Structured Swinging Types

Peter Padawitz
peter.padawitz@udo.edu
University of Dortmund, Germany
February 18, 2006

Abstract

Swinging types (STs) provide an axiomatic specification formalism for designing and verifying software in terms of many-sorted logic and *canonical* models. STs are *one-tiered* insofar as static and dynamic, structural and behavioral aspects of a system are treated on the same syntactic and semantic level. Canonical models interpret relations as least or greatest fixpoints. All reasoning about a particular ST can be reduced to deductive processes, from built-in simplifications via resolution upon relations, narrowing upon functions, up to interactive proofs employing induction and coinduction rules.

In this paper, the different possibilities of building up an ST are clearly separated from each other. The designer of an ST may choose among six specification patterns when extending a given ST by new components. Semantically, this leads to *stratified* models, similar to those known from the semantics of stratified logic programs.

Predicates (relations interpreted as least fixpoints) and functions are axiomatized by Horn clauses, copredicates (relations interpreted as greatest fixpoints) are axiomatized by co-Horn clauses. These notions are generalized in this paper such that quantifiers may now occur at any place and even negation is permitted in axioms. For ensuring monotonicity and thus the existence of fixpoints, each relation preceded by a negation symbol must be axiomatized on a lower specification level. Under this assumption, any ST can be transformed into an equivalent one without negation symbols.

When an ST is developed stepwise, particular attention must be paid to the addition of defined functions and behavioral equalities in order to guarantee that they are fully compatible with other signature elements. Here *functionality* and *behavioral consistency* are the crucial requirements to an ST. Moreover *strong constructors* are introduced as a further means for specifying behavioral equalities, which are usually axiomatized only in terms of observers. Provided that the ST is behaviorally consistent, *behavioral* equations whose sides are dominated by strong constructors may be decomposed, such as *structural* equations may be splitted if their sides are dominated by (arbitrary) constructors.

Moreover, a simple and intuitive notion of *refinement* for STs is presented along with a powerful and completely deductive criterion for refinement correctness. *** sections 8, 9

Contents

1 Introduction

Swinging types (STs) provide an axiomatic specification formalism for designing and verifying software in terms of many-sorted logic and canonical models. STs are *one-tiered* insofar as static and dynamic, structural and behavioral aspects of a system are treated on the same syntactic and semantic level. Since the canonical models are collections of least and greatest relational fixpoints, all testing or verifying of the specification can be reduced to deductive processes, from narrowing upon functions, resolution upon relations via built-in simplifications to interactive inductive or coinductive proofs.

Unstructured swinging types were introduced in [37]. Here we add structure to the development of STs by providing several specification patterns for building up a swinging type stepwise:

- extension by **sorts**. Data sets are specified in terms of new sorts, their constructors, structural equality, inequality and definedness relations.
- extension by **local relations**. Relations are specified in terms of generalized Horn axioms (that may involve universal quantifiers in their premises). Local relations are fully compatible with behavioral equalities.
- extension by **transition relations**. Relations are specified in terms of generalized Horn axioms. Transition relations are *zigzag compatible* with behavioral equalities and thus ensure that the latter are (greatest) bisimulations.
- extension by **defined functions**. Total functions are specified in terms of conditional equations. (Partiality is expressed in terms of sum sorts.)
- extension by **behavioral equalities**. Behavioral equalities are specified in terms of co-Horn axioms involving destructors, relational observers and/or strong constructors.
- extension by **copredicates**. Relations are specified in terms of co-Horn axioms.

Any (finite) number of these "building blocks" forms a (structured) ST. The order in which they are put together is unconstrained. For instance, axioms for defined functions may refer to already specified copredicates, axioms for copredicates may use predicates and data sets may be introduced in different extensions as long as they do not involve mutually-recursive constructors. The above patterns capture a number of applications that could previously be handled only by different formalisms. For instance, traditional algebraic specifications are restricted to functions, relational algebra and logic programming only deal with *local* relations, and modal and temporal logics are tailored to *transition* relations.

The above patterns are restricted to the specification of finitely generated types. However, the concept of a structured ST admits further patterns, in particular those that add coalgebraic, non-finitely-generated types. As far as the coalgebras are constructor-based, all important proof-theoretical issues, such as powerful criteria for specifying behavioral equalities and coinductive proof rules, are already covered by finitely generated STs. The integration of coalgebraic extensions specifying non-finitely-generated types will be presented in a subsequent paper. The goal of integrating several ways of axiomatizing data types is also pursued by the specification languages Maude [9], CafeOBJ [10] and, most recently, CoCASL [30]. Maude and CafeOBJ are based upon on equational logic and (associative-commutative) *rewriting logic*, CoCASL combines algebraic with coalgebraic specifications and is built upon CASL [6], which is a joint development of several European universities.

Since STs are many-sorted specifications, the way they are connected by signature morphisms (for realizing parameterization and refinement) and the way formulas and proofs are translated along the morphisms is essentially the same as in purely algebraic specification languages. However, signature morphisms may cross the boundaries between different classes of functions or relations. For instance, when refinements are specified in terms of morphisms, constructors are often mapped to defined functions and structural to behavioral equalities.

One of the main benefits one draws from designing a specification as a sequence of ST extensions is the fact that their canonical models do not only reflect the intended semantics in a quite simple way, but also provide powerful proof rules and strategies when one reasons about the specification. Many of these rules were implemented in the proof editor *Expander2* [40, 41]. Here the user interacts at three levels of decreasing control over a proof or a computation. At the high level, analytic and synthetic inference rules, including induction, coinduction and lemma application, are applied individually and locally to selected subformulas. At the medium level, rewriting, narrowing and resolution realize the iterated and exhaustive application of all axioms of the ST. At the low level, built-in Haskell functions simplify or (partially) evaluate terms and formulas and thus hide from the user most routine steps of the current proof or computation. The simplifier also handles higher-order functions, higher-order predicates and AC operators.

STs are located somewhere between purely axiomatic and purely model-based specifications. On the one hand, designers have rather, though abstract, but individual, canonical models than entire model classes in mind when they build a system. On the other hand, only axioms as the core of a specification provide the basis for powerful proof and evaluation rules for reasoning about the models. The deductive analysis with *Expander2* of various swinging types pertaining to quite different application areas provided the main impetus for the further developments of the ST approach presented in this paper.

[37] provides the model-theoretic and deductive foundations of unstructured STs, in particular the construction of canonical models as relational fixpoints, the modal-logic issues that come with the integration of transition relations, criteria for *continuity* and *behavioral consistency*, and basic proof rules.

A swinging type starts out from constructors for building up visible as well as hidden data domains. A visible domain is characterized by the coincidence of its structural with its behavioral equality. Further predicates (μ-predicates in terms of [37]), *copredicates* (ν-predicates in terms of [37]) and defined functions are axiomatized by Horn or *co-Horn* clauses. In this paper, the notion of a Horn clause is generalized insofar as its premise may involve universal quantifiers[1] (as co-Horn clauses may involve existential quantifiers in the conclusion). Horn clauses provide the usual syntax for functional-logic programs, SOS rules as well as labelled transition systems. Copredicates often express behaviorial properties "in the infinity", such as safety or invariance conditions on sequences of states. If an ST has hidden sorts, then at least the associated behavioral equalities come as copredicates.

The unrestricted order in which an ST is built up admits, for instance, the use of predicates or even copredicates in the axioms for a defined function. Behavioral equalities, which are usually defined in terms of destructors (functional observers) or relational observers, may now be axiomatized also in terms

[1] [23, 28, 50] have been investigated similar generalizations.

of *strong constructors*. Relations are either *local* or *transitional*. The axioms for a behavioral equality $\sim$ that is defined by destructors or local observers express a congruence property of $\sim$ (compatibility with the observers), transitional observers make $\sim$ into a bisimulation (*zigzag compatibility* with the observers), and strong constructors enforce a decomposability property of $\sim$ (*inverse compatibility* with the constructors). The most common (strong) constructors are the injections into a sum sort. (Sum sorts serve as ranges of totalized partial functions.) The most common destructors are the projections mapping a product sort into its components. In fact, one may be content with these two examples if there would not be the more interesting *recursive* data types, which are characterized by constructors or destructors whose range sort also occurs as an argument.

Section 2 presents the syntax of STs and shows which kind of signature elements and axioms yield the above-mentioned specification patterns. Section 3 provides the semantics of STs in terms of *canonical* models. These models are stratified in accordance with the inductive definition of STs as a sequence of extensions. Section 3 also defines the *logical completion* of an ST that removes all negation symbols from the axioms. Semantically, logical completion preserves canonicity. An ST-model A is *continuous* if the step functions induced by the axioms of the ST on A are continuous and thus admit inductive proofs of properties of A. We extend to structured STs the almost syntactical continuity criterion called *image finiteness*, which was introduced in [37].

Section 4 presents the basic (synthetic) proof system for an ST, the *deductive calculus*, which generalizes the synonymous one introduced for unstructured STs in [37]. Due to the generalization of Horn clauses to implications with universal quantifiers in the premise, the deductive calculus involves a $\forall$-introduction rule with infinitely many premises. This enforces the use of ordinal numbers for measuring proofs and inducing on proof lengths. The deductive calculus for an ST SP defines the *SP-initial model*, which will be shown to be the initial object in the category of reachable SP-models with equality—provided that SP is *functional*, i.e. each ground term is structurally SP-equivalent to a unique normal form (= constructor term).

Section 5 focuses on the possibilities to axiomatize behavioral equalities in terms of observers *or* (the new concept of) strong constructors. Moreover, the main inference rules for reasoning about an ST's inductive theory (= theory of the initial model) are listed and discussed here. More details on this issue can be found in [41].

Section 6 shows how the copredicates of an ST are translated into predicates so that the ST is turned into its *Horn version*. For ensuring that this transformation preserves canonicity, both the original ST and its Horn version must be continuous. The Horn version is crucial for strengthening the deductive calculus to the *reductive calculus*, which is an analytic proof system with "oriented" axiom application (rewriting of functions and resolution of predicates). This calculus had already been used in several papers written by the author of this one and is generalized here to the extended syntax of Horn clauses, similarly to the generalization of the deductive calculus. As in its previous versions, it allows us to formulate powerful criteria for functionality and to show that functional STs can be translated into equivalent completely relational ones.

Section 7 deals with relations between several STs that are set up by signature morphisms. We present criteria for monotonicity and (relative) consistency along such morphisms. A refinement notion for STs is introduced that captures the usual correctness conditions on such a development step in terms of initial ST-models: (1) the "implementation" satisfies the "abstract" axioms; (2) the "implementation" is sharp in the sense that it does not add "unwanted" properties to the "abstract" requirements. We show that (2) can be concluded immediately from one of the consistency criteria.

Section 8 adapts the criterion for behavioral consistency (= weak congruence property of behavioral equivalence) given by [37], Thm. 6.5, to the hierarchical notion of an ST introduced in this paper. In particular, strong constructors are taken into account here.

In Section 9, types with "exceptions" are turned into STs whose behavioral equivalence coincides with strong equality, and partial-recursive functions built up of regular primitives are expressed in terms of swinging types.

2 The syntax of structured STs

We assume familiarity with the basic notions of many-sorted logic with equality (cf., e.g., [16, 13, 51]). Given a term or formula φ, **var**(φ) and **freevar**(φ) denote the sets of all resp. free variables of φ. φ is **ground** if $var(\varphi)$ is empty. Given an expression (term, formula, specification, etc.) e and terms or formulas $t_1,\dots,t_n,u_1,\dots,u_n$, $e[t_1/u_1,\dots,t_n/u_n]$ denotes the expression obtained from e by substituting t_i for u_i for all $1\le i\le n$.

Definition 2.1 (signature, terms, substitutions) A **signature** $\Sigma=(S,F,LR,TR)$ consists of a set S of **sorts**, S^+-sorted sets F of **functions**[2] and LR of **local relations** and an $S\times S^+$-sorted set TR of **transition relations** such that for all $s\in S$, LR implicitly includes the **structural s-equality** $\equiv_s$: ss, the **structural s-inequality** $\not\equiv_s$: ss and the **definedness predicate** $Def_s : s$. A relation is **logical** if it is not a structural equality.[3]

Given an S-sorted set X of variables, $T_\Sigma(X)$ denotes the S-sorted sets of Σ-terms whose variables are taken from X. If X is empty, we write T_Σ instead of $T_\Sigma(X)$.

An expression $p=r(t_1,\dots,t_n)$ is a **Σ-atom** if $r:s_1\dots s_n\in LR\cup TR$ and for all $1\le i\le n$, $t_i\in T_{\Sigma,s_i}$. p is **logical** if r is a logical relation. p and $\neg p$ are called **Σ-literals**.

An S-sorted function $\sigma:X\to T_\Sigma(X)$ is called a **substitution**. The **domain of** σ, $dom(\sigma)$, is the set of all variables x with $x\sigma\ne x$. Given $Y\subseteq X$, σ_Y denotes the restriction of σ that is defined by $x\sigma_Y=x\sigma$ for all $x\in Y$ and $x\sigma_Y=x$ for all $x\in X\setminus Y$. Given a further substitution τ, we write $\sigma=_Y\tau$ if σ and τ agree on all variables of $X\setminus Y$.

If σ maps each variable of $dom(\sigma)$ to a term in some given set T of terms, we write $\sigma:X\to T$ in order to indicate that σ satisfies $\sigma(dom(X))\subseteq T$. The **instance** $t\sigma$ of a term or formula t **by** σ is obtained from t by replacing each free occurrence in t of a variable x by $x\sigma$. We also use the bracket notation for substitutions (see above).

Let $\Sigma=(S,F,LR,TR)$ and $\Sigma'=(S',F',LR',TR')$ be signatures. A **signature morphism** $\sigma:\Sigma\to\Sigma'$ consists of a function $\sigma_{sorts}:S\to S'$ and S^+-sorted sets of functions $\sigma_{funs}=\{\sigma_w:F_w\to F'_{\sigma(w)}\}$, $\sigma_{preds}=\{\sigma_w:LR_w\to LR'_{\sigma(w)}\}$ and $\sigma_{trans}=\{\sigma_w:TR_w\to TR'_{\sigma(w)}\}$ such that for all $f:w\to s\in F$, $\sigma(f):\sigma(w)\to\sigma(s)$ and for all $r:w\in LR\cup TR$, $\sigma(r):\sigma(w)$. ❏

Similarly to our transition relations, *Dynamic Data Types* and *Labelled Transition Logic* [8, 1] incorporate transition systems as relations into specifications and axiomatize them in terms of Horn clauses, which, by the way, amount to nothing else but SOS ("structural operational semantics") rules, the classical syntax of transition system specifications. The logic used for reasoning about dynamic data types is a

[2] Of course, on this syntactic level, F, LR and TR are just sets of *symbols*.

[3] Whether a relation is a local relation, a transition relation or a copredicate depends on the type of axioms that specify it (cf. Def. 2.3). Structural equalities are the only relations that are not associated with these categories.

temporal one. Swinging types go a step further and admit to integrate not only transitions systems, but also temporal- and modal-logic operators to reason about them. Such operators come as (higher-order) local relations and are specified by Horn or co-Horn clauses (see below), which are direct translations of their interpretations in the modal μ-calculus as least resp. greatest fixpoints (cf. [37], Section 2). Maude [9] and CafeOBJ [10] specify unary relations in terms of membership predicates and transition relations in terms of rewrite rules and categorical initial models. ??

Definition 2.2 (formulas) Let $\Sigma = (S, F, LR, TR)$ be a signature. A **Σ-formula** is a first-order formula consisting of symbols from Σ and variables from an S-sorted set X of variables. A Σ-formula is **positive** if it does not contain implication symbols and all negation symbols are at literal positions.

Let φ be a positive Σ-formula. Given a logical Σ-atom $p = r(t)$, $p \Leftarrow \varphi$ is a **Horn clause**[4] for r and $p \Rightarrow \varphi$ is a **co-Horn clause** for r. Given a Σ-atom $p = (f(t) \equiv u)$, $p \Leftarrow \varphi$ is a **Horn clause** for f. A Horn clause $p \Leftarrow \mathit{True}$ is identified with p.

By convention, any structural property of a Σ-formula φ—except for the position of the implication sign in a Horn or co-Horn clause—holds true as well for all formulas that are logically equivalent to vfi w.r.t. the validity of first-order formulas.

The set of **poly-modal formulas** is inductively defined as follows:

- An atom $r(t)$ with $r \in LR$ is poly-modal.
- If φ and ψ are poly-modal, then $\neg\varphi$ and $\varphi \wedge \psi$ are poly-modal.
- If φ is poly-modal, then for all atoms $\delta(t, x)$ with $\delta \in TR$ and $x \in X \setminus \mathit{var}(t)$, $\exists x : (\delta(t, x) \wedge \varphi)$ is poly-modal.

The set of **weakly modal formulas with output** $\mathit{out}(\varphi) \subseteq X$ is inductively defined as follows:

- A poly-modal formula is weakly modal with output $\emptyset$.
- An atom $\delta(t, x)$ with $\delta \in TR$ and $x \in X \setminus \mathit{var}(t)$ is weakly modal with output $\{x\}$.
- If φ and ψ are weakly modal with disjoint outputs Y resp. Z such that $Z \cap \mathit{freevar}(\varphi) = \emptyset$, then $\varphi \wedge \psi$ is weakly modal with output $Y \cup Z$.
- If φ is weakly modal with output Y, then for all $x \in X$, $\exists x : \varphi$ is weakly modal with output $Y \setminus \{x\}$. ❏

Poly-modal and weakly modal formulas are invariant with respect to the replacement of a variable valuation by a *weakly congruent* one (cf. Def. 3.1 and [37], Thm. 3.8).

Substitutions extend to formulas as usual. Let $Q \in \{\forall, \exists\}$. Quantified variables are not substituted, i.e., for all $x \in X$, $(Qx : \varphi)\sigma = Qx : \varphi\sigma_{X \setminus \{x\}}$. Moreover, given a set $Y = \{x_1, \ldots, x_n\}$ of variables, the formula $QY : \varphi$ stands for $Qx_1 : \cdots : Qx_n : \varphi$.

Definition 2.3 A **swinging type (ST)** is ... see [39]

Note that each function symbol, predicate or copredicate of an ST SP is axiomatized *either* in the base type *or* in the extension of SP. The only (non-variable) symbols that do not fall into one of these categories, but may occur in SP are structural equalities. If the extension contains defined functions (see

[4] Since φ is not confined to finite conjunctions of atoms, our notion of a Horn clause deviates from the classical one. It also does not coincide with the notion of a *hereditary Harrop formula* [27]. Premises with universal quantifiers, which do not occur in classical Horn clauses, are allowed both in Harrop formulas and in our Horn clauses. But Harrop formulas impose further restrictions on their premises.

2.3(2)), conditional equations to axiomatize them are introduced that may modify (semantically) their leading structural equalities. This impact on structural equalities, which, of course, cannot be avoided in a hierarchical construction of combined functional and relational specifications, is the reason for excluding structural equalities from the set of *predicates* of an ST. Still, structural equalities share with predicates the property that all axioms with such a relation as the leading one are Horn clauses.

Although Def. 2.3 excludes the specification of alternating fixpoints [31], it is sufficient for axiomatizing all common modal- or temporal-logic operators in terms of an ST of order 3 (cf. [37], Example 2.7).

3 Canonical and continuous models

Definition 3.1 (semantical notions) see ... [39]

Note that a relation $\approx\ \subseteq A \times B$ is zigzag compatible with $\delta : ws$ iff it is compatible with the "non-deterministic" function $f_\delta : A_w \to \wp(A_s)$ that maps $a \in A_w$ to the set of all $a' \in A_s$ with $(a, a') \in \delta^A$. Here $\approx$ is extended to a subset of $\wp(A_s) \times \wp(B_s)$ as follows:

$$A' \approx B' \quad\Longleftrightarrow\quad \begin{cases} \forall\, a \in A'\ \exists\, b \in B' : a \approx_s b, \\ \forall\, b \in B'\ \exists\, a \in A' : a \approx_s b. \end{cases}$$

A Σ-structure A interprets 1 by the set $\{()\}$, a product sort $s_1 \times \cdots \times s_n$ by the Cartesian product $A_{s_1} \times \ldots \times A_{s_n}$ and a sum sort $\amalg_{i \in I} w_i$ by $\{\kappa_i(a) \mid a \in A_{w_i},\ i \in I\}$. Projections and injections are interpreted accordingly: For all $1 \le i \le n$ and $a = (a_1, \ldots, a_n) \in A_{s_1 \ldots s_n}$, $\pi_i^A(a) =_{def} a_i$. For all $i \in I$ and $a \in A_{s_i}$, $\kappa_i^A(a) =_{def} \kappa_i(a)$. Hence κ_i^A is injective and for all $i, j \in I$, $i \neq j$ implies that $\kappa_i^A(A_{s_i})$ and $\kappa_j^A(A_{s_j})$ are disjoint.

$<$ (see Section 2) yields an ordering $<^A$ between elements of different carriers of A: for all $s < s'$, $a \in A_s$ and $a' \in A_{s'}$,

$$a <^A a' \quad \Longleftrightarrow_{def} \quad \text{there are injections } c_1, \ldots, c_n \text{ such that } c_1(\ldots(c_n(a))\ldots) = a'.$$

Note that for all $a' \in A_{s'}$ there is at most one $a \in A_s$ such that $a <^A a'$. Hence the following interpretation of $f_+ : s' \to 1 + s''$ in A is well-defined: for all $a' \in A_{s'}$,

$$f_+^A(a') \ =_{def} \ \begin{cases} () & \text{if for all } a \in A_s,\ a \not<^A a', \\ (f^A(a)) & \text{if } a <^A a'. \end{cases}$$

Analogously, the interpretation of $r : s$ in A determines an interpretation of $r_+ : s'$ in A:

$$r_+^A \ =_{def} \ \{a' \in A_{s'} \mid \exists\, a \in r^A : a <^A a'\}.$$

Proposition 3.2 *Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism, A be a Σ'-structure and φ be a Σ-formula. $A|_\sigma$ satisfies φ iff A satisfies $\sigma(\varphi)$. A_σ satisfies φ iff for all $\tau : X \to T_\Sigma$, A satisfies $\sigma(\varphi\tau)$.* ❏

We are not interested in all models of a swinging type, but only in canonical ones where the interpretation of relations by least or greatest fixpoints of a *step function*:

Definition 3.3 see ... [39]

Definition and Theorem 3.4 (continuity, fixpoints) see ... [39]

Definition 3.5 (continuous and canonical models) Let $SP = (\Sigma, AX)$ be a swinging type with base type $baseSP = (base\Sigma, baseAX)$ and extension (Σ', AX'), B be a reachable Σ-structure with equality and inequality and Φ be the SP-step function on $A = B|_{base\Sigma}$.

B is a **continuous SP-model** if SP is the empty ST or A is a continuous $baseSP$-model and Φ is upward resp. downward continuous (in case 2.3(1/3/4) resp. (5/6)). B is a **canonical SP-model** if SP is the empty ST or the following conditions hold true:

➢ A is a canonical $baseSP$-model.

➢ In case 2.3(1/3/4), for all predicates $r \in \Sigma'$, $r^B = r^{lfp(\Phi)}$.

