# Basic Inference Rules for Algebraic and Coalgebraic Specifications

Peter Padawitz
peter.padawitz@udo.edu
University of Dortmund, Germany
September 6, 2002

**Abstract.** We give a survey of proof and computation rules, which, viewed from a long-standing experience with formal design methods, seem to be necessary and sufficient for executing as well as verifying algebraic or coalgebraic data type specifications. Based on many-sorted logics, swinging types [17, 21] provide the uniform syntax and the semantics with respect to which the set of rules is sound and, to a certain extent, complete.

## 1 Introduction

**Swinging types** (STs) and **cospecifications** [17, 21] are axiomatic specification formalisms that combine equational, Horn and modal logic with functional, relational and state-oriented programming. The semantics of an **algebraic swinging type** is a Herbrand model consisting of constructor terms. The (standard) model of a cospecification is a final coalgebra consisting of *context interpretations*. Algebraic STs are hierarchical because they are constructed as a sequence of stepwise extensions that start out from *basic Horn specifications* [21] whose axioms are Horn clauses. Coalgebraic specifications are integrated into the sequence as the *domain completions* of **coalgebraic swinging types**. A coalgebraic ST $CST$ combines an algebraic ST with a cospecification $CSP$. The domain completion of $CST$ is an algebraic ST whose set of constructors includes the elements of the model of $CSP$.

Herbrand models seem to be restrictive, but, in fact, the (ground) terms a Herbrand model consists of may denote all kinds of data: static object structures, sets of attribute values, structured states, object histories, processes, "infinite" objects, distinguished elements of an uncountable domain, etc. STs and their models take into account the conceptual and formal differences between constructors, destructors and other defined functions; between *static* predicates (properties of individual states) and *dynamic* ones (transition systems); between structural and behavioral equalities; and between static predicates that are interpreted as least fixpoints ($\mu$*-predicates*, e.g. structural equalities) and those that are interpreted as greatest fixpoints ($\nu$*-predicates*, e.g. behavioral equalities). While all these functions or relations may be part of an algebraic ST, cospecifications provide additional, often dual features: *coinductive axiomatizations* of functions instead of the inductive definitions that are typical for an algebraic ST; the representation of an uncountable domain of "infinite" objects in contrast to the finitely generated domain of an algebraic ST (unless it is not the domain completion of a coalgebraic ST); the restriction to a particular subdomain in terms of *assertions* and not in terms of *generating* functions that build up *reducts* of algebraic STs.

Deductive reasoning about formal specifications means not only proving requirements, but also solving queries and evaluating functional expressions. In fact, many inference rules are used for all three purposes. In the case of swinging types, the rules applied most frequently are the **unfolding** rules that "expand" functions or logical predicates occurring in terms to be evaluated or formulas to be solved or proved by "applying" axiomatic definitions. Unfolding rules for predicates are equivalence transformations because the axioms define the predicates as unique (least or greatest) fixpoints. The same holds true for defined functions if the specification is *functional*, i.e. each ground term has a unique structurally equivalent *normal form* (ground term consisting of constructors). Functionality splits into *completeness* (existence of normal forms) and *consistency* (uniqueness of normal forms). If an ST is complete, then its consistency is equivalent to rewriting confluence and thus to the completeness of the reduction calculus (= resolution + rewriting) that is associated with any ST. Weak syntactic criteria for confluence that also provide guidelines for the stepwise design of an ST can be found in [16, 19, 18].

Functionality also ensures that **constructor elimination** and **clash** rules, which reduce the number of constructors in a formula to be proved or solved, are equivalence transformations. Another purpose of these rules is the removal of *structural* equations and inequations. Since *behavioral* equalities are logical predicates, they are handled by unfolding rules. The axioms for behavioral equalities are co-Horn clauses that express their compatibility with destructors and transition predicates and their *inverse compatibility* with *object constructors*. This makes object constructors the behavioral counterpart of usual constructors and there are corresponding **object constructor elimination** and **clash** rules that allow us to remove behavioral equations.

The correctness of unfolding rules for logical predicates follows from the predicates' interpretation as solutions of their axioms. **Fixpoint induction** and **coinduction** rules also exploit the fact that the solutions are *least* or *greatest* fixpoints, respectively. The difference between just fixpoints on the one hand and least or greatest ones on the other hand is sometimes blurred in the literature. For instance, the $\Delta$-coinduction rules of [24] are rather unfolding rules because their soundness relies on the property that behavioral equivalence is a fixpoint, not on the fact that it is a greatest fixpoint. Of course, if the axioms for a predicate $r$ have a unique solution, then the difference vanishes. A unique solution is obtained if the axioms define $r$ in non-recursively, i.e. $r(x)$ is equivalent to a first-order formula over defined functions and predicates specified on a lower level. In this case, (co)induction on $r$ coincides with unfolding of $r$.

The first-order versions of fixpoint induction and coinduction are **expansion** or **analytic rules**, i.e. their forward application establishes backward implications from the succedent to the antecedent of a rule. However, generalizations resp. specializations of the antecedent turn fixpoint induction and coinduction into (second-order) equivalence transformations. Fixpoint induction is mainly used for proving reachability conditions, pre/post conditions and invariants. The classical Hoare and subgoal induction rules are special cases of fixpoint induction. Coinduction is used to prove behavioral equations, including bisimilarities determined by transition systems, fairness conditions, transition invariants and *non-inductive* properties of infinite objects whose validity for an object $a$ cannot be concluded from its validity for all finite approximations of $a$.

While it is mostly rather obvious when and where to apply unfolding, constructor rules or (co)induction, there are other rules whose reasonable use needs more anticipation concerning the goal the proof heads for. We propose a coarse classification into **simplifications** (equivalence transformations), the already mentioned expansion rules and their converse: **contraction** or **synthetic rules** whose forward application establishes forward implications from the antecedent to the succedent of a rule. Following the design principles *separation of concerns*, *divide and conquer* and *compositionality*, the applicability of a rule should depend only on that part of a formula, which provides the actual redex of the rule. The global structure of a *natural* verifier meeting those principles is sketched in Section 2.

Simplifications have the great advantage of being applicable to any matching subformula. So theorem provers often perform them automatically. For instance, *Expander* [14] contains an extendable partial evaluator that rewrites functional and relational expressions by calling built-in ML function definitions and applies the above-mentioned constructor rules as well as certain Boolean simplifications on the level of formulas. Moreover, a function $f$ can be labelled in a way that prompts the evaluator to apply a specific a program transformation to $f$. Such features speed up the proof process considerably and allow the user to concentrate on crucial steps that require her or his anticipating intervention.

Proper expansions or contractions may be needed for applying lemmas and induction hypotheses that come as **Gentzen clauses** $\varphi \Leftarrow \psi$ where $\varphi$ and $\psi$ are negation- and implication-free disjunctions of conjunctions of atoms. In particular, **inductive contraction/expansion** provides an alternative to fixpoint induction. Fixpoint induction is "efficient" because if applied to a formula expressing a desired property of a function or predicate $r$, it *eliminates* $r$. However, the desired property must be given in a special form, while the generality of contraction/expansion redices sometimes allow more direct proofs with inductive contraction/expansion. On the other hand, since they perform induction steps, these rules are never applied at the beginning of a proof, but somewhere in the "middle", which is rarely accomplished without a fair amount of user interaction. Rippling [3] and other strategies that guide a proof towards induction hypotheses are useful, but far from complete. However, proofs involving inductive expansion steps justify many program synthesis rules (cf. Section 6 and [12, 13, 18]). Besides the above-mentioned automatic simplifications, Expander [14] implements unfolding as well as lemma and inductive expansion/contraction, but is so far restricted to the proof of Gentzen clauses.

User-controlled proof guidance is inevitable when non-trivial requirements to a formally specified data type shall be verified. Consequently, the steps a verification tool performs and the subgoals it produces should be natural, i.e. comprehensible by human beings. This is hardly achieved if a conjecture must be subjected to extensive normalizations, Skolemization, etc., before the tool can handle it. Moreover, it should be poosible to switch off built-in simplifications when they would change a formula in a way that redices for crucial expansion or contraction steps disappear before the steps are carried out. The schema of a natural-proof environment is outlined in Section 2.

Section 3 recapitulates the syntax and semantics of swinging types and thus provides a uniform basis for the presentation of simplification rules in Section 4, expansion and contraction rules in Section 5 and (co)induction rules in Section 6. Complete proof samples using these rules can be found in [12–14, 18–20].

