# On Deciding Functional Lists with Sublist Sets

Thomas Wies[1], Marco Muñiz[2], and Viktor Kuncak[3]

[1] Institute of Science and Technology (IST), Austria
wies@ist.ac.at
[2] University of Freiburg, Germany
muniz@informatik.uni-freiburg.de
[3] EPFL, Switzerland
viktor.kuncak@epfl.ch

**Abstract.** Motivated by the problem of deciding verification conditions for the verification of functional programs, we present new decision procedures for automated reasoning about functional lists. We first show how to decide in NP the satisfiability problem for logical constraints containing equality, constructor, selectors, as well as the transitive sublist relation. We then extend this class of constraints with operators to compute the set of all sublists, and the set of objects stored in a list. Finally, we support constraints on sizes of sets, which gives us the ability to compute list length as well as the number of distinct list elements. We show that the extended theory is reducible to the theory of sets with linear cardinality constraints, and therefore still in NP. This reduction enables us to combine our theory with other decidable theories that impose constraints on sets of objects, which further increases the potential of our decidability result in verification of functional and imperative software.

## 1 Introduction

Specifications using high-level data types, such as sets and algebraic data types have proven effective for describing the behavior of functional and imperative programs [13, 25]. Functional lists are particularly convenient and widespread in both programs and specifications. Efficient decision procedures for reasoning about lists can therefore greatly help automate software verification tasks.

Theories that allow only constructing and decomposing lists correspond to term algebras and have efficient decision procedures for quantifier-free fragments [1, 15]. However, these theories do not support list concatenation or sublists. Adding list concatenation makes the logic difficult because it subsumes the existential problem for word equations [6, 12, 18], which has been well-studied and is known to be difficult.

This motivates us to use as a starting point the logic of lists with a sublist (suffix) relation, which can express some (even if not all) of the properties expressible using list concatenation. We give an axiomatization of this theory where quantifiers can be instantiated in a complete and efficient way, following the methodology of local theory extensions [19]. Although local theory extensions have been applied to term algebras with certain recursive functions [20], they have not been applied to term algebras in the presence of the sublist operation. The general subterm relation in term algebras

was shown to be in NP using different techniques [22], without discussion of practical implementation procedures and without support for set operators. Several expressive logics of list-like imperative structures have been proposed [3, 11]. In these logics, variables range over graph nodes, as opposed to lists viewed as terms. In other words, the theories that we consider have an additional extensionality axiom, which ensures that no two list objects in the universe have identical tail and head. This axiom has non-trivial consequences on the set of satisfiable formulas and requires a new decision procedure.

**Contributions.** We summarize the contributions of our paper as follows:

- We give a set of local axioms for lists with sublist relation that admits efficient implementation in the spirit of [11] and can leverage general implementation methods for local theory extensions [4].
- We show how to extend this theory with an operator to compute the longest common suffix of two lists. We also give local axioms that give the decision procedure for the extended logic.
- We show how to further extend the theory by defining sets of elements that correspond to all sublists of a list, and then stating set algebra and size operations on such sets. Using a characterization of the models of this theory, we establish that the theory admits a reduction to the logic BAPA of sets with cardinality constraints [9, 10]. We obtain a decidable logic that supports reasoning about the contents of lists as well as about the number of elements in the list.

**Impact on verification tools.** We have found common functions in libraries of functional programming languages that can be verified to meet a detailed specification using our logic. We discuss several examples in the paper. Moreover, the reduction to BAPA makes it possible to combine this logic with a number of other BAPA-reducible logics [17, 21, 23, 24]. Therefore, we believe that our logic will be a useful component of verification tools in the near future.

## 2 Examples

We describe our contributions through two examples written in a notation similar to the Scala programming language [14]. In each example we show how we use our decision procedure to verify functional correctness of a Scala function that manipulates functional lists. Throughout this section we use the term sublist for a suffix of a list.

**Example: dropping elements from a list.** Our first example, listed in Figure 1, is the function drop of the List class in the Scala standard library (such functions also occur in standard libraries for other functional languages, such as Haskell). The function takes as input an integer number n and a parametrized functional list xs. The function returns a functional list zs which is the sublist obtained from xs after dropping the initial n elements.

The **ensuring** statement specifies the postcondition of the function (a precondition is not required). The postcondition is expressed in our logic $\mathsf{FLS}^2$ of functional lists with sublist sets shown in Figure 8. It states that (1) zs is a sublist of the input list xs, denoted by $\mathsf{zs} \preceq \mathsf{xs}$, and (2) if the input n is a positive number and smaller than the length of xs then the length of zs is equal to the length of xs discounting the n dropped elements.

```
def drop[T](n: Int, xs: List[T]): List[T] returning (zs) = {
  if (n ≤ 0) xs
  else xs match {
    case nil ⇒ nil
    case cons(x, ys) ⇒ drop(n−1, ys)
  }
} ensuring (zs ≼ xs ∧ (n ≥ 0 ∧ length(xs) ≥ n → length(zs) = length(xs) − n))
```

Fig. 1: Function drop that drops the first n elements of a list xs

$$\begin{array}{l} \mathsf{n} > 0 \ \wedge \ \mathsf{xs} \neq \mathsf{nil} \ \wedge \ \mathsf{cons}(\mathsf{x}, \mathsf{ys}) = \mathsf{xs} \ \wedge \ \mathsf{zs} \preceq \mathsf{ys} \ \wedge \\ (\mathsf{n} - 1 \geq 0 \wedge \mathsf{length}(\mathsf{ys}) \geq \mathsf{n} - 1 \ \rightarrow \mathsf{length}(\mathsf{zs}) = \mathsf{length}(\mathsf{ys}) - (\mathsf{n} - 1)) \rightarrow \\ \quad \underbrace{\mathsf{zs} \preceq \mathsf{xs}}_{G_1} \wedge \underbrace{(\mathsf{n} \geq 0 \wedge \mathsf{length}(\mathsf{xs}) \geq \mathsf{n} \rightarrow \mathsf{length}(\mathsf{zs}) = \mathsf{length}(\mathsf{xs}) - \mathsf{n})}_{G_2} \end{array}$$

Fig. 2: One of the verification conditions for the function drop

**Deciding verification conditions.** To verify the correctness of the drop function, we generate verification conditions and use our decision procedure to decide their validity. Figure 2 shows one of the generated verification conditions. This verification condition corresponds to the case when n is greater than 0 and xs is not the empty list. It is expressed in our logic of Functional Lists with Sublists and Sets ($\mathsf{FLS}^2$).

We further split the verification condition into two subgoals $G_1$ and $G_2$ (see Figure 2). Each subgoal corresponds to one of the conjuncts in the postcondition of function drop. For proving subgoal $G_1$ we only need to reason about lists and sublists, but not about their lengths. We can prove this subgoal using the decision procedure for the simpler theory of Functional Lists with Sublists (FLS) that we present in Section 6. In the following, we concentrate on the more interesting subgoal $G_2$.

Subgoal $G_2$ can be proved using the $\mathsf{FLS}^2$ decision procedure presented in Section 7. The theory $\mathsf{FLS}^2$ is a combination of the theory FLS and the theory of sets with linear cardinality constraints (BAPA) [9]. Our decision procedure follows the methodology of [23] that enables the combination of such set-sharing theories via reduction to BAPA. Figure 3 illustrates how this decision procedure proves subgoal $G_2$. We first negate the subgoal and then eliminate the length function. For every list xs we encode its length $\mathsf{length}(\mathsf{xs})$ using sublist sets as follows. We introduce a set variable $Xs$ and define it as the set of all sublists of xs: $Xs = \{l.\, l \preceq \mathsf{xs}\}$. We then introduce an integer variable $xs\_length$ that denotes the length of xs by defining $xs\_length = \mathsf{card}(Xs) - 1$, where $\mathsf{card}(Xs)$ denotes the cardinality of set $Xs$. Note that we have to subtract 1, since nil is also a sublist of xs. We then purify the resulting formula and separate it into two conjuncts for the individual fragments. These two conjuncts are depicted in Figure 3. The two separated conjuncts share the set variables $Xs$, $Ys$, and $Zs$. After the separation the underlying decision procedure of each fragment computes a projection of the corresponding conjunct onto the shared set variables. These projections are the strongest BAPA consequence that are expressible over the shared sets. After the projections have been computed, we check satisfiability of their conjunction using the BAPA decision procedure. In our example the conjunction of the two projections is unsatis-

FLS fragment:
$Xs = \{l.\, l \preceq \mathsf{xs}\} \wedge Ys = \{l.\, l \preceq \mathsf{ys}\} \wedge Zs = \{l.\, l \preceq \mathsf{zs}\} \wedge$
$\mathsf{xs} \neq \mathsf{nil} \wedge \mathsf{cons}(\mathsf{x}, \mathsf{ys}) = \mathsf{xs} \wedge \mathsf{zs} \prec \mathsf{ys}$
Projection onto shared sets $Xs$, $Ys$, $Zs$:
$Zs \subseteq Ys \ \wedge \ Ys \subseteq Xs \ \wedge \ \mathsf{card}(Xs) > 1 \ \wedge \ \mathsf{card}(Xs) = \mathsf{card}(Ys) + 1$

BAPA fragment:
$xs\_length = \mathsf{card}(Xs) - 1 \ \wedge \ ys\_length = \mathsf{card}(Ys) - 1 \ \wedge \ zs\_length = \mathsf{card}(Zs) - 1 \wedge$
$\mathsf{n} > 0 \ \wedge \ (\mathsf{n} - 1 \geq 0 \ \wedge \ ys\_length \geq \mathsf{n} - 1 \rightarrow zs\_length = ys\_length - (\mathsf{n} - 1)) \wedge$
$\mathsf{n} \geq 0 \ \wedge \ xs\_length \geq \mathsf{n} \ \wedge \ zs\_length \neq xs\_length - \mathsf{n}$
Projection onto shared sets $Xs$, $Ys$, $Zs$: $\mathsf{card}(Xs) \neq \mathsf{card}(Ys) + 1$

Fig. 3: Separated conjuncts for the negated subgoal $G_2$ of the VC in Figure 2 with the projections onto shared sets

```
def gcs[T](xs: List[T], lxs: Int, ys: List[T], lys: Int): (List[T], Int) returning (zs, lzs) =
require (length(xs)=lxs ∧ length(ys)=lys)
  (xs,ys) match {
    case (nil, _) ⇒ (nil, 0)
    case (_, nil) ⇒ (nil, 0)
    case (cons(x, x1s), cons(y, y1s)) ⇒
      if (lxs > lys) gcs(x1s, lxs−1, ys, lys)
      else if (lxs < lys) gcs(xs, lxs, y1s, lys−1)
      else {
        val (z1s, lz1s) = gcs(x1s, lxs−1, y1s, lys−1)
        if (x = y ∧ lz1s = (lxs − 1)) (cons(x, z1s), lz1s+1) else (z1s, lz1s)
      }
  } ensuring (length(zs) = lzs ∧ zs = xs ⊓ ys)
```

Fig. 4: Function gcs that computes the greatest common suffix of two lists

fiable, which proves that $G_2$ is valid. In Section 7 we describe how to construct these projections onto set variables for the $\mathsf{FLS}^2$ theory.

