Interpolation Procedures

Deliverable D1-3

ANR project VECOLIB

September 2016

Abstract

This deliverable reports on the state of the art concerning interpolation procedures for theories other than integer/real arithmetic with uninterpreted functions. We investigate high-level theories of strings and algebraic data types (ADT) as well as low-level theories of arrays, lists and separation logic.

A *signature* $\Sigma = (\Sigma^{\mathrm{s}}, \Sigma^{\mathrm{f}})$ consists of a set Σ^{s} of *sort symbols* and a set Σ^{f} of *function symbols* $f^{\sigma_1 \cdots \sigma_n \sigma}$, where $n \geq 0$ is its arity, $\sigma_1, \ldots, \sigma_n \in \Sigma^{\mathrm{s}}$ are the sorts of its arguments and $\sigma \in \Sigma^{\mathrm{s}}$ is the sort of its result. If $n = 0$, we call f^{σ} a *constant symbol*. We assume that every signature contains the boolean sort, and write $\top$ and $\bot$ for the boolean constants *true* and *false*. Let Var be a countable set of *first-order variables*, each variable $x^{\sigma} \in \mathsf{Var}$ having an associated sort σ. We omit specifying the sorts of the function symbols and variables, whenever they are not important. Terms and formulas are defined recursively as usual.

A *partial* Σ*-structure* M is a function that maps each sort $\sigma \in \Sigma^{\mathrm{s}}$ to a non-empty set $M(\sigma)$ and each function symbol $f^{\sigma_1 \cdots \sigma_n \sigma} \in \Sigma^{\mathrm{f}}$ into a partial function $M(f) : M(\sigma_1) \times \ldots \times M(\sigma_n) \rightharpoonup M(\sigma)$. We denote by $[M] = \bigcup_{\sigma \in \Sigma^{\mathrm{s}}} M(\sigma)$ the *support* of M.

Given a formula ϕ, a (partial) Σ-structure M and an valuation ν of the free variables in ϕ, we write $M, \nu \models \phi$ if M is a model of ϕ under ν, defined recursively, as usual. We write $\phi \models \psi$ if, for every Σ-structure M and every valuation ν, $M, \nu \models \phi$ implies $M, \nu \models \psi$. A Σ*-theory* $\mathcal{T}$ is a set of formulae with no free variables (sentences) that is closed under entailments, i.e. $\phi \in \mathcal{T}$ and $\phi \models \psi$ implies $\psi \in \mathcal{T}$.

Given an unsatisfiable conjunction of formulae $A \wedge B \models \bot$, a *Craig interpolant* (interpolant) for $A \wedge B$ is a formula I such that: (1) $A \models I$, (2) $I \wedge B \models \bot$, and (3) I contains only logical symbols that are common to both A and B. Using classical proof-theoretic arguments, it is possible to prove the following points [5, Theorem 8]:

1. every r.e.[1] theory is interpolating.
2. every r.e. theory that has quantifier elimination is quantifier-free interpolating.
3. every quantifier-free interpolating theory eliminates quantifiers.

Basically every interpolant inference (interpolation) procedure is based on a decision procedure for the theory of A and B and depends highly on the particular techniques used by the decision procedure. For this reason, interpolation is integrated within Satisfiability Modulo Theories (SMT) solvers. In general, interpolation procedures exist for linear integer/real arithmetic (LIA/LRA) possibly with uninterpreted functions (UFLIA/UFLRA). This deliverable aims to document the state of the art for the interpolation procedures other than (UF)LIA/LRA.

[1] A theory is r.e. if the set of sentences in the theory is r.e.

The VECOLIB proposal mentions the possibility of applying interpolation-based abstraction refinement model checking to the verification of low-level pointer handling programs. This entails the development of interpolation techniques for Separation Logics (SL), required to refine the abstract model of the program, by annotations used to exclude spurious counterexamples from the state search.

However, an interpolation-based verifier needs a fast entailment checker (SMT solver) for the underlying theory of pointers (in our case, SL) in order to detect abstract states that are covered (subsumed) by other abstract states, labeled with SL assertions. Our experience with the development of decision procedures for SL (reported in the deliverables D1.1 and D4.3) shows that, currently, the performance of SL solvers is unsuitable for the needs of lazy annotation abstraction refinement checkers, based on predicate abstraction or the IMPACT algorithm. This is partly due to the complexity of the satisfiability problem (PSPACE -complete for quantifier-free SL, instead of NP-complete for LIA or LRA) and partly due to the lack of maturity of the existing solvers.

Due to this reason, in the rest of the VECOLIB project, we envisage experimenting with interpolation in high-level logics, by adapting our alternating data automata (ADA) emptiness checker (reported in deliverable D4.5) to work with strings and algebraic data types, in addition to linear arithmetic.

