Tower Induction and Up-To Techniques for CCS with Fixed Points

Steven Schäfer and Gert Smolka

Saarland University,
Saarbrücken, Germany
{schaefer, smolka}@ps.uni-saarland.de

Abstract. We present a refinement of Pous' companion-based coinductive proof technique and apply it to CCS with general fixed points. We construct companions based on inductive towers and thereby obtain a powerful induction principle. Our induction principle implies a new sufficient condition for soundness of up-to techniques subsuming respectfulness and compatibility. For bisimilarity in CCS, companions yield a notion of relative bisimilarity. We show that relative bisimilarity is a congruence, a basic result implying soundness of bisimulation up to context. The entire development is constructively formalized in Coq.

1 Introduction

Coinductive definitions and their associated reasoning principles are one of the basic tools for studying equivalences in programming languages and process calculi. For process calculi, the idea has been developed by Milner [7] in the form of bisimilarity and the bisimulation proof method. In the context of programming languages, coinductive simulations are an important tool in the field of compiler verification [6].

Coinductive definitions can be realized as greatest fixed points of monotone functions on complete lattices. Tarski's [16] construction of greatest fixed points yields a primitive coinduction principle, which is dual to structural induction. Unfortunately, this coinduction principle can be inconvenient in practice because it requires the construction of an often involved invariant. In the context of bisimilarity, these invariants are known as bisimulations and can become quite complicated. This is especially cumbersome in the context of interactive theorem proving. The construction of an appropriate bisimulation is often the lion's share of a bisimilarity proof.

Fortunately, there are several enhancements of the coinductive proof method, which mitigate these problems.

One useful enhancement consists in changing the function underlying a coinductive definition to simplify proofs by coinduction. There is significant room for improvement here, as many different functions can have the same greatest fixed point. These enhancements of the coinductive proof method are known as up-to techniques [7,13]. As we will see, the gains from using up-to techniques can be dramatic. There are simple examples (Section 4) of bisimilarity proofs, where the smallest bisimulation is infinite, yet there is a finite bisimulation up-to.

Recently, Hur et al. [5] have introduced another enhancement of the coinductive proof method in the form of parameterized coinduction. Parameterized coinduction yields both modular and incremental proof principles for coinduction. In addition, Hur et al. show how to combine up-to techniques with parameterized coinduction, which yields an easier way to apply up-to techniques in coinductive proofs.

Pous [12] starts from this combination of parameterized coinduction and up-to techniques to introduce another substantial simplification and extension of the coinductive proof method. Pous shows that for every monotone function there is a canonical best up-to function, its *companion*. Not only does the use of companions lead to simple proof principles for up-to techniques, it also subsumes the work of Hur et al. and allows for a smooth integration of parameterized coinduction. In this context, up-to techniques for the companion are simply functions below the companion.

If we apply the companion construction to bisimilarity, we obtain a notion of *relative bisimilarity*. Intuitively, two processes are bisimilar relative to a relation R if we can show that they are bisimilar under the assumption that R-related processes are bisimilar. Up-to techniques correspond to properties of relative bisimilarity. For example, the statement that relative bisimilarity is a congruence implies the soundness of bisimulation up to context. In fact, the congruence property of relative bisimilarity is a stronger result.

Despite these advances it remains difficult to show the soundness of up-to techniques. Our aim in this paper is to introduce new proof techniques for the companion, which simplify soundness proofs for up-to techniques. We demonstrate our proof techniques with a non-trivial case study.

The key to our results is a novel inductive construction of companions (Section 3). Our construction yields the *tower induction* principle for companions (Theorem 6), which implies a complete characterization of companion-based up-to functions (Lemma 8).

We apply our construction to strong bisimilarity for CCS with fixed points (Section 4). Our main result is that relative bisimilarity extended to open terms is a congruence. This roughly corresponds to the soundness of bisimulation up-to context. To the best of our knowledge, this result has not appeared in the literature before. Our proofs make extensive use of our characterization of up-to functions for the companion.

Beyond these case studies, we combine parameterized coinduction with our inductive construction of companions. This leads to the *parameterized tower induction* principle (Section 5). The accumulation rule of Hur et al. appears as a consequence of parameterized tower induction.

Finally, we report on a Coq formalization of our results (Section 6). The main difference to the paper presentation is in our treatment of binders. We use a de Bruijn representation in the form of J_f-relative monads [2] to distinguish open and closed terms. Following [14], we establish all substitution lemmas using the rules of a convergent rewriting system.

Contributions We consider the tower induction principle for companions and the formal development of open relative bisimilarity for CCS with general recursive definitions the two main contributions of the paper.

2 Lattice Theory Preliminaries

We recall some basic definitions and results about fixed points in complete lattices [4].

A *complete lattice* may be defined as a triple $(A, \leq, \bigcup)$, where $\leq$ is a partial order on A, such that every set $M \subseteq A$ has a supremum $\bigcup M$. For every $M \subseteq A$ we have:

$$\bigcup M \leq y \quad \Leftrightarrow \quad \forall x \in M.\ x \leq y$$

Every complete lattice has a greatest element $\top$, as well as binary suprema $x \cup y$. Arbitrary infima $\bigcap M$ can be obtained as suprema of lower bounds. In particular, every complete lattice has a least element $\bot$, along with binary infima $x \cap y$.

A function f is *monotone* if $f(x) \leq f(y)$ whenever $x \leq y$. For a monotone function $f : A \to A$ we say that x is a *prefixed point* of f if $f(x) \leq x$ and a *postfixed point* if $x \leq f(x)$. An element x is a *fixed point* of f if $f(x) = x$.

By Tarski's theorem [16], every monotone function on a complete lattice has a complete lattice of fixed points. In particular, every monotone function has a greatest fixed point.