**Example: greatest common suffix.** Figure 4 shows our second example, a Scala function gcs, which takes as input two functional lists xs, ys and their corresponding lengths lxs, lys. This precondition is specified by the **require** statement. The function returns a pair (zs,lzs) such that zs is the greatest common suffix of the two input lists and lzs its length. This is captured by the postcondition. Our logic provides the operator xs ⊓ ys that denotes the greatest common suffix of two lists xs and ys. Thus, we can directly express the desired property. Figure 5 depicts two constellations of lists xs, ys, and their greatest common suffix zs that may arise during the computation of gcs.

Figure 6 shows one of the verification conditions that are generated for the function gcs. This verification condition captures the case when the lists xs, ys are not empty, their lengths are equal, their head elements x, y are equal, and lz1s is equal to length(xs)-1. This situation is depicted on the right hand side of Figure 5, i.e., in this case the lists xs and ys are identical. The verification condition can again be split into two subgoals. We focus on subgoal $G_1$. Figure 7 shows the separated conjuncts for this subgoal and their projections onto the shared set variables $Xs$, $Ys$, $Zs$, and $Z1s$. Using the BAPA

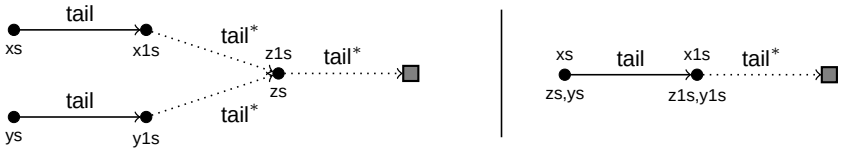


Fig. 5: Lists xs, ys and their greatest common suffix zs

$$\begin{aligned}&\mathsf{length}(\mathsf{xs}) = \mathsf{lxs} \ \wedge \ \mathsf{length}(\mathsf{ys}) = \mathsf{lys} \ \wedge \ \mathsf{xs} \neq \mathsf{nil} \ \wedge \ \mathsf{ys} \neq \mathsf{nil} \ \wedge \ \mathsf{lxs} = \mathsf{lys} \wedge \mathsf{x} = \mathsf{y} \wedge \\ &\mathsf{cons}(\mathsf{x}, \mathsf{x1s}) = \mathsf{xs} \ \wedge \ \mathsf{cons}(\mathsf{y}, \mathsf{y1s}) = \mathsf{ys} \ \wedge \ \mathsf{lz1s} = \mathsf{lxs} - 1 \wedge \\ &\mathsf{length}(\mathsf{z1s}) = \mathsf{lz1s} \ \wedge \ \mathsf{z1s} = \mathsf{xs1} \sqcap \mathsf{y1s} \ \wedge \ \mathsf{zs} = \mathsf{cons}(\mathsf{x}, \mathsf{z1s}) \ \wedge \ \mathsf{lzs} = \mathsf{lz1s} + 1 \ \rightarrow \\ &\quad \underbrace{\mathsf{length}(\mathsf{zs}) = \mathsf{lzs}}_{G_1} \ \wedge \ \underbrace{\mathsf{zs} = \mathsf{xs} \sqcap \mathsf{ys}}_{G_2}\end{aligned}$$

Fig. 6: One of the verification conditions for the function gcs

decision procedure, we can again prove that the conjunction of the two projections is unsatisfiable.

## 3 Logic FLS$^2$ of Functional Lists with Sublists Sets

The grammar of our logic of functional lists with sublist sets is shown in Figure 8. It supports reasoning about lists built from list constructors and selectors, sublists, the length of lists, and cardinality and set algebraic constraints over the sets of sublists of lists $\sigma(l)$ as well their content sets $\tau(l)$.

The remainder of the paper is structured as follows. In Section 5 we first present the fragment FLS of the logic FLS$^2$ that only subsumes formulas over lists and sublists, but not sets, cardinalities, or length constraints. We then formally define the semantics of the logic FLS and give a decision procedure for its satisfiability problem in Section 6. Finally, in Section 7 we show how to use this decision procedure for a BAPA reduction that decides the full logic FLS$^2$.

## 4 Preliminaries

In the following, we define the syntax and semantics of formulas. We further recall the notions of partial structures and local theory extensions from [19].

**Sorted logic.** We present our problem in sorted logic with equality. A *signature* $\Sigma$ is a tuple $(S, \Omega)$, where $S$ is a countable set of sorts and $\Omega$ is a countable set of function symbols $f$ with associated arity $n \geq 0$ and associated sort $s_1 \ldots s_n \rightarrow s_0$ with $s_i \in S$ for all $i \leq n$. Function symbols of arity 0 are called *constant symbols*. We assume that all signatures contain the sort bool. We treat predicates of sort $s_1, \ldots, s_n$ as function symbols of sort $s_1 \times \ldots \times s_n \rightarrow$ bool. Terms are built as usual from the function symbols in $\Omega$ and (sorted) variables taken from a countably infinite set $X$ that is disjoint from $\Omega$. We denote by $t : s$ that term $t$ has sort $s$. A term $t$ is said to be *ground*, if no

FLS fragment:
$Xs = \{l.l \preceq \mathsf{xs}\} \wedge \mathsf{Ys} = \{l.\, l \preceq \mathsf{ys}\} \wedge \mathsf{Zs} = \{l.\, l \preceq \mathsf{zs}\} \wedge \mathsf{Z1s} = \{l.\, l \preceq \mathsf{z1s}\} \wedge$
$\mathsf{cons}(\mathsf{x}, \mathsf{x1s}) = \mathsf{xs} \wedge \mathsf{cons}(\mathsf{y}, \mathsf{y1s}) = \mathsf{ys} \wedge \mathsf{xs} \neq \mathsf{nil} \wedge \mathsf{x} = \mathsf{y} \wedge \mathsf{ys} \neq \mathsf{nil} \wedge$
$\mathsf{z1s} = \mathsf{x1s} \sqcap \mathsf{y1s} \wedge \mathsf{zs} = \mathsf{cons}(\mathsf{x},\mathsf{z1s})$
Projection onto shared sets $Xs$, $Ys$, $Zs$, $Z1s$:
$\mathsf{card}(Xs) > 1 \wedge \mathsf{card}(Ys) > 1 \wedge \mathsf{card}(Zs) > 1 \wedge$
$Z1s \subseteq Zs \wedge \mathsf{card}(Zs) = \mathsf{card}(Z1s) + 1 \wedge$
$((\mathsf{card}(Z1s) = \mathsf{card}(Xs) - 1 \vee \mathsf{card}(Z1s) = \mathsf{card}(Ys) - 1) \rightarrow Zs = Xs = Ys)$

BAPA fragment:
$xs\_length = \mathsf{card}(Xs) - 1 \wedge ys\_length = \mathsf{card}(Ys) - 1 \wedge$
$zs\_length = \mathsf{card}(Zs) - 1 \wedge z1s\_length = \mathsf{card}(Z1s) - 1 \wedge$
$xs\_length = \mathsf{lxs} \wedge ys\_length = \mathsf{lys} \wedge z1s\_length = \mathsf{lz1s} \wedge$
$\mathsf{lxs} = \mathsf{lys} \wedge \mathsf{lz1s} = \mathsf{lxs} - 1 \wedge \mathsf{lzs} = \mathsf{lz1s} + 1 \wedge zs\_length \neq \mathsf{lzs}$
Projection onto shared sets $Xs$, $Ys$, $Zs$, $Z1s$:
$\mathsf{card}(Z1s) = \mathsf{card}(Xs) - 1 \wedge \mathsf{card}(Z1s) = \mathsf{card}(Ys) - 1 \wedge \mathsf{card}(Zs) \neq \mathsf{card}(Z1s) + 1$

Fig. 7: Separated conjuncts for the negated subgoal $G_1$ of the VC in Figure 6 with projections onto the shared sets

$$
\begin{aligned}
F &::= A_L \mid A_S \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \\
A_L &::= T_L \preceq T_L \mid T_L = T_L \mid T_H = T_H \\
T_L &::= v_L \mid \mathsf{nil} \mid \mathsf{cons}(T_H, T_L) \mid \mathsf{tail}(T_L) \mid T_L \sqcap T_L \\
T_H &::= v_H \mid \mathsf{head}(T_L) \\
A_S &::= B_L = B_L \mid B_L \subseteq B_L \mid T_I = T_I \mid T_I < T_I \\
B_L &::= s_L \mid \emptyset \mid \{T_L\} \mid \sigma(T_L) \mid B_L \cup B_L \mid B_L \setminus B_L \\
B_H &::= s_H \mid \emptyset \mid \{T_H\} \mid \tau(T_L) \mid \mathsf{head}[B_L] \mid B_H \cup B_H \mid B_H \setminus B_H \\
T_I &::= v_I \mid K \mid T_I + T_I \mid K \cdot T_I \mid \mathsf{card}(B_L) \mid \mathsf{card}(B_H) \mid \mathsf{length}(T_L) \\
K &::= \ldots -2 \mid -1 \mid 0 \mid 1 \mid 2 \ldots
\end{aligned}
$$

Fig. 8: Logic $\mathsf{FLS}^2$ of lists, sublist, sublist sets, list contents, and size constraints

variable appears in $t$. We denote by $\mathsf{Terms}(\Sigma)$ the set of all ground $\Sigma$-terms. A $\Sigma$-formula is built from propositional operations that connect terms of sort bool. We write $\forall x : s.F$ to denote a universally quantified formula where the quantified variable has sort $s$ (analogously for $\exists x : s.F$). We further define atoms, literals, and clauses as usual. We denote by $\mathsf{FV}(F)$ the set of all variables that occur free in $F$. The equality symbol applies only to terms of the same sort. We can assume to have a distinct equality symbol for each sort of interest, but we use the same symbol $=$ to denote all of them.

