\Gamma)$ to denote the set of all variables in Γ, and use $\Gamma(x)$ to denote the type assigned to the variable x in Γ.

A *term-in-context* is an expression of the form $\Gamma \rhd M : \sigma$, which means that M is a 'raw' term of type σ in context Γ. This may be abbreviated to $\Gamma \rhd M$ when the type of the term is understood. The following rules give how to generate terms-in-context, i.e. how to

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

assign types to raw terms.

$$(\textit{Variables})\quad \frac{}{\Gamma, x:\sigma, \Gamma' \rhd x:\sigma}$$

$$(\textit{Constants})\quad \frac{}{\Gamma \rhd k:\sigma}\ (k:\sigma)$$

$$(\textit{Function Symbols})\quad \frac{\Gamma \rhd M_1:\sigma_1, \ldots, \Gamma \rhd M_n:\sigma_n}{\Gamma \rhd f(M_1,\ldots,M_n):\sigma}\ (f:\sigma_1,\ldots,\sigma_n \to \sigma)$$

$$(\textit{Observation paths})\quad \frac{\Gamma \rhd M:\mathsf{State}}{\Gamma \rhd p^\sharp(M):\sigma_D}\ (p:\boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D)$$

$$(\textit{State paths})\quad \frac{\Gamma \rhd M:\mathsf{State}}{\Gamma \rhd p^\sharp(M):\mathsf{State}}\ (p:\boldsymbol{T} \rightsquigarrow \mathbf{Id}).$$

In the sequel, a term-in-context is called a term for simplicity. For a term $\Gamma \rhd M:\sigma$, if $\sigma = \mathsf{State}$, we call it *state term*, otherwise we call it *observation term*. It is easy to show that if we obtain $\Gamma \rhd M:\sigma$ and $\Gamma \rhd M:\sigma'$ then $\sigma = \sigma'$. In other words, the type assigned to a term is unique. By induction on terms, we can also prove the following lemma easily.

Lemma 3.1. We have the following derived rules.

1. *Permutation rule*. If $\pi\Gamma$ is a permutation of Γ, then
$$\frac{\Gamma \rhd M:\sigma}{\pi\Gamma \rhd M:\sigma}.$$
2. *Weakening rule*. If $\Gamma \subseteq \Gamma'$, then
$$\frac{\Gamma \rhd M:\sigma}{\Gamma' \rhd M:\sigma}.$$
3. *Substitution rule*
$$\frac{\Gamma, x:\sigma, \Gamma' \rhd M:\tau \qquad \Gamma \rhd N:\sigma}{\Gamma, \Gamma' \rhd M[N/x]:\tau}.$$
Generally, let $\Gamma' = [x_1:\sigma_1, x_2:\sigma_2, \ldots, x_n:\sigma_n]$, and then
$$\frac{\Gamma' \rhd M:\sigma \quad \Gamma \rhd N_1:\sigma_1,\ \Gamma \rhd N_2:\sigma_2,\ldots,\ \Gamma \rhd N_n:\sigma_n}{\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}]:\sigma}.$$

Proof. By induction on terms. □

A *semantic model* of the signature Sg is a tuple $\mathcal{M} = (\mathcal{X}, I)$, where $\mathcal{X} = (X, \alpha : X \to \boldsymbol{T}X)$ is a $\boldsymbol{T}$-coalgebra, and I is an interpretation of the signature Sg, i.e. for each type $\sigma \in \mathsf{Sort}$, there is a set $[\![\sigma]\!]_{\mathcal{M}}$ as the interpretation of σ, and for each function symbol $f:\sigma_1,\ldots,\sigma_n \to \sigma$, there is a function

$$[\![f]\!]_{\mathcal{M}} : [\![\sigma_1]\!]_{\mathcal{M}} \times \cdots \times [\![\sigma_n]\!]_{\mathcal{M}} \to [\![\sigma]\!]_{\mathcal{M}}$$

as the interpretation of f. Moreover, I satisfies: (i) if $\sigma = \mathsf{State}$, then $[\![\sigma]\!]_{\mathcal{M}} = X_\perp$, (ii) if $\sigma = \sigma_D$ for a set D occurring in an observation path $p:\boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D$, then $[\![\sigma]\!]_{\mathcal{M}} = D_\perp$, and (iii) for any other type σ, there is a set A such that $[\![\sigma]\!] = A_\perp$.

Given a semantic model $\mathcal{M} = (\mathcal{X}, I)$, the semantics of a term $\Gamma \rhd M:\sigma$ is a function

$$[\![\Gamma \rhd M:\sigma]\!]_{\mathcal{M}} \ :\ [\![\Gamma]\!]_{\mathcal{M}} \to [\![\sigma]\!]_{\mathcal{M}},$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

where we assume that $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, and then

$$[\![\Gamma]\!]_{\mathcal{M}} = [\![\sigma_1]\!]_{\mathcal{M}} \times \cdots \times [\![\sigma_n]\!]_{\mathcal{M}}.$$

The semantics of terms can be inductively defined as follows (for convenience, we omit the subscript $\mathcal{M}$).

$$\frac{}{[\![\Gamma, x : \sigma, \Gamma' \rhd x : \sigma]\!] = \pi : [\![\Gamma]\!] \times [\![\sigma]\!] \times [\![\Gamma']\!] \to [\![\sigma]\!]}$$

$$\frac{}{[\![\Gamma \rhd k : \sigma]\!] = [\![k]\!] \circ\ ! : [\![\Gamma]\!] \to 1 \to [\![\sigma]\!]}$$

$$\frac{\varphi_1 = [\![\Gamma \rhd M_1 : \sigma_1]\!], \ldots, \varphi_n = [\![\Gamma \rhd M_n : \sigma_n]\!]}{[\![\Gamma \rhd f(M_1, \ldots, M_n) : \sigma]\!] = [\![f]\!] \circ \langle \varphi_1, \ldots, \varphi_n \rangle : [\![\Gamma]\!] \to [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \to [\![\sigma]\!]}$$

$$\frac{\varphi = [\![\Gamma \rhd M : \mathsf{State}]\!] : [\![\Gamma]\!] \to X_\perp}{[\![\Gamma \rhd p^\sharp(M) : \sigma_D]\!] = p_\alpha^\sharp \circ \varphi : [\![\Gamma]\!] \to X_\perp \to (\boldsymbol{T}X)_\perp \to D_\perp} \quad (p : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_D)$$

$$\frac{\varphi = [\![\Gamma \rhd M : \mathsf{State}]\!] : [\![\Gamma]\!] \to X_\perp}{[\![\Gamma \rhd p^\sharp(M) : \mathsf{State}]\!] = p_\alpha^\sharp \circ \varphi : [\![\Gamma]\!] \to X_\perp \to (\boldsymbol{T}X)_\perp \to X_\perp} \quad (p : \boldsymbol{T} \rightsquigarrow \mathbf{Id}).$$

The next theorem gives an important property of the semantics of terms.

Theorem 3.2. Let $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, and $\Gamma' \rhd M : \sigma$ be a term. Then

$$[\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \sigma]\!] = [\![\Gamma' \rhd M : \sigma]\!] \circ \langle [\![\Gamma \rhd N_1 : \sigma_1]\!], \ldots, [\![\Gamma \rhd N_n : \sigma_n]\!] \rangle.$$

Proof. By induction on $\Gamma' \rhd M : \sigma$. □

Unlike our earlier work (Zhou *et al.* 2010), here we present a typed version of the specification language. There are several reasons motivating us to introduce typed version of the language. First, a term-in-context (and a formula-in-context in the sequel) is easier to understand than an untyped term/formula since the types of the variables occurring in the term give extra information. Second, the semantics of typed terms and formulas can be given in a more elegant way. In our earlier work (Zhou *et al.* 2010), we have to use partial functions in an assignment for the variables, and do a special treatment on the assignments when we want to give the semantics of substitutions. Here, we can give the semantics of terms and substitutions as usual in many other type systems. Last but not the least, we may need to extend the language to use constructors on types. For example, we may need to use *transfer* : $\mathsf{State}, \mathsf{State}, \mathbb{R} \to \mathsf{State} \times \mathsf{State}$ to specify the operator *transfer* on the class Account as follows.

$$x : \mathsf{State}, y : \mathsf{State}, r : \mathbb{R} \rhd balance(\pi_1(transfer(x, y, r))) \approx (balance(x) - r);$$
$$x : \mathsf{State}, y : \mathsf{State}, r : \mathbb{R} \rhd balance(\pi_2(transfer(x, y, r))) \approx (balance(y) + r).$$

These formulas specify the changes of the balances of two accounts x, y after transferring amount r from x to y. Obviously, a typed language is easier to support such extension.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

3.2. *Syntax and semantics of formulas*

Like algebraic specification languages, we use an equational approach to construct formulas for our coalgebraic specification language. However, a major difference between our language and algebraic languages is that we specify that two state terms are bisimilar rather than equal since the state space of a coalgebra is viewed as a black box. Moreover, because the semantic function of a term may leads to an undefined value (i.e. $\perp$), we need a formula of the form $M \Downarrow$ to specify whether the semantic function gives $\perp$ or not. Therefore, the raw formulas of the language are defined by the following abstract syntax.

$$\varphi, \psi \ ::= \ M \Downarrow : \sigma \mid M_1 \approx M_2 : \sigma \mid \neg\varphi \mid \varphi \to \psi,$$

where M, M_1, M_2 are raw terms. We can also introduce other boolean operators (i.e. $\wedge, \vee, \leftrightarrow$) by using the standard way. Unlike our previous paper (Zhou *et al.* 2010) using different operators $\approx$ and $\simeq$ respectively to specify that two observation terms are equal and two state terms are bisimlar, here we use the same operator $\approx$ to specify these two cases since the types of the terms are enough to distinct them.

A formula-in-context has the form of $\Gamma \rhd \varphi$, which is generated inductively by the following rules.

$$\frac{\Gamma \rhd M : \sigma}{\Gamma \rhd M \Downarrow : \sigma} \qquad \frac{\Gamma \rhd M_1 : \sigma \qquad \Gamma \rhd M_2 : \sigma}{\Gamma \rhd M_1 \approx M_2 \ : \ \sigma}$$

$$\frac{\Gamma \rhd \varphi}{\Gamma \rhd \neg\varphi} \qquad \frac{\Gamma \rhd \varphi \qquad \Gamma \rhd \psi}{\Gamma \rhd \varphi \to \psi}.$$

A formula $\Gamma \rhd M \Downarrow : \sigma$ is called a *term definition formula*, while a formula $\Gamma \rhd M_1 \approx M_2 : \sigma$ is called a *state bisimulation formula* when $\sigma = \mathsf{State}$; in other cases, it is called an *observation equation formula.*

Similarly, a formula-in-context will be called a formula for simplicity. The formulas $\Gamma \rhd M \Downarrow : \sigma$ and $\Gamma \rhd M \approx M' : \sigma$ are often abbreviated to $\Gamma \rhd M \Downarrow$ and $\Gamma \rhd M \approx M'$, respectively. Moreover, for convenience, we use $\Gamma \rhd M \Downarrow M' : \sigma$ as the abbreviation of $\Gamma \rhd M \Downarrow : \sigma \leftrightarrow M' \Downarrow : \sigma$. In general, we use $\Gamma \rhd M \Downarrow (M_1, \ldots, M_n) : \sigma$ as the abbreviation of $\Gamma \rhd M \Downarrow : \sigma \leftrightarrow M_1 \Downarrow : \sigma \wedge \cdots \wedge M_n \Downarrow : \sigma$.

We use $\varphi[N_1/x_1, \ldots, N_n/x_n]$, or shortly $\varphi[\overrightarrow{N}/\overrightarrow{x}]$, to denote the simultaneous substitution of $N_1, \ldots, N_n$ for variables $x_1, \ldots, x_n$. Assume that $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$. It is not hard to show that $\Gamma' \rhd \varphi$ and $\Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n$ imply that $\Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]$.

Let $\mathcal{M} = ((X, \alpha), I)$ be a semantic model of the signature. The semantics of a formula $\Gamma \rhd \varphi$ is a function $[\![\Gamma \rhd \varphi]\!] : [\![\Gamma]\!] \to \mathbf{2} (= \{\mathsf{true}, \mathsf{false}\})$, which can be inductively defined as follows.

1. $[\![\Gamma \rhd M \Downarrow : \sigma]\!] : [\![\Gamma]\!] \to \mathbf{2}$ is defined as, for any $t \in [\![\Gamma]\!]$,

$$[\![\Gamma \rhd M \Downarrow : \sigma]\!](t) = \mathsf{true} \ \text{ iff } \ [\![\Gamma \rhd M : \sigma]\!](t) \neq \perp.$$

2. If $\sigma \neq \mathsf{State}$, $[\![\Gamma \rhd M_1 \approx M_2 : \sigma]\!] : [\![\Gamma]\!] \to \mathbf{2}$ is defined as, for any $t \in [\![\Gamma]\!]$,

$$[\![\Gamma \rhd M_1 \approx M_2 : \sigma]\!](t) = \mathsf{true} \ \text{ iff } \ [\![\Gamma \rhd M_1 : \sigma]\!](t) = [\![\Gamma \rhd M_2 : \sigma]\!](t).$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

3. $[\![\Gamma \rhd M_1 \approx M_2 : \mathsf{State}]\!] : [\![\Gamma]\!] \to \mathbf{2}$ is defined as, for any $t \in [\![\Gamma]\!]$,

$$[\![\Gamma \rhd M_1 \approx M_2 : \mathsf{State}]\!](t) = \mathsf{true} \ \text{ iff } \ [\![\Gamma \rhd M_1 : \mathsf{State}]\!](t) \sim_\perp [\![\Gamma \rhd M_2 : \mathsf{State}]\!](t),$$

where $\sim_\perp$ is the lifting of the bisimilarity $\sim$ on $\boldsymbol{T}$-coalgebra (X, α).

4. If the formula has the forms of $\neg M$ or $M_1 \to M_2$, its truth value can be calculated in the standard way, i.e. for any $t \in [\![\Gamma]\!]$,

$$[\![\Gamma \rhd \neg\varphi]\!](t) = \mathsf{true} \ \text{ iff } \ [\![\Gamma \rhd \varphi]\!](t) = \mathsf{false}$$
$$[\![\Gamma \rhd \varphi \to \psi]\!](t) = \mathsf{false} \ \text{ iff } \ [\![\Gamma \rhd \varphi]\!](t) = \mathsf{true} \text{ and } [\![\Gamma \rhd \psi]\!](t) = \mathsf{false}.$$

We say a formula $\Gamma \rhd \varphi$ is *valid* in a semantic model $\mathcal{M}$, or say the model $\mathcal{M}$ satisfies the formula $\Gamma \rhd \varphi$, written as $\mathcal{M} \vDash \Gamma \rhd \varphi$, if $[\![\Gamma \rhd \varphi]\!](t) = \mathsf{true}$ for all $t \in [\![\Gamma]\!]$.

For the semantics of the substitutions of formulas, we can prove the following important result.