➢ In case 2.3(2), B satisfies AX'.

➢ In case 2.3(5/6), for all copredicates $r \in \Sigma'$, $r^B = r^{gfp(\Phi)}$.

A formula satisfied by all canonical SP-models is an **inductive theorem of SP**. Given a parameterized swinging type PSP, an **inductive theorem of PSP** is an inductive theorem of all actualizations of PSP. ❑

Proposition 3.6 Let $SP = (\Sigma, AX)$ be a swinging type with base type $baseSP = (base\Sigma, baseAX)$ and extension (Σ', AX') and B be a continuous and canonical SP-model. Then by Theorem 3.4 (Kleene), the following conditions hold true:

➢ In case 2.3(1/3/4), for all predicates $r \in \Sigma'$, $r^B = \cup_{i\in\mathbb{N}} r^{\Phi^i(\bot)}$.

➢ In case 2.3(5/6), for all copredicates $r \in \Sigma'$, $r^B = \cap_{i\in\mathbb{N}} r^{\Phi^i(\top)}$. ❑

Canonicity is sufficient for turning an ST SP into its logical completion where all relations of SP have complements and thus negation symbols can be removed from the axioms:

Definition 3.7 Let $\Sigma = (S, F, LR, TR)$ be a signature. The signature

$$compl(\Sigma) \;=\; (S, F, LR \cup \{\overline{r} : w \mid r : w \in LR \cup TR\}, TR)$$

is called the **logical completion of** Σ.[5] Let φ be a Σ-formula. The $compl(\Sigma)$-formula **pos**(φ) is obtained from φ by first transforming φ into an equivalent positive formula φ' and then replacing each literal $\neg r(t)$ of φ with $\overline{r}(t)$. The $compl(\Sigma)$-formula **neg**(φ) is obtained from φ' by dualizing quantifiers, conjunctions and disjunctions and replacing each atom $r(t)$ of φ' with $\overline{r}(t)$ and each literal $\neg r(t)$ of φ' with $r(t)$.

Let CL be a set of (co-)Horn clauses, CLP be the set consisting of all clauses of CL for predicates and

$$pos(CL) = \{p \Leftarrow pos(\varphi) \mid p \Leftarrow \varphi \in CL\} \cup \{p \Rightarrow pos(\varphi) \mid p \Rightarrow \varphi \in CL\}.$$

The set

$$\begin{aligned} compl(CL) \;=\;& pos(CL) \cup \{\overline{r}(t) \Rightarrow False \mid r(t) \in CLP\} \cup \{\overline{r}(t) \Rightarrow neg(\varphi) \mid r(t) \Leftarrow \varphi \in CLP\} \\ & \cup \{\overline{r}(t) \mid r(t) \Rightarrow False \in CL\} \cup \{\overline{r}(t) \Leftarrow neg(\varphi) \mid r(t) \Rightarrow \varphi \in CL\} \end{aligned}$$

of (co-)Horn clauses is called the **logical completion of** CL.

Given a swinging type $SP = (\Sigma, AX)$, the ST $compl(SP) = (compl(\Sigma), compl(AX))$ is called the **logical completion of** SP. ❑

Example 3.8 The following parameterized swinging type specifies the set of all finite sequences. For the parameter type ENTRY, see Example **??**.

[5] The actual name for $\overline{r}$ depends on r (cf. Example 3.8).

LIST = ENTRY **then**

`vissorts`	$list = list(entry)$	
`constructs`	$[] :\rightarrow list$	
	$\_ : \_ : entry \times list \rightarrow list$	
`local preds`	$\_ \in \_ : entry \times list$	
	$sorted : list$	
	$exists, forall : (entry \rightarrow bool) \times list$	
`vars`	$x, y : entry \quad L, L' : list \quad g : entry \rightarrow bool$	
`Horn axioms`	$x \in y : L \quad \Leftarrow \quad x \equiv y \vee x \in L$	(1)
	$sorted([])$	(2)
	$sorted(x : [])$	(3)
	$sorted(x : y : L) \quad \Leftarrow \quad x \leq y \wedge sorted(y : L)$	(4)
	$exists(g, x : L) \quad \Leftarrow \quad g(x) \equiv true \vee exists(g, L)$	(5)
	$forall(g, [])$	(6)
	$forall(g, x : L) \quad \Leftarrow \quad g(x) \equiv true \wedge forall(g, L)$	(7)

The complement of $\{(1), \ldots, (7)\}$ reads as follows:

$$
\begin{array}{l}
x \notin y : L \quad \Rightarrow \quad x \not\equiv y \wedge x \notin L \\
unsorted([]) \quad \Rightarrow \quad False \\
unsorted(x : []) \quad \Rightarrow \quad False \\
unsorted(x : y : L) \quad \Rightarrow \quad x \not\leq y \vee unsorted(y : L) \\
notExists(g, x : L) \quad \Rightarrow \quad g(x) \not\equiv true \wedge notExists(g, L) \\
notForall(g, []) \quad \Rightarrow \quad False \\
notForall(g, x : L) \quad \Rightarrow \quad g(x) \not\equiv true \vee notForall(g, L)
\end{array}
$$

The use of $compl(SP)$ in proofs about SP may be regarded as a way of simulating default logics [44] with swinging types.

Theorem 3.9 (logical completion preserves canonicity) *Let $SP = (\Sigma, AX)$ be a swinging type, A be a canonical SP-model and B be the $compl(\Sigma)$-structure that is defined by $B|_\Sigma = A|_\Sigma$ and $\overline{r}^B = A_w \setminus r^A$ for each logical predicate or copredicate $r : w \in \Sigma$. B is a canonical $compl(SP)$-model.*

Proof. Let $baseSP = (base\Sigma, baseAX)$ be the base type of SP and A be a canonical SP-model A. We show the existence of B by induction on the structure of SP (cf. Def. 2.3). Since $A|_{base\Sigma}$ is a canonical $baseSP$-model, the induction hypothesis implies that $B|_{base\Sigma}$ is a canonical $compl(baseSP)$-model. In case 2.3(2), the proof is complete because then SP adds neither predicates nor copredicates to $baseSP$.

Let case 2.3(1/3/4) hold true, $P = \Sigma \setminus base\Sigma$, (Σ', AX') be the extension of SP, $compl(baseSP) = (\Sigma_0, AX_0)$, $compl_0(SP) = (\Sigma_0 \cup \Sigma', AX_0 \cup pos(AX'))$, Φ be the $compl_0(SP)$-step function on $B|_{base\Sigma}$ and Ψ be the $compl(SP)$-step function on $B|_{\Sigma_0}$. Since $B|_{base\Sigma}$ is a canonical $compl(baseSP)$-model, B is a canonical $compl(SP)$-model if for all $r \in P$,

$$r^B \;=\; r^{lfp(\Phi)} \tag{1}$$

and

$$\overline{r}^B = \overline{r}^{gfp(\Psi)}. \tag{2}$$

So let $r : w \in P$ and $cl_1 = (r(t_1) \Leftarrow \varphi_1), \ldots, cl_n = (r(t_n) \Leftarrow \varphi_n)$ be the axioms of SP for r. Then

$$r(t_1) \Leftarrow pos(\varphi_1), \ldots, r(t_n) \Leftarrow pos(\varphi_n) \quad \text{are the axioms of } compl_0(SP) \text{ for } r \tag{3}$$

and

$$\overline{r}(t_1) \Rightarrow neg(\varphi_1), \ldots, \overline{r}(t_n) \Rightarrow neg(\varphi_n) \quad \text{are the axioms of } compl(SP) \text{ for } \overline{r}. \tag{4}$$

By (3), Φ coincides with the SP-step function Θ on $A|_{base\Sigma}$ and thus $r^B = r^A = r^{lfp(\Theta)} = r^{lfp(\Phi)}$ because $B|_\Sigma = A|_\Sigma$ and $A|_\Sigma$ is a canonical SP-model. Hence (1) holds true.

The axioms of $compl_0(SP)$ for r can be combined into a single Horn clause $r(x) \Leftarrow \varphi$ where $x \in X \setminus \cup_{i=1}^n freevar(cl_i)$ and

$$\varphi \quad = \quad \bigvee_{i=1}^{n} \exists freevar(cl_i) : (x \equiv t_i \wedge pos(\varphi_i)).$$

Since B is a fixpoint of Φ, B satisfies $r(x) \Leftrightarrow \varphi$ and thus $\overline{r}(x) \Leftrightarrow \psi$ where

$$\psi \quad = \quad \bigwedge_{i=1}^{n} \forall freevar(cl_i) : (x \not\equiv t_i \vee neg(\varphi_i)).$$

By (4), $\overline{r}(x) \Rightarrow \psi$ is the single co-Horn clause the axioms of $compl(SP)$ for $\overline{r}$ can be combined into. Hence B satisfies the axioms of $compl(SP)$ for $\overline{r}$ and thus is a fixpoint of Ψ. Since $gfp(\Psi)$ is the greatest one, $\overline{r}^B \subseteq \overline{r}^{gfp(\Psi)}$. Hence by the definition of $\overline{r}^B$, (2) holds true if $\overline{r}^{gfp(\Psi)} \subseteq A_w \setminus r^A$. Since $r^A = r^{lfp(\Phi)}$, this inclusion reduces to $\overline{r}^{gfp(\Psi)} \subseteq A_w \setminus r^{lfp(\Phi)}$ and thus, by Theorem 3.4 (Knaster-Tarski), to:

$$\cup_{C \in \mathcal{C}} \{\overline{r}^C \mid \overline{r}^C \subseteq \overline{r}^{\Psi(C)}\} \quad \subseteq \quad \cup_{D \in \mathcal{D}} \{A_w \setminus r^D \mid r^{\Phi(D)} \subseteq r^D\} \tag{5}$$

where $\mathcal{C}$ is the class of $compl(\Sigma)$-structures C with $C|_{\Sigma_0} = B|_{\Sigma_0}$ and $\mathcal{D}$ is the class of all Σ_0-structures D with $D|_{base\Sigma} = B|_{base\Sigma}$. (5) reduces to:

$$\forall\, C \in \mathcal{C}\ \forall\, t \in \overline{r}^C : (\overline{r}^C \subseteq \overline{r}^{\Psi(C)} \ \Rightarrow\ \exists\, D \in \mathcal{D} : (t \notin r^D \wedge r^{\Phi(D)} \subseteq r^D)). \tag{6}$$

So let $C \in \mathcal{C}$ and $t \in \overline{r}^C$ such that $\overline{r}^C \subseteq \overline{r}^{\Psi(C)}$. We define $D \in \mathcal{D}$ by $D|_{base\Sigma} = C|_{base\Sigma}$ and $q^D = A_v \setminus \overline{q}^C$ for all $q : v \in P$. Then $t \in \overline{r}^C$ implies $t \notin r^D$. Moreover, for all $u \in A_w$,

$$\begin{aligned} u \in r^{\Phi(D)} \quad &\Longleftrightarrow \quad D \models_{[u/x]} \varphi \quad \Longleftrightarrow \quad D \not\models_{[u/x]} \neg\varphi \quad \Longleftrightarrow \quad C \not\models_{[u/x]} \psi \quad \Longleftrightarrow \quad u \notin \overline{r}^{\Psi(C)} \\ &\stackrel{(*)}{\Longleftrightarrow} \quad u \notin \overline{r}^C \quad \Longleftrightarrow \quad u \in r^D \end{aligned} \tag{7}$$

where $(*)$ follows from $\overline{r}^C \subseteq \overline{r}^{\Psi(C)}$. (7) implies $r^{\Phi(D)} \subseteq r^D$. Hence (6) holds true and we conclude that, in case 2.3(1/3/4), B is a canonical $compl(SP)$-model.

The proof for case 2.3(5/6) proceeds analogously if one dualizes the arguments of the proof for case 2.3(1/3/4). ❑

4 The deductive calculus and the initial model

Given a swinging type SP, canonical SP-models can be derived from an extension $DTh(SP)$. The extension uses two additional concepts. The first one is the notion of an *deductive set* [2], which allows us to define derived as well as refuted formulas and thus least as well as greatest relational fixpoints. The second concept is the measurement of proof sizes in terms of ordinal numbers. Both the cut calculus given below and the reductive calculus (Def. 6.4) contain a rule with (countably) infinitely many premises ($\forall$-introduction). In order to relate both calculi to each other we must induce on the respective proof sizes. The involvement of $\forall$-introduction forces us to use *transfinite induction* for this purpose: A property P holds true for all ordinal numbers if for all ordinals b, if P holds true for all ordinals $a < b$, then P

holds true for b. The correctness of transfinite induction is easily deduced from the fact that the ordinal numbers form a well-ordered set (see [48], §13). Ordinal numbers are sets and the well-order ($<$) is set inclusion. A *successor ordinal* has an immediate predecessor w.r.t. $<$, while a *limit ordinal* is the union of all its predecessors. Ordinal numbers for measuring proofs via inference systems involving $\forall$-introduction rules have already been used in, e.g., [48], §20, and [43], Sect. 1.3.

Definition 4.1 (deductive calculus) Let $SP = (\Sigma, AX)$ be an ST with base type $baseSP$ and extension (Σ', AX'). The **cut calculus for** SP consists of the following rules for deriving (sets of) Σ-formulas.[6]

base $\qquad \dfrac{True}{\varphi} \Downarrow \quad$ for all $\varphi \in DTh(baseSP) \cup AX'$

instantiation $\qquad \dfrac{\varphi}{\varphi[t/x]} \Downarrow \quad$ for all $t \in T_{\Sigma,sort(x)}$ and $x \in var(\varphi)$

cut $\qquad \dfrac{\{\varphi \Rightarrow (\psi \vee \gamma),\ (\gamma \wedge \varphi') \Rightarrow \psi'\}}{(\varphi \wedge \varphi') \Rightarrow (\psi \vee \psi')} \Downarrow$

$\wedge$-introduction $\qquad \dfrac{\{\varphi, \psi\}}{\varphi \wedge \psi} \Downarrow$

$\vee$-introduction $\qquad \dfrac{\varphi}{\varphi \vee \psi} \Downarrow$

$\exists$-introduction $\qquad \dfrac{\varphi[t/x]}{\exists x \varphi} \Downarrow \quad$ for all $t \in T_{\Sigma,sort(x)}$ and $x \in var(\varphi)$

$\forall$-introduction $\qquad \dfrac{\{\varphi[t/x] \mid t \in T_{\Sigma,sort(x)}\}}{\forall x : \varphi} \Downarrow \quad$ for all $x \in var(\varphi)$

A set F of Σ-formulas is SP**-deductive** if for each rule of the cut calculus for SP, the conclusion belongs to F whenever the premises belong to F. In case 2.3(1/2/3/4), $\mathbf{DTh}(SP)$ denotes the intersection of all SP-deductive sets containing *True*. In case 2.3(5/6), $\mathbf{DTh}(SP)$ denotes the union of all SP-deductive sets that do not contain *False*. Elements of $DTh(SP)$ are called the **deductive theorems of** SP.

The **structural** SP**-equivalence** $\equiv_{SP}$ consists of all $(t, u) \in T_\Sigma^2$ such that $t \equiv u \in DTh(SP)$.[7] The **behavioral** SP**-equivalence** $\sim_{SP}$ consists of all $(t, u) \in T_\Sigma^2$ such that $t \sim u \in DTh(SP)$.

Let a be an ordinal number. The **deductive inference relation** $\vdash_{SP}^a$ is inductively defined as follows:

- For all $\varphi \in DTh(baseSP) \cup AX'$, $\vdash_{SP}^a \varphi$.
- Let $\varphi_1, \ldots, \varphi_n$ be the premises and φ be the conclusion of the instantiation, cut, $\wedge$-introduction, $\vee$-introduction or $\exists$-introduction rule. If $\vdash_{SP}^{a_i} \varphi_i$ and $a_i < a$ for all $1 \le i \le n$, then $\vdash_{SP}^a \varphi$.
- If $x \in var(\varphi)$ and for all $t \in T_{\Sigma,sort(x)}$, $\vdash_{SP}^{a_t} \varphi[t/x]$ and $a_t < a$, then $\vdash_{SP}^a \forall x : \varphi$.

The **proof length** of φ in the cut calculus for SP is the least ordinal number a such that $\vdash_{SP}^a \varphi$. ❏

Proposition 4.2 *For all $s \in S$ and $t, u \in NF_{\Sigma,s}$, $t \neq u$ implies $t \not\equiv_s u \in DTh(SP)$.* ❏

Definition 4.3 (initial model) Let $SP = (\Sigma, AX)$ be an ST with base type $baseSP = (base\Sigma, baseAX)$ and extension (Σ', AX').

[6] Arrows attached to a rule indicate the direction of consequence, here with respect to validity in Σ-structures.
[7] By Def. 2.3, $\equiv_{SP}$ is a Σ-congruence.

Let $\Sigma' = (S, F, LR, TR)$. The **initial** SP**-model** $Ini(SP)$ is the reachable Σ-structure A that is inductively defined as follows: If SP is the empty ST, then A is the empty Σ-structure. Otherwise $A|_{base\Sigma} = Ini(baseSP)$,

- ➢ for all $s \in S$, $A_s = T_{\Sigma,s}/\equiv_{SP}$,
- ➢ for all $f : w \to s \in F$ and $t \in T_{\Sigma,w}$, $f^A([t]) = [f(t)]$,
- ➢ for all $r : w \in LR \cup TR$, $r^A = \{[t] \in A_w \mid r(t) \in DTh(SP)\}$. ❏

Proposition 4.4 *$Ini(SP)$ is a Σ-structure with equality.*

Proof. Let $t, u \in T_\Sigma$. Then $[t] \equiv^{Ini(SP)} [u]$ iff $t \equiv u \in DTh(SP)$ iff $t \equiv_{SP}$ iff $[t] = [u]$. ❏

Proposition 4.5 *For all Σ-formulas φ, $Ini(SP) \models \varphi$ iff $\varphi \in DTh(SP)$.*

Proof. If φ is atomic, then the statement holds true by the definition of $Ini(SP)$. Otherwise proceed by induction on the structure of φ. ❏

Lemma 4.6 *For all predicates $r : w \in \Sigma$, $r^{Ini(SP)}$ is the least relation on $T_{\Sigma,w}/\equiv_{SP}$ that satisfies the axioms of SP for r.*

Proof. Let $A = Ini(SP)$. By a simple inductive argument, it is sufficient to consider case 2.3(1/3/4) and to show the statement for all predicates $r : w \in \Sigma'$. Let Φ be the SP-step function on $A|_{base\Sigma}$ and $B = \mathit{lfp}(\Phi)$. Since B is a fixpoint of Φ, B satisfies AX'. Since B is reachable and thus all rules of the cut calculus for SP are sound w.r.t. validity in B, the Σ-formulas φ with $B \models \varphi$ form an SP-deductive set F that contains *True*. Since $DTh(SP)$ is the intersection of all SP-deductive sets containing *True*, we conclude that for all predicates $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$,

$$t \in r^A \iff r(t) \in DTh(SP) \implies r(t) \in F \iff B \models r(t) \iff t \in r^B.$$

Hence $r^A \subseteq r^B$. On the other hand, B is the least fixpoint of Φ and thus $r^B \subseteq r^A$ provided that A is also a fixpoint of Φ. This holds true if A satisfies AX'.

So let $p \Leftarrow G$ be a Horn clause of AX' and $\sigma : X \to T_\Sigma$ such that A satisfies $G\sigma$. By Prop. 4.5, $G\sigma \in DTh(SP)$. Since $DTh(SP)$ is the intersection of all SP-deductive sets containing *True*, $G\sigma \in F$ for all SP-deductive sets F that contain *True*. Moreover, $p \Leftarrow G \in AX'$ implies $p\sigma \Leftarrow G\sigma \in F$ and thus, by applying the modus ponens, $p\sigma \in F$. Hence $p\sigma \in DTh(SP)$ and thus by Prop. 4.5, $A \models p\sigma$. ❏

Lemma 4.7 *For all copredicates $r : w \in \Sigma$, $r^{Ini(SP)}$ is the greatest relation on $T_{\Sigma,w}/\equiv_{SP}$ that satisfies the axioms of SP for r.*

Proof. Let $A = Ini(SP)$. By a simple inductive argument, it is sufficient to consider case 2.3(5/6) and to show the statement for all copredicates $r : w \in \Sigma'$. Let Φ be the SP-step function on $A|_{base\Sigma}$ and $B = \mathit{gfp}(\Phi)$. Since B is a fixpoint of Φ, B satisfies AX'. Since B is reachable and thus all rules of the cut calculus for SP are sound w.r.t. validity in B, the Σ-formulas φ with $B \models \varphi$ form an SP-deductive set F that does not contain *False*. Since $DTh(SP)$ is the union of all SP-deductive sets not containing *False*, we conclude that for all copredicates $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$,

$$t \in r^B \iff B \models r(t) \iff r(t) \in F \implies r(t) \in DTh(SP) \iff t \in r^A.$$

Hence $r^B \subseteq r^A$. On the other hand, B is the greatest fixpoint of Φ and thus $r^A \subseteq r^B$ provided that A is also a fixpoint of Φ. This holds true if A satisfies AX'.

