On Spatial Conjunction as Second-Order Logic

Viktor Kuncak and Martin Rinard

{vkuncak,rinard}@csail.mit.edu
MIT CSAIL Technical Report 970, October 2004
Computer Science and AI Lab
Massachusetts Institute of Technology
Cambridge, MA 02139, USA**

Abstract. Spatial conjunction is a powerful construct for reasoning about dynamically allocated data structures, as well as concurrent, distributed and mobile computation. While researchers have identified many uses of spatial conjunction, its precise expressive power compared to traditional logical constructs was not previously known.

In this paper we establish the expressive power of spatial conjunction. We construct an embedding from first-order logic with spatial conjunction into second-order logic, and more surprisingly, an embedding from full second order logic into first-order logic with spatial conjunction. These embeddings show that the satisfiability of formulas in first-order logic with spatial conjunction is equivalent to the satisfiability of formulas in second-order logic. These results explain the great expressive power of spatial conjunction and can be used to show that adding unrestricted spatial conjunction to a decidable logic leads to an undecidable logic. As one example, we show that adding unrestricted spatial conjunction to two-variable logic leads to undecidability.

On the side of decidability, the embedding into second-order logic immediately implies the decidability of first-order logic with a form of spatial conjunction over trees. The embedding into spatial conjunction also has useful consequences: because a restricted form of spatial conjunction in two-variable logic preserves decidability, we obtain that a correspondingly restricted form of second-order quantification in two-variable logic is decidable. The resulting language generalizes the first-order theory of boolean algebra over sets and is useful in reasoning about the contents of data structures in object-oriented languages.

Keywords: program specification, separation logic, spatial conjunction, second-order logic, shape analysis, two-variable logic

1 Introduction

Separation logic with spatial conjunction operator was introduced as a technique for local reasoning about shared mutable data structures [25,44] and proved to be remarkably effective [4,5,12,13,43,45]. Similar constructs are present in formalisms based on process calculi and ambient calculi [10,11,14–17,35].

Despite the increasing range of results and applications of separation logic, the precise expressive power of spatial conjunction constructs is often not known. For example, the authors in [14,20] use the formalism of edge-labelled multigraphs and observe great expressive power of spatial logic for describing paths in a graph, but suggest that the relationship with second-order logic in this setting is not straightforward.

In [30,31] we defined the notion of spatial conjunction for arbitrary relational structures. Our notion of spatial conjunction splits relations into disjoint subsets and has a natural semantics that works for relations of any arity. The interpretation over relational

** Version produced October 25, 2004, 10:37am.

structures is an important step in enabling the combination of spatial conjunction with the traditional first-order and second-order logics [2, 24, 39] and their fragments. One such decidable fragment of first-order logic that is useful for reasoning about the heap is two-variable logic with counting [23], whose variable-free counterpart is role logic [28]. In [30, 31] we present a combination of two-variable logic with spatial conjunction defined on relational structures and show that it is useful for specifying generalized records that formalize role constraints [27]. To preserve the decidability of the notation, [30] imposes the following restriction on spatial conjunction: spatial conjunction may only be applied to formulas of (counting) quantifier nesting at most one. Under this assumption, we show that spatial conjunction can be eliminated using syntactic operations on formulas, which means that spatial conjunction not only preserves decidability, but leaves the expressive power of two-variable logic with counting unchanged.

Given the results in [30], a natural question to ask is: are we imposing an unnecessarily strong restriction by not allowing application of spatial conjunction to formulas with nested quantifiers; in particular, what is the decidability of logic that allows spatial conjunction of formulas with two nested quantifiers? The present paper gives an answer to this question: we establish that allowing spatial conjunction for formulas with nested quantifiers leads to an undecidable logic. This undecidability result turns out to be a consequence of an unexpectedly fundamental connection: *spatial conjunction can represent second-order quantification*. We obtain a striking contrast on the expressive power of logic depending on the use of spatial conjunction: if applied to formulas with no nesting of first-order counting quantifiers, the result is still two-variable logic with counting; if applied to formulas with nested first-order quantifiers, the resulting formulas can represent second-order formulas. This contrast can be viewed as a justification for the restriction imposed in [30].

Because it applies to both decidable and undecidable logics, the embedding of second-order logic into spatial conjunction yields not only undecidability, but also decidability results. Using the restriction on the use of spatial conjunction with the translation of second-order quantifiers yields a decidable notation with second-order quantifiers. This notation leads to a generalization of boolean algebra of sets to two-variable logic with counting extended with a form of second-order quantification; such notation is useful for reasoning about data structure abstractions [32, 33].

We also note that graph reachability, inductive definitions, spatial implication, and a parameterized version of spatial conjunction are all expressible in second-order logic. An interesting consequence of the embedding of second-order quantifiers into spatial conjunction is that all these constructs are expressible using spatial conjunction alone.

Moreover, the converse embedding holds as well: spatial conjunction is expressible in second-order logic. Together, these two results lead to a particularly simple characterization: *spatial conjunction and second-order logic are equivalent* (see Proposition 1 and Proposition 2 for the precise formulation of this equivalence).

The translation from spatial logic to second-order logic also has useful consequences. Namely, if we restrict the set of models to unions of trees, then monadic second-order logic is decidable. By translating restricted spatial logic formulas to monadic second-order logic, we obtain that spatial logic is decidable over trees as well.

In general, the equivalence for satisfiability between spatial conjunction and second-order logic improves our understanding of spatial conjunction and suggests that the definition of spatial conjunction on relational structures is a natural one. While it is less surprising that second-order logic can express the definition of spatial conjunction (we have observed this already in the technical report [31]), we found it quite surprising that spatial conjunction in first-order logic can express the entire second-order logic. The idea of both directions of our translation is remarkably simple, and this simplicity is reflected in the linear time complexity of formula translations: translation of spatial conjunction connectives into second-order logic mimics the semantics of spatial conjunction in terms of the existence of disjoint relations, and the translation from second-order logic into spatial conjunction takes the advantage of the non-determinism in splitting of the heap to simulate the existential quantifier.

Contributions. We summarize the contributions of this paper as follows.