**Total and partial structures.** Given a signature $\Sigma = (S, \Omega)$, a *partial $\Sigma$-structure* $\alpha$ is a function that maps each sort $s \in S$ to a non-empty set $\alpha(s)$ and each function symbol $f \in \Omega$ of sort $s_1 \times \cdots \times s_n \rightarrow s_0$ to a partial function $\alpha(f) : \alpha(s_1) \times \cdots \times \alpha(s_n) \rightharpoonup \alpha(s_0)$. We assume that all partial structures interpret the sort bool by the two-element set of Booleans $\{\mathsf{true}, \mathsf{false}\}$. A partial structure $\alpha$ is called *total structure* or simply *structure* if it interprets all function symbols by total functions.

Given a total structure $\alpha$ and a *variable assignment* $\beta : X \rightarrow \alpha(S)$, the evaluation $[\![t]\!]_{\alpha,\beta}$ of a term $t$ (respectively a formula) in $\alpha, \beta$ is defined as usual. In particular, we

use the standard interpretations for the equality symbol and propositional connectives of classical logic. A quantified variable of sort $s$ ranges over all elements of $\alpha(s)$. The notions of satisfiability, validity, and entailment of formulas, clauses, and sets of clauses in total structures are also defined as usual. We write $\alpha, \beta \models F$ if $\alpha$ satisfies $F$ under $\beta$ where $F$ is a formula, a clause, or a set of clauses. Similarly, we write $\alpha \models F$ if $F$ is valid in $\alpha$. In this case we also call $\alpha$ a *model* of $F$.

The interpretation $[\![t]\!]_{\alpha,\beta}$ of a term $t$ in a partial structure $\alpha$ is as for total structures, except that if $t = f(t_1, \ldots, t_n)$ then $[\![t]\!]_{\alpha,\beta}$ is undefined if either $[\![t_i]\!]_{\alpha,\beta}$ is undefined for some $i$, or $f \in \Omega$ and $([\![t_1]\!]_{\alpha,\beta}, \ldots, [\![t_n]\!]_{\alpha,\beta})$ is not in the domain of $\alpha(f)$. We say that a partial structure $\alpha$ *weakly satisfies* a literal $L$ under $\beta$, written $\alpha, \beta \models_w L$, if either $[\![L]\!]_{\alpha,\beta}$ is undefined or $[\![L]\!]_{\alpha,\beta} = \mathsf{true}$. The notion of weak satisfiability is extended to clauses and sets of clauses as for total structures.

**Theories and local theory extensions.** A *theory* $\mathcal{T}$ for a signature $\Sigma$ is simply a set of $\Sigma$-formulas. We consider theories $\mathcal{T}(\mathcal{M})$ defined as a set of $\Sigma$-formulas that are valid in a given set of models $\mathcal{M}$, as well as theories $\mathcal{T}(\mathcal{K})$ defined as a set of $\Sigma$-formulas that are consequences of a given set of formulas $\mathcal{K}$. In the latter case, we call $\mathcal{K}$ the *axioms* of the theory $\mathcal{T}(\mathcal{K})$ and we often identify $\mathcal{K}$ and $\mathcal{T}(\mathcal{K})$. In particular, we call $\mathcal{K} = \emptyset$ the *empty theory*. The *decision problem* for a theory $\mathcal{T}$ is to decide whether a given $\Sigma$-formula $F$ belongs to $\mathcal{T}$.

In what follows, we consider theories that are defined by a set of axioms. Let $\Sigma_0 = (S, \Omega_0)$ be a signature and assume that signature $\Sigma_1 = (S, \Omega_0 \cup \Omega_1)$ extends $\Sigma_0$ by new function symbols $\Omega_1$. We call the function symbols in $\Omega_1$ *extension symbols* and terms starting with extension symbols *extension terms*. Now, a theory $\mathcal{T}_1$ over $\Sigma_1$ is an *extension* of a theory $\mathcal{T}_0$ over $\Sigma_0$, if $\mathcal{T}_1$ is obtained from $\mathcal{T}_0$ by adding a set of (universally quantified) clauses $\mathcal{K}$. In the following, when we refer to a set of ground clauses $G$, we assume they are over the signature $\Sigma_1^c = (S, \Omega_0 \cup \Omega_1 \cup \Omega_c)$ where $\Omega_c$ is a set of new constant symbols. Let $\mathcal{K}$ be a set of (universally quantified) clauses. We denote by $\mathsf{st}(\mathcal{K}, G)$ the set of all ground terms that appear in $\mathcal{K}$ or $G$ and by $\mathcal{K}[G]$ the set of all instantiations of clauses in $\mathcal{K}$ where variables appearing below extension terms have been instantiated by the terms in $\mathsf{st}(\mathcal{K}, G)$. Then an extension $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is a *local extension* if it satisfies condition (Loc)[4]:

(Loc) For every finite set of ground clauses $G$, $G \cup \mathcal{T}_1 \models \mathsf{false}$ iff there is no partial $\Sigma_1^c$-structure $\alpha$ such that $\alpha_{|\Sigma_0}$ is a total model of $\mathcal{T}_0$, all terms in $\mathsf{st}(\mathcal{K}, G)$ are defined in $\alpha$, and $\alpha$ weakly satisfies $\mathcal{K}[G] \cup G$.

## 5 Logic FLS of Funcional Lists with Sublists

We now define the logic of functional lists with sublists (FLS) and its accompanying theory. The logic FLS is given by all quantifier-free formulas over the signature $\Sigma_{\mathsf{FLS}} = (S_{\mathsf{FLS}}, \Omega_{\mathsf{FLS}})$. The signature $\Sigma_{\mathsf{FLS}}$ consists of sorts $S_{\mathsf{FLS}} = \{\mathsf{bool}, \mathsf{list}, \mathsf{data}\}$ and function symbols $\Omega_{\mathsf{FLS}} = \{\mathsf{nil}, \mathsf{cons}, \mathsf{head}, \mathsf{tail}, \sqcap, \preceq\}$. The sorts of the function symbols in $\Omega_{\mathsf{FLS}}$ are shown in Figure 9. We use infix notation for the symbols $\sqcap$ and $\preceq$.

[4] We here use a weaker notion of locality restricted to finite sets of ground clauses $G$.

$$\begin{array}{lll} \mathsf{nil} : \mathsf{list} & \mathsf{tail} : \mathsf{list} \to \mathsf{list} & \preceq : \mathsf{list} \times \mathsf{list} \to \mathsf{bool} \\ \mathsf{cons} : \mathsf{list} \times \mathsf{data} \to \mathsf{list} & \mathsf{head} : \mathsf{list} \to \mathsf{data} & \sqcap : \mathsf{list} \times \mathsf{list} \to \mathsf{list} \end{array}$$

Fig. 9: Sorts of function symbols in the signature $\Sigma_{\mathsf{FLS}}$

$$\begin{aligned} \alpha_{\mathsf{FLS}}(\mathsf{list}) &\stackrel{\text{def}}{=} \mathsf{L} \stackrel{\text{def}}{=} \{\, t \in \mathsf{Terms}(\Sigma_{\mathsf{L}}) \mid t : \mathsf{list} \,\} \\ \alpha_{\mathsf{FLS}}(\mathsf{data}) &\stackrel{\text{def}}{=} \mathsf{D} \stackrel{\text{def}}{=} \{\, t \in \mathsf{Terms}(\Sigma_{\mathsf{L}}) \mid t : \mathsf{data} \,\} \\ \alpha_{\mathsf{FLS}}(\mathsf{cons}) &\stackrel{\text{def}}{=} \mathsf{cons}_{\mathsf{L}} \stackrel{\text{def}}{=} \lambda(l, d).\, \mathsf{cons}(l, d) \\ \alpha_{\mathsf{FLS}}(\mathsf{nil}) &\stackrel{\text{def}}{=} \mathsf{nil} \\ \alpha_{\mathsf{FLS}}(\mathsf{tail}) &\stackrel{\text{def}}{=} \mathsf{tail}_{\mathsf{L}} \stackrel{\text{def}}{=} \lambda l.\, \text{if } l = \mathsf{nil} \text{ then } \mathsf{nil} \text{ else } l' \text{ where } l = \mathsf{cons}(l', d) \\ \alpha_{\mathsf{FLS}}(\mathsf{head}) &\stackrel{\text{def}}{=} \mathsf{head}_{\mathsf{L}} \stackrel{\text{def}}{=} \lambda l.\, \text{if } l = \mathsf{nil} \text{ then } d_1 \text{ else } d \text{ where } l = \mathsf{cons}(l', d) \\ \alpha_{\mathsf{FLS}}(\preceq) &\stackrel{\text{def}}{=} \lambda(l_1, l_2).\, l_1 \preceq_{\mathsf{L}} l_2 \\ \alpha_{\mathsf{FLS}}(\sqcap) &\stackrel{\text{def}}{=} \lambda(l_1, l_2).\, l_1 \sqcap_{\mathsf{L}} l_2 \end{aligned}$$