So let $p \Rightarrow G$ be a co-Horn clause of AX' and $\sigma : X \to T_\Sigma$ such that A satisfies $p\sigma$. By Prop. 4.5, $p\sigma \in DTh(SP)$. Since $DTh(SP)$ is the union of all SP-deductive sets not containing *False*, $p\sigma \in F$ for some SP-deductive set F that does not contain *False*. Moreover, $p \Rightarrow G \in AX'$ implies $p\sigma \Rightarrow G\sigma \in F$ and thus, by applying the modus ponens, $G\sigma \in F$. Hence $G\sigma \in DTh(SP)$ and thus by Prop. 4.5, $A \models G\sigma$. ❑

Lemma 4.8 *Let case 2.3(2) be valid. Then $Ini(SP)$ satisfies AX'.*

Proof. Let $p \Leftarrow G$ be a Horn clause of AX' and $\sigma : X \to T_\Sigma$ such that $Ini(SP)$ satisfies $G\sigma$. By Prop. 4.5, $G\sigma \in DTh(SP)$. Since $DTh(SP)$ is the intersection of all SP-deductive sets containing *True*, $G\sigma \in F$ for all SP-deductive sets F containing *True*. Moreover, $p \Leftarrow G \in AX'$ implies $p\sigma \Leftarrow G\sigma \in F$ and thus, by applying the modus ponens, $p\sigma \in F$. Hence $p\sigma \in DTh(SP)$ and thus by Prop. 4.5, $Ini(SP) \models p\sigma$. ❑

Lemma 4.9 *Let B be a reachable SP-model. Then for all predicates and structural equalities $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$, $r(t) \in DTh(SP)$ implies $B \models r(t)$.*

Proof. Let $A = Ini(SP)$. By a simple inductive argument, it is sufficient to consider case 2.3(1/2/3/4) and to show the statement for all predicates and structural equalities $r : w \in \Sigma'$. Let Φ be the SP-step function on $A|_{base\Sigma}$. Since B be a reachable SP-model and thus all rules of the cut calculus for SP are sound w.r.t. validity in B, the Σ-formulas φ with $B \models \varphi$ form an SP-deductive set F that contains *True*. Since $DTh(SP)$ is the intersection of all SP-deductive sets containing *True*, we conclude that for all predicates and structural equalities $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$, $r(t) \in DTh(SP)$ implies $r(t) \in F$ and thus $B \models r(t)$. ❑

Lemma 4.10 *Let B be a reachable SP-model. Then for all copredicates $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$, $B \models r(t)$ implies $r(t) \in DTh(SP)$.*

Proof. Let $A = Ini(SP)$. By a simple inductive argument, it is sufficient to consider case 2.3(5/6) and to show the statement for all copredicates $r : w \in \Sigma'$. Let Φ be the SP-step function on $A|_{base\Sigma}$. Since B be a reachable SP-model and thus all rules of the cut calculus for SP are sound w.r.t. validity in B, the Σ-formulas φ with $B \models \varphi$ form an SP-deductive set F that does not contain *False*. Since $DTh(SP)$ is the union of all SP-deductive sets containing *True*, we conclude that for all copredicates $r : w \in \Sigma$ and $t \in T_{\Sigma,w}$, $r(t) \in F$ implies $r(t) \in DTh(SP)$. Hence $B \models r(t)$ implies $r(t) \in DTh(SP)$. ❑

Definition 4.11 (completeness, consistency, functionality) Let $SP = (\Sigma, AX)$ be an ST. A Σ-term **has a normal form** u if $t \equiv_{SP} u$ and $u \in NF_\Sigma$. SP is **complete** if each ground Σ-term there has a normal form. SP is **consistent** if all structurally SP-equivalent normal forms are equal. SP is **functional** if SP is complete and consistent. In this case, 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 Σ-congruence. ❑

Proposition 4.12 *SP is complete iff for all $t \in T_\Sigma$, $Def(t) \in DTh(SP)$.* ❑

Functionality implies that the initial SP-model interprets structural equalities and inequalities as complements of each other:

Lemma 4.13 *Let SP be complete. SP is consistent iff for all $t, u \in T_\Sigma$,*

$$t \not\equiv u \in DTh(SP) \qquad \Longleftrightarrow \qquad t \equiv u \notin DTh(SP). \tag{1}$$

Proof. Suppose that (1) holds true. Let $s \in S$ and $t, u \in NF_{\Sigma,s}$ such that $t \neq u$. By Prop. 4.2, $t \not\equiv_s u \in DTh(SP)$. By (1), $t \equiv u \notin DTh(SP)$. Hence SP is consistent.

Suppose that SP is consistent. Let $t, u \in T_\Sigma$ such that $t \equiv u \notin DTh(SP)$. Since SP is functional, this is equivalent to: $nf(t) \neq nf(u)$. By Prop. 4.2, $nf(t) \not\equiv nf(u) \in DTh(SP)$. Hence $t \not\equiv u \in DTh(SP)$.

Conversely, let $A = Ini(SP)$, $s \in S$ and $t, u \in T_{\Sigma,s}$ such that $t \not\equiv u \in DTh(SP)$. Then $[t] \not\equiv^A [u]$. By Lemma 4.6, $\not\equiv_s^A$ is the least relation on A_s^2 that satisfies the inequality axioms of SP. Suppose that the complement $\overline{\equiv^A}$ of $\equiv$ w.r.t. A also satisfies the inequality axioms of SP. Then $[t] \not\equiv^A [u]$ implies $[t]\overline{\equiv^A}[u]$. Hence $([t],[u]) \not\in \equiv^A$ and thus $t \equiv u \not\in DTh(SP)$. Hence it remains to show that $\overline{\equiv^A}$ satisfies the inequality axioms of SP.

$\overline{\equiv^A}$ satisfies I1 (cf. Def. 2.3): Let $f : w \to s$ be a constructor and $t \in T_{\Sigma,w}$ such that $f^A([t]) \equiv^A f^A([u])$. By Prop. 4.4, $f^A([t]) = f^A([u])$, i.e. $[f(t)] = [f(u)]$ and thus $f(t) \equiv f(u) \in DTh(SP)$. Hence $f(nf(t)) \equiv f(nf(u)) \in DTh(SP)$. Since both sides of the last equation are normal forms and SP is consistent, they are equal. Hence $nf(t) = nf(u)$ and we proceed backwards: $nf(t) = nf(u)$ implies $t \equiv u \in DTh(SP)$ and thus $[t] = [u]$. By Prop. 4.4, $[t] \equiv^A [u]$.

$\overline{\equiv^A}$ satisfies I2 (cf. Def. 2.3): Let $f : v \to s$ and $g : w \to s'$ be different constructors, $t \in T_{\Sigma,v}$ and $u \in T_{\Sigma,w}$ such that $f^A([t]) \equiv^A g^A([u])$. By Prop. 4.4, $f^A([t]) = g^A([u])$, i.e. $[f(t)] = [g(u)]$ and thus $f(t) \equiv g(u) \in DTh(SP)$. Hence $f(nf(t)) \equiv g(nf(u)) \in DTh(SP)$. Since both sides of the last equation are normal forms and SP is consistent, they are equal, which contradicts $f \neq g$. ❑

Corollary 4.14 *Suppose that SP is functional or does not contain strong constructors. For all hidden sorts $s \in \Sigma$, $\equiv_{SP,s} \subseteq \sim_{SP,s}$.*

Proof. Let $A = Ini(SP)$. By a simple inductive argument, it is sufficient to consider case 2.3(5) and to show the statement for all hidden sorts $s \in \Sigma'$. It is easy to see that $\equiv$ solves all axioms of type C5, B1, B2 and B3 (cf. Def. 2.3) in $\sim$ with respect to A. We show that the same holds true for all axioms of type B4 if SP is functional.

Let $s \in \Sigma'$ be a hidden sort, $f : w \to s \in \Sigma'$ be a strong s-constructor and $t = (t_1, \ldots, t_n), u = (u_1, \ldots, u_n) \in T_{\Sigma,w}$. Then

$$\begin{array}{l} A \models f(t) \equiv f(u) \stackrel{4.5}{\Longleftrightarrow} f(t) \equiv f(u) \in DTh(SP) \stackrel{4.13}{\Longleftrightarrow} f(t) \not\equiv f(u) \not\in DTh(SP) \\ \stackrel{4.5}{\Longleftrightarrow} A \not\models f(t) \not\equiv f(u) \stackrel{I1}{\Longrightarrow} \forall\, 1 \leq i \leq n : A \not\models f(t_i) \not\equiv f(u_i) \\ \stackrel{4.5}{\Longleftrightarrow} \forall\, 1 \leq i \leq n : f(t_i) \not\equiv f(u_i) \not\in DTh(SP) \stackrel{4.13}{\Longleftrightarrow} \forall\, 1 \leq i \leq n : f(t_i) \equiv f(u_i) \in DTh(SP) \\ \stackrel{4.5}{\Longleftrightarrow} \forall\, 1 \leq i \leq n : A \models f(t_i) \equiv f(u_i). \end{array}$$

Let $g : v \to s \in \Sigma'$ be a strong s-constructor with $g \neq f$ and $u \in T_{\Sigma,v}$,

$$\begin{array}{l} A \models f(t) \equiv g(u) \stackrel{4.5}{\Longleftrightarrow} f(t) \equiv g(u) \in DTh(SP) \stackrel{4.13}{\Longleftrightarrow} f(t) \not\equiv g(u) \not\in DTh(SP) \\ \stackrel{4.5}{\Longleftrightarrow} A \not\models f(t) \not\equiv g(u) \stackrel{I2}{\Longrightarrow} A \not\models True. \end{array}$$

Hence for all hidden sorts $s \in \Sigma'$, $\equiv_s^A$ is a relation on $T_{\Sigma,ss}/\equiv_{SP}$ that satisfies the axioms of SP for $\sim_s$. Since by Lemma 4.7, $\sim_s^A$ is the greatest relation with this property, we conclude $\equiv_{SP,s} \subseteq \sim_{SP,s}$. ❑

Theorem 4.15 *Let $SP = (\Sigma, AX)$ be a functional ST with base type $baseSP = (base\Sigma, baseAX)$. $Ini(SP)$ is a canonical SP-model that is initial in $RModEq(SP)$.*

Proof. Let $A = Ini(SP)$. By Prop. 4.4, A is a structure with equality. By Prop. 4.12, A is reachable. By Lemma 4.13, A is a structure with inequality. By induction hypothesis, $A|_{base\Sigma}$ is a canonical $baseSP$-model. Hence 3.5(1) holds true. Let case 2.3(1/3/4) be valid. By Lemma 4.6, 3.5(2) holds true. Let case 2.3(2) be valid. By Lemma 4.8, 3.5(3) holds true. Let case 2.3(5/6) be valid. By Lemma 4.7, 3.5(4) holds true. This completes the proof that A is a canonical SP-model. It remains to show that A is initial in $RModEq(SP)$.

Let B be a reachable SP-model with equality. Define a mapping $h : A \to B$ by $h([t]) = t^B$ for all $t \in T_\Sigma$. By Lemma 4.9, $t \equiv u \in DTh(SP)$ implies $B \models t \equiv u$ and thus $t^B = u^B$ because B is a structure with equality. Hence h is well-defined.

h is a Σ-homomorphism: Let $f : w \to s \in \Sigma$ and $t \in T_{\Sigma,w}$. Then

$$h(f^A([t])) = h([f(t)]) = f(t)^B = f^B(t^B) = f^B(h([t])). \tag{1}$$

Let $r : w \in \Sigma$ be a predicate and $b \in h(r^A)$. Then $h([t]) = b$ for some $[t] \in r^A$. Hence $r(t) \in DTh(SP)$. By Lemma 4.9, $B \models r(t)$, i.e. $b = h([t]) = t^B \in r^B$. Let $r : w \in \Sigma$ be a copredicate and $[t] \in h^{-1}(r^B)$. Then $t^B = h([t]) \in r^B$ and thus $B \models r(t)$. By Lemma 4.10, $r(t) \in DTh(SP)$, i.e. $[t] \in r^A$.

The uniqueness of h follows from the uniqueness of the initial homomorphism $eval^B : T_\Sigma \to B$ that maps $t \in T_\Sigma$ to t^B (induction on t): Let $nat : T_\Sigma \to A$ be the natural homomorphism sending terms to their SP-equivalence classes. Given a homomorphism $h' : A \to B$, both $h' \circ nat$ and $h \circ nat$ are homomorphisms. Hence the uniqueness of $eval^B$ implies $h' \circ nat = eval^B = h \circ nat$ and thus $h' = h$ because nat is surjective. ❑

Criteria for completeness and consisteny and conditions under which deductive theorems can be proved by term rewriting are presented in Section 6.

5 On behavioral equality, constructors, observers and inference rules

Behaviorally consistent swinging types satisfy a "Hennessy-Milner property":

Theorem 5.1 ([37], Thm. 3.8) *Let $SP = (\Sigma, AX)$ be a behaviorally consistent ST, A be the initial SP-model and φ be a weakly modal formula with output Y and $b, c : X \to A$. Then $b \sim_{SP} c$ and $A \models_b \varphi$ imply $A \models_{c'} \varphi$ for some c' with $b \sim_{SP} c' =_Y c$. In particular, $b \sim_{SP} c$ iff for all poly-modal formulas φ, $A \models_b \varphi$ iff $A \models_c \varphi$.*

Proof. By assumption, $\sim^A = \sim_{SP}$ is a weak congruence on A. Hence the statement follows from [37], Theorem 3.8 (2). ❑

By Def. 3.1, behavioral consistency of SP does not require that behavioral SP-equivalence is compatible with structural SP-equivalence. If this were the case, we would obtain

$$\sim_{SP} \subseteq \sim_{SP} \circ \equiv_{SP} \circ \sim_{SP} \subseteq \equiv_{SP} .$$

By Corollary 4.14, the inverse inclusion $\equiv_{SP} \subseteq \sim_{SP}$ holds true if SP is functional. Hence, in this case, the compatibility of behavioral with structural equivalence would imply that both relations coincide, which is not intended, unless all constructors are strong (see Lemma 5.5).

Definition 5.2 SP is **head complete** if each ground Σ-term of hidden sort is SP-equivalent to a Σ-term whose leftmost symbol is a strong constructor (cf. Def. 2.3). ❑

Lemma 5.3 *Let $SP = (\Sigma, AX)$ be a functional and head complete ST. Then B4 (cf. Def. 2.3) is equivalent to: for all strong constructors $f : w \to s$,*

$$\begin{aligned} x \sim_s y \ &\Rightarrow\ (x \equiv_s f(x') \ \Rightarrow\ \exists\, y'(y \equiv_s f(y') \wedge x' \sim_w y')), \\ x \sim_s y \ &\Rightarrow\ (y \equiv_s f(y') \ \Rightarrow\ \exists\, x'(x \equiv_s f(x') \wedge x' \sim_w y')). \end{aligned} \tag{1}$$

Proof. "$\Rightarrow$": Let $t \sim_{SP} u$. Since SP is head complete, there are strong constructors f, g and term tuples t', u' such that $t \equiv_{SP} f(t')$ and $u \equiv_{SP} g(u')$. Since SP is functional, $\equiv_{SP}$ solves the behavior axioms of SP in $\sim$. Hence $f(t') \sim_{SP} g(u')$ because $\sim_{SP}$ is symmetric, transitive and, by Corollary 4.14, includes $\equiv_{SP}$. B4 implies $f = g$ and thus $t' \sim_{SP} u'$.

"$\Leftarrow$": Let $f(t) \sim_{SP} f(u)$. By (1), there is u' such that $f(u) \equiv_{SP} f(u')$ and $t \sim_{SP} u'$. Since SP is functional, $u \equiv_{SP} u'$. Hence $t \sim_{SP} u$ because $\sim_{SP}$ is an equivalence relation that includes $\equiv_{SP}$. Let $f(t) \sim_{SP} g(u)$. By (1), there is u' such that $g(u) \equiv_{SP} f(u')$. This contradicts the completeness and consistency of SP. ❏

Lemma 5.4 *Let $SP = (\Sigma, AX)$ be a functional and head complete ST, t be a strong normal form, $\sigma : X \to NF_\Sigma$ and $u \in NF_\Sigma$ such that $t\sigma \sim_{SP} u$. Then $u = t\tau$ and $\sigma \sim_{SP} \tau$ for some $\tau : X \to NF_\Sigma$. In particular, if t is a ground term, then $t \sim_{SP} u$ implies $t = u$.*

Proof. Let $t\sigma \sim_{SP} u$. By repeated applications of Lemma 5.3, there is $\vartheta : X \to T_\Sigma$ such that $u \equiv_{SP} t\vartheta$ and $\sigma \sim_{SP} \vartheta$. Since SP is complete, $\vartheta \equiv_{SP} \tau$ for some $\tau : X \to NF_\Sigma$. Hence $u \equiv_{SP} t\tau$ and $\sigma \sim_{SP} \tau$. Since u and $t\tau$ are normal forms and SP is consistent, both terms are equal. ❏

Lemma 5.5 *Let $SP = (\Sigma, AX)$ be a functional and head complete ST and s be a hidden sort of Σ such that all s-constructors of Σ are strong constructors. Then $\equiv_{SP,s} = \sim_{SP,s}$.*

Proof. Let $t, t' \in T_{\Sigma,s}$ such that $t \sim_{SP} t'$. Since SP is complete, there are $u, u' \in NF_{\Sigma,s}$ such that $t \equiv_{SP} u$ and $t' \equiv_{SP} u'$. Hence $u \sim_{SP} u'$ because $\sim_{SP}$ is symmetric, transitive and, by Corollary 4.14, includes $\equiv_{SP}$. Since u and u' are ground strong normal forms, Lemma 5.4 implies $u = u'$ and thus $t \equiv_{SP} t'$. ❏

If all s-constructors are strong constructors, observers are superfluous because they do not affect behavioral equivalence: Let SP and SP' be functional STs with the same signature such that all s-constructors are strong constructors and SP differs from SP' only with respect to the symbols declared as observers. Then both specifications have the same structural equivalence and thus, under the assumptions of Lemma 5.5, they also have the same behavioral equivalence.

Strong s-constructors induce a destructor

$$d_s : s \to \coprod_{f:w \to s \text{ is a strong constructor}} w$$

specified by an axiom $d_s(f(x)) \equiv \kappa_f(x)$ for each strong s-constructor f. d_s is the identity if s is a sum sort and thus all strong s-constructors are injections.

Since behavioral equivalence is a transitive relation that includes structural equivalence,

$$x \sim_s y \quad \Rightarrow \quad d_s(x) \sim d_s(y). \tag{2}$$

is equivalent to:

$$\begin{aligned} x \sim_s y \ &\Rightarrow \ (d_s(x) \equiv x' \ \Rightarrow \ \exists\, y'(d_s(y) \equiv y' \wedge x' \sim y')), \\ x \sim_s y \ &\Rightarrow \ (d_s(y) \equiv y' \ \Rightarrow \ \exists\, x'(d_s(x) \equiv x' \wedge x' \sim y')). \end{aligned} \tag{3}$$

(3) is equivalent to the conjunction of (4) over all strong s-constructors $f : w \to s$:

$$\begin{aligned} x \sim_s y \ &\Rightarrow \ (d_s(x) \equiv \kappa_f(x') \ \Rightarrow \ \exists\, y'(d_s(y) \equiv \kappa_f(y') \wedge x' \sim y')), \\ x \sim_s y \ &\Rightarrow \ (d_s(y) \equiv \kappa_f(y') \ \Rightarrow \ \exists\, x'(d_s(x) \equiv \kappa_f(x') \wedge x' \sim y')). \end{aligned} \tag{5}$$

Since (4) is equivalent to (1), we conclude that (1) is equivalent to (2).

Let SP be behaviorally consistent. Then the following inference rules for decomposing resp. removing "behavioral equations" are sound:

strong constructor elimination $\dfrac{f(t_1,\dots,t_n) \sim f(u_1,\dots,u_n)}{t_1 \sim u_1 \wedge \dots \wedge t_n \sim u_n} \Updownarrow$ if f is a strong constructor

strong constructor clash $\dfrac{f(t) \sim g(u)}{\mathit{False}} \Updownarrow$ if f and g are different strong constructors

Strong constructor elimination and clash are the behavioral-equality counterparts of constructor elimination and clash that allow us to remove constructors or equations whenever SP is functional (cf. [37], Section 4). The $\Uparrow$-direction of strong constructor elimination immediately follows from behavioral consistency. The $\Downarrow$-direction of strong constructor elimination, called **inverse compatibility** of $\sim$ with f, and the $\Downarrow$-direction of behavioral clash coincide with B4 (cf. Def. 2.3).

Note the similarity between B3 and (1). These pairs of formulas express that behavioral equality is zigzag compatible with δ and $\lambda(x,x').x \equiv f(x')$, respectively. Moreover, the equivalence between (1) and B4 reveals a duality between compatibility and zigzag compatibility like the one between category-theoretic notions of congruence (= compatibility) and bisimulation (= zigzag compatibility) (cf. [46, 18]): Behavioral equivalence is compatible with $f : w \to s$ iff it is zigzag compatible with $\lambda(x,x').f(x) \equiv x'$, and, by Lemma 5.3, it is inverse compatible with a strong constructor $f : w \to s$ iff it is zigzag compatible with $\lambda(x,x').x' \equiv f(x)$.

Fig. 1 associates the symbols of a swinging type with their compatibility properties. Those of the equivalence relations can be summarized as follows: structural equivalence is

- *compatible* with functions and relations,
- *inverse compatible* with constructors,

while behavioral equivalence is

- *compatible* with functions and local relations,
- *zigzag compatible* with transition relations,
- *inverse compatible* with strong constructors.

Coinductivity, functionality and continuity remain criteria for behavioral consistency. In particular, the behavior axioms for strong constructors entail that these functions build up only strong normal forms. These are required at certain places in coinductive axioms. Section 7 provides the details about coinductivity and its consequences if the respective ST has strong constructors. For examples with strong constructors and coinductive axioms, see [37], Sections 1.1, 1.2.1, 4.2, 4.5.

Strong constructor elimination and clash are the "behavioral versions" of constructor elimination and clash, which are indispensible simplification rules in almost every computation or proof based on a constructor-based specification: Let SP be a functional ST and f, g be different constructors of SP.

elimination of f $\dfrac{f(t_1,\dots,t_n) \equiv f(u_1,\dots,u_n)}{t_1 \equiv u_1 \wedge \dots \wedge t_n \equiv u_n} \Updownarrow$ $\dfrac{f(t_1,\dots,t_n) \not\equiv f(u_1,\dots,u_n)}{t_1 \not\equiv u_1 \vee \dots \vee t_n \not\equiv u_n} \Updownarrow$

clash of f and g $\dfrac{f(t) \equiv g(u)}{\mathit{False}} \Updownarrow$ $\dfrac{f(t) \not\equiv g(u)}{\mathit{True}} \Updownarrow$

Constructor elimination and clash belong to the class of **simplification rules**, which transform formulas into equivalent ones (indicated by the $\Updownarrow$ arrow) by (partially) evaluating logical operators and

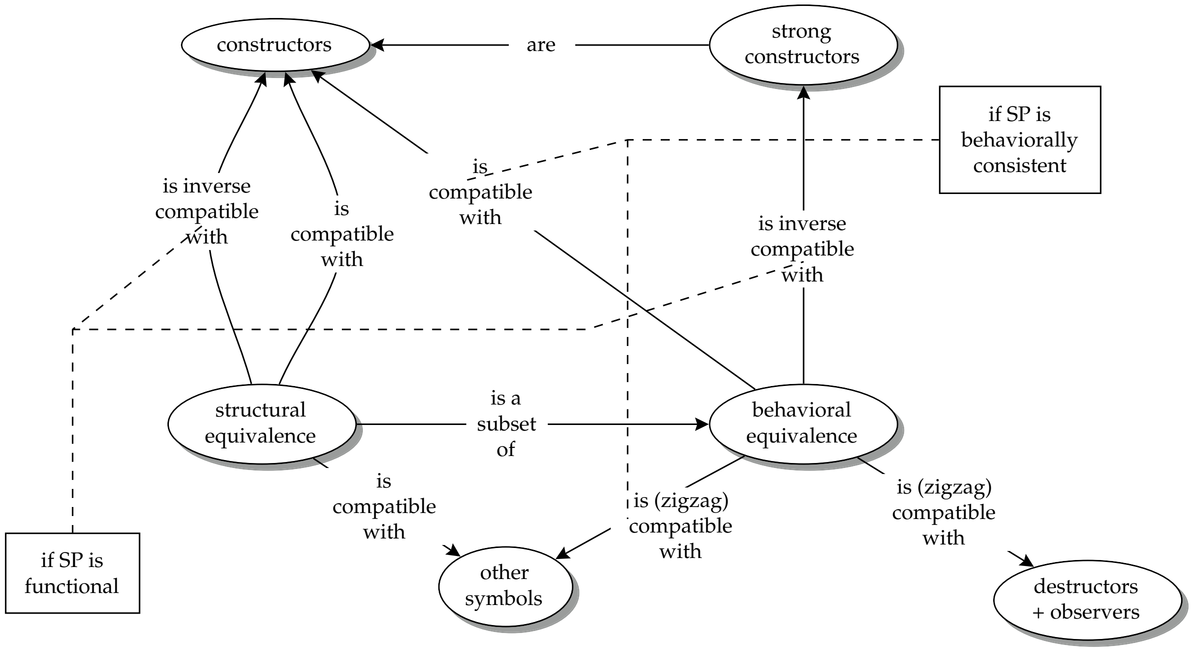

Figure 1. *Compatibilities of functions and relations.*

certain standard relations or functions. The equivalence between the antecedent and the succedent of a simplification rule holds at least with respect to the initial model.