## 2 The global structure of a natural verifier

Instead of maintaining complicated proof trees, tableaus, etc., we propose a simple linear proof structure. Given a formula $\varphi$ to be proved or solved, we proceed "backwards" and construct a derivation sequence like

$$\varphi = \varphi_1 \quad \vdash \quad \varphi_2 \quad \vdash \quad \dots \quad \vdash \quad \varphi_n = \psi.$$

We assume that the formulas are interpreted in a two-valued logic such that each proof step represents an inverse implication with respect to some given semantics, such as a Herbrand model:

$$\varphi = \varphi_1 \quad \Leftarrow \quad \varphi_2 \quad \Leftarrow \quad \dots \quad \Leftarrow \quad \varphi_n = \psi.$$

There are three cases as to what $\psi$ may represent. If $\psi$ is an arbitrary formula, then we can conclude that $\varphi$ holds under the "constraint" $\psi$. If $\psi$ is a conjunction of equations

$$\exists\ Z : (x_1 \equiv t_1 \wedge \cdots \wedge x_n \equiv t_n)$$

such that $x_1, \dots, x_n$ are different variables and for all $1 \leq i, j \leq n$, $x_i \notin var(t_j)$ and $var(t_i) \subseteq Z$, then $\psi$ represents a substitution and thus a set of solutions of $\varphi$. If $\psi$ is a tautology, then $\varphi$ has been proved.

The proof is natural because we do not require a special form of $\varphi_1, \dots, \varphi_n$. The rules we present in this paper may be applied to any first- or second-order[1] formula built up from functions and predicates of the underlying type and usual predicate-logic operators. In contrast to most automated theorem provers, proof

[1] Predicate variables are confined to existentially quantified ones.

goals need not be normalized, neither in the beginning nor in the course of a proof. There is one exception: negation signs must be shifted inwards and the resulting negative literals $\neg r(t)$ be replaced by positive ones $\overline{r}(t)$ where $\overline{r}$ is interpreted as the complement of $r$. This normalization does not require additional space (not a single symbol is added the formula) and is needed only at the beginning of a proof because none of our rules introduces negation signs. Many proofs even go without the initial normalization, simply because most verification conditions are "positive-conditional".

However, there are implicit negations in a formula whenever it involves implication signs. Since implication is *the* natural logic operator and its elimination would reduce the comprehensibility of a proof considerably, we keep implication signs and take them into account in the following way when applying a rule.

Each rule is an **expansion**, a **contraction** or a **simplification**, denoted by:

$$\frac{\varphi}{\psi}\Uparrow, \qquad \frac{\varphi}{\psi}\Downarrow, \qquad \frac{\varphi}{\psi}\Updownarrow,$$

respectively. The arrow indicates the direction of consequence with respect to the interpretation of $\varphi$ and $\psi$: in the case of an expansion, $\psi$ implies $\varphi$; in the case of a contraction, $\varphi$ implies $\psi$; in the case of a simplification, $\varphi$ and $\psi$ are equivalent. The category a rule belongs to determines the positions of possible redices $\varphi$ within a greater formula $C[\varphi]$. To make this precise we define the *polarity* of the subformula at position $w$ of $\varphi$ in terms of a Boolean function whose definition also provides part of an algorithm for deciding whether or not a rule is applicable at $w$: for all formulas $\varphi$ and $w \in 2^*$, $polarity(\psi, w) = acc(\psi, w, true)$ where for all $b \in \{true, false\}$,

$$\begin{aligned}
acc(\varphi \Rightarrow \psi, 0w, b) &= acc(\varphi, w, not(b)),\\
acc(\varphi \Rightarrow \psi, 1w, b) &= acc(\psi, w, b),\\
acc(\varphi \overset{\wedge}{\vee} \psi, 0w, b) &= acc(\varphi, w, b),\\
acc(\varphi \overset{\wedge}{\vee} \psi, 1w, b) &= acc(\psi, w, b),\\
acc(\overset{\forall}{\exists} x\varphi), 0w, b) &= acc(\varphi, w, b),\\
acc(\varphi, \varepsilon, b) &= b.
\end{aligned}$$

An expansion is applicable to $\varphi$ at position $w$ if $polarity(\varphi, w) = true$. A contraction is applicable to $\varphi$ at $w$ if $polarity(\varphi, w) = false$. Hence a simplification is always applicable. To make it short:

$$\begin{array}{lr}
\frac{\varphi}{\psi}\Uparrow \wedge\, polarity(C[\varphi], position(\varphi)) = true & \text{implies } \frac{C[\varphi]}{C[\psi]}\Uparrow\\
\frac{\varphi}{\psi}\Downarrow \wedge\, polarity(C[\varphi], position(\varphi)) = false & \text{implies } \frac{C[\varphi]}{C[\psi]}\Uparrow\\
\frac{\varphi}{\psi}\Updownarrow & \text{implies } \frac{C[\varphi]}{C[\psi]}\Updownarrow
\end{array}$$

## 3 Swinging types

We recapitulate the syntax and semantics of swinging types. Lack of space prevents us from commenting on any details here. For these as well as for motivations and examples the reader is referred to [17, 21].

**Definition 3.1 (swinging signature)** A **basic Horn signature** $\Sigma = (S, F, P)$ consists of a set $S$ of **sorts** including implicitly the "unit sort" 1, an $S^+$-sorted set $F$ of **function symbols** and an $S^+$-sorted set $P$ of **static predicates** such that $F$ splits into sets $CO$ of $s$**-constructors** $f : w \to s$, $s \in S$, and a set $DF$ of **defined functions** and for all $s \in S$, $P$ implicitly includes the **structural** $s$**-equality** $\equiv_s$: $ss$ and the **behavioral** $s$**-equality** $\sim_s$: $ss$, which are supposed to denote identical relations.

The set of **swinging signatures** is defined inductively as follows:

- Basic Horn signatures are swinging.
- Let $vis\Sigma = (visS, visF, visP)$ be a swinging signature, $hidS$ be a set of sorts, $S = visS \cup hidS$ and $hidF$ and $hidP$ be $S^+$-sorted sets of function symbols and predicates, respectively, such that
  - $hidF$ splits into sets $CO$ of $s$**-constructors** $f : w \to s$, $s \in hidS$, and a set $DF$ of **defined functions**,
  - for all $s \in hidS$, $CO$ includes a set of $s$**-object constructors** $f : w \to s$,
  - $hidP$ splits into sets $\mu P$ and $\mu P'$ of $\mu$**-predicates** and $\nu P$ of $\nu$**-predicates**,
  - $\mu P$ is the union of a set $statP$ of **static predicates** and a set $dynP$ of **dynamic predicates**,
  - for all $s \in hidS$, $DF$ includes a set of $s$**-destructors** $f : sw \to s'$,
  - for all $s \in hidS$, $statP$ includes a set of $s$**-destructors** $r : sw$,
  - for all $s \in hidS$, $dynP$ includes a set of $s$**-transition predicates** $\delta$ : $sws'$,
  - for all $s \in hidS$, $dynP$ implicitly includes the **structural** $s$**-equality** $\equiv_s$: $ss$,
  - for all $s \in hidS$, $\nu P$ implicitly includes the **behavioral** $s$**-equality** $\sim_s$: $ss$.

  Then $\Sigma = (S, visF \cup hidF, visP \cup hidP)$ is a swinging signature, $vis\Sigma$ is the **visible subsignature of** $\Sigma$ and the symbols of $vis\Sigma$ are called **visible**. The other symbols of $\Sigma$ are called **hidden**. Visible structural equalities are both static and dynamic.

A predicate is **logical** if it is not a structural equality. $s$-destructors and $s$-transition predicates are called $s$**-observers**. ❏

$\Sigma$-terms consisting of (object) constructors and variables are called **(object)** $\Sigma$**-normal forms**. **Ground terms** are terms without variables. Given an $S$-sorted set $X$ of variables, $T_\Sigma(X)$ and $NF_\Sigma(X)$ denote the $S$-sorted sets of $\Sigma$-terms resp.

$\Sigma$-normal forms whose variables are taken fom $X$. If $X$ is empty, we write $T_\Sigma$ and $NF_\Sigma$ instead of $T_\Sigma(X)$ resp. $NF_\Sigma(X)$.

**Definition 3.2 (formulas)** A **simple goal** is a finite conjunction of $\Sigma$-atoms. Given a finite subset $Y$ of $X$ and simple goals $G$ and $H$, $\exists YG$ is an **existential goal** and $\forall Y(G \Rightarrow H)$ is a **universal goal**. The empty conjunction (resp. disjunction) is denoted by $TRUE$ (resp. $FALSE$).

Let $G$ be a simple goal, $r$ be a logical predicate and $f$ be a defined function. Given a finite conjunction $\varphi$ of universal goals, $r(t) \Leftarrow \varphi$ resp. $f(t) \equiv u \Leftarrow \varphi$ is a **Horn clause for** $r$ resp. $f$. A Horn clause $p \Leftarrow \varphi$ is **basic** if all universal goals of $\varphi$ are unconditional. Given a finite disjunction $\varphi$ of existential goals, $r(t) \Rightarrow (G \Rightarrow \varphi)$ is a **co-Horn clause for** $r$.

A (first-order) formula $\varphi$ is **poly-modal** if $\varphi$ is equivalent to a formula built up as follows:

- Each atom $r(t)$ with $r \in statP \cup \nu P$ is poly-modal.
- If $\varphi$ and $\psi$ are poly-modal, then $\neg\varphi$ and $\varphi \wedge \psi$ are poly-modal.
- If $\varphi$ is poly-modal, then for all atoms $\delta(t,x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$, $\exists x(\delta(t,x) \wedge \varphi)$ is poly-modal.