Fig. 10: The canonical model $\alpha_{\mathsf{FLS}}$ of functional lists with sublists

The theory of functional lists with sublist relationship $\mathcal{T}_{\mathsf{FLS}}$ is the set of all formulas in FLS that are true in the canonical model of lists. We denote this canonical model by $\alpha_{\mathsf{FLS}}$. The structure $\alpha_{\mathsf{FLS}}$ is the term algebra generated by the signature $\Sigma_{\mathsf{L}} = (S_{\mathsf{FLS}}, \{\mathsf{cons}, \mathsf{nil}, d_1, d_2, \dots\})$, where $d_1, d_2, \dots$ are infinitly many constant symbols of sort data. The complete definition of $\alpha_{\mathsf{FLS}}$ is given in Figure 10. The canonical model interprets the sort list as the set of all $\Sigma_{\mathsf{L}}$-terms of sort list. We denote this set by L. Likewise, the sort data is interpreted as the set of all $\Sigma_{\mathsf{L}}$-terms of sort data. We denote this set by D. The function symbols cons and nil are interpreted as the corresponding term constructors. The function symbols head and tail are interpreted as the appropriate selectors $\mathsf{head}_{\mathsf{L}}$ and $\mathsf{tail}_{\mathsf{L}}$. The predicate symbol $\preceq$ is interpreted as the sublist relation $\preceq_L \subseteq \mathsf{L} \times \mathsf{L}$ on lists. The sublist relation is defined as the reflexive transitive closure of the tail selector function:

$$l_1 \preceq_{\mathsf{L}} l_2 \iff^{\text{def}} (l_1, l_2) \in \{\, (l, \mathsf{tail}_{\mathsf{L}}(l)) \mid l \in \mathsf{L} \,\}^*$$

The relation $\preceq_{\mathsf{L}}$ is a partial order on lists. In fact, one can show more.

**Proposition 1.** *The relation $\preceq_{\mathsf{L}}$ induces a meet-semilattice on the set* L.

We denote by $\sqcap_{\mathsf{L}}$ the meet operator of the semilattice induced by $\preceq_{\mathsf{L}}$. Given two lists $l_1$ and $l_2$, the list $l_1 \sqcap_{\mathsf{L}} l_2$ denotes the greatest common suffix of $l_1$ and $l_2$. The structure $\alpha_{\mathsf{FLS}}$ interprets the function symbol $\sqcap$ as the operator $\sqcap_{\mathsf{L}}$.

We further define the theory of all finite substructures of $\alpha_{\mathsf{FLS}}$. Let $\Sigma_{\mathsf{FLSf}}$ be the signature $\Sigma_{\mathsf{FLS}}$ without the function symbol cons and let $\alpha_{\mathsf{FLSf}}$ be the structure $\alpha_{\mathsf{FLS}}$ restricted to the signature $\Sigma_{\mathsf{FLSf}}$. We define $\mathcal{M}_{\mathsf{FLSf}}$ to be the set of all finite total substructures of $\alpha_{\mathsf{FLSf}}$. The theory $\mathcal{T}_{\mathsf{FLSf}}$ is the set of all FLS formulas that are true in all structures $\mathcal{M}_{\mathsf{FLSf}}$.

## 6 Decision Procedure for FLS

In the following, we show that the theory $\mathcal{T}_{\mathsf{FLS}}$ is decidable. For this purpose we reduce the decision problem for $\mathcal{T}_{\mathsf{FLS}}$ to the decision problem of the theory $\mathcal{T}_{\mathsf{FLSf}}$. We then give a finite first-order axiomatization of the theory $\mathcal{T}_{\mathsf{FLSf}}$ and show that it is a local theory extension of the empty theory. In total, this implies that deciding satisfiability of a ground formula $F$ with respect to the theory $\mathcal{T}_{\mathsf{FLS}}$ can be reduced to deciding satisfiability of $F$ conjoined with finitely many ground instances of the first-order axioms of $\mathcal{T}_{\mathsf{FLSf}}$.

### 6.1 Reducing FLS to FLSf

We first note that satisfiability of an FLS formula $F$ in the canonical model can be reduced to checking satisfiability in the finite substructures, if the function symbol cons does not occur in $F$.

**Proposition 2.** *Let $F$ be a quantifier-free $\Sigma_{\mathsf{FLSf}}$-formula. Then $F$ is satisfiable in $\alpha_{\mathsf{FLS}}$ if and only if $F$ is satisfiable in some structure $\alpha \in \mathcal{M}_{\mathsf{FLSf}}$.*

The proof of Proposition 2 is similar to the proof of [7, Theorem 2].

We can now exploit the fact that, in the term algebra $\alpha_{\mathsf{FLS}}$, the constructor $\mathsf{cons}_{\mathsf{L}}$ is uniquely determined by the functions $\mathsf{head}_{\mathsf{L}}$ and $\mathsf{tail}_{\mathsf{L}}$. Let $F$ be an FLS formula. Then we can eliminate an occurrence $F(\mathsf{cons}(t_d, t_l))$ of function symbol cons in a term of $F$ by rewriting $F(\mathsf{cons}(t_d, t_l))$ into:

$$x \neq \mathsf{nil} \wedge \mathsf{head}(x) = t_d \wedge \mathsf{tail}(x) = t_l \wedge F(x)$$

where $x$ is a fresh variable of sort list that does not appear elsewhere in $F$. Let $\mathsf{elim\text{-}cons}(F)$ be the formula that results from rewriting recursively all appearances of function symbol cons in $F$. Clearly, in the canonical model $\alpha_{\mathsf{FLS}}$, the formulas $F$ and $\mathsf{elim\text{-}cons}(F)$ are equisatisfiable. Thus, with Proposition 2 we can conclude.

**Lemma 3.** *Let $F$ be an FLS formula. Then $F$ is satisfiable in $\alpha_{\mathsf{FLS}}$ if and only if $\mathsf{elim\text{-}cons}(F)$ is satisfiable in some structure $\alpha \in \mathcal{M}_{\mathsf{FLSf}}$.*

### 6.2 Axiomatizing FLSf

We next show that there exists a first-order axiomatization $\mathcal{K}_{\mathsf{FLSf}}$ of the theory $\mathcal{T}_{\mathsf{FLSf}}$ that is a local theory extension of the empty theory. The axioms $\mathcal{K}_{\mathsf{FLSf}}$ are given in Figure 11. The free variables appearing in the formulas are implicitly universally quantified. We now explain each of these axioms and argue their soundness, i.e., that each axiom is true in the canonical model $\alpha_{\mathsf{FLS}}$.

The axiom Pure is a logical consequence of the following formula, which is true in the canonical model $\alpha_{\mathsf{FLS}}$: $\forall x.\, \mathsf{cons}(\mathsf{head}(x), \mathsf{tail}(x)) = x \vee \mathsf{nil} = x$. Hence, Pure is true in all finite total substructures of $\alpha_{\mathsf{FLS}}$. The axiom Nil expresses that all lists have nil as a sublist. This axiom is true because all lists are constructed from nil. The axioms Refl, Trans, and AntiSym express that $\preceq$ is a partial order. These axioms follow from Proposition 1. The axiom Total expresses the fact that, for a fixed list $x$, all sublists of

$$\textbf{Pure: } \mathsf{head}(x) = \mathsf{head}(y) \wedge \mathsf{tail}(x) = \mathsf{tail}(y) \rightarrow x = y \vee x = \mathsf{nil} \vee y = \mathsf{nil}$$

$$\begin{array}{rlrl}
\textbf{Nil:} & \mathsf{nil} \preceq x & \textbf{UnfoldL:} & \mathsf{tail}(x) \preceq x \\
\textbf{Refl:} & x \preceq x & \textbf{UnfoldR:} & x \preceq y \rightarrow x = y \vee x \preceq \mathsf{tail}(y) \\
\textbf{Trans:} & x \preceq y \wedge y \preceq z \rightarrow x \preceq z & \textbf{GCS1:} & x \sqcap y \preceq x \\
\textbf{AntiSym:} & x \preceq y \wedge y \preceq x \rightarrow x = y & \textbf{GCS2:} & x \sqcap y \preceq y \\
\textbf{Total:} & y \preceq x \wedge z \preceq x \rightarrow y \preceq z \vee z \preceq y & \textbf{GCS3:} & z \preceq x \wedge z \preceq y \rightarrow z \preceq x \sqcap y
\end{array}$$

Fig. 11: First-order axiomatization $\mathcal{K}_{\mathsf{FLSf}}$ of the theory $\mathcal{T}_{\mathsf{FLSf}}$

$x$ are totally ordered by the sublist relation. This axiom follows from the definition of $\preceq_{\mathsf{L}}$ as the reflexive transitive closure of a functional relation. The axioms UnfoldL and UnfoldR express that, by applying tail to either side of the sublist relation, we stay in the sublist relation. Finally, the axioms GCS1, GCS2, and GCS3 express that $\sqcap$ is the greatest lower bound operator of the partial order $\preceq$. These axioms follow from the definition of $\sqcap_{\mathsf{L}}$ and Proposition 1.