Constructor elimination and clash reduce the number or size of equations or inequations, i.e. partially evaluate equality resp. inequality relations. Other relations are evaluated by **narrowing**, i.e. applying all axioms for the relations in parallel.

Let p be a predicate, q be a copredicate and f be a defined function of SP and t be a tuple of Σ-terms. For its more efficient use, the premise resp. conclusion of a Horn resp. co-Horn clause is divided into a *guard* γ and a "real" premise resp. conclusion. Semantically, $\gamma \Rightarrow (p(u) \Longleftarrow \varphi)$ coincides with $p(u) \Longleftarrow (\gamma \wedge \varphi)$ and $\gamma \Rightarrow (q(u) \Longrightarrow \varphi)$ with $q(u) \Longrightarrow (\gamma \Rightarrow \varphi)$.

narrowing on p

$$\frac{p(t)}{\bigvee_{i=1}^{k} \exists Z_i : (\varphi_i\sigma_i \wedge \vec{x} = \vec{x}\sigma_i)} \Uparrow$$

where $\gamma_1 \Rightarrow (p(u_1) \Longleftarrow \varphi_1), \dots, \gamma_n \Rightarrow (p(u_n) \Longleftarrow \varphi_n)$ are the (Horn) axioms for p,

$(*)$ $\vec{x}$ is a list of the variables of t,
for all $1 \leq i \leq k$, $t\sigma_i = u_i\sigma_i$, $\gamma_i\sigma_i \vdash \mathit{True}$ and $Z_i = var(u_i, \varphi_i)$,
for all $k < i \leq n$, t is not unifiable with u_i.

narrowing upon q

$$\frac{q(t)}{\bigwedge_{i=1}^{k} \forall Z_i : (\vec{x} = \vec{x}\sigma_i \Rightarrow \varphi_i\sigma_i)} \Uparrow$$

where $\gamma_1 \Rightarrow (q(u_1) \Longrightarrow \varphi_1), \dots, \gamma_n \Rightarrow (q(u_n) \Longrightarrow \varphi_n)$ are the (co-Horn) axioms for q and $(*)$ holds true.

narrowing upon f

$$\frac{\varphi(f(t))}{\begin{array}{c}\bigvee_{i=1}^{k} \exists Z_i : (\varphi(v_i\sigma_i) \wedge \varphi_i\sigma_i \wedge \vec{x} = \vec{x}\sigma_i) \vee \\ \bigvee_{i=k+1}^{l} (\varphi(f(t\sigma_i)) \wedge \vec{x} = \vec{x}\sigma_i)\end{array}} \Uparrow$$

where $\gamma_1 \Rightarrow (f(u_1) = v_1) \Longleftarrow \varphi_1, \ldots, \gamma_n \Rightarrow (f(u_n) = v_n) \Longleftarrow \varphi_n$ are the (Horn) axioms for f,

(∗) $\vec{x}$ is a list of the variables of t,
for all $1 \le i \le k$, $t\sigma_i = u_i\sigma_i$, $\gamma_i\sigma_i \vdash \mathit{True}$ and $Z_i = var(u_i, \varphi_i)$,
for all $k < i \le l$, σ_i is a partial unifier of t and u_i,
for all $l < i \le n$, t is not partially unifiable with u_i.

If SP is functional, structural equalities and inequalities can be removed by the above clash rules. The same condition ensures that the following rules are sound with respect to the initial model:

elimination of p $\qquad \dfrac{p(t)}{\mathit{False}} \Updownarrow$

where $t \in NF_\Sigma(X)$ and for all axioms $p(u) \Leftarrow \varphi$ of SP, t and u are not unifiable

elimination of q $\qquad \dfrac{q(t)}{\mathit{True}} \Updownarrow$

where $t \in NF_\Sigma(X)$ and for all axioms $q(u) \Rightarrow \varphi$ of SP, t and u are not unifiable

Neither the constructor rules nor unfolding removes equations or inequations with a variable on one side. Simplification rules doing this job are the following ones:

elimination/introduction of x

$$\frac{\forall x : ((x \equiv t \wedge \varphi(x)) \Rightarrow \psi(x))}{\varphi(t) \Rightarrow \psi(t)} \Updownarrow \qquad \frac{\forall x : (\varphi(x) \Rightarrow (x \equiv t \wedge \psi(x)))}{\varphi(t) \Rightarrow \psi(t)} \Updownarrow$$

$$\frac{\exists x : (x \equiv t \wedge \varphi(x))}{\varphi(t)} \Updownarrow \qquad \frac{\forall x : (x \not\equiv t \vee \varphi(x))}{\varphi(t)} \Updownarrow \quad \text{if } x \notin var(t)$$

Instead of generating a complete case analysis and applying all axioms for a relation or function in parallel, axioms that would introduce unsolvable equations need not be applied. The implementation of narrowing in the interactive theorem prover Expander2 [40] take this into account by restricting the above narrowing rules to those summands resp. factors of the rule succedent that are not "syntactically" unsolvable in the initial model. This holds true, for instance, if SP is functional and the leading symbols of t resp. t_i (see above) are different constructors). More general unsolvability checks require subproofs: a formula is unsolvable if it can be transformed into *False* via constructor rules, narrowing, variable elimination and other simplifications. For increasing the efficiency of narrowing, its actual implementation in Expander2 also employs **guarded** and **needed** narrowing (see [41, 40]).

If SP is functional, the above rules provide a *solution complete* calculus, more precisely: for all atoms p and $\sigma : X \to T_\Sigma$ such that $Ini(SP)$ satisfies $p\sigma$, σ can be derived from p by the above rules (cf., e.g., [32], Chapter 8; [33], Cor. 7.3; [34], Section 8).

The full completeness of the above rules (not only for deriving solutions) is not achieved whenever a function or relation r of SP has axioms with "recursive calls". Then unfolding may not be able to remove all occurrences of r from a given formula. This is the point where *induction rules* (Noetherian induction, fixpoint induction and coinduction) may be needed. Unfortunately, induction rules are not equivalence transformations. One may be forced to *generalize* a conjecture to be proved by (co)induction before submitting it to the rule. Fixpoint induction and coinduction are dual to each other. Fixpoint induction removes (an occurrence of) a defined function or predicate r, coinduction removes a copredicate q. The atom where r resp. q occurs must be the premise resp. conclusion of an implication: Let p, f, q be as above and AX_r be the axioms of SP for r.

fixpoint induction on p
$$\frac{r(x) \Rightarrow \psi(x)}{\bigwedge_{\varphi\in AX_p} \varphi[\psi(t)/p(t) \mid p(t) \text{ occurs in } \varphi]} \Uparrow$$

fixpoint induction on f
$$\frac{f(x) \equiv y \Rightarrow \psi(x,y)}{\bigwedge_{\varphi\in flat(AX_f)} \varphi[\psi(t,u)/(f(t) \equiv u) \mid f(t) \equiv u \text{ occurs in } \varphi]} \Uparrow$$ [8]

coinduction on q
$$\frac{\psi(x) \Rightarrow q(x)}{\bigwedge_{\varphi\in AX_q} \varphi[\psi(t)/q(t) \mid q(t) \text{ occurs in } \varphi]} \Uparrow$$

The soundness of fixpoint induction on predicates and coinduction follows directly from the interpretation of predicates resp. copredicates as the least resp. greatest solutions of their axioms. By Theorem 6.11, fixpoint induction on (a defined function) f is sound if SP is functional. Here the rule antecedent may in fact read as $f(x) \equiv y \Leftrightarrow \psi(x,y)$ because the functionality of SP implies that ψ is the *unique* solution of AX_f.

More details on rules and strategies for reasoning about STs can be found in [41].

While the correctness of unfolding follows from the interpretation of (co)predicates as fixpoints, the soundness of fixpoint induction and coinduction depends on the interpretation as *least* resp. *greatest* fixpoints. As mentioned above, induction rules are indispensable if the proof requires the (partial) evaluation of a function or relation that has axioms with "recursive calls". On the other hand, the fixpoints are unique if the axioms do not involve recursive calls. Hence, roughly said, no recursion means no induction means least fixpoint = greatest fixpoint means ... almost automatic proofs!

While a formula $r(x) \Rightarrow \varphi(x)$ amenable to fixpoint induction requires that the *predicate* r satisfies φ, a formula $\varphi(x) \Rightarrow q(x)$ to be transformed by coinduction tells us that the copredicate q holds true for all *objects* that satisfy φ. Hence the expansion of $\varphi(x) \Rightarrow q(x)$ is an *evaluation* of (instance of) q rather than a proof. If q were a predicate, then the expansion of $\varphi(x) \Rightarrow q(x)$ usually starts with evaluation steps, i.e. unfoldings of q. The same applies to an expansion of $q(x) \Rightarrow \varphi(x)$ if q is a copredicate, although this expansion amounts to a *proof* (of validity of φ four q).

Hence the four conjecture schemata are not only expandable by pairwise dual rules, the expansions also aim at dual goals: proofs versus evaluations (or solutions). The symmetries are illustrated in Fig. 2 where *narrowing* stands for unfoldings together with simplifications that are indispensible for achieving a *solved* formula, i.e. *True*, *False* or a set of (in)equations that represents a solution of the conjecture.

Narrowing and fixpoint induction/coinduction complement each other concerning the way axioms are related to conjectures: In the first case, axioms are applied to conjectures, and the proof proceeds by transforming the modified conjectures. In the second case, conjectures are applied to axioms and the proof proceeds by transforming the modified axioms.

6 Horn STs and the reductive calculus

Under certain assumptions (see Theorem 6.3), an ST can be turned into a Horn ST by translating co-Horn axioms into equivalent Horn clauses:

Definition 6.1 Let $SP = (\Sigma, AX)$ be a swinging type with base type $baseSP = (base\Sigma, baseAX)$ and extension (Σ', AX') such that case 2.3(5/6) holds true. Let

$$Horn(\Sigma) \;=\; base\Sigma \cup \{nat, 0 :\to nat, suc : nat \to nat\} \cup \{r_{loop} : nat \times w \mid r : w \in \Sigma'\},$$

[8] $flat(AX_f)$ is the set of *flattened* axioms for f (see Theorem 6.11).

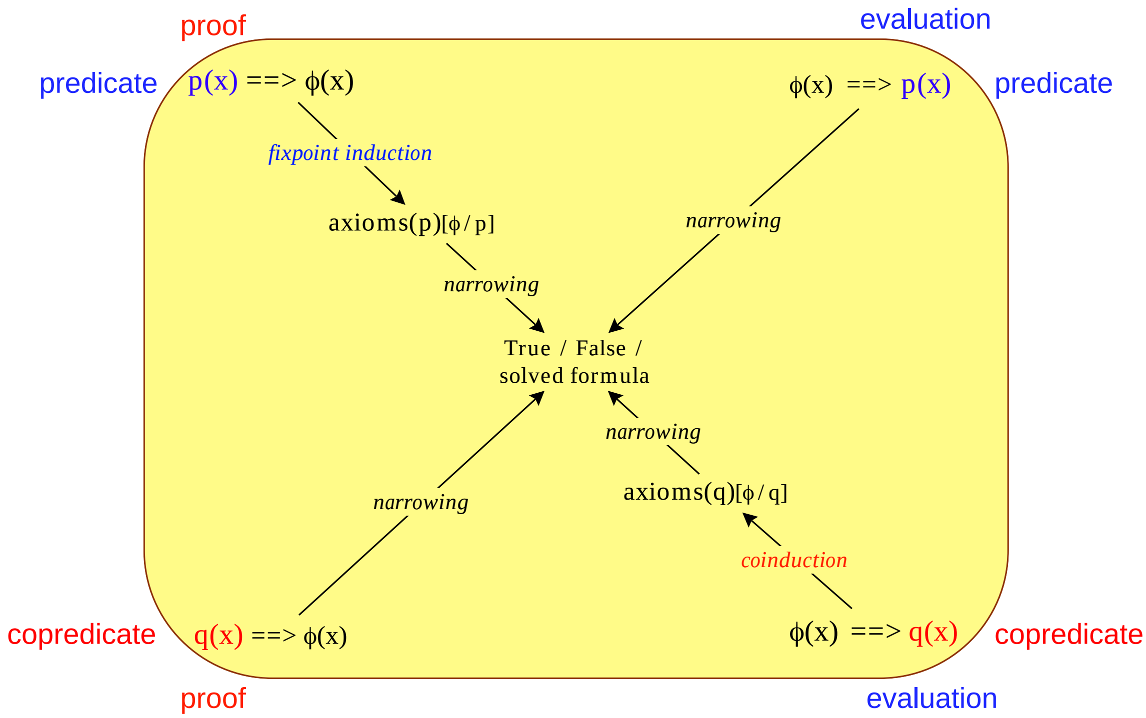

Figure 2. *Four types of conjectures and how they are proved/disproved/solved*

$n, x \in X \setminus freevar(AX')$ and $Horn(AX)$ consist of $baseAX$ and all Horn clauses

$$
\begin{array}{l}
r(x) \;\Leftarrow\; \forall n : r_{loop}(n,x) \\
r_{loop}(0,x) \\
r_{loop}(suc(n),x) \;\Leftarrow\; \bigwedge_{cl=(r(t)\Rightarrow\varphi)\in AX'} \forall freevar(cl) : (x \not\equiv t \vee \varphi)[q_{loop}(n,u)/q(u) \mid q(u) \in \varphi,\ q \in \Sigma']
\end{array}
$$

such that $r \in \Sigma'$. The Horn swinging type

$$Horn(SP) \;=\; (Horn(\Sigma), Horn(AX))$$

with base type $baseSP$ is called the **Horn version of** SP. The Horn version of the empty ST is the empty ST. ❏

Example 6.2 The following parameterized ST specifies countable sets of finite and infinite sequences (cf. [38], Section 4.2). For the parameter ENTRY, see Example **??**.

COLIST = ENTRY `then`

`vissorts`	*bool* *nat*
`hidsorts`	$colist = colist(entry)$
`constructs`	$0 :\rightarrow nat$
	$suc : nat \rightarrow nat$
	$true, false :\rightarrow bool$
	$nil :\rightarrow colist$
	$\_\&\_ : entry \times colist \rightarrow colist$
	$blink :\rightarrow colist(nat)$

	$nats : nat \to colist$
	$\_@\_ : colist \times colist \to colist$
`destructs`	$ht : colist \to 1 + (entry \times colist)$
`local preds`	$exists : (entry \to bool) \times colist$
`copreds`	$forall : (entry \to bool) \times colist$
	$fair : (entry \to bool) \times colist$
`vars`	$n : nat \quad x : entry \quad s, s', t : colist \quad g : entry \to bool$
`Horn axioms`	$ht(nil) \equiv ()$
	$ht(x\&s) \equiv (x, s)$
	$ht(blink) \equiv (0, suc(0)\&blink)$
	$ht(nats(n)) \equiv (n, nats(suc(n)))$
	$ht(s@s') \equiv ht(s') \quad \Leftarrow \quad ht(s) \equiv ()$
	$ht(s@s') \equiv (x, t@s') \quad \Leftarrow \quad ht(s) \equiv (x, t)$
	$exists(g, s) \quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge g(x) \equiv true$
	$exists(g, s) \quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge exists(g, t)$
`co-Horn axioms`	$forall(g, s) \quad \Rightarrow \quad ht(s) \not\equiv (x, t) \vee (g(x) \equiv true \wedge forall(g, t))$
	$fair(g, s) \quad \Rightarrow \quad ht(s) \not\equiv (x, t) \vee (exists(g, s) \wedge fair(g, t))$

The Horn version of COLIST reads as follows:

HCOLIST = ENTRY `then`

`vissorts`	$bool \quad nat$
`hidsorts`	$colist = colist(entry)$
`constructs`	$0 :\to nat$
	$suc : nat \to nat$
	$true, false :\to bool$
	$nil :\to colist$
	$\_\&\_ : entry \times colist \to colist$
	$blink :\to colist(nat)$
	$nats : nat \to colist$
	$\_@\_ : colist \times colist \to colist$
`destructs`	$ht : colist \to 1 + (entry \times colist)$
`local preds`	$exists, forall : (entry \to bool) \times colist$
	$fair : (entry \to bool) \times colist$
`vars`	$n : nat \quad x : entry \quad s, s', t : colist \quad g : entry \to bool$
`Horn axioms`	$ht(nil) \equiv ()$
	$ht(x\&s) \equiv (x, s)$
	$ht(blink) \equiv (0, suc(0)\&blink)$
	$ht(nats(n)) \equiv (n, nats(suc(n)))$
	$ht(s@s') \equiv ht(s') \quad \Leftarrow \quad ht(s) \equiv ()$
	$ht(s@s') \equiv (x, t@s') \quad \Leftarrow \quad ht(s) \equiv (x, t)$
	$exists(g, s) \quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge g(x) \equiv true$
	$exists(g, s) \quad \Leftarrow \quad ht(s) \equiv (x, t) \wedge exists(g, t)$
	$forall(g, s) \quad \Leftarrow \quad \forall n : forall_{loop}(n, g, s)$
	$forall_{loop}(0, g, s)$
	$forall_{loop}(suc(n), g, s) \quad \Leftarrow \quad \forall x, t : (ht(s) \not\equiv (x, t) \vee (g(x) \equiv true \wedge forall_{loop}(n, g, t)))$

$$\begin{array}{lll} fair(g,s) & \Leftarrow & \forall n : fair_{loop}(n,g,s) \\ fair_{loop}(0,g,s) & & \\ fair_{loop}(suc(n),g,s) & \Leftarrow & \forall x,t : (ht(s) \not\equiv (x,t) \vee (exists(g,s) \wedge fair(g,t))) \end{array}$$

Theorem 6.3 (elimination of copredicates preserves canonicity) *Given the assumptions of Def. 6.1, suppose that the SP-step function Φ on $A|_{base\Sigma}$ is downward continuous and the $Horn(SP)$-step function Ψ on $A|_{base\Sigma}$ is upward continuous. A Σ-structure A is a canonical SP-model iff there is a canonical $Horn(SP)$-model B with $B|_{base\Sigma} = A|_{base\Sigma}$.*

Proof. "$\Rightarrow$": Let A be a canonical SP-model. A $Horn(\Sigma)$-structure B is defined as follows: $B|_{base\Sigma} = A|_{\Sigma}$, $B_{nat} = \mathbb{N}$, $0^B = 0$, for all $n \in \mathbb{N}$, $suc^B(n) = n+1$, and for all copredicates $r \in \Sigma'$,

$$r^B_{loop} = \{(i,a) \mid a \in r^{\Phi^i(\top)},\ i \in \mathbb{N}\} \quad \text{and} \quad r^B = \{a \mid \forall\, i \in \mathbb{N} : (i,a) \in r^B_{loop}\}.$$

By induction on i, one shows that for all $i \in \mathbb{N}$ and $a \in A$,

$$a \in r^{\Phi^i(\top)} \iff (i,a) \in r^{\Psi^{i+1}(\bot)}_{loop}. \tag{1}$$

Hence by Prop. 3.6,

$$\begin{array}{c} r^A = r^{gfp(\Phi)} = \cap_{i\in\mathbb{N}} r^{\Phi^i(\top)} = \{a \mid \forall\, i \in \mathbb{N} : a \in r^{\Phi^i(\top)}\} = r^B, \\ (i,a) \in r^B_{loop} \iff a \in \cup_{i\in\mathbb{N}} r^{\Phi^i(\top)} \iff (i,a) \in \cup_{i\in\mathbb{N}} r^{\Psi^i(\bot)}_{loop} \iff (i,a) \in r^{lfp(\Psi)}_{loop} \end{array}$$

and thus

$$a \in r^B \iff \forall\, i \in \mathbb{N} : (i,a) \in r^B_{loop} \iff \forall\, i \in \mathbb{N} : (i,a) \in r^{lfp(\Psi)}_{loop} \iff a \in r^{lfp(\Psi)}.$$

We conclude that B is a canonical $Horn(SP)$-model with $B|_{\Sigma} = A|_{\Sigma}$.

"$\Leftarrow$": Let B be a canonical $Horn(SP)$-model B with $B|_{\Sigma} = A|_{\Sigma}$. Let r be a copredicate of Σ'. By (1) and Prop. 3.6,

$$\begin{array}{c} a \in r^B \iff a \in r^{lfp(\Psi)}_{loop} \iff \forall\, i \in \mathbb{N} : (i,a) \in r^{lfp(\Psi)}_{loop} \iff \forall\, i \in \mathbb{N} : (i,a) \in \cup_{i\in\mathbb{N}} r^{\Psi^i(\bot)}_{loop} \\ \iff \forall\, i \in \mathbb{N} : (i,a) \in r^{\Psi^{i+1}(\bot)}_{loop} \iff \forall\, i \in \mathbb{N} : a \in r^{\Phi^{i+1}(\top)} \iff a \in \cap_{i\in\mathbb{N}} r^{\Phi^i(\top)} \iff a \in r^{gfp(\Phi)}. \end{array}$$

Since $B|_{\Sigma} = A|_{\Sigma}$, we conclude that A is a canonical SP-model. ❏

The reductive calculus presented in this section provides the basis for criteria on the axioms of a swinging type that ensure its functionality. The reductive calculus makes use of the fact that any ST can be transformed into a semantically equivalent Horn ST (Theorem 6.3). By Prop. 4.12 and Lemma 4.13, the functionality of a swinging type does not depend on its copredicates. Hence functionality carries over from the Horn version to the original ST.

A **fresh variable** of a Horn clause $\varphi = (f(t)\{\equiv u\} \Leftarrow \vartheta)$ is a free variable of φ that occurs in u or ϑ, but not in t. **fresh**(φ) denotes the set of fresh variables of φ.