Theorem 1 Let f be a monotone function on a complete lattice. The supremum over all postfixed points $\nu f := \bigcup \{\, x \mid x \leq f(x) \,\}$ is the greatest fixed point of f.

3 Towers and Companions

Pous [12] gives a useful characterization of the greatest fixed point as $\nu f = t(\bot)$ using a function t called the companion for f. Parrow and Weber [8] give an ordinal-based construction of the companion in classical set theory. It turns out that the companion can be obtained in constructive type theory with an inductive tower construction [15].

Definition 2 Let f be a function on a complete lattice. The *f-tower* is the inductive predicate T_f defined by the following rules.

$$\frac{x \in T_f}{f(x) \in T_f} \qquad\qquad \frac{M \subseteq T_f}{\bigcap M \in T_f}$$

Using T_f we define t_f, the *companion* of f.

$$t_f(x) \ := \ \bigcap \{\, y \in T_f \mid x \leq y \,\}$$

We will omit the index on T_f and t_f when the function f is clear from the context.

Note that $t(x)$ is the least element of T above x, since the tower is closed under infima. The following are further consequences of the closure under infima.

Fact 3 t is a closure operator with image T.

a) t is monotone
b) $x \leq t(x)$
c) $t(t(x)) = t(x)$
d) $x \in T \leftrightarrow t(x) = x$.

Additionally, since the tower is closed under f, we have:

Fact 4 $f(t(x)) = t(f(t(x)))$

As a consequence of our inductive construction of T, we obtain an induction principle for t.

Definition 5 A predicate P is *inf-closed* if $\bigcap M \in P$ whenever $M \subseteq P$.

Theorem 6 (Tower Induction) Let P be an inf-closed predicate such that $P(t(x))$ implies $P(f(t(x)))$ for all x. Then $P(t(x))$ holds for all x.

Proof Follows from Fact 3, by induction on the derivation of $t_f(x) \in T_f$. ∎

Standard inf-closed predicates include $\lambda x.\ y \leq x$ for a fixed y and $\lambda x.\ g(x) \leq x$ for monotone g. Both instantiations yield useful statements about t.

Using the predicate $\lambda x.\ \nu f \leq x$ in Theorem 6, we can reconstruct the greatest fixed point of f in terms of t.

Lemma 7 If f is monotone, then $\nu f = t_f(\bot)$.

Proof We have $t(\bot) \leq t(f(t(\bot))) = f(t(\bot))$ by monotonicity of t and Fact 4. It follows that $t(\bot) \leq \nu f$.

In the reverse direction we show $\nu f \leq t(x)$ for all x using Theorem 6. It suffices to show that $\nu f \leq x$ implies that $\nu f \leq f(x)$. This follows from $\nu f = f(\nu f)$ and the monotonicity of f. ∎

More generally, we have $t(x) = \nu f$ for all $x \leq \nu f$, since t is monotone and idempotent.

Using the predicate $\lambda x.\ g(x) \leq x$ in Theorem 6, we prove a characterization of the up-to functions for t, i.e., the monotone functions below t.

Lemma 8 (Up-to Lemma) Let g be monotone. Then the following statements are equivalent.

a) $g \leq t$
b) $g \circ t \leq t$
c) $\forall x.\ g(t(x)) \leq t(x) \rightarrow g(f(t(x))) \leq f(t(x))$

Proof The implication from (c) to (b) follows by tower induction. From (b) to (a) we have $g \leq g \circ t \leq t$, by Fact 3 and the monotonicity of g. The implication from (a) to (c) follows from Fact 4. ■

In particular, this shows that f is below t.

Lemma 9 Let f be monotone. Then $f(t(x)) \leq t(x)$.

Proof By Lemma 8 (b) using the monotonicity of f. ■

We now relate our companion construction to Pous' construction [12].

Definition 10 A function g is *compatible* for f if it is monotone and $g \circ f \leq f \circ g$.

Lemma 11 For monotone f, we have $t_f = \bigcup \{\, g \mid g \text{ is compatible for } f \,\}$.

Proof Let g be compatible for f. We have $g(f(t(x))) \leq f(g(t(x)))$ by compatibility, and by Lemma 8 this implies $g \leq t$. Additionally, the companion is compatible for f, since it is monotone by Fact 3 and $t \circ f \leq t \circ f \circ t = f \circ t$ by Fact 3 and Fact 4. ■

In light of Lemma 11, we can see most results in this section as rederivations of results from [12]. The exceptions are Theorem 6 and Lemma 8, which are new results for the companion. In the sequel, we will make extensive use of the new results to show soundness of up-to functions.

4 CCS with Recursive Processes

In this section we apply the companion construction to strong bisimilarity in CCS [7] with fixed point expressions. Using the companion we obtain proof principles analogous to bisimulation up-to context for the extension of bisimilarity to open terms. Our proofs are similar to Milner's proof [7] that strong bisimilarity is a congruence for CCS, yet our results are strictly stronger. We illustrate this by giving a straightforward proof of the well-known fact that weakly guarded equations have unique solutions modulo strong bisimilarity.

The syntax of CCS processes and actions is given by the following grammar.

$$\begin{aligned} P, Q &::= 0 \mid \alpha.P \mid P \parallel Q \mid P + Q \mid (\nu a)P \mid X \mid \mu X.\, P \\ \alpha, \beta &::= a \mid \overline{a} \mid \tau \end{aligned}$$

For the paper presentation we assume that there is some countably infinite type of variables X, Y, Z. The fixed point expression $\mu X.\, P$ binds the variable X in P. We use the standard notions of free and bound variables. We adopt the Barendregt convention and consider processes up to renaming of bound variables.