Theorem 3.3. Let Γ, Γ' be two contexts and $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, $N_1, \ldots, N_n$ be raw terms and $\Gamma \rhd N_i : \sigma_i$ for $i = 1, \ldots, n$, and φ be a raw formula and $\Gamma' \rhd \varphi$. Then, for any semantic model $\mathcal{M}$,

$$[\![\Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]]\!] = [\![\Gamma' \rhd \varphi]\!] \circ \langle [\![\Gamma \rhd N_1 : \sigma_1]\!], \ldots, [\![\Gamma \rhd N_n : \sigma_n]\!] \rangle.$$

Furthermore, $\mathcal{M} \vDash \Gamma' \rhd \varphi$ implies $\mathcal{M} \vDash \Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]$.

Proof. By induction on $\Gamma' \rhd \varphi$ and using Theorem 3.2, it is not hard to prove that

$$[\![\Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]]\!] = [\![\Gamma' \rhd \varphi]\!] \circ \langle [\![\Gamma \rhd N_1 : \sigma_1]\!], \ldots, [\![\Gamma \rhd N_n : \sigma_n]\!] \rangle$$

Then, if $\mathcal{M} \vDash \Gamma' \rhd \varphi$, let $f \stackrel{\text{def}}{=} \langle [\![\Gamma \rhd N_1 : \sigma_1]\!], \ldots, [\![\Gamma \rhd N_n : \sigma_n]\!] \rangle : [\![\Gamma]\!] \to [\![\Gamma']\!]$, we have

$$[\![\Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]]\!](t) \ = \ [\![\Gamma' \rhd \varphi]\!](f(t)) \ = \ \mathsf{true}$$

for any $t \in [\![\Gamma]\!]$. This means $\mathcal{M} \vDash \Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]$. □

We give some examples to illustrate the usage of the language. These examples will be continued in the next section to show how to validate a state bisimulation formula logically by using bisimulation proof methods in the language. Such a process of validation is thought of as reasoning about bisimulations in the paper.

Example 3.4. In the first example, we consider the signature functor of streams, i.e. $\boldsymbol{T}X = A \times X$, which has an observation path $\pi_1 : \boldsymbol{T} \rightsquigarrow \boldsymbol{K}_A$ and a state path $\pi_2 : \boldsymbol{T} \rightsquigarrow \mathbf{Id}$. We give a signature $\mathsf{Sg} = (\mathsf{Sort}, \mathsf{Op})$ over $\boldsymbol{T}$, defined as $\mathsf{Sort} = \{\mathsf{State}, \sigma_A\}$ and

$$\mathsf{Op} = \{odd : \mathsf{State} \to \mathsf{State},\ even : \mathsf{State} \to \mathsf{State},\ zip : \mathsf{State}, \mathsf{State} \to \mathsf{State}\}.$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

For any state term M, we use more meaningful notation $head(M)$ for the term $\pi_1^\sharp(M)$, and $tail(M)$ for $\pi_2^\sharp(M)$. The following are some formulas in the language.

$$\begin{aligned}
x : \mathsf{State} &\rhd head(odd(x)) \approx head(x) : \sigma_A \\
x : \mathsf{State} &\rhd tail(odd(x)) \approx odd(tail(tail((x)))) : \mathsf{State} \\
x : \mathsf{State} &\rhd head(even(x)) \approx head(tail(x)) : \sigma_A \\
x : \mathsf{State} &\rhd tail(even(x)) \approx even(tail(tail(x))) : \mathsf{State} \\
x : \mathsf{State}, y : \mathsf{State} &\rhd head(zip(x, y)) \approx head(x) : \sigma_A \\
x : \mathsf{State}, y : \mathsf{State} &\rhd tail(zip(x, y)) \approx zip(y, tail(x)) : \mathsf{State}
\end{aligned}$$

The operator *odd* turns a stream x into a new stream $odd(x)$ consisting only of the elements of x at odd positions (note that, the first position is 1 not 0); the operator *even* extracts elements at even positions; and the operator *zip* merges two streams x, y by taking elements from x and y in turn. These operators of streams (or infinite lists) are widely discussed in the literature (such as (Capretta 2011; Jacobs 2005; Rutten 2000) etc.). The above formulas specify the basic properties of these operators by giving how the basic observations (modelled as paths) applied to the results of the operators.

It is easy to give a semantic model for the above signature Sg. For example, the $\boldsymbol{T}$-coalgebra of the model is $(A^{\mathbb{N}}, \langle \mathsf{head}, \mathsf{tail} \rangle)$, i.e. the final coalgebra of $\boldsymbol{T}$; types and operators in Sg can be interpreted in a straightforward way.

Example 3.5. In the second example, we also consider the signature functor of streams, but the data set is the set of real numbers, i.e. $\boldsymbol{T}X = \mathbb{R} \times X$. We give a signature $\mathsf{Sg} = (\mathsf{Sort}, \mathsf{Op})$ over $\boldsymbol{T}$, defined as $\mathsf{Sort} = \{\mathsf{State}, \sigma_{\mathbb{R}}\}$ and

$$\begin{aligned}
\mathsf{Op} \;=\; \{\; &+ : \sigma_{\mathbb{R}}, \sigma_{\mathbb{R}} \to \sigma_{\mathbb{R}},\;\; \cdot : \sigma_{\mathbb{R}},\; \sigma_{\mathbb{R}} \to \sigma_{\mathbb{R}},\; 0 : \sigma_{\mathbb{R}},\; [\,] : \sigma_{\mathbb{R}} \to \mathsf{State}, \\
&zero : \mathsf{State}, \oplus : \mathsf{State}, \mathsf{State} \to \mathsf{State},\; \times : \mathsf{State}, \mathsf{State} \to \mathsf{State}\;\}.
\end{aligned}$$

The intending meanings of these operators can refer to the stream calculus presented by Rutten (2001). For any state term M, like the above example, we use $head(M)$ for the term $\pi_1^\sharp(M)$, and $tail(M)$ for $\pi_2^\sharp(M)$. The following are some formulas in the language (for convenience, we use the function symbols as infix operators).

$$\begin{aligned}
&x : \mathsf{State}, y : \mathsf{State} \rhd head(x \oplus y) \approx head(x) + head(y) \;:\; \sigma_{\mathbb{R}} \\
&x : \mathsf{State}, y : \mathsf{State} \rhd tail(x \oplus y) \approx tail(x) \oplus tail(y) \;:\; \mathsf{State} \\
&x : \mathsf{State}, y : \mathsf{State} \rhd head(x \times y) \approx head(x) \cdot head(y) \;:\; \sigma_{\mathbb{R}} \\
&x : \mathsf{State}, y : \mathsf{State} \rhd tail(x \times y) \approx ([head(x)] \times tail(y)) \oplus (tail(x) \times y) \;:\; \mathsf{State}.
\end{aligned}$$

In fact, the above formulas give the specification of stream operators *zero*(the constant stream with all elements are 0), [ ](transferring a real number to a constant stream), $\oplus$(sum) and $\times$(convolution product) defined in the stream calculus in Rutten (2001).

Example 3.6. In the third example, we consider the functor $\boldsymbol{T}X = A \times (1 + X \times X)$, which is the signature functor of infinite binary trees given in Jacobs (2002). Each node of such a tree has a label from a fixed set A, and has either no successor trees (also called children),

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

or has two children. The functor has two observation paths p_1, p_2 and two state paths p_3, p_4:

$$p_1 = \pi_1 : \boldsymbol{T} \rightsquigarrow \mathbf{K}_A \qquad p_2 = \pi_2 \cdot \kappa_1 : \boldsymbol{T} \rightsquigarrow \mathbf{K}_1$$
$$p_3 = \pi_2 \cdot \kappa_2 \cdot \pi_1 : \boldsymbol{T} \rightsquigarrow \mathbf{Id} \qquad p_4 = \pi_2 \cdot \kappa_2 \cdot \pi_2 : \boldsymbol{T} \rightsquigarrow \mathbf{Id}.$$

We give a signature Sg over $\boldsymbol{T}$, which has types $\mathsf{State}, \sigma_A, \sigma_1$, and a function symbol *mirror* : State→State. For any state term M, we use $label(M)$ to stand for $p_1^\sharp(M)$, $nochild(M)$ for $p_2^\sharp(M)$, $lchild(M)$ for $p_3^\sharp(M)$ and $rchild(M)$ for $p_4^\sharp(M)$. The following are some formulas in the language.

$$x : \mathsf{State} \rhd label(mirror(x)) \approx label(x) : \sigma_A$$
$$x : \mathsf{State} \rhd nochild(mirror(x)) \Downarrow nochild(x) : \sigma_1$$
$$x : \mathsf{State} \rhd rchild(mirror(x)) \Downarrow lchild(x) : \mathsf{State}$$
$$x : \mathsf{State} \rhd lchild(mirror(x)) \Downarrow rchild(x) : \mathsf{State}$$
$$x : \mathsf{State} \rhd rchild(mirror(x)) \approx mirror(lchild(x)) : \mathsf{State}$$
$$x : \mathsf{State} \rhd lchild(mirror(x)) \approx mirror(rchild(x)) : \mathsf{State}$$

Here, the term definition formulas are critical, because a tree node x may has no child, and thus $lchild(x)$ is not a valid transition on such nodes. Then, we need the term definition formulas, such as, $x : \mathsf{State} \rhd nochild(x) \Downarrow\!\!\leftrightarrow \neg(lchild(x) \Downarrow)$ and $x : \mathsf{State} \rhd lchild(x) \Downarrow rchild(x)$, to specify a tree node has either no children, or has two children.

In fact, the above formulas specify the properties of *mirror*, which swap the children of trees (if any). Jacobs gives some semantic models for the above signature Sg (Jacobs 2002). The specification presented by Jacobs includes one more function *size*, which gives the size of a binary tree. This function can also be specified in our specification language. However, we omit its specification here, because it is unimportant for our latter discussion of bisimulation proof methods in the next section.

4. Bisimulation proof methods in the language

In this section, we present some bisimulation proof methods for polynomial coalgebras. We discuss certain properties of semantic consequences of formulas, since our bisimulation proof methods are given in the form of semantic consequence so far. Then, we discuss what are bisimulation proofs in our language, giving a basic result on how to define a relation on state terms for bisimulation proofs. In order to reason about bisimulations in a syntactic way, we further apply the idea of Sangiorgi's bisimulation up-to techniques to our language. We define a notion of progression based on the relationship between bisimulations and paths, as well as a notion of sound function based on this progression. Then, we discuss how to use these notions to form reasoning rules on bisimulation proofs from a semantic prospective for the language presented in Section 3. Finally, we define a notion of faithful context and give a sound function based on faithful contexts to form a bisimulation up-to context proof method for the langauge.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

4.1. *Semantic consequence*

Let Σ be a set of formulas, we say a formula $\Gamma \rhd \varphi$ is a *semantic consequence* of Σ, written $\Sigma \vDash \Gamma \rhd \varphi$, if, for any model $\mathcal{M}$, $\mathcal{M} \vDash \Sigma$ implies $\mathcal{M} \vDash \Gamma \rhd \varphi$, where $\mathcal{M} \vDash \Sigma$ means $\mathcal{M} \vDash \Gamma' \rhd \psi$ for every $\Gamma' \rhd \psi \in \Sigma$. Actually, a formula $x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_n : \sigma_n \rhd \varphi$ can be regarded as a formula $\forall x_1 : \sigma_1 \forall x_2 : \sigma_2 \cdots \forall x_n : \sigma_n \ \varphi$ in first-order logic. Therefore, certain knowledge of first-order logic is needed to prove the properties of the semantic consequence defined here. However, we just discuss its important properties which are specific to the language in this paper.

Let $\Sigma = \{\Gamma_1 \rhd \varphi_1, \ldots, \Gamma_n \rhd \varphi_n\}$. Obviously, $\Sigma \vDash \Gamma \rhd \varphi$ can also be written as a rule of the following form: for any set of formulas Σ',

$$\frac{\Sigma' \vDash \Gamma_1 \rhd \varphi_1, \quad \ldots, \quad \Sigma' \vDash \Gamma_n \rhd \varphi_n}{\Sigma' \vDash \Gamma \rhd \varphi},$$

which means $\Sigma' \vDash \Gamma_1 \rhd \varphi_1$, …, and $\Sigma' \vDash \Gamma_n \rhd \varphi_n$ imply $\Sigma' \vDash \Gamma \rhd \varphi$. For convenience, we use rules to give how to derive the semantic consequences, and thus give the properties of the semantic consequences in the sequel.

Lemma 4.1. It is easy to show the following rules by the semantics of formulas.

$$\frac{\Gamma \rhd M : \sigma}{\Sigma \vDash \Gamma \rhd M \approx M : \sigma} \qquad \frac{\Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \sigma}{\Sigma \vDash \Gamma \rhd M_2 \approx M_1 : \sigma}$$

$$\frac{\Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \sigma \qquad \Sigma \vDash \Gamma \rhd M_2 \approx M_3 : \sigma}{\Sigma \vDash \Gamma \rhd M_1 \approx M_3 : \sigma}.$$

Lemma 4.2. The substitution rule and the weakening rule for the semantic consequences:

$$\frac{\Sigma \vDash \ x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd \varphi \qquad \Gamma \rhd N_1 : \sigma_1, \ \ldots, \ \Gamma \rhd N_n : \sigma_n}{\Sigma \vDash \Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]}$$

$$\frac{\Sigma \vDash \Gamma \rhd \varphi \qquad \Gamma \subseteq \Gamma'}{\Sigma \vDash \Gamma' \rhd \varphi}.$$

Proof. Because $\mathcal{M} \vDash \Gamma' \rhd \varphi$ implies $\mathcal{M} \vDash \Gamma \rhd \varphi[\overrightarrow{N}/\overrightarrow{x}]$ for any semantic model $\mathcal{M}$ (see Theorem 3.3). □

The following corollary explicitly gives the cases of the above lemma when φ is a basic formula.

Corollary 4.3. Let Σ be a formula set, $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, and Γ be two contexts.

1. *Substitutions preserve term definition formulas*:

$$\frac{\Sigma \vDash \Gamma' \rhd M_1 \Downarrow \qquad \Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n}{\Sigma \vDash \Gamma \rhd M_1[N_1/x_1, \ldots, N_n/x_n] \Downarrow}.$$

2. *Substitutions preserve observation equations*:

$$\frac{\Sigma \vDash \Gamma' \rhd M_1 \approx M_2 : \sigma \qquad \Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n}{\Sigma \vDash \Gamma \rhd M_1[N_1/x_1, \ldots, N_n/x_n] \approx M_2[N_1/x_1, \ldots, N_n/x_n] : \sigma},$$

where $\sigma \neq \mathsf{State}$.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

3. *Substitutions preserve state bisimulation formulas*:

$$\frac{\Sigma \vDash \Gamma' \rhd M_1 \approx M_2 : \mathsf{State} \qquad \Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n}{\Sigma \vDash \Gamma \rhd M_1[N_1/x_1, \ldots, N_n/x_n] \approx M_2[N_1/x_1, \ldots, N_n/x_n] : \mathsf{State}}.$$

It is not hard to understand the above properties of the semantic consequences, since all variables in a formula are quantified by universal quantifiers.

