From Separation Logic to Hyperedge Replacement and Back

- Extended Abstract -

Mike Dodds

md466@cl.cam.ac.uk
University of Cambridge, UK[*]

1 Introduction

Hyperedge replacement grammars and separation-logic formulas both define classes of graph-like structures. In this paper, we describe two effective translations between restricted hyperedge replacement grammars and formulas in a fragment of separation logic. These translations preserve the semantics of formulas and grammars.

Hyperedge-replacement grammars [1] are a natural extension of context-free string grammars to the world of hypergraphs. An HR grammar defines language of structures that can be constructed from an initial graph.

Separation logic [2] is a recently-developed logic for specifying the properties of heaps that extends normal first-order logic with a so-called *separating* conjunction. This allows a formula to specify the *spatial relationships* between assertions in the heap. Recent work based on separation logic has made considerable progress in verifying programs with pointers [3].

Our translations demonstrate that formulas in our fragment of separation logic are of *corresponding expressive power* to HR grammars under our restrictions. We have used this correspondence to prove some interesting results about our fragment of separation logic using the theoretical results for hyperedge-replacement grammars.

2 The Intuitive Correspondence

A correspondence exists between separation logic formulas and hyperedge replacement grammars because (1) the recursive definitions commonly used in separation logic closely resemble hyperedge replacement productions, and (2) the separating property enforced by separating conjunction corresponds to the context-free property of hyperedge-replacement grammars.

The following example illustrates this correspondence.

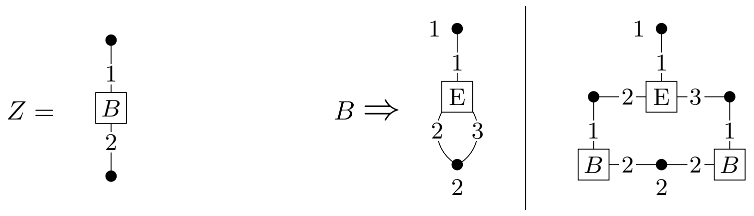
[*] Work completed during study towards a PhD at the University of York.

Example 1 (Binary tree). The predicate $bt(x)$ is defined as the least predicate satisfying the following equality.[1]

$$bt(x_1) = (x_1 \mapsto \mathsf{nil}, \mathsf{nil}) \vee (\exists x_2, x_3 : (x_1 \mapsto x_2, x_3) * bt(x_2) * bt(x_3))$$

$bt(x)$ is satisfied if either x points to a location holding a pair of nil-values, or if x points to a pair of locations, both of which also satisfy bt. The separating conjunction $*$ between the branch and the two subtrees differs from conventional conjunction in that it prohibits sharing between its conjuncts. This enforces the tree property by preventing sharing between the subtrees.

This predicate definition corresponds to the hyperedge replacement grammar $BT = \langle T, N, Z, P \rangle$. This defines the language of binary tree graphs with a shared leaf. The sets of terminal and non-terminal edge labels are respectively $T = \{E\}$ and $N = \{B\}$. The initial graph Z and set of productions P are:



The root (top node) of the initial graph Z corresponds to the variable x with which the predicate bt is called, while the leaf (bottom node) of Z corresponds to the nil constant. The individual cases of the production defined for label B in the grammar correspond to the two disjuncts defining the predicate bt. The first disjunct corresponds to a terminal branch, and the second a branch and a pair of child trees.

3 Defining the Translations

We define a mapping $g[\![\circ]\!]$ from separation logic formulas to hyperedge replacement grammars, and a reverse mapping $s[\![\circ]\!]$ from hyperedge-replacement grammars to separation logic formulas.

The mappings $g[\![\circ]\!]$ and $s[\![\circ]\!]$ are defined syntactically as functions over the structure of a grammar and a formula respectively. The elements of hyperedge replacement grammars and separation logic formulas are related as follows:

- The productions over a single nonterminal symbol in the grammar corresponds to the definition of a single recursive predicate in separation logic.
- Terminal edges in the grammar's initial graph and in the right-hand sides of productions correspond to separation logic's points-to assertion ($\mapsto$).
- Non-terminal edges in the grammar correspond to instances of recursively-defined predicates in separation logic. The attachment nodes of the edges correspond to the arguments passed to the predicate.

[1] To express this formula in our fragment of separation logic we make use of a recursive 'let' operator. For simplicity this is omitted from the example.

Hyperedge replacement grammars define languages of graphs, while separation logic formulas define classes of graph-like states. To resolve this mismatch between the domains we define the notion of a *heap-graph* for graphs which correspond to states, and a bijective mapping α from states to heap-graphs. Each node in a heap graph can be the first attachment point to at most one terminal edge. Our mappings are defined only over grammars which construct sets of heap-graphs.

Our mappings are defined over a fragment of full separation logic as given in (for example) [2]. This fragment includes separating conjunction ($*$), disjunction ($\vee$), 'points-to' ($\mapsto$) and existential quantification ($\exists$). To allow recursive definitions, we introduce a recursive let construct ($\mathsf{let}\ \Gamma\ \mathsf{in}\ P$). Omitted operators include conjunction, negation, and separating implication (the adjoint of separating conjunction).

4 Results

Our major results are as follows:

1. We have proved that our definitions of both $g[\![\circ]\!]$ and $s[\![\circ]\!]$ are *semantics preserving* modulo the mapping α. That is, $\alpha \circ g = g \circ \alpha$, and $\alpha^{-1} \circ s = s \circ \alpha^{-1}$.
2. As a consequence of (1), our fragment of separation logic is of *equivalent expressive power* to hyperedge-replacement grammars for heap graphs.
3. We have proved that the operators omitted from our fragment of separation logic *cannot* be simulated by a corresponding hyperedge replacement grammar. Notably conjunction corresponds to language intersection, and negation to language complement, both of which are known to be HR-inexpressible.
4. As a consequence of (2), results for hyperedge replacement languages, such as the inexpressibility results, can be imported into the fragment of separation logic. For example, the languages of red-black trees, balanced binary trees, grid graphs are all known to be HR-inexpressible. Therefore no formula exists in our fragment which is satisfied by the class of states containing these structures.

References

1. Drewes, F., Kreowski, H.J., Habel, A.: Hyperedge replacement graph grammars. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1. World Scientific (1997) 95–162
2. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proceedings of the Seventeenth Annual IEEE Symposium on Logic in Computer Science. (2002)
3. Distefano, D., O'Hearn, P.W., Yang, H.: A local shape analysis based on separation logic. In: 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. (2006)