A *substitution* σ is a mapping from variables to processes. We can *instantiate* a process P under a substitution σ by replacing all free variables according to σ, while keeping the bound variables fixed. We write $P[\sigma]$ for the process P

$$\frac{}{\alpha.P \overset{\alpha}{\rightsquigarrow} P} \qquad \frac{P \overset{\alpha}{\rightsquigarrow} P'}{P \parallel Q \overset{\alpha}{\rightsquigarrow} P' \parallel Q} \qquad \frac{Q \overset{\alpha}{\rightsquigarrow} Q'}{P \parallel Q \overset{\alpha}{\rightsquigarrow} P \parallel Q'} \qquad \frac{P \overset{a}{\rightsquigarrow} P' \qquad Q \overset{\overline{a}}{\rightsquigarrow} Q'}{P \parallel Q \overset{\tau}{\rightsquigarrow} P' \parallel Q'}$$

$$\frac{P \overset{\overline{a}}{\rightsquigarrow} P' \qquad Q \overset{a}{\rightsquigarrow} Q'}{P \parallel Q \overset{\tau}{\rightsquigarrow} P' \parallel Q'} \qquad \frac{P \overset{\alpha}{\rightsquigarrow} P'}{P + Q \overset{\alpha}{\rightsquigarrow} P'} \qquad \frac{Q \overset{\alpha}{\rightsquigarrow} Q'}{P + Q \overset{\alpha}{\rightsquigarrow} Q'} \qquad \frac{P \overset{\alpha}{\rightsquigarrow} P' \qquad \alpha \neq a, \overline{a}}{(\nu a)P \overset{\alpha}{\rightsquigarrow} (\nu a)P'}$$

$$\frac{P[X \mapsto \mu X.\, P] \overset{\alpha}{\rightsquigarrow} Q}{\mu X.\, P \overset{\alpha}{\rightsquigarrow} Q}$$

Fig. 1. Labeled transition system for CCS.

instantiated under σ. The expression $X \mapsto P$ denotes the substitution that replaces the variable X by P. We combine substitutions by juxtaposition.

A process is *closed* if it does not contain any free variables.

The semantics of CCS is given by a labeled transition system (LTS), i.e., an indexed relation between closed processes $P \overset{\alpha}{\rightsquigarrow} Q$. Intuitively, the relation $P \overset{\alpha}{\rightsquigarrow} Q$ means that process P can reduce to Q and perform the action α in a single step. The labeled transition system for CCS is defined inductively by the rules in Figure 1.

Fixed point expressions allow us to specify arbitrary recursive processes. For instance, replication can be expressed as $!P := \mu X.\ X \parallel P$, where X is not free in P. Under this encoding, we have $!P \overset{\alpha}{\rightsquigarrow} Q$ whenever $!P \parallel P \overset{\alpha}{\rightsquigarrow} Q$. Note that this yields an infinitely branching LTS for, e.g., $!a.0$.

We define strong bisimilarity as the greatest fixed point of a function b mapping binary relations into binary relations. First, let s be the function expressing one step of simulation.

$$s(R) := \lambda PQ.\ \forall \alpha P'.\ P \overset{\alpha}{\rightsquigarrow} P' \ \to\ \exists Q'.\, Q \overset{\alpha}{\rightsquigarrow} Q' \wedge R\, P'\, Q'$$

The function b is just simulation in both directions. More precisely, it is the greatest symmetric function below s, where a function between relations is symmetric if it maps symmetric relations to symmetric relations.

$$\begin{aligned} b(R) &:= \lambda R.\ s(R) \wedge s(R^\dagger)^\dagger \\ R^\dagger &:= \lambda PQ.\ R(Q, P) \end{aligned}$$

Bisimilarity is the greatest fixed point of b. We write t for the companion of b. Bisimilarity between processes P, Q is denoted by $P \sim Q$.

We are trying to develop effective proof techniques for bisimilarity. Before we consider how to show bisimilarity using the companion, let us recall the classical bisimulation proof method. Following the literature, the postfixed points of b are called *bisimulations*. Bisimilarity is the union of all bisimulations by Theorem 1. In order to show that $P \sim Q$ holds, it suffices to find a bisimulation R containing the pair (P, Q). The problem with this proof technique is that R can be arbitrarily complicated and has to be explicitly constructed.

Consider two processes A, B such that

$$\begin{aligned} A &\sim (\overline{a}.B) \parallel (a.!A) \\ B &\sim (a.!B) \parallel (\overline{a}.A) \end{aligned}$$

The processes A and B are obviously bisimilar, as parallel composition is commutative and the only difference beyond this is a simple renaming. Yet the smallest bisimulation containing the pair (A, B) is infinite.

Instead of using the definition of bisimilarity, we can use the companion and tower induction. The companion gives us a notion of *relative bisimilarity*, or R-bisimilarity, which we write as $P \sim_R Q$. Intuitively, processes are bisimilar relative to R, if we can show that they are bisimilar, assuming that all R-related processes are bisimilar. In coinductive proofs, we can frequently assume that some processes are bisimilar after a step of reduction. We can express this in terms of relative bisimilarity, by introducing *guarded assumptions* $\circ R$. Given a relation R, we define $\circ R \coloneqq b(t(R))$.

$$\begin{aligned} P \sim Q &\coloneqq (P, Q) \in \nu b && \text{bisimilarity} \\ P \sim_R Q &\coloneqq (P, Q) \in t(R) && \text{relative bisimilarity} \\ P \sim_{\circ R} Q &\ = (P, Q) \in t(b(t(R))) && \text{guarded relative bisimilarity} \end{aligned}$$

The different notions of bisimilarity are related by the following laws, which instantiate lemmas from Section 3.