1 Interpolation for High Level Logics

1.1 Interpolation for Strings

This section summarizes the work of Abdulla et al [1] reporting on a decision procedure used for interpolation-based model checking of string-manipulating programs. It is worth pointing out that both the decision and interpolation procedures are incomplete, due to the intrinsic hardness of word problems [6]. We consider the signature Σ_{str} with sorts $\Sigma^{s}_{str} = \{\mathsf{Bool}, \mathsf{Int}, \mathsf{String}, \mathsf{Reg}\}$ and the following constant symbols:

$$\begin{array}{rcl} \varepsilon & : & \mathsf{String} \\ \cdot & : & \mathsf{String} \times \mathsf{String} \rightarrow \mathsf{String} \\ |.| & : & \mathsf{String} \rightarrow \mathsf{Int} \\ \in & : & \mathsf{String} \times \mathsf{Reg} \rightarrow \mathsf{Bool} \end{array}$$

The atoms of the string logic are equalities $s \approx t$ and disequalities $s \not\approx t$ between string terms involving variables, interpreted over a set U of strings, arithmetic inequalities $e \leq e'$ between linear terms involving length terms $|s|$ and membership constraints $s \in R$, where R is a regular language described by a regular expression, or, equivalently, a finite automaton.

The satisfiability of quantifier-free first order formulae of this theory is an open problem. However the subfragment of the logic consisting of word equalities and disequalities in which no string variable occurs more than once in an equality or disequality, and furthermore there are no cyclic dependencies between string variables, is shown to be decidable. The algorithm is based on a set of sound and locally complete inference rules whose application strictly decreases the size of the consequent formulae.

The interpolation procedure is based on the decision procedure for the fragment of the logic. The interpolation is incomplete, because it returns only interpolants of the form $x_1\$x_2\$\ldots\$x_n \in R$, where \$ is a delimiter symbol and R is a regular language of a bounded size (the bound is either on the size of the regular expression or the minimal automaton for R) over the alphabet of strings including \$.

Consider an interpolation problem $\phi(x_1, \ldots, x_n) \wedge \psi(x_1, \ldots, x_n)$. Starting with two sets of strings A and B, initially empty, the algorithm repeats the following steps while there is a regular language R of size L, such that $A \subseteq R$ and $B \cap R = \emptyset$:

1. if $\phi(x_1,\ldots,x_n) \wedge \neg(x_1\$\ldots\$x_n \in R)$ is satisfiable, let ν be a satisfying assignment and update A to $A \cup \{\nu(x_1)\$\ldots\$\nu(x_n)\}$;
2. else if $\psi(x_1,\ldots,x_n) \wedge x_1\$\ldots\$x_n \in R$ is satisfiable, let ν be a satisfying assignment and update B to $B \cup \{\nu(x_1)\$\ldots\$\nu(x_n)\}$;
3. otherwise return the interpolant $x_1\$\ldots\$x_n \in R$.

1.2 Interpolation by Reduction

This section summarizes the work of Kapur, Majumdar and Zarba [5] that computes (not necessarily quantifier-free) interpolants in the theories of arrays, sets and multisets by reduction to the theory of equality and uninterpreted functions. Let $\mathcal{T}$ be a Σ-theory and let $\mathcal{R}$ be a Ω-theory such that $\Omega^{\mathrm{s}} \subseteq \Sigma^{\mathrm{s}}$ and $\Omega^{\mathrm{f}} \subseteq \Sigma^{\mathrm{f}}$. We say that $\mathcal{T}$ *reduces to* $\mathcal{R}$ if there is a computable map from flat Σ-atoms to Ω-formulae such that φ^* denotes the homomorphic application of the map to a Σ-formula φ and:

- φ and φ^* are $\mathcal{T}$-equivalent,
- if φ^* is $\mathcal{R}$-satisfiable then φ is $\mathcal{T}$-satisfiable.

The reduction approach to interpolation assumes that $\mathcal{T}$ reduces to $\mathcal{R}$ and it is possible to compute an $\mathcal{R}$-interpolant of any unsatisfiable conjunction $\varphi \wedge \psi \models \bot$. Then it is possible to compute a $\mathcal{T}$-interpolant for any unsatisfiable quantifier-free conjunction $\varphi \wedge \psi \models \bot$. In this work, it is assumed that interpolants are computed from Gentzen-style or tableaux proofs in the theory of equality with uninterpreted functions [3].