1. We construct an equivalence-preserving translation of spatial conjunction into second-order quantifiers. We then show that this translation implies decidability of the first-order logic with a spatial conjunction interpreted over tree structures, when spatial conjunction splits only unary predicates.
2. We construct a satisfiability-preserving translation of second-order quantifiers into spatial conjunction, and derive the following consequences:
   (a) first-order logic with spatial conjunction has the expressive power of second-order logic, even if restricted to two first-order variables, and even if spatial conjunction is applied only to formulas of first-order quantifier nesting at most two (similar result holds for parameterized spatial conjunction that splits only *unary* predicates: the resulting logic is equivalent to *monadic* second-order logic);
   (b) two-variable logic with counting extended with second-order quantifiers that apply only to formulas with quantifier nesting at most one can be translated into two-variable logic with counting, and is therefore decidable;
   (c) graph reachability, inductive definitions, spatial implication, and generalized spatial conjunction are all expressible using first-order logic with spatial conjunction.

2 Preliminaries

In this section we present our definitions of relational structures as well as the semantics of second-order logic and spatial conjunction.

2.1 Relational Structures

Figure 1 presents the semantics of second-order logic formulas in relational structures, which is mostly standard. We use Var to denote first-order variables with typical representatives x, x_i. We use Σ to denote second-order variables (predicates), with a typical representative P, or $P^{(k)}$ when we wish to specify that the predicate symbol has arity k; alternatively we write $\mathsf{ar}(P) = k$.

For convenience we fix a universe U of all relational structures in a given context; we assume U is countable, but the cardinality of U does not play an important role for us. A relational structure, denoted e, is a valuation for first-order and second-order variables. As in first-order logic, for a first-order variable x, $e(x) \in U$ is an element of the domain, and for a predicate symbol P of arity $\mathsf{ar}(P) = k$, $e(P) \subseteq U^{\mathsf{ar}(k)}$ is a

$$
\begin{array}{rcl}
\mathsf{Var} & - & \text{fixed set of first-order variables} \\
\Sigma & - & \text{finite vocabulary of relational symbols (2nd order variables)} \\
\Sigma^{(k)} & = & \{P_1^{(k)}, P_2^{(k)}, \ldots, P_{C_k}^{(k)}\} \\
& & \text{relational symbols of arity } k, \mathsf{ar}(P^{(k)}) = k \\
\Sigma & = & \cup_{k=0}^{C} \Sigma^{(k)},\ C - \text{maximal arity} \\
U & - & \text{fixed universe (domain) of relational structures} \\
e & - & \text{relational structure (interpretation)} \\
e & : & (\mathsf{Var} \to U) \cup \bigcup_k (\Sigma^{(k)} \to 2^{U^k})
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{M}[\![x_1 = x_2]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & (e(x_1) = e(x_2)) \\
\mathcal{M}[\![P^{(k)}(x_1, \ldots, x_k)]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & (e(x_1), \ldots, e(x_k)) \in e(P^{(k)}) \\
\mathcal{M}[\![F_1 \wedge F_2]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & \mathcal{M}[\![F_1]\!]e \wedge \mathcal{M}[\![F_2]\!]e \\
\mathcal{M}[\![\neg F]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & \neg\mathcal{M}[\![F]\!]e \\
\mathcal{M}[\![\exists x.F]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & \exists v.\ \mathcal{M}[\![F]\!](e[x := v]) \\
\mathcal{M}[\![\exists P^{(k)}.F]\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & \exists r \subseteq U^k.\ \mathcal{M}[\![F]\!](e[P^{(k)} := r])
\end{array}
$$

Fig. 1. Semantics (Interpretation) of Second-Order Formulas in Relational Structures

relation of arity k. In this way we merge the model and the variable assignment, which makes it natural to define second-order quantification as in Figure 1. If v is a first-order or second-order variable, we use the standard notation $e[v := a]$ to denote the updated relational structure such that $e[v := a](v) = a$ and $e[v := a](v_1) = e(v_1)$ for $v_1 \neq v$. We treat equality in formulas as a logical symbol and interpret it in the standard way.

2.2 Spatial Conjunction

Figure 2 introduces our notion of spatial conjunction, denoted $\circledast$. We illustrate the intuition behind the definition of $\circledast$ in terms of combining the structures for which the formula is true. Suppose $\Sigma = \{P^{(2)}\}$ has only one binary relation symbol, so the relational structures are graphs. If e_1 is a structure such that $\mathcal{M}[\![F_1]\!]e_1$ and e_2 is a structure such that $\mathcal{M}[\![F_2]\!]e_2$, then if the edges of $e_1(P^{(2)})$ and $e_2(P^{(2)})$ are disjoint, the structure with relation $e(P^{(2)}) = e_1(P^{(2)}) \cup e_2(P^{(2)})$ satisfies $\mathcal{M}[\![F_1 \circledast F_2]\!]e$. In general, there is one relation e for each pair of models e_1 and e_2 that can be combined. There are three models of $F_1 \circledast F_2$ in Figure 2; there is only one pair of relations that cannot be combined, because of an overlapping edge from w to x.

The definition of spatial conjunction in Figure 2 is identical to the one we use in [30, 31]. In our setup, similarly to other notions of spatial conjunction [15, 25], a formula $F_1 \circledast F_2$ holds for a relational structure if and only if the structure can be split into two disjoint structures where F_1 holds for the first component and F_2 holds for the second component. The difference with [15] is that we use general relational structures which correspond to labelled graphs as opposed to multigraphs. Our notion of splitting of relational structures, given by condition $\mathsf{splitStruct}_\sigma(e, e_1, e_2)$, reduces to partitioning each relation in σ. For the definition of spatial conjunction $\circledast$ we let $\sigma = \Sigma$ where

$$\mathcal{M}[\![F_1 \circledast F_2]\!]e \overset{\text{def}}{\iff} \exists e_1, e_2.\ \mathsf{splitStruct}_{\Sigma}(e, e_1, e_2)\ \wedge\ \mathcal{M}[\![F_1]\!]e_1 \wedge \mathcal{M}[\![F_2]\!]e_2$$

$$\mathsf{splitStruct}_{\sigma}(e, e_1, e_2) \overset{\text{def}}{\iff} \bigwedge_{r \in \sigma} \mathsf{splitRel}(e(r), e_1(r), e_2(r)) \wedge \bigwedge_{r \in \Sigma \setminus \sigma} (e_1(r) = e(r) \wedge e_2(r) = e(r))$$

$$\mathsf{splitRel}(r, r_1, r_2) \overset{\text{def}}{\iff} (r = r_1 \cup r_2\ \wedge\ r_1 \cap r_2 = \emptyset)$$