Fact 12 $\sim_\bot = \sim \subseteq \sim_{\circ R} = \circ R \subseteq \sim_R \supseteq R$

Tower induction gives us a proof principle for showing bisimilarity in terms of relative bisimilarity.

Lemma 13 If $P \sim_R Q$ implies $P \sim_{\circ R} Q$ for all R, then $P \sim Q$.

Proof By Fact 12 together with tower induction using the inf-closed predicate $\lambda R.\ (P, Q) \in R$. ■

Lemma 13 corresponds to the statement that bisimulation up-to the companion is sound. To show that two processes are bisimilar, it suffices to show that they are bisimilar relative to an assumption which states that they are bisimilar after unfolding at least one reduction step.

Phrased in our vocabulary, Pous [12] has shown that relative bisimilarity is a congruence for CCS with replication. This simplifies the proof of the bisimilarity $A \sim B$. By Lemma 13, it suffices to show $A \sim_{\circ R} B$, assuming that $A \sim_R B$. We have

$$\begin{aligned} A &\sim (\overline{a}.B) \parallel (a.!A) \sim (a.!A) \parallel (\overline{a}.B) \\ B &\sim (a.!B) \parallel (\overline{a}.A) \end{aligned}$$

Since $\sim \subseteq \sim_{\circ R}$, and $\sim_{\circ R}$ is transitive, it thus suffices to show that

$$(a.!A) \parallel (\overline{a}.B) \sim_{\circ R} (a.!B) \parallel (\overline{a}.A)$$

Since $\sim_{\circ R}$ is a congruence, this follows from $a.!A \sim_{\circ R} a.!B$ and $\overline{a}.B \sim_{\circ R} \overline{a}.A$. Unfolding the definition of b, we have to show $!A {\sim_R} !B$ and $B \sim_R A$. The former follows by compatibility with replication, the latter follows from the symmetry of $\sim_R$. We conclude that $A \sim B$.

In this case, we can further simplify the proof to avoid unfolding the definition of b. We simply strengthen the compatibility with action prefixes to $P \sim_R Q \to \alpha.P \sim_{\circ R} \alpha.Q$ since action prefixes can perform a step of reduction.

As defined, relative bisimilarity is not a congruence for CCS with recursive processes, since it is only defined for closed terms. In order to proceed, we lift relative bisimilarity to open terms.

Two processes P, Q are in open bisimilarity $P \mathrel{\dot\sim} Q$ if they are bisimilar under all closing substitutions, i.e., substitutions which replace all free variables by closed processes. We write θ for closing substitutions. As before, we also consider the relative variant of open bisimilarity. To distinguish open bisimilarity from ordinary bisimilarity, we will refer to the latter as closed bisimilarity.

$$\begin{aligned} P \mathrel{\dot\sim} Q &\coloneqq \forall\theta.\ P[\theta] \sim Q[\theta] && \text{open bisimilarity} \\ P \mathrel{\dot\sim_R} Q &\coloneqq \forall\theta.\ P[\theta] \sim_R Q[\theta] && \text{open relative bisimilarity} \end{aligned}$$

Open and closed (relative) bisimilarity coincide for closed processes.

Even though open bisimilarity is not defined coinductively, we obtain a reasoning principle analogous to Lemma 13 using tower induction.

Lemma 14 If $P \mathrel{\dot\sim_R} Q$ implies $P \mathrel{\dot\sim_{\circ R}} Q$ for all R, then $P \mathrel{\dot\sim} Q$.

Proof Tower induction with $P(R) = \forall\theta.\ (P[\theta], Q[\theta]) \in R$. ∎

In the remainder of this section we show that relative open bisimilarity is a congruence.

Lemma 15 Relative open bisimilarity is an equivalence relation.

Proof By the definition of relative open bisimilarity, it suffices to show that relative bisimilarity is an equivalence relation. This follows by tower induction. The intersection of a family of equivalence relations is an equivalence relation and it is easy to see that $b(R)$ is an equivalence relation if R is. ∎

For the compatibility with the various connectives of CCS, we define local context operators as follows:

$$\begin{aligned} c^{\cdot}(R) &\coloneqq \{\, (\alpha.P, \alpha.Q) \mid R(P, Q) \,\} \\ c^{\parallel}(R) &\coloneqq \{\, (P_1 \parallel Q_1, P_2 \parallel Q_2) \mid R(P_1, P_2), R(Q_1, Q_2) \,\} \\ c^{+}(R) &\coloneqq \{\, (P_1 + Q_1, P_2 + Q_2) \mid R(P_1, P_2), R(Q_1, Q_2) \,\} \\ c^{\nu}(R) &\coloneqq \{\, ((\nu a)P, (\nu a)Q) \mid R(P, Q) \,\} \end{aligned}$$

Compatibility under all contexts not containing fixed point expressions corresponds to the statements $c(\sim_R) \subseteq \sim_R$ for all local context operators. These results have already been shown in [12] using a second order companion construction. We give alternative proofs using the up-to lemma (Lemma 8).

As in [12] we encapsulate some of the symmetries in the problem using the following lemma.

Fact 16 [12] Let g be symmetric and $g(b(R)) \leq s(R)$. Then $g(b(R)) \leq b(R)$.

For compatibility with action prefixes we show a slightly stronger statement.

Lemma 17 If $P \mathrel{\dot\sim_R} Q$, then $\alpha.P \mathrel{\dot\sim_{\circ R}} \alpha.Q$ and $\alpha.P \mathrel{\dot\sim_R} \alpha.Q$.

Proof We have $c^{\cdot} \leq b$, by unfolding the definitions. The statement follows using Fact 12, since $c^{\cdot}(t(R)) \subseteq b(t(R)) \subseteq \sim_R$. ■

All remaining proofs follow the same pattern. After applying Lemma 8 and Fact 16 (which allows us to focus on establishing the simulation condition), our work boils down to a simple case analysis. We illustrate only the case of parallel composition in detail.