A formula $\varphi$ is **weakly modal with output out**$(\varphi) \subseteq X$ if $\varphi$ is equivalent to a formula built up as follows:

- Each poly-modal formula is weakly modal with output $\emptyset$.
- Each atom $\delta(t,x)$ with $\delta \in dynP$ and $x \in X \setminus var(t)$ is weakly modal with output $\{x\}$.
- If $\varphi$ and $\psi$ are weakly modal with outputs $Y$ resp. $Z$ such that $Z \cap freevar(\varphi) = \emptyset$, then $\varphi \wedge \psi$ is weakly modal with output $Y \cup Z$.
- If $\varphi$ is weakly modal with output $Y$, then for all $x \in X$, $\exists x\varphi$ is weakly modal with output $Y \setminus \{x\}$. ❑

**Definition 3.3 (algebraic swinging type)** Given a basic Horn signature $\Sigma$ and a set $AX$ of basic Horn clauses for defined functions and logical predicates of $\Sigma$, $SP = (\Sigma, AX)$ is a **basic Horn specification** if $AX = AX' \cup EQ_\Sigma$[2] such that for all $(f(t) \equiv u \Leftarrow \varphi)$ and $(r(t) \Leftarrow \varphi) \in AX'$, $f \in DF$, $r$ is logical,

(a) $t$ is a tuple of normal forms and $var(u) \subseteq var(t) \cup freevar(\varphi)$.

$AX$ is the set of **axioms of** $SP$.

The set of **algebraic swinging types** is defined inductively as follows:

- Each basic Horn specification is an algebraic swinging type.

[2] $EQ_\Sigma$ denotes the set of congruence axioms for $\Sigma$ (cf. [17], Section 2).

- Let $\Sigma$ be a swinging signature that is not basic Horn, $vis\Sigma$ be the visible subsignature of $\Sigma$, $visSP = (vis\Sigma, visAX)$ be an algebraic swinging type and $AX = visAX \cup \mu AX' \cup \nu AX \cup \mu AX''$ (the **axioms of** $SP$) where
  - $\mu AX' = \mu AX \cup (EQ_\Sigma \setminus EQ_{vis\Sigma})$ consists of basic Horn clauses,
  - $\nu AX$ consists of co-Horn clauses,
  - $\mu AX''$ consists of Horn clauses,
  - for all $f \in DF$, $r \in statP$, $\delta \in dynP$ and $(f(t) \equiv u \Leftarrow \varphi), (r(t) \Leftarrow \varphi), (\delta(t,u) \Leftarrow \varphi) \in \mu AX$, (a) holds true,

    (b) $\varphi$ is weakly modal, $var(t) \cap out(\varphi) = \emptyset$,
    (c) $\varphi$ does not include predicates of $\nu P$ or $\mu P'$,

  - for all $(r(t) \Rightarrow (\varphi \Rightarrow \psi)) \in \nu AX$, $r \in \nu P$, (a) and (c) hold true, $\varphi \Rightarrow \psi$ is poly-modal and $\psi$ does not include predicates of $\mu P'$,
  - for all $(r(t) \Leftarrow \varphi) \in \mu AX''$ and universal goals $\forall Y(G \Rightarrow H)$ of $\varphi, r \in \mu P'$, $G$ does not include predicates of $\mu P'$,
  - the axioms for behavioral equalities are called **behavior axioms** and read as follows:

$$
\boxed{
\begin{array}{l}
x \sim_s y \;\Rightarrow\; f(x,z) \sim_{s'} f(y,z) \\
\qquad\qquad \text{for all } s \in hidS \text{ and destructors } f : sw \to s', \\
x \sim_s y \;\Rightarrow\; r(x,z) \Leftrightarrow r(y,z) \\
\qquad\qquad \text{for all } s \in hidS \text{ and destructors } r : sw, \\
x \sim_s y \;\Rightarrow\; (\delta(x,z,x') \;\Rightarrow\; \exists\, y'(\delta(y,z,y') \wedge x' \sim_{s'} y')), \\
x \sim_s y \;\Rightarrow\; (\delta(y,z,y') \;\Rightarrow\; \exists\, x'(\delta(x,z,x') \wedge x' \sim_{s'} y')) \\
\qquad\qquad \text{for all } s \in hidS \text{ and transition predicates } \delta : sws', \\
x \sim_s y \;\Rightarrow\; (x \equiv_s f(x') \;\Rightarrow\; \exists\, y'(y \equiv_s f(y') \wedge x' \sim_{s'} y')), \\
x \sim_s y \;\Rightarrow\; (y \equiv_s f(y') \;\Rightarrow\; \exists\, x'(x \equiv_s f(x') \wedge x' \sim_{s'} y')) \\
\qquad\qquad \text{for all } s \in hidS \text{ and object constructors } f : w \to s, \\
\kappa_i(x) \sim_s \kappa_i(y) \;\Rightarrow\; x \sim_{s_i} y, \\
\kappa_i(x) \sim_s \kappa_j(y) \;\Rightarrow\; FALSE \\
\qquad\qquad \text{for all } s = \amalg_{i \in I} s_i \in hidS \text{ and } i,j \in I \text{ such that } i \neq j, \\
(x_1,\dots,x_n) \sim_s (y_1,\dots,y_n) \;\Rightarrow\; x_i \sim_{s_i} y_i \\
\qquad\qquad \text{for all } s = s_1 \times \cdots \times s_n \in hidS \text{ and } 1 \leq i \leq n.
\end{array}
}
$$

  Then $SP = (\Sigma, AX)$ is an algebraic swinging type and $visSP$ is the **visible subspecification of** $SP$. ❏

**Definition 3.4 ($SP$-models)** Let $SP = (\Sigma, AX)$ be a swinging type whose components are named as in Def. 3.3. A $\Sigma$-structure is a **Herbrand structure** if for all $s \in S$, $A_s = T_{\Sigma,s}$, and for all $f : w \to s \in F$ and $t \in T_\Sigma(X)_w$, $f^A(t) = f(t)$. $A$ is **reachable** if for all $a \in A$ there is $t \in T_\Sigma$ with $t^A = a$. Given $\approx_s$: $ss \in P$ for all $s \in S$, $A$ is a **structure with $\approx$-equality** if for all $s \in S$, $\approx_s^A = \{(a,a) \mid a \in A_s\}$. $A$ is an $SP$**-model** if $A$ satisfies $AX \cup EQ_\Sigma$. **Mod(SP)**

denotes the class of all $SP$-models. $\mathbf{Mod}_{\equiv}\mathbf{(SP)}$ is the class of $SP$-models with $\equiv$-equality. $\mathbf{Mod}_{bcr}\mathbf{(SP)}$ is the class of reachable $SP$-models $A$ such that $\sim^A$ is a weak $\Sigma$-congruence. $\mathbf{Mod}_{\mu\nu}\mathbf{(SP)}$ is the class of $SP$-models that interpret all $\mu$-predicates as the least relations and all $\nu$-predicates as the greatest relations satisfying $AX$.

The **Herbrand** $SP$**-model**, $Her(SP)$, is the Herbrand $\Sigma$-structure that is defined as follows: If $SP$ is basic Horn, then for all predicates $r : w \in \Sigma$, $r^{Her(SP)}$ is the least relation on $T_\Sigma$ that satisfies $AX \cup EQ_\Sigma$. If $SP$ is not basic Horn, then let $visSP = (vis\Sigma, visAX)$ be the visible subspecification of $SP$, $SP' = (\Sigma', AX')$ be the union of $(\mu P', \mu AX')$ and the *basic Horn translation* of $visSP$ (cf. [21], Def. 2.5),

- for all predicates $r : w \in \Sigma'$, $r^{Her(SP)}$ is the least relation on $T_\Sigma$ that satisfies $AX'$,
- for all $r \in \nu P$, $r^{Her(SP)} =_{def} r^{gfp(\Phi)}$ where $\Phi$ is the $\nu AX$-consequence operator on $Her(SP)|_{\Sigma'}$ (cf. [17], Def. 4.3),
- for all $r \in \mu P'$, $r^{Her(SP)} =_{def} r^{lfp(\Psi)}$ where $\Psi$ is the $\mu AX''$-consequence operator on $Her(SP)|_{\Sigma' \cup \nu P}$.

A first-order formula satisfied by $Her(SP)$ is an **inductive theorem of** $SP$. Given two predicates $r : w, \overline{r} : w \in \Sigma$, $\overline{r}$ is the **complement** of $r$ if $\overline{r}^{Her(SP)} = T_{\Sigma,w} \setminus r^{Her(SP)}$.[3]

The **structural** $SP$**-equivalence**, $\equiv_{SP}$, and the **behavioral** $SP$**-equivalence**, $\sim_{SP}$, are the interpretations of $\equiv$ and $\sim$, respectively, in the $SP$-Herbrand model. A sort $s \in \Sigma$ is **strongly visible** if $\equiv_{SP,s}$ coincides with $\sim_{SP,s}$.