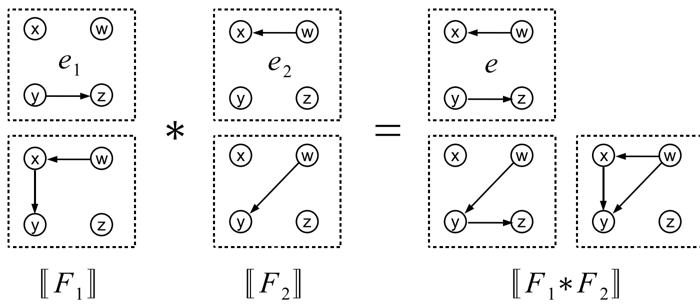

Fig. 2. Semantics of Spatial Conjunction $\circledast$.

$$\mathcal{M}[\![F_1 \circledast_{\sigma} F_2]\!]e \overset{\text{def}}{\iff} \exists e_1, e_2.\ \mathsf{splitStruct}_{\sigma}(e, e_1, e_2)\ \wedge\ \mathcal{M}[\![F_1]\!]e_1 \wedge \mathcal{M}[\![F_2]\!]e_2$$

$$F_1 \circledast F_2 \iff F_1 \circledast_{\Sigma} F_2$$

Fig. 3. Parameterized Spatial Conjunction $\circledast_{\sigma}$

Σ is the set of all relation symbols; it is also natural to allow a generalized spatial conjunction $\circledast_{\sigma}$ in Figure 3 that takes the set of predicate symbols σ as an argument, then splits relations in σ and preserves the relations in $\Sigma \setminus \sigma$. For example, if we let $\sigma = \Sigma^{(1)}$, then the conjunction $\circledast_{\sigma}$ splits only unary relations. The results of this paper imply that $\circledast_{\Sigma}$ corresponds to full second-order logic, whereas $\circledast_{\Sigma^{(1)}}$ corresponds to monadic second-order logic.

Our defi nition of spatial conjunction above is not the only one possible, but there are several reasons to consider it as a natural defi nition of spatial conjunction:

- Our defi nition is close to the defi nition of [25]. A relational structure can represent a store by modelling each store location as a pair of an object and one of the fi nitely many predicate symbols; this view is appropriate for type-safe languages such as Java, ML, and O'Caml.
- The only difference compared to [15] is that we use relations as sets of tuples where [15] uses multigraphs as multisets of tuples; we believe that our results can provide useful insight into languages such as [15] as well.
- With the appropriate defi nition of spatial implication $-\!\!\circledast$ (Figure 8) corresponding to conjunction $\circledast$, our model validates the axioms of bunched implications [25,42].
- We can naturally describe concatenation of generalized records [30,31], which cannot be expressed using standard logical operations.

The main claim of this paper is that our notion of spatial conjunction is equivalent for satisfi ability to second-order quantifi cation. This equivalence can be viewed as an-other argument in favor of the definitions we adopt. We proceed to demonstrate both directions of the equivalence, and then present some consequences of the result.

3 Representing Spatial Conjunction $\circledast$ in Second-Order Logic

In this section we give a translation from the first-order logic with spatial conjunction to second-order logic. The consequence of this translation is an upper bound on the expressive power of spatial conjunction. Because our translation applies to all relational structures, if we restrict the set of relational structures so that second-order logic becomes decidable, then the corresponding spatial logic is decidable on the restricted set of structures as well.

$$\sigma = \{P_1, \ldots, P_n\}$$

spatial conjunction elimination:

$$\begin{aligned}
\mathcal{T}_{\circledast\mapsto 2}[\![F' \circledast_\sigma F'']\!] = {} & \exists P'_1, \ldots, P'_n, P''_1, \ldots, P''_n. \\
& \textstyle\bigwedge_{i=1}^{n} \mathsf{synSplitRel}(P_i, P'_i, P''_i) \wedge \\
& F'[P_i := P'_i]_{i=1}^{n} \ \wedge \ F''[P_i := P''_i]_{i=1}^{n}
\end{aligned}$$

$$\begin{aligned}
\mathsf{synSplitRel}(P, P', P'') = {} & \forall x_1, \ldots, x_k. \\
& \quad (P(x_1, \ldots, x_k) \Leftrightarrow P'(x_1, \ldots, x_k) \vee P''(x_1, \ldots, x_k)) \wedge \\
& \quad \neg(P'(x_1, \ldots, x_k) \wedge P''(x_1, \ldots, x_k)) \\
& k = \mathsf{ar}(P) = \mathsf{ar}(P') = \mathsf{ar}(P'')
\end{aligned}$$

recursive translation function:

$$\begin{aligned}
\mathcal{RT}_{\circledast\mapsto 2}[\![x_1 = x_2]\!] &= (x_1 = x_2) \\
\mathcal{RT}_{\circledast\mapsto 2}[\![P^{(k)}(x_1, \ldots, x_k)]\!] &= P^{(k)}(x_1, \ldots, x_k) \\
\mathcal{RT}_{\circledast\mapsto 2}[\![F_1 \wedge F_2]\!] &= \mathcal{RT}_{\circledast\mapsto 2}[\![F_1]\!] \wedge \mathcal{RT}_{\circledast\mapsto 2}[\![F_2]\!] \\
\mathcal{RT}_{\circledast\mapsto 2}[\![\neg F]\!] &= \neg \mathcal{RT}_{\circledast\mapsto 2}[\![F]\!] \\
\mathcal{RT}_{\circledast\mapsto 2}[\![\exists x. F]\!] &= \exists x. \mathcal{RT}_{\circledast\mapsto 2}[\![F]\!] \\
\mathcal{RT}_{\circledast\mapsto 2}[\![\exists P^{(k)}. F]\!] &= \exists P^{(k)}.\ \mathcal{RT}_{\circledast\mapsto 2}[\![F]\!] \\
\mathcal{RT}_{\circledast\mapsto 2}[\![F_1 \circledast F_2]\!] &= \mathcal{T}_{\circledast\mapsto 2}[\![\mathcal{RT}_{\circledast\mapsto 2}[\![F_1]\!] \circledast_\Sigma \mathcal{RT}_{\circledast\mapsto 2}[\![F_2]\!]]\!] \\
\mathcal{RT}_{\circledast\mapsto 2}[\![F_1 \circledast_{\Sigma^{(1)}} F_2]\!] &= \mathcal{T}_{\circledast\mapsto 2}[\![\mathcal{RT}_{\circledast\mapsto 2}[\![F_1]\!] \circledast_{\Sigma^{(1)}} \mathcal{RT}_{\circledast\mapsto 2}[\![F_2]\!]]\!]
\end{aligned}$$