Lemma 18 $P_1 \mathrel{\dot\sim_R} P_2 \to Q_1 \mathrel{\dot\sim_R} Q_2 \to P_1 \parallel Q_1 \mathrel{\dot\sim_R} P_2 \parallel Q_2$

Proof We show $c^{\parallel}(\sim_R) \subseteq \sim_R$ using Lemma 8. Assume that the statement holds for a relation R and all P_1, P_2, Q_1, Q_2. We will refer to this as the *coinductive hypothesis.*

We have to show that $P_1 \sim_{\circ R} P_2$ and $Q_1 \sim_{\circ R} Q_2$ imply $P_1 \parallel Q_1 \sim_{\circ R} P_2 \parallel Q_2$. By Fact 16 it suffices to show this for one step of simulation. Let $P_1 \parallel Q_1 \overset{\alpha}{\leadsto} U$, we have to find a process V such that $P_2 \parallel Q_2 \overset{\alpha}{\leadsto} V$ and $U \sim_R V$.

We proceed by case analysis on $P_1 \parallel Q_1 \overset{\alpha}{\leadsto} U$. Formally, there are four cases to consider of which two follow by symmetry.

- *Communication between P_1 and Q_1.* We have $P_1 \overset{a}{\leadsto} P_1'$, $Q_1 \overset{\overline{a}}{\leadsto} Q_1'$, $\alpha = \tau$, and $U = P_1' \parallel Q_1'$. By the assumptions $P_1 \sim_{\circ R} P_2$ and $Q_1 \sim_{\circ R} Q_2$ there are P_2', Q_2' such that $P_2 \overset{a}{\leadsto} P_2'$, $Q_2 \overset{\overline{a}}{\leadsto} Q_2'$, $P_1' \sim_R P_2'$ and $Q_1' \sim_R Q_2'$. We pick $V = P_2' \parallel Q_2'$, as $P_2 \parallel Q_2 \overset{\tau}{\leadsto} P_2' \parallel Q_2$. The statement $P_1' \parallel Q_1' \sim_R P_2' \parallel Q_2'$ follows from the coinductive hypothesis.
- *Reduction in P_1.* We have $P_1 \overset{\alpha}{\leadsto} P_1'$ and $U = P_1' \parallel Q_1$. By assumption, $P_2 \overset{\alpha}{\leadsto} P_2'$ and $P_1' \sim_R P_2'$. We pick $V = P_2' \parallel Q_2$, as $P_2 \parallel Q_2 \overset{\alpha}{\leadsto} P_2' \parallel Q_2$. The statement $P_1' \parallel Q_1 \sim_R P_2' \parallel Q_2$ follows from the coinductive hypothesis if we can show $Q_1 \sim_R Q_2$. This follows from the assumption that $Q_1 \sim_{\circ R} Q_2$ and Fact 12. ■

Finally, we have to show that bisimilarity is compatible with fixed point expressions. Schematically, the proof remains similar to the proof of Lemma 18, except that we replace the case analysis on the reduction relation by a nested induction.

Substitutions add an additional complication, however, since reducing a fixed point expression instantiates a variable. Intuitively, this means that we have to show that open bisimilarity is compatible with fixed points and instantiation at the same time.

First, let us consider the following context operators on closed relative bisimilarity.

$$\begin{aligned} c^{\mu}(R) &\coloneqq \{\, (\mu X.\, P,\ \mu X.\, Q) \mid \\ &\qquad \forall S \text{ closed. } R(P[X \mapsto S],\ Q[X \mapsto S]) \,\} \\ c^{[]}(R) &\coloneqq \{\, (P[\theta_1], P[\theta_2]) \mid \forall x.\ R(\theta_1(x), \theta_2(x)) \,\} \end{aligned}$$

If $c^{\mu}(R) \subseteq R$, then we can show that two fixed points $\mu X.\ P$, $\mu X.\ Q$ are related if they are related whenever we substitute the same closed process for X in both P and Q. We are implicitly assuming that X is the only free variable in P, Q.

If $c^{[]}(R) \subseteq R$, then R is compatible under related closing substitutions. Specifically, if P is an open process and θ_1, θ_2 are two pointwise related closing substitutions, we can show that $P[\theta_1]$ and $P[\theta_2]$ are related.

One half of the relationship between c^{μ} and $c^{[]}$ is captured by the following lemma.

Lemma 19 If $c^{\mu}(\sim_R) \subseteq \sim_R$, and $\theta_1(x) \sim_R \theta_2(x)$ for all x, then $P[\theta_1] \sim_R P[\theta_2]$ for all P.

Proof By induction on P. The cases for action prefixes, choice, parallel composition and restriction follow from the compatibility of $\sim_R$ with the structure of P. The case for variables follows from the assumption on θ_1 and θ_2.

Finally, let $P = \mu X.\ Q$. By compatibility with μ it suffices to show that $Q[\theta_1, X \mapsto S] \sim_R Q[\theta_2, X \mapsto S]$. This follows by induction, since the extended substitutions are related by reflexivity of $\sim_R$. ■

In fact, we do have $c^{\mu}(\sim_R) \subseteq \sim_R$, and the first assumption in the previous lemma is vacuously true.

Lemma 20 If $P[X \mapsto S] \sim_R Q[X \mapsto S]$ holds for all R and closed S, where X is the only free variable in P, Q, then $\mu X.P \sim_R \mu X.Q$.

Proof We show $c^{\mu}(\sim_R) \subseteq \sim_R$ by Lemma 8. We can assume that the statement holds for a relation R and have to show that it holds for $\circ R$.