$u \in NF_\Sigma$ is a **normal form of** $t \in T_\Sigma$ if $t$ and $u$ are structurally $SP$-equivalent. $SP$ is **complete** if each ground $\Sigma$-term has a normal form. $SP$ is **object constructor complete** if each ground $\Sigma$-term $t$ has a behaviorally (!) $SP$-equivalent object normal form. $SP$ is **consistent** if structurally $SP$-equivalent ground normal forms are equal. $SP$ is **functional** if $SP$ is complete and consistent. Then for all $t \in T_\Sigma$, $\mathbf{nf(t)}$ denotes the unique normal form of $t$.

$SP$ is **behaviorally consistent** if $\sim_{SP}$ is a weak $\Sigma$-congruence, i.e. $\sim_{SP}$ is compatible with $\Sigma \setminus dynP$ and zigzag compatible with $dynP$ (cf. [17], Def. 3.1). $SP$ is **continuous** if the above consequence operators $\Phi$ and $\Psi$ are downward resp. upward continuous. Provided that $SP$ is behaviorally consistent, the **final** $SP$**-model**, $Fin(SP)$, is the $\Sigma$-quotient of $Her(SP)$ by $\sim_{SP}$. ❑

**Theorem 3.5** ([17], Thms. 3.8, 5.1) *Let $SP = (\Sigma, AX)$ be a behaviorally consistent specification. $Fin(SP)$ is in $Mod_{\mu\nu}(SP)$ and final in $Mod_{bcr}(SP)$. Moreover, for all poly-modal formulas $\varphi(x)$ and $t, t' \in T_\Sigma$,*

$$\begin{aligned} Her(SP) \models \varphi(t) \wedge t \sim_{SP} t' \quad &\Longrightarrow \quad Her(SP) \models \varphi(t'), \\ Her(SP) \models \varphi(x) \quad &\Longleftrightarrow \quad Fin(SP) \models \varphi(x) \quad ❑ \end{aligned}$$

[3] We assume that $\overline{r}$ always denotes the complement of $r$.

Weak criteria that ensure functionality, continuity or behavioral consistency are [18], Korollar 6.1.5, [17], Thm. 5.15, and [21], Thm. 6.4, respectively. Certain inference rules are sound only if $SP$ is functional resp. behaviorally consistent (see below).

**Definition 3.6 (cosignature)** Let $vis\Sigma = (visS, F, P)$ be a swinging signature. The elements of $visS$ are called **visible sorts**. Let $hidS$ be a set of sorts, called **hidden sorts**, $S = visS \cup hidS$ and $des$ be a set of function symbols $d : sw \to \amalg_{i\in I} w_i$, called **destructors**, such that $s \in hidS$, $w \in visS^*$ and for all $i \in I$, $w_i \in S^+$. Then $\Delta = (vis\Sigma, hidS, des)$ is a **cosignature** and $vis\Sigma$ is the **visible subsignature of** $\Delta$. ❏

**Definition 3.7 (coalgebras)** Let $\Delta = (vis\Sigma, hidS, des)$ be a cosignature, $vis\Sigma = (visS, F, P)$, $S = visS \cup hidS$ and $C$ be a $vis\Sigma$-structure with $\sim$-equality. A $(\Delta, C)$**-coalgebra** $A$ is an $(S, F \cup des, P)$-structure such that $A|_{vis\Sigma} = C$ and for all $s, s' \in S$, the sort $s \to s'$ is interpreted as the function space $[A_s \to A_{s'}]$. $coAlg(\Delta, C)$ denotes the class of $(\Delta, C)$-coalgebras. The **final** $(\Delta, C)$**-coalgebra** $Fin(\Delta, C)$ is the final object in $coAlg(\Delta, C)$, which exists by [21], Thm. 4.1.6.❏

**Definition 3.8 (cospecification)** Let $\Delta = (vis\Sigma, hidS, des)$ be a cosignature, $vis\Sigma = (visS, visF, visP)$, $S = visS \cup hidS$ and $aux$ and $cof$ be $S^+$-sorted sets of function symbols, called **auxiliary functions** and **cofunctions**, respectively, such that for all $f : sw \to s' \in aux$, $s$ is visible, and for all $f : w \to s \in cof$, $s$ is hidden. Moreover, let $\nu P$ be an $S^+$-sorted set of $\nu$**-predicates** that includes the set $BE = \{\sim_s : ss \mid s \in hidS\}$ of **behavioral equalities** and the set $MP = \{is_s : s \mid s \in hidS\}$ of **membership predicates**. Moreover, for all $s \in hidS$, there is the **structural** $s$**-equality** $\equiv_s : ss$.

Let $C$ be a $vis\Sigma$-structure with $\sim$-equality and $>$ be a well-founded ordering on $C$. Let $s \in visS$. A subset $PAT$ of $NF_{vis\Sigma}(X)_s$ is $s$**-complete** if for all $a \in NF_{vis\Sigma,s}$ there are $t \in PAT$ and $b : X \to C$ such that $a = b^*(t)$.

Given an auxiliary function $f : sw \to s'$ and an $s$-complete set $PAT$, let $AX_f = \cup_{t\in PAT} AX_{f,t}$ be an **inductive axiomatization of** $f$, i.e. a set of basic Horn clauses (cf. Def. 3.2) such that for all $t \in PAT$, $AX_{f,t}$ has the form

$$\{f(t, x) \equiv t_1 \Leftarrow \varphi_1, \quad \dots, \quad f(t, x) \equiv t_n \Leftarrow \varphi_n\},$$

for all $1 \le i \le n$, $t_i$ and $\varphi_i$ do not contain neither $\nu$-predicates nor cofunctions, for all $1 \le i \le n$, $g \in aux$, subterms $g(u, v)$ of $t_i$ or $\varphi_i$ and $b : X \to C$, $b^*(t) > b^*(u)$ and $coAlg(\Delta, C)$ satisfies $\varphi_1 \dot\vee \dots \dot\vee \varphi_n$.

For all $s \in S$, let $AX_{\sim_s} = \{x \sim_s y \Rightarrow d(x, z) \sim d(y, z) \mid s \in hidS,\ d : sw \to s' \in des\}$.

For all $s \in hidS$, let $AX_s$ be a set of poly-modal implications $G \Rightarrow \varphi$, called $s$**-assertions**, such that $G$ is a simple goal that does not contain $\nu$-predicates, $\varphi$ is a finite disjunction of existential goals and $G \Rightarrow \varphi$ does not contain cofunctions, but a distinguished free variable $x \in X_s$.

Given $r : w \in \nu P$, let $AX_r$ be a set of co-Horn clauses $r(x) \Rightarrow (G \Rightarrow \varphi)$ such that $G$ does not contain $\nu$-predicates and $G \Rightarrow \varphi$ does not contain cofunctions.

Given a cofunction $f : w \to s$, let $AX_f = \cup_{d:s\to s'\in des} AX_{f,d}$ be a **coinductive axiomatization of** $f$, i.e. a set of basic Horn clauses such that for all $d : sv \to s' \in des$, $AX_{f,d}$ has the form

$$\{d(f(x), z) \equiv t_1 \ \Leftarrow\ \varphi_1, \qquad \ldots, \qquad d(f(x), z) \equiv t_n \ \Leftarrow\ \varphi_n\},$$

for all $1 \leq i \leq n$, $\varphi_i$ does not contain neither $\nu$-predicates nor cofunctions and $coAlg(\Delta, C)$ satisfies $\varphi_1 \dot\vee \ldots \dot\vee \varphi_n$.

Then $CSP = (\Delta, C, aux, \nu P, cof, AX_{aux\cup hidS\cup\nu P\cup cof})$ is a **cospecification**.[4] ❑

**Definition 3.9 ($CSP$-models)** Let $CSP$ be a cospecification as in Def. 3.8 and $\Sigma = vis\Sigma \cup (hidS, des \cup aux, \nu P)$. A $\Sigma$-structure $A$ with $\equiv$-equality is a $CSP$**-model** if $A$ satisfies $AX \setminus AX_{cof}$. $Mod(CSP)$ denotes the class of $CSP$-models. The **final** $CSP$**-model** $Fin(CSP)$ is the final object in $Mod(CSP)$, which exists by [21], Thm. 4.1.11(3). ❑

**Theorem 3.10** ([21], Thm. 4.1.12) *Let $CSP$ be a cospecification as in Def. 3.8, $\Sigma = vis\Sigma \cup (hidS, des \cup aux \cup cof, \nu P)$ and $SP = (\Sigma, AX)$. $Fin(CSP)$ can be extended to an $SP$-model.* ❑

**Definition 3.11 (coalgebraic swinging type)** Let $SP = (\Sigma, AX)$ be an algebraic swinging type that is not basic Horn and whose components are named as in Def. 3.3. Suppose that all $hidS$-observers are functional destructors with a single hidden argument and there are no $hidS$-object constructors. Let $des$ denote the set of $hidS$-destructors, $\Delta = (vis\Sigma, hidS, des)$, $C = Fin(visSP)$ and $CSP = (\Delta, C, aux, \nu P, cof, CAX)$ be a cospecification such that the auxiliary functions of $CSP$ are defined functions of $SP$ and the axioms for $aux$ and $\nu P$ are the same in $SP$ and $CSP$. Then $CST = (SP, aux, cof, AX_{hidS\cup cof})$ is a **coalgebraic swinging type**. $SP$ is the **algebraic part** and $CSP$ is the **cospecification of** $CST$. ❑