Arrays The extensional theory of arrays, with signature Σ_{arr}, with sorts $\Sigma^{\mathrm{s}}_{arr} = \{\mathsf{array}, \mathsf{index}, \mathsf{elem}\}$ and function symbols $\Sigma^{\mathrm{f}}_{arr} = \{rd, wr\}$, where:

$$\begin{array}{rcl} rd & : & \mathsf{array} \times \mathsf{index} \rightarrow \mathsf{elem} \\ wr & : & \mathsf{array} \times \mathsf{index} \times \mathsf{elem} \rightarrow \mathsf{array} \end{array}$$

is reduced to Ω_{arr}, where $\Omega^{\mathrm{s}}_{arr} = \Sigma^{\mathrm{s}}_{arr}$ and $\Omega^{\mathrm{f}}_{arr} = \{rd\}$ as follows:

$$\begin{array}{rcl} a \approx b & \mapsto & \forall i \,.\, rd(a,i) \approx rd(b,i) \\ a \approx wr(b,i,e) & \mapsto & rd(a,i) \approx e \wedge \forall j \,.\, j \not\approx i \rightarrow rd(a,j) \approx rd(b,j) \end{array}$$

Sets The theory of sets with finite cardinality constraints has the signature Σ_{set}, with sorts $\Sigma^{\mathrm{s}}_{set} = \{\mathsf{set}, \mathsf{elem}\}$ and function symbols $\Sigma^{\mathrm{f}}_{set} = \{\emptyset, 1, \cup, \cap, \setminus, \{.\}, \in, \{|.| \geq k\}_{k\in\mathbb{N}}, \{|.| = k\}_{k\in\mathbb{N}}\}$, where:

$$\begin{array}{rcl@{\qquad}rcl} \emptyset, 1 & : & \mathsf{set} & \in & : & \mathsf{elem} \times \mathsf{set} \rightarrow \mathsf{set} \\ \cup, \cap, \setminus & : & \mathsf{set} \times \mathsf{set} \rightarrow \mathsf{set} & |.| \geq k, |.| = k & : & \mathsf{set} \rightarrow \mathsf{bool} \end{array}$$

This theory is reduced to the theory with signature Ω_{set}, with sorts $\Omega_{set} = \Sigma_{set}$ and function symbols $\Omega^{\mathrm{f}}_{set} = \{\in\}$, as follows:

$$\begin{array}{rcl} x \approx y & \mapsto & \forall e \,.\, e \in x \leftrightarrow e \in y \\ x \approx \emptyset & \mapsto & \forall e \,.\, e \notin x \\ x \approx 1 & \mapsto & \forall e \,.\, e \in x \\ x \approx y \cup z & \mapsto & \forall e \,.\, e \in x \leftrightarrow (e \in y \vee e \in z) \\ x \approx y \cap z & \mapsto & \forall e \,.\, e \in x \leftrightarrow (e \in y \wedge e \in z) \\ x \approx y \setminus z & \mapsto & \forall e \,.\, e \in x \leftrightarrow (e \in y \wedge e \notin z) \\ x \approx \{e_0\} & \mapsto & \forall e \,.\, e \in x \leftrightarrow e \approx e_0 \\ |x| \geq k & \mapsto & \exists e_1 \ldots e_k \,.\, \bigwedge_{i=1}^{k} e_i \in x \wedge \bigwedge_{1\leq i<j\leq k} e_i \not\approx e_j \\ |x| = k & \mapsto & \exists e_1 \ldots e_k \,.\, \bigwedge_{i=1}^{k} e_i \in x \wedge \bigwedge_{1\leq i<j\leq k} e_i \not\approx e_j \wedge \forall e \,.\, e \in x \rightarrow \bigvee_{i=1}^{k} e \approx e_i \end{array}$$

Multisets The theory of multisets has signature Σ_{bag}, with sorts $\Sigma^{\mathsf{s}}_{bag} = \mathsf{Int}^{\mathsf{s}} \cup \{\mathsf{bag}, \mathsf{elem}\}$ and function symbols $\Sigma^{\mathsf{f}}_{bag} = \mathsf{Int}^{\mathsf{f}} \cup \{[\![.]\!], \sqcup, \sqcap, \uplus, \mathit{cnt}\}$, where:

$$\begin{array}{rcl} [\![.]\!] & : & \mathsf{elem} \times \mathsf{int} \rightarrow \mathsf{bag} \\ \sqcup, \sqcap, \uplus & : & \mathsf{bag} \times \mathsf{bag} \rightarrow \mathsf{bag} \\ \mathit{cnt} & : & \mathsf{elem} \times \mathsf{bag} \rightarrow \mathsf{int} \end{array}$$