**Lemma 4.** *The axioms $\mathcal{K}_{\mathsf{FLSf}}$ are sound, i.e., for all $\alpha \in \mathcal{M}_{\mathsf{FLSf}}$, $\alpha \models \mathcal{K}_{\mathsf{FLSf}}$.*

As a prerequisite for proving completeness of the axioms, we next show that the finite models of the axioms $\mathcal{K}_{\mathsf{FLSf}}$ are structurally equivalent to the finite substructures of the canonical model of functional lists.

**Proposition 5.** *Every finite model of $\mathcal{K}_{\mathsf{FLSf}}$ is isomorphic to some structure in $\mathcal{M}_{\mathsf{FLSf}}$.*

The proof is in Appendix B.1.

### 6.3 FLSf as a Local Theory Extension

We will now prove that the theory induced by $\mathcal{K}_{\mathsf{FLSf}}$ is a local theory extension of the empty theory and, at the same time, prove that $\mathcal{K}_{\mathsf{FLSf}}$ is a complete axiomatization of the theory $\mathcal{T}_{\mathsf{FLSf}}$.

In what follows, the signature $\Sigma_{\mathsf{FLSf}}$ is the signature of the theory extension $\mathcal{K}_{\mathsf{FLSf}}$. We also have to determine the signature of the base theory by fixing the extension symbols. We treat the function symbols $\Omega_e \stackrel{\text{def}}{=} \{\mathsf{head}, \mathsf{tail}, \sqcap\}$ as extension symbols, but the sublist relation $\preceq$ as a symbol in the signature of the base theory. As in the definition of local theory extensions in Section 4, for a set of ground clauses $G$, we denote by $\mathcal{K}_{\mathsf{FLSf}}[G]$ all instances of axioms $\mathcal{K}_{\mathsf{FLSf}}$ where the variables occurring below extension symbols $\Omega_e$ are instantiated by all ground terms $\mathsf{st}(\mathcal{K}_{\mathsf{FLSf}}, G)$ that appear in $\mathcal{K}_{\mathsf{FLSf}}$ and $G$. Furthermore, we denote by $\Sigma^c_{\mathsf{FLSf}}$ the signature $\Sigma_{\mathsf{FLSf}}$ extended with finitely many new constant symbols $\Omega_c$.

**Lemma 6.** *For every finite set of $\Sigma^c_{\mathsf{FLSf}}$ ground clauses $G$, if $\alpha$ is a partial $\Sigma^c_{\mathsf{FLSf}}$-structure such that $\alpha(\preceq)$ is total, all terms in $\mathsf{st}(\mathcal{K}_{\mathsf{FLSf}}, G)$ are defined in $\alpha$, and $\alpha$ weakly satisfies $\mathcal{K}_{\mathsf{FLSf}}[G] \cup G$ then there exists a finite total $\Sigma^c_{\mathsf{FLSf}}$-structure that satisfies $\mathcal{K}_{\mathsf{FLSf}} \cup G$.*

We sketch the proof of Lemma 6. Let $\alpha$ be a partial $\Sigma^c_{\mathsf{FLSf}}$-structure as required in the lemma. We can obtain a finite partial substructure $\alpha'$ from $\alpha$ by restricting the interpretations of sorts data and list to the elements that are used in the interpretations of the ground terms $\mathsf{st}(\mathcal{K}_{\mathsf{FLSf}}, G)$. Then $\alpha'$ still weakly satisfies $\mathcal{K}_{\mathsf{FLSf}}[G] \cup G$, since all axioms in $\mathcal{K}_{\mathsf{FLSf}}$ are universal. We can then complete $\alpha'$ to a finite total model of $\mathcal{K}_{\mathsf{FLSf}} \cup G$ as follows. First, for every $u \in \alpha'(\mathsf{list})$ where $\alpha'(\mathsf{head})$ is not defined, we can extend $\alpha'(\mathsf{data})$ by a fresh element $d_u$ and define $\alpha'(\mathsf{head})(u) = d_u$. Now, let $u \in \alpha'(\mathsf{list})$ such that $\alpha(\mathsf{tail})$ is not defined on $u$. If $u = \alpha'(\mathsf{nil})$, we define $\alpha'(\mathsf{tail})(u) = u$. Otherwise, from the fact that $\alpha'$ satisfies axioms Nil, AntiSym, and Total we can conclude that there exists a maximal element $v \in \alpha'(\mathsf{list}) \setminus \{u\}$ such that $(v, u) \in \alpha'(\preceq)$. However, we cannot simply define $\alpha'(\mathsf{tail})(u) = v$. The resulting structure would potentially violate axiom Pure. Instead, we extend $\alpha'(\mathsf{list})$ with a fresh element $w$ and $\alpha'(\mathsf{data})$ with a fresh element $d_w$, and define: $\alpha'(\mathsf{head})(w) = d_w$, $\alpha'(\mathsf{tail})(w) = v$, and $\alpha'(\mathsf{tail})(u) = w$. We further extend the definition of $\alpha'(\preceq)$ for the newly added element $w$, as expected. The completion of $\alpha'(\sqcap)$ to a total function is then straightforward.

From Lemma 6 we can now immediately conclude that the theory $\mathcal{K}_{\mathsf{FLSf}}$ satisfies condition (Loc), where the base theory $\mathcal{T}_0$ is given by the empty theory.

**Theorem 7.** *$\mathcal{K}_{\mathsf{FLSf}}$ is a local theory extension of the empty theory.*

Similarly, from Proposition 5 and Lemma 6, we can conclude that the axioms $\mathcal{K}_{\mathsf{FLSf}}$ are complete.

**Theorem 8.** *$\mathcal{K}_{\mathsf{FLSf}}$ is an axiomatization of the theory $\mathcal{T}_{\mathsf{FLSf}}$, i.e., $\mathcal{T}(\mathcal{K}_{\mathsf{FLSf}}) = \mathcal{T}_{\mathsf{FLSf}}$.*

### 6.4 Deciding FLS

Following the reduction scheme for reasoning in local theory extensions [5, 19], we can now give a decision procedure that reduces the decision problem of the theory $\mathcal{T}_{\mathsf{FLS}}$ to the satisfiability problem of the Bernays-Schönfinkel-Ramsey class. The decision procedure is depicted in Figure 12.

Soundness and completeness of the decision procedure follow from Lemma 3, Theorems 7 and 8, and [19, Lemma 4]. Note that the formula $\hat{F}_o$ obtained in step 4 is indeed in the Bernays-Schönfinkel-Ramsey class, since $\hat{F}_0$ contains only relation symbols and equality, and all quantifiers are universal.

**Complexity.** For formulas in the Bernays-Schönfinkel-Ramsey class that have a bounded number of universal quantifiers, the satisfiability problem is known to be NP-complete [2, page 258]. The only quantified variables appearing in the formula $\hat{F}_o$ obtained in step 4 are those in $\mathcal{K}_0$. In fact, we can write $\mathcal{K}_0$ in such a way that it uses exactly 3 quantified variables. Furthermore, the size of the formula $\hat{F}_0$ is polynomial in the size of the input formula $F$.

**Theorem 9.** *The decision problem for the theory $\mathcal{T}_{\mathsf{FLS}}$ is NP-complete.*

We presented the theory $\mathcal{T}_{\mathsf{FLSf}}$ as a local theory extension with extension symbols head, tail, and $\sqcap$. Alternatively, one can also treat the symbol $\preceq$ as an extension symbol, i.e., in this case the signature of the base theory only contains constant symbols. This

**Input:** an FLS formula $F$.

**Step 1:** Compute $\hat{F} = \mathsf{elim\text{-}cons}(\neg F)$, replace all variables in $\hat{F}$ with fresh constant symbols, and transform the result into a set of ground clauses $G$.

**Step 2:** Purify and flatten the set of clauses $\mathcal{K}_{\mathsf{FLSf}}[G] \wedge G$ by recursively replacing each (sub)term $t = f(g_1, \ldots, g_n)$ starting with an extension function $f \in \Omega_e$ by a fresh constant $c_t$. Then introduce fresh constants $c_1, \ldots, c_n$ for the arguments $g_1, \ldots, g_n$ and add corresponding definitions $f(c_1, \ldots, c_n) = c_t$ and $c_i = g_i$ for all $i$. The set of clauses thus obtained has the form $\mathcal{K}_0 \wedge G \wedge D$ where $D$ is a set of ground unit clauses of the form $f(c_1, \ldots, c_n) = c$, where $f \in \Omega_e$, and $c_1, \ldots, c_n, c$ are constants, and $\mathcal{K}_0, G_0$ are clauses without function symbols in $\Omega_e$.

**Step 3:** Represent each function symbol $f \in \Omega_e$ as a partial but functional relation $r_f$, i.e., obtain $D^*$ from $D$ by replacing each literal $f(c_1, \ldots, c_n) = c$ in $D$ by $r_f(c_1, \ldots, c_n, c)$ and introduce corresponding functionality axioms

$$\mathsf{Fun}(D^*) = \{\textstyle\bigwedge_{i=1}^{n} c_i = d_i \wedge r_f(c_1, \ldots, c_n, c) \wedge r_f(d_1, \ldots, d_n, d) \rightarrow c = d \mid \\ f \in \Omega_e, r_f(c_1, \ldots, c_n, c), r_f(d_1, \ldots, d_n, d) \in D^*\}$$

**Step 4:** Check satisfiability of $\hat{F}_0 \equiv \mathcal{K}_0 \wedge G_0 \wedge D^* \wedge \mathsf{Fun}(D^*)$. If $\hat{F}_0$ is satisfiable, return "$F \notin \mathcal{T}_{\mathsf{FLS}}$", otherwise return "$F \in \mathcal{T}_{\mathsf{FLS}}$".

Fig. 12: Decision procedure for FLS

gives an alternative decision procedure where the axioms $\mathcal{K}_{\mathsf{FLSf}}$ are fully instantiated by ground terms and the set $\hat{F}_0$ obtained in step 4 is a set of ground clauses. However, the decision procedure described in Figure 12 can take advantage of the more compact representation of formulas in the Bernays-Schönfinkel-Ramsey class by applying specialized decision procedures for this class, e.g., [16].