4.2. *Bisimulation proofs in the language*

Here, we use an equational approach to construct our coalgebraic specification language for polynomial coalgebras. A benefit of such approach is that we can use first-order equational logic to deduce observation equations since they are the same as in algebraic languages. Thus, we focus on state bisimulation formulas because what rules can be used to deduce the state bisimulation formulas is an important issue in coalgebraic specification languages.

Formally, we want to prove that $\Gamma \rhd M_1 \approx M_2 : \mathsf{State}$ is a semantic consequence of a formula set Σ. We write

$$\sim_\Sigma \;=\; \{(\Gamma \rhd M_1 : \mathsf{State}, \Gamma \rhd M_2 : \mathsf{State}) \mid \Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \mathsf{State}\},$$

i.e. $\sim_\Sigma$ is the set of all bisimilar state term pairs which are semantic consequence of Σ. Then, if we want to prove $\Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \mathsf{State}$, it is sufficient to find a binary relation R on the set of state terms such that $R \subseteq \sim_\Sigma$ and $(\Gamma \rhd M_1, \Gamma \rhd M_2) \in R$. By the semantics of formulas, it is easy to show the following lemma.

Lemma 4.4. Let Σ be a formula set, R be a binary relation on the set of state terms. $R \subseteq \sim_\Sigma$ if and only if

1. for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$, $\Sigma \vDash \Gamma \rhd M \Downarrow N$;
2. $[\![R]\!] \subseteq \sim_X$ for each semantic model $\mathcal{M} = ((X, \alpha), I)$ with $\mathcal{M} \vDash \Sigma$, where

$$[\![R]\!] = \{([\![\Gamma \rhd M]\!](t), [\![\Gamma \rhd N]\!](t)) \mid (\Gamma \rhd M, \Gamma \rhd N) \in R, t \in \mathbf{wdom}([\![\Gamma \rhd M]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd N]\!])\}$$

and $\sim_X$ is the largest bisimulation on (X, α).

Proof. Let $(\Gamma \rhd M, \Gamma \rhd N) \in R$. For each model $\mathcal{M}$ with $\mathcal{M} \vDash \Sigma$, and all $t \in [\![\Gamma]\!]$, (1) implies $t \in \mathbf{wdom}([\![\Gamma \rhd M]\!])$ iff $t \in \mathbf{wdom}([\![\Gamma \rhd N]\!])$, and (2) implies $[\![\Gamma \rhd M]\!](t) \sim [\![\Gamma \rhd N]\!](t)$ when $t \in \mathbf{wdom}([\![\Gamma \rhd M]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd N]\!])$. Thus, $\mathcal{M} \vDash \Gamma \rhd M \approx N$. □

The above lemma cannot be used to deduce state bisimulation formulas in a syntactic way because it use the relation $[\![R]\!]$ on the coalgebra of the semantic model. However, using Proposition 2.6, we can get a syntactic way to deduce state bisimulation formulas.

Lemma 4.5. Let Σ be a formula set, R be a binary relation on the set of state terms. $R \subseteq \sim_\Sigma$ if, for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$,

1. $\Sigma \vDash \Gamma \rhd M \Downarrow N$;
2. for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M) \approx p^\sharp(N)$;
3. for all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$, $(p^\sharp(M), p^\sharp(N)) \in R$.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Proof. Let $\mathcal{M} = ((X, \alpha), I)$ be a semantic model with $\mathcal{M} \models \Sigma$. $[\![R]\!]$ is defined as in Lemma 4.4. For all $t \in [\![\Gamma]\!]$, (2) implies $[\![\Gamma \rhd p^\sharp(M)]\!](t) = [\![\Gamma \rhd p^\sharp(N)]\!](t)$ for all observation path p, (3) and (1) imply $t \in \mathbf{wdom}([\![\Gamma \rhd p^\sharp(M)]\!])$ iff $t \in \mathbf{wdom}([\![\Gamma \rhd p^\sharp(N)]\!])$, and (3) implies $([\![\Gamma \rhd p^\sharp(M)]\!], [\![\Gamma \rhd p^\sharp(N)]\!]) \in [\![R]\!]$ when $t \in \mathbf{wdom}([\![\Gamma \rhd M]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd N]\!])$. Thus, by Proposition 2.6, $[\![R]\!]$ is a bisimulation on (X, α). □

Example 4.6. Consider a simple specification for a class FLAG, which is given by Jacobs in (Jacobs 1998). There are four methods $isUp$: State$\to\sigma_{Bool}$, $setUp$: Steat$\to$ State, $setDown$: State$\to$State, $revert$: State$\to$State. We can set $isUp, setUP, setDown$ as the basic observation/transition (and then modelled as paths of the signature functor) on the objects (i.e. state) of the class, and *revert* as an algebraic operator. Thus, the signature functor of this specification is $\boldsymbol{T}X = \mathsf{Bool} \times X \times X$, which has one observation path $p_1 = \pi_1$ and two state paths $p_2 = \pi_1 \cdot \pi_2, p_3 = \pi_2 \cdot \pi_2$. We then use $isUp(M)$ to denote $p_1^\sharp(M)$, $setUp(M)$ to denote $p_2^\sharp(M)$ and $setDown(M)$ to denote $p_3^\sharp(M)$. The following formulas specify the basic properties of these methods.

$$
\begin{aligned}
&x : \mathsf{State} \rhd isUp(x) \Downarrow : \sigma_{Bool} \leftrightarrow x \Downarrow : \mathsf{State}\\
&x : \mathsf{State} \rhd isUp(setUp(x)) \approx \mathsf{true}\\
&x : \mathsf{State} \rhd isUp(setDown(x)) \approx \mathsf{false}\\
&x : \mathsf{State} \rhd isUp(revert(x)) \approx not(isUp(x))\\
&b : \sigma_{Bool} \rhd not(not(b)) \approx b,
\end{aligned}
$$

where $\mathsf{true} : \sigma_{Bool}, \mathsf{false} : \sigma_{Bool}, \mathsf{note} : \sigma_{Bool} \to \sigma_{Bool}$ are functional symbols on σ_{Bool}. Then, it is easy to that show the relation R defined as follows satisfies the conditions in Lemma 4.5, and thus $R \subseteq \sim_\Sigma$.

$$
\begin{aligned}
R \;=\; \{&(x : \mathsf{State}, y : \mathsf{State} \rhd x : \mathsf{State}, x : \mathsf{State}, y : \mathsf{State} \rhd y : \mathsf{State}) \mid\\
&\Sigma \models x : \mathsf{State}, y : \mathsf{State} \rhd isUp(x) \approx isUp(y)\},
\end{aligned}
$$

where the set Σ consists of the above five formulas. And then, we have $\Sigma \models revert(revert(x)) \approx x$ since it is easy to show that $\Sigma \models isUp(revert(revert(x))) \approx isUp(x)$.

However, it is not easy to find such a relation on state terms to deduce a state bisimulation formula. Moreover, Lemma 4.5 cannot give us a reasoning rule about bisimulation formulas because the condition $(p^\sharp(M), p^\sharp(N)) \in R$ cannot be expressed by the notation of semantic consequences.

4.3. *Bisimulation-up-to bisimilarity*

This section applies the idea of Sangiorgi's bisimulation up-to techniques to our language for seeking more methods to prove state bisimulation formulas in our language. First, we give the notions of progressions, sound functions and respectful functions in terms of paths.

Let (X, α) and (Y, β) be two $\boldsymbol{T}$-coalgebras. A relation $R \subseteq X \times Y$ is called a *quasi-bisimulation*, if, for all $x \in X$ and $y \in Y$, $(x, y) \in R$ implies

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

i. for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$, $x \in \mathbf{wdom}(p^{\sharp}_{\alpha})$ iff $y \in \mathbf{wdom}(p^{\sharp}_{\beta})$; and
ii. for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$, $p^{\sharp}_{\alpha}(x) = p^{\sharp}_{\beta}(y)$.

Let R be a quasi-bisimulation between two $\boldsymbol{T}$-coalgebras (X, α) and (Y, β), and $S \subseteq X \times Y$ is an ordinary relation. We say that R *progresses to* S, written $R \overset{\alpha,\beta}{\Longrightarrow} S$ ($R \Longrightarrow$ S for short), if $x \in \mathbf{wdom}(p^{\sharp}_{\alpha})$ and $y \in \mathbf{wdom}(p^{\sharp}_{\beta})$ imply $(p^{\sharp}_{\alpha}(x), p^{\sharp}_{\beta}(y)) \in S$ for all $(x, y) \in R$ and all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$.

By Proposition 2.6, a quasi-bisimulation R is a bisimulation iff $R \Longrightarrow R$. Intuitively, for a quasi-bisimulation R, $(s, t) \in R$ implies that s and t have the same observational outputs and the same possible state transitions. In addition to the notion of quasi-bisimulation, the notion of progression describes which relation a state pair in a quasi-bisimulation reaches after one possible state transition. These two notions motivate us to use two steps to verify a bisimulation relation: the first step verifies that it is a quasi-bisimulation, and the second step verifies that it progresses to itself. In most cases, the first step is much easier than the second step. However, the first step is critical for polynomial coalgebras because such coalgebras usually have more complex observational outputs than labelled transition systems.

Let (X, α) and (Y, β) be two $\boldsymbol{T}$-coalgebras. A function $\phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ is called a *sound function*, if $R \Longrightarrow \phi(R)$ implies $R \subseteq \sim$ for all quasi-bisimulations $R \subseteq X \times Y$, where $\sim$ is the largest bisimulation between (X, α) and (Y, β). A function $\phi : \mathcal{P}(X \times Y) \to \mathcal{P}(X \times Y)$ is called a *respectful function*, if

i. ϕ preserves quasi-bisimulations, that is, if R is a quasi-bisimulation, so is $\phi(R)$; and
ii. ϕ preserves progressions, that is, for any quasi-bisimulation R and any ordinary relation S, if $R \subseteq S$ and $R \Longrightarrow S$, then $\phi(R) \subseteq \phi(S)$ and $\phi(R) \Longrightarrow \phi(S)$.

As in Sangiorgi (1998), it is not hard to prove that all respectful functions are sound.

Theorem 4.7. Let (X, α) and (Y, β) be two $\boldsymbol{T}$-coalgebras, ϕ be a respectful function, and $R \subseteq X \times Y$ be a quasi-bisimulation. Then, $R \Longrightarrow \phi(R)$ implies $R \subseteq \sim$.

Proof. As the proof of Theorem 2.11 in Sangiorgi (1998), we inductively define a sequence of relations $(R_i)_{i \in \mathbb{N}}$:

$$\begin{aligned} R_0 &= R \\ R_1 &= R_0 \cup \phi(R_0) \\ &\vdots \qquad\qquad \vdots \\ R_{n+1} &= R_n \cup \phi(R_n). \end{aligned}$$

Using the same proof in Sangiorgi (1998), it is easy to show two facts: $R_n \subseteq R_{n+1}$ and $R_n \Longrightarrow R_{n+1}$ for all $n \in \mathbb{N}$. Besides the proof in Sangiorgi (1998), to show $\bigcup_{i \in \mathbb{N}} R_i$ is a bisimulation relation, we have to show one more fact that such relations are quasi-bisimulations. We can use mathematical induction to show that R_n is a quasi-bisimulation relation for all $n \in \mathbb{N}$, since R_n and $\phi(R_n)$ are quasi-bisimulation relations, and the union of two quasi-bisimulation relations is a quasi-bisimulation relation too. □

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Actually, the condition (2) in the above definition of respectful function is the same as the property of respectful functions in Sangiorgi (1998). Here, we require a respectful function satisfies one more condition, i.e. preserving quasi-bisimulations, which is critical for showing that R_n is a quasi-bisimulation in the above proof.

By Lemma 4.4, we immediately get the following corollary.

Corollary 4.8. Let Σ be a formula set, R be a binary relation on the set of state terms. $R \subseteq \sim_\Sigma$ if

1. for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$, $\Sigma \vDash \Gamma \rhd M \Downarrow N$;
2. for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$, and $p \in \mathtt{OSPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M) \Downarrow p^\sharp(N)$;
3. for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$, and $p \in \mathtt{OPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M) \approx p^\sharp(N)$;
4. there exists a sound/respectful function ϕ such that $[\![R]\!] \Longrightarrow \phi([\![R]\!])$ for each semantic model $\mathcal{M}$ with $\mathcal{M} \vDash \Sigma$, where $[\![R]\!]$ is defined as in Lemma 4.4.

Proof. Obviously, (2) and (3) imply $[\![R]\!]$ is a quasi-bisimulation, and then (4) implies $[\![R]\!] \subseteq \sim$. □

As in Sangiorgi (1998), the identity function and the constant-to-$\sim$, mapping every relation to $\sim$, are two primitive respectful functions, and respectful functions are closure under constructors including composition, union and chaining. Using these primitive respectful functions and constructors, the function $\phi_B(R) = \sim \circ R \circ \sim$ is a respectful function. As a corollary, we obtain the following bisimulation proof method, which is called bisimulation-up-to bisimilarity in process algebras.

Corollary 4.9. Let (X, α) be a $\boldsymbol{T}$-coalgebra. A relation R on X is contained in the largest bisimulation $\sim$ on X, if it satisfies that, for all $x, y \in X$, $(x, y) \in R$ implies

1. for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$, $x \in \mathbf{wdom}(p^\sharp_\alpha)$ iff $y \in \mathbf{wdom}(p^\sharp_\alpha)$;
2. for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$, $p^\sharp_\alpha(x) = p^\sharp_\alpha(y)$; and
3. for all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$, if $x, y \in \mathbf{wdom}(p^\sharp_\alpha)$, then there exist $x', y' \in X$ such that $x' \sim \widehat{p}_X(\alpha(x))$, $y' \sim \widehat{p}_X(\alpha(y))$ and $(x', y') \in R$.

Proof. The condition (1) and (2) imply that R is a quasi-bisimulation relation, and the condition (3) means $R \Longrightarrow \sim \circ R \circ \sim$. □

As Lemma 4.5, we can transform the above corollary to a syntactic method to deduce state bisimulation formulas in the language.

Corollary 4.10. Let Σ be a formula set, R be a binary relation on the set of state terms. $R \subseteq \sim_\Sigma$ if, for all $(\Gamma \rhd M, \Gamma \rhd N) \in R$,

1. $\Sigma \vDash \Gamma \rhd M \Downarrow N$;
2. for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M) \approx p^\sharp(N)$;
3. for all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$, there exist $(\Gamma \rhd M', \Gamma \rhd N') \in R$ such that

$$\Sigma \vDash p^\sharp(M) \approx M' \qquad \text{and} \qquad \Sigma \vDash p^\sharp(N) \approx N'.$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Proof. Note that (1) (2), and (3) imply that $\Sigma \vDash \Gamma \rhd p^\sharp(M) \Downarrow p^\sharp(N)$ for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$, and then (3) implies $[\![R]\!] \Longrightarrow \sim \circ [\![R]\!] \circ \sim$, where $[\![R]\!]$ is defined as in Lemma 4.4. □

As mentioned before, the above result cannot be used as a syntactic reasoning rule to deduce state bisimulation formulas since the condition $(\Gamma \rhd M', \Gamma \rhd N') \in R$ cannot be expressed by the notation of semantic consequences. However, when M', N' have certain special forms, we can use the relation R implicitly, and then all conditions can be expressed by the notation of semantic consequences.