**Definition 3.12 (domain completion)** Let $CST$ be a coalgebraic swinging type such that $visSP$ is functional and object constructor complete (cf. Def. 3.4). We regard the elements of $Fin(CSP)$ as additional (nullary) constructors and the cofunctions of $CSP$ as additional defined functions and add the following sets of axioms for destructors and cofunctions, respectively:

$$\begin{aligned} AX_{des} = \{d(a,u) \equiv t \mid\ & d : sw \to s' \in des,\ s' \in visS,\ a \in Fin(CSP)_s, \\ & u, t \in NF_{vis\Sigma},\ t \text{ is an object normal form}, \\ & d^{Fin(CSP)}(a, [u]) = [t]\} \ \cup \\ \{d(a,u) \equiv b \mid\ & d : sw \to s' \in des,\ s' \in hidS,\ a \in Fin(CSP)_s, \\ & u \in NF_{vis\Sigma},\ d^{Fin(CSP)}(a, [u]) = b\} \end{aligned}$$

[4] For a set $M$ of symbols, $AX_M$ denotes the union of $AX_f$ over all $f \in M$.

$$\begin{aligned}AX_{cof} = \{f(t_1,\dots,t_n) \equiv b \mid f : s_1\dots s_n \to s \in cof,\ &\text{for all } 1 \le i \le n,\\ &s_i \in visS \text{ implies } t_i \in NF_{vis\Sigma,s_i} \text{ and } t'_i = [t_i],\\ &s_i \in hidS \text{ implies } t_i = t'_i \in Fin(CSP)_{s_i},\\ &f^{Fin(CSP)}(t'_1,\dots,t'_n) = b\}\end{aligned}$$

Let $\Sigma' = \Sigma \cup (\emptyset, Fin(CSP) \cup cof, \emptyset)$ and $AX' = AX \cup AX_{des\cup cof}$. Then $SP' = (\Sigma', AX')$ is an algebraic swinging type, called the **domain completion of** $CST$.

$CST$ is **cospec closed** if $CST$ has no assertions or no $hidS$-constructors or for all $s \in hidS$ and $t \in NF_{\Sigma',s}$ there is $a \in Fin(CSP)_s$ such that $t \sim_{SP'} a$. $CST$ is **functional**, **continuous** or **behaviorally consistent**, respectively, if $SP'$ is functional, continuous or behaviorally consistent, respectively. ❑

**Theorem 3.13** ([21], Thm. 4.2.5) *Let $CST$ be a cospec closed, functional, continuous and behaviorally consistent coalgebraic swinging type and $SP' = (\Sigma', AX')$ be the domain completion of $CST$ such that $visSP$ is functional and object constructor complete. Then $Fin(CSP)$ and $Fin(SP')$ are $\Sigma'$-isomorphic.* ❑

For the rest of this paper, let $SP = (\Sigma, AX)$ be an algebraic swinging type as in Def. 3.3. For each predicate or defined function $r \in \Sigma$, $AX_r$ denotes the subset of $AX$ that consists of all axioms for $r$.

## 4 Basic simplification rules

Applications of **unfolding rules** are the proof counterparts of "procedure calls":

**unfolding of a logical $\mu$-predicate $r$**

$$\frac{r(t)}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \varphi_i)} \Updownarrow$$

where $r(t_1) \Leftarrow \varphi_1, \dots, r(t_n) \Leftarrow \varphi_n$ are the axioms for $r$ and $Z_i = var(t_i, \varphi_i)$

**unfolding of a $\nu$-predicate $r$**

$$\frac{r(t)}{\bigwedge_{i=1}^{n} \forall Z_i : (t \equiv t_i \Rightarrow \varphi_i)} \Updownarrow$$

where $AX_r = \{r(t_1) \Rightarrow \varphi_1, \dots, r(t_n) \Rightarrow \varphi_n\}$ and $Z_i = var(t_i, \varphi_i)$

**unfolding of the complement $\overline{r}$ of $r \in \mu P$** [5]

$$\frac{\overline{r}(t)}{\bigwedge_{i=1}^{n} \forall Z_i : ((t \equiv t_i \wedge \varphi_i) \Rightarrow \overline{q}(u_i))} \Updownarrow$$

where $r(t_1) \Leftarrow \varphi_1 \wedge q(u_1), \dots, r(t_n) \Leftarrow \varphi_n \wedge q(u_n)$ are the axioms for $r$ and $Z_i = var(t_i, \varphi_i)$

[5] Cf. Def. 3.1. $\overline{r}$ is an implicit element of $\nu P$.

**unfolding of the complement $\overline{r}$ of $r \in \nu P$** [6]

$$\frac{\overline{r}(t)}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \varphi_i \wedge \overline{q}(u_i))} \Updownarrow$$

where $AX_r = \{r(t_1) \Rightarrow (\varphi_1 \Rightarrow q(u_1)), \ldots, r(t_n) \Rightarrow (\varphi_n \Rightarrow q(u_n))\}$ and $Z_i = var(t_i, \varphi_i)$

**unfolding of a defined function $f$**

$$\frac{\varphi(f(t))}{\bigvee_{i=1}^{n} \exists Z_i : (t \equiv t_i \wedge \varphi(u_i) \wedge \varphi_i)} \Updownarrow \quad \textit{if } SP \textit{ is functional}$$

where $AX_f = \{f(t_1) \equiv u_1 \Leftarrow \varphi_1, \ldots, f(t_n) \equiv u_n \Leftarrow \varphi_n\}$ and $Z_i = var(t_i, u_i, \varphi_i)$

**unfolding of a cofunction $f$**

$$\frac{\varphi(d(f(t), u))}{\bigvee_{i=1}^{n} (\varphi(t_i[t/x, u/z]) \wedge \varphi_i[t/x, u/z])} \Updownarrow \quad \textit{if the assumptions of Thm. 3.13 are satisfied}$$

where $\{d(f(x), z) \equiv t_1 \Leftarrow \varphi_1, \ \ldots, \ d(f(x), z) \equiv t_n \Leftarrow \varphi_n\}$ is the coinductive axiomatization of $f$ (cf. Def. 3.8)

The rules for complements are derived from the first two rules. Their soundness follows from the fact that complements of least (resp. greatest) fixpoints are greatest (resp. least) fixpoints (cf. [18], Satz 8.3.4).

Instead of generating a complete case analysis and applying all axioms for a predicate or function in parallel, one may discard all axioms that lead to unsolvable equations $t \equiv t_i$. The resulting **optimized unfolding** rule agree with the respective unfolding rule except that for all $1 \leq i \leq n$, the rule succedent does not contain the i-th summand or factor if the equation $t \equiv t_i$ is unsolvable (in the $SP$-Herbrand model). i.e. if, for instance, $t$ and $t_i$ begin with different constructors and $SP$ is functional (see the clash rule given below).

Checks for unsolvability of subformulas are integrated into the proof: $\varphi$ is unsolvable if the proof reduces $\varphi$ into $FALSE$. $\varphi_i$ is valid (in the $SP$-Herbrand model) if it turns $\varphi$ into $TRUE$. If each $\varphi_i$ in the succedent of an unfolding rule is valid or unsolvable, the proof leads to a solution resp. dissolution of the unfolded atom, say $r(x)$. If $r$ is a $\mu$-predicate, then

$$\begin{array}{rcl} r(x) & \stackrel{unfolding}{\vdash} & \bigvee_{i=1}^{n} (x \equiv t_i \wedge \varphi_i) \\ & \vdash & \bigvee_{i=1}^{k} (x \equiv t_i \wedge TRUE) \vee \bigvee_{i=k+1}^{n} (x \equiv t_i \wedge FALSE) \\ & \vdash & \bigvee_{i=1}^{k} x \equiv t_i. \end{array}$$

[6] Cf. Def. 3.1. $\overline{r}$ is an implicit element of $\mu P'$.

If $r$ is a $\nu$-predicate, then

$$\begin{array}{lll} r(x) & \stackrel{unfolding}{\vdash} & \bigwedge_{i=1}^{n}(x \equiv t_i \Rightarrow \psi_i) \\ & \vdash & \bigwedge_{i=1}^{k}(x \equiv t_i \Rightarrow \mathit{TRUE}) \wedge \bigwedge_{i=k+1}^{n}(x \equiv t_i \Rightarrow \mathit{FALSE}) \\ & \vdash & \bigwedge_{i=k+1}^{n} x \not\equiv t_i. \end{array}$$

Optimized unfolding of functions simulates the well-known **narrowing** rule, which applies axioms for functions as (input) **resolution** applies axioms for predicates. If $SP$ is a functional basic Horn specification, then narrowing/resolution is solution complete, i.e. for each simple goal $\varphi$ and $\sigma : X \to NF_\Sigma$ such that $Her(SP)$ satisfies $\varphi\sigma$, $\sigma$ can be derived from $\varphi$ by narrowing and resolution (cf., e.g., [11], Chapter 8; [13], Cor. 7.3; [15], Section 8). While narrowing and resolution process substitutions and formulas separately from each other, unfolding represents substitutions as subformulas of the form

$$\exists\, Z : (x_1 \equiv t_1 \wedge \cdots \wedge x_n \equiv t_n). \tag{1}$$

(1) stands for the substitution $t_1/x_1, \ldots, t_n/x_n$. Hence the solution completeness of narrowing/resolution is passed to optimized unfolding, at least if $SP$ is basic Horn and functional.