Definition 6.4 (reductive calculus) Let $SP = (\Sigma, AX)$ be a swinging type with base type $baseSP$ and extension (Σ', AX'). The **reductive calculus for** SP consists of the following rules for reducing (sets of) Σ-formulas. Let $\tau : X \to T_{\Sigma}(X)$ and p be a Σ-atom.

base $\quad \dfrac{\varphi}{True} \Uparrow \quad$ for all $\varphi \in RTh(baseSP)$

rewriting $\quad \dfrac{p[t\tau/x]}{p[u\tau/x] \wedge \vartheta\tau} \Uparrow \quad$ for all $x \in var(p)$, $\varphi = (t \equiv u \Leftarrow \vartheta) \in Horn(AX')$ and $fresh(\varphi)\tau \subseteq NF_{\Sigma}$

resolution $\dfrac{r(t\tau)}{\vartheta\tau} \Uparrow$ for all $r \neq \equiv$, $\varphi = (r(t) \Leftarrow \vartheta) \in Horn(AX')$ and $fresh(\varphi)\tau \subseteq NF_\Sigma$

reflection $\dfrac{t \equiv t}{True} \Uparrow$

$\vee$**-elimination** $\dfrac{\varphi \vee \psi}{\varphi} \Uparrow$

$\exists$**-elimination** $\dfrac{\exists x \varphi}{\varphi[t/x]} \Uparrow$ for all $t \in T_{\Sigma,sort(x)}$ and $x \in var(\varphi)$

$\wedge$**-elimination** $\dfrac{\varphi \wedge \psi}{\{\varphi, \psi\}} \Uparrow$

$\forall$**-elimination** $\dfrac{\forall x : \varphi}{\{\varphi[t/x] \mid t \in T_{\Sigma,sort(x)}\}} \Uparrow$ for all $x \in var(\varphi)$

A set F of Σ-formulas is SP**-reductive** if for each rule of the reductive calculus for SP, the premise belongs to F whenever the conclusions belong to F. $\mathbf{RTh}(SP)$ denotes the intersection of all SP-reductive sets containing *True*. Elements of $RTh(SP)$ are called the **reductive theorems of** SP.

Let a be an ordinal number. The **reductive inference relation** $\vdash^{r,a}_{SP}$ is inductively defined as follows:

- For all $\varphi \in RTh(baseSP)$, $\vdash^{r,a}_{SP} \varphi$.
- Let $\varphi_1, \ldots, \varphi_n$ be the conclusions and φ be the premise of the rewriting, resolution, reflection, $\wedge$-elimination or $\forall$-elimination rule. If $\vdash^{r,a_i}_{SP} \varphi_i$ and $a_i < a$ for all $1 \leq i \leq n$, then $\vdash^{r,a}_{SP} \varphi$.
- If $x \in var(\varphi)$ and for all $t \in NF_{\Sigma,sort(x)}$, $\vdash^{r,a_t}_{SP} \varphi[t/x]$ and $a_t < a$, then $\vdash^{r,a}_{SP} \forall x : \varphi$.

The **proof length** of φ in the reductive calculus for SP is the least ordinal number a such that $\vdash^{r,a}_{SP} \varphi$.

The SP**-rewrite relation** is the binary relation on Σ-terms and -formulas that is defined as follows:

$$t \longrightarrow_{SP} t' \iff_{def} \begin{cases} \exists\ \varphi = (l \equiv r \Leftarrow \vartheta) \in AX,\ \sigma : X \to T_\Sigma : l\sigma = t, \\ r\sigma = t',\ fresh(\varphi)\sigma \subseteq NF_\Sigma,\ \vartheta\sigma \in RTh(SP). \end{cases}$$

$$\varphi(t) \longrightarrow_{SP} \varphi(t') \iff_{def} t \longrightarrow_{SP} t'.$$

ψ is an SP**-reduct** of φ if $\varphi \stackrel{*}{\longrightarrow}_{SP} \psi$. $\varphi \in RTh(SP)$ is SP**-convergent** if all SP-reducts of φ are reductive theorems of SP. φ is SP**-reduced** if φ is the only SP-reduct of φ. Two terms are SP**-joinable** if they have a common SP-reduct. SP is **confluent** if for all ground terms t, each two SP-reducts of t are SP-joinable. SP is **strongly complete** if each ground Σ-term has an SP-reduct in NF_Σ. ❏

$RTh(SP)$ is nonempty, the least SP-reductive set and the set of all φ such that $\vdash^{r,a}_{SP} \varphi$ for some ordinal number a.

Lemma 6.5 *Let $SP = (\Sigma, AX)$ be an ST. For all $t, u \in T_\Sigma$,*

- $t \stackrel{*}{\longrightarrow}_{SP} u$ *implies* $(t \equiv u) \in RTh(SP)$*,*
- $\varphi(t) \stackrel{*}{\longrightarrow}_{SP} \varphi(u) \in RTh(SP)$ *implies* $\varphi(t) \in RTh(SP)$*,*
- $(t \equiv u) \in RTh(SP)$ *iff t and u are SP-joinable,*
- $RTh(SP) \subseteq DTh(SP)$*,*
- *if* $\varphi \in RTh(SP)$ *and* $\varphi \stackrel{*}{\longrightarrow}_{SP} \psi$*, then* $\psi \in DTh(SP)$*.*

Proof. It is easy to see that $DTh(SP)$ is SP-reductive. Hence $RTh(SP)$ is a subset of $DTh(SP)$. The other properties can be shown by induction on the length of $\varphi(t) \xrightarrow{*}_{SP} \varphi(u)$. ❑

Lemma 6.6 *An ST SP is confluent iff all closed reductive theorems of SP are SP-convergent.*

Proof. "$\Leftarrow$": Let u, u' be SP-reducts of a ground term t. Since $t \xrightarrow{*}_{SP} u$ and $(u \equiv u) \in RTh(SP)$, Lemma 6.5 implies $(t \equiv u) \in RTh(SP)$. Hence $(u' \equiv u) \in RTh(SP)$ because $t \xrightarrow{*}_{SP} u'$ and reductive theorems are convergent. By Lemma 6.5, u and u' are joinable.

"$\Rightarrow$": We show that the set F of SP-convergent formulas is SP-reductive. Then we can conclude that F contains $RTh(SP)$ and the proof is complete. Let (Σ, AX) be the Horn version of SP.

Let $p[t\sigma/x]$ and $p[u\sigma/x] \wedge \vartheta\sigma$ be the premise resp. conclusion of a rewriting rule instance such that $p[u\sigma/x] \wedge \vartheta\sigma \in F$ and $p[t\sigma/x] \xrightarrow{*}_{SP} r(v)$ for some predicate r and ground term v. Then $p[t\sigma/x] = r(t')$, $p[u\sigma/x] = r(u')$, $t' \xrightarrow{*}_{SP} v$ and $t' \longrightarrow_{SP} u'$ for some ground terms t', u'. Since SP is confluent, u' and v have a common SP-reduct, say v'. Since $r(u') \in F$, $r(v') \in RTh(SP)$ and thus $r(v) \in RTh(SP)$ by Lemma 6.5. Therefore, the premise $p[t\sigma/x] = r(t')$ of the rewriting rule instance is in F.

Let $r(t\sigma)$ and $\vartheta\sigma$ be the premise resp. conclusion of a resolution rule instance such that $\vartheta\sigma \in F$ and $r(t\sigma) \xrightarrow{*}_{SP} r(u)$ for some u. Then $t\sigma = (t_1, \ldots, t_n)$, $u = (u_1, \ldots, u_n)$ for some $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ such that for all $1 \leq i \leq n$, $t_i\sigma \xrightarrow{*}_{SP} u_i$. Since $r(t) \Leftarrow \vartheta$ is an axiom, t is a normal form. Hence there are a linear term tuple $v = (v_1, \ldots, v_n)$, a variable renaming ρ and a substitution τ such that $v\rho = t$, $v\tau = u$ and for all $x \in var(v)$, $x\rho\sigma \xrightarrow{*}_{SP} x\tau$. Let $x, y \in X$ such that $x\rho = y\rho$. Then $x\tau \xleftarrow{*}_{SP} x\rho\sigma = y\rho\sigma \xrightarrow{*}_{SP} y\tau$ implies $x\tau \xrightarrow{*}_{SP} x\rho\tau' \xleftarrow{*}_{SP} y\tau$ for some $\tau' : X \rightarrow T_\Sigma$ because SP is confluent. Hence for all $x \in X$, $x\rho\sigma \xrightarrow{*}_{SP} x\tau \xrightarrow{*}_{SP} x\rho\tau'$, and thus $var(t) = var(v\rho) \subseteq var(\rho(X))$ implies $x\sigma \xrightarrow{*}_{SP} x\tau'$ for all $x \in var(t)$. Define σ' by $x\sigma' = x\tau'$ if $x \in var(t)$ and $x\sigma' = x\sigma$ otherwise. Then $\vartheta\sigma \xrightarrow{*}_{SP} \vartheta\sigma'$ and thus $\vartheta\sigma' \in RTh(SP)$ because $\vartheta\sigma$ is SP-convergent. Hence $r(t\sigma') \in RTh(SP)$. Since $u = v\tau \xrightarrow{*}_{SP} v\rho\tau' = t\tau' = t\sigma'$, Lemma 6.5 implies $r(u) \in RTh(SP)$. Therefore, the premise $r(t\sigma)$ of the resolution rule instance is in F.

Let $t \equiv t$ be the premise of a reflection rule instance and $t \xrightarrow{*}_{SP} u$. Since $u \equiv u \in RTh(SP)$ and $u \equiv t \xrightarrow{*}_{SP} u \equiv u$, Lemma 6.5 implies $u \equiv t \in RTh(SP)$. Therefore, the premise $t \equiv t$ of the reflection rule instance is in F.

It is easy to see that the premise of each $\wedge$- or $\forall$-elimination rule instance is in F whenever the conclusion is in F. ❑

Definition 6.7 (reductive model) Let $SP = (\Sigma, AX)$ be an ST with base type $baseSP = (base\Sigma, baseAX)$ and extension (Σ', AX').

Let $\Sigma' = (S, F, LR, TR)$. The **reductive** SP**-model** $Red(SP)$ is the reachable Σ-structure A that is inductively defined as follows: If SP is the empty ST, then A is the empty Σ-structure. Otherwise $A|_{base\Sigma} = Ini(baseSP)$,

- ➢ for all $s \in S$, $A_s = T_{\Sigma,s}/{\equiv_{SP}}$,
- ➢ for all $f : w \rightarrow s \in F$ and $t \in T_{\Sigma,w}$, $f^A([t]) = [f(t)]$,
- ➢ for all $r : w \in LR \cup TR$, $r^A = \{[t] \in A_w \mid r(t) \in RTh(SP)\}$. ❑

Theorem 6.8 (Church-Rosser Theorem) *Let $SP = (\Sigma, AX)$ be a complete Horn ST. SP is confluent iff for all closed and positive deductive theorems of SP are reductive theorems of SP.*

Proof. Suppose that SP is functional. We show that $Red(SP)$ satisfies AX. Let $(p \Leftarrow \vartheta) \in AX$ and $\sigma : X \to T_\Sigma$ such that $\vartheta\sigma$ is a reductive theorem of SP. Since SP is complete, there is $\tau : X \to NF_\Sigma$ such that for all $x \in X$, $x\sigma \equiv_{SP} x\tau$ and thus $x\sigma \stackrel{*}{\longrightarrow}_{SP} x\tau$ because SP is confluent and normal forms are SP-reduced. Hence by Lemma 6.6, $\vartheta\sigma \in RTh(SP)$ implies $\vartheta\tau \in RTh(SP)$. It remains to show that $p\tau$ and thus $p\sigma$ are reductive theorems of SP.

Let $(p \Leftarrow \vartheta) \in AX$. If $p = (t \equiv u)$, then $\vartheta\tau \in RTh(SP)$ implies $(\vartheta\tau \wedge u\tau \equiv u\tau) \in RTh(SP)$ and thus $p\tau \in RTh(SP)$. If $p = r(t)$, then $\vartheta\tau \in RTh(SP)$ implies $p\tau \in RTh(SP)$.

Let $(p \Leftarrow \vartheta) \in EQ_\Sigma$. If $(p \Leftarrow \vartheta) = (x \equiv x)$, then $p\tau \in RTh(SP)$. If $(p \Leftarrow \vartheta) = (y \equiv x \Leftarrow x \equiv y)$, then by Lemma 6.5, $\vartheta\tau \in RTh(SP)$ implies that $x\tau$ and $y\tau$ are SP-joinable. Hence by Lemma 6.5, $p\tau \in RTh(SP)$. If $(p \Leftarrow \vartheta) = (f(x_1,\dots,x_n) \equiv f(y_1,\dots,y_n) \Leftarrow \wedge_{i=1}^n x_i \equiv y_i)$, then by Lemma 6.5, $\vartheta\tau \in RTh(SP)$ implies that $x_i\tau$ and $y_i\tau$ have a common SP-reduct, say t_i. Hence $f(t_1,\dots,t_n)$ is a common SP-reduct of $f(x_1,\dots,x_n)\tau$ and $f(y_1,\dots,y_n)$ and thus by Lemma 6.5, $p\tau \in RTh(SP)$. If $(p \Leftarrow \vartheta) = (r(x_1,\dots,x_n) \Leftarrow r(y_1,\dots,y_n) \wedge \wedge_{i=1}^n x_i \equiv y_i)$, then by Lemma 6.5, $\vartheta\tau \in RTh(SP)$ implies that $x_i\tau$ and $y_i\tau$ have a common SP-reduct, say t_i. Hence by Lemma 6.6, $r(y_1,\dots,y_n)\tau \in RTh(SP)$ implies $r(t_1,\dots,t_n)\tau \in RTh(SP)$, and thus by Lemma 6.5, $p\tau \in RTh(SP)$.

We have shown that $Red(SP)$ satisfies AX. Hence by Thm. 4.15, for all predicates $r \in \Sigma$, $r^{Ini(SP)}$ is a subset of $r^{Red(SP)}$. We conclude that all closed and positive deductive theorems of SP are reductive theorems of SP. Conversely, suppose that the latter holds true. By Lemma 6.6, SP is confluent if all reductive theorems of SP are SP-convergent. Let $\varphi \in RTh(SP)$ and $\varphi \stackrel{*}{\longrightarrow}_{SP} \psi$. By Lemma 6.5, $\psi \in DTh(SP)$. W.l.o.g. ψ is closed. Hence by assumption, $\psi \in RTh(SP)$. ❏

Corollary 6.9 *A strongly complete ST is consistent iff it is confluent.*

Proof. Suppose that SP is a strongly complete ST.

"$\Leftarrow$": Let t, t' be two SP-equivalent ground normal forms. By Thm. 6.8, $t \equiv t' \in RTh(SP)$ and thus by Lemma 6.5, t and t' are joinable. But normal forms are joinable only if they are equal.

"$\Rightarrow$": Let u, u' be SP-reducts of a ground term t. Since SP is strongly complete, there are normal forms v, v' such that $u \stackrel{*}{\longrightarrow}_{SP} v$ and $u' \stackrel{*}{\longrightarrow}_{SP} v'$. By Lemma 6.5, u and u' and thus v and v' are SP-equivalent. Since SP is consistent, v and v' are equal and thus a common reduct of t. ❏

Theorem 6.8 also implies that a functional ST SP can be transformed into an equivalent relational one by turning each defined function into its graph or input-output relation. For this purpose, each axiom of $Horn(SP)$ is transformed into an equivalent Horn clause cl such that all equations $t \equiv u$ of cl are **flat**, i.e., either $root(t)$ is a defined function and all other symbols of t or u are constructors or variables or $u \equiv t$ is flat. A first-order formula φ is flattened by repeatedly applying the following rules to it:[9] Let x be a variable that does not occur in the rule antecedents.

$$\frac{p[t/x] \ \Leftarrow \ \varphi}{p \ \Leftarrow \ t \equiv x \wedge \varphi} \qquad \frac{p \ \Leftarrow \ \varphi[t/x] \wedge \psi}{p \ \Leftarrow \ \varphi \wedge t \equiv x \wedge \psi}$$

The flattened formula $flat(\varphi)$ is then turned into its **relational version** by replacing each flat equation $f(t) \equiv u$ or $u \equiv f(t)$ with the logical atom $r_f(t,u)$. If f is of type $w \to s$, then the predicate r_f is of type ws and called the **graph of** f.

Definition 6.10 Given a set F of formulas, let $flat(F) = \{flat(\varphi) \mid \varphi \in F\}$. Moreover, $rel(\Sigma)$ is obtained from Σ by replacing each defined function $f : w \to s \in \Sigma$ by the graph $r_f : ws$ of f. $rel(F)$ is obtained from F by replacing each equation $f(t) \equiv u$ of F with defined function f by the atom $r_f(t,u)$.

[9] A corresponding algorithm has been implemented in Expander2 [40].

Let $SP = (\Sigma, AX)$ be a swinging type with base type $baseSP$ and extension (Σ', AX'). The ST $flat(SP) = (\Sigma, baseAX \cup flat(AX'))$ is called the **flat version of** SP. The ST $rel(SP) = (base\Sigma \cup rel(\Sigma'), baseAX \cup rel(flat(AX')))$ is called the **relational version of** SP. ❏

Theorem 6.11 (equivalence of a functional ST and its relational version) Let SP be a functional ST. Then $rel(SP)$ is functional and for all first-order formulas φ,

$$Ini(SP) \models \varphi \iff Ini(rel(SP)) \models rel(flat(\varphi)). \tag{1}$$

Proof. Let $SP = (\Sigma, AX)$ and $rel(SP) = (\Sigma', AX')$. Since relational specifications are functional, $rel(SP)$ is functional. (1) holds true if for all ground Σ-atoms p,

$$Her(SP) \models p \iff Her(rel(SP)) \models rel(flat(p)). \tag{2}$$

By Thm. 6.8, (2) holds true if for all defined functions $f \in \Sigma'$, predicates and copredicates $p \in \Sigma'$ and $t, u \in NF_\Sigma$,

$$\begin{aligned} f(t) \equiv u \in RTh(SP) &\iff r_f(t,u) \in RTh(rel(SP)), &(3)\\ t \equiv u \in RTh(SP) &\iff t \equiv u \in RTh(rel(SP)), &(4)\\ p(t) \in RTh(SP) &\iff p(t) \in RTh(rel(SP)). &(5)\end{aligned}$$

(3), (4) and (5) can be shown by induction on proof lengths in the reductive calculus for SP and $rel(SP)$, respectively, along the lines of the proof of [37], Thm. 4.13. ❏

Corollary 6.12 *Let SP be a functional specification and $\approx$ be a binary relation on T_Σ that includes $\equiv_{SP} \circ \approx \circ \equiv_{SP}$. Given a defined function $f \in \Sigma$, $\approx$ is compatible with f iff $\approx$ is zigzag compatible with the graph r_f of f.*

Proof. Let $f : w \to s$ and $t, t' \in T_{\Sigma,w}$ such that $t \approx t'$ and $\approx$ is zigzag compatible with r_f. By Thm. 6.11, $Ini(SP) \models f(nf(t)) \equiv nf(f(t))$ implies $Ini(rel(SP)) \models r_f(nf(t), nf(f(t)))$. Since $\approx$ includes $\equiv_{SP} \circ \approx \circ \equiv_{SP}$, $t \approx t'$ implies $nf(t) \approx nf(t')$. Since $\approx$ is zigzag compatible with r_f, there is $u \in NF_\Sigma$ such that $nf(f(t)) \approx u$ and $Ini(rel(SP)) \models r_f(nf(t'), u)$. Hence by Thm. 6.11, $Ini(SP) \models f(nf(t')) \equiv u$ and thus $f(t') \equiv_{SP} u$. Since $\approx$ includes $\equiv_{SP} \circ \approx \circ \equiv_{SP}$, we conclude $f(t) \approx f(t')$. Hence $\approx$ is compatible with f. The converse can be shown in a similar way. ❏

7 Monotonicity, consistency and refinement

Definition 7.1 Let $SP = (\Sigma, AX)$ and $SP' = (\Sigma', AX')$ be swinging types and $\sigma : \Sigma \to \Sigma'$ be a signature morphism.

Let $r : w$ be a predicate or a structural equality of Σ. SP' is **r-monotone w.r.t.** (SP, σ) if for all $t \in T_{\Sigma,w}$, $Ini(SP) \models r(t)$ implies $Ini(SP')|_\sigma \models r(t)$ (cf. Def. 3.1). SP' is **r-consistent w.r.t.** (SP, σ) if, conversely, for all $t \in T_{\Sigma,w}$, $Ini(SP')|_\sigma \models r(t)$ implies $Ini(SP) \models r(t)$.

Let $r : w$ be a copredicate of Σ. SP' is **r-monotone w.r.t.** (SP, σ) if for all $t \in T_{\Sigma,w}$, $Ini(SP')|_\sigma \models r(t)$ implies $Ini(SP) \models r(t)$. SP' is **r-consistent w.r.t.** (SP, σ) if, conversely, for all $t \in T_{\Sigma,w}$, $Ini(SP) \models r(t)$ implies $Ini(SP')|_\sigma \models r(t)$.

SP' is **monotone** resp. **consistent w.r.t.** (SP, σ) if for all relations $r \in \Sigma$, SP' is r-monotone resp. r-consistent w.r.t. (SP, σ).

If σ is an inclusion, i.e., $\Sigma \subseteq \Sigma'$, we write SP instead of (SP, σ). ❏

Definition 7.2 Given a signature Σ, a Horn clause

$$f(t)\{\equiv u\} \quad \Leftarrow \quad \delta \wedge \bigwedge_{i=1}^{n} t_i \equiv u_i \qquad \text{(i)}$$

is **deterministic up to** δ if $var(u) \subseteq V_n$ and for all $1 \leq i \leq n$, $u_i \in NF_\Sigma(X)$ and $var(t_i) \subseteq V_{i-1}$ where $V_0 =_{def} var(t)$ and $V_i =_{def} V_{i-1} \uplus var(u_i)$. ❏

Theorem 7.3 *Let $SP = (\Sigma, AX)$ and $SP' = (\Sigma', AX')$ be swinging types, $\sigma : \Sigma \to \Sigma'$ be a signature morphism, $A = Ini(SP)$ and $B = Ini(SP')_\sigma$.*

(1) Suppose that B satisfies AX. Then SP' is monotone w.r.t. (SP, σ).

(2) Suppose that B satisfies AX. SP' is consistent w.r.t. (SP, σ) iff there is a Σ-homomorphism from B to A.

(3) Suppose that B satisfies AX. For all predicates and copredicates $r \in \Sigma$ such that the complement $\overline{r}$ w.r.t. A is also a predicate resp. copredicate of SP and $\sigma(\overline{r})$ is the complement of $\sigma(r)$ w.r.t. $Ini(SP')$, SP' is r-consistent w.r.t. (SP, σ).

(4) Suppose that SP is functional, B satisfies AX and for all structural equalities $\equiv \in \Sigma$, $\sigma(\not\equiv)$ is the complement of $\sigma(\equiv)$ w.r.t. $Ini(SP')$. Then for all structural equalities, structural inequalities and definedness predicates $r \in \Sigma$, SP' is r-consistent w.r.t. (SP, σ).

(5) Let SP and SP' be Horn. SP' is consistent w.r.t. (SP, σ) if SP' is strongly complete and confluent, $AX' \setminus \sigma(AX)$ consists of axioms for $\Sigma' \setminus \sigma(\Sigma)$,

(5.1) $\sigma(NF_\Sigma) = NF_{\Sigma'}$ **or**

(5.2) $\sigma(NF_\Sigma) \subseteq NF_{\Sigma'}$, SP is complete, SP' is monotone w.r.t. (SP, σ) and each axiom of AX is deterministic up to some δ such that

$$freevar(\delta) \;\subseteq\; V_n \cup \bigcup_{s \in \Sigma} \{X_s \mid \sigma(NF_{\Sigma,s}) = NF_{\Sigma',\sigma(s)}\} \qquad \text{(ii)}$$

(cf. Def. 7.2).