Theorem 4.11. Let Σ be a set of formulas, $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ be a context, and $\Gamma \rhd M_1 : \mathsf{State}, \Gamma \rhd M_2 : \mathsf{State}$ be two terms. Then $\Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \mathsf{State}$ if

1. $\Sigma \vDash \Gamma \rhd M_1 \Downarrow M_2 : \mathsf{State}$;
2. for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \approx p^\sharp(M_2)$; and
3. for all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$, there exist terms $\Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n$ such that

$$\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \approx M_1[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}$$
$$\Sigma \vDash \Gamma \rhd p^\sharp(M_2) \approx M_2[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}.$$

Proof. We define a relation R on the set of state terms as follows: $(\Gamma \rhd M_1, \Gamma \rhd M_2) \in R$ iff they satisfy the above conditions (1), (2) and (3). Note that (1) implies $\Sigma \vDash \Gamma \rhd M_1[\overrightarrow{N}/\overrightarrow{x}] \Downarrow M_2[\overrightarrow{N}/\overrightarrow{x}]$ since substitutions preserve term definition formulas. Then, (2) and (3) imply that $\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \Downarrow p^\sharp(M_2)$ for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$.

Let $\mathcal{M}$ be a semantic model with $\mathcal{M} \vDash \Sigma$. $[\![R]\!]$ is defined as in Lemma 4.4. For convenience, let

$$f \stackrel{\text{def}}{=} \langle [\![\Gamma \rhd N_1 : \sigma_1]\!], \ldots, [\![\Gamma \rhd N_n : \sigma_n]\!] \rangle : [\![\Gamma]\!] \to [\![\Gamma]\!].$$

For all $t \in [\![\Gamma]\!]$, if $t \in \mathbf{wdom}([\![p^\sharp(\Gamma \rhd M_1)]\!]) \cap \mathbf{wdom}([\![p^\sharp(\Gamma \rhd M_2)]\!]$, then (3) implies $f(t) \in \mathbf{wdom}([\![\Gamma \rhd M_1]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M_1]\!])$. And thus

$$([\![\Gamma \rhd M_1[\overrightarrow{N}/\overrightarrow{x}]]\!](t), [\![\Gamma \rhd M_2[\overrightarrow{N}/\overrightarrow{x}]]\!](t)) = ([\![\Gamma \rhd M_1]\!](f(t)), [\![\Gamma \rhd M_2]\!](f(t))) \in [\![R]\!].$$

And then (3) implies $[\![R]\!] \Longrightarrow \sim \circ [\![R]\!] \circ \sim$. Therefore $R \subseteq \sim_\Sigma$ by Corollary 4.8. □

The above theorem gives us a method to deduce state bisimulation formulas in a syntactic way, since all conditions to prove $\Sigma \vDash \Gamma \rhd M_1 \approx M_2$ are expressed by the notation of semantic consequences, and then we can prove such state bisimulation formulas by only judging the syntax forms of the involving formulas.

One may expect to define a 'sound function' for the relations on the set of state terms to generalize the above theorem. For example, the above theorem likely uses a function ϕ defined as $(\Gamma \rhd P, \Gamma \rhd Q) \in \phi(R)$ if and only if there exist $(\Gamma \rhd M, \Gamma \rhd N) \in R$, and $\Gamma \rhd L_i : \sigma_i (i = 1, \ldots, n)$ such that $P = M[\overrightarrow{L}/\overrightarrow{x}]$ and $Q = N[\overrightarrow{L}/\overrightarrow{x}]$. However, it is

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

not hard to define $R \Longrightarrow \phi(R)$ for the relations on the set of state terms, whereas it is unknown how to define a function for the relations on the set of state terms is respectful so far. Therefore, here we depend on respectful functions defined on the semantic models of terms to deduce the state bisimulations formulas.

The following example illustrates how to use Theorem 4.11 to deduce state bisimulation formulas.

Example 4.12. Considering the signature Sg given in Example 3.4. Let Σ be the set of formulas:

$$\begin{aligned}
x : \mathsf{State} &\rhd head(x) \Downarrow : A \leftrightarrow x \Downarrow : \mathsf{State}\\
x : \mathsf{State} &\rhd tail(x) \Downarrow x : \mathsf{State}\\
x : \mathsf{State} &\rhd odd(x) \Downarrow x : \mathsf{State}\\
x : \mathsf{State} &\rhd even(x) \Downarrow x : \mathsf{State}\\
x : \mathsf{State} &\rhd head(odd(x)) \approx head(x)\\
x : \mathsf{State} &\rhd tail(odd(x)) \approx odd(tail(tail(x)))\\
x : \mathsf{State} &\rhd head(even(x)) \approx head(tail(x))\\
x : \mathsf{State} &\rhd tail(even(x)) \approx even(tail(tail(x)))\\
x : \mathsf{State}, y : \mathsf{State} &\rhd zip(x, y) \Downarrow (x, y) : \mathsf{State}\\
x : \mathsf{State}, y : \mathsf{State} &\rhd head(zip(x, y)) \approx head(x)\\
x : \mathsf{State}, y : \mathsf{State} &\rhd tail(zip(x, y)) \approx zip(y, tail(x))
\end{aligned}$$

Let $\Gamma = [x : \mathsf{State}]$. First, by the fact that substitutions preserve term definition formulas, $\Sigma \vDash \Gamma \rhd odd(tail(x)) \Downarrow even(x)$ since $\Sigma \vDash \Gamma \rhd odd(x) \Downarrow x$, $\Sigma \vDash \Gamma \rhd tail(x) \Downarrow x$, and $\Sigma \vDash \Gamma \rhd even(x) \Downarrow x$.

Second, by the fact that substitutions preserve observation equations,

$$\Sigma \vDash \Gamma \rhd head(odd(tail(x))) \approx head(even(x)),$$

since $\Sigma \vDash \Gamma \rhd head(odd(x)) \approx head(x)$ and $\Sigma \vDash \Gamma \rhd head(even(x)) \approx head(tail(x))$.

Finally, by the fact that substitutions preserve state bisimulation formulas and $\Sigma \vDash \Gamma \rhd tail(odd(x)) \approx odd(tail(tail((x))))$,

$$\Sigma \vDash \Gamma \rhd tail(odd(tail(x))) \approx odd(tail(tail((tail(x))))).$$

That is,

$$\begin{aligned}
\Sigma \vDash \Gamma &\rhd tail(odd(tail(x))) \approx odd(tail(x))[tail(tail(x))/x]\\
\Sigma \vDash \Gamma &\rhd tail(even(x)) \approx even(x)[tail(tail(x))/x].
\end{aligned}$$

Therefore, by Theorem 4.11, $\Sigma \vDash \Gamma \rhd even(x) \approx odd(tail(x))$.

However, we cannot prove that $\Sigma \vDash x : \mathsf{State} \rhd zip(odd(x), even(x)) \approx x$ so far. It is easy to show that $\Sigma \vDash x : \mathsf{State} \rhd head(zip(odd(x), even(x)) \approx head(x)$. Also, we have $\Sigma \vDash x : \mathsf{State} \rhd tail(zip(odd(x), even(x))) \approx zip(even(x), tail(odd(x))$, and then we expect

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

that $even(x)$ can be substituted by $odd(tail(x))$, and $tail(odd(x))$ can be substituted by $odd(tail(tail(x)))$ and then by $even(tail(x))$. However, it is unclear whether we have

$$\Sigma \vDash \Gamma \rhd x_1 \approx x_2 \ \wedge \ y_1 \approx y_2 \ \rightarrow \ zip(x_1, y_1) \approx zip(x_2, y_2),$$

where $\Gamma = [x_1 : \mathsf{State}, x_2 : \mathsf{State}, y_1 : \mathsf{State}, y_2 : \mathsf{State}]$.

In general, it does not directly follow from the semantics of formulas that a state term, e.g, $x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \mathsf{State}$, preserves bisimulations, i.e.

$$\mathcal{M} \vDash \Gamma \rhd (N_1 \approx L_1 \wedge \cdots \wedge N_n \approx L_n) \rightarrow M[\overrightarrow{N}/\overrightarrow{x}] \approx M[\overrightarrow{L}/\overrightarrow{x}]$$

for arbitrary model $\mathcal{M}$, and state terms $\Gamma \rhd N_i : \sigma_i (i = 1, \ldots, n)$. This motivate us to discover what conditions ensure that a state term preserve bisimulations, and to seek more powerful bisimulation proof methods.

4.4. *Bisimulation-up-to contexts*

In this subsection, we want to introduce the notion of faithful context and build a bisimulation up-to context proof technique for our language. Intuitively, a faithful context in our language is a set of state terms which preserve bisimulations as mentioned above. We observe that a set of conditions, i.e. being well-defined, preserving observation and having faithful progressions, is sufficient for a state term in our language to preserve bisimulations. These conditions can be expressed in the language, and can be verified in a syntactic way. Hence, we use them to define the notion of faithful context in the language.

To give the definition of faithful contexts, we fix a set of formulas Σ. In addition, for a state term $\Gamma \rhd M : \mathsf{State}$, we always assume that $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ and $fv(M) = \{x_1, \ldots, x_m\} \subseteq \mathbf{dom}(\Gamma)$.

We say a state term $\Gamma \rhd M : \mathsf{State}$ is *well-defined with respect to* Σ, if (i) $\Sigma \vDash \Gamma \rhd M \Downarrow (x_1, \ldots, x_m)$, and (ii) for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$,

$$\Sigma \vDash \Gamma, \Gamma' \rhd P_1 \Downarrow Q_1 \wedge \cdots \wedge P_m \Downarrow Q_m \rightarrow (p^\sharp(M) \Downarrow p^\sharp(M[\overrightarrow{y}/\overrightarrow{x}])),$$

where $\Gamma' = [y_1 : \sigma_1, \ldots, y_n : \sigma_n]$, and for each $i = 1, \ldots, m$,

$$P_i \stackrel{\text{def}}{=} \begin{cases} p^\sharp(x_i) & \text{if } \sigma_i = \mathsf{State} \\ x_i & \text{otherwise} \end{cases} \qquad Q_i \stackrel{\text{def}}{=} \begin{cases} p^\sharp(y_i) & \text{if } \sigma_i = \mathsf{State} \\ y_i & \text{otherwise.} \end{cases}$$

Example 4.13. We use an example to clarify the above definition. Consider the signature functor $\boldsymbol{T}$ in Example 3.6, we hope that $x : \mathsf{State} \rhd mirror(x) : \mathsf{State}$ will be a faithful context. It is well-defined with respect to Σ if (i) $\Sigma \vDash x : \mathsf{State} \rhd mirror(x) \Downarrow x$, and (ii) for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$,

$$\Sigma \vDash x : \mathsf{State}, y : \mathsf{State} \rhd p^\sharp(x) \Downarrow p^\sharp(y) \rightarrow p^\sharp(mirror(x)) \Downarrow p^\sharp(mirror(y)).$$

For example, we should have

$$\Sigma \vDash x : \mathsf{State}, y : \mathsf{State} \rhd lchild(x) \Downarrow lchild(y) \rightarrow lchild(mirror(x)) \Downarrow lchild(mirror(y)).$$

Intuitively, (i) require that $mirror(x)$ should be a valid operation provided x is a valid state, and (ii) require that $mirror(x)$ has left child iff $mirror(y)$ has left child when x has

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

left child iff y has left child, i.e. $mirror(x)$ and $mirror(y)$ are in the same situation of left child provided x and y are in the same situation. These requirement are reasonable if the operation *mirror* behaves well. Note that (ii) is more reasonable than the requirement that $mirror(x)$ has left child provided x has left child. Actually, $mirror(x)$ should have left child only when x has right child.

Generally, if we want to deduce $\Sigma \vDash \Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] \approx M[\overrightarrow{L}/\overrightarrow{x}]$ from $\Sigma \vDash \Gamma \rhd N_i \approx L_i : \sigma_i (i = 1, \ldots, n)$, we need to deduce at first that $\Sigma \vDash \Gamma \rhd p^\sharp(M[\overrightarrow{N}/\overrightarrow{x}]) \Downarrow p^\sharp(M[\overrightarrow{L}/\overrightarrow{x}])$ for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$. By the semantics of formulas, we only can obtain that $\Sigma \vDash \Gamma \rhd N_i \Downarrow L_i$ and $\Sigma \vDash \Gamma \rhd p^\sharp(N_i) \Downarrow p^\sharp(L_i)$ (only when $\sigma = \mathsf{State}$) from $\Sigma \vDash \Gamma \rhd N_i \approx L_i$. However, if $\Gamma \rhd M$ satisfies the two conditions in the definition of a well-defined term, then it is sufficient to obtain the required result as showed in the following lemma. This is our motivation to introduce the notion of well-defined terms.

Lemma 4.14. Let $\mathcal{M} = ((X, \alpha), I)$ be a semantic model with $\mathcal{M} \vDash \Sigma$. If $\Gamma \rhd M : \mathsf{State}$ is well-defined with respect to Σ, then

1. for any terms $\Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n$, and all $t \in [\![\Gamma]\!]$,

$$t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}]\!]) \text{ iff } \forall i = 1, \ldots, m,\ t \in \mathbf{wdom}([\![\Gamma \rhd N_i : \sigma_i]\!])$$

2. for any path $p \in \mathtt{OSPATH}(\boldsymbol{T})$, and any term $\Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n, \Gamma \rhd L_1 : \sigma_1, \ldots, \Gamma \rhd L_n : \sigma_n$, if, for all $i = 1, \ldots, m$,

$$t \in \mathbf{wdom}([\![\Gamma \rhd P_i[N_i/x_i] : \sigma_i]\!]) \text{ iff } t \in \mathbf{wdom}([\![\Gamma \rhd Q_i[L_i/y_i] : \sigma_i]\!])$$

for all $t \in [\![\Gamma]\!]$, where P_i, Q_i are defined as above, then

$$t \in \mathbf{wdom}([\![\Gamma \rhd p^\sharp(M[\overrightarrow{N}/\overrightarrow{x}]) : \mathsf{State}]\!]) \text{ iff } t \in \mathbf{wdom}([\![\Gamma \rhd p^\sharp(M[\overrightarrow{L}/\overrightarrow{x}]) : \mathsf{State}]\!])$$

for all $t \in [\![\Gamma]\!]$.