Unfolding rules are applied to defined functions or logical predicates and supposed to reduce the number of occurrences of these symbols. The following simplifications do the same with constructors, equalities or inequalities.

**elimination of a constructor** $f$

$$\frac{f(t_1,\ldots,t_n) \equiv f(u_1,\ldots,u_n)}{t_1 \equiv u_1 \wedge \cdots \wedge t_n \equiv u_n} \Updownarrow \qquad \frac{f(t_1,\ldots,t_n) \not\equiv f(u_1,\ldots,u_n)}{t_1 \not\equiv u_1 \vee \cdots \vee t_n \not\equiv u_n} \Updownarrow \qquad \textit{if } SP \textit{ is functional}$$

**clash of two different constructors** $f, g$

$$\frac{f(t) \equiv g(u)}{\mathit{FALSE}} \Updownarrow \qquad \frac{f(t) \not\equiv g(u)}{\mathit{TRUE}} \Updownarrow \qquad \textit{if } SP \textit{ is functional}$$

Analogous rules for object constructors $f$ and $g$ and behavioral instead of structural (in)equalities are sound if $SP$ is functional and behaviorally consistent.

**variable elimination**

$$\frac{\exists x : (x \equiv t \wedge \varphi(x))}{\varphi(t)} \Updownarrow \qquad \textit{if } x \in X \setminus var(t)$$

$$\frac{\forall x : ((x \equiv t \wedge \varphi(x)) \Rightarrow \psi(x))}{\varphi(t) \Rightarrow \psi(t)} \Updownarrow \qquad \textit{if } x \in X \setminus var(t)$$

The following rule that removes or reduces Gentzen clauses (cf. Section 1) is an essential feature of Expander [14]:

**subsumption**

$$\frac{\bigvee_{i=1}^m \varphi_i \;\Rightarrow\; \bigvee_{i=1}^n \exists X_i\psi_i}{TRUE} \Updownarrow \qquad \frac{\bigvee_{i=1}^m \varphi_i \;\vee\; \bigvee_{i=1}^n \exists X_i\psi_i}{\bigvee_{i=1}^n \exists X_i\psi_i} \Updownarrow$$

if $\forall\ 1 \leq i \leq m\ \exists\ 1 \leq k \leq n,\ \sigma : X_k \to T_\Sigma(X),\ \psi : \varphi_i = (\psi_k\sigma \wedge \psi)$

Further useful simplifications, *logical* ones, which remove or distribute logical operators, and *non-logical* ones, which apply inductive theorems of $SP$, may be useful. The latter are often derived from axioms and thus do not lead to proof results that would not obtained without employing the theorems. *Logical* theorems need not be applied for obtaining redices for the above rules, which work on the term or atom level of a formula and thus do not require a particular distribution of logical operators.

## 5 Basic expansions and contractions

Variable eliminations are simplifications. Each of the following more general term replacement rules is either a contraction or an expansion. The actual formula context determines the direction in which it is used (cf. Section 2).

**equation elimination/introduction**

$$\frac{t \equiv u \wedge \varphi(t)}{\varphi(u)} \Downarrow \qquad \frac{(t \equiv u \wedge \varphi(t)) \;\Rightarrow\; \psi(t)}{\varphi(u) \;\Rightarrow\; \psi(u)} \Uparrow$$

$$\frac{t \sim u \wedge \varphi(t)}{\varphi(u)} \Downarrow \qquad \frac{(t \sim u \wedge \varphi(t)) \;\Rightarrow\; \psi(t)}{\varphi(u) \;\Rightarrow\; \psi(u)} \Uparrow \qquad \textit{if } \varphi \textit{ and } \psi \textit{ are poly-modal}$$

Quantor rules are also used in both directions:

**quantor elimination/introduction**

$$\frac{\exists x : \varphi(x)}{\varphi(t)} \Uparrow \qquad \frac{\forall x : \varphi(x)}{\varphi(t)} \Downarrow$$

[13, 15] deal with proofs of Gentzen clauses (cf. Section 1). If a Gentzen clause $F$ of the form[7]

$$\exists X_0\varphi \;\;\Rightarrow\;\; \bigvee_{i=1}^{k} \exists X_i u_i \vee \bigvee_{i=k+1}^{m} \exists X_i u \equiv u_i \vee \bigvee_{i=m+1}^{n} \exists X_i v \sim u_i$$

is applied from left to right (contraction) or from right to left (expansion) to the premise resp. conclusion of a Gentzen clause, the result is again a Gentzen clause.

[7] For all $1 \leq i \leq k$, $u_i$ is a *logical* atom, i.e. starts with a logical predicate, while for all $k < i \leq n$, $u_i$ is a term.

**lemma contraction with $F$ = forward application of $F$**

$$\frac{\exists X_0(\varphi\sigma \wedge \psi(u\sigma) \wedge \delta(v\sigma) \wedge \mathit{TRUE})}{\bigvee_{i=1}^{k} \exists X_i u_i\sigma \vee \bigvee_{i=k+1}^{m} X_i\psi(u_i\sigma) \vee \bigvee_{i=m+1}^{n} X_i\delta(u_i\sigma)} \Downarrow \ \textit{if } \delta \textit{ is poly-modal}$$

where for all $1 \leq i \leq n$, $(X_i \cup Y) \cap \mathit{dom}(\sigma) = \emptyset$

**lemma expansion with $F$ = backward application of $F$**

$$\frac{\bigvee_{i=1}^{k} \exists X_i(\psi_i \wedge t_i) \ \vee \ \bigvee_{i=k+1}^{n} \exists X_i\psi_i(t_i)}{\exists X_0 \uplus \cdots \uplus X_n\left(\begin{array}{c}\varphi\sigma \wedge \bigwedge_{i=1}^{k} \psi_i\sigma \wedge \bigwedge_{i=k+1}^{n} \psi_i(u)\sigma \wedge \\ \bigwedge_{i=m+1}^{n} \psi_i(v)\sigma \wedge \bigwedge_{x\in V} x \equiv x\sigma \wedge \mathit{TRUE}\end{array}\right)} \Uparrow \ \begin{array}{l}\textit{if for all} \\ m < i \leq n, \\ \psi_i \textit{ is poly-modal}\end{array}$$

where $V$ is the set of free variables of the premise and for all $1 \leq i \leq n$, $t_i\sigma = u_i\sigma$ and $(X_i \cup Z) \cap \mathit{dom}(\sigma) = \emptyset = X_i \cap \mathit{var}(t_i) \cap \mathit{var}(\psi_i)$

$F$ may be an assertion (cf. Def. 3.8), an axiom of a parameter specification or a previously proved theorem. In the inductive versions of the rules given in the next section, $F$ is an induction hypothesis and the above occurrences of *TRUE* are replaced by *descent conditions*. Due to their generality the rules look complicated. In many cases, however, the conclusion of $F$ consists either of logical atoms or of structural equations or of behavioral equations ($k = m = n$ or $k = 0 \wedge m = n$ or $k = m = 0$). Anyway, lemma expansion/contraction should not be used in a trial-and-error manner, but as carefully as lemmas are employed in classical mathematical proofs. Not the redex selection, but only the actual rule application should be automated here. The latter is what a machine can do much faster and more reliably than a human being and one of the reasons why these rules have been implemented in Expander [14].

## 6 Induction and coinduction rules

Let $\rho : X \to X$ be a renaming of variables, $F$ be a Gentzen clause as above, $\boldsymbol{x}$ be a list of the free variables of $F$ and $\gg$: $\mathit{sort}(\boldsymbol{x})^2$ be an *induction ordering*, i.e. a binary predicate whose interpretation in the Herbrand $SP$-model is well-founded. The rules of **inductive contraction/expansion with** $F$ agree with lemma contraction/expansion with $F$ except that $F$ is used as an induction hypothesis, i.e. agrees with the formula to be proved, and *TRUE* is replaced by the *descent condition* $\boldsymbol{x} \gg \boldsymbol{x}\rho\sigma$. Since $\gg$ is left open, inductive contraction/expansion covers all instances of *Noetherian* or *structural induction* [4], such as those employed by the Boyer-Moore prover [2], INKA [8], CLAM [3], the Larch Prover [6] or Isabelle [23]. In contrast to these realizations of structural induction, inductive contraction/expansion allows us to introduce descent conditions involving $\gg$, but postpone the definition of $\gg$ until the point where the descent conditions must be removed in order to finish the proof. The soundness of inductive contraction/expansion is shown in [13], Section 5.4, and [15], Sections 3 and 4.

