Context Logic as Modal Logic: Completeness and Parametric Inexpressivity

Cristiano Calcagno Philippa Gardner Uri Zarfaty

Department of Computing, Imperial College London

{ccris,pg,udz}@doc.ic.ac.uk

Abstract

Separation Logic, Ambient Logic and Context Logic are based on a similar style of reasoning about structured data. They each consist of a structural (separating) composition for reasoning about disjoint subdata, and corresponding structural adjoint(s) for reasoning hypothetically about data. We show how to interpret these structural connectives as modalities in Modal Logic and prove completeness results. The structural connectives are essential for describing properties of the underlying data, such as weakest preconditions for Hoare reasoning for Separation and Context Logic, and security properties for Ambient Logic. In fact, we introduced Context Logic to reason about tree update, precisely because the structural connectives of the Ambient Logic did not have enough expressive power. Despite these connectives being essential, first Lozes then Dawar, Gardner and Ghelli proved elimination results for Separation Logic and Ambient Logic (without quantifiers). In this paper, we solve this apparent contradiction. We study parametric inexpressivity results, which demonstrate that the structural connectives are indeed fundamental for this style of reasoning.

Categories and Subject Descriptors D.2.4 [*Software/Program verification*]: Correctness proofs, Formal methods, Validation

General Terms Languages, Theory, Verification

Keywords Logic, Expressivity, Structured Data, Contexts

1. Introduction

Separation Logic (SL) and Ambient Logic (AL) are related logics for reasoning about heaps and trees respectively. O'Hearn, Reynolds and Yang introduced SL [8, 11, 13] to develop local Hoare reasoning about heap update, based on the general theory of Bunched Logic (BL) due to O'Hearn and Pym [10]. Cardelli and Gordon independently introduced AL [5] for reasoning about static trees. AL has been used to reason about security properties of firewalls and structural properties of XML [6]. We have integrated these two lines of research. In [3], we showed that it is not possible to use AL to reason about tree update (XML update). Instead, we introduced the general theory of Context Logic (CL) for reasoning about structured data, which generalises BL. We demonstrated that the application of CL-reasoning to trees can be used as a basis for local Hoare reasoning about tree update, whilst the application of CL-reasoning to heaps exactly corresponds to SL-reasoning.

These logics are based on a similar style of reasoning about structured data. They each extend propositional connectives with a structural (separating) composition for reasoning about disjoint subdata, and the corresponding structural adjoint(s).[1] We show how to interpret the structural connectives of BL and CL as modalities in modal logic (ML). We present additional axioms for these modalities to give a precise correspondence between the original presentation of BL and CL, and their ML-interpretations. These axioms are well-behaved, in that they satisfy the conditions necessary for us to apply a general completeness result about ML (Sahlqvist's theorem). We thus prove that the CL-proof theory is sound and complete with respect to the set of CL-models (and analogously for BL). This work follows previous unpublished work by Calcagno and Yang, who proved completeness for CL from first principles.

The structural connectives are essential for modular reasoning about programs, and for describing weakest preconditions and safety properties. However, recent expressivity results for SL and AL due to Lozes [9] appear to contradict this fundamental claim. Lozes concentrates on expressivity for closed formulae, determining whether an arbitrary closed formula specifying a *set* of data in one logic can be expressed by a formula in the other logic specifying the same set of data. For example, Lozes has shown that SL and Propositional Logic (PL) with simple atomic heap formulae are equally expressive using this definition of expressivity. However, our experience says that SL is more expressive than PL, since for example we can reason directly about disjointness and dynamic update of linked lists. We solve this apparent mismatch between the theoretical results and our practical experience by proving inexpressivity results for stronger definitions of expressivity.

SL forms the basis of local Hoare reasoning about heap update. An important part of the reasoning is to be able to express the weakest preconditions, which provide completeness for straight-line code and have a key role in some verification tools (to verify a Hoare triple, first find the weakest precondition of a given postcondition and then prove that the given precondition implies the weakest precondition). Our results show that the weakest preconditions cannot be expressed in PL for heaps. To illustrate this, consider the weakest precondition of allocation $(n \mapsto 0) \multimap p$, specifying that whenever a cell with address n and value 0 is added to the given heap then the resulting heap satisfies post-condition p. Lozes' result says that, for every interpretation of p as a set of data determined by a closed formula, there is a corresponding PL-formula. However, these PL-formulae are highly non-uniform with respect to the post-condition p, and Lozes' result says nothing about whether the weakest precondition itself can be specified in PL. We show that it is not possible, by studying expressivity for open formulae con-

[1] In this paper, we do not consider quantification.

POPL'07 January 17–19, 2007, Nice, France.
Copyright © 2007 ACM 1-59593-575-4/07/0001. . . $5.00

taining propositional variables. This notion of expressivity determines whether an arbitrary open formula in one logic, specifying a *function* from sets to data to sets of data, can be expressed by a formula in the other logic. There are two choices for the domain of this function, either as sets of data specified by closed formulae or as arbitrary sets of data. The first option is enough to determine whether the weakest precondition can be specified in PL. It does not however allow for natural extensions to SL, such as the addition of inductive predicates. We therefore study the second option. We call this type of expressivity *parametric expressivity*, and show that SL is parametrically more expressive than PL for heaps by demonstrating that $(n{\mapsto}0) \mathrel{-\!\!*} p$ cannot be expressed in PL.