Proof. By the definition of a state term to be well-defined with respect to Σ, and the fact that substitutions preserve term definition formulas. □

We say a state term $\Gamma \rhd M : \mathsf{State}$ *preserves observation with respect to* Σ, if, for any observation path $p \in \mathtt{OPATH}(\boldsymbol{T})$,

$$\Sigma \vDash \Gamma, \Gamma' \rhd P_1 \approx Q_1 \wedge \cdots \wedge P_m \approx Q_m \ \rightarrow\ (p^\sharp(M) \approx p^\sharp(M[\overrightarrow{y}/\overrightarrow{x}])),$$

where $\Gamma' = [y_1 : \sigma_1, \ldots, y_n : \sigma_n]$, and for each $i = 1, \ldots, m$,

$$P_i \stackrel{\text{def}}{=} \begin{cases} p^\sharp(x_i) & \text{if } \sigma_i = \mathsf{State} \\ x_i & \text{otherwise} \end{cases} \qquad Q_i \stackrel{\text{def}}{=} \begin{cases} p^\sharp(y_i) & \text{if } \sigma_i = \mathsf{State} \\ y_i & \text{otherwise.} \end{cases}$$

Example 4.15. It is easy to understand the condition of preserving observation. For the state term $x : \mathsf{State} \rhd mirror(x)$, it requires that

$$\Sigma \vDash \Gamma \rhd nochild(x) \approx nochild(y) \rightarrow nochild(mirror(x)) \approx nochild(mirror(y))$$
$$\Sigma \vDash \Gamma \rhd label(x) \approx label(y) \rightarrow label(mirror(x)) \approx label(mirror(y)),$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

where $\Gamma = [x : \mathsf{State}, y : \mathsf{State}]$. The first formula means that $mirror(x)$ has no children iff $mirror(y)$ has no children provided x has no children iff y has no children, and the second formula means $mirror(x)$ and $mirror(y)$ has the same label provided x and y has the same label.

Again, in general, the next step to deduce $\Sigma \vDash \Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] \approx M[\overrightarrow{L}/\overrightarrow{x}]$ from $\Sigma \vDash \Gamma \rhd N_i \approx L_i : \sigma_i (i = 1, \ldots, n)$ is to show that $\Sigma \vDash \Gamma \rhd p^\sharp(M[\overrightarrow{N}/\overrightarrow{x}]) \approx p^\sharp(M[\overrightarrow{L}/\overrightarrow{x}])$ for all observation paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$. By the semantics of formulas, we only can obtain that $\Sigma \vDash \Gamma \rhd N_i \approx L_i$ and $\Sigma \vDash \Gamma \rhd p^\sharp(N_i) \approx p^\sharp(L_i)$ (only when $\sigma = \mathsf{State}$) from $\Sigma \vDash \Gamma \rhd N_i \approx L_i$. However, if $\Gamma \rhd M$ satisfies the condition in the definition of a state term preserving observation, then it is sufficient to obtained the required result as showed in the following lemma.

Lemma 4.16. Let $\mathcal{M} = ((X, \alpha), I)$ be a semantic model with $\mathcal{M} \vDash \Sigma$. If $\Gamma \rhd M : \mathsf{State}$ preserves observation, then, for any observation path $p \in \mathtt{OSPATH}(\boldsymbol{T})$, and any terms $\Gamma \rhd N_1 : \sigma_1, \ldots, \Gamma \rhd N_n : \sigma_n, \Gamma \rhd L_1 : \sigma_1, \ldots, \Gamma \rhd L_n : \sigma_n$, if, for all $i = 1, \ldots, m$,

$$[\![\Gamma \rhd P_i[N_i/x_i] : \sigma_i]\!](t) = [\![\Gamma \rhd Q_i[L_i/y_i] : \sigma_i]\!](t), \quad \text{for all } t \in [\![\Gamma]\!]$$

then

$$[\![\Gamma \rhd p^\sharp(M[\overrightarrow{N}/\overrightarrow{x}]) : \mathsf{State}]\!](t) \;=\; [\![\Gamma \rhd p^\sharp(M[\overrightarrow{L}/\overrightarrow{x}]) : \mathsf{State}]\!](t), \quad \text{for all } t \in [\![\Gamma]\!].$$

Proof. By the definition of a state term to preserve observations with respect to Σ, and the fact that substitutions preserve observation equations. □

To give the notion of state terms having faithful progressions, the maps between contexts are needed. A *map* from a context Γ' to Γ, written as $\rho : \Gamma' \to \Gamma$, is a function $\rho : \mathbf{dom}(\Gamma') \to \mathbf{dom}(\Gamma) \times (1 + \mathtt{SPATH}(\boldsymbol{T}))$ such that (1) for all $y \in \mathbf{dom}(\Gamma')$, $\pi_1(\rho(y)) = x$ implies $\Gamma(x) = \Gamma'(y)$, and (2) for all $y \in \mathbf{dom}(\Gamma')$, $\Gamma'(y) \neq \mathsf{State}$ implies $\pi_2(\rho(y)) = \kappa_1 *$.

Let $\mathcal{C}$ be a set of state terms. We say $\mathcal{C}$ *has faithful progressions with respect to* Σ, if, for all $\Gamma \rhd M : \mathsf{State}$ in $\mathcal{C}$, and for all state paths $p \in \mathtt{SPATH}(\boldsymbol{T})$, there exists a state term $\Gamma_p \rhd M_p : \mathsf{State}$ in $\mathcal{C}$ and a map $\rho : \Gamma_p \to \Gamma$ such that

$$\Sigma \vDash \Gamma \rhd p^\sharp(M) \approx M_p[\overrightarrow{N}/\overrightarrow{y}] : \mathsf{State},$$

where we assume that $\mathbf{dom}(\Gamma_p) = \{y_1, \ldots, y_m\}$, and for each $j = 1, \ldots, m$,

$$N_j \stackrel{\text{def}}{=} \begin{cases} x_i & \text{if } \rho(y_j) = (x_i,\ \kappa_1 *) \\ q^\sharp(x_i) & \text{if } \rho(y_j) = (x_i,\ \kappa_2 q). \end{cases}$$

Note that, the conditions in the definition of ρ imply N_j is well-defined for all $j = 1, \ldots, m$.

Example 4.17. It is not easy to understand the conditions of state terms having faithful progressions. However, it is easy to show that $x : \mathsf{State} \rhd mirror(x)$ has faithful progressions because it requires that

$$\Sigma \;\vDash lchild(mirror(x)) \approx M_l \qquad\qquad \Sigma \;\vDash rchild(mirror(x)) \approx M_r.$$

We can easily set $M_l = mirror(x)[rchild(x)/x]$ and $M_r = mirror(x)[lchild(x)/x]$. We defer the definition of the formula set Σ for the sake of clarity. Example 4.22 will give the

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

definition of Σ, which is enough to deduce the formulas given in Examples 4.13, 4.15 and 4.17.

Simply, for a state term $\Gamma \rhd M$ having faithful progressions, we hope that $\Gamma \rhd p^{\sharp}(M)$ can fall into a state term having faithful progressions again for all state path p. And thus, we require that there exists a state term $\Gamma' \rhd M_p$ such that $\Sigma \vDash \Gamma \rhd p^{\sharp}(M) \approx M_p[\overrightarrow{N}/\overrightarrow{y}]$, and a map $\rho : \Gamma' \to \Gamma$ determines how to substitute the variables in M_p.

Lemma 4.18. Let $\mathcal{M} = ((X, \alpha), I)$ be a semantic model with $\mathcal{M} \vDash \Sigma$, and $\mathcal{C}$ be a set of state terms. If $\mathcal{C}$ has faithful progressions with respect to Σ and $\Gamma \rhd M : \mathsf{State} \in \mathcal{C}$, then, for any state path $p \in \mathtt{SPATH}(\boldsymbol{T})$, and any state terms $\Gamma' \rhd N_1 : \sigma_1, \ldots, \Gamma' \rhd N_n : \sigma_n$, there exists a state term $\Gamma_p \rhd M_p : \mathsf{State} \in \mathcal{C}$ and a map $\rho : \Gamma_p \to \Gamma$ such that

$$[\![\Gamma \rhd p^{\sharp}(M[\overrightarrow{N}/\overrightarrow{x}])]\!](t) \ \sim_{\perp} \ [\![\Gamma \rhd M_p[\overrightarrow{L}/\overrightarrow{y}]]\!](t)$$

for all $t \in [\![\Gamma]\!]$, where we assume that $\mathbf{dom}(\Gamma_p) = \{y_1, \ldots, y_m\} = \mathbf{dom}(\Gamma_p)$, and for each $j = 1, \ldots, m$,

$$L_j \stackrel{\text{def}}{=} \begin{cases} N_i & \text{if } \rho(y_j) = (x_i, \kappa_1\ *) \\ q^{\sharp}(N_i) & \text{if } \rho(y_j) = (x_i, \kappa_2\ q). \end{cases}$$

Proof. By the definition of a state term set having faithful progressions and the fact that substitutions preserve state bisimulation formulas. □

We can now give the definition of faithful context. We say a set $\mathcal{C}$ of state terms is a *faithful context set with respect to* Σ, if $\mathcal{C}$ has faithful progressions with respect to Σ, and every $\Gamma \rhd M : \mathsf{State}$ in $\mathcal{C}$ is well-defined and preserves observations with respect to Σ. We say a state term $\Gamma \rhd M : \mathsf{State}$ is a *faithful term with respect to* Σ, if it belongs to a faithful context $\mathcal{C}$ with respect to Σ, i.e. $\Gamma \rhd M : \mathsf{State} \in \mathcal{C}$.

Example 4.19. Let Σ be a set of formulas, $\Gamma = [x_1 : \mathsf{State}, \ldots, x_n : \mathsf{State}]$ be a context. $\Gamma \rhd x_i : \mathsf{State}$ is a faithful term with respect to Σ for all $i = 1, \ldots, n$.

Example 4.20. Considering the signature Sg given in Example 3.4, let Σ be the set of formulas defined in Example 4.12. Then, we can verify that the state term $x : \mathsf{State},\ y : \mathsf{State} \rhd zip(x, y) : \mathsf{State}$ is a faithful term with respect to Σ as follows.

Let $\Gamma = [x : \mathsf{State}, y : \mathsf{State}]$ and $\Gamma' = [x' : \mathsf{State}, y' : \mathsf{State}]$. Obviously, $\Gamma \rhd zip(x, y) : \mathsf{State}$ is well-defined, because it is not hard to show

$$\Sigma \vDash \Gamma, \Gamma' \rhd head(x) \Downarrow head(x') \wedge head(y) \Downarrow head(y') \to head(zip(x, y)) \Downarrow head(zip(x', y'))$$
$$\Sigma \vDash \Gamma, \Gamma' \rhd tail(x) \Downarrow tail(x') \wedge tail(y) \Downarrow tail(y') \to tail(zip(x, y)) \Downarrow tail(zip(x', y')).$$

Similarly, it is not hard to show

$$\Sigma \vDash \Gamma, \Gamma' \rhd (head(x) \approx head(x') \wedge head(y) \approx head(y')) \to head(zip(x, y)) \approx head(zip(x', y')).$$

This means $\Gamma \rhd zip(x, y)$ preserves observations.

Finally, let $\mathcal{C} = \{\Gamma \rhd zip(x, y) : \mathsf{State}\}$, the fact that

$$\Sigma \vDash \Gamma \rhd tail(zip(x, y)) \approx zip(x, y)[y/x, tail(x)/y]$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

means $\mathcal{C}$ has faithful progressions, since there is a map from $\rho : \Gamma \to \Gamma$, given by $\rho(x) = (y, \kappa_1 *)$ and $\rho(y) = (x, \kappa_2\ tail)$.

Example 4.21. Considering the signature Sg given in Example 3.5, let Σ be the set of formulas:

$$\begin{aligned}
\forall f : \sigma_1, \ldots, \sigma_n \to \sigma,\ x_1 : \sigma_1, \ldots, x_n : \sigma_n &\rhd f(x_1, \ldots, x_n) \Downarrow (x_1, \ldots, x_n)\\
r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}}, r_1' : \sigma_{\mathbb{R}}, r_2' : \sigma_{\mathbb{R}} &\rhd r_1 \approx r_1' \wedge r_2 \approx r_2' \to r_1 + r_2 \approx r_1' + r_2'\\
r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}}, r_1' : \sigma_{\mathbb{R}}, r_2' : \sigma_{\mathbb{R}} &\rhd r_1 \approx r_1' \wedge r_2 \approx r_2' \to r_1 \cdot r_2 \approx r_1' \cdot r_2'\\
r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}} &\rhd r_1 + r_2 \approx r_2 + r_1\\
r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}} &\rhd r_1 \cdot r_2 \approx r_2 \cdot r_1\\
r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}}, r_3 : \sigma_{\mathbb{R}} &\rhd r_1 \cdot (r_2 + r_3) \approx r_1 \cdot r_2 + r_1 \cdot r_3\\
&\rhd head(zero) \approx 0\\
&\rhd tail(zero) \approx zero\\
r : \sigma_{\mathbb{R}} &\rhd head([r]) \approx r\\
r : \sigma_{\mathbb{R}} &\rhd tail([r]) \approx zero\\
x : \mathsf{State}, y : \mathsf{State} &\rhd head(x \oplus y) \approx head(x) + head(y)\\
x : \mathsf{State}, y : \mathsf{State} &\rhd tail(x \oplus y) \approx tail(x) \oplus tail(y)\\
x : \mathsf{State}, y : \mathsf{State} &\rhd head(x \times y) \approx head(x) \cdot head(y)\\
x : \mathsf{State}, y : \mathsf{State} &\rhd tail(x \times y) \approx ([head(x)] \times tail(y)) \oplus (tail(x) \times y).
\end{aligned}$$

Then, it is easy to verify that $x : \mathsf{State}, y : \mathsf{State} \rhd x \oplus y$ is a faithful context with respect to Σ. However, $x : \mathsf{State}, y : \mathsf{State} \rhd x \times y$ is not a faithful context with respect to Σ, since $[head(x)]$ in the specification of $tail(x \times y)$ is not an observation or state transition induced by paths.

Example 4.22. Considering the signature Sg given in Example 3.6, let Σ be the set of formulas:

$$\begin{aligned}
x : \mathsf{State} &\rhd nochild(x) \Downarrow \leftrightarrow \neg(lchild(x) \Downarrow)\\
x : \mathsf{State} &\rhd lchild(x) \Downarrow \leftrightarrow rchild(x) \Downarrow\\
x : \mathsf{State} &\rhd label(x) \Downarrow x\\
x : \mathsf{State} &\rhd mirror(x) \Downarrow x\\
x : \mathsf{State} &\rhd nochild(mirror(x)) \Downarrow nochild(x)\\
x : \mathsf{State} &\rhd label(mirror(x)) \approx label(x)\\
x : \mathsf{State}, y : \mathsf{State} &\rhd nochild(x) \approx nochild(y)\\
x : \mathsf{State} &\rhd rchild(mirror(x)) \simeq mirror(lchild(x))\\
x : \mathsf{State} &\rhd lchild(mirror(x)) \simeq mirror(rchild(x)).
\end{aligned}$$

As explained in Examples 4.13, 4.15 and 4.17, it is not hard to show that the state term $x : \mathsf{State} \rhd mirror(x)$ is a faithful context with respect to Σ.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

We present the following theorem as the main technical result of the paper, which gives how to define sound functions based on faithful contexts.