Proof. (1) By assumption $B \in RModEq(SP)$. Hence by (the proof of) Theorem 4.15, the mapping $h : A \to B$ that sends $[t]$ to t^B for all $t \in T_\Sigma$ is a Σ-homomorphism. Let $r \in \Sigma$ be a predicate and $A \models r(t)$, i.e. $[t] \in r^A$. Since h is homomorphic, $t^B = h([t]) \in h(r^A) \subseteq r^B$ and thus $B \models r(t)$. Let $\equiv \in \Sigma$ be a structural equality and $A \models t \equiv u$, i.e. $[t] = [u]$. Then $t^B = h([t]) = h([u]) = u^B$ and thus $B \models t \equiv u$. Let $r \in \Sigma$ be a copredicate and $B \models r(t)$, i.e. $h([t]) = t^B \in r^B$. Since h is homomorphic, $[t] \in r^A$ and thus $A \models r(t)$. We conclude that for all relations $r \in \Sigma$, SP' is r-monotone w.r.t. (SP, σ).

(2) "$\Leftarrow$": Let $h : A \to B$ be the initial Σ-homomorphism from A to B (see (1)). Given a Σ-homomorphism $g : B \to A$, $g \circ h$ is also homomorphic and thus equal to id^A because, by the initiality of A, there is only one Σ-homomorphism from A to A. Hence for all $t \in T_\Sigma$, $g(t^B) = g(h([t])) = [t]$. Let $r \in \Sigma$ be a predicate and $B \models r(t)$, i.e. $t^B \in r^B$. Since g is homomorphic, $[t] = g(t^B) \in r^A$ and thus $A \models r(t)$. Let $\equiv \in \Sigma$ be a structural equality and $B \models t \equiv u$, i.e. $t^B = u^B$. Then $[t] = g(t^B) = g(u^B) = [u]$ and thus $A \models t \equiv u$. Let $r \in \Sigma$ be a copredicate and $A \models r(t)$, i.e. $g(t^B) = [t] \in r^A$. Since g is homomorphic, $t^B \in r^B$ and thus $B \models r(t)$. We conclude that for all relations $r \in \Sigma$, SP' is r-consistent w.r.t. (SP, σ).

"$\Rightarrow$": Define a mapping $g : B \to A$ by $g(t^B) = [t]$ for all $t \in T_\Sigma$. Since B is reachable, the domain of g covers B. g is well-defined: Let $t^B = u^B$. Then $B \models t \equiv u$. Since SP' is $\equiv$-consistent w.r.t. (SP, σ),

$A \models t \equiv u$, i.e. $[t] = [u]$. The initial Σ-homomorphism $h : A \to B$ sends $[t]$ to t^B (see the proof of (1)). Hence $g \circ h = id^A$ and thus $g \circ h$ is homomorphic. Moreover, h is a surjective homomorphism. Hence simple "diagram chasing" shows that g is also homomorphic.

(3) Let $r : w \in \Sigma$ be a predicate and $A \not\models r(t)$. Then $A \models \overline{r}(t)$. By assumption, $\overline{r}$ is also a predicate. Since B satisfies AX and, by Lemma 4.6, $\overline{r}^A$ is the least relation on $T_{\Sigma,w}/\!\equiv_{SP}$ that satisfies the axioms of SP for $\overline{r}$, $B \models \overline{r}(t)$ and thus by assumption,

$$Ini(SP') \quad\models\quad \sigma(\overline{r}(t)) \;=\; \sigma(\overline{r})(\sigma(t)) \;=\; \overline{\sigma(r)}(\sigma(t)).$$

Hence $Ini(SP') \not\models \sigma(r)(\sigma(t)) = \sigma(r(t))$, i.e. $B \not\models r(t)$. Hence SP' is r-consistent w.r.t. (SP, σ).

Let $r : w \in \Sigma$ be a copredicate and $B \not\models r(t)$. Then $Ini(SP') \not\models \sigma(r(t)) = \sigma(r)(\sigma(t))$ and thus by assumption,

$$Ini(SP') \quad\models\quad \overline{\sigma(r)}(\sigma(t)) \;=\; \sigma(\overline{r})(\sigma(t)) \;=\; \sigma(\overline{r}(t)).$$

Hence $B \models \overline{r}(t)$. By assumption, $\overline{r}$ is also a copredicate. Since B satisfies AX and, by Lemma 4.7, $\overline{r}^A$ is the greatest relation on $T_{\Sigma,w}/\!\equiv_{SP}$ that satisfies the axioms of SP for $\overline{r}$, $A \models \overline{r}(t)$ and thus $A \not\models r(t)$. Hence SP' is r-monotone w.r.t. (SP, σ).

(4) Suppose that SP is functional. Let $\equiv : ss \in \Sigma$ be a structural equality and $A \not\models t \equiv u$. By Prop. 4.5, $t \equiv u \notin DTh(SP)$. Since SP is functional, Lemma 4.13 implies $t \not\equiv u \in DTh(SP)$ and thus $A \models t \not\equiv u$, again by Prop. 4.5. Since B satisfies AX and, by Lemma 4.6, $\not\equiv^A$ is the least relation on $T_{\Sigma,ss}/\!\equiv_{SP}$ that satisfies the axioms of SP for $\not\equiv$, $B \models t \not\equiv u$ and thus by assumption,

$$Ini(SP') \quad\models\quad \sigma(t \not\equiv u) \;=\; \sigma(t)\sigma(\not\equiv)\sigma(u) \;=\; \sigma(t)\overline{\sigma(\equiv)}\sigma(u).$$

Hence $Ini(SP') \not\models \sigma(t)\sigma(\equiv)\sigma(u) = \sigma(t \equiv u))$, i.e. $B \not\models t \equiv u$. Hence SP' is $\equiv$-consistent w.r.t. (SP, σ).

Let $\not\equiv : ss \in \Sigma$ be a structural inequality. By assumption, there is a structural equality $\equiv : ss \in \Sigma$ such that $\sigma(\not\equiv) = \overline{\sigma(\equiv)}$ and by Lemma 4.13, for all $t, u \in T_\Sigma$, $t \equiv u \notin DTh(SP)$ iff $t \not\equiv u \in DTh(SP)$, and thus $\not\equiv$ is the complement of $\equiv$ w.r.t. A. Hence

$$\overline{\sigma(\overline{\not\equiv})} = \overline{\sigma(\equiv)} = \sigma(\not\equiv) = \sigma(\not\equiv)$$

and thus

$$\sigma(\overline{\not\equiv}) = \overline{\overline{\sigma(\overline{\not\equiv})}} = \overline{\sigma(\not\equiv)}.$$

Hence by (3), SP' is $\overline{\not\equiv}$-consistent w.r.t. (SP, σ).

Let $Def : s \in \Sigma$ be a definedness predicate and $t \in T_{\Sigma,s}$. Since SP is functional, Prop. 4.12 implies $Def(t) \in DTh(SP)$ and thus $A \models Def(t)$. Hence SP' is Def-consistent w.r.t. (SP, σ).

(5) Suppose that the conditions of (3) hold true. By Cor. 6.9, SP' is functional. Let p be a ground Σ-atom such that $B \models p$. Then $Ini(SP') \models \sigma(p)$ and thus $\sigma(p) \in DTh(SP')$. Since SP' is functional and confluent, Thm. 6.8 implies $\sigma(p) \in RTh(SP')$. Let F be the set of SP'-convergent formulas φ such that φ is $\sigma(SP)$-convergent or contains a symbol of $\Sigma' \setminus \sigma(\Sigma)$.

We claim that F is SP'-reductive. W.l.o.g. let $r(t\tau)$ and $G\tau$ be the premise resp. conclusion of a resolution rule instance such that $G\tau \in F$ (cf. Def. 6.4). If $r(t\tau)$ contains a symbol of $\Sigma' \setminus \sigma(\Sigma)$, then $G\tau \in F$ immediately implies $r(t\tau) \in F$. Otherwise $r(t\tau)$ is a $\sigma(\Sigma)$-atom. Since all axioms of $SP' \setminus \sigma(SP)$ are axioms for symbols of $SP' \setminus \sigma(SP)$, the applied axiom $\varphi = (r(t) \Leftarrow G) \in AX'$ is in $\sigma(AX)$. Hence all variables of φ have their sorts in $\sigma(\Sigma)$.

Suppose that (5.1) holds true. Then $fresh(\varphi)\tau \subseteq NF_{\Sigma'}$ is a set of $\sigma(\Sigma)$-normal forms.

Suppose that (5.2) holds true. Since $\varphi \in \sigma(AX)$, there is a $\psi \in AX$ such that $\varphi = \sigma(\psi)$ and ψ has the form (i). Let H be the premise of ψ and $1 \leq i \leq n$. Since $\sigma(H)\tau = G\tau$ is SP'-convergent, Lemma 6.5 implies that $\sigma(t_i)\tau$ and $\sigma(u_i)\tau$ are SP'-joinable. Since $var(u_i) \subseteq fresh(\psi)$ and $fresh(\varphi)\tau \subseteq NF_{\Sigma'}$, $\sigma(u_i)\tau$ is a normal form. Hence $\sigma(t_i)\tau \xrightarrow{*}_{SP'} \sigma(u_i)\tau$. We show

$$V_i\tau \;\subseteq\; T_{\sigma(\Sigma)} \qquad \text{(iii)}$$

by induction on i. If $i = 0$, then (iii) holds true because $V_0 = var(t)$ and $r(t\tau)$ is a $\sigma(\Sigma)$-atom. Let $i > 0$. Since $t_i \in T_{\sigma(\Sigma)}(X)$ and $var(t_i) \in V_{i-1}$, the induction hypothesis implies $\sigma(t_i)\tau \in T_{\sigma(\Sigma)}$. Since SP is complete, $\sigma(NF_\Sigma) \subseteq NF_{\Sigma'}$ and $\sigma(t_i)\tau \in T_{\sigma(\Sigma)}$, there is a Σ'-normal form $v \in T_{\sigma(\Sigma)}$ such that $\sigma(t_i)\tau$ and v are $\sigma(SP)$-equivalent. Since $\sigma(t_i)\tau \xrightarrow{*}_{SP'} \sigma(u_i)\tau$, Lemma 6.5 implies that $\sigma(t_i)\tau$ and $\sigma(u_i)\tau$ are SP'-equivalent. Hence $\sigma(u_i)\tau \equiv_{SP'} v$ because SP' is monotone w.r.t. (SP,σ) and thus $\sigma(SP)$-equivalence is a subrelation of SP'-equivalence. Since $\sigma(u_i)\tau$ and v are Σ'-normal forms and SP' is consistent, both terms are equal. Hence $\sigma(u_i)\tau \in T_{\sigma(\Sigma)}$ and thus $var(u_i)\tau \subseteq T_{\sigma(\Sigma)}$. Hence (iii) follows from $V_i = V_{i-1} \cup var(u_i)$. In particular, (iii) holds true for $i = n$. Since $fresh(\varphi) = fresh(\psi) \subseteq freevar(\delta) \cup V_n$, 7.3(ii) implies that $fresh(\varphi)\tau \subseteq NF_{\Sigma'}$ is a set of $\sigma(\Sigma)$-normal forms.

Hence in both cases, $\varphi \in \sigma(AX)$ and $fresh(\varphi)\tau \subseteq NF_{\sigma(\Sigma)}$. Since $RTh(\sigma(SP))$ is $\sigma(SP)$-reductive and $G\tau$ is $\sigma(SP)$-convergent, we conclude that $r(t\tau)$ is also $\sigma(SP)$-convergent, i.e., $r(t\tau) \in F$. Therefore, F is SP'-reductive.

Since $RTh(SP')$ is the least SP'-reductive set, F contains $RTh(SP')$. Hence $\sigma(p) \in RTh(SP')$ implies $\sigma(p) \in F$. Since $\sigma(p)$ does not contain a symbol of $\Sigma' \setminus \sigma(\Sigma)$, $\sigma(p)$ is $\sigma(SP)$-convergent and thus a reductive theorem of $\sigma(SP)$. Hence p is a reductive theorem of SP. By Lemma 6.5, $p \in DTh(SP)$ and thus $A \models p$. We conclude that SP' is consistent w.r.t. (SP,σ). ❏

Condition 7.3(5.1) implies that for all sorts of the subtype SP of SP', all s-constructors of SP' are already in SP, while 7.3(5.2) admits additional s-constructors in $\Sigma' \setminus \Sigma$. In both cases, the crucial requirement of 7.3(5) is the confluence of SP'. This property is usually reduced to syntactic conditions on the axioms of AX' such as the following ones:

Theorem 7.4 ([35], Thm. 10.46) *Let $SP = (\Sigma, AX)$ and $SP' = (\Sigma', AX')$ be Horn swinging types such that $SP \subseteq SP'$, SP is confluent, all SP'-reduced terms are normal forms, there is a reduction ordering $>$ for SP'*[10]*, for all sorts $s \in \Sigma$, $NF_{\Sigma,s} = NF_{\Sigma',s}$, $AX' \setminus AX$ consists of axioms for $\Sigma' \setminus \Sigma$. SP' is confluent if for all conditional equations $\varphi, \psi \in SP' \setminus SP$, $\varphi = \psi$ and φ is deterministic or $Ini(SP)$ satisfies all overlays*[11] *induced by φ and ψ.* ❏

For a full proof of Theorem 7.4, consult [42], Satz 5.2.8.

Refinement or abstract implementation notions have a long tradition in data type theory (cf., e.g., [11]). All of them use more or less implicit operators that transform models of the implementing—*concrete*—specification, say SP, into models of the implemented—*abstract*—specification, say SP'. While the original approaches focused on particular implementations, i.e., particular models of SP such as the initial or final one (cf. [19, 17, 16, 12]), later refinement notions take into the account the entire class of SP-models (cf. [49, 47, 24, 4]). In addition to the requirement that certain operators transform SP-models into SP'-models, [12] and [24] also demand the converse: distinct data of a structure to be refined must not be identified in the implementing structure.

[10] Cf. [35], Def. 10.38.
[11] Cf. [35], Def. 10.40.

Which are the operators supposed to transform an SP-model A into an SP'-model B? Intuitively, A implements B if B can be constructed from A by

- **translating** the symbols of SP' to symbols of SP along a signature morphism *rep*,
- **restricting** the set of *concrete* data of A to the *rep*-images of *abstract* data of B,
- **identifying** concrete data with respect to the kernel of an *abstraction function* from the concrete to the abstract data.

[49, 24, 4] pay particular attention to the identification step and consider congruences induced by behavioral or contextual equivalence relations. Consequently, domains of the abstract specifications are often refined to *hidden* sorts. Since the identification step is supposed to hide implementation details, this is quite natural, although swinging types cope with visible as well as hidden sorts. Hence they allow us to design in the same framework both behavioral refinements and initially-algebraic implementations as proposed in, e.g., [12]. However, we always base refinements on representation *functions*, in contrast to the (bisimulation) *relations* that establish Jacobs' behavioral refinements of coalgebras [20, 21].

Definition 7.5 Let $SP = (\Sigma, AX)$ and $SP' = (\Sigma', AX')$ be swinging types and $rep : \Sigma \to \Sigma'$ be a signature morphism, called **representation morphism**. SP' **refines** or **implements** SP **along** *rep* if $Ini(SP')_{rep}$ is an SP-model and SP' is consistent w.r.t. (SP, rep). ❏

By [37], Lemma 3.5, SP' is consistent w.r.t. (SP, rep) iff there is Σ-homomorphism $abs : Ini(SP')_{rep} \to Ini(SP)$, which yields the above-mentioned abstraction function. Hence Def. 7.5 reflects the three steps of a refinement: **translating** $Ini(SP')$ via *rep* results in $Ini(SP')|_{rep}$, **restricting** $Ini(SP')|_{rep}$ to Σ leads to $Ini(SP')_{rep}$, and **identifying** data of $Ini(SP')_{rep}$ means to apply the abstraction homomorphism.

The condition that the "concrete" specification SP' is consistent w.r.t. the "abstract" specification SP has also been called *RI-correctness*[12] (cf. [12]). In terms of Def. 7.5, RI-correctness says that SP' is $\equiv$-consistent w.r.t. (SP, rep). Intuitively, RI-correctness ensures that the implementing specification SP' identifies two data *only if* they are equal w.r.t. SP. While this occurs as a general refinement condition only in [12, 24], all formal approaches agree about the requirement that an implementation is correct only if it satisfies the "abstract" axioms. [12] enforces this condition by factoring the implementation model through SP-equivalence. The resulting quotient is isomorphic to the final model of SP iff SP' is RI-correct. Therefore, [12] avoids the proof of axiom validity, but at the expense of including the implementation of equality predicates into the refinement approach. Moreover, as a consistency condition, RI-correctness usually resists a mechanical proof unless it can be reduced to a tractable criterion. Fortunately, Theorem 7.3(4) allows us to reduce RI-correctness to the functionality of the "abstract" specification SP:

Corollary 7.6 *Let $SP = (\Sigma, AX)$ and $SP' = (\Sigma', AX')$ be swinging types and $rep : \Sigma \to \Sigma'$ be a signature morphism such that SP is functional, the only relations of Σ are structural equalities, inequalities and definedness predicates and for all structural equalities $\equiv \in \Sigma$, $\sigma(\not\equiv)$ is the complement of $\sigma(\equiv)$ w.r.t. $Ini(SP')$.*

SP' refines SP along rep if $Ini(SP')_{rep}$ is an SP-model.

Proof. The statement follows immediately from Theorem 7.3(4). ❏

Example 7.7 (MAP refines STACK) This example is a popular benchmark for refinement approaches (cf., e.g., [17], Sect. 4.4; [49]; [33], Ex. 7.20; [24], Sect. 4.1; [21], Sect. 4.3). Stacks are imple-

[12] "RI" refers to the restriction step (R) and the identification step (I) of a refinement.

mented as pairs consisting of a finite array (= map with finite domain) and a top pointer. Formally, the refinement SP' of STACK reads as follows (cf. Ex. **??**):

SP' = ENTRY `and` NAT `then`

`vissorts`	$nat + entry$
`hidsorts`	$map \quad map \times nat$
`constructs`	$\kappa_1 : nat \to nat + entry$
	$\kappa_2 : entry \to nat + entry$
	$new :\to map$
	$upd : nat \times entry \times map \to map$
`strong constructs`	$(\_,\_) : map \times nat \to map \times nat$
`destructs`	$get : map \times nat \to nat + entry$
	$top : map \times nat \to nat + entry$
	$pop : map \times nat \to map \times nat$
`defuncts`	$pred : nat \to nat$
	$empty :\to map \times nat$
	$push : entry \times (map \times nat) \to map \times nat$
`local preds`	$\not\sim: (map \times nat) \times (map \times nat)$
`vars`	$i, j : nat \quad x : entry \quad f : map \quad s, s' : map \times nat$
`Horn axioms`	$pred(0) \equiv 0$
	$pred(i+1) \equiv i$
	$get(new, i) \equiv \kappa_1(i)$
	$get(upd(i, x, f), i) \equiv \kappa_2(x)$
	$get(upd(i, x, f), j) \equiv get(f, j) \quad \Leftarrow \quad i \not\equiv j$
	$empty \equiv (new, 0)$
	$push(x, (f, i)) \equiv (upd(i+1, x, f), i+1)$
	$top((f, i)) \equiv get(f, i)$
	$pop((f, i)) \equiv (f, pred(i))$
	$s \not\sim s' \quad \Leftarrow \quad top(s) \not\equiv top(s')$
	$s \not\sim s' \quad \Leftarrow \quad pop(s) \not\sim pop(s')$

SP and SP' are functional because both types are complete, *terminating* and *weakly orthogonal* (see [35]). We prove that SP' refines SP along the signature morphism *rep* that maps the sorts 1 and *stack* to *nat*, resp. $map \times nat$, the 1-constant () to the *nat*-term $\kappa_1(0)$, the structural *stack*-equality and -inequality to behavioral $map \times nat$-equality resp. its complement and all other symbols of STACK to themselves. Moreover, the constructors *empty* and *push* of STACK become defined functions. *top* and *pop* remain defined functions, but serve as destructors in the implementation. Since constructors are turned into non-constructors, we cannot apply Thm. 7.3(5) for proving that SP' is consistent w.r.t. (STACK, *rep*). Corollary 7.6, however, allows us to conclude that SP' refines STACK along *rep* if the following *rep*-images of STACK-axioms are inductive theorems of SP':

$$top(empty) \equiv \kappa_1(0) \tag{1}$$
$$top(push(x, s)) \equiv \kappa_2(x) \tag{2}$$
$$pop(empty) \sim empty \tag{3}$$
$$pop(push(x, s)) \sim s \tag{4}$$
$$empty \not\sim push(x, s) \tag{5}$$
$$push(x, s) \not\sim empty \tag{6}$$

$$push(x,s) \not\sim push(y,s') \quad \Leftarrow \quad x \not\equiv y \tag{7}$$
$$push(x,s) \not\sim push(y,s') \quad \Leftarrow \quad s \not\sim s' \tag{8}$$

In addition, STACK implicitly includes the equality axioms for $\equiv_{stack}$ and thus their *rep*-images must also be inductive theorems of SP'. Since these formulas describe the congruence property of behavioral $map \times nat$-equality, we may conclude their validity from the behavioral consistency of SP'. Indeed, the criteria for behavioral consistency given by Theorem 8.5 hold true both for SP and SP': It is easy to show that both types are head complete. They are image finite and thus, by Theorem **??** continuous. Finally, SP and SP' are coinductive.

We present top-down proofs of (1)-(8) using the inference rules discussed in Section 5.

Proof of (1):

$top(empty) \equiv \kappa_1(0)$

narrowing on *empty*

$\vdash \quad top((new,0)) \equiv \kappa_1(0)$

narrowing on *top*

$\vdash \quad get(new,0) \equiv \kappa_1(0)$

narrowing on *get*

$\vdash \quad \kappa_1(0) \equiv \kappa_1(0)$

simplification

$\vdash \quad True$

Proof of (2):

$top(push(x,s)) \equiv \kappa_2(x)$

narrowing on *push*

$\vdash \quad \exists f,i : top((upd(i+1,x,f),i+1)) \equiv \kappa_2(x) \wedge s \equiv (f,i)$

narrowing on *top*

$\vdash \quad \exists f,i : get(upd(i+1,x,f),i+1) \equiv \kappa_2(x) \wedge s \equiv (f,i)$

narrowing on *get*

$\vdash \quad \exists f,i : \kappa_2(x) \equiv \kappa_2(x) \wedge s \equiv (f,i)$

simplification

$\vdash \quad \exists f,i : s \equiv (f,i)$

completeness of SP'

$\vdash \quad True$

Proof of (3):

$pop(empty) \sim empty$

narrowing on *empty*

$\vdash \quad pop((new,0)) \sim empty$

narrowing on *pop*

$\vdash \quad (new,pred(0)) \sim empty$

narrowing on *pred*

$\vdash \quad (new,0) \sim empty$

narrowing on *empty*

$\vdash \quad (new,0) \sim (new,0)$

simplification

$\vdash \quad True$

Proof of (4):

$pop(push(x,s)) \sim s$

narrowing on *push*

$\vdash \quad \exists f,i : pop((upd(i+1,x,f),i+1)) \sim s \wedge s \equiv (f,i)$

narrowing on *pop*

$\vdash \quad \exists f,i : (upd(i+1,x,f),pred(i+1)) \sim s \wedge s \equiv (f,i)$

narrowing on *pred*

$\vdash \quad \exists f,i : (upd(i+1,x,f),i) \sim s \wedge s \equiv (f,i)$

application of lemma (9) (see below)

$\vdash \quad \exists f,i : (f,i) \sim (f,i) \wedge i+1 > i \wedge s \equiv (f,i)$

simplification

$\vdash \quad \exists f,i : i+1 > i \wedge s \equiv (f,i)$

application of the lemma $i+1 > i$

$\vdash \quad \exists f,i : s \equiv (f,i)$

completeness of SP'

$\vdash \quad True$

Proof of (5):

$empty \not\sim push(x,s)$

narrowing on $\not\sim$

$\vdash \quad top(empty) \not\equiv top(push(x,s)) \vee pop(empty) \not\sim pop(push(x,s))$

summand removal

$\vdash \quad top(empty) \not\equiv top(push(x,s))$

application of (1), (2) and equality axioms

$\vdash \quad \kappa_1(0) \not\equiv \kappa_2(x)$

narrowing on $\not\equiv$

$\vdash \quad True$

Proof of (6): Analogously.