Although the SL-adjoint $\mathrel{-\!\!*}$ is important for the weakest preconditions and has a key role in some proofs [12], it is not typically used for specifying safety properties. For example, it plays no role in the verification tool Smallfoot [1], which combines inductive predicates with a cut-down decidable fragment of SL (with quantification). A more fundamental SL-formula is $p * q$ specifying that the heap can be split into two disjoint parts, one satisfying p and the other q. Lozes' results imply that, for every interpretation of p and q as sets of data corresponding to closed formulae, there is a corresponding PL-formula. Again, the PL-formulae are highly non-uniform. We show that it is not possible to express parametrically this formula in PL. However, with the Smallfoot application in mind, it is perhaps more interesting to determine a specific inexpressivity result, that it is not possible to express $p * q$ in PL with the interpretation of p as $list(3)$, denoting the existence of a 0-terminated linked list starting at address 3, and q as $list(4)$. To do this, we study the notion of *strong expressivity* which states that, for a *specific* interpretation of the propositional variables as arbitrary sets of data, every formula with propositional variables in one logic can be expressed in the other logic.

To prove our strong inexpressivity results, we use a standard bisimulation technique from ML. For example, consider the heaps

$$\begin{aligned} h_1 &= [3 \mapsto n', 4 \mapsto n'', n' \mapsto 0, n'' \mapsto 0] \\ h_2 &= [3 \mapsto n', 4 \mapsto n', n' \mapsto 0, n'' \mapsto 0]. \end{aligned}$$

These heaps are distinguished by SL-formula $p * q$, with the interpretation of p and q as the lists $list(3)$ and $list(4)$ respectively, since h_1 can be split into the appropriate disjoint lists whereas h_2 cannot due to the sharing at address n'. Our proof shows that there is a PL-bisimulation relation relating h_1 and h_2. Bisimulation has the well-known property that it is contained in logical equivalence. Thus, the heaps h_1 and h_2 are indistinguishable using PL.

Our original motivation for studying parametric inexpressivity results came from studying CL for trees and AL. We introduced CL to provide local Hoare reasoning about tree update [3], demonstrating that it was not possible to base our Hoare reasoning on AL since it had a missing adjoint. Whilst we believe that our argument was convincing, it was an argument given by example rather than by a formal inexpressivity result. Lozes' expressivity results, focussing on the closed formulae, show that the argument is subtle since AL (without quantifiers) is as expressive as the logic without the structural adjoints [9, 6]. We prove that CL for trees is parametrically more expressive than AL. Unlike the results for SL, we do not know how to prove this directly. Instead, we prove a strong inexpressivity result using an analogous proof method to that outlined above. Since strong inexpressivity implies parametric inexpressivity, we have the result. In addition, we prove that CL for trees minus the extra adjoint is parametrically as expressive as AL, thus showing that the additional strength of the CL-reasoning does indeed come from this additional adjoint. We also prove similar results for CL for sequences and a variation of BL applied to sequences ($*$ is non-commutative). Sequences provide the simplest example of the differences between CL- and BL-reasoning. Again, we prove our parametric inexpressivity result via strong inexpressivity.

2. Context Logic and Bunched Logic

We review the general theory of CL and BL.

2.1 Context Logic

We introduced CL to reason about data update [3]. Local data update typically identifies the portion of data to be replaced, removes it, and inserts the new data *in the same place*. With CL, we reason about both data and this place of insertion (contexts). CL consists of data formula denoted by P, and context formulae denoted by K. In each case, these include standard formulae from propositional logic, and less familiar structural formulae for directly analysing the data and context structure.

Definition 1 (CL-Formulae). *The set of CL-formulae consists of disjoint sets of* data formulae $\mathcal{P}$ *and* context formulae $\mathcal{K}$*, constructed from a set of propositional variables* $\mathcal{V} = \mathcal{V}_{\mathcal{P}} \cup \mathcal{V}_{\mathcal{K}}$ *where* $\mathcal{V}_{\mathcal{P}}$ *and* $\mathcal{V}_{\mathcal{K}}$ *are disjoint, countably infinite sets of propositional data variables and context variables respectively. The formulae are given by the grammars:*

data formulae

$$\begin{array}{rlll} P ::= & K(P) \mid K \lhd P & & \textit{structural formulae} \\ & P \vee P \mid \neg P \mid \textit{false} & & \textit{additive formulae} \\ & p, p_1, p_2, \ldots & & \textit{prop vars in } \mathcal{V}_{\mathcal{P}} \end{array}$$

context formulae

$$\begin{array}{rlll} K ::= & I \mid P \rhd P & & \textit{structural formulae} \\ & K \vee K \mid \neg K \mid \textit{False} & & \textit{additive formulae} \\ & k, k_1, k_2, \ldots & & \textit{prop vars in } \mathcal{V}_{\mathcal{K}} \end{array}$$