Theorem 4.23. Suppose Σ is a set of formulas, $\mathcal{C}$ is a faithful context set with respect to Σ, and $\mathcal{M} = ((X,\alpha),I)$ is a semantic model with $\mathcal{M} \models \Sigma$. We define a function $\phi_{\mathcal{C}} : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ as, for any relation $R \subseteq X \times X$,

$$\begin{aligned}\phi_{\mathcal{C}}(R) \;=\; \{\; & (t_1,t_2) \mid t_1,t_2 \in X, \text{there exist } x_1:\sigma_1,\ldots,x_n:\sigma_n \rhd M : \mathsf{State} \in \mathcal{C},\\ & \Gamma \rhd P_1:\sigma_1,\ldots,\Gamma \rhd P_n:\sigma_n, \Gamma \rhd Q_1:\sigma_1,\ldots,\Gamma \rhd Q_n:\sigma_n, \text{ and } t \in [\![\Gamma]\!]\\ & \text{such that } t_1 \sim [\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t),\ t_2 \sim [\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t),\\ & \text{where } t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!]),\\ & \quad \text{and for all } i=1,\ldots,n, \text{ if } \sigma_i \neq \mathsf{State} \text{ then } P_i = Q_i, \text{ else } P_i = Q_i \text{ or}\\ & \quad ([\![\Gamma \rhd P_i]\!](t),\ [\![\Gamma \rhd Q_i]\!](t)) \in R_\perp \;\}.\end{aligned}$$

Then $\phi_{\mathcal{C}}$ is a respectful function.

Proof. It is sufficient to show that $\phi_{\mathcal{C}}$ preserves quasi-bisimulations, and for any quasi-bisimulation R, and any ordinary relation S, $R \subseteq S$ and $R \Longrightarrow S$ imply $\phi_{\mathcal{C}}(R) \subseteq \phi_{\mathcal{C}}(S)$ and $\phi_{\mathcal{C}}(R) \Longrightarrow \phi_{\mathcal{C}}(S)$.

For any $t_1,t_2 \in X$, $(t_1,t_2) \in \phi_{\mathcal{C}}(R)$ means that, there exist $x_1:\sigma_1,\ldots,x_n:\sigma_n \rhd M \in \mathcal{C}$, $\Gamma \rhd P_i : \sigma_i, \Gamma \rhd Q_i : \sigma_i (i = 1,\ldots n)$ and $t \in [\![\Gamma]\!]$ such that $t_1 \sim [\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t)$ and $t_2 \sim [\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t)$, where $t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!])$, and for all $i = 1,\ldots,n$, $P_i = Q_i$ or $([\![\Gamma \rhd P_i]\!](t),\ [\![\Gamma \rhd Q_i]\!](t)) \in R_\perp$.

(1) Assume that R is a quasi-bisimulation, i.e. for any $\langle x,y\rangle \in R$, $x \in \mathbf{wdom}(p^\sharp_\alpha)$ iff $y \in \mathbf{wdom}(p^\sharp_\alpha)$ for all $p \in \mathtt{OSPATH}(\boldsymbol{T})$, and $p^\sharp_\alpha(x) = p^\sharp_\alpha(y)$ for all observation paths $p \in \mathtt{OPATH}(\boldsymbol{T})$. To show $\phi_{\mathcal{C}}(R)$ is a quasi-bisimulation, it is sufficient to show that, for any $t_1,t_2 \in X, (t_1,t_2) \in \phi_{\mathcal{C}}(R)$ implies that $t_1 \in \mathbf{wdom}(p^\sharp_\alpha)$ iff $t_2 \in \mathbf{wdom}(p^\sharp_\alpha)$ for all paths p and $p^\sharp_\alpha(t_1) = p^\sharp_\alpha(t_2)$ for all observation paths p.

Suppose $(t_1,t_2) \in \phi_{\mathcal{C}}(R)$. By Proposition 2.6, for any path p,

$$\begin{aligned} t_1 \in \mathbf{wdom}(p^\sharp_\alpha) \text{ iff } [\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha)\\ t_2 \in \mathbf{wdom}(p^\sharp_\alpha) \text{ iff } [\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha)\end{aligned}$$

and, for any observation path p,

$$\begin{aligned} p^\sharp_\alpha(t_1) \;&=\; p^\sharp_\alpha([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t))\\ p^\sharp_\alpha(t_2) \;&=\; p^\sharp_\alpha([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t)).\end{aligned}$$

By Lemma 4.14,

$$\begin{aligned} [\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha) \text{ iff } \forall i = 1,\ldots,m,\ [\![\Gamma \rhd P_i]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha)\\ [\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha) \text{ iff } \forall i = 1,\ldots,m,\ [\![\Gamma \rhd Q_i]\!](t) \in \mathbf{wdom}(p^\sharp_\alpha),\end{aligned}$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

where we assume that $fv(M) = \{x_1, \ldots, x_m\}$. Because $P_i = Q_i$ or $([\![\Gamma \rhd P_i]\!](t),\ [\![\Gamma \rhd Q_i]\!](t)) \in R_\perp$ for all $i = 1, \ldots, n$, and R is a quasi-bisimulation,

$$[\![\Gamma \rhd P_i]\!](t) \in \mathbf{wdom}(p_\alpha^\sharp) \text{ iff } [\![\Gamma \rhd Q_i]\!](t) \in \mathbf{wdom}(p_\alpha^\sharp).$$

And hence, $t_1 \in \mathbf{wdom}(p_\alpha^\sharp)$ iff $t_2 \in \mathbf{wdom}(p_\alpha^\sharp)$ for any path $p \in \mathtt{OSPATH}(\boldsymbol{T})$.

Furthermore, if p is an observation path, for all $i = 1, \ldots, n$, $P_i = Q_i$, i.e $[\![\Gamma \rhd P_i]\!](t) = [\![\Gamma \rhd Q_i]\!](t)$ when $\sigma_i \neq \mathsf{State}$, and $p_\alpha^\sharp([\![\Gamma \rhd P_i]\!](t)) = p_\alpha^\sharp([\![\Gamma \rhd Q_i]\!](t))$ when $\sigma_i = \mathsf{State}$. Thus, by Lemma 4.16,

$$p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t)) = p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t)).$$

Therefore, $p_\alpha^\sharp(t_1) = p_\alpha^\sharp(t_2)$. This completes the proof of that $\phi_{\mathcal{C}}(R)$ is a quasi-bisimulation.

(2) For any ordinary relation $S \subseteq X \times X$, obviously, $R \subseteq S$ implies $\phi_{\mathcal{C}}(R) \subseteq \phi_{\mathcal{C}}(S)$. Then the only thing we have to do is to show $R \Longrightarrow S$ implies $\phi_{\mathcal{C}}(R) \Longrightarrow \phi_{\mathcal{C}}(S)$. That is, we have to show that if $(t_1, t_2) \in \phi_{\mathcal{C}}(R)$ then $(p_\alpha^\sharp(t_1),\ p_\alpha^\sharp(t_2)) \in \phi_{\mathcal{C}}(S)$ when $t_1 \in \mathbf{wdom}(p_\alpha^\sharp)$ and $t_2 \in \mathbf{wdom}(p_\alpha^\sharp)$.

Note that, if $t_1 \in \mathbf{wdom}(p_\alpha^\sharp)$ and $t_2 \in \mathbf{wdom}(p_\alpha^\sharp)$, then $[\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p_\alpha^\sharp)$, $[\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t) \in \mathbf{wdom}(p_\alpha^\sharp)$, $p_\alpha^\sharp(t_1) \sim p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t))$, and $p_\alpha^\sharp(t_2) \sim p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t))$. By Lemma 4.18, there exist $\Gamma_p \rhd M_p : \mathsf{State} \in \mathcal{C}$ and $\rho : \Gamma_p \to \Gamma'$ such that

$$p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t)) \sim [\![\Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}]]\!](t)$$
$$p_\alpha^\sharp([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t)) \sim [\![\Gamma \rhd M_p[\overrightarrow{L}/\overrightarrow{y}]]\!](t),$$

where we assume that $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ and $\mathbf{dom}(\Gamma_p) = \{y_1, \ldots, y_m\}$, and for all $j = 1, \ldots, m$, $N_j = P_i$ and $L_j = Q_i$ when $\rho(y_j) = (x_i, \kappa_1\ *)$, and $N_j = q^\sharp(P_i)$ and $L_j = q^\sharp(Q_i)$ when $\rho(y_j) = (x_i, \kappa_2\ q)$.

Obviously, when $P_i = Q_i$ we have $N_j = L_j$. And if $([\![\Gamma \rhd P_i]\!](t),\ [\![\Gamma \rhd Q_i]\!](t)) \in R_\perp$, then, by $R \subseteq S$, $([\![\Gamma \rhd N_j]\!](t),\ [\![\Gamma \rhd L_j]\!](t)) \in S_\perp$ when $\rho(y_j) = (x_i, \kappa_1\ *)$, and by $R \Longrightarrow S$, $([\![\Gamma \rhd N_j]\!](t),\ [\![\Gamma \rhd L_j]\!](t)) \in S_\perp$ when $\rho(y_j) = (x_i, \kappa_2\ q)$. Thus, for all $j = 1, \ldots, m$, we always have

$$N_j = L_j \text{ or } ([\![\Gamma \rhd N_j]\!](t),\ [\![\Gamma \rhd L_j]\!](t)) \in S_\perp.$$

Moreover, if $t_1 \in \mathbf{wdom}(p_\alpha^\sharp)$ and $t_2 \in \mathbf{wdom}(p_\alpha^\sharp)$, then $t \in \mathbf{wdom}([\![\Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}]]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M_p[\overrightarrow{L}/\overrightarrow{y}]]\!])$, $p_\alpha^\sharp(t_1) \sim [\![\Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}]]\!](t)$ and $p_\alpha^\sharp(t_2) \sim [\![\Gamma \rhd M_p[\overrightarrow{L}/\overrightarrow{y}]]\!](t)$. Therefore, by the definition of $\phi_{\mathcal{C}}(S)$, $(p_\alpha^\sharp(t_1), p_\alpha^\sharp(t_2)) \in \phi_{\mathcal{C}}(S)$. This completes the proof of that $\phi_{\mathcal{C}}(R) \Longrightarrow \phi_{\mathcal{C}}(S)$. □

Note that, we cannot prove the following simpler function $\phi'_{\mathcal{C}}$ is respectful:

$$\begin{aligned}\phi'_{\mathcal{C}}(R) = \{([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!](t), [\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!](t)) \mid\ & x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \mathsf{State} \in \mathcal{C},\\ & \Gamma \rhd P_1 : \sigma_1, \ldots, \Gamma \rhd P_n : \sigma_n, \Gamma \rhd Q_1 : \sigma_1, \ldots, \Gamma \rhd Q_n : \sigma_n,\\ & t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}]]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}]]\!]),\\ & \quad \text{and for all } i = 1, \ldots, n, \text{ if } \sigma_i \neq \mathsf{State} \text{ then } P_i = Q_i, \text{ else } P_i = Q_i \text{ or}\\ & \quad ([\![\Gamma \rhd P_i]\!](t),\ [\![\Gamma \rhd Q_i]\!](t)) \in R_\perp\ \}.\end{aligned}$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Because, we can only use state bisimulation formulas rather than state equations, and thus, by the definition of a state term having faithful progressions, we only have

$$[\![\Gamma \rhd p^\sharp(M[\overrightarrow{P}/\overrightarrow{x}])]\!](t) \ \sim_\perp \ [\![\Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}]]\!](t)$$

rather than

$$[\![\Gamma \rhd p^\sharp(M[\overrightarrow{P}/\overrightarrow{x}])]\!](t) \ = \ [\![\Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}]]\!](t)$$

(see Lemma 4.18). And then, we have to use bisimilarities in the definition of $\phi_{\mathcal{C}}$.

By Theorem 4.23, we have the following result, which can be regarded as a bisimulation up-to context proof method in our language for polynomial coalgebras.

Theorem 4.24. Let Σ be a set of formulas, $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ and $\Gamma \rhd M_1 : \mathsf{State}, \Gamma \rhd M_2 : \mathsf{State}$ be two terms. Then $\Sigma \vDash \Gamma \rhd M_1 \approx M_2 : \mathsf{State}$ if

1. $\Sigma \vDash \Gamma \rhd M_1 \Downarrow M_2 : \mathsf{State}$;
2. for any observation path $p \in \mathtt{OPATH}(\boldsymbol{T})$, $\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \approx p^\sharp(M_2)$; and
3. for any state path $p \in \mathtt{SPATH}(\boldsymbol{T})$, there exist a faithful term $y_1 : \tau_1, \ldots, y_m : \tau_m \rhd M_p : \mathsf{State}$ with respect to Σ, and terms $\Gamma \rhd N_1 : \tau_1, \ldots, \Gamma \rhd N_m : \tau_m, \Gamma \rhd L_1 : \tau_1, \ldots, \Gamma \rhd L_m : \tau_m$ such that

$$\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \approx M_p[\overrightarrow{N}/\overrightarrow{y}] : \mathsf{State}$$
$$\Sigma \vDash \Gamma \rhd p^\sharp(M_2) \approx M_p[\overrightarrow{L}/\overrightarrow{y}] : \mathsf{State},$$

where, for all $j = 1, \ldots, m$, either $N_j = L_j$ or there exist terms $\Gamma \rhd (Q_j)_1 : \sigma_1, \ldots, \Gamma \rhd (Q_j)_n : \sigma_n$ such that $N_j = M_1[\overrightarrow{Q_j}/\overrightarrow{x}]$ and $L_j = M_2[\overrightarrow{Q_j}/\overrightarrow{x}]$.

Proof. We define a relation R on the set of state terms: for two state terms $\Gamma \rhd M_1, \Gamma \rhd M_2$, $(\Gamma \rhd M_1, \Gamma \rhd M_2) \in R$ iff they satisfy the above conditions (1), (2) and (3).

By Lemma 4.14,

$$\Sigma \vDash \Gamma \rhd M_p[\overrightarrow{N}/\overrightarrow{y}] \Downarrow\!\leftrightarrow (N_1 \Downarrow \ \wedge \ \cdots \ \wedge \ N_m \Downarrow)$$
$$\Sigma \vDash \Gamma \rhd M_p[\overrightarrow{L}/\overrightarrow{y}] \Downarrow\!\leftrightarrow (L_1 \Downarrow \ \wedge \ \cdots \ \wedge \ L_m \Downarrow).$$

Because $N_j = L_j$ or $N_j = M_1[\overrightarrow{Q_j}/\overrightarrow{x}], L_j = M_2[\overrightarrow{Q_j}/\overrightarrow{x}]$, (1) implies that we always have

$$\Sigma \vDash \Gamma \rhd N_j \Downarrow\!\leftrightarrow L_j \Downarrow, \quad \text{for } j = 1, \ldots, m$$

And then, (2) and (3) imply $\Sigma \vDash \Gamma \rhd p^\sharp(M_1) \Downarrow p^\sharp(M_2)$ for all paths $p \in \mathtt{OSPATH}(\boldsymbol{T})$.