This theory is reduced to the theory of integers with uninterpreted functions and equality as follows:

$$\begin{array}{rcl} x \approx y & \mapsto & \forall e \,.\, \mathit{cnt}(x,e) \approx \mathit{cnt}(y,e) \\ x \approx y \sqcup z & \mapsto & \forall e \,.\, \mathit{cnt}(e,x) \approx \max(\mathit{cnt}(e,y), \mathit{cnt}(e,z)) \\ x \approx y \sqcap z & \mapsto & \forall e \,.\, \mathit{cnt}(e,x) \approx \min(\mathit{cnt}(e,y), \mathit{cnt}(e,z)) \\ x \approx y \uplus z & \mapsto & \forall e \,.\, \mathit{cnt}(e,x) \approx \mathit{cnt}(e,y) + \mathit{cnt}(e,z) \\ x \approx [\![e_0]\!]^{(u)} & \mapsto & \mathit{cnt}(e_0,x) = \max(0,u) \wedge \forall e \,.\, e \neq e_0 \rightarrow \mathit{cnt}(e,x) \approx 0 \end{array}$$

A recent extension of this work reduces the theory of algebraic data types to the theory of integers with uninterpreted functions and equality [4].

2 Interpolation for Low Level Logics

2.1 Instantiation-based Interpolation for Arrays and Lists

This section summarizes the work of Totla and Wies [8]. This work generalizes the notion of local theory extension to cope with ground interpolation. For a signature $\Sigma_0 = (\Sigma^{\mathsf{s}}_0, \Sigma^{\mathsf{f}}_0)$ a *theory* $\mathcal{T}_0$ is a set of formulae over Σ_0 that are consequences of a set of axioms $\mathcal{K}_0$. A *theory extension* is a theory $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ where $\mathcal{K}$ is a set of axioms over the extended signature $\Sigma_1 = (\Sigma^{\mathsf{s}}_0 \cup \Sigma^{\mathsf{s}}_e, \Sigma^{\mathsf{f}}_0 \cup \Sigma^{\mathsf{f}}_e)$. An *embedding closure* for $\mathcal{T}_1$ is a function Ψ associating a finite set of ground terms T a finite set of ground terms $\Psi(T)$ such that:

- $\mathit{st}(\mathcal{K}) \subseteq \Psi(T)$, where $\mathit{st}(\mathcal{K})$ is the set of ground subterms in $\mathcal{K}$,
- $T \subseteq T'$ implies $\Psi(T) \subseteq \Psi(T')$,
- $\Psi(\Psi(T)) \subseteq \Psi(T)$ for all sets of ground terms T,
- $\Psi(h(T)) = h(\Psi(T))$, where h is a map between constants extended homomorphically to terms.

We say that $\mathcal{T}_1$ is a Ψ-*local theory extension* of $\mathcal{T}_0$ if, for every set of ground clauses G, $\mathcal{T}_1 \cup G \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}[\Psi(G)] \cup G \models \bot$, where $\mathcal{K}[\Psi(G)]$ is the set of instances of $\mathcal{K}$ in which all extension terms are in $\Psi(G)$. In other words, $\Psi(G)$ extracts those subterms from G that are necessary to prove unsatisfiability.

The notion of local theory extension is however not suitable for interpolation, because $\Psi(G)$ may contain terms over function symbols that are not shared between A and B. A possible solution is to impose strict syntactic restrictions on the form of the axioms in $\mathcal{K}$, as it is the case in the work of Sofronie-Stokkermans [7].

The solution adopted in [8] is model-theoretic. Given partial Σ-structures M and N, a *weak embedding* is an injective function $h : [M] \rightarrow [N]$ such that for all $f \in \Sigma^{\mathsf{f}}$ and all $a_1, \ldots, a_n \in [M]$, if $M(f)$ is defined on $a_1, \ldots, a_n$ then $N(f)(h(a_1), \ldots, h(a_n)) = h(M(f)(a_1, \ldots, a_n))$. If there exists a weak embedding $h : [M] \rightarrow [N]$ and $[M] \subseteq [N]$ then M is a *partial substructure* of N, denoted $M \subseteq N$.

An *amalgamation closure* for a theory extension $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is a function associating with finite set of ground terms T_A and T_B a finite set $W(T_A, T_B)$ of ground terms such that:

- $\mathit{st}(\mathcal{K}) \cup \mathit{st}(T_A) \subseteq W(T_A, T_B)$,

- $T_A \subseteq T'_A$ and $T_B \subseteq T'_B$ implies $W(T_A, T_B) \subseteq W(T'_A, T'_B)$,
- $W(W(T_A, T_B), W(T_B, T_A)) \subseteq W(T_A, T_B)$,
- $W(h(T_A), h(T_B)) = h(W(T_A, T_B))$ for any map on constant symbols such that $h(c_1) \neq h(c_2)$ for every $c_1 \in st(T_A)$ and $c_2 \in st(T_B)$ not shared between T_A and T_B,
- $W(T_A, T_B)$ only contains T_A-pure terms.