The key formulae are the structural formulae $K(P)$, $K \lhd P$, $P_1 \rhd P_2$ and I. The *application formula* $K(P)$ specifies that the given data element can be split into a context satisfying K applied to data satisfying P. For example, if we define the context formula True $\triangleq$ $\neg$False, then the formula True(P) states that some subdata satisfies property P. The next two formulae are both (right) adjoints of application. The formula $K \lhd P$ is satisfied by the given data if, *whenever* we insert the data into a context satisfying K, then the result satisfies P. For example, the formula (True $\lhd P$) states that, when the data is put in any context, the resulting data satisfies property P. The analogous connectives for AL have been used to describe security properties of trees (ambients). Meanwhile, $P_1 \rhd P_2$ is a statement on contexts. It is satisfied by a given context if, *whenever* we insert in the context some data satisfying P_1, then the result satisfies P_2. Given the derived data formula true $\triangleq$ $\neg$false, the context formula (true$\rhd P_2$) states that, regardless of what data is put in the context hole, the resulting data satisfies property P_2. This adjoint is essential for expressing weakest preconditions for update commands, as we demonstrated in [3], and has no counterpart in AL. The context formula I specifies the empty context.

Definition 2 (CL-Model). *A* CL-model $\mathcal{M}$ *is a tuple* $(\mathcal{D}, \mathcal{C}, ap, \mathbf{I})$ *such that*

1. $\mathcal{D}$ *and* $\mathcal{C}$ *are sets;*
2. $ap \subseteq (\mathcal{C} \times \mathcal{D}) \times \mathcal{D}$ *is a relation, called* application*: we use the notation* $ap(c, d_1) = d_2$ *for* $((c, d_1), d_2) \in ap$*;*
3. $\mathbf{I} \subseteq \mathcal{C}$ *acts as a left identity to* ap*: that is,*
   - $\forall d \in \mathcal{D}, \exists i \in \mathbf{I}, d' \in \mathcal{D}.\ ap(i, d) = d'$*;*
   - $\forall d, d' \in \mathcal{D}, \forall i \in \mathbf{I}.\ ap(i, d) = d'$ *implies* $d = d'$.

We often call $\mathcal{D}$ the *data set* and $\mathcal{C}$ the *context set*, because of the form of our motivating examples. Of course, there are models which do not fit this structured data intuition. We prove completeness for these CL-models (theorem 23) and the analogous BL-models (section 3.2).

Example 3.

- $Mon_{\mathcal{D}} = (\mathcal{D}, \mathcal{D}, \cdot, \{e\})$ *where $\mathcal{D}$ is a partial monoid with binary operation $\cdot : (\mathcal{D} \times \mathcal{D}) \rightharpoonup \mathcal{D}$ and unit $e \in \mathcal{D}$.*
- $Heap$ *is an example of $Mon_{\mathcal{D}}$ where $\mathcal{D} = \mathbb{N}^+ \rightharpoonup_{fin} \mathbb{N}$ is the set of finite partial functions denoting the heaps and e denotes the empty heap. The domain $\mathbb{N}^+ = \mathbb{N} - \{0\}$ does not include 0 as it is reserved for the null location. Given heaps h, h', the heap composition $h \cdot h'$ is function union which is only defined when $dom(h) \cap dom(h') = \emptyset$.*
- $Term_{\Sigma} = (\mathcal{D}_{\Sigma}, \mathcal{C}_{\Sigma}, ap, \{\_\})$ *where $\mathcal{D}_{\Sigma}$ is the data set of terms constructed from the n-ary function symbols in signature Σ, $\mathcal{C}_{\Sigma}$ is the corresponding set of contexts, ap denotes the standard application of contexts to terms, and $\_$ denotes the empty context.*
- $Seq_A = (\mathcal{D}_A, \mathcal{C}_A, ap, \{\_\})$ *where $\mathcal{D}_A$ is the set of sequences constructed from the elements in alphabet A, $\mathcal{C}_A$ is the corresponding set of contexts, and ap and $\_$ are as for $Term_{\Sigma}$.*
- $Tree_A$ *is an example of $Term_{\Sigma}$ with an additional equality relation on terms. The terms are generated by the signature Σ constructed from the sets $\Sigma_0 = \{0\}$, $\Sigma_1 = A$ and $\Sigma_2 = \{|\}$, where Σ_i denotes the function symbols of arity i. We use the notation $t \mid t'$ for $|(t, t')$ and $a[t]$ for $a(t)$. Terms are considered modulo an equality relation generated by the axioms $0 \mid t \equiv t$, $t \mid t' \equiv t' \mid t$, $(t \mid t') \mid t'' \equiv t \mid (t' \mid t'')$, and closed by the obvious structural rules for the function symbols.*
- $Rel_{\mathcal{D}} = (\mathcal{D}, \mathcal{P}(\mathcal{D} \times \mathcal{D}), ap, \{i\})$ *where $\mathcal{D}$ is an arbitrary set, $\mathcal{P}(\mathcal{D} \times \mathcal{D})$ denotes the set of binary relations on $\mathcal{D}$, ap is relational application, and i is the identity relation.*