translation correctness lemmas:

$$\mathcal{M}[\![F'[P_i := P'_i]_{i=1}^{n}]\!](e[P'_i := r_i]_{i=1}^{n}) = \mathcal{M}[\![F']\!](e[P_i := r_i]_{i=1}^{n}) \qquad \text{for } P_i \text{ fresh in } F'$$

$$\mathcal{M}[\![\mathcal{T}_{\circledast\mapsto 2}[\![F]\!]]\!]e = \mathcal{M}[\![F]\!]e$$

$$\mathcal{M}[\![\mathcal{RT}_{\circledast\mapsto 2}[\![F]\!]]\!]e = \mathcal{M}[\![F]\!]e$$

Fig. 4. Translation of Spatial Conjunction into Second-Order Logic

Figure 4 presents the translation from first-order logic extended with spatial conjunction into second-order logic. The translation directly mimics the semantics of $\circledast$ and follows from the fact that second-order logic can essentially quantify over its entire domain and can express disjointness of relations. Indeed, the truth value of a formula depends only on finitely many first and second-order variables, and second-order logic can quantify over each of these variables, which amounts to quantification over relational structures.

The translation in Figure 4 introduces two fresh predicate symbols P_i', P_i'' for each predicate symbol P_i and asserts that P_i' and P_i'' split P_i. The translation then replaces the predicates P_i with the corresponding predicates P_i' in the first formula F', and replaces the predicates P_i with the predicate P_i'' in the second formula F''. The correctness of the translation follows from the definitions, using lemmas in Figure 4 and structural induction. We conclude the following.

Proposition 1. *If F is a second-order logic formula potentially containing spatial conjunction, then $\mathcal{RT}_{\circledast\mapsto 2}[\![F]\!]$ is an equivalent second-order logic formula without spatial conjunction; we have $\mathcal{M}[\![\mathcal{RT}_{\circledast\mapsto 2}[\![F]\!]]\!]e = \mathcal{M}[\![F]\!]e$ for all relational structures e that interpret F. Moreover, if F is a monadic second-order logic formula with $\circledast_{\Sigma^{(1)}}$ as the only spatial conjunction operator, then the resulting formula is a monadic second-order logic formula.*

4 Representing Second-Order Quantifiers using $\circledast$

This section shows that second-order quantifiers can be represented using spatial conjunction. Among the consequences of this result are the fact that first-order logic with spatial conjunction has the expressive power of second-order logic (even if restricted to two first-order variables where the spatial conjunction connects only formulas of first-order quantifier nesting at most two), that two-variable logic with counting extended with second-order quantifiers that apply only to formulas with quantifier nesting at most one is decidable, and that inductive definitions, spatial implication, and generalized spatial conjunction are expressible using first-order logic with spatial conjunction.

Figure 5 presents the translation of second-order quantifiers into spatial conjunction. As in the case of the converse translation in Section 3, the intuition behind the translation is to exploit the semantics of spatial conjunction in Figure 3. This time, however, we use the more complex operation—splitting of relational structures—to simulate an existential quantifier over relations, which leads to apparent difficulties. At first sight it appears that heap splitting fails to have the effect of an existential quantifier over a relation predicate, for two reasons:

1. splitting relational structures splits existing relations, which means that the interpretations of relations in the resulting structure are subsets of the interpretation of relations in the enclosing structure;
2. splitting of relational structures splits all relations, and not just the interpretation of one predicate.

We solve both of these problems when translating a formula F_0 with second-order quantifiers, as follows. We first rename all bound second-order variables (denoted $\mathsf{BV2}(F_0)$) to ensure that they are all distinct and that they differ from the free variables in F_0. In the translated formula, even the bound second-order variables $\mathsf{BV2}(F_0)$ become free second-order variables, which are allowed in first-order logic. To solve the first problem, instead of considering all possible relational structures e, we consider only those relational structures that map the variables $\mathsf{BV2}(F_0)$ to full relations; we use the conjunct $\mathsf{allpreds}$ to ensure that only such structures are considered for the interpretation of the final translated formula $\mathcal{FT}_{2\mapsto\circledast}[\![F_0]\!]$. We translate the formula using the recursive translation function denoted $\mathcal{T}_{2\mapsto\circledast}[\![F]\!]$, which walks the formula tree and applies the

$$\begin{aligned}
\mathsf{BV2}(F) &- \text{second-order variables bound in } F \\
\mathsf{V2}(F) &- \text{second-order variables in } F \\
F_0 &- \text{a formula without spatial conjunction } \circledast
\end{aligned}$$

assumption: all bound variables in F_0 are mutually distinct and distinct from free variables in F_0

$$\begin{aligned}
\mathsf{allpreds} &\equiv \forall x. \bigwedge_{Q \in \mathsf{BV2}(F_0)} Q(x) \\
\mathsf{nonebut}(P) &\equiv \forall x. \bigwedge_{Q \in \mathsf{V2}(F_0) \setminus \{P\}} \neg Q(x)
\end{aligned}$$

translation of second-order quantifier:

$\mathcal{T}_{2\mapsto\circledast}[\![\exists P.\, F]\!] \;=\; \mathsf{nonebut}(P) \circledast F$

recursive translation function:

$$\begin{aligned}
\mathcal{RT}_{2\mapsto\circledast}[\![x_1 = x_2]\!] &= (x_1 = x_2) \\
\mathcal{RT}_{2\mapsto\circledast}[\![P^{(k)}(x_1, \ldots, x_k)]\!] &= P^{(k)}(x_1, \ldots, x_k) \\
\mathcal{RT}_{2\mapsto\circledast}[\![F_1 \wedge F_2]\!] &= \mathcal{RT}_{2\mapsto\circledast}[\![F_1]\!] \wedge \mathcal{RT}_{2\mapsto\circledast}[\![F_2]\!] \\
\mathcal{RT}_{2\mapsto\circledast}[\![\neg F]\!] &= \neg \mathcal{RT}_{2\mapsto\circledast}[\![F]\!] \\
\mathcal{RT}_{2\mapsto\circledast}[\![\exists x. F]\!] &= \exists x. \mathcal{RT}_{2\mapsto\circledast}[\![F]\!] \\
\mathcal{RT}_{2\mapsto\circledast}[\![\exists P^{(k)}. F]\!] &= \mathcal{T}_{2\mapsto\circledast}[\![\exists P^{(k)}.\, \mathcal{RT}_{2\mapsto\circledast}[\![F]\!]]\!]
\end{aligned}$$

final translation of a formula:

$\mathcal{FT}_{2\mapsto\circledast}[\![F_0]\!] \;=\; \mathsf{allpreds} \;\wedge\; \mathcal{RT}_{2\mapsto\circledast}[\![F_0]\!]$

translation correctness lemmas:

$$\begin{aligned}
\mathcal{M}[\![\exists P.\, F]\!]e \quad &= \quad \mathcal{M}[\![\mathcal{T}_{2\mapsto\circledast}[\![\exists P.\, F]\!]]\!](e[P := U^{\mathsf{ar}(P)}]) \\
\mathcal{M}[\![F_0]\!]e \quad &= \quad \mathcal{M}[\![\mathcal{FT}_{2\mapsto\circledast}[\![F_0]\!]]\!](e[P := U^{\mathsf{ar}(P)}]_{P \in \mathsf{BV2}(F_0)}) \\
\exists e. \mathcal{M}[\![F_0]\!]e \;&\iff\; \exists e. \mathcal{M}[\![\mathcal{FT}_{2\mapsto\circledast}[\![F_0]\!]]\!]e
\end{aligned}$$

Fig. 5. Translation of Second-Order Quantifiers into Spatial Conjunction

translation of the existential quantifier. The translation of the existential quantifier, denoted $\mathcal{T}_{2\mapsto\circledast}[\![\_]\!]$, replaces the quantifier $\exists P.\, F$ with the formula $\mathsf{nonebut}(P) \circledast F$. The spatial conjunct $\mathsf{nonebut}(P)$ solves the second problem above, by asserting that all relations other than P are empty, and leaving the interpretation of relation P unconstrained. As a result, the interpretation of P in F is arbitrary, achieving the effect of existential quantification, and the interpretations of the remaining quantifiers remain the same, as desired.

Soundness of the translation in Figure 5 is given by equisatisfiability, or equivalence on a reasonably restricted class of structures, as summarized by the following proposition.

Proposition 2. *Let F_0 be a second-order logic formula in which each bound variable is distinct from all other variables in F_0. Then $\mathcal{F}\mathcal{T}_{2\mapsto\circledast}[\![F_0]\!]$ is a formula in first-order logic with spatial conjunction, such that F_0 has a model if and only if $\mathcal{F}\mathcal{T}_{2\mapsto\circledast}[\![F_0]\!]$ has a model. Moreover, if e ranges over structures that assign full relations to predicate symbols not free in F_0, then the transformation is equivalence preserving, that is, $\mathcal{M}[\![F_0]\!]e$ if and only if $\mathcal{M}[\![\mathcal{F}\mathcal{T}_{2\mapsto\circledast}[\![F_0]\!]]\!]e$. Finally, if all second-order quantifiers in F_0 are monadic, then F_0 can be translated into formula containing only $\circledast_{\Sigma^{(1)}}$ instead of $\circledast$.*

5 Consequences of the Equivalence

This section presents the consequences of the equivalence between spatial conjunction and second-order quantification.

5.1 Spatial Conjunction on Tree Structures is Decidable

This section summarizes one interesting consequence of the equivalence between spatial conjunction and second-order logic with respect to tree structures.

Let us restrict our attention to relational structures that interpret predicates of arity at most two. Such relational structures correspond to graphs with labelled nodes and edges. We say that a relational structure is a *forest* if the directed graph obtained by erasing all labels is a directed forest, where by a directed forest we mean a directed graph with no cycles where each node has an in-degree at most one. We then have the following lemma.

Lemma 3. *If e is a forest, and $\mathsf{splitStruct}_{\Sigma}(e, e_1, e_2)$ holds, then both e_1 and e_2 are forests.*

The previous lemma easily follows by contraposition: if e_1 or e_2 have a cycle so does e, and if e_1 or e_2 have a node with in-degree two or more, so does e. This lemma implies that, when evaluating the meaning $\mathcal{M}[\![F]\!]e$ of formula in first-order logic with spatial conjunction, it suffices to restrict the top-level structure e to be a forest for all structures occurring in the semantics of subformulas of F to be forests, which means that the semantics of spatial conjunction over forests is equivalent to the semantics in Figure 1. Using Proposition 1 we then obtain as a special case $\mathcal{M}[\![F]\!]e \iff \mathcal{M}[\![\mathcal{R}\mathcal{T}_{\circledast\mapsto 2}[\![F]\!]]\!]e$. By decidability of monadic second-order logic over trees [18], we conclude the following.

Proposition 4. *The satisfiability (and therefore the validity) problem of first-order logic extended with spatial conjunction $\circledast_{\Sigma^{(1)}}$ is decidable.*

5.2 Undecidable Extension of Two-Variable Logic

This section notes a consequence of Proposition 2 on extensions of decidable fragments of first-order logic with spatial conjunction. It is motivated by the following fact, proven in [30]:

Fact 5. *Two variable logic with counting extended with spatial conjunction on formulas with no nested counting quantifiers is decidable.*

A natural question to ask is: what is the decidability of the notation if we allow spatial conjunction of formulas with quantifier nesting two or more. The answer is that the resulting notation is undecidable. Namely, if we have only binary relation symbols, we obtain a logic equivalent to full second-order logic, and already first-order logic in the language with binary relation symbols is undecidable.