The general idea of *deductive program synthesis* [5, 9, 22] is to derive axioms (= programs) for a function or predicate $f$ in the course of the proof of a correctness condition on $f$. For instance, one may want to derive axioms for a function $f$ such that, for given functions $g$ and $h$, $f(x) \equiv g(h(x), x)$ holds true. By unfolding $g$ and $h$, the equation is turned into a disjunction of the form

$$\bigvee_{i=1}^{n} (x \equiv t_i \wedge f(x) \equiv u_i(g,h) \wedge \varphi_i(g,h)). \tag{1}$$

If the occurrences of $g$ and $h$ in $u_i$ and $\varphi_i$ are confined to subterms of the form $g(h(v), v)$, a **folding** step, i.e. the replacement of $g(h(v), v)$ by $f(v)$, leads to a disjunction of the form

$$\bigvee_{i=1}^{n} (x \equiv t_i \wedge f(x) \equiv v_i(f) \wedge \psi_i(f)). \tag{2}$$

In fact, (2) provides axioms for $f$, namely

$$\{f(t_1) \equiv v_1(f) \Leftarrow \psi_1(f), \dots, f(t_n) \equiv v_n(f) \Leftarrow \psi_n(f),$$

which allow us to unfold $f$ in (2):

$$\bigvee_{i=1}^{n} (x \equiv t_i \wedge \psi_i). \tag{3}$$

Provided that (3) is a complete case analysis, i.e. reduces to *TRUE* (by applying lemma expansion with (3)), the proof is complete, and we have both derived axioms for $f$ and shown that they satisfy the correctness condition $f(x) \equiv g(h(x), x)$. The example reveals that folding is a special case of inductive expansion. For more details and examples of such program derivations, see [12, 13, 18].

Structural induction is also called *explicit induction* because induction hypotheses are applied explicitly. This does not apply to **fixpoint induction** whose soundness is entailed by the fixpoint property of *inductive domains* like the least relations of a Herbrand model (cf. Def. 3.4).

**fixpoint induction on $\mu$-predicates** $r_1 : w_1, \dots, r_n : w_n$

$$\frac{\bigwedge_{i=1}^{n}(r_i(x) \Rightarrow \psi_i(x))}{\exists\{q_i : w_i\}_{i=1}^{n} : \bigwedge_{i=1}^{n}(q_i(x) \Rightarrow \psi_i(x) \ \wedge \ \bigwedge_{\varphi \in AX_{r_i}} \varphi[q_j/r_j \mid 1 \leq j \leq n])} \Updownarrow$$

The rule leads to a successful proof only if $q_i$ is instantiated eventually such that the rule succedent becomes true. This happens only if $q_i$ *generalizes* $\psi_i$, but *specializes* $r_i$, i.e. $r_i(x) \Rightarrow q_i(x) \Rightarrow \psi_i(x)$. Hence $q_i$ is a conjunction of $\psi_i$ and further consequences of $r_i$.

*Flattening* transforms a formula $\varphi$ into an equivalent formula $flat(\varphi)$ where defined functions occur only at the leftmost position of a structural equation (cf. [17], Section 4). In a second step, all occurrences of a defined function $f$ are removed by replacing an equation $f(t) \equiv u$ with the atom $r_f(t,u)$ where $r_f : ws$ is a new logical predicate called the *graph* of $f$. If both transformations are applied to all axioms of $SP$, we obtain a specification $rel(SP)$ without defined functions that is semantically equivalent to $SP$ provided that $SP$ is functional and continuous ([17], Thm. 4.13). Hence the following rule is sound because fixpoint induction on $r_f$ is sound:

**fixpoint induction on a defined function** $f : w \to s$

$$\frac{f(x) \equiv y \Rightarrow \psi(x,y)}{\exists\, q : ws : q(x,y) \Rightarrow \psi(x,y) \ \wedge\ \bigwedge_{\varphi \in rel(AX_f)} \varphi[q/r_f]} \Updownarrow \qquad \textit{if } SP \textit{ is functional and continuous}$$

The rule can be extended easily to several functions and to a combination with fixpoint induction on $\mu$-predicates.

An **iterative program** for a function $f : I \to O$ can be presented as a set of axioms of the form:

$$\begin{array}{l} f(x) \equiv g(in(x)) \\ g(y) \equiv g(loop_1(y)) \ \Leftarrow\ \varphi_1(y) \\ \dots \\ g(y) \equiv g(loop_n(y)) \ \Leftarrow\ \varphi_n(y) \\ g(y) \equiv out(y) \ \Leftarrow\ \varphi_{n+1}(y) \end{array}$$

where $in : I \to L$, $loop_1, \dots, loop_n : L \to L$ and $g, out : L \to O$ are auxiliary functions using a *loop domain* $L$. The program is *correct* with respect to a given input-output relation $\psi : I \times O$ if $f(x) \equiv y$ implies $\psi(x,y)$. The classical methods to prove this are to find a *Hoare invariant* or a *subgoal invariant* (cf. [7, 10]). A relation $sub : L \times O$ is a subgoal invariant iff it satisfies the succedent of the following rule, which coincides with the case $q = \psi$ of fixpoint induction on $f, g$:

**subgoal induction**

$$\frac{\begin{array}{c} f(x) \equiv z \ \Rightarrow\ \psi(x,z) \\ \wedge\ g(y) \equiv z \ \Rightarrow\ sub(y,z) \end{array}}{\begin{array}{ll} & \psi(x,z) \ \Leftarrow\ sub(in(x),z) \\ \wedge & sub(y,z) \ \Leftarrow\ sub(loop_1(y),z) \wedge \varphi_1(y) \\ \dots & \\ \wedge & sub(y,z) \ \Leftarrow\ sub(loop_n(y),z) \wedge \varphi_n(y) \\ \wedge & sub(y,out(y)) \ \Leftarrow\ \varphi_{n+1}(y) \end{array}} \Uparrow$$

A relation $hoi : I \times L$ is a Hoare invariant iff it satisfies the succedent of the following rule whose soundness is a consequence of the soundness of subgoal induction where $sub(x,y) \Longleftrightarrow \forall z : (hoi(x,y) \Rightarrow \psi(x,z))$:

**Hoare induction**

$$\frac{f(x) \equiv z \;\Rightarrow\; \psi(x,z)}{\begin{array}{ll} & hoi(x,in(x)) \\ \wedge & hoi(x,loop_1(y)) \;\Leftarrow\; hoi(x,y) \wedge \varphi_1(y) \\ \ldots & \\ \wedge & hoi(x,loop_n(y)) \;\Leftarrow\; hoi(x,y) \wedge \varphi_n(y) \\ \wedge & \psi(x,out(y)) \;\Leftarrow\; hoi(x,y) \wedge \varphi_{n+1}(y) \end{array}} \Uparrow$$

Non-iterative presentations of $f$ lead to other instances of fixpoint induction on $f$. The input-output relation $\psi$ is often given as a *pre/post condition*: $\psi(x,z) \Longleftrightarrow (pre(x) \Rightarrow post(x,z))$. If $f$ is a kind of state modifier, $\psi$ often requires that $f$ preserves some property $p$: $\psi(x,z) \Longleftrightarrow (p(x) \Rightarrow p(z))$. In these two cases, the succedent of fixpoint induction on $f$ involves nested implications whenever some axioms for $f$ contain "recursive calls" of $f$.

If $SP$ includes a dynamic predicate $\rightarrow: s \times s$, Horn axioms specify the modal $\Diamond$-operator: for all predicates $r : s$,

$$\begin{array}{lcl} \Diamond r(x) & \Leftarrow & r(x), \\ \Diamond r(x) & \Leftarrow & x \rightarrow y \wedge \Diamond r(y). \end{array}$$

A condition of the form "if some state that is reachable from $x$ satisfies $r$, then $x$ satisfies $\psi$" can be checked by applying fixpoint induction on $\Diamond r$:

**fixpoint induction on** $\Diamond r$

$$\frac{\Diamond r(x) \Rightarrow \psi(x)}{\exists\, q : s :\; q(x) \Rightarrow \psi(x) \;\;\wedge\;\; q(x) \Leftarrow r(x) \;\;\wedge\;\; q(x) \Leftarrow (x \rightarrow y \wedge q(y))} \Updownarrow$$

Complete proofs using fixpoint induction are given in [16, 18, 20]. By dualizing fixpoint induction we obtain the coinduction rule:

**coinduction on $\nu$-predicates** $r_1 : w_1, \ldots, r_n : w_n$

$$\frac{\bigwedge_{i=1}^{n} (\psi_i(x) \Rightarrow r_i(x))}{\exists \{q_i : w_i\}_{i=1}^{n} : \bigwedge_{i=1}^{n} (\psi_i(x) \Rightarrow q_i(x) \;\wedge\; \bigwedge_{\varphi \in AX_{r_i}} \varphi[q_j/r_j \mid 1 \leq j \leq n])} \Updownarrow$$

Again, the rule leads to a successful proof only if $q_i$ is instantiated eventually such that the rule succedent becomes true. This happens only if $q_i$ *specializes* $\psi_i$, but *generalizes* $r_i$, i.e. $\psi_i(x) \Rightarrow q_i(x) \Rightarrow r_i(x)$. Hence $q_i$ is a disjunction of $\psi_i$ with further "cases" that satisfy $r_i$.