Definition 4 (CL-Satisfaction Relation)**.** *Given a CL-model $\mathcal{M} = (\mathcal{D}, \mathcal{C}, ap, \mathbf{I})$, the CL-satisfaction relation $\vDash_{CL}$ consists of two relations $\sigma, \mathcal{M}, d \vDash_{\mathcal{P}} P$ and $\sigma, \mathcal{M}, c \vDash_{\mathcal{K}} K$ where $d \in \mathcal{D}$, $c \in \mathcal{C}$ and interpretation function $\sigma : \mathcal{V} \rightarrow \mathcal{P}(\mathcal{D} \cup \mathcal{K})$ maps data propositional variables to sets of data, and context propositional variables to sets of contexts. The two relations are defined by induction on the structure of the formulae: the cases for the propositional variables and the boolean additive connectives are standard; the cases for the structural connectives are*

$$\begin{array}{lll}
\sigma, \mathcal{M}, d \vDash_{\mathcal{P}} K(P) & \textit{iff} & \exists c \in \mathcal{C}, d' \in \mathcal{D}. \\
\quad ap(c, d') = d \wedge \sigma, \mathcal{M}, c \vDash_{\mathcal{K}} K \wedge \sigma, \mathcal{M}, d' \vDash_{\mathcal{P}} P & & \\
\sigma, \mathcal{M}, d \vDash_{\mathcal{P}} K \lhd P & \textit{iff} & \forall c \in \mathcal{C}, d' \in \mathcal{D}. \\
\quad \sigma, \mathcal{M}, c \vDash_{\mathcal{K}} K \wedge\ ap(c, d) = d' \Rightarrow \sigma, \mathcal{M}, d' \vDash_{\mathcal{P}} P & & \\
\sigma, \mathcal{M}, c \vDash_{\mathcal{K}} I & \textit{iff} & c \in \mathbf{I} \\
\sigma, \mathcal{M}, c \vDash_{\mathcal{K}} P_1 \rhd P_2 & \textit{iff} & \forall\, d, d' \in \mathcal{D}. \\
\quad \sigma, \mathcal{M}, d \vDash_{\mathcal{P}} P_1 \wedge\ ap(c, d) = d' \Rightarrow \sigma, \mathcal{M}, d' \vDash_{\mathcal{P}} P_2 & &
\end{array}$$

We sometimes omit the subscripts $\mathcal{P}$ and $\mathcal{K}$.

In section 4, we study applications of CL to heaps, sequences and trees, which extend CL with simple atomic formulae specific to these models. Here, we use the Seq_A model and the additional zero formula 0, denoting the empty sequence, to illustrate our CL-reasoning. Consider the derived formula $1 \triangleq \neg 0 \wedge \neg(\neg I)(\neg 0)$, which states that the sequence only contains one element: that is, it is non-empty and cannot be split into a non-empty context and non-empty data. Now consider the judgement

$$\sigma, Seq_A, s \vDash (0 \rhd p)(1),$$

where $\sigma(p)$ denotes the set of sequences with equal elements and s denotes a sequence. This judgement only holds if s is non-empty and all the elements in s are equal except possibly one: for example, it holds when s is $b \cdot a \cdot b$, but not when s is $b \cdot a \cdot c$.

We use the standard derived classical formulae for both data and context formulae: true, $P \wedge P$ and $P \Rightarrow P$; similarly for contexts, writing True for the context formula that is always satisfied. We shall also use the following derived formulae:

- $\diamond P \triangleq \text{True}(P)$ specifies that somewhere property P holds;
- $P_1 \blacktriangleright P_2 \triangleq \neg(P_1 \rhd \neg P_2)$ specifies that *there exists* some data element satisfying property P_1 such that, when it is put in the hole of the given context, the resulting data satisfies P_2;
- $K \blacktriangleleft P_2 \triangleq \neg(K \lhd \neg P_2)$ specifies that *there exists* a context satisfying property K such that, when the given data element is put in the hole, the resulting data satisfies P_2.

We give a Hilbert-style proof theory, following the style for BL in [10]. The axioms and rules for the structural operators state that $K \lhd P_2$ and $P_1 \rhd P_2$ are right adjoints of $K(P_1)$, and I is the identity of application.

Definition 5 (CL-Proof Theory)**.** *The Hilbert-style CL-proof theory consists of the standard axioms and rules for the boolean additive connectives, and the following axioms and rules for the structural connectives:*

$$P \dashv\vdash_{\mathcal{P}} I(P) \qquad \frac{K_1 \vdash_{\mathcal{K}} K_2 \qquad P_1 \vdash_{\mathcal{P}} P_2}{K_1(P_1) \vdash_{\mathcal{P}} K_2(P_2)}$$

$$\frac{K(P_1) \vdash_{\mathcal{P}} P_2}{K \vdash_{\mathcal{K}} P_1 \rhd P_2} \qquad \frac{K \vdash_{\mathcal{K}} P_1 \rhd P_2 \qquad P \vdash_{\mathcal{P}} P_1}{K(P) \vdash_{\mathcal{P}} P_2}$$

$$\frac{K(P_1) \vdash_{\mathcal{P}} P_2}{P_1 \vdash_{\mathcal{P}} K \lhd P_2} \qquad \frac{P_1 \vdash_{\mathcal{P}} K \lhd P_2 \qquad K_1 \vdash_{\mathcal{K}} K}{K_1(P_1) \vdash_{\mathcal{P}} P_2}$$

We sometimes omit the subscripts in $\vdash_{\mathcal{P}}$ and $\vdash_{\mathcal{K}}$, and sometimes write $\vdash_{CL}$ to refer explicitly to this CL-proof theory.