## 7 Extension with Sets of Sublists

We next show decidability of the logic that extends FLS with constraints on sets of sublists and the contents of lists. We do this by reducing such extended logic to constraints on sets. For this we need a normal form of formulas in our logic. To show this normal form, we start from $\Sigma^2_{\mathsf{FLSf}}$ partial models, but refine them further to be able to reduce them to constraints on disjoint sets. We then give a BAPA reduction for each of the refined models.

### 7.1 Predecessor-Refined Partial Structures

**Definition 10.** *$\alpha$ is a Predecessor-Refined Partial (PRP) Structure if it is a partial substructure of a structure in $\mathcal{M}_{\mathsf{FLSf}}$ and the following conditions hold in $\alpha$ for all elements $x, y \in \alpha(\mathsf{list})$:*

1. *$x \preceq y$ is totally defined on $\alpha(\mathsf{list})$*
2. *$(x \sqcap y) \in \alpha(\mathsf{list})$. Moreover, if $x, y, (x \sqcap y)$ are three distinct elements, then there exists $x_1 \in \alpha(\mathsf{list})$ such that $x_1 \preceq x$ and $\mathsf{tail}(x_1) = (x \sqcap y)$.*

**Input:** a PRP structure $\alpha$. **Output:** a set constraint $G_\alpha$.

**Step 1:** Define the relation $\preceq_1$ as irreflexive transitive reduct of $\preceq$ without the tail relation. Formally, for all $x, y \in \alpha(\mathsf{list})$, define $x \preceq_1 y$ iff all of the following conditions hold: (1) $x \preceq y$, (2) $x \neq y$, (3) $\mathsf{tail}(y)$ is undefined, and (4) there is no $z$ in $\alpha(\mathsf{list})$ such that $x, y, z$ are distinct, $x \preceq z$, and $z \preceq y$.

**Step 2:** Introduce sets $S_{x,y}$ with the meaning $S_{x,y} = (\sigma(y) \setminus \sigma(x)) \setminus \{y\}$ and define $\mathsf{Segs} = \{S_{x,y} \mid x \preceq_1 y\}$.

**Step 3:** Generate the conjunction $\hat{G}_\alpha$ of the following constraints:

1. $\sigma(\mathsf{nil}) = \{\mathsf{nil}\}$
2. $\sigma(y) = \{y\} \cup \sigma(x)$, for each $x, y$ such that $\alpha$ satisfies $\mathsf{tail}(y) = x$
3. $\sigma(y) = \{y\} \cup S_{x,y} \cup \sigma(x)$, for each $x, y$ such that $\alpha$ satisfies $x \preceq_1 y$
4. $\mathsf{disjoint}((S)_{S \in \mathsf{Segs}}, (\{x\})_{x \in \alpha(\mathsf{list})})$

**Step 4:** Existentially quantify over all Segs variables in $\hat{G}_\alpha$. If the goal is to obtain a formula without Segs variables, replace each variable $S_{x,y}$ with $(\sigma(y) \setminus \sigma(x)) \setminus \{y\}$.

**Step 5:** Return the resulting formula $G_\alpha$.

Fig. 13: Generation of set constraints from a PRP structure

3. *if* $x \neq y$, *if* $\mathsf{tail}(x)$ *and* $\mathsf{tail}(y)$ *are defined and equal, then both* $\mathsf{head}(x)$ *and* $\mathsf{head}(y)$ *are defined.*

**Definition 11.** *With each* PRP *structure* $\alpha$ *we associate a conjunction of literals that are (strongly) satisfied in* $\alpha$*. We call this formula a* PRP *conjunction.*

**Theorem 12.** *Each* FLS *formula is equivalent to an existentially quantified finite disjunction of* PRP *conjunctions.*

The proof of the theorem is in Appendix B.2. We can compute the PRP structures for an FLS formula by using a simple modification of the decision procedure for FLS in Figure 12.

### 7.2 Constraints on Sets of Sublists

Define $\sigma(y) = \{x.\ x \preceq y\}$. Our goal is to show that extending FLS with the $\sigma(\_)$ operator and the set algebra of such sets yields in a decidable logic. To this extent, we consider an FLS formula $F$ with free variables $x_1, \ldots, x_n$ and show that the defined relation on sets $\rho = \{(\sigma(x_1), \ldots, \sigma(x_n)).\ F(x_1, \ldots, x_n)\}$ is definable as $\rho = \{(s_1, \ldots, s_n).\ G(s_1, \ldots, s_n)\}$ for some quantifier-free BAPA [10] formula $G$. By Theorem 12, it suffices to show this property when $F$ is a PRP conjunction, given by some PRP structure $\alpha$. Figure 13 shows the generation of set constraints from a PRP structure. By replacing each $\sigma(x)$ with a fresh set variable $s_x$ in the resulting constraint we obtain a formula in set algebra. We can check the satisfiability of such formulas following the algorithms in [9, 10] and generate explicit models or perform synthesis as in [8]. Soundness and completeness of the reduction in Figure 13 are proved in Appendicies B.3 and B.4. In Appendix A we demonstrate the reduction for the example shown in Figure 3.

Among the consequences of this reduction is NP-completeness of a logic containing atomic formulas of FLS, along with formulas $s = \sigma(x)$, set algebra expressions

**Input:** a PRP structure $\alpha$ and an image constraint $C$.
**Output:** a set constraint $C_\alpha$ without $\mathsf{head}[s]$ expressions
**Step 1:** Replace each $\tau(x)$ in $C$ with $\mathsf{head}[\sigma(x) \setminus \{\mathsf{nil}\}]$.
**Step 2:** Let $P_i$ be all sets of the form $\{x_i\}$ or $S_{x_i,x_j}$ from Figure 13. If $s$ is a boolean combination of expressions of the form $\sigma(x)$, $\{x\}$, let $J(s)$ be such that $s = \bigcup_{i \in J(s)} P_i$ is the decomposition of $s$ into disjoint sets, derived from set equalities in Figure 13. Then replace each expression $\mathsf{head}[s]$ with $\bigcup_{i \in J(s)} \mathsf{head}[P_i]$.
**Step 3:** Replace each $\mathsf{head}[P_i]$ with a fresh set variable $Q_i$ and conjoin the result with the following constraints on the image sets $Q_i$:
1. $\mathsf{card}(Q_i) \leq \mathsf{card}(P_i)$
2. $Q_i = \emptyset \rightarrow P_i = \emptyset$
3. $Q_i \cap Q_j = \emptyset$, for each $x, y \in \alpha(\mathsf{list})$ such that $P_i = \{x\}$, $P_j = \{y\}$, $x \neq y$ and $\mathsf{tail}(x) = \mathsf{tail}(y)$.

**Step 4:** Existentially quantify over all $Q_i$ and return the resulting formula $C_\alpha$.

Fig. 14: Eliminating $\mathsf{head}[s]$ from image constraints by introducing additional constraints on top of Figure 13.

containing $\subseteq, \cap, \cup, \setminus, =$ on sets, and the cardinality operator $\mathsf{card}(s)$ that computes the size of the set $s$ along with integer linear arithmetic constraints on such sizes. Because the length of the list $x$ is equal to $\mathsf{card}(\sigma(x)) - 1$, this logic also naturally supports reasoning about list lengths. We note that such a logic can also support a large class of set comprehensions of the form $S = \{x.\, F(x, y_1, \ldots, y_n)\}$ when the atomic formulas within $F$ are of the form $u \preceq v$ and at least one atomic formula of the form $x \preceq y_i$ occurs positively in disjunctive normal form of $F$. Because $\forall x.F$ is equivalent to $\mathsf{card}(\{x. \neg F\}) = 0$, sets give us a form of universal quantification on top of FLS.

### 7.3 Additional Constraints on List Content

We next extend the previous constraints to impose set constraints not only on the set of sublists $\sigma(x)$ but also on the images of such sets under the head function. We define the list content function by $\tau(x) = \mathsf{head}[\sigma(x) \setminus \{\mathsf{nil}\}]$ where we define $\mathsf{head}[s] = \{\mathsf{head}(x) \mid x \in s\}$. We then obtain our full logic $\mathsf{FLS}^2$ shown in Figure 8 that introduces constraints of the form $\mathsf{head}[s] = v$ on top of FLS and constraints on sets of sublists. To show decidability of this logic, we use techniques inspired by [24] to eliminate the image constraints. The elimination procedure is shown in Figure 14. We use the properties of PRP structures that the elements for which $\mathsf{tail}(x_L) = \mathsf{tail}(x_R)$ holds have defined values $\mathsf{head}(x_L)$ and $\mathsf{head}(x_R)$. This allows us to enforce sufficient conditions on sets of sublists and sets of their heads to ensure that the axiom Pure can be enforced. The elimination procedure assumes that we have $\mathsf{head}(s)$ expressions only in the cases where $s$ is a combination of sets of the form $\sigma(x)$ and $\{x\}$, which ensures that $s$ is a disjoint combination of polynomially many partitions. This restriction is not necessary [24], but is natural in applications and ensures the membership in NP.

We claim that we have thus obtained an interesting point in the design space of decidable verification logics: the logic can express many interesting properties and has a reasonably efficient decision procedure.