It suffices to show that $Q[X \mapsto \mu X.\, P] \sim_{\circ R} Q[X \mapsto \mu X.\, Q]$. The statement then follows from $\mu X.\, P \sim P[X \mapsto \mu X.\, P]$ and transitivity:

$$\mu X.\ P \sim P[X \mapsto \mu X.\ P] \sim_{\circ R} Q[X \mapsto \mu X.\ P] \sim_{\circ R} Q[X \mapsto \mu X.\ Q] \sim \mu X.\ Q$$

where in the second step, we have used the assumption that P and Q are $\circ R$-related under the same closing substitution.

What is left to show is almost compatibility under instantiation with related substitutions. We show that $Q_0[X \mapsto \mu X.\ P] \sim_{\circ R} Q_0[X \mapsto \mu X.\ Q]$ for all Q_0. By Fact 16, it suffices to show this statement for one step of simulation. Let $Q_0[X \mapsto \mu X.\ P] \overset{\alpha}{\rightsquigarrow} Q'$. We have to find Q'' such that $Q_0[X \mapsto \mu X.\ Q] \overset{\alpha}{\rightsquigarrow} Q''$ and $Q' \sim_R Q''$.

We proceed by induction on the derivation of $Q_0[X \mapsto \mu X.\ P] \overset{\alpha}{\rightsquigarrow} Q'$. There are nine cases to consider in total. We illustrate three representative cases.

- $Q_0 = S \parallel T$ and $S[X \mapsto \mu X.\ P] \overset{\alpha}{\rightsquigarrow} S'$. By the inductive hypothesis, there is an S'' such that $S[X \mapsto \mu X.\ Q] \overset{\alpha}{\rightsquigarrow} S''$ and $S' \sim_R S''$. It suffices to show that $S' \parallel T[X \mapsto \mu X.\ P] \sim_R S'' \parallel T[X \mapsto \mu X.\ Q]$.

  This follows from compatibility with parallel composition and instantiation (Lemma 19). We have $S' \sim_R S''$ by assumption and $\mu X.\, P \sim_R \mu X.\, Q$ follows from the coinductive hypothesis.
- $Q_0 = \mu Y.\, S$ and $S[Y \mapsto \mu Y.\, S][X \mapsto \mu X.\, P] \overset{\alpha}{\rightsquigarrow} S'$. By the inductive hypothesis, there is an S'' such that $S[Y \mapsto \mu Y.\, S][X \mapsto \mu X.\, Q] \overset{\alpha}{\rightsquigarrow} S''$ and $S' \sim_R S''$, which is what we needed to show.
- $Q_0 = X$ and $P[X \mapsto \mu X.\, P] \overset{\alpha}{\rightsquigarrow} P'$. By the inductive hypothesis, there is a P'' such that $P[X \mapsto \mu X.\ Q] \overset{\alpha}{\rightsquigarrow} P''$ and $P' \sim_R P''$.

  By the assumption on P and Q there is a Q' such that $Q[X \mapsto \mu X.\ Q] \overset{\alpha}{\rightsquigarrow} Q'$ and $P'' \sim_R Q'$. We have $P' \sim_R Q'$ by transitivity of $\sim_R$ and the statement follows. ■

At this point we have all we need to prove that open relative bisimilarity is a congruence.

Theorem 21 Open relative bisimilarity is a congruence.

Proof Congruence under action prefixes, choice, parallel composition and restrictions follows from the corresponding statements for relative bisimilarity, and the fact that instantiation is homomorphic in the process structure.

For fixed points, we have to show that $P \mathrel{\dot\sim_R} Q$ implies $\mu X.\, P \mathrel{\dot\sim_R} \mu X.\, Q$. Unfolding the definitions, we have to show that $(\mu X.\, P)[\theta] \sim_R (\mu X.\, Q)[\theta]$. Note that θ leaves X invariant, since X is bound.

By Lemma 20, it suffices to show $P[\theta, X \mapsto S] \sim_R Q[\theta, X \mapsto S]$ for all closed processes S. This follows from $P \mathrel{\dot\sim_R} Q$, with an extended closing substitution. ■

Furthermore, we can use Lemma 19 to show that $\dot\sim_R$ is compatible with instantiation.

Theorem 22 Let σ_1, σ_2 be substitutions such that $\sigma_1(x) \mathrel{\dot\sim_R} \sigma_2(x)$ for all x. If $P \mathrel{\dot\sim_R} Q$, then $P[\sigma_1] \mathrel{\dot\sim_R} Q[\sigma_2]$.

Proof By the definition of $P \mathrel{\dot\sim_R} Q$, we have $P[\sigma_1] \mathrel{\dot\sim_R} Q[\sigma_1]$. The statement follows from Lemma 19, Lemma 20 and transitivity. ■

As a small additional application, we use Theorem 21 to show that weakly guarded equations have unique solutions in CCS.

A *context* C is a process with holes.

$$C ::= [\,] \mid P \mid \alpha.C \mid C + C \mid C \parallel C \mid (\nu a)C \mid \mu X.\, C$$

A context is *weakly guarded* if every hole appears under an action prefix, where the action in question may be τ. A context C can be *filled* with a process P resulting in a process $C[P]$, by replacing every hole in C with P. For example, the context $C = \alpha.[\,] \parallel \tau.[\,]$ is weakly guarded and we have $C[X] = \alpha.X \parallel \tau.X$.

Weakly guarded equations are bisimilarities of the form $P \sim C[P]$, for weakly guarded contexts C. By a result of Milner [7], such equations have unique solutions. Intuitively, this is because reduction must take a step before reaching a hole. We can formalize this intuition in terms of relative bisimilarities, which leads to a simple proof.

Lemma 23 If C is weakly guarded and $P \sim_R Q$, then $C[P] \sim_{\circ R} C[Q]$.