The proof theory given here emphasises the right adjoint properties of $\lhd$ and $\rhd$. In the next section, we show that this proof theory is equivalent to the standard ML-proof theory plus an additional set of axioms specific to CL. This alternative formulation emphasises the derived connectives $\blacktriangleright$ and $\blacktriangleleft$ instead.

2.2 Bunched Logic

We also present (a variant of) BL [10], its models and satisfaction relation, and compare it to CL. We use the notation $\circ$ and $-\!\!\circ$, instead of the standard $*$ and $-\!\!*$ for the multiplicative conjunction and its adjoint. Our variation of standard BL does not require $\circ$ to be commutative, since one of our key example models is sequences where $\circ$ denotes concatenation.

Definition 6 (BL-Formulae)**.** *The set of BL formulae $\mathcal{P}$ is constructed from a countably infinite set of propositional variables $\mathcal{V}_{\mathcal{P}}$, and defined by the grammar:*

$$\begin{array}{lll}
P ::= & 0 \mid P \circ P \mid P \circ\!\!- P \mid P -\!\!\circ P & \textit{structural formulae} \\
 & P \vee P \mid \neg P \mid \textit{false} & \textit{additive formulae} \\
 & p, p_1, p_2, \ldots & \textit{prop vars in } \mathcal{V}_{\mathcal{P}}
\end{array}$$

The key formulae are the structural formulae 0, $P_1 \circ P_2$, $P_1 \circ\!\!- P_2$ and $P_1 -\!\!\circ P_2$. The *zero* formula 0 specifies empty data. The *composition* formula splits the given data into two parts, the first satisfying P_1 and the second P_2. For example, the formula $\neg 0 \circ \neg 0$ specifies that the given data can be split into two disjoint, non-empty parts. Unlike the original BL, we have two right adjoints, due to $\circ$ being non-commutative: $P_1 \circ\!\!- P_2$ specifies that, *whenever* some data satisfying P_1 is placed to the left of the given data, then the result satisfies P_2; the other adjoint $P_1 \multimap P_2$ places data to the right. This distinction has no effect in the heap model, but is important in the sequence model. As in CL, we define the negation duals of the adjoints as $P_1 -\!\!\bullet P_2 \triangleq \neg(P_1 \multimap \neg P_2)$ and $P_1 \bullet\!\!- P_2 \triangleq \neg(P_1 \circ\!\!- \neg P_2)$.

Definition 7 (BL-Model). *A* BL-model $\mathcal{M}$ *is a tuple* $(\mathcal{D}, \cdot, \mathbf{e})$ *such that*

1. $\mathcal{D}$ *is a set;*
2. $\cdot \subseteq (\mathcal{D} \times \mathcal{D}) \times \mathcal{D}$ *is an associative relation: we use the notation* $d_1 \cdot d_2 = d_3$ *for* $((d_1, d_2), d_3) \in \cdot$*;*
3. $\mathbf{e} \subseteq \mathcal{D}$ *acts as a left and right identity to* $\cdot$ *: that is,*
   - $\forall d \in \mathcal{D}, \exists e \in \mathbf{e}, d' \in \mathcal{D}.\ e \cdot d = d'$
   - $\forall d \in \mathcal{D}, \exists e \in \mathbf{e}, d' \in \mathcal{D}.\ d \cdot e = d'$
   - $\forall d, d' \in \mathcal{D}, \forall e \in \mathbf{e}.\ e \cdot d = d'$ *or* $d \cdot e = d'$ *implies* $d = d'$.

Any BL-model $\mathcal{M} = (\mathcal{D}, \cdot, \mathbf{e})$ can be transformed into a CL-model $\mathcal{M}_{BL} = (\mathcal{D}, \mathcal{D}, \cdot, \mathbf{e})$. We highlight specific BL-models for heaps, sequences and trees, since we will use them throughout this paper.

Example 8.

- $Heap = (\mathcal{D}, \cdot, \{e\})$ *where* $\mathcal{D}$, $\cdot$ *and* e *are as in Example 3.*
- $Seq_A = (\mathcal{D}_A, \cdot, \{0\})$ *where* $\mathcal{D}_A$ *is the set of sequences constructed from the elements in set* A, $\cdot$ *is sequence concatenation, and* 0 *is the empty sequence.*
- $Tree_A = (\mathcal{D}_A, |, \{0\})$ *where* $\mathcal{D}_A$ *is the set of trees in Example 3,* $|$ *is horizontal tree composition, and* 0 *is the empty tree.*

Contrast these BL-models with the analogous CL-models given in Example 3, which also emphasise the context structure. The heap model is essentially the same, with the context set being the same as the data set. However, the sequence and tree models are different, since the context set is more complex than the data set.