Given formulae A and B, we write $W(A, B)$ for $W(st(A), st(B))$. An extension theory $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is W-*separable* if for all sets of ground clauses A and B, $\mathcal{T}_1 \cup A \cup B \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}[W(A, B)] \cup \mathcal{K}[W(B, A)] \cup B \models \bot$. Clearly, W-separable theory extensions are local but not viceversa. Moreover, the interpolation problem for a W-separable theory extension $\mathcal{T}_1$ can be reduced to the ground interpolation problem for the base theory $\mathcal{T}_0$.

The question is which theories are W-separable and how to define the amalgamation closure W ? An *amalgam* for a theory $\mathcal{T}$ is a tuple (M_A, M_B, M_C), where M_A, M_B, M_C are models of $\mathcal{T}$ such that $M_A \supseteq M_C \subseteq M_B$ and $[M_C] = [M_A] \cap [M_B]$. The theory $\mathcal{T}$ has the *amalgamation property* if for each amalgam (M_A, M_B, M_C) there exists a model M_D such that $M_A \subseteq M_D \supseteq M_B$. This notion allows to prove the existence of ground interpolants for an theory extension but gives no means to build them using ground interpolation for the base theory. For this reason, the authors of [8] use *partial W-amalgams*, defined below.

For a partial model M of $\mathcal{T}_1$, let $T(M) = \left\{ f(a_1, \ldots, a_n) \mid a_i \in [M], f \in \Sigma_e^{\mathrm{f}}, M(f)(a_1, \ldots, a_n) \text{ defined} \right\}$. If W is an amalgamation closure for $\mathcal{T}_1$, a partial W-amalgam is a tuple (M_A, M_B, M_C) such that:

- M_A, M_B, M_C are partial models of $\mathcal{T}_1$,
- $M_A \supseteq M_C \subseteq M_B$,
- $[M_C] = [M_A] \cap [M_B]$,
- $W(T(M_A), T(M_B)) \subseteq T(M_A)$ and $W(T(M_B), T(M_A)) \subseteq T(M_B)$,
- $T(M_A) \cap T(M_B) \subseteq T(M_C)$.

A theory extension $\mathcal{T}_1$ has the *partial W-amalgamation property* if for all partial W-amalgams (M_A, M_B, M_C) there exists a model M_D of $\mathcal{T}_1$ such that $M_A \subseteq M_D \supseteq M_B$. Finally, a theory extension $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is W-separable if it has the partial W-amalgamation property. This semantic criterion is applied to give interpolation procedures for two theory extensions, namely a theory of arrays with difference function and a theory of linked lists.

2.1.1 Theory of Arrays

Given a base theory $\mathcal{T}_0$ with sorts index and elem, the theory of arrays $\mathcal{T}_{arr} = \mathcal{T}_0 \cup \mathcal{K}_{arr}$ is the extension of $\mathcal{T}_0$ with the extended signature and axioms below:

$$\begin{array}{rcll} rd & : & \mathsf{array} \times \mathsf{index} \to \mathsf{elem} & rd(wr(a,i,e),i) = e \\ wr & : & \mathsf{array} \times \mathsf{index} \times \mathsf{elem} \to \mathsf{array} & i \neq j \Rightarrow rd(wr(a,i,e),j) = rd(a,j) \\ diff & : & \mathsf{array} \times \mathsf{array} \to \mathsf{index} & a \neq b \Rightarrow rd(a, diff(a,b)) \neq rd(b, diff(a,b)) \end{array}$$

where rd and wr are the usual read and write functions and $diff(a,b)$ returns the first index where a and b differ, undefined if $a = b$. The following function is an amalgamation closure and the theory $\mathcal{T}_{arr}$ has the partial amalgamation property w.r.t. to it:

$$\begin{array}{rcl} W_{arr}(T_A, T_B) & = & T_1 \cup \{rd(a,i) \mid a, i \in st(T_1), rd(b,i) \in T_1 \cup st(T_B)\} \\ T_1 & = & st(T_0 \cup \{rd(a, diff(a,b)) \mid a, b \in st(T_A)\}) \\ T_0 & = & st(T_A \cup \{a \xrightarrow{k} b \mid a, b \in st(T_A \cap T_B)\}) \\ \text{where } a \overset{0}{\rightsquigarrow} b & = & a \text{ and } a \overset{k}{\rightsquigarrow} b = wr(a \xrightarrow{k-1} b, diff(a \overset{k-1}{\rightsquigarrow} b, b), rd(b, diff(a \overset{k-1}{\rightsquigarrow} b, b))), \text{ for all } k > 0 \end{array}$$

It is important to notice that the *diff* function is essential for the existence of ground interpolants. For instance, there is no ground interpolant, containing only the constants a, b and the function symbols rd, wr for the unsatisfiable conjunction:

$$\underbrace{b = wr(a,i,e)}_{A} \wedge \underbrace{j \neq k \wedge rd(a,j) \neq rd(b,j) \wedge rd(a,k) \neq rd(b,k)}_{B} \models \bot$$

On the other hand, $b = wr(a, diff(a,b), rd(b, diff(a,b)))$ is an interpolant for $A \wedge B$.