The reason for obtaining second-order logic when allowing spatial conjunction of formulas with nested quantifiers is that it is possible to simulate first-order quantifiers using second-order quantifiers. We can represent a first-order variable such as x by a second-order variable P_x bounded by the property $\exists_1 z.P_x(z)$, and then replace each binary relation symbol $f(x, y)$ with a formula of the form

$$\forall u.\forall v.\ P_x(u) \wedge P_y(v) \Rightarrow f(u, v), \tag{1}$$

which uses only two first-order variables and has quantifier nesting of two. Similarly, the use of a unary relation symbol $P(x)$ can be replaced by $\forall u.\ P_x(u) \Rightarrow P(u)$.

Now consider a second-order logic formula with binary and unary relation symbols and no restrictions on the number of first-order variables. As described above, we can reduce such formula to an equisatisfiable formula that uses only two first-order variables. We can then apply the translation in Figure 5 to eliminate second-order quantifiers. Because formulas $\mathsf{allpreds}$ and $\mathsf{nonebut}(P)$ have the quantifier depth at most one, the result is a formula with spatial conjunction that is applied to quantifiers of depth at most two and that uses at most two first-order variable names. Moreover, the resulting formula is equisatisfiable by Proposition 2. Because the satisfiability of second-order logic formulas is undecidable, the translation of second-order logic formulas into formulas with spatial conjunction implies undecidability of formulas with spatial conjunction applied to formulas with quantifiers depth of two or more.

Proposition 6. *Two variable logic with counting extended with spatial conjunction $\circledast$ on formulas with quantifier nesting at most two is undecidable. The result applies to spatial conjunction $\circledast_{\Sigma(1)}$ as well.*

5.3 Decidable Second-Order Quantification in Two-Variable Logic

We next state a positive consequence of the Fact 5 and Proposition 2.

Proposition 7. *Two variable logic with counting extended with second-order quantification on formulas with no nested counting first-order quantifiers is decidable.*

Just like the previous Proposition 6, Proposition 7 follows from applying the translation in Figure 5 and observing that the resulting formula has no nested first-order quantifiers, and is equisatisfiable by Proposition 2. Applying Proposition 5, we can decide the satisfiability of the resulting formula, which gives the satisfiability of the original formula as well.

To see why Proposition 7 is interesting, note that Proposition 7 places no restrictions on the number of second-order quantifiers used on a formula with no nested first-order quantifiers. Next, recall that monadic second-order logic of a set (with no relation symbols) is just the first-order logic of boolean algebra of sets, which is decidable by quantifier elimination [48] (for an overview of quantifier elimination for boolean algebra see, for example, [29]). We therefore observe that the language permitted by Proposition 7 is a proper generalization of boolean algebra of sets; it is a generalization that allows stating set properties in a neighborhood of a pair of objects given by two free variables of a formula in two-variable logic with counting.

While we have found the first-order theory of boolean algebra of sets to be useful for reasoning about the content of global data structures [32], the generalization presented in this section allows reasoning about sets that exist in the neighborhood of an object denoted by a first-order variable. In other words, this specification language allows us to reason about the content of data structures associated with individual objects (which are common in object-oriented programming languages), as opposed to just reasoning about global data structures.

Comparing the results of this section and Section 5.2, we note the crucial role of the restriction on quantifier nesting: with no nested first-order quantifiers, it is not possible to use second-order variables to simulate first-order variables because it is not possible to establish the correlation of the form (1).

5.4 Expressing Inductive Definitions and Spatial Implication

We next review the fact that inductive definitions (and therefore transitive closure) are definable in second-order logic. This fact is of interest because it implies that inductive definitions can be represented using spatial conjunction, which leads to a surprising conclusion that inductive definitions do not increase the expressive power of first-order logic with spatial conjunction. We similarly observe that the spatial implication corresponding to spatial conjunction is expressible in second-order logic and therefore expressible using spatial conjunction. All these consequences follow from Proposition 2.

$$\mathcal{M}[\![\mathsf{letrec}\ P^{(k)}(x_1,\ldots,x_k) = F\ \mathsf{in}\ G]\!] \overset{\mathsf{def}}{\iff} \mathcal{M}[\![G[P^{(k)} := \mathrm{LFP}_{P^{(k)},x_1,\ldots,x_k} F]]\!]$$

$$\mathcal{M}[\![\mathrm{LFP}_{P^{(k)},x_1,\ldots,x_k} F(y_1,\ldots,y_k)]\!]e \overset{\mathsf{def}}{\iff}$$
$$(e(y_1),\ldots,e(y_k)) \in \mathsf{lfp}(\lambda r.\{(v_1,\ldots,v_k) \mid \mathcal{M}[\![F]\!]e[P^{(k)} := r, x_1 := v_1,\ldots,x_k := v_k])$$

Fig. 6. Semantics of Inductive Definitions

Figure 6 presents the semantics of inductive definitions. The syntax of the least-fixpoint operator is

$$\mathrm{LFP}_{P^{(k)},x_1,\ldots,x_k} F(y_1,\ldots,y_k)$$

where F is a formula that may contain new free variables $P^{(k)}, x_1,\ldots,x_k$. The meaning of the least-fixpoint operator is that the relation which is the least fixpoint of the monotonic transformation on predicates

$$(\lambda x_1,\ldots,x_k.P^{(k)}(x_1,\ldots,x_k)) \mapsto (\lambda x_1,\ldots,x_k.F)$$

holds for $y_1,\ldots,y_k$. To ensure the monotonicity of the transformation on predicates, we require that $P^{(k)}$ occurs only positively in F.

$$\mathcal{T}_{\mathsf{Ind}\to 2}[\![\mathrm{LFP}_{P^{(k)},x_1,\ldots,x_n}\, F(y_1,\ldots,y_n)]\!] = \\ \quad \forall P.\, (\forall x_1,\ldots,x_n.\, (F \Leftrightarrow P(x_1,\ldots,x_n))) \Rightarrow P(y_1,\ldots,y_n)$$

Soundness:

$$\mathcal{M}[\![\mathcal{T}_{\mathsf{Ind}\to 2}[\![\mathrm{LFP}_{P^{(k)},x_1,\ldots,x_n}\, F(y_1,\ldots,y_n)]\!]]\!]e = \mathcal{M}[\![\mathrm{LFP}_{P^{(k)},x_1,\ldots,x_n}\, F(y_1,\ldots,y_n)]\!]e$$