Definition 9 (BL-Satisfaction Relation). *Given a BL-model* $\mathcal{M} = (\mathcal{D}, \cdot, \mathbf{e})$*, the BL-satisfaction relation is of the form* $\sigma, \mathcal{M}, d \vDash_{BL} P$ *where* $d \in \mathcal{D}$*, and* $\sigma : \mathcal{V}_\mathcal{P} \to \mathcal{P}(\mathcal{D})$*. As before, it is defined by induction on the structure of formulae. We only give the cases for the structural connectives:*

$$\begin{array}{lll}
\sigma, \mathcal{M}, d \vDash_{BL} P_1 \circ P_2 & \textit{iff} & \exists d_1, d_2 \in \mathcal{D}. \\
\quad d_1 \cdot d_2 = d \wedge \sigma, \mathcal{M}, d_1 \vDash_{BL} P_1 \wedge \sigma, \mathcal{M}, d_2 \vDash_{BL} P_2 & & \\
\sigma, \mathcal{M}, d \vDash_{BL} 0 & \textit{iff} & d \in \mathbf{e} \\
\sigma, \mathcal{M}, d \vDash_{BL} P_1 \multimap P_2 & \textit{iff} & \forall d_1, d_2 \in \mathcal{D}. \\
\quad \sigma, \mathcal{M}, d_1 \vDash_{BL} P_1 \wedge\ d \cdot d_1 = d_2 \Rightarrow \sigma, \mathcal{M}, d_2 \vDash_{BL} P_2 & & \\
\sigma, \mathcal{M}, d \vDash_{BL} P_1 \circ\!\!- P_2 & \textit{iff} & \forall d_1, d_2 \in \mathcal{D}. \\
\quad \sigma, \mathcal{M}, d_1 \vDash_{BL} P_1 \wedge\ d_1 \cdot d = d_2 \Rightarrow \sigma, \mathcal{M}, d_2 \vDash_{BL} P_2 & &
\end{array}$$

Consider the BL-model Seq_A, the derived BL-formula $1 \triangleq \neg 0 \wedge \neg(\neg 0 \circ \neg 0)$ specifying sequences of length one, and the BL-relation

$$\sigma, Seq_A, s \vDash_{BL} (0 \multimap p) \circ 1,$$

where $\sigma(p)$ again denotes the set of sequences with equal elements. This relation only holds if s is a non-empty sequence consisting of equal elements except the last one which can be anything: for example, the relation holds when s is $b \cdot b \cdot a$, but does not hold when s is $b \cdot a \cdot b$ and $b \cdot a \cdot c$. This simple example illustrates the difference between BL- and CL-reasoning: BL-reasoning analyses the ends of the sequences, whereas CL-reasoning also analyses the middle. However, when the CL-model arises from a BL-model, there is a strong relationship between BL-reasoning and and CL-reasoning. We give this correspondence explicitly for heaps in Proposition 26.

The Hilbert-style BL-proof theory consists of analogous rules to those given for the CL-proof theory (Definition 5), with an additional axiom for the associativity of $\circ$.

3. Connection to Modal Logic

We recall some general theory of ML, and show how CL and BL fit within this formalism. We prove completeness results relating the CL- and BL-proof theories with their respective models, by appealing to a general theorem of ML due to Sahlqvist.

Definition 10 (ML-Signature). *A ML-signature is a triple* $\Sigma = (\mathcal{S}, O, \rho : O \to \mathcal{T})$*, where* $\mathcal{S}$ *is a set of sorts ranged over by* S*,* O *is a set of modalities ranged over by* Δ*,* $\mathcal{T}$ *is a set of types of the form* $(S_1, \ldots, S_n) \to S$ *for* $S_i, S \in \mathcal{S}$*, and* ρ *is a function. We write* $\Delta : T$ *when* $\rho(\Delta) = T$*.*

Definition 11 (ML-Model[2]). *Given a ML-signature* $\Sigma = (\mathcal{S}, O, \rho)$*, a ML-model* $\mathcal{M}_\Sigma$ *generated from* Σ *consists of a set* $\mathcal{M}_S$ *for each* $S \in \mathcal{S}$*, and an interpretation* $\mathcal{M}_\Delta \subseteq (\mathcal{M}_{S_1} \times \cdots \times \mathcal{M}_{S_n}) \times \mathcal{M}_S$ *for each modality* Δ *of type* $(S_1, \ldots, S_n) \to S$*.*

Example 12 (ML-signature for CL). *Consider the ML-signature* Σ_{CL} *consisting of two sorts D, C with modalities* $ap : (C, D) \to D$*,* $I : (\,) \to C$*,* $\blacktriangleright : (D, D) \to C$ *and* $\blacktriangleleft : (C, D) \to D$*. Given a CL-model* $\mathcal{M} = (\mathcal{D}, \mathcal{C}, ap, \mathbf{I})$*, we can view it as a ML-model* $\mathcal{M}_{\Sigma_{CL}}$*, where* $\mathcal{M}_D = \mathcal{D}$*,* $\mathcal{M}_C = \mathcal{C}$*, the interpretations* $\mathcal{M}_{ap}$ *and* $\mathcal{M}_I$ *are inherited from the CL-model, and* $\mathcal{M}_\blacktriangleright$ *and* $\mathcal{M}_\blacktriangleleft$ *are given by* $(c, d, d') \in \mathcal{M}_{ap}$ *iff* $(d, d', c) \in \mathcal{M}_\blacktriangleright$ *iff* $(c, d', d) \in \mathcal{M}_\blacktriangleleft$*. Hence, every CL-model can be interpreted as a ML-model. Notice that not all ML-models over signature* Σ_{CL} *are CL-models, since the* I *modality need not have any relationship to the* ap *modality.*