2.1.2 Theory of Lists with Reachability

The theory of lists and reachability is defined as the extension $\mathcal{T}_{llr} = \mathcal{T}_0 \cup \mathcal{K}_{llr}$ of the base theory $\mathcal{T}_0$ with a dedicated sort addr and extension symbols:

$$\begin{array}{rcl} rd & : & \mathsf{field} \times \mathsf{addr} \rightarrow \mathsf{addr} \\ wr & : & \mathsf{field} \times \mathsf{addr} \times \mathsf{addr} \rightarrow \mathsf{field} \\ R & : & \mathsf{field} \times \mathsf{addr} \times \mathsf{addr} \times \mathsf{addr} \rightarrow \mathsf{bool} \\ jp, lb & : & \mathsf{field} \times \mathsf{addr} \times \mathsf{addr} \rightarrow \mathsf{addr} \\ df & : & \mathsf{field} \times \mathsf{field} \rightarrow \mathsf{addr} \\ cy & : & \mathsf{field} \times \mathsf{addr} \rightarrow \mathsf{bool} \end{array}$$

Where field is an extension sort interpreted over functions between addr, rd and wr are the usual function read/write operations, $R(f,x,y,z)$ denotes reachability of y from x through f avoiding z, $jp(f,x,y)$ denotes the join point of the f-segments starting in x and y, $lb(f,x,y)$ is the address reachable from x through f just before y and $cy(f,x)$ is true iff x stands on an f-cycle. As before, df is the first address where two fields differ. The axioms from $\mathcal{K}_{llr}$ include the usual read-over-write, reflexivity and transitivity of reachability and the following axioms for jp, lb and cy, where $x \xrightarrow{f/z} y$ stands for $R(f,x,y,z)$ and $x \xrightarrow{f} y$ for $x \xrightarrow{f/y} y$:

$$\begin{array}{l} x \xrightarrow{f} jp(f,x,y) \\ x \xrightarrow{f} z \wedge y \xrightarrow{f} z \Rightarrow y \xrightarrow{f} jp(f,x,y) \\ x \xrightarrow{f} z \wedge y \xrightarrow{f} z \Rightarrow x \xrightarrow{f/z} jp(f,x,y) \\ y \xrightarrow{f} jp(f,x,y) \vee jp(f,x,y) = x \end{array} \qquad \begin{array}{l} x \xrightarrow{f/y.f} y \Rightarrow lb(f,x,y.f) = y \\ x \xrightarrow{f} y \wedge y \xrightarrow{f} x \Rightarrow cy(f,x) \vee x = y \\ cy(f,x) \wedge x \xrightarrow{f} y \Rightarrow y \xrightarrow{f} x \end{array}$$

As before, several examples show that ground interpolation requires the extension symbols jp, lb, df and cy, for instance in:

$$\underbrace{c \xrightarrow{f/c'} a \wedge a \neq c' \wedge a.f = c' \wedge P(a)}_{A} \wedge \underbrace{c \xrightarrow{f/c'} b \wedge b \neq c' \wedge b.f = c' \wedge \neg P(b)}_{B} \models \bot$$

where $P : \mathsf{addr} \rightarrow \mathsf{bool}$ is a predicate symbol. A ground interpolant here is $P(lb(f,c,c'))$.

The following function W_{llr} is an amalgamation closure. It is proved that the theory $\mathcal{T}_{llr}$ has the partial amalgamation property w.r.t. W_{llr} and thus, the ground interpolation property.

$$\begin{array}{rcl} W_{llr}(T_A, T_B) & = & T_5 \cup \{cy(f,a) \mid f,a \in T_5 \text{ shared with } T_B\} \\ T_5 & = & T_4 \cup \{a \xrightarrow{f/c}_b \mid a,b,c \in T_4\} \\ T_4 & = & T_3 \cup \{jp(f,a,b) \mid a,b,f \in T_3 \text{ shared with } T_B\} \\ T_3 & = & T_2 \cup \{lb(f,a,b) \mid a,b,f \in T_2\} \\ T_2 & = & T_1 \cup \{a.f \mid f,a \in st(T_1), a.g \in T_1 \cup st(T_B)\} \\ T_1 & = & st(T_0 \cup \{df(f,g).f \mid f,g \in st(T_A)\}) \\ T_0 & = & st(T_A \cup \{f \overset{k}{\rightsquigarrow} g \mid f,g \in st(T_A \cup T_B)\}) \end{array}$$

2.2 Interpolation for Separation Logic

This section summarizes the work of Albargouthi et al. [2]. This sets the grounds for interpolation-based abstraction refinement model checking of heap manipulating programs, describing an interpolation procedure for program paths labeled with Separation Logic (SL) assertions.