## References


1. C. Barrett, I. Shikanian, and C. Tinelli. An abstract decision procedure for satisfiability in the theory of recursive data types. *Electronic Notes in Theoretical Computer Science*, 174(8):23–37, 2007.
2. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.
3. A. Bouajjani, C. Dragoi, C. Enea, and M. Sighireanu. A logic-based framework for reasoning about composite data structures. In *CONCUR*, 2009.
4. S. Jacobs. Incremental instance generation in local reasoning. In *CAV*, pages 368–382, 2009.
5. S. Jacobs. *Hierarchic Decision Procedures for Verification*. PhD thesis, Saarland University, 2010.
6. J. Jaffar. Minimal and complete word unification. *J. ACM*, 37(1):47–85, 1990.
7. V. Kuncak and D. Jackson. Relational analysis of algebraic datatypes. In *Joint 10th ESEC and 13th ACM SIGSOFT FSE*, 2005.
8. V. Kuncak, M. Mayer, R. Piskac, and P. Suter. Complete functional synthesis. In *PLDI*, 2010.
9. V. Kuncak, H. H. Nguyen, and M. Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *J. of Automated Reasoning*, 2006.
10. V. Kuncak and M. Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In *CADE-21*, 2007.
11. S. Lahiri and S. Qadeer. Back to the future: revisiting precise program verification using SMT solvers. In *POPL*, 2008.
12. G. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, pages 129–198, 1977. (In AMS, (1979)).
13. H. H. Nguyen, C. David, S. Qin, and W.-N. Chin. Automated verification of shape, size and bag properties via separation logic. In *VMCAI*, 2007.
14. M. Odersky, L. Spoon, and B. Venners. *Programming in Scala: a comprehensive step-by-step guide*. Artima Press, 2008.
15. D. C. Oppen. Reasoning about recursively defined data structures. In *POPL*, pages 151–157, 1978.
16. R. Piskac, L. de Moura, and N. Bjørner. Deciding Effectively Propositional Logic using DPLL and substitution sets. *J. of Automated Reasoning*, 44(4):401–424, 2010.
17. R. Piskac, P. Suter, and V. Kuncak. On decision procedures for ordered collections. Technical Report LARA-REPORT-2010-001, EPFL, 2010.
18. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3), 2004.
19. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In *CADE*, pages 219–234, 2005.
20. V. Sofronie-Stokkermans. Locality results for certain extensions of theories with bridging functions. In *CADE*, 2009.
21. P. Suter, M. Dotta, and V. Kuncak. Decision procedures for algebraic data types with abstractions. In *POPL*, 2010.
22. K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM (JACM)*, 34(2):492–510, 1987.
23. T. Wies, R. Piskac, and V. Kuncak. Combining theories with shared set operations. In *FroCoS*, 2009.
24. K. Yessenov, V. Kuncak, and R. Piskac. Collections, cardinalities, and relations. In *VMCAI*, 2010.
25. K. Zee, V. Kuncak, and M. Rinard. Full functional verification of linked data structures. In *PLDI*, 2008.

# A BAPA Reduction for Drop Example

We use the procedure given in Section 7 to reduce the FLS conjunct shown in Figure 3 to a BAPA formula over the set variables $Xs$, $Ys$, $Zs$. Figure 15 depicts the PRP structures that are equivalent to the FLS conjunct in Figure 3. Every node denotes an element of sort list. The square nodes denote the nil element.

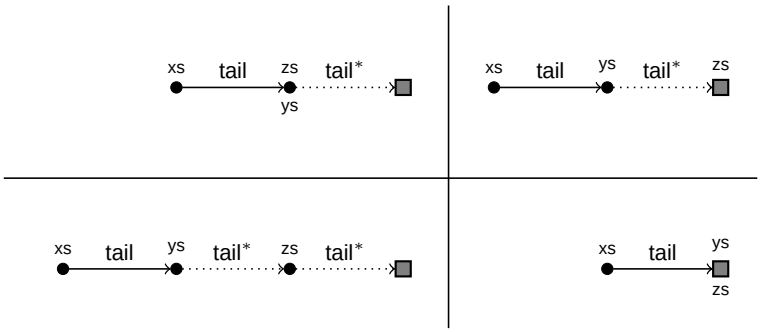


Fig. 15: PRP structures for the FLS conjunct in Figure 3

Figure 16 shows the set constraints generated from the four PRP structures. There is one disjunct for each PRP in Figure 15. In each disjunct we can now replace the sets $\sigma(\mathsf{xs})$, $\sigma(\mathsf{ys})$, and $\sigma(\mathsf{zs})$ by the shared set variables $Xs$, $Ys$, and $Zs$. Existentially quantifying over all remaining variables and applying quantifier elimination to the resulting disjunction then yields the BAPA formula

$$Zs \subseteq Ys \ \wedge \ Ys \subseteq Xs \ \wedge \ \mathsf{card}(Xs) > 1 \ \wedge \ \mathsf{card}(Xs) = \mathsf{card}(Ys) + 1 \ .$$

This formula is the projection onto the shared set variables shown in Figure 3.

# B Additional Proofs

## B.1 Proof of Proposition 5

**Proposition 5.** *Every finite model of $\mathcal{K}_{\mathsf{FLSf}}$ is isomorphic to some structure in $\mathcal{M}_{\mathsf{FLSf}}$.*

*Proof.* Let $\alpha$ be a finite model of $\mathcal{K}_{\mathsf{FLSf}}$. In the following, for a function symbol $f$, we use $f_\alpha$ as a short-hand for $\alpha(f)$. Similarly, we denote by $\mathsf{tail}^*_\alpha$ the reflexive transitive closure of the function $\alpha(\mathsf{tail})$.

First, note that the axioms Refl, Trans, AntiSym, and GCS1, GCS2, GCS3 ensure that $(\alpha(\mathsf{list}), \preceq_\alpha, \sqcap_\alpha)$ is a meet-semilattice. We next prove that $\preceq_\alpha$ is the inverse of $\mathsf{tail}^*_\alpha$.

$$
\begin{aligned}
&\exists S_{\mathsf{nil,ys}}.\ \sigma(\mathsf{nil}) = \{\mathsf{nil}\} \wedge \sigma(\mathsf{ys}) = \{\mathsf{ys}\} \cup S_{\mathsf{nil,ys}} \cup \sigma(\mathsf{nil}) \wedge \sigma(\mathsf{zs}) = \sigma(\mathsf{ys}) \wedge \\
&\qquad \sigma(\mathsf{xs}) = \{\mathsf{xs}\} \cup \sigma(\mathsf{ys}) \wedge S_{\mathsf{nil,ys}} = (\sigma(\mathsf{ys}) \setminus \sigma(\mathsf{nil})) \setminus \{\mathsf{ys}\} \ \wedge \\
&\qquad \mathsf{disjoint}(S_{\mathsf{nil,ys}}, \{\mathsf{xs}\}, \{\mathsf{ys}\}, \{\mathsf{nil}\}) \\
\vee\ &\exists S_{\mathsf{nil,ys}}.\ \sigma(\mathsf{nil}) = \{\mathsf{nil}\} \wedge \sigma(\mathsf{zs}) = \sigma(\mathsf{nil}) \wedge \sigma(\mathsf{ys}) = \{\mathsf{ys}\} \cup S_{\mathsf{nil,ys}} \cup \sigma(\mathsf{nil}) \wedge \\
&\qquad \sigma(\mathsf{xs}) = \{\mathsf{xs}\} \cup \sigma(\mathsf{ys}) \wedge S_{\mathsf{nil,ys}} = (\sigma(\mathsf{ys}) \setminus \sigma(\mathsf{nil})) \setminus \{\mathsf{ys}\} \ \wedge \\
&\qquad \mathsf{disjoint}(S_{\mathsf{nil,ys}}, \{\mathsf{xs}\}, \{\mathsf{ys}\}, \{\mathsf{nil}\}) \\
\vee\ &\exists S_{\mathsf{zs,ys}}, S_{\mathsf{nil,zs}}.\ \sigma(\mathsf{nil}) = \{\mathsf{nil}\} \wedge \sigma(\mathsf{zs}) = \{\mathsf{zs}\} \cup S_{\mathsf{nil,zs}} \wedge \sigma(\mathsf{ys}) = \{\mathsf{ys}\} \cup S_{\mathsf{zs,ys}} \cup \sigma(\mathsf{zs}) \wedge \\
&\qquad \sigma(\mathsf{xs}) = \{\mathsf{xs}\} \cup \sigma(\mathsf{ys}) \wedge S_{\mathsf{nil,zs}} = (\sigma(\mathsf{zs}) \setminus \sigma(\mathsf{nil})) \setminus \{\mathsf{zs}\} \ \wedge \\
&\qquad S_{\mathsf{zs,ys}} = (\sigma(\mathsf{ys}) \setminus \sigma(\mathsf{zs})) \setminus \{\mathsf{ys}\} \wedge \mathsf{disjoint}(S_{\mathsf{nil,zs}}, S_{\mathsf{zs,ys}}, \{\mathsf{xs}\}, \{\mathsf{ys}\}, \{\mathsf{zs}\}, \{\mathsf{nil}\}) \\
\vee\ &\sigma(\mathsf{nil}) = \{\mathsf{nil}\} \wedge \sigma(\mathsf{ys}) = \sigma(\mathsf{nil}) \wedge \sigma(\mathsf{zs}) = \sigma(\mathsf{nil}) \wedge \sigma(\mathsf{xs}) = \{\mathsf{xs}\} \cup \sigma(\mathsf{nil}) \wedge \\
&\mathsf{disjoint}(\{\mathsf{xs}\}, \{\mathsf{nil}\})
\end{aligned}
$$

Fig. 16: Set constraints generated from PRP structures in Figure 15.

For proving $\mathsf{tail}^*_\alpha \subseteq \preceq_\alpha^{-1}$, let $u, v \in \alpha(\mathsf{list})$ such that $(u, v) \in \mathsf{tail}^*_\alpha$. Then there exist $u_1, \ldots, u_n$ such that $u = u_1$, $v = u_n$, and for all $1 \leq i < n$, $\mathsf{tail}_\alpha(u_i) = u_{i+1}$. If $n = 1$ then $u = v$ and by axiom Refl we immediately have $v \preceq_\alpha u$. If on the other hand $u \neq v$ then by UnfoldL we have for all $1 \leq i < n$, $\mathsf{tail}_\alpha(u_i) \preceq_\alpha u_i$ and thus $u_{i+1} \preceq_\alpha u_i$. Using axiom Trans we then conclude by induction on $i$ that for all $1 < i \leq n$, $u_i \preceq_\alpha u_1$. Hence, $v \preceq_\alpha u$.