Definition 13 (ML-Formulae). *Given ML-signature* $\Sigma = (\mathcal{S}, O, \rho)$ *and disjoint, countably infinite sets* $\mathcal{V}_S$ *of propositional variables for each sort* S*, the set* $\mathcal{P}_\Sigma$ *of ML-formulae over* Σ *is given by*

$$P ::= p_S \mid P_1 \vee P_2 \mid \neg P \mid \mathit{false}_S \mid \Delta(P_1, \ldots, P_n)$$

where $p_S \in \mathcal{V}_S$ *and, for each* $\Delta : (S_1, \ldots, S_n) \to S$*, the formula* $\Delta(P_1, \ldots, P_n)$ *has sort* S *provided the* P_i *have sort* S_i*. We write* $\mathcal{P}_{\Sigma_S}$ *for the set of formulae of sort* S *generated from signature* Σ*.*

Definition 14 (ML-Satisfaction Relation). *Given ML-signature* $\Sigma = (\mathcal{S}, O, \rho)$ *and ML-model* $\mathcal{M}_\Sigma$*, the ML-satisfaction relation* $\vDash_{ML}$ *consists of relations of the form* $\sigma, \mathcal{M}, m \vDash_S P$ *for each sort* $S \in \mathcal{S}$*, where* $m \in \mathcal{M}_S$*, formula* P *has sort* S *and, for each propositional variable* p'_S*,* $\sigma(p'_S) \subseteq \mathcal{M}_S$*. It is defined by induction on the structure of ML-formulae, with the modality case given by*

$$\begin{array}{l}
\sigma, \mathcal{M}, m \vDash_S \Delta(P_1, \ldots, P_n) \Leftrightarrow \forall i \in \{1, \ldots, n\}. \exists m_i. \\
\qquad\qquad \sigma, \mathcal{M}, m_i \vDash_{S_i} P_i \wedge ((m_1, \ldots, m_n), m) \in \mathcal{M}_\Delta.
\end{array}$$

The other cases are evident.

Example 15. *Given CL-model* $\mathcal{M}$ *and corresponding ML-model* $\mathcal{M}_{\Sigma_{CL}}$ *from example 12, the CL- and ML-satisfaction relations are equal in the following sense. Define a translation function* $|\_| : \mathcal{P} \cup \mathcal{K} \to \mathcal{P}_{\Sigma_{CL}}$ *from CL-formulae to ML-formulae over* Σ_{CL}*, by induction on the structure of the CL-formulae such that each case follows the structure of the formulae except that* $|P_1 \lhd P_2| \triangleq \neg(|P_1| \blacktriangleleft \neg|P_2|)$ *and* $|P_1 \rhd P_2| \triangleq \neg(|P_1| \blacktriangleright \neg|P_2|)$*. We have*

$$\begin{array}{lcl}
\sigma, \mathcal{M}, d \vDash_{CL} P & \Leftrightarrow & \sigma, \mathcal{M}_{\Sigma_{CL}}, d \vDash_{ML} |P| \\
\sigma, \mathcal{M}, c \vDash_{CL} K & \Leftrightarrow & \sigma, \mathcal{M}_{\Sigma_{CL}}, c \vDash_{ML} |K|
\end{array}$$

Recall that not all ML-models over signature Σ_{CL} correspond to CL-models. To get a precise correspondence with CL, we will restrict the class of ML-models to those satisfying a certain set of CL-axioms. We first describe the general theory.

Definition 16 (AX-Model). *Given a ML-signature* Σ *and a set of axioms* $AX \subseteq \mathcal{P}_\Sigma$*, an* AX*-model generated from* Σ *is a ML-model* $\mathcal{M}$ *generated from* Σ *which also satisfies* $\sigma, \mathcal{M}, m \vDash_S P$ *for all* $m \in \mathcal{M}_S, P \in AX$ *and* σ*.*

[2] Note that what we call a ML-model is typically called a frame in e.g. [2].

Definition 17 (AX-Proof Theory)**.** *Given a ML-signature Σ and a set of axioms $AX \subseteq \mathcal{P}_\Sigma$, the ML-proof theory* [3] *generated by AX consists of the following axioms and rules:*

$$\frac{\vdash P \Rightarrow Q \quad \vdash P}{\vdash Q} \qquad \frac{P \in AX}{\vdash P} \qquad \frac{P \textit{ tautology}}{\vdash P}$$

$$\frac{\vdash P}{\vdash P[P'/p]} \qquad \frac{\triangle : (S_1, \dots, S_n) \to S}{\vdash \triangle(p_1, \dots, \textit{false}_{S_i}, \dots, p_n) \Leftrightarrow \textit{false}_S}$$

$$\frac{P(\_) = \triangle(p_1, \dots, \_, \dots, p_n)}{\vdash P(p_i \vee p_i') \Leftrightarrow P(p_i) \vee P(p_i')}$$

We sometimes write $\vdash_{AX}$ to emphasise the set AX.