Proof By induction on C, using Theorem 21 and in particular using Lemma 17 to move from guarded relative bisimilarity to relative bisimilarity. ∎

Lemma 24 (Unique Solutions) If C is weakly guarded, and P, Q are two processes such that $P \sim C[P]$, and $Q \sim C[Q]$ then $P \sim Q$.

Proof By Lemma 13, it suffices to show $P \sim_{\circ R} Q$, assuming that $P \sim_R Q$. Using Lemma 23, we have $C[P] \sim_{\circ R} C[Q]$ and the statement follows by transitivity, as $P \sim C[P] \sim_{\circ R} C[Q] \sim Q$. ∎

5 Parameterized Tower Induction

We return to the abstract setting and establish an induction principle similar in spirit to Hur et al.'s parameterized coinduction [5].

Lemma 25 (Parameterized Tower Induction) Let u be an element of a complete lattice A, f a monotone endofunction, and P an inf-closed predicate. We have $P(t(u))$ and $P(f(t(u)))$, whenever

$$\forall x.\ u \leq t(x) \to P(t(x)) \to P(f(t(x))).$$

Proof The statement $P(f(t(u)))$ follows from $P(t(u))$ and the assumption together with Fact 3.

To show $P(t(u))$, we generalize the statement to $\forall x.\ Q(t(x))$ for the inf-closed predicate $Q(x) = u \leq x \to P(x)$. By tower induction, it suffices to show that $P(f(t(x)))$ follows from $u \leq t(x) \to P(t(x))$ and $u \leq f(t(x))$. From Lemma 9 we know that $u \leq f(t(x)) \leq t(x)$. Thus $P(t(x))$ holds and $P(f(t(x)))$ follows by assumption. ∎

Hur et al. [5] implement parameterized coinduction with an accumulation rule for parameterized fixed points. Pous shows that the same accumulation rule is applicable to the companion. We present a different proof of the accumulation rule by instantiating Lemma 25 with the predicate $\lambda x.\ y \leq x$.

Lemma 26 For monotone f we have $x \leq f(t(x \cup y)) \ \leftrightarrow\ x \leq f(t(y))$.

Proof The right-to-left direction follows from $y \leq x \cup y$ together with the monotonicity of t and f. In the left-to-right direction we use Lemma 25. It suffices to show that

$$\forall z.\ y \leq t(z) \to x \leq t(z) \to x \leq f(t(z)).$$

Combining the two assumptions, we have $x \cup y \leq t(z)$. Using Fact 3, this is equivalent to $t(x \cup y) \leq t(z)$. The statement follows from $x \leq f(t(x \cup y))$ and the monotonicity of f. ■

Together with Lemma 7, Lemma 26 implies a sound and complete coinduction principle.

Fact 27 If f is monotone, then $x \leq f(t(x)) \leftrightarrow x \leq \nu f$.

Proof We have $\nu f = f(\nu f) = f(t(\bot))$. ■

Pous observed that every function below the companion is a sound up-to function [13] for f. This is a consequence of Fact 27.

Definition 28 g is a *sound up-to function* for f, if $x \leq \nu f$ whenever $x \leq f(g(x))$.

Lemma 29 If $g \leq t_f$, then g is a sound up-to function for f.

Proof This follows from Fact 27: $x \leq f(g(x)) \leq f(t(x))$. ■

6 Coq Formalization

All results in this paper have been formalized in Coq. We make use of the Ssreflect plugin and library, for its improved tactic language and the formalization of finite types (for J_f-relative monads). To avoid working with pre-lattices, we assume propositional and functional extensionality. The development is available at: `www.ps.uni-saarland.de/extras/companions`.

The main divergence of the formalization from the paper is our treatment of variable binding in CCS with fixed points. We represent variable binding using a de Bruijn representation. Since we often have to distinguish between open and closed terms we index our term language with an upper bound on the number of free variables. This technique was first used by Adams [1], and later thoroughly explained by Alternkirch et al. in the framework of relative monads [2]. Using the terminology of Altenkirch et al., we formalize terms as J_f-relative monads.

In addition to the laws of a J_f-relative monad, we show all equations from [14]. This allows us to show all substitution lemmas by rewriting.

7 Related Work

Coinduction Hur et al. [5] introduce parameterized coinduction as an incremental proof technique for coinduction. For a monotone function f on a complete lattice, they construct the function $G_f(x) = \nu(\lambda y.\, f(x \cup y))$. They show that G_f can be used for modular and incremental coinductive reasoning and describe several examples and extensions.

One extension of parameterized coinduction incorporates up-to techniques. Specifically, Hur et al. consider respectful up-to functions. Respectfulness is another sufficient criterion for soundness of up-to functions. They use the fact that the set of respectful up-to functions is closed under union to construct the greatest respectful up-to function t. The parameterized fixed point $G_{f \circ t}$ turned out to obey an “unfolding” lemma, which allowed them to freely use any respectful up-to technique in a coinductive proof.

Recently, Pous [12] noticed that the greatest compatible up-to function already admits the parameterized coinduction principle. It turns out that the greatest compatible and the greatest respectful up-to function coincide. Moreover we have $f \circ t = G_{f \circ t}$. This means that the function t is everything we require for incremental and modular coinductive proofs compatible with up-to techniques.

Pous dubbed the greatest compatible up-to function the companion.

At the same time, Parrow and Weber [8] considered the greatest respectful function for strong bisimilarity in the context of classical set theory. Their construction avoids the quantification over respectful functions by using the theory of ordinals in set theory. They use that bisimilarity may be defined by transfinite iteration to construct the companion for bisimilarity.