We consider a signature Σ, such that $\Sigma^s = \{\mathsf{Loc}, \mathsf{Data}, \mathsf{Bool}\}$, i.e. the only sorts are the boolean, data and *location* sort, with no function symbols defined on it, other than equality. Most definitions of common recursive data structures employed by programers (e.g. lists, trees, etc.) use a restricted fragment of quantifier-free SL, consisting of formulae $\Pi \wedge \Theta$, called *symbolic heaps*, in the following syntax, for *pure* (Π) and *spatial* (Θ) formulae defined as follows:

$$\Pi \quad ::= \quad x \approx y \mid \neg x \approx y \mid \varphi \mid \Pi_1 \wedge \Pi_2 \qquad\qquad \Theta \quad ::= \quad \mathsf{emp} \mid x \mapsto (y_1, \ldots, y_k) \mid P(\mathbf{R}, \mathbf{x}) \mid \Theta_1 * \Theta_2$$

where φ is any quantifier-free formula belonging to a theory $\mathcal{T}_{\mathsf{Data}}$ of the data sort. Inductive definitions are introduced as $P(\mathbf{R}, \mathbf{x}) \equiv \exists X_1 . \Pi_1 \wedge \Theta_1 \vee \ldots \vee \exists X_n . \Pi_n \wedge \Theta_n$, where $\mathbf{R}$ is a vector of second order variables and $\mathbf{x}$ is a vector of first order variables. Second order variables are interpreted as data sets, while first order variables as either location or data elements. Essentially, the second-order variables constrain the data values in every instance of the inductive predicate to belong to a given set.

The sort Loc is interpreted as an infinite countable set L. A *heap* is a finite partial mapping $h : \mathsf{L} \rightharpoonup_{\mathsf{fin}} (\mathsf{L} \cup \mathsf{D})^k$ associating locations with k-tuples of values, either locations or data. We denote by $\mathrm{dom}(h)$ the set of locations on which h is defined, by $\mathrm{img}(h)$ the set of locations occurring in the range of h, and by Heaps the set of heaps. Two heaps h_1 and h_2 are disjoint if $dom(h_1) \cap dom(h_2) = \emptyset$. In this case $h_1 \uplus h_2$ denotes their union, which is undefined if h_1 and h_2 are not disjoint. Given a valuation $\nu : \mathsf{Var} \to \mathsf{L} \cup \mathsf{D}$ and a heap h, the semantics of SL formulae is defined as follows:

$$\begin{array}{lcl}
\nu, h \models^{\mathsf{sl}} x \approx y & \iff & \nu(x) = \nu(y) \\
\nu, h \models^{\mathsf{sl}} \mathsf{emp} & \iff & h = \emptyset \\
\nu, h \models^{\mathsf{sl}} x \mapsto (y_1, \ldots, y_k) & \iff & h = \{\langle \nu(x), (\nu(y_1), \ldots, \nu(y_k)) \rangle\} \\
\nu, h \models^{\mathsf{sl}} \phi_1 * \phi_2 & \iff & \text{there exist } h_1, h_2 \in \mathsf{Heaps} : \ h = h_1 \uplus h_2 \text{ and } \mathcal{I}, h_i \models_{\mathsf{sl}} \phi_i, i = 1, 2 \\
\nu, h \models^{\mathsf{sl}} \exists x \,.\, \varphi(x) & \iff & \nu[x \leftarrow \ell], h \models_{\mathsf{sl}} \varphi(x), \text{ for some } \ell \in \mathsf{L}
\end{array}$$

The semantics of boolean connectives and data constraints is the usual one, omitted for brevity.