Fig. 7. Expressing Least Fixpoint in Second-Order Logic

Figure 7 shows that least fi xpoint operator is expressible in second-order logic. The property that P is a fi xpoint of F is easily expressible. To encode that $y_1,\ldots,y_n$ hold for the *least* fi xpoint of F, we state that $y_1,\ldots,y_n$ hold for all fi xpoints of F, using universal second-order quantifi cation over P.

$$\begin{array}{rcl} \mathcal{M}[\![F' {-\!\!\circledast} F'']\!]e & \stackrel{\mathsf{def}}{\Longleftrightarrow} & \forall e', e''.\, (\mathsf{splitStruct}_{\Sigma}(e'', e, e') \;\wedge\; \mathcal{M}[\![F']\!]e') \Rightarrow \mathcal{M}[\![F'']\!]e'' \\ \mathcal{T}_{\circledast\mapsto 2}[\![F' {-\!\!\circledast} F'']\!] & = & \forall P_1', \ldots, P_n', P_1'', \ldots, P_n''. \\ & & \Big(\bigwedge_{i=1}^{n} \mathsf{synSplitRel}(P_i'', P_i, P_i') \;\wedge\; F'[P_i := P_i']_{i=1}^{n} \Big) \Rightarrow \\ & & \quad F''[P_i := P_i'']_{i=1}^{n} \end{array}$$

Fig. 8. Semantics of Spatial Implication

Figure 8 presents the semantics of the spatial implication operation that along with spatial conjunction $\circledast$ validates the axioms of bunched implications [25, 42]. Figure 8 also presents the translation of ${-\!\!\circledast}$ into second-order logic; the translation is analogous to the translation of spatial conjunction in Figure 4. (As usual, the universal quantifi ers can be expressed using the existential quantifi er and negation.)

We summarize the results of this section as follows.

Proposition 8. *Graph reachability, inductive definitions, spatial implication, and generalized spatial conjunction are all expressible using first-order logic with spatial conjunction.*

6 Related Work

The use of separation logic for reasoning about shared mutable data structures started recently [25, 44] using ideas from [9] and proved very fruitful [5, 12, 13, 43, 45]. Our notion of spatial conjunction is defi ned on relational structures rather than on mappings from memory locations to values, but our model can represent a location as a pair containing 1) an object and 2) one of the fi nitely many fi eld names. Relational structures can naturally represent memory models of languages with destructive updates [8, 34, 36, 37, 46, 47, 51] and can also model concurrency and temporal logic specifi cations [52, 53].

Process calculi [11] and ambient calculi [17] can reason about space and locality as well as concurrency; these ideas also extend to graph-based structures [14, 15]. The results closest to ours are are [14, 15, 20]; they are based on edge-labelled multigraphs, and do not establish the full equivalence with second-order logic. Graph-based structures in [15] are close to the relational structures that we use, but use multisets of edges instead of sets of edges. Similarly to spatial logic, type systems for reasoning about aliasing [21, 22, 49, 50] typically contain join operators that combine independent portions of store, although they are often based on linear types as opposed to separation logic.

Our work clarifies the relationship between separation logic and traditional first-order logic [39] and second-order logic [2] and explains surprising expressive power of spatial conjunction without inductive definitions in expressing reachability properties [14, 15]. The understanding of separation logic in connection to other formalisms is useful both for manual reasoning [5] and automated reasoning about programs with shared mutable data structures [1,6,7,10,19,34,36,37,40,41,46,47]. Decidability and complexity results of underlying logics and constraint systems are particularly important for automated reasoning [3,4,12,13,26,35,38].

We have previously used the notion of spatial logic on relational structures in [30, 31] and presented a novel use of spatial conjunction to describe concatenation of generalized records. In [30, 31] we take advantage of the definition of spatial conjunction on relational structure to combine it with a fragment of first-order logic: we present a decidable extension of two-variable logic with counting and its variable-free version role logic [28]. The encoding of spatial conjunction in second-order logic appears in the technical report [31]; we have since discovered the converse (and to us more surprising) encoding. The converse encoding gives justification to the restriction in [30] by showing that the absence of the restriction leads to an undecidable, and in fact, extremely expressive, logic. Moreover, the results of the present paper show how to use second-order quantifiers in two-variable logic while preserving decidability. The resulting notation generalizes the language of boolean algebra of sets, which we have found useful in reasoning about data structure abstractions [32, 33].

7 Conclusions

In this paper we established the expressive power of spatial conjunction by constructing an embedding from first-order logic with spatial conjunction into second-order logic and an embedding from full second order logic into first-order logic with spatial conjunction. These embeddings show that the satisfiability of formulas in first-order logic with spatial conjunction is equivalent to the satisfiability of formulas in second-order logic. This equivalence implies new decidability and undecidability results for extensions of two-variable logic with counting, decidability of (unary-predicate) spatial logic over trees, and the fact that inductive definitions, spatial implication, and a parameterized spatial conjunction are all expressible using first-order logic with spatial conjunction. Finally, our connection opens up the possibility of using second-order logic as a unifying framework for integrating several formalisms for reasoning about dynamic data structures: spatial logic, monadic second-order logic on trees and graphs, and three-valued structures.

Acknowledgements. We thank the participants of the Dagstuhl Seminar 03101 "Reasoning about Shape" for useful discussions on separation logic, shape analysis, and techniques for reasoning about mutable data structures in general. The consequences of the translation from second-order logic to spatial conjunction for two-variable logic with counting were crystallized in discussion with Greta Yorsh.