Formally, the idea is that if κ is an ordinal larger than the cardinality of the underlying lattice, then $f^\kappa(\top)$ is the greatest fixed point of f. This can be used to construct the companion as $t_f(x) = \bigcap \{ f^\alpha(\top) \mid x \le f^\alpha(\top),\ \alpha \text{ ordinal} \}$.

The tower construction [15] may be seen as the type theoretic analogue of transfinite iteration in set theory. Under this view, we define the set of points reachable from $\top$ by transfinite f-iteration as an inductive predicate $T \approx \{ f^\alpha(\top) \mid \alpha \text{ ordinal} \}$.

Up-To Techniques The study of up-to techniques for bisimilarity originates with Milner [7]. Milner considers bisimulation up-to bisimilarity to keep proofs of bisimilarity manageable. Practical applications usually require combining several different up-to functions. Even our toy example in Section 4 requires bisimulation up-to context and bisimilarity to mimic the proof using the companion.

One problem with using only sound up-to functions in the sense of Definition 28 is that sound up-to functions do not compose. This drawback led Sangiorgi [13] to propose the notion of respectful up-to functions. Respectful up-to functions are sound and closed under composition and union.

Sangiorgi [13] studies bisimilarity, but notes that the same definition of respectfulness makes sense in the more general context of greatest fixed points in complete lattices.

Pous [11] extends and simplifies the work of Sangiorgi by abstracting it to the setting of complete lattices and by introducing the notion of compatibility. This abstraction yields concrete gains, as the set of compatible maps forms another complete lattice. In particular, this implies that we can use up-to techniques to establish soundness of up-to techniques. Pous refers to this as “second order techniques”.

Recently, Pous [12] adapted this development to the companion. For every companion t, there exists a second-order companion, classifying the compatible up-to functions. Pous uses the second-order companion extensively to show soundness of bisimulation up-to context for CCS with replication and other case studies.

8 Conclusions and Future Work

We have presented a tower based construction of the companion of a monotone function on a complete lattice. The new tower induction principle derived from this construction allows us to show a number of improved results for companions. We instantiate the abstract lattice theoretic development with strong bisimilarity in CCS with general recursive processes. This instantiation yields a particularly simple proof system for bisimilarity and we show the admissibility of reasoning up-to context about bisimilarity. Our results imply the classical soundness result for bisimulation up-to context in CCS with replication.

There are several avenues for future work.

All case studies in this paper consider up-to techniques for strong bisimilarity in CCS. It is well known [10] that the case of weak bisimilarity is much more subtle. If we try to adapt the development in Section 4 to weak bisimilarity, we find that relative weak bisimilarity is not transitive and not compatible with choice. This mirrors the failure of soundness of weak bisimulation up-to weak bisimilarity and the fact that weak bisimilarity is not compatible with choice.

Despite these problems, there are useful up-to techniques for weak bisimilarity. Pous [10] developed weak bisimulation up-to elaboration, which combines weak bisimulations with a limited form of unfolding under a termination hypothesis. At this point it is not clear whether these techniques yield corresponding reasoning principles for relative weak bisimilarity.

There are also open questions concerning the companion construction itself.

Assuming the axiom of excluded middle, it can be shown [15] that T is well-ordered. In particular, in classical type theory, we can use this to expand the tower induction principle to all predicates which are closed under infima of well-ordered subsets. However, this principle is not provable in constructive type theory [3].

It might yet be possible to show a slightly weaker statement constructively. Pataraia [9] gives a constructive proof of Tarski's theorem for least-fixed points on directed complete partial orders. We conjecture that a similar construction can be used to extend the tower induction principle to predicates which are closed under infima of lower directed subsets.

References

1. Robin Adams. Formalized metatheory with terms represented by an indexed family of types. In *Types for Proofs and Programs*, volume 3839 of *LNCS*, pages 1–16. Springer Berlin Heidelberg, 2006.
2. Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. In *International Conference on Foundations of Software Science and Computational Structures*, volume 6014 of *LNCS*, pages 297–311. Springer Berlin Heidelberg, 2010.

3. Andrej Bauer and Peter Lefanu Lumsdaine. On the Bourbaki–Witt principle in toposes. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 155, pages 87–99. Cambridge Univ Press, 2013.
4. Brian Davey and Hilary Priestley. *Introduction to lattices and order*. Cambridge University Press, 2002.
5. Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. The power of parameterization in coinductive proof. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 193–206, 2013.
6. Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.
7. Robin Milner. *Communication and concurrency*, volume 84. Prentice Hall, 1989.
8. Joachim Parrow and Tjark Weber. The largest respectful function. *Logical Methods in Computer Science*, 12(2), 2016.
9. Dito Pataraia. A constructive proof of Tarski's fixed-point theorem for dcpo's. Presented in the 65th Peripatetic Seminar on Sheaves and Logic, Aarhus, Denmark, November 1997.
10. Damien Pous. Weak bisimulation up to elaboration. In *CONCUR 2006–Concurrency Theory*, volume 4137 of *LNCS*, pages 390–405. Springer, 2006.
11. Damien Pous. Complete lattices and up-to techniques. In *Asian Symposium on Programming Languages and Systems*, volume 4867 of *LNCS*, pages 351–366. Springer, 2007.
12. Damien Pous. Coinduction all the way up. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 307–316, New York, NY, USA, 2016. ACM.
13. Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(05):447–479, 1998.
14. Steven Schäfer, Gert Smolka, and Tobias Tebbi. Completeness and decidability of de Bruijn substitution algebra in Coq. In *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 67–73. ACM, 2015.
15. Gert Smolka, Steven Schäfer, and Christian Doczkal. Transfinite constructions in classical type theory. In *International Conference on Interactive Theorem Proving*, volume 9236 of *LNCS*, pages 391–404. Springer, 2015.
16. Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.