Let $\mathcal{M}$ be a semantic model with $\mathcal{M} \vDash \Sigma$, and $\mathcal{C}$ be the set consisting of all faithful terms with respect to Σ. For any state path p, (3) implies that

$$[\![p^\sharp(M_1)]\!](t) \sim [\![M_p[\overrightarrow{N}/\overrightarrow{y}]]\!](t) \qquad [\![p^\sharp(M_2)]\!](t) \sim [\![M_p[\overrightarrow{L}/\overrightarrow{y}]]\!](t)$$

for all $t \in \mathbf{wdom}([\![p^\sharp(M_1)]\!]) \cap \mathbf{wdom}([\![p^\sharp(M_2)]\!])$. Therefore, $([\![p^\sharp(M_1)]\!](t), [\![p^\sharp(M_2)]\!](t)) \in \phi_{\mathcal{C}}([\![R]\!])$, where $\phi_{\mathcal{C}}$ is defined as in Theorem 4.23. This means $[\![R]\!] \Longrightarrow \phi_{\mathcal{C}}(R)$. And then $R \subseteq \sim_\Sigma$ by Corollary 4.8. □

Example 4.25. Continue Example 4.21. Let $\Gamma' = [x : \mathsf{State}, y : \mathsf{State}]$ and $\Gamma = [x : \mathsf{State}, y : \mathsf{State}, z : \mathsf{State}]$. First, it is not hard to prove that $\Sigma \vDash \Gamma' \rhd (x \oplus y) \approx (y \oplus x)$ by

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

using Theorem 4.11. Then, using Theorem 4.24, we can prove that

$$\Sigma \vDash \Gamma \rhd x \oplus (y \times z) \approx (x \times y) \oplus (x \times z).$$

On the one hand, from $\Sigma \vDash r_1 : \sigma_{\mathbb{R}}, r_2 : \sigma_{\mathbb{R}}, r_3 : \sigma_{\mathbb{R}} \rhd r_1 \cdot (r_2 + r_3) \approx (r_1 \cdot r_2) + (r_1 \cdot r_3)$, we obtain

$$\Sigma \vDash \Gamma \rhd head(x \oplus (y \times z)) \approx head((x \times y) \oplus (x \times z)).$$

On the other hand, from $\Sigma \vDash \Gamma' \rhd tail(x \times y) \approx ([head(x)] \times tail(y)) \oplus (tail(x) \times y)$ and $\Sigma \vDash \Gamma' \rhd (x \oplus y) \approx (y \oplus x)$, we obtain

$$\begin{aligned}\Sigma \vDash \Gamma \rhd\ tail((x \times y) \oplus (x \times z)) \ \approx\ & ([head(x)] \times tail(y)) \oplus ([head(x)] \times tail(z)) \ \oplus \\ & (tail(x) \times y) \oplus (tail(x) \times z).\end{aligned}$$

Obviously, we have

$$\Sigma \vDash \Gamma \rhd\ tail(x \times (y \oplus z)) \ \approx\ ([head(x) \times tail(y \oplus z)) \oplus (tail(x) \times (y \oplus z)).$$

And hence,

$$\begin{aligned}\Sigma \vDash \Gamma \rhd tail((x \times y) \oplus (x \times z)) \ &\approx\ (x \oplus y)[N_1/x, N_2/y] \\ \Sigma \vDash \Gamma \rhd tail(x \times (y \oplus z)) \ &\approx\ (x \oplus y)[L_1/x, L_2/y],\end{aligned}$$

where N_1, N_2, L_1, L_2 are defined as follows. Let $M_1 = (x \times y) \oplus (x \times z)$ and $M_2 = x \times (y \oplus z)$,

$$\begin{aligned} N_1 &= M_1[[head(x)]/x, tail(y)/y, tail(z)/z] & N_2 &= M_1[tail(x)/x, y/y, z/z] \\ L_1 &= M_2[[head(x)]/x, tail(y)/y, tail(z)/z] & L_2 &= M_1[tail(x)/x, y/y, z/z]. \end{aligned}$$

Therefore, $\Sigma \vDash \Gamma \rhd\ x \oplus (y \times z) \ \approx\ (x \times y) \oplus (x \times z)$ as required by Theorem 4.24. Generally, it is not hard to prove that $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ is a faithful context with respect to Σ, and then the bisimulation up-to proof method for Rutten's stream calculus (c.f. Theorem 4.2 in Rutten (2001)) is an instance of Theorem 4.24.

The following theorem show that a faithful term preserves bisimulations indeed.

Theorem 4.26. Suppose that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \mathsf{State}$ is a faithful term with respect to Σ, and $\Gamma \rhd N_i : \sigma_i, \Gamma \rhd L_i : \sigma_i$, for all $i = 1, \ldots, n$, are terms. Then

$$\Sigma \vDash \Gamma \rhd N_1 \approx L_1 \wedge \cdots \wedge N_n \approx L_n \ \rightarrow\ M[\overrightarrow{N}/\overrightarrow{x}] \approx M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}.$$

Proof. Let $\mathcal{M}$ be a semantic model with $\mathcal{M} \vDash \Sigma$, and $\mathcal{C}$ be the set consisting of all faithful terms with respect to Σ. We define a relation R

$$\begin{aligned} R = \bigcup_{M \in \mathcal{C}} \{\ & (\llbracket \Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}] \rrbracket(t),\ \ \llbracket \Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}] \rrbracket(t)) \ \mid\ t \in \llbracket \Gamma \rrbracket, \\ & x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \mathsf{State},\ \Gamma \rhd P_i : \sigma_i \text{ and } \Gamma \rhd Q_i : \sigma_i\ (i = 1, \ldots, n), \\ & t \in \mathbf{wdom}(\llbracket \Gamma \rhd M[\overrightarrow{P}/\overrightarrow{x}] \rrbracket) \cap \mathbf{wdom}(\llbracket \Gamma \rhd M[\overrightarrow{Q}/\overrightarrow{x}] \rrbracket), \\ & \text{for all } i = 1, \ldots, n, \text{ if } \sigma_i = \mathsf{State},\ \llbracket \Gamma \rhd P_i : \sigma_i \rrbracket(t) \sim_{\perp} \llbracket \Gamma \rhd Q_i : \sigma_i \rrbracket(t), \\ & \quad \text{else } \llbracket \Gamma \rhd P_i : \sigma_i \rrbracket(t) = \llbracket \Gamma \rhd Q_i : \sigma_i \rrbracket(t)\ \ \}. \end{aligned}$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

It is not hard to see that $R \subseteq \phi_{\mathcal{C}}(\sim)$. Because $\phi_{\mathcal{C}}$ preserves progressions and $\sim \Longrightarrow \sim$, $\phi_{\mathcal{C}}(\sim) \Longrightarrow \phi_{\mathcal{C}}(\sim)$. This means $\phi_{\mathcal{C}}$ is a bisimulation, and then $R \subseteq \phi_{\mathcal{C}} \subseteq \sim$.

For all $t \in [\![\Gamma]\!]$, if $t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}]\!]) \cap \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}]\!])$, then

$$([\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}]\!](t),\ [\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}]\!](t)) \in R.$$

By $R \subseteq \sim$, $[\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}]\!](t) \sim [\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}]\!](t)$.

Because $\Gamma' \rhd M' : \mathsf{State}$ is well-defined with respect to Σ, by Lemma 4.14,

$$t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}]]\!]) \text{ iff } \forall i \in \{1,\ldots,m\},\ t \in \mathbf{wdom}([\![\Gamma \rhd N_i]\!])$$

$$t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}]]\!]) \text{ iff } \forall i \in \{1,\ldots,m\},\ t \in \mathbf{wdom}([\![\Gamma \rhd L_i]\!]).$$

where we assume that $fv(M) = \{x_1,\ldots,x_m\}$. And then, for all $i = 1,\ldots,n$, $\Sigma \vDash \Gamma \rhd N_i \approx L_i$. That is, for all $i = 1,\ldots,n$, $t \in \mathbf{wdom}([\![\Gamma \rhd N_i]\!])$ iff $t \in \mathbf{wdom}([\![\Gamma \rhd L_i]\!])$. Thus, $t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}]]\!])$ iff $t \in \mathbf{wdom}([\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}]]\!])$. Therefore,

$$[\![\Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] : \mathsf{State}]\!](t) \sim_{\perp} [\![\Gamma \rhd M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}]\!](t),$$

that is, $\mathcal{M} \vDash \Gamma \rhd M[\overrightarrow{N}/\overrightarrow{x}] \approx M[\overrightarrow{L}/\overrightarrow{x}] : \mathsf{State}$. □

Example 4.27. Continue Example 4.20. Let $\Gamma = [x : \mathsf{State}, y : \mathsf{State}]$. Following from $\Sigma \vDash \Gamma \rhd tail(zip(x,y)) \approx zip(y, tail(x))$, we obtain

$$\Sigma \vDash x : \mathsf{State} \rhd tail(zip(odd(x), even(x))) \approx zip(even(x), tail(odd(x))).$$

Because $\Gamma \rhd zip(x,y)$ is a faithful term with respect to Σ, $\Sigma \vDash x : \mathsf{State} \rhd even(x) \approx odd(tail(x))$, and $\Sigma \vDash x : \mathsf{State} \rhd tail(odd(x)) \approx odd(tail(tail(x)))$, we have

$$\Sigma \vDash x : \mathsf{State} \rhd tail(zip(odd(x), even(x))) \approx zip(odd(tail(x)), even(tail(x))).$$

Then, by Theorem 4.11, we obtain $\Sigma \vDash x : \mathsf{State} \rhd zip(odd(x), even(x)) \approx x$.

Example 4.28. Continue Example 4.22. Using Theorem 4.26, it is easy to verify that

$$\Sigma \vDash x : \mathsf{State} \rhd mirror(mirror(x)) \approx x.$$

Obviously, we have

$$\Sigma \vDash x : \mathsf{State} \rhd mirror(mirror(x)) \Downarrow \ \leftrightarrow\ x \Downarrow$$

$$\Sigma \vDash x : \mathsf{State} \rhd label(mirror(mirror(x))) \approx label(x)$$

$$\Sigma \vDash x : \mathsf{State} \rhd nochild(mirror(mirror(x))) \approx nochild(x).$$

In addition, from the facts that

$$\Sigma \vDash x : \mathsf{State} \rhd lchild(mirror(x)) \approx mirror(rchild(x))$$

$$\Sigma \vDash x : \mathsf{State} \rhd rchild(mirror(x)) \approx mirror(lchild(x)),$$

we obtain

$$\Sigma \vDash x : \mathsf{State} \rhd lchild(mirror(mirror(x))) \approx mirror(rchild(mirror(x)))$$

$$\Sigma \vDash x : \mathsf{State} \rhd rchild(mirror(mirror(x))) \approx mirror(lchild(mirror(x))).$$

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Further, since $x : \mathsf{State} \rhd mirror(x)$ is a faithful term respect to Γ, we have

$$\Sigma \vDash x : \mathsf{State} \rhd mirror(rchild(mirror(x))) \approx mirror(mirror(lchild(x)))$$
$$\Sigma \vDash x : \mathsf{State} \rhd mirror(lchild(mirror(x))) \approx mirror(mirror(rchild(x))).$$

And hence,

$$\Sigma \vDash x : \mathsf{State} \rhd lchild(mirror(mirror(x))) \approx mirror(mirror(lchild(x)))$$
$$\Sigma \vDash x : \mathsf{State} \rhd rchild(mirror(mirror(x))) \approx mirror(mirror(rchild(x))).$$

Then, if we let $M_1 = mirror(mirror(x)), M_2 = x$, we have

$$\Sigma \vDash x : \mathsf{State} \rhd lchild(M_1) \approx M_1[lchild(x)/x]$$
$$\Sigma \vDash x : \mathsf{State} \rhd lchild(M_2) \approx M_2[lchild(x)/x]$$
$$\Sigma \vDash x : \mathsf{State} \rhd rchild(M_1) \approx M_1[rchild(x)/x]$$
$$\Sigma \vDash x : \mathsf{State} \rhd rchild(M_2) \approx M_2[rchild(x)/x].$$

These imply $\Sigma \vDash x : \mathsf{State} \rhd mirror(mirror(x)) \approx x$ by Theorem 4.11.

Note that if the state equations such as

$$x : \mathsf{State} \rhd lchild(mirror(x)) = mirror(rchild(x))$$
$$x : \mathsf{State} \rhd rchild(mirror(x)) = mirror(lchild(x))$$

is permitted, then it is easy to obtain that

$$x : \mathsf{State} \rhd lchild(mirror(mirror(x))) = mirror(mirror(lchild(x))$$
$$x : \mathsf{State} \rhd rchild(mirror(mirror(x))) = mirror(mirror(rchild(x)).$$

And then, we immediately obtain that $x : \mathsf{State} \rhd mirror(mirror(x)) \approx x$ by 4.11. In other words, we do not require that $x : \mathsf{State} \rhd mirror(x)$ preserves bisimulations.

Also, if we ignore the operation $label, nochild$ and regard $lchild, rchild, mirror$ as actions on processes, by the transition rules given in Section 1, we can obtain that

$$x \xrightarrow{lchild} x' \text{ iff } mirror(mirror(x)) \xrightarrow{lchild} mirror(mirror(x'))$$
$$x \xrightarrow{rchild} x' \text{ iff } mirror(mirror(x)) \xrightarrow{rchild} mirror(mirror(x')).$$

And thus, $R = \{(mirror(mirror(x)), x) \mid x \in Var\}$ is a bisimulation on processes as defined in Section 1, where Var is the set of variables to construct terms of the process language. Note that this also does not require that $mirror$ preserves bisimulation. Therefore, we say that the transition rules correspond to state equations rather than state bisimulation formulas.

Example 4.29. Finally, we show an application of faithful terms to prove the unicity of solutions of corecursive equations. Capretta proposes an algorithm to generate ad hoc bisimulations from corecursive equations to prove the unicity of solutions (Capretta 2010). He gives some examples on streams. We observe that all of his examples involve faithful terms in specifying the functions by corecursive equations. For example, he wants to prove that the following corecursive equations of the function $f : \mathsf{State} \to \mathsf{State}$ has at most one

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

solution (c.f. Example 4.1 in Capretta (2010) or Example 1 in Capretta (2011)):

$$\Sigma \vDash x : \mathsf{State} \rhd head(f(x)) \approx head(x)$$
$$\Sigma \vDash x : \mathsf{State} \rhd tail(f(x)) \approx zip(odd(f(even(x))), even(f(odd(x))))$$

where Σ is defined as in Example 4.12. Here, we use our notations to give the corecursive equation. We need two formulas to specify $head(f(x))$ and $tail(f(x))$, respectively, while Capretta use only one equation to combine these two cases. Note that, the function $\ltimes$ in Capretta (2010) is *zip* in this paper, *odd* in Capretta (2010) is *even* in this paper and *even* in Capretta (2010) is *odd* in this paper[†].

To prove the unicity of f, we show that $\Sigma \vDash x : \mathsf{State} \rhd f_1(x) \approx f_2(x)$ if both f_1 and f_2 satisfy the above corecursive equations (i.e. using f_1 and f_2 to replace f in the above equations respectively). Obviously, $\Sigma \vDash x : \mathsf{State} \rhd head(f_1(x)) \approx head(f_2(x))$. Note that

$$\Sigma \vDash x : \mathsf{State} \rhd tail(f_i(x)) \approx zip(odd(f_i(even(x))), even(f_i(odd(x)))), i = 1, 2$$

and

$$zip(odd(f_i(even(x))), even(f_i(odd(x)))) = zip(odd(x), even(y))[f_i(even(x))/x, f_i(odd(x))/y].$$

For proving $\Sigma \vDash x : \mathsf{State} \rhd f_1(x) \approx f_2(x)$, by Theorem 4.24, it is sufficient to show that $x : \mathsf{State}, y : \mathsf{State} \rhd zip(odd(x), even(y))$ is a faithful term with respect to Σ.