If $SP$ contains a dynamic predicate $\rightarrow: s \times s$, co-Horn axioms specify the modal $\Box$-operator: for all predicates $r : s$,

$$\begin{array}{lcl} \Box r(x) & \Rightarrow & r(x), \\ \Box r(x) & \Rightarrow & (x \rightarrow y \Rightarrow \Box r(y)). \end{array}$$

A condition of the form "if $x$ satisfies $\psi$, then all states that are reachable from $x$ satisfy $r$" can be checked by applying the coinduction on $\Box r$:

**coinduction on $\Box r$**

$$\frac{\psi(x) \Rightarrow \Box r(x)}{\exists q : s :\ \psi(x) \Rightarrow q(x) \quad \wedge \quad q(x) \Rightarrow r(x) \quad \wedge \quad (x \to y \wedge q(x)) \Rightarrow q(y)} \Updownarrow$$

Since behavioral equalities are $\nu$-predicates, a behavioral equation $t \equiv t'$ can be checked by the instance of coinduction on $\sim$ where

$$\psi(x, y) \iff \exists\ \boldsymbol{z} : (x \equiv t(\boldsymbol{z}) \wedge y \equiv t'(\boldsymbol{z})) \tag{1}$$

and $\boldsymbol{z}$ is a list of the variables of $t$ and $t'$. Due to the axioms for $\sim$ (cf. Def. 3.3), the case $q_1 = \psi$ of this instance reads as follows provided that all $s$-observers are destructors $f : sw \to s'$ and $s' = s$ or $s'$ is strongly visible (cf. Def. 3.4):

**coinduction on $\sim$: $ss$ (special case)**

$$\frac{t(\boldsymbol{u}) \sim t'(\boldsymbol{u})}{\begin{array}{l} \bigwedge\{\exists\ \boldsymbol{z} : (f(t(\boldsymbol{u}), a) \equiv t(\boldsymbol{z}) \wedge f(t'(\boldsymbol{u}), a) \equiv t'(\boldsymbol{z})) \mid f : sw \to s \text{ is a destructor}\} \\ \wedge \bigwedge\{f(t(\boldsymbol{u}), a) \equiv f(t'(\boldsymbol{u}), a) \mid f : sw \to s' \text{ is a destructor, } s' \neq s\} \end{array}} \Uparrow$$

Let us confront this rule with unfolding of $\sim$: $ss$ (cf. Section 4) under the above assumption on $s$-observers:

**unfolding of $\sim$: $ss$ (special case)**

$$\frac{t(\boldsymbol{u}) \sim t'(\boldsymbol{u})}{\begin{array}{l} \bigwedge\{f(t(\boldsymbol{u}), a) \sim f(t'(\boldsymbol{u}), a) \mid f : sw \to s \text{ is a destructor}\} \\ \wedge \bigwedge\{f(t(\boldsymbol{u}), a) \equiv f(t'(\boldsymbol{u}), a) \mid f : sw \to s' \text{ is a destructor, } s' \neq s\} \end{array}} \Uparrow$$

The comparison of both rules reveals a significant difference between unfolding and (co)induction. Unfolding just replaces occurrences of a predicate (here: $\sim$) by other occurrences of the same predicate, (co)induction removes them! We mentioned the reason in Section 1: unfolding exploits fixpoint properties, (co)induction exploits *least/greatest*-fixpoint properties. Only if the predicate is *derived*, i.e. lacks axioms with "recursive calls", unfolding and (co)induction coincide. In the above case, $\sim$: $ss$ is derived iff all $s$-destructors have strongly visible ranges.

(Co)induction and unfolding are dual to each other insofar as unfolding applies axioms to conjectures theorems, while (co)induction applies conjectures to theorems.

Since the Herbrand $SP$-model interprets $\sim$ as an equivalence relation, the succedent of (the general case of) coinduction on $\sim$ can be weakened as follows: Let $S$ be the set of sorts of $SP$. For all $s \in S$, $eq(q_s)$ denotes the equivalence closure of $q_s$.

**optimized coinduction on $\sim$**

$$\frac{\bigwedge_{s \in S} \psi \Rightarrow t_s \sim_s u_s}{\exists \{q_s : ss\}_{s \in S} : \bigwedge_{s \in S} \left\{ \begin{array}{c} \psi \Rightarrow eq(q_s)(t_s, u_s) \quad \wedge \\ \bigwedge_{(x \sim_s y \Rightarrow \varphi) \in AX} (q_s(x, y) \Rightarrow \varphi[eq(q_s)/\sim_s \mid s \in S]) \end{array} \right.} \Updownarrow$$

Similarly to $\psi$ in (1), $\psi_1, \ldots, \psi_n$ in coinduction on $\nu$-predicates $r_1, \ldots, r_n$ usually describe *sets of objects*, while $\psi_1, \ldots, \psi_n$ in fixpoint induction on $\mu$-predicates $r_1, \ldots, r_n$ represent (input-output) *relations*. This explains what it means that, in coinduction rules, $q_i$ specializes $\psi_i$: $q_i$ presents a set of objects that includes the one presented by $\psi_i$. Fixpoint induction may force us to strengthen, i.e. *reduce*, an input-output relation, while coinduction may require that the set of objects for which a $\nu$-property is to be shown is *extended*.

## 7 Conclusion

The set of rules presented in this paper claims completeness, not in the sense of covering a particular class of models, but in the sense of what the "working verifier" really needs. We have sketched typical situations where the respective rules are applied. Lack of space prevents us from giving complete proof samples, but these can be found in [12–14, 18–20]. The rules and the arguments concerning their soundness refer to the swinging type approach, a uniform framework that covers a number of more specialized approaches, ranging from purely algebraic, coalgebraic, logic- or state-oriented ones to combinations that consider both static and dynamic or both structural and behavioral aspects of a system.

## References


1. E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner, eds., *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Report, Springer 1999
2. R.S. Boyer, J.S. Moore, *A Computational Logic*, Academic Press 1979
3. A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, A. Smaill, *Rippling: A Heuristic for Guiding Inductive Proofs*, J. Artificial Intelligence 62 (1993) 185-253
4. R.M. Burstall, *Proving Properties of Programs by Structural Induction*, Comput. J. 12 (1969) 41-48
5. R.M. Burstall, J. Darlington, *A Transformation System for Recursive Programs*, J. ACM (1977) 44-67
6. J.V. Guttag, J.J. Horning, *Larch: Languages and Tools for Formal Specification*, Springer 1993
7. C.A.R. Hoare, *An Axiomatic Basis for Computer Programming*, Comm. ACM 12 (1969) 576-580
8. D. Hutter, *Guiding Induction Proofs*, Proc. CADE '90, Springer LNCS 449 (1990) 147-161
9. Z. Manna, R. Waldinger, *A Deductive Approach to Program Synthesis*, ACM TOPLAS 2 (1980) 90-121
10. J.H. Morris and B. Wegbreit, *Subgoal Induction*, Comm. ACM 20 (1977) 209-222
11. P. Padawitz, *Computing in Horn Clause Theories*, Springer 1988
12. P. Padawitz, *Inductive Expansion: A Calculus for Verifing and Synthesizing Functional and Logic Programs*, J. Automated Reasoning 7 (1991) 27-103
13. P. Padawitz, *Deduction and Declarative Programming*, Cambridge University Press 1992
14. P. Padawitz, *Expander: A System for Testing and Verifying Functional-Logic Programs*, Report No. 522, Dept. of Comp. Sci., University of Dortmund 1994, revised version: ls5.cs.uni-dortmund.de/~peter/ExpaTex.ps.gz, 1996

15. P. Padawitz, *Inductive Theorem Proving for Design Specifications*, J. Symbolic Computation 21 (1996) 41-99
16. P. Padawitz, *Proof in Flat Specifications*, in [1]
17. P. Padawitz, *Swinging Types = Functions + Relations + Transition Systems*, Theoretical Computer Science 243 (2000) 93-165
18. P. Padawitz, *Theorie der Programmierung*, Course Notes, University of Dortmund 1998-2000, ls5.cs.uni-dortmund.de/~peter/TdP96.ps.gz
19. P. Padawitz, *Modular Swinging Types*, Report, University of Dortmund 1999, ls5.cs.uni-dortmund.de/~peter/MST.ps.gz
20. P. Padawitz, *Sample Swinging Types*, Report, University of Dortmund 2000, ls5.cs.uni-dortmund.de/~peter/BehExa.ps.gz
21. P. Padawitz, *Swinging Types II: Hierarchical and Coalgebraic Specifications*, Report, University of Dortmund 2001, ls5.cs.uni-dortmund.de/~peter/BehCoalg.ps.gz
22. H.A. Partsch, *Specification and Transformation of Programs*, Springer 1990
23. L.C. Paulson, *Isabelle: A Generic Theorem Prover*, Springer LNCS 828 (1994)
24. G. Roşu, J.A. Goguen, *Circular Coinduction*, UCSD Report, San Diego 2000, www-cse.ucsd.edu/users/goguen/ps/ccoind.ps.gz