References

1. A. Aiken, J. S. Foster, J. Kodumal, and T. Terauchi. Checking and inferring local non-aliasing. In *PLDI 2003*, 2003.
2. J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Springer, 1985.
3. M. Benedikt, T. Reps, and M. Sagiv. A decidable logic for linked data structures. In *Proc. 8th ESOP*, 1999.
4. J. Berdine, C. Calcagno, and P. O'Hearn. A decidable fragment of separation logic. In *FSTTCS*, 2004.
5. L. Birkedal, N. Torp-Smith, and J. C. Reynolds. Local reasoning about a copying garbage collector. In *31st ACM POPL*, pages 220–231. ACM Press, 2004.
6. B. Blanchet. Escape analysis: correctness proof, implementation and experimental results. In *Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 25–37. ACM Press, 1998.
7. B. Blanchet. Escape Analysis for Java(TM). Theory and Practice. *ACM Transactions on Programming Languages and Systems*, 2003. To appear.
8. E. Börger and R. Stärk. *Abstract State Machines*. Springer-Verlag, 2003.
9. R. Burstall. Some techniques for proving correctness of programs which alter data structures. *Machine Intelligence*, 7, 1972.
10. L. Caires. Behavioral and spatial observations in a logic for the pi-calculus. In *FoSSaCS*, 2004.
11. L. Caires and L. Cardelli. A spatial logic for concurrency (part i). *Information and Computation*, 186(2):194–235, 2003.
12. C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *ACM TLDI'02*, 2002.
13. C. Calcagno, H. Yang, and P. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *FSTTCS*, 2001.
14. L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for quering graphs. Presentation at the 18th Workshop on Mathematical Foundations of Programming Semantics, March 2002.
15. L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *Proc. 29th ICALP*, volume 2380 of *LNCS*, 2002.
16. L. Cardelli and G. Ghelli. A query language based on the ambient logic. In *Proc. 10th ESOP*, volume 2028 of *LNCS*, 2001.
17. L. Cardelli and A. D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *27th ACM POPL*, 2000.
18. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997. release 1999.
19. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. 6th POPL*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.
20. A. Dawar, P. Gardner, and G. Ghelli. Expressiveness and complexity of graph logic. Technical Report 3, Imperial College, Department of Computing Technical Report, 2004.
21. R. DeLine and M. Fähndrich. Enforcing high-level protocols in low-level software. In *Proc. ACM PLDI*, 2001.
22. M. Fahndrich and R. DeLine. Adoption and focus: Practical linear types for imperative programming. In *Proc. ACM PLDI*, 2002.
23. E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science LICS '97, Warschau*, 1997.
24. W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.

25. S. Ishtiaq and P. W. O'Hearn. BI as an assertion language for mutable data structures. In *Proc. 28th ACM POPL*, 2001.
26. J. Kodumal and A. Aiken. The set constraint/CFL reachability connection in practice. In *Proc. ACM PLDI*, June 2004.
27. V. Kuncak, P. Lam, and M. Rinard. Role analysis. In *Proc. 29th POPL*, 2002.
28. V. Kuncak and M. Rinard. On role logic. Technical Report 925, MIT CSAIL, 2003.
29. V. Kuncak and M. Rinard. The first-order theory of sets with cardinality constraints is decidable. Technical Report 958, MIT CSAIL, July 2004.
30. V. Kuncak and M. Rinard. Generalized records and spatial conjunction in role logic. In *11th Annual International Static Analysis Symposium (SAS'04)*, Verona, Italy, August 26–28 2004.
31. V. Kuncak and M. Rinard. On generalized records and spatial conjunction in role logic. Technical Report 942, MIT CSAIL, April 2004.
32. P. Lam, V. Kuncak, and M. Rinard. On our experience with modular pluggable analyses. Technical Report 965, MIT CSAIL, September 2004.
33. P. Lam, V. Kuncak, K. Zee, and M. Rinard. The Hob project web page. http://catfish.csail.mit.edu/∼plam/hob/, 2004.
34. T. Lev-Ami, T. Reps, M. Sagiv, and R. Wilhelm. Putting static analysis to work for verification: A case study. In *International Symposium on Software Testing and Analysis*, 2000.
35. É. Lozes and L. Caires. Elimination of quantifiers and undecidability in spatial logics for concurrency. In *CONCUR*, 2004.
36. R. Manevich, G. Ramalingam, J. Field, D. Goyal, and M. Sagiv. Compactly representing first-order structures for static analysis. In *Proc. 9th International Static Analysis Symposium*, pages 196–212, 2002.
37. R. Manevich, M. Sagiv, G. Ramalingam, and J. Field. Partially disjunctive heap abstraction. In R. Giacobazzi, editor, *Proceedings of the 11th International Symposium, SAS 2004*, volume 3148 of *Lecture Notes in Computer Science*, pages 265–279. Springer, aug 2004. Available at http://www.cs.tau.ac.il/∼rumster/sas04.pdf.
38. D. Melski and T. Reps. Interconvertibility of a class of set constraints and context-free language reachability. *TCS*, 248:29–98, November 2000.
39. E. Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, London, 4th edition, 1997.
40. A. Møller and M. I. Schwartzbach. The Pointer Assertion Logic Engine. In *Proc. ACM PLDI*, 2001.
41. F. Nielson, H. R. Nielson, and M. Sagiv. Kleene's logic with equality. *Inf. Process. Lett.*, 80(3):131–137, 2001.
42. P. O'Hearn and D. Pym. The logic of bunched implications. *Bulleting of Symbolic Logic*, 5(2):215–244, 1999.
43. P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *Proc. CSL, Paris 2001*, volume 2142 of *LNCS*, 2001.
44. J. C. Reynolds. Intuitionistic reasoning about shared mutable data structure. In *Proceedings of the Symposium in Celebration of the Work of C.A.R. Hoare*, 2000.
45. J. C. Reynolds. Separation logic: a logic for shared mutable data structures. In *17th LICS*, pages 55–74, 2002.
46. M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. In *Proc. 26th ACM POPL*, 1999.
47. M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM TOPLAS*, 24(3):217–298, 2002.
48. T. Skolem. Untersuchungen über die Axiome des Klassenkalküls and über "Produktations- und Summationsprobleme", welche gewisse Klassen von Aussagen betreffen. Skrifter utgit av Vidnskapsselskapet i Kristiania, I. klasse, no. 3, Oslo, 1919.

49. F. Smith, D. Walker, and G. Morrisett. Alias types. In *Proc. 9th ESOP*, Berlin, Germany, Mar. 2000.
50. D. Walker and G. Morrisett. Alias types for recursive data structures. In *Workshop on Types in Compilation*, 2000.
51. R. Wilhelm, M. Sagiv, and T. Reps. Shape analysis. In *Proc. 9th International Conference on Compiler Construction*, Berlin, Germany, 2000. Springer-Verlag.
52. E. Yahav. Verifying safety properties of concurrent Java programs using 3-valued logic. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 27–40. ACM Press, 2001.
53. E. Yahav, T. Reps, M. Sagiv, and R. Wilhelm. Verifying temporal heap properties specified via evolution logic. In *Proc. 12th ESOP*, 2003.