Proof of (7):

$push(x,s) \not\sim push(y,s') \;\Leftarrow\; x \not\equiv y$

narrowing on $\not\sim$

$\vdash \quad top(push(x,s)) \not\equiv top(push(y,s')) \vee pop(push(x,s)) \not\sim pop(push(y,s')) \;\Leftarrow\; x \not\equiv y$

summand removal

$\vdash \quad top(push(x,s)) \not\equiv top(push(y,s')) \;\Leftarrow\; x \not\equiv y$

application of (2) and equality axioms

$\vdash \quad \kappa_2(x) \not\equiv \kappa_2(y) \;\Leftarrow\; x \not\equiv y$

narrowing on $\not\equiv$

$\vdash \quad x \not\equiv y \;\Leftarrow\; x \not\equiv y$

simplification

$\vdash \quad True$

Proof of (8):

$push(x,s) \not\sim push(y,s') \;\Leftarrow\; s \not\sim s'$

narrowing on $\not\sim$

$\vdash \quad top(push(x,s)) \not\equiv top(push(y,s')) \vee pop(push(x,s)) \not\sim pop(push(y,s')) \;\Leftarrow\; s \not\sim s'$

summand removal

$\vdash \quad pop(push(x,s)) \not\sim pop(push(y,s')) \;\Leftarrow\; s \not\sim s'$

application of (4) and compatibility of $\not\sim$ with $\sim$ (part of the behavioral consistency of SP')

$\vdash \quad s \not\sim s' \;\Leftarrow\; s \not\sim s'$

simplification

$\vdash \quad True$

The proof of (4) uses the following lemma that characterizes behavioral stack equivalence in terms of a constructor property:

$$(upd(i,x,f),j) \sim (f,j) \;\Leftarrow\; i > j. \tag{9}$$

Proof of (9):

$(upd(i,x,f),j) \sim (f,j) \;\Leftarrow\; i > j$

introduction of variables

$\vdash \quad s \sim t \;\Leftarrow\; s \equiv (upd(i,x,f),j) \wedge t \equiv (f,j) \wedge i > j$

coinduction on $\sim$

$\vdash \quad s \equiv (upd(i,x,f),j) \wedge t \equiv (f,j) \wedge i > j$
$\quad\Rightarrow\; \exists i',x',f',j' : (top(s) \equiv top(t) \wedge pop(s) \equiv (upd(i',x',f'),j')$
$\qquad\qquad \wedge\; pop(t) \equiv (f',j') \wedge i' > j')$

elimination of variables

$\vdash \quad \exists i',x',f',j' : (top((upd(i,x,f),j)) \equiv top((f,j))$
$\qquad\qquad \wedge\; pop((upd(i,x,f),j)) \equiv (upd(i',x',f'),j')$
$\qquad\qquad \wedge\; pop((f,j)) \equiv (f',j') \wedge i' > j') \;\Leftarrow\; i > j$

narrowing on *top* and *pop*

$\vdash \quad \exists i',x',f',j' : (get(upd(i,x,f),j) \equiv get(f,j)$
$\qquad\qquad \wedge\; (upd(i,x,f),pred(j)) \equiv (upd(i',x',f'),j')$
$\qquad\qquad \wedge\; (f,pred(j)) \equiv (f',j') \wedge i' > j') \;\Leftarrow\; i > j$

elimination of variables

$\vdash \quad (get(upd(i,x,f),j) \equiv get(f,j)$
$\qquad \wedge\; (upd(i,x,f),pred(j)) \equiv (upd(i,x,f),pred(j))$
$\qquad \wedge\; (f,pred(j)) \equiv (f,pred(j)) \wedge i > pred(j)) \;\Leftarrow\; i > j$

simplification

$\vdash \quad (get(upd(i,x,f),j) \equiv get(f,j) \wedge i > pred(j)) \;\Leftarrow\; i > j$

narrowing on *get*

$\vdash \quad (i \not\equiv j \wedge i > pred(j)) \;\Leftarrow\; i > j \qquad \text{(A)}$

application of (A)

$\vdash \quad i > j \;\Leftarrow\; i > j$

simplification

$\vdash \quad True$ ❑

At first sight, implementing visible sorts by hidden ones seems to contradict the goal of a refinement to make data visible instead of hiding them. But a closer look reveals that a visible type with a structural equality relation is actually more abstract than a corresponding hidden type whose equality does not draw on the abstract structure of normal form representations. Apart from a rather few data types where all relevant information about an object is encoded in a normal form, specifications deal with objects that cannot be identified unambiguously from their symbolic representations. Normal forms are pure abstract syntax. They often abstract from the "real" identity of an object that can only be concluded from its behavior in response to applying observers. From this point of view, a refinement of a visible type interprets abstract data within a given environment where constructors still serve the purpose of building up data representations, but these do not define the objects uniquely. Hence it is quite natural to implement a visible type by a hidden one.

Given a swinging type SP, SP-equivalence is always included in behavioral SP-equivalence. Hence the final SP-model can be represented as a quotient the initial one. This fact suggests the attempt to axiomatize the factorization and to add the axioms to those of SP such that the initial model of the extended specification agrees with the final model of SP. Data type theorists have pursued this goal for quite a long time. The problem is that the additional axioms are of a completely different kind than the axioms of SP. While the latter represent functional-logic programs for functions or predicates, the additional axioms equate different normal forms with each other. For instance, in the previous example, SP might be extended by the equations

$$\begin{array}{l} (upd(i,x,f),j) \equiv (f,j) \ \Leftarrow \ i > j \\ upd(i,x,upd(i,y,f)) \equiv upd(i,x,f) \\ upd(i,x,upd(j,y,f)) \equiv upd(j,y,upd(i,x,f)) \ \Leftarrow \ i \not\equiv j \end{array}$$

in order to obtain a specification whose equivalence coincides with behavioral SP-equivalence. Besides the problem of finding appropriate constructor axioms that capture a behavioral equivalence relation, such axioms make almost all manageable proof methods inapplicable. For example, easily checkable syntactic criteria for functionality do not hold any more so that powerful automatable proof rules are no longer sound. Constructor axioms induce non-trivial *critical clauses* that must be checked for *subjoinability* in order to ensure functionality (cf. [35]). One gets entangled in additional verification problems that are far from the original goals that led one to applying deductive methods at all. One of the most important benefits from presenting data types as swinging types is the fact that these additional verification problems simply disappear.

8 A criterion for behavioral consistency

[37], Thm. 6.5, is adapted to the new definition of swinging types (see Def. 2.3).

Definition 8.1 (defining formulas) An atom $\delta(t,a,u)$ is **defining** if either δ is a transition relation or $\delta(t,u) = r(t)$ and r is a local relation or $\delta(t,u) = (f(t) \equiv u)$ and f is a defined function. $\delta(t,u)$ is **observing** if δ, r or f, respectively, is an observer and thus t may be written as (t,a) where t is a hidden-sorted term and a is—possibly empty—term tuple. A **non-observing formula** is a conjunction of defining atoms that are not observing. ❏

Definition 8.2 (coinductive specification) Let $SP = (\Sigma, AX)$ be a swinging type and $visSP$ be the greatest visible subtype of SP (cf. Def. 2.3).

A Horn clause $p \Leftarrow \varphi$ is **coinductive** if either $p = \delta(t,u)$ is non-observing, t is strongly normal (cf. Def. 2.3) and φ does not contain observers or $p = \delta(t,a,u)$ is observing,

$$\varphi \quad = \quad G \wedge \delta_1(t_1,a_1,u_1) \wedge G_1 \wedge \cdots \wedge \delta_n(t_n,a_n,u_n) \wedge G_n$$

and the following conditions hold true: Let $V_0 = var(t,a,G)$ and for all $1 \leq i \leq n$, $V_i = V_{i-1} \cup var(a_i,u_i,G_i)$.

(1) t is strongly normal or $t = c(t')$ for a constructor c and a strong normal form t', a is strongly normal, u is normal, G is weakly modal (see Def. 2.2) and non-observing, $var(u) \subseteq V_n$ and $out(G) \cap var(t,a) = \emptyset$.

(2) For all $1 \leq i \leq n$, $\delta_i(t_i,a_i,u_i)$ is observing, (t_i,a_i) is normal, u_i is strongly normal, G_i is weakly modal and non-observing, $var(t_i) \subseteq V_{i-1}$, $(var(u_i) \cup out(G_i)) \cap (V_{i-1} \cup var(a_i,u_i)) = \emptyset$ and $var(a_i) \subseteq var(a) \cup visvar(a_i)$ where, for any term t, **visvar(t)** is the set of visibly-sorted variables of t.

A co-Horn clause $r(t) \Rightarrow \varphi$ is **coinductive** if t is strongly normal. SP is **coinductive** if all axioms φ of $SP \setminus visSP$ are coinductive. ❏

The BH and RG congruence criteria [5, 45] capture simple classes of both coinductive and functional specifications. Their correctness can be derived from Thm. 8.5.

Definition 8.3 Let $SP = (\Sigma, AX)$ be a swinging type. The **constructor closure** $\approx \subseteq T_\Sigma^2$ of $\sim_{SP}$ is defined inductively as follows:

- $\sim_{SP} \subseteq \approx$,
- for all $t,t' \in T_\Sigma$, constructors $c : w \to s$ and $u,u' \in T_{\Sigma,w}$,

$$t \equiv_{SP} c(u) \wedge u \approx u' \wedge c(u') \equiv_{SP} t' \quad \text{implies} \quad t \approx t'. \quad ❏$$

It is easy to see that the composition $\equiv_{SP} \circ \approx \circ \equiv_{SP}$ of relations is a subrelation of $\approx$. Moreover, the property of behavioral equivalence stated by Lemma 5.4 holds true for the constructor closure of behavioral equivalence as well:

Lemma 8.4 *Let SP be functional and head complete, t be a strong normal form, $\sigma : X \to NF_\Sigma$ and $u \in NF_\Sigma$ such that $t\sigma \approx u$. Then $u = t\tau$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$.*

Proof by induction on the size of t. Let $t\sigma \approx u$. If $t\sigma \sim_{SP} u$, then by Lemma 5.4, $t\tau = u$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$. Otherwise $t\sigma \equiv_{SP} c(v)$, $v \approx v'$ and $c(v') \equiv_{SP} u$ for some constructor c and $v,v' \in T_\Sigma$. If t is a variable, then define $\tau : X \to NF_\Sigma$ by $t\tau = u$ and $\tau =_{X \setminus \{t\}} \sigma$. Hence $t\sigma \approx u$ implies $\sigma \approx \tau$. If t is not a variable, then there is $t' \in NF_\Sigma(X)$ such that $t = c(t')$ and $t'\sigma \equiv_{SP} v$ because SP is consistent. Since SP is functional, there are $u', v'' \in NF_\Sigma$ such that $v'' \equiv_{SP} v' \equiv_{SP} u'$ and $c(u') = u$. Hence $t'\sigma \equiv_{SP} v \approx v' \equiv_{SP} v''$ and thus $t'\sigma \approx v''$. By induction hypothesis, $t'\tau = v''$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$. Hence $t\tau = c(t'\tau) = c(v'') \equiv_{SP} c(u') = u$ and thus $t\tau = u$ because v'' and u' are normal forms and SP is consistent. ❏

Theorem 8.5 (criteria for behavioral consistency) *A coinductive, functional, head complete and continuous specification SP is behaviorally consistent.*

Proof. Let $SP = (\Sigma, AX)$ and $visSP = (vis\Sigma, visAX)$ be the greatest visible subtype of SP (cf. Def. 2.3). By [37], Lemma 3.6, $\sim_{SP}$ is zigzag compatible with all structural equalities of Σ and compatible with all behavioral equalities of Σ and with all symbols of $vis\Sigma$ that are not structural equalities. By Def. 8.2, the following two parts of SP can be separated from the rest:

- The **non-observer level** $SP_1 = (\Sigma_1, AX_1)$ consists of all defined functions and predicates of SP that are not observers and their axioms.
- The **observer level** consists of all observers of SP and their axioms.

Let $\approx$ be the constructor closure of $\sim_{SP}$. By definition, $\approx$ is compatible with the constructors of Σ. By Corollary 4.14, $\equiv_{SP}$ is a subset of $\sim_{SP}$ and thus of $\approx$.

For all visible sorts s, $\approx_s$ is a subrelation of $\equiv_{SP}$: Let s be a visible sort and $t \approx_s t'$. We prove $t \equiv_{SP} t'$ by induction on the size of t, t'. If $t \sim_s t'$, then $t \equiv_{SP} t'$. Otherwise $t \equiv_{SP} c(u)$, $u \approx u'$ and $t' \equiv_{SP} c(u')$ for a constructor c and term tuples u, u'. By induction hypothesis, $u \equiv_{SP} u'$. Hence $t \equiv_{SP} t'$.

We conclude that for all visible sorts s, $\approx_s = \equiv_{SP,s}$. Hence $\approx$ is compatible with $vis\Sigma$ and thus with Σ_1 if

$$\approx \text{ is (zigzag) compatible with all symbols of } \Sigma_1 \setminus vis\Sigma. \tag{1}$$

Let us show (1). Since $\equiv_{SP} \circ \approx \circ \equiv_{SP}$ is a subrelation of $\approx$, Cor. 6.12 implies that (1) is equivalent to (2): for all non-observing ground $r\Sigma_1$-atoms $\delta(t,u)$ there is $u' \in T_\Sigma \cup \{\varepsilon\}$ such that

$$Ini(SP_1) \models \delta(t,u) \ \wedge \ t \approx t' \quad \text{implies} \quad Ini(SP_1) \models \delta(t',u') \ \wedge \ u \approx u'. \tag{2}$$

By Prop. 3.6, (2) follows from a corresponding property of an approximation of $Ini(SP_1)$: for all non-observing ground Σ_1-atoms $\delta(t,u)$ specified on the non-observer level and $i \in \mathbb{N}$ there is $u' \in NF_\Sigma \cup \{\varepsilon\}$ such that

$$\Phi^i(\emptyset) \models \delta(t,u) \ \wedge \ t \approx t' \quad \text{implies} \quad \Phi^i(\emptyset) \models \delta(t',u') \ \wedge \ u \approx u' \tag{3}$$

where Φ is the $(AX_1 \setminus visAX)$-consequence operator on $Ini(SP_1)|_{vis\Sigma}$.

We prove (3) by induction on i. Since for all predicates $r \in r\Sigma_1$, $r^\emptyset = \emptyset$, (3) holds true for $i = 0$. Let $i > 0$. By induction hypothesis, (3) is valid for $i-1$ and thus

$$\approx \text{ is a behavioral } \Sigma_1\text{-congruence on } \Phi^{i-1}(\emptyset). \tag{4}$$

Let $\Phi^i(\emptyset) \models \delta(t,u)$ and $t \approx t'$. By the definition of Φ and since SP_1 is coinductive, there are an axiom $\delta(t_0,u_0) \Leftarrow \varphi$ on the non-observer level and $\sigma : X \to NF_\Sigma$ such that t_0 is strongly normal, $(t_0,u_0)\sigma = (t,u)$ and $\Phi^{i-1}(\emptyset) \models \varphi\sigma$. Hence $t_0\sigma = t \approx t'$ and thus Lemma 8.4 implies $t_0\tau = t'$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$. By Def. 2.3(c), φ is weakly modal with output Y such that $var(t_0) \cap Y = \emptyset$. Since $\sigma \approx \tau$, (4) and [37], Thm. 3.8(2), imply $\Phi^{i-1}(\emptyset) \models \varphi\tau'$ for some $\tau' \approx \tau$ with $\tau' =_Y \tau$. Hence $\Phi^i(\emptyset) \models \delta(t_0,u_0)\tau'$ and $u = u_0\sigma \approx u_0\tau \approx u_0\tau'$. Since $var(t_0) \cap Y = \emptyset$, $t_0\tau' = t_0\tau = t'$. Hence $\Phi^i(\emptyset) \models \delta(t',u')$ for $u' = u_0\tau' \approx u$.

This completes the proof of (1). Next we show that $\sim_{SP}$ is compatible with all constructors of Σ.

Suppose that $\approx$ satisfies all behavior axioms for Σ (cf. Def. 2.3). Since behavioral SP-equivalence is the greatest relation satisfying the behavior axioms and $\sim_{SP}$ is a subrelation of $\approx$, $\approx$ agrees with $\sim_{SP}$. Hence $\sim_{SP}$ is compatible with all constructors of Σ because $\approx$ has this property by definition.

Since for all visible sorts s, $\approx_s$ is a subrelation of $\equiv_{SP}$, it remains to show that $\approx$ solves the behavior axioms for all hidden sorts of Σ. We start with B4 (cf. Def. 1.3). By Lemma 5.3, we have to show that for all $t, t' \in T_{\Sigma,s}$, strong constructors c and $u, u' \in T_\Sigma^*$,

$$\begin{array}{lll} \text{(i)} & t \approx t' \ \wedge \ t \equiv_{SP} c(u) \quad \text{implies} & \exists\, u' : (t' \equiv_{SP} c(u') \wedge u \approx u')), \\ \text{(ii)} & t \approx t' \ \wedge \ t' \equiv_{SP} c(u') \quad \text{implies} & \exists\, u : (t \equiv_{SP} c(u) \wedge u \approx u')). \end{array}$$

We show (i). (ii) can be proved analogously. So let $t \approx t'$ and $t \equiv_{SP} c(u)$. Since SP is functional, we may assume that t, t', u are normal forms. Moreover, $c(u) \approx t'$ because $\equiv_{SP} \circ \approx$ is a subrelation of $\approx$. Hence by Lemma 8.4, $t' = c(u')$ and $u \approx u'$ for some $u' \in NF_\Sigma \cup \{\varepsilon\}$.

Since $\equiv_{SP} \circ \approx \circ \equiv_{SP}$ is a subrelation of $\approx$, $\approx$ satisfies the remaining behavior axioms if for all observing ground atoms $\delta(t,a,u)$,

$$t \approx t' \;\wedge\; Ini(SP) \models \delta(t,a,u) \quad \text{implies} \quad \exists\, u' : (Ini(SP) \models \delta(t',a,u') \;\wedge\; u \approx u'). \tag{5}$$

Since SP is functional and δ is compatible with SP-equivalence, Prop. 4.5 and Thm. 6.8 imply that (5) is equivalent to (6): for all ground normal forms t,a,u and observing atoms $\delta(t,a,u)$,

$$t \approx t' \;\wedge\; \delta(t,a,u) \in RTh(SP) \quad \text{implies} \quad \exists\, u' \in NF_\Sigma \cup \{\varepsilon\} : (\delta(t',a,u') \in RTh(SP) \;\wedge\; u \approx u'). \tag{6}$$

It remains to show (6). First note that by (1) and [37], Thm. 3.8(2), for all weakly modal Σ_1-formulas φ and $\sigma,\tau : X \to NF_\Sigma$,

$$\sigma \approx \tau \;\wedge\; \varphi\sigma \in RTh(SP) \quad \text{implies} \quad \exists\, \tau' : X \to NF_\Sigma : \varphi\tau' \in RTh(SP) \;\wedge\; \sigma \approx \tau' =_{out(\varphi)} \tau. \tag{7}$$

Let $t \approx t'$ and $\delta(t,a,u) \in RTh(SP)$. We show the conclusion of (6) by induction on the proof length of $\delta(t,a,u)$ in the reductive calculus for SP. Since SP is coinductive, there are a formula

$$\varphi \;=\; G_0 \wedge \delta_1(t_1,a_1,u_1) \wedge G_1 \wedge \cdots \wedge \delta_n(t_n,a_n,u_n) \wedge G_n$$

and an axiom $\delta(t_0,a_0,u_0) \Leftarrow \varphi$ on the 2nd hidden level such that Def. 8.2(1) and (2) hold true for t_0,a_0,u_0,G_0 instead of t,a,u,G. Moreover, there is $\sigma : X \to NF_\Sigma$ such that $(t_0,a_0,u_0)\sigma = (t,a,u)$, $G_0 \in RTh(SP)$ and for all $1 \le i \le n$, the proof length of $\delta_i(t_i,a_i,u_i)\sigma$ is smaller than the one of $\delta(t,a,u)$. By the definition of $\approx$, we have one of two cases:

(A) $t \sim_{SP} t'$,

(B) $t \equiv_{SP} d(v)$, $v \approx v'$ and $d(v') \equiv_{SP} t'$ for some constructor d and ground terms v and v'.

Case A. $\delta(t,a,u) \in RTh(SP)$ implies $Ini(SP) \models \delta(t,a,u)$. Suppose that $\delta(t,a,u) = (f(t,a) \equiv u)$ for some functional destructor $f : w \to s$. Hence $f(t,a) \sim_{SP} f(t',a)$ because $\sim_{SP}$ satisfies the behavior axioms. We conclude $Ini(SP) \models (f(t',a) \equiv u') = \delta(t',a,u')$ for $u' = f(t',a) \sim_{SP} f(t,a) = u$. If δ is a relational destructor, then $Ini(SP) \models \delta(t',a,u)$ because $\sim_{SP}$ satisfies the behavior axioms. Hence $Ini(SP) \models \delta(t',a,u')$ for $u' = u$. If δ is a transition relation, then $Ini(SP) \models \delta(t',a,u')$ for some $u' \sim_{SP} u$ because $\sim_{SP}$ satisfies the behavior axioms.

Hence in all three subcases, $Ini(SP) \models \delta(t',a,u')$ for some $u' \sim_{SP} u$. Since $\sim_{SP}$ is a subset of $\approx$, we conclude $u' \approx u$.