It is easy to show this term is well-defined and preserves observation with respect to Σ. To show it has faithful progressions, we note that

$$\Sigma \vDash \Gamma \rhd tail(zip(odd(x), even(y))) \approx zip(even(y), tail(odd(x)))$$
$$\Sigma \vDash y : \mathsf{State} \rhd even(y) \approx odd(tail(x))$$
$$\Sigma \vDash x : \mathsf{State} \rhd tail(odd(x)) \approx odd(tail(tail(x))),$$

where $\Gamma = [x : \mathsf{State}, y : \mathsf{State}]$. Because $zip(x, y)$ is a faithful term,

$$\Sigma \vDash \Gamma \rhd tail(zip(odd(x), even(y))) \approx zip(odd(tail(y)), even(tail(x))).$$

This means $\Gamma \rhd zip(odd(x), even(y))$ is indeed a faithful term, since we can define $\rho : \Gamma \to \Gamma$ as $\rho(x) = (y, \kappa_2\ tail)$ and $\rho(y) = (x, \kappa_2\ tail)$.

However, we cannot show that $\Gamma \rhd zip(even(x), odd(y))$ is a faithful term. This explains why the equations would not have a unique solution if we swap the roles of *odd* and *even* (Capretta 2011).

Moreover, for the corecursive equation in Example 3.1 in Capretta (2010),

$$\Sigma \vDash x : \mathsf{State} \rhd head(f(x)) \approx head(x)$$
$$\Sigma \vDash x : \mathsf{State} \rhd tail(f(x)) \approx zip(f(even(tail(x))), f(odd(tail(x))))$$

[†] Sorry for that we choice different start position of streams. The start position of streams in Capretta (2010) is 0 rather than 1 as in this paper.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

it has at most one solution because $zip(x, y)$ is a faithful term. For the corecursive equation in Example 3.3 in Capretta (2010),

$$\Sigma \vDash x : \mathsf{State} \triangleright head(f(x)) \approx head(x)$$
$$\Sigma \vDash x : \mathsf{State} \triangleright tail(f(x)) \approx zip(tail(x), tail(f(tail(x)))).$$

It is sufficient to show that $x : \mathsf{State}, y : \mathsf{State} \triangleright zip(x, tail(y))$ is a faithful term for proving the unicity of the solution of this corecursive equation.

5. Related work

The coalgebraic specification language presented in this paper can be thought of as a simple version of CCSL (Rothe *et al.* 2001). A major difference between our language and CCSL stands in the use of paths of polynomial functors. Names of class methods are used to construct terms in CCSL for specifying classes in object-oriented programming. From Example 3.6, one may note that paths can give more detailed observations and state transitions than class methods. Most importantly, the relationship between paths and bisimulations enable us to express and reason about bisimulations through using state bisimulation formulas in our language.

There exist other equational approaches (e.g. (Cîrstea 2000; Goldblatt 2003a)), as well as modal logic approaches (e.g. (Goldblatt 2006; Jacobs 2001; Kurz 2001; Moss 1999; Pattinson 2003; Rößiger 2001)), to specify coalgebras. However, none of them provide a way to express and reason about bisimulations directly.

Recently, Bonsangue, Rutten and Silva generalized Kleene's regular expressions to a specification language for polynomial coalgebras, and present a sound and complete (with respect to bisimulation) axiomatization for the equational system of this language (Bonsangue *et al.* 2007, 2009). Actually, an expression in this language indicates how to construct an element of the final coalgebra, and all expressions (modulo equivalence defined in the equational system) form a subcoalgebra of the final coalgebra. Compared with this language, a distinctive feature of our language is that functional operators can be used to construct terms. And then, the bisimulation proof methods are used to reason about the relationship between bisimulations and those operators. It would be interesting to investigate the precise connection between our language and this language, and see whether the bisimulation proof methods could also be formulated in terms of the regular expressions.

Instead of investigating bisimulation proof methods in coalgebraic specification languages, there exist a few categorical descriptions of bisimulation proof principles for coalgebras in the literature.

Lenisa presented a notion of $\boldsymbol{F}$-bisimulation 'up-to $\boldsymbol{T}$', which is a span (R, r_1, r_2) on $\boldsymbol{T}X$ and $\boldsymbol{T}Y$, such that there exists an arrow $\gamma : R \to \boldsymbol{T}R$ making $r_1 : R \to \boldsymbol{T}X, r_2 : R \to \boldsymbol{T}Y$ $\boldsymbol{F}$-coalgebra morphisms, where $(\boldsymbol{T}X, \alpha)$ and $(\boldsymbol{T}Y, \beta)$ are $\boldsymbol{F}$-coalgebras, and $\boldsymbol{T}$ is a monad (Lenisa 1999). Lenisa showed that an $\boldsymbol{F}$-bisimulation 'up-to $\boldsymbol{T}$' is contained in an $\boldsymbol{F}$-bisimulation under certain conditions (c.f. Theorem 3.5 in Lenisa (1999)).

Bartels generalized Lenisa's idea by introducing a notion of λ-bisimulation, which is a span (R, r_1, r_2) on two λ-bialgebras (X, β_X, α) and (Y, β_Y, α_Y), such that there exists

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

an arrow $\gamma : R \rightarrow \boldsymbol{FT}R$ making certain diagrams commutative, where $\lambda : \boldsymbol{TF} \Rightarrow \boldsymbol{FT}$ is a distributive law of the functor $\boldsymbol{T}$ over the functor $\boldsymbol{F}$ (Bartels 2003). Bartels also showed that a λ-bisimulation is contained in an $\boldsymbol{F}$-bisimulation between $\boldsymbol{F}$-coalgebras (X, α_X) and (Y, α_Y) when the underlying category has countable coproducts (c.f. Theorem 4.2 in Bartels (2003)). To construct a λ-bisimulation is probably easier than to construct an $\boldsymbol{F}$-bisimulation. However, it is still hard to verify the required properties of the involved structures for building a distributive law.

Luo introduced a notion of consistent function (Luo 2006), which is close to the notion of sound function in Sangiorgi (1998). Actually, Luo defined a notion of progression by using relation lifting $Rel(\boldsymbol{T})(-)$ (Jacobs 2005) of functor $\boldsymbol{T}$: a relation R progresses to S iff, $(x, y) \in R$ implies $(\alpha(x), \beta(y)) \in Rel(\boldsymbol{T})(S)$, where (X, α) and (Y, β) are $\boldsymbol{T}$-coalgebras. And then, the notion of consistent function in Luo (2006) is essentially the same as the notion of sound function. So, Luo applied Sangiorgi's ideas from labelled transition systems to the coalgebras of the functors whose relation lifting has certain properties (e.g. Proposition 2.2 in Luo (2006)). However, he could not give a bisimulation up-to context proof method, since his study is not based on a specification language.

Our work can be seen as an extension of Luo's work for polynomial coalgebras. The notion of sound function in this paper is essentially equivalent to the notion of consistent function for polynomial coalgebras, but we can provide more powerful sound functions based on faithful contexts to further facilitate the bisimulation proofs for polynomial coalgebras since our work is based on a coalgebraic specification language. Compared with the work presented by Lenisa (1999) and Bartels (2003), unlike λ-bisimulation is an categorical description of bisimulation up-to techniques, we study the bisimulation up-to techniques in a coalgebraic specification language and use them as syntactic rules to reason about the state bisimulation formulas in the language.

6. Conclusions

In this paper, a term calculus based on paths of polynomial coalgebras is presented for building formulas to specify polynomial coalgebras. The fact that bisimulations can be defined by paths enables the specification language to express bisimulations explicitly and to reason about them by using certain bisimulation proof methods as reasoning rules. Sangiorgi's bisimulation up-to proof techniques for labelled transition systems are transformed to reasoning rules in the form of semantic consequence relations for the language.

The major contribution of this paper is twofold: (i) we use paths of polynomial functors to define an equational coalgebraic specification language. As to our knowledge, this is a novel approach to specify properties of polynomial coalgebras. Obviously, the main ideas of the paper can be applied to other algebraic specification languages, that is, paths of polynomial functors can be added to an algebraic specification language as function symbols, and then this algebraic language can be extended to a language to specify polynomial coalgebras. (ii) We provide the first study on bisimulation proof methods in coalgebraic specification languages for the theory of coalgebras. We give path-based definitions of progression and sound function, and introduce the notion of

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

faithful context for our proposed coalgebraic specification language. And then, we are able to apply Sangiorgi's bisimulation up-to proof methods, including bisimulation up-to context proof methods, to this language.

The work presented in this paper is a fundamental study of the path-based specification language for polynomial coalgebras. We shall study the language further in the future, for example, to investigate the connections with other languages, to explore the congruence formats for polynomial coalgebras, such as GSOS-like formats for polynomial coalgebras, and to look at under what conditions for the specification of function symbols a semantic model of the specification can form a λ-bialgebra. We shall use the language to specify more examples of polynomial coalgebras, especially classes in object-oriented programs. Moreover, in order to automate the bisimulation proofs, we shall implement tools for translating the rules induced by the bisimulation proof methods presented in this paper to theorem proving assistant systems (e.g. PVS).

Acknowledgements

Many thanks to the anonymous referees for detailed comments to help us to improve the paper.

References

Bartels, F. (2003) Generalised coinduction. *Mathematical Structures in Computer Science* **13** 321–348.

Bonsangue, M., Rutten, J. and Silva, A. (2007) Regular expressions for polynomial coalgebras. *Technical Report SEN-E0703*, Centrum voor Wiskunde en Informatica (CWI).

Bonsangue, M., Rutten, J. and Silva, A. (2009) An algebra for Kripke polynomial coalgebras. In: *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society, Los Angeles, CA, USA 49–58.

Capretta, V. (2010) Bisimulations generated from corecursive equations. In: Mislove, M. and Selinger, P. (eds.) Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics, *Elsevier Electronic Notes in Theoretical Computer Science* **265** 245–258.

Capretta, V. (2011) Coalgebras in functional programming and type theory. *Theoretical Computer Science* **412** 5006–5024.

Cîrstea, C. (2000) *Integrating Observations and Computations in the Specification of State-based, Dynamical Systems*, Ph.D. thesis, University of Oxford.

Goldblatt, R. (2001) A calculus of terms for coalgebras of polynomial functors. In: Corradini, A., Lenisa, M. and Montanari, U. (eds.) Proceedings of Coalgebraic Methods in Computer Science. *Elsevier Electronic Notes in Theoretical Computer Science* **44** (1) 161–184.

Goldblatt, R. (2003a) Equational logic of polynomial coalgebras. In: Balbiani, P., Suzuki, N.-Y., Wolter, F. and Zakharyaschev, M. (eds.) *Advances in Modal Logic*, volume 4, King's College Publications 149–184.

Goldblatt, R. (2003b) Observational ultraproducts of polynomial coalgebras. *Annals of Pure and Applied Logic* **123** 235–290.

Goldblatt, R. (2006) A modal proof theory for final polynomial coalgebras. *Theoretical Computer Science* **360** 1–22.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Hughes, J. and Jacobs, B. (2004) Simulations in coalgebra. *Theoretical Computer Science* **327** (1–2) 71–108.

Jacobs, B. (1998) Coalgebraic reasoning about classes in object-oriented languages. In: Proceedings of Coalgebraic Methods in Computer Science. *Elsevier Electronic Notes in Theoretical Computer Science* **11** 231–242.

Jacobs, B. (1999) The temporal logic of coalgebras via Galois algebras, *Technical Report CSI-R9906*, Computer Science Institution, University of Nijmegen.

Jacobs, B. (2000) Towards a duality result in coalgebraic modal logic. In: Reichel, H. (ed.) Proceedings of Coalgebraic Methods in Computer Science. *Elsevier Electronic Notes in Theoretical Computer Science* **33** 160–195.

Jacobs, B. (2001) Many-sorted coalgebraic modal logic: a model-theoretical study. *Theoretical Informatics and Applications* **35** (1) 31–59.

Jacobs, B. (2002) Exercises in coalgebraic specification. In: Backhouse, R., Crole, R. and Gibbons, J. (eds.) Algebraic and Coalgebraic Methods in the Mathematics of Program Construction. *Springer Lecture Notes in Computer Science* **2297** 237–280.

Jacobs, B. (2005) *Introduction to Coalgebra: Towards Mathematics of States and Observations*. Book Draft, Available at http://www.cs.ru.nl/∼bart.

Kupke, C. and Pattinson, D. (2011) Coalgebraic semantics of modal logics: An overview. *Theoretical Computer Science* **412** 5070–5094.

Kupke, C. and Venema, Y. (2008). Coalgebraic automata theory: basic results. *Logical Methods in Computer Science* **4** 1–43.

Kurz, A. (2001) Specifying coalgebras with modal logic. *Theoretical Computer Science* **260** (1–2) 119–138.

Lenisa, M. (1999) From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. In: Jacobs, B. and Rutten, J. (eds.) Coalgebraic Methods in Computer Science. *Electronic Notes in Theoretical Computer Science* **19** 1–21.

Luo, L. (2006) An effective coalgebraic bisimulation proof method. In: Proceedings of the 8th Workshop on Coalgebraic Methods in Computer Science. *Elsevier Electronic Notes in Theoretical Computer Science* **164** 105–119.

Moss, L. (1999) Coalgebraic logic. *Annals of Pure and Applied Logic* **96** (1–3) 277–317.

Pattinson, D. (2003) Coalgebraic modal logic: Soundness, completeness and decidability of local consequence. *Theoretical Computer Science* **309** (1–3) 177–193.

Rößiger, M. (1999) Languages for coalgebras on datafunctors. In: Jacobs, B. and Rutten, J. (eds.) Proceedings of Coalgebraic Methods in Computer Science. *Elsevier Electronic Notes in Theoretical Computer Science* **19** 39–60.

Rößiger, M. (2001) From modal logic to terminal coalgebras. *Theoretical Computer Science* **260** 209–228.

Rothe, J., Tews, H. and Jacobs, B. (2001) The coalgebraic class specification language CCSL. *Journal of Universal Computer Science* **7** (2) 175–193.

Rutten, J. (2000) Universal coalgebra: A theory of systems. *Theoretical Computer Science* **249** 3–80.

Rutten, J. (2001) Elements of stream calculus (an extensive exercise in coinduction). In: Brooks, S. and Mislove, M. (eds.) Proceedings of Mathematical Foundations of Programming Semantics. *Elsevier Electronic Notes in Theoretical Computer Science* **45** 1–66.

Sangiorgi, D. (1998) On the bisimulation proof method. *Mathematical Structures in Computer Science* **8** 447–479.

Sangiorgi, D. (2009) On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems* **31** (4) Article 15.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press

Sangiorgi, D. and Walker, D. (2001) *The pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press.

Zhou, X.-C., Li, Y.-J., Li, W.-J., Qiao, H.-Y. and Shu, Z.-M. (2010) Bisimulation proof methods in a path-based specification language for polynomial coalgebras. In: Ueda, K. (eds.) Proceedings of ASIAN Symposium on Program Languages and Systems, APLAS'10. *Springer Lecture Notes in Computer Science* **6461** 239–254.

https://doi.org/10.1017/S0960129513000030 Published online by Cambridge University Press