Example The following defines a binary tree parameterized by three second order variables: Q is a constraint on the data in each node and L (R) is a relation between the data in the current node and each data element in the left (right) subtree.

$$\begin{array}{lcll}
bt(Q, L, R, x) & \equiv & x = \mathsf{nil} \wedge \mathsf{emp} \vee & \\
 & & \exists d, l, r \,.\, Q(d) \wedge x \mapsto (d, l, r) & * \quad bt((\lambda a \,.\, Q(a) \wedge L(d, a)), L, R, l) \\
 & & & * \quad bt((\lambda a \,.\, Q(a) \wedge R(d, a)), L, R, r)
\end{array}$$

Instances of this predicate are: a binary search tree $bt((\lambda a.\top), (\lambda a, b.a \geq b), (\lambda a, b.a \leq b), x)$, and a heap tree of positive elements $bt((\lambda a.a \geq 0), (\lambda a, b.a \leq b), (\lambda a, b.a \leq b), x)$. □

Given a path $\pi = e_1; \ldots e_n$ of program statements e_i, the spatial interpolation algorithm proceeds in three phases:

1. the path is proved safe by annotation with strongest postconditions obtained from symbolic execution $\{\phi_0\}\, e_1\, \{\phi_1\} \ldots \{\phi_{n-1}\}\, e_n\, \{\phi_n\}$, where $\phi_0 = \mathsf{emp}$ and $\phi_i = \mathsf{Post}(e_i, \phi_{i-1})$, for all $i > 0$.

2. the safety proof is weakened backwards by computing spatial interpolants, namely for a Hoare triple $\{\phi\}\, e\, \{\psi\}$, we define $\mathsf{itp}(\phi, e, \psi)$ to be a formula such that $\phi \models \mathsf{itp}(\phi, e, \psi)$ and $\{\mathsf{itp}(\phi, e, \psi)\}\, e\, \{\psi\}$ is valid. At this point, the data constraints are not considered, but inductive predicates are introduced whenever possible, to weaken the strongest postconditions.
3. a recursion-free system of Horn clauses relating the second order variables introduced at the previous step is inferred. Any solution of this system in the theory $\mathcal{T}_{\mathsf{Data}}$ is used to refine the interpolants by adding data constraints.

The crux of the interpolation procedure is a *bounded abduction* procedure used in step 2 above. Namely, given SL formulae ϕ, φ, ψ, a solution to the bounded abduction problem $\phi \models \varphi * [] \models \psi$ is any SL formula such that $\phi \models \varphi * A \models \psi$ holds. The bounded abduction procedure is sound but not complete, thus the interpolation is not complete either. The most interesting case is $e \equiv \mathsf{assume}(\Pi)$, which is the only place where inductive predicates are introduced. Given a predicate atom $P(\mathbf{E})$, where P and $\mathbf{E}$ range over a finite set of choices, we define:

$$\mathsf{intro}(P(\mathbf{E}), \phi, \psi) = \begin{cases} P(\mathbf{E}) * A & \text{if } A \text{ is a solution of the bounded abduction } \phi \models P(\mathbf{E}) * [] \models \psi \\ \psi & \text{otherwise} \end{cases}$$

Then, considering all possible choices of predicate atoms $\{P_i(\mathbf{E}_i)\}_{i=1}^{n}$, we define:

$$\mathsf{itp}(\phi, \mathsf{assume}(\Pi), \psi) = \mathsf{intro}(P_1(\mathbf{E}_1), \phi \wedge \Pi, \mathsf{intro}(P_1(\mathbf{E}_1), \phi \wedge \Pi, \ldots, \mathsf{intro}(P_n(\mathbf{E}_n, \phi \wedge \Pi, \psi) \ldots))$$

This work [2] is a first step towards using abstraction refinement model checking for heap-manipulating programs. The method suffers however from the lack of performant and complete decision procedures for SL, in particular procedures for abductive reasoning. In D1.4 we propose a solution for abduction in quantifier-free SL without inductive predicates. An interesting direction, to be pursued in the rest of the VECOLIB project, is is the inference of inductive predicates from abduction proofs. Observe that the interpolation procedure from [2] uses a predefined set of inductive predicates when deriving interpolants, see e.g. the definition of $\mathsf{itp}(\phi, \mathsf{assume}(\Pi), \psi)$.

References

[1] P. A. Abdulla, M. F. Atig, Y.-F. Chen, L. Holík, A. Rezine, P. Rümmer, and J. Stenman. *String Constraints for Verification*, pages 150–166. Springer International Publishing, Cham, 2014.

[2] A. Albargouthi, J. Berdine, B. Cook, and Z. Kincaid. *Spatial Interpolants*, pages 634–660. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[3] W. Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. *J. Symbolic Logic*, 22(3):250–268, 09 1957.

[4] H. Hojjat and P. Ruemmer. Deciding and interpolating algebraic data types by reduction. Personal communication.

[5] D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for data structures. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '06/FSE-14, pages 105–116, 2006.

[6] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Mathematics of the USSR-Sbornik*, 32(2):129, 1977.

[7] V. Sofronie-Stokkermans. *Interpolation in Local Theory Extensions*, pages 235–250. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[8] N. Totla and T. Wies. Complete instantiation-based interpolation. *Journal of Automated Reasoning*, 57(1):37–65, Jun 2016.