Comparison Note

Logical Characterizations of Heap Abstractions

Abstract. This note contains a detailed discussion of a recent work that provides an alternative characterization of 3-valued structures using logical formulas, equivalent to the characterization presented in [4].

This note concerns a recent paper [1], which is an abbreviated version of a more extensive presentation of the results reported in [3]. That work addresses problems closely related to those addressed in [4], and in an extended and elaborated version of [4], submitted for publication. The purpose of this note is to explain the technical similarities and differences between the two works.

Overview of Main Results Both [3] and [4] provide two ways to identify the concretization of 3-valued structures using formulas: (i) concretization based on embedding, and (ii) concretization based on tight embedding. We will refer to these as *standard concretization* and *tight concretization*, respectively.

The work described in [3], inspired by [4], builds on the study of closure properties of related constraints [2], but with a different motivation: to prove algebraic closure properties of 3-valued structures. Towards this end, [3] starts by presenting the class of R_1 formulas, which explicitly represent the information stored in 3-valued structures. A structure's R_1 formula enumerates all different combinations of values of unary predicates,[1] and is therefore always exponential in the number of abstraction predicates, regardless of the size of the input 3-valued structure. This is used to derive important closure properties and subsequent reductions.

The motivation of [4] is to allow the extraction of most-precise information from 3-valued structures, by means of a theorem prover. The method presented in [4] generates a formula that is linear in the size of the input 3-valued structure. In the worst case, a 3-valued structure can include an exponential number of nodes, and in such cases a formula generated by this method can be as large as a formula generated by [3]. Inspired by [3], the extended version of [4] gives (in Appendix A.2) a simple semantic argument for the property of closure under negation, shown in [3] using a different formalism.

In the rest of the section, we provide a detailed comparison between the two papers' results on characterization of 3-valued logical structures using formulas. First, we compare the approaches for characterization taken in [4] and [3], under the assumption that there are no integrity rules.

Concretization of 3-Valued Structures In [4, Sec. 3], we present an algorithm $\widehat{\gamma}$ for generating a characteristic formula for the 2-valued structures that a 3-valued structure represents, assuming standard concretization. [3, Sec. 3] presents a method to generate a set of constraints in a canonical from, denoted by R_1, that also identify the 2-valued structures that a 3-valued structure represents, assuming standard concretization. The two methods are equivalent; that is, it is possible to convert a $\widehat{\gamma}$ formula into an R_1 formula that represents the same set of 2-valued structures, and vice versa.

The definitions in [3] are slightly different from those in [4], as explained below. Different definitions were chosen to simplify the presentation, and do not impose any limitations in

[1] To be precise, [3] refers to abstraction predicates. To be consistent with the terminology of [4], we assume that all unary predicates are abstraction predicates.

practice. First, the 3-valued structures defined in [3] always have a non-empty universe, as opposed to the definition used in [4]. Second, the notion of bounded structures defined in [3] forces unary predicates to have definite values, in contrast to the notion of bounded structures in [4].[2]

To identify the mapping of concrete node v into abstract node u, the formula $\widehat{\gamma}$ uses a subformula $node_u(v)$. The $node_u(v)$ formula for bounded structures, defined in [4, Sec. 3.3], guarantees that v and u agree on values of unary predicates $\mathcal{A}$, meaning that they have the same properties. The R_1 formulas use the notion of *cube* formulas over $\mathcal{A}$ to identify concrete nodes with certain properties; the *cube* formulas are *identical* to the $node_u(v)$ formulas for bounded structures.

The $\widehat{\gamma}$ and R_1 representations differ in the way that they enforce the presence and absence of nodes with certain properties. An R_1 formula is a normal form that explicitly represents all constraints. An R_1 formula explicitly enumerates all *cubes* defined over a vocabulary: the positive occurrence of a *cube* ensures the presence of a node with certain properties; the negative occurrence ensures the absence of such a node. As a result, the size of an R_1 formula is exponential in the number of unary predicates: the size of an R_1 formula does not depend on the size of the input 3-valued structure because the formula always includes all possible *cubes*.

In contrast, $\widehat{\gamma}$ does not perform an explicit enumeration of all *cubes*, but states that each concrete node must satisfy one of the positively occurring *cubes* (according to the ξ_{total} conjunct from [4, Eq. (3.4)]). As a result, the size of $\widehat{\gamma}$ is linear in the size of the 3-valued structure. The other classes of formulas introduced in [3], such as R_5, happen to allow similar compactness of representation, although they were not designed with the compactness of representation as the primary goal.

Other parts of $\widehat{\gamma}$ and R_1 formulas are the same. In an R_1 formula, condition 3 in [3, Def. 10] guarantees that no constraints are generated that involve negatively occurring *cubes* (which correspond to the absent nodes). In a $\widehat{\gamma}$ formula, the conjuncts $\xi[p]$ from [4, Eq. (3.6)] are generated only for positively occurring *cubes*.

Consequently, the conversion from R_1 to $\widehat{\gamma}$ formulas is trivial: eliminate R_1 literals with negative *cubes*; add the conjunct ξ_{total}, which ranges over all positive *cubes*; and perform some symbolic manipulations on the resulting formula. The conversion from $\widehat{\gamma}$ to R_1 is as follows: remove the conjunct ξ_{total} from $\widehat{\gamma}$; enumerate over all possible *cubes*; if a *cube* does not occur in $\widehat{\gamma}$, add its negation; and finally apply some symbolic manipulations. (The $\widehat{\gamma}$-to-R_1 conversion may be exponential in the size of $\widehat{\gamma}$.)

While formulas in [4] are used as an input to a theorem prover, R_1 formulas are merely the starting point for the richer syntactic classes, such as R_4 and R_5, that are used to obtain certain closure properties. The extended version of [4] obtains similar closure properties via a different approach.

Tight Concretization and ICA structures [4, Sec.7] provides an algorithm $\widehat{\gamma}_c$ for generating a characteristic formula that accepts the 2-valued structures that an ICA structure represents (i.e., $\widehat{\gamma}_c$ generates a characteristic formula assuming *tight* concretization). [3, Sec. 4] provides a set of constraints in canonical form, denoted by TR_1, that accepts 2-valued structures represented by a 3-valued structure using tight concretization. Because bounded structures in [3] allow only definite values of unary predicates, the tight concretization of a bounded structure in [3] defines the same set of 2-valued structures as the concretization of an ICA structure in [4]; therefore, $\widehat{\gamma}_c$ and TR_1 formulas are essentially

[2] Thus, [4] provides a characterization of a broader set of 3-valued structures, which means that for some 3-valued structures to be represented by R_1 formulas, they first need to be transformed into equivalent simpler structures.

equivalent. (The nuances discussed earlier in this section, in the context of standard concretization, remain relevant in the context of tight concretization.)
In the presence of integrity rules, $\widehat{\gamma}_c$ is equivalent to cTR_1 by simply joining the integrity formula to a TR_1 formula.

References

1. V. Kuncak and M. Rinard. Boolean algebra of shape analysis constraints. In *VMCAI*, pages 59–72, 2003.
2. V. Kuncak and M. Rinard. Existential heap abstraction entailment is undecidable. In *Static Analysis Symp.*, pages 418–438, 2003.
3. V. Kuncak and M. Rinard. On Boolean algebra of shape analysis constraints. Technical report, MIT, CSAIL, 2003. Available at "http://www.mit.edu/∼ vkuncak/papers/index.html".
4. G. Yorsh. Logical characterizations of heap abstractions. Master's thesis, Tel-Aviv University, Tel-Aviv, Israel, 2003. Available at "http://www.math.tau.ac.il/∼ gretay".