For proving the other direction, let $u \in \alpha(\mathsf{list})$ and let $S_u = \{\, v \mid v \preceq_\alpha u \,\}$. We show that for all $v \in S_u$, $(u, v) \in \mathsf{tail}^*_\alpha$. Since $\alpha$ is finite, $S_u$ is finite, as well. Thus, using axioms Total and AntiSym we can construct an enumeration $u_1, \ldots, u_n$ of the elements of $S_u$ such that for all $1 \leq i < j \leq n$, $u_j \preceq_\alpha u_i$ but not $u_i \preceq_\alpha u_j$. In particular $u_1 = u$. We prove by induction on $i$ that for all $1 \leq i \leq n$, $(u_1, u_i) \in \mathsf{tail}^*_\alpha$. By reflexivity of $\mathsf{tail}^*_\alpha$ we immediately have $(u_1, u_1) \in \mathsf{tail}^*_\alpha$. Now assume that $(u_1, u_i) \in \mathsf{tail}^*_\alpha$. Since $u_i \in S_u$, we know by UnfoldL and Trans that $\mathsf{tail}_\alpha(u_i) \in S_u$. Hence, $\mathsf{tail}_\alpha(u_i) = u_j$ for some $j \geq i$. By UnfoldR we know that for all $j \geq i$, $u_j = u_i$ or $u_j \preceq_\alpha \mathsf{tail}_\alpha(u_i)$. Hence, by construction of the enumeration it follows that $u_{i+1} = \mathsf{tail}_\alpha(u_i)$. Together with the induction hypothesis we then conclude $(u_1, u_{i+1}) \in \mathsf{tail}^*_\alpha$.

We can now define a structure isomorphism $\phi$ between $\alpha$ and some structure $\alpha' \in \mathcal{M}_{\mathsf{FLSf}}$, i.e., for each sort $s$, $\phi$ is a structure-preserving bijection from $\alpha(s)$ to $\alpha'(s)$. Since all structures agree on the interpretation of sort bool, we first define $\phi$ as the identity mapping on $\alpha(\mathsf{bool})$. For the sort data, we let $\phi$ be some injective mapping from $\alpha(\mathsf{data})$ to the set of data terms D, such that $\phi(\alpha(\mathsf{nil})) = d_1$. In order to define $\phi$ on $\alpha(\mathsf{list})$, note that $\alpha$ satisfies axioms Nil and AntiSym. Thus, we know that the inverse of the relation $\mathsf{tail}^*_\alpha$ is well-founded and $\alpha(\mathsf{nil})$ is the smallest element of $\alpha(\mathsf{list})$ with respect to this relation. We can hence recursively define $\phi$ as a mapping from $\alpha(\mathsf{list})$ to L as follows:

$$
\phi(u) = \begin{cases} \mathsf{nil} & \text{if } u = \alpha(\mathsf{nil}) \\ \mathsf{cons}(\phi(v), \phi(w)) & \text{if } \alpha(\mathsf{tail})(u) = v \text{ and } \alpha(\mathsf{head})(u) = w \end{cases}
$$

From axiom Pure follows that $\phi$ is an injective mapping from $\alpha(\mathsf{list})$ to L. Now define the structure $\alpha'$ as follows: for all sorts $s \in \Sigma_{\mathsf{FLSf}}$, let $\alpha'(s) = \phi(\alpha(s))$, and for all function symbols $f \in \Sigma_{\mathsf{FLSf}}$ of sort $s_1 \times \cdots \times s_n \to s$, let $\alpha'(f)(u_1, \ldots, u_n) =$

$\phi(\alpha(f)(\phi^{-1}(u_1), \ldots, \phi^{-1}(u_n)))$. By construction $\alpha' \in \mathcal{M}_{\mathsf{FLSf}}$ and $\alpha'$ is isomorphic to $\alpha$. □

### B.2 Proof of Theorem 12

Consider a FLS formula $F$. A minor modification of the algorithm in Figure 12 can enumerate a finite set of partial substructures of structures in $\mathcal{M}_{\mathsf{FLSf}}$ for which $F$ is true. It suffices to show that each partial substructure can be represented by finitely many PRP structures.

To ensure that a relation (such as $x \preceq y$) is defined, we perform case analysis on whether $x \preceq y$ holds or not. To ensure the remaining properties, proceed as follows.

Define an *ultimately converging triple* to be a triple of distinct elements $(x_L, y, x_R)$ such that $y \preceq x_L$ and $y \preceq x_R$. Then define a *converging triple* to be an ultimately converging triple such that there is no distinct ultimately converging triple $(x'_L, y', x'_R)$ with the property $x'_L \preceq x_L$, $z' \preceq z$, $x'_R \preceq x_R$.

An *unresolved converging triple* is a converging triple $(x_L, y, x_R)$ such that $x_L \sqcap x_R$ is not defined in the structure. Given an unresolved converging triple $(x_L, y, x_R)$, we consider the case 1) $x_L \sqcap x_R = y$ and the case 2) $x_L \sqcap x_R = z$ for a fresh elements of the structure such that $x_L \sqcap x_R = z$ and $y \preceq z$. In each step of this process the number of unresolved converging triples reduces by one, so we can construct finitely many structures where all converging triples are resolved.

In the next step, we ensure that for all converging triples $(x_L, y, x_R)$ we have that $\mathsf{tail}(x_L)$ and $\mathsf{tail}(x_R)$ are defined and equal to $y$. We call a triple for which this does not hold *non-refined*. Given a non-refined triple $(x_L, y, x_R)$ then it cannot be the case that both $\mathsf{tail}(x_L)$ or $\mathsf{tail}(x_R)$ are undefined, otherwise there would be a converging triple $(\mathsf{tail}(x_L), y, x_R)$ or $(x_L, y, \mathsf{tail}(x_R))$. Suppose without loss of generality that $\mathsf{tail}(x_L)$ is undefined. We then consider either $\mathsf{tail}(x_L) = y$, or we introduce $z$ such that $\mathsf{tail}(z) = y$ and $z \preceq x_L$. By repeating this process at most twice for each non-refined triple, we ensure that all triples are both resolved and refined.

Suppose every converging triple $(x_L, y, x_R)$ is refined. Then define both $\mathsf{head}(x_L)$ and $\mathsf{head}(y_L)$ to be either existing or fresh elements. The Pure axiom ensures $\mathsf{head}(x_L) \neq \mathsf{head}(y_L)$. As a result we obtain a finite set of PRP structures. By existentially quantifying over the freshly introduced elements we obtain a disjunction of PRP conjunctions equivalent to $F$. □

### B.3 Soundness of the Reduction in Figure 13

Consider a $\mathcal{M}_{\mathsf{FLSf}}$ element whose substructure if $\alpha$. We show that the generated formula $G_\alpha$ is a consequence of the PRP conjunction corresponding to $\alpha$. The condition $\sigma(\mathsf{nil}) = \{\mathsf{nil}\}$ follows from the definition, as well as $\sigma(y) = \{y\} \cup \sigma(x)$ for $\mathsf{tail}(y) = x$. To ensure $\sigma(y) = \{y\} \cup S_{x,y} \cup \sigma(x)$ we define $S_{x,y} = (\sigma(y) \setminus \sigma(x)) \setminus \{y\}$. It remains to show that for this definition the sets $S_{x,y}$ and the singleton sets $\{x\}$ are all disjoint. By construction $\{x\} \cap \{y\} = \emptyset$ for distinct $x, y$. Consider two distinct set variables $S_{x,y}$ and $S_{u,v}$. Let $z = x \sqcap u$. Then also $z = y \sqcap v$. By definition of $\sigma(y)$ and $\sigma(v)$, we have $\sigma(y) \cap \sigma(v) = \sigma(z)$. We also have $\sigma(z) \subseteq \sigma(x)$ and $\sigma(z) \subseteq \sigma(u)$. This implies that

the sets $\sigma(y) \setminus \sigma(x)$ and $\sigma(v) \setminus \sigma(u)$ are disjoint, so $S_{x,y}$ and $S_{u,v}$ are also disjoint. Showing that $\{x\}$ and $S_{u,v}$ are disjoint is similarly straightforward.

### B.4 Completeness of the Reduction in Figure 13

Given a partial structure $\alpha$ and the values of sets that satisfy the generated formula $G_\alpha$ in Figure 13, we extend $\alpha$ to a finite total structure in $\mathcal{M}_{\mathsf{FLSf}}$. To do this, consider each pair $x, y$ in the domain of $\alpha$ for which $x \preceq_1 y$. Let $k$ be the size of the set $S_{x,y}$. If $k = 0$ then let $\mathsf{tail}(y) = x$. Otherwise, introduce $k$ fresh and distinct list elements $z_1, \ldots, z_k$ and extend $\mathsf{tail}$ such that

$$\mathsf{tail}(y) = z_k \wedge \left( \bigwedge_{i=1}^{k-1} \mathsf{tail}(z_{i+1}) = z_i \right) \wedge \mathsf{tail}(z_1) = x$$

To ensure that the elements are distinct, define $\mathsf{head}(z_i) = h_i$ where $h_i$ are fresh head elements. Extend $\preceq$ and $\_ \sqcap \_$ according to $\mathsf{tail}$ to ensure that the structure belongs to $\mathcal{M}_{\mathsf{FLSf}}$. This completes the construction showing the completeness of a reduction from FLS to BAPA.