Case B. By Def. 8.2(1), there are two subcases: (B1) t_0 is strongly normal, (B2) $t_0 = c(t_0')$ for a constructor c and a strong normal form t_0'. In case B1, $(t_0,a_0)\sigma = (t,a) \approx (t',a)$ and Lemma 8.4 implies $(t_0,a_0)\tau = (t',a)$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$. In case B2, $c(t_0'\sigma) = t_0\sigma = t \equiv_{SP} d(v)$ and thus $c = d$ and $t_0'\sigma \equiv_{SP} v$ because SP is consistent. Hence $(t_0',a_0)\sigma \equiv_{SP} (v,a) \approx (v',a)$. Since $\equiv_{SP} \circ \approx$ is a subrelation of $\approx$, Lemma 8.4 implies $(t_0',a_0)\tau = (v',a)$ and $\sigma \approx \tau$ for some $\tau : X \to NF_\Sigma$.

We construct a substitution $\tau' : X \to NF_\Sigma$ with

(a) $t_0\tau = t_0\tau'$

and prove by induction on i that for all $0 \le i \le n$,

(b) $a_i\sigma = a_i\tau'$,

(c) $\delta_i(t_i,a_i,u_i)\tau' \in RTh(SP)$ if $i > 0$,

(d) $G_i\tau' \in RTh(SP)$,

(e) $x\sigma \approx x\tau'$ for all $x \in V_i$.

Define $x\tau' = x\tau$ for all $x \in var(t_0, a_0)$. Then (a) holds true. Since $a_0\tau = a = a_0\sigma$, (b) holds true for $i = 0$. By (7), $\sigma \approx \tau$ and $G_0 \in RTh(SP)$ imply $G_0\tau'' \in RTh(SP)$ for some $\tau'' : X \to NF_\Sigma$ with $\sigma \approx \tau'' =_{out(G_0)} \tau$. Define $x\tau' = x\tau''$ for all $x \in var(G_0) \setminus var(t_0, a_0)$. Since $out(G_0) \cap var(t_0, a_0) = \emptyset$, we have $x\tau'' = x\tau = x\tau'$ for all $x \in var(G_0) \cap var(t_0, a_0)$. Hence $G_0\tau'' \in RTh(SP)$ implies (d) for $i = 0$. Moreover, for all $x \in var(G_0) \setminus var(t_0, a_0)$, $x\sigma \approx x\tau'' = x\tau'$. Hence $V_0 = var(t_0, a_0, G_0)$, (a) and (b) for $i = 0$ imply (e) for $i = 0$.

Let $i > 0$. Suppose that (b)-(e) hold true for $i-1$. Since $var(a_i) \subseteq var(a_0) \cup visvar(a_i)$ and $a_0\sigma = a_0\tau'$, we have $x\sigma = x\tau'$ for all $x \in var(a_i) \setminus visvar(a_i)$. Define $x\tau' = x\sigma$ for all $x \in visvar(a_i) \setminus V_{i-1}$. Hence (e) for $i-1$ implies $x\sigma \approx x\tau'$ and thus $x\sigma \equiv_{SP} x\tau'$ for all $x \in visvar(a_i)$. We conclude $x\sigma = x\tau'$ for all $x \in visvar(a_i)$ because $x\sigma$ and $x\tau'$ are normal and SP is consistent. Hence for all $x \in var(a_i)$, $x\sigma = x\tau'$, and thus (b) holds true.

Since $var(t_i) \subseteq V_{i-1}$ and t_i is normal, (e) for $i-1$ implies $t_i\sigma \approx t_i\tau'$. Since $\delta_i(t_i, a_i, u_i)\sigma$ has a smaller proof length than the one of $\delta(t, a, u)$, the induction hypothesis (6) implies $\delta_i(t_i\tau', a_i\sigma, u') \in RTh(SP)$ and $u_i\sigma \approx u'$ for some $u' \in NF_\Sigma$. Since u_i is strongly normal, u' is normal and $u_i\sigma \approx u'$, Lemma 8.4 implies $u_i\vartheta = u'$ and $\sigma \approx \vartheta$ for some $\vartheta : X \to NF_\Sigma$. Define $x\tau' = x\vartheta$ for all $x \in var(u_i) \setminus (V_{i-1} \cup var(a_i))$. Since $(V_{i-1} \cup var(a_i)) \cap var(u_i) = \emptyset$, $u_i\theta = u'$ implies $u_i\tau' = u'$. Hence by (b), $\delta_i(t_i\tau', a_i\sigma, u') \in RTh(SP)$ implies (c).

Define $\eta : X \to NF_\Sigma$ by $x\eta = x\tau'$ for all $x \in V_{i-1} \cup var(a_i, u_i)$ and $x\eta = x\sigma$ otherwise. By (e) for $i-1$, $x\sigma \approx x\tau' = x\eta$ for all $x \in V_{i-1}$. By (b), $x\sigma = x\tau' = x\eta$ for all $x \in var(a_i)$. Since $x\sigma \approx x\vartheta = x\tau' = x\eta$ for all $x \in var(u_i) \setminus (V_{i-1} \cup var(a_i))$, we conclude $\sigma \approx \eta$. Hence by (7), $G_i\sigma \in RTh(SP)$ implies $G_i\tau'' \in RTh(SP)$ for some $\tau'' : X \to NF_\Sigma$ with $\sigma \approx \tau'' =_{out(G_i)} \eta$. Define $x\tau' = x\tau''$ for all $x \in var(G_i) \setminus (V_{i-1} \cup var(a_i, u_i))$. Since $out(G_i) \cap (V_{i-1} \cup var(a_i, u_i)) = \emptyset$, we have $x\tau'' = x\eta = x\tau'$ for all $x \in var(G_i) \cap (V_{i-1} \cup var(a_i, u_i))$. Hence $G_i\tau'' \in RTh(SP)$ implies (d). Moreover, for all $x \in var(G_i) \setminus (V_{i-1} \cup var(a_i, u_i))$, $x\sigma \approx x\tau'' = x\tau'$, and for all $x \in var(u_i) \setminus (V_{i-1} \cup var(a_i))$, $x\sigma \approx x\vartheta = x\tau'$. Hence $V_i = V_{i-1} \cup var(a_i, u_i, G_i)$, (e) for $i-1$ and (b) imply (e).

(c) for all $0 \le i \le n$ and (d) for all $1 \le i \le n$ imply $\varphi\tau' \in RTh(SP)$. Hence $\delta(t_0, a_0, u_0)\tau' \in RTh(SP)$. In case B1 (see above), (a) and (b) for $i = 0$ imply $(t_0, a_0)\tau' = (t_0\tau, a_0\sigma) = (t', a)$. In case B2 (see above), (a) and (b) for $i = 0$ imply $(t_0, a_0)\tau' = (t_0\tau, a_0\sigma) = (c(t_0'\tau), a_0\sigma) = (c(v'), a) = (d(v'), a) \equiv_{SP} (t', a)$. Hence, in both subcases, $\delta(t_0, a_0, u_0)\tau' \in RTh(SP)$ implies $\delta(t', a, u_0\tau') \in RTh(SP)$.

Since $var(u_0) \subseteq V_n$, (e) for $i = n$ implies $x\sigma \approx x\tau'$ for all $x \in var(u_0)$. Since u_0 is normal, (1) implies $u_0\sigma \approx u_0\tau'$. Therefore, the conclusion of (6) holds true for $u' = u_0\tau'$.

This finishes case B of the proof of (6) from which we have already concluded that $\sim_{SP}$ is compatible with the constructors of Σ. Since $\sim_{SP}$ is (zigzag) compatible with Σ_1 and all behavioral equalities and since $Ini(SP)$ satisfies the behavior axioms for Σ, it remains to show the following properties whose proof is part of the proof of [37], Thm. 6.5:

(8) For all functional destructors $f : sw \to s'$, $t \in T_{\Sigma,s}$ and $a \sim_{SP} a' \in T_{\Sigma,w}$, $f(t, a) \sim_{SP} f(t, a')$.

(9) For all relational destructors $r : sw$ and $t \in T_{\Sigma,s}$, $Ini(SP) \models r(t, a) \ \wedge\ a \sim_{SP} a'$ implies $Ini(SP) \models r(t, a')$.

(10) For all transition relations $\delta : sws'$ and $t \in T_{\Sigma,s}$,
$Ini(SP) \models \delta(t, a, u) \ \wedge\ a \sim_{SP} a'$ implies $Ini(SP) \models \delta(t, a', u') \ \wedge\ u \sim_{SP} u'$ for some u'.

(11) $\sim_{SP}$ is (zigzag) compatible with all symbols of $SP \setminus visSP$ that belong neither to the observer nor to the non-observer level of SP. ❏

9 On strong equality and the specification of partial-recursive functions

Definedness predicates or membership predicates in the sense of [26] are well-known from partial data type approaches (cf., e.g., [7, 14, 22]). They are unary and split a carrier set into "defined" values on the one hand and "undefined", "error" or "exception" values on the other hand. A better way for handling partial types is the use of sum types. Totalizing a partial function within a single range causes an enormous specification overhead when several function definitions must be extended to the undefined arguments. This is avoided by introducing a sum supersort whose subsorts keep defined and undefined values separately from each other.

Term models and their proof rules adopt a constructive view of data types that actually enforces the totalization. Non-totalizable functions that arise as solutions of "non-terminating" recursive equations have values as well, e.g. in a hidden sort of infinite computation sequences. "Real" partiality is not in accordance with *design* specifications. Even the most abstract specification should be complete in the sense that each ground term has a normal form and thus represents *some*thing.

Two objects are called **strongly** equal if they are (structurally) equivalent or both undefined. For **weak equality** only one object needs to be undefined. Objects that are strongly equal and defined are called **existentially equal**. Since existential equality does not say anything about undefinedness and weak equality is not compatible with definedness predicates, strong equality is the only equivalence that captures the meaning of definedness and preserves validity. For achieving the latter, the equivalence needs to be a congruence, i.e., compatible with all functions and predicates that may have undefined arguments. It has been shown elsewhere that a function or predicate f is compatible with strong equality if f is **strict** or, more generally, **regular**. Regularity admits "error recovery", i.e., f may map an argument tuple t with an undefined i-th component () to a defined value. But then all tuples t' differing from t only in the $i-th$ component must be mapped to the same value, i.e., the respective Herbrand model satisfies

$$f(x_1,\dots,x_{i-1},(),x_i,\dots,x_n) \equiv y \wedge y \not\equiv () \quad\Rightarrow\quad f(x_1,\dots,x_{i-1},x,x_i,\dots,x_n) \equiv y.$$

Typical regular functions are Boolean operators and conditionals. For a predicate r, the condition reads as follows:

$$r(x_1,\dots,x_{i-1},(),x_i,\dots,x_n) \quad\Rightarrow\quad r(x_1,\dots,x_{i-1},x,x_i,\dots,x_n) \equiv y.$$

The question arises whether strong equality is a particular behavioral equality, induced by particular observers. If so, we need not establish special criteria ensuring that a function or predicate f is compatible with strong equality, but only demand that the axioms for f are coinductive (see [37]). In fact, a functional specification $SP = (\Sigma, AX)$ with definedness predicates can be extended in a such way that strong equality is a behavioral one, induced by a destructor that identifies all exceptions, but leaves the defined values unchanged (up to renaming).

Let $S' \subseteq S$ be a set of visible sorts such that each S'-sorted normal form denotes either a "defined element" or an "exception". Suppose that the distinction originates from a set EC of exception constructors, definedness predicates $Def : s$, $s \in S$, and the following axioms for Def:

$$Def(c(x_1,\dots,x_n)) \quad\Leftarrow\quad Def(x_{i_1}) \wedge \dots \wedge Def(x_{i_k}) \qquad \text{for all } c : s_1 \dots s_n \to s \in \Sigma \setminus EC$$

where $x_{i_1}, \ldots, x_{i_k}$ are the S'-sorted variables among $x_1, \ldots, x_n$. Note that these axioms imply the strictness of c because $Ini(SP)$ satisfies the inverse implication

$$Def(c(x_1, \ldots, x_n)) \quad \Rightarrow \quad Def(x_{i_1}) \wedge \cdots \wedge Def(x_{i_k}).$$

All $s \in S'$ are regarded as hidden sorts, and for each sort $s \in S$, we augment Σ with a destructor $copy_s : s \to 1 + s$ that is supposed to map each "defined" element (= normal form over $\Sigma \setminus EC$) t to its copy in $NF_{\Sigma,1+s}$ and each "exception" to (). This is accomplished by providing, for each $c : s_1 \ldots s_n \to s \in \Sigma \setminus EC$, a defined function $eval_c : s_1 \ldots s_n \to 1 + s$, specified by the axiom $eval_c(x) \equiv (c(x))$. In contrast to c, $eval_c$ or, more precisely, $eval_{c,+}$ (cf. Section 2) propagates exceptions. The axioms for $copy_s$ read as follows:

$$\begin{array}{ll}
copy_s(c(x)) \equiv () & \text{for all constructors } c : w \to s \in EC, \\
copy_s(c(x_1, \ldots, x_n)) \equiv eval_{c,+}(copy_{s_1}(x_1), \ldots, copy_{s_n}(x_n)) & \text{for all constructors } c : s_1 \ldots s_n \to s \in \Sigma \setminus EC \\
 & \text{if } s \in S', \\
copy_s(x) \equiv (x) & \text{if } s \notin S'.
\end{array}$$

As part of SP, strong equality can be specified as a ν-predicate by the following co-Horn axioms:

$$\begin{array}{l}
x \sim_s y \quad \Rightarrow \quad (Def(x) \quad \Rightarrow \quad x \equiv y), \\
x \sim_s y \quad \Rightarrow \quad (Def(y) \quad \Rightarrow \quad x \equiv y) \quad \text{for all } s \in S'.
\end{array}$$

As part of the described extension of SP, strong equality is the greatest solution of the behavior axioms:

$$x \sim_s y \quad \Rightarrow \quad copy_s(x) \equiv copy_s(y) \quad \text{for all } s \in S'.$$

Hence Theorem 8.5 ensures that strong equality is a weak congruence provided that SP satisfies the assumptions of the theorem.

Let us close this section on partiality with a schema for specifying an arbitrary partial-recursive function $f : w \to s$. Suppose that f is presented by a set of recursive equations or, more generally, by a set AX_f of Horn clauses of the form $f(t) \equiv u \Leftarrow \varphi$. The usual model of f is the *supremum* $\sqcup_{i \in \mathbb{N}} f_i$ of *approximations* f_i of f. These can be specified in terms of the exception monad [29], as abstractions of a defined function $f' : nat \times w \to 1 + s$ with the following Horn axioms:

$$\begin{array}{rcl}
f'(0, x) & \equiv & (), \\
f'(suc(i), t) & \equiv & x_1 \leftarrow f'(i, t_1);\ \ldots; x_n \leftarrow f'(i, t_n);\ (u[x_i/f(t_i) \mid 1 \le i \le n]) \quad \Leftarrow \quad \varphi[x_i/f(t_i) \mid 1 \le i \le n]
\end{array}$$

for all $(f(t) \equiv u \Leftarrow \varphi) \in AX_f$ where $f(t_1), \ldots, f(t_n)$ are the subterms of u or φ with leading function f. f itself is specified in a further extension:

$spec(f)$ = specification of f' `then`

`defuncts`	$f : w \to s$
`vars`	$i : nat \quad x : w \quad y : s$
`Horn axioms`	$f(x) \equiv y \quad \Leftarrow \quad f'(i, x) \equiv (y)$
	$f(x) \equiv () \quad \Leftarrow \quad \forall i : f'(i, x) \equiv ()$

With the help of consistency criteria like Thm. 7.3(5) it is easy to show that $spec(f)$ is consistent with respect to the specification of f' provided that all functions and predicates occurring in AX_f except f are regular.

References

[1] E. Astesiano, M. Broy, G. Reggio, *Algebraic Specification of Concurrent Systems*, in [3]

[2] P. Aczel, *An Introduction to Inductive Definitions*, in: J. Barwise, ed., Handbook of Mathematical Logic, North-Holland (1977) 739-781

[3] E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner, eds., *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Report, Springer 1999

[4] M. Bidoit, R. Hennicker, *Proving the Correctness of Behavioural Implementations*, Proc. AMAST '95, Springer LNCS 936 (1995) 152-168

[5] M. Bidoit, R. Hennicker, *Observer Complete Definitions are Behaviourally Coherent*, Report, University of Munich (1999)

[6] M. Bidoit, P.D. Mosses, *CASL User Manual*, Springer LNCS 2900 (2004)

[7] M. Broy, M. Wirsing, *Partial Abstract Types*, Acta Informatica 18 (1982) 47-64

[8] G. Costa, G. Reggio, *Specification of Abstract Dynamic Data Types: A Temporal Logic Approach*, Theoretical Computer Science 173 (1997) 513-554

[9] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martî-Oliet, J. Meseguer, J.F. Quesada, *A Maude Tutorial*, SRI International 2000, http://maude.csl.sri.com

[10] R. Diaconescu, K. Futatsugi, *CafeOBJ Report*, World Scientific 1998

[11] H. Ehrig, H.J. Kreowski, *Refinement and Implementation*, in [3]

[12] H. Ehrig, H.-J. Kreowski, B. Mahr, P. Padawitz, *Algebraic Implementation of Abstract Data Types*, Theoretical Computer Science 20 (1982) 209-263

[13] H. Ehrig, B. Mahr, *Fundamentals of Algebraic Specification 1*, Springer 1985

[14] J.A. Goguen, *Stretching First Order Equational Logic: Proofs with Partiality, Subtypes and Retracts*, UCSD Report, San Diego 1997, www-cse.ucsd.edu/users/goguen/ps/ftp97.ps.gz

[15] J.A. Goguen, R. Diaconescu, *An Oxford Survey of Order Sorted Algebra*, Mathematical Structures in Computer Science 4 (1994) 363-392

[16] J.A. Goguen, J.W. Thatcher, E.G. Wagner, *An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types*, in: R. Yeh, ed., Current Trends in Programming Methodology 4, Prentice-Hall (1978) 80-149

[17] J. Guttag, E. Horowitz, D.R. Musser *Abstract Data Types and Software Validation*, Report ISI/RR-76-48, University of Southern California 1976

[18] R. Hennicker, A. Kurz, *(Ω,Ξ)-Logic: On the Algebraic Extension of Coalgebraic Specifications*, Proc. CMCS '99, Elsevier ENTCS 19 (1999) 195-211

[19] C.A.R. Hoare, *Proof of Correctness of Data Representations*, Acta Informatica 1 (1972) 271-281

[20] B. Jacobs, *Invariants, Bisimulations and the Correctness of of Coalgebraic Refinements*, Proc. AMAST '97, Springer LNCS 1349 (1997) 267-291

[21] B. Jacobs, *Behaviour-Refinement of Coalgebraic Specifications with Coinductive Correctness Proofs*, Proc. TAPSOFT '97, Springer LNCS 1214 (1997) 787-802

[22] U. Kühler, C.-P. Wirth, *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving*, Proc. RTA '97, Springer LNCS 1232 (1997) 38-52

[23] B. Mahr, J.A. Makowsky, *Characterizing Specification Languages Which Admit Initial Semantics*, Theoretical Computer Science 31 (1984) 49-59

[24] G. Malcolm, J.A. Goguen, *Proving Correctness of Refinement and Implementation*, Technical Monograph PRG-114, Oxford University Computing Lab 1994

[25] E.G. Manes, M.A. Arbib, *Algebraic Approaches to Program Semantics*, Springer 1986

[26] J. Meseguer, *Membership Algebra as a Logical Framework for Equational Specification*, Proc. WADT '97, Springer LNCS 1376 (1998) 18-61

[27] D. Miller, G. Nadathur, F. Pfenning, A. Scedrov, *Uniform Proofs as a Foundation for Logic Programming*, Annals of Pure and Applied Logic 51 (1991) 125-157

[28] B. Möller, A. Tarlecki, M. Wirsing, *Algebraic Specifications of Reachable Higher-Order Algebras*, Proc. 5th ADT Workshop, Springer LNCS 332 (1988) 154-169

[29] E. Moggi, *Notions of Computation and Monads*, Information and Computation 93 (1991) 55-92

[30] Till Mossakowski, Horst Reichel, Markus Roggenbach, Lutz Schröder, *Algebraic-coalgebraic specification in CoCASL*, Proc. WADT 2002, Springer LNCS 2755 (2003) 376-392

[31] M. Müller-Olm, D. Schmidt, B. Steffen, *Model Checking: A Tutorial Introduction*, Proc. SAS '99, Springer LNCS 1694 (1999) 330-394

[32] P. Padawitz, *Computing in Horn Clause Theories*, Springer 1988

[33] P. Padawitz, *Deduction and Declarative Programming*, Cambridge University Press 1992

[34] P. Padawitz, *Inductive Theorem Proving for Design Specifications*, J. Symbolic Computation 21 (1996) 41-99

[35] P. Padawitz, *Proof in Flat Specifications*, in [3]

[36] P. Padawitz, *Towards the One-Tiered Design of Data Types and Transition Systems*, Proc. WADT '97, Springer LNCS 1376 (1998) 365-380

[37] P. Padawitz, *Swinging Types = Functions + Relations + Transition Systems*, Theoretical Computer Science 243 (2000) 93-165

[38] P. Padawitz, *Swinging Types At Work*, Report, University of Dortmund, ls5-www.cs.uni-dortmund.de/∼peter/BehExa.ps.gz

[39] P. Padawitz, *Dialgebraic Specification and Modeling*, Report, University of Dortmund, ls5-www.cs.uni-dortmund.de/∼peter/Dialg.ps

[40] P. Padawitz, *Expander2: A Formal Methods Presenter and Animator*, ls5-www.cs.uni-dortmund.de/∼peter/Expander2/Expander2.html

[41] P. Padawitz, *Expander2: Towards a Workbench for Interactive Formal Reasoning*, ls5-www.cs.uni-dortmund.de/∼peter/Expander2/Chiemsee.ps

[42] P. Padawitz, *Formale Methoden des Systementwurfs*, Course Notes, University of Dortmund 2002, ls5-www.cs.uni-dortmund.de/∼peter/TdP96.ps.gz

[43] W. Pohlers, *Subsystems of set Theory and Second Order Number Theory*, in: S.R. Buss, ed., Handbook of Proof Theory, Elsevier (1998) 209-335

[44] R. Reiter, *A Logic for Default Reasoning*, Aritificial Intelligence 13 (1980) 81-132

[45] G. Roşu, J.A. Goguen, *Circular Coinduction*, UCSD Report, San Diego 2000, www-cse.ucsd.edu/users/goguen/ps/ccoind.ps.gz

[46] J.J.M.M. Rutten, *Universal Coalgebra: A Theory of Systems*, Report CS-R9652, CWI, SMC Amsterdam 1996

[47] D. Sannella, A. Tarlecki, *Toward Formal Development of Programs from Algebraic Specifications: Implementations Revisited*, Acta Informatica 25 (1988) 233-281

[48] K. Schütte, *Proof Theory*, Springer 1977

[49] M. Wand, *Specifications, Models, and Implementations of Data Abstractions*, Theoretical Computer Science 20 (1982) 3-32

[50] W. Wechler, *Universal Algebra for Computer Scientists*, Springer 1992

[51] M. Wirsing, *Algebraic Specification*, in: J. van Leeuwen, ed., Handbook of Theoretical Computer Science, Elsevier (1990) 675-788