There is a well-known general completeness result for ML due to Sahlqvist, which relates the AX-satisfaction relation and the AX-proof theory as long as the axioms have a certain form. We state the result here, since we use it to show completeness for CL.

Definition 18 (Very Simple Sahlqvist Formulae)**.** *Given ML-signature $\Sigma = (\mathcal{S}, O, \rho)$, a very simple Sahlqvist antecedent A is a formula given by the grammar:*

$$A ::= \textit{true}_S \mid \textit{false}_S \mid p_S \mid A \wedge A \mid \triangle(A_1, \dots, A_n)$$

for $p_S \in \mathcal{V}_S$ and $\triangle : (S_1, \dots, S_n) \to S$. A very simple Sahlqvist formula is an implication of the form $A \Rightarrow P^+$, where P^+ is a positive formula, in that every propositional letter p_S appears under an even number of negations.

Theorem 19 (Sahlqvist (see [2]))**.** *For every axiom set AX consisting of very simple Sahlqvist formulae, the ML-proof theory generated by AX is complete with respect to the class of AX-models.*

3.1 Context Logic as ML

We have shown that a CL-model can be viewed as a ML-model over signature Σ_{CL} (example 12), and that the corresponding satisfaction relations agree (example 15). Now we identify an axiom set AX_{CL} over signature Σ_{CL}, such that the AX_{CL}-models correspond exactly to the CL-models and the proof theories coincide. Since the AX_{CL}-axioms are Sahlqvist axioms, the general completeness result for ML (Theorem 19) implies completeness for CL.

Definition 20 (CL-Axioms)**.** *Given ML-signature Σ_{CL}, the axiom set AX_{CL} over Σ_{CL} consists of the following formulae, where $p, q \in \mathcal{V}_D$ and $k \in \mathcal{V}_C$:*

1. $I(p) \Rightarrow p$
2. $p \Rightarrow I(p)$
3. $q \wedge k(p) \Rightarrow \textit{True}(p \wedge (k \blacktriangleleft q))$
4. $q \wedge k(p) \Rightarrow (k \wedge (p \blacktriangleright q))(\textit{true})$
5. $p \wedge (k \blacktriangleleft q) \Rightarrow \textit{True} \blacktriangleleft (q \wedge k(p))$
6. $k \wedge (p \blacktriangleright q) \Rightarrow \textit{true} \blacktriangleright (q \wedge k(p))$

The axioms in AX_{CL} are very simple Sahlqvist formulae. The first two axioms correspond directly to the identity axiom of CL. The other axioms capture the relationship between $\mathcal{M}_{ap}$, $\mathcal{M}_\blacktriangleleft$, and $\mathcal{M}_\blacktriangleright$, which simply permutes elements (Example 12). For example, the third axiom species that, if the given data satisfies q and can be split into a context satisfying k and subdata satisfying p, then there exists subdata satisfying p and $k \blacktriangleleft q$ (think of the same subdata). This axiom shifts the emphasis from the given data to the subdata. The fifth axiom is a sort of converse. It states that, if the given data satisfies p and $k \blacktriangleleft q$, then it is possible to enclose it in a context (actually one satisfying k), such that q and $k(p)$ are satisfied. The third and fifth axiom together describe the exact connection between $\mathcal{M}_{ap}$ and $\mathcal{M}_\blacktriangleleft$. Similarly, the fourth and sixth axiom describe the exact connection between $\mathcal{M}_{ap}$ and $\mathcal{M}_\blacktriangleright$.

[3] This is called the normal modal proof theory in [2].

We have already illustrated how a CL-model can be interpreted as a ML-model (Example 12). This ML-model is indeed a $AX_{\Sigma_{CL}}$-model. Conversely, every $AX_{\Sigma_{CL}}$-model gives rise to a CL-model.

Lemma 21.

1. *Every CL-Model $\mathcal{M}$ gives rise to an AX_{CL}-model $\mathcal{M}_{\Sigma_{CL}}$.*
2. *Every AX_{CL}-model $\mathcal{M}$ gives rise to a CL-model $\mathcal{M}_{AX_{CL}}$.*
3. *The CL-model $\mathcal{M}$ equals the CL-model $(\mathcal{M}_{\Sigma_{CL}})_{AX_{CL}}$.*
4. *The AX_{CL}-model $\mathcal{M}$ equals the AX_{CL}-model $(\mathcal{M}_{AX_{CL}})_{\Sigma_{CL}}$.*
5. *The satisfaction relations agree.*

Proof. Part 1 follows from Example 12 by observing that the AX-axioms are satisfied by $\mathcal{M}_{\Sigma_{CL}}$. The construction of $\mathcal{M}_{AX_{CL}}$ and the proof of part 2 is given below. Parts 3 and 4 follow from the constructions of the models. Part 5 is stated in more detail and proved in Example 15. For part 2, let $\mathcal{M}$ be a AX_{CL}-model, with sets $\mathcal{M}_\mathrm{D}$ and $\mathcal{M}_\mathrm{C}$, and interpretations $\mathcal{M}_{ap}$, $\mathcal{M}_I$, $\mathcal{M}_\blacktriangleleft$ and $\mathcal{M}_\blacktriangleright$. The tuple $\mathcal{M}_{AX_{CL}} = (\mathcal{M}_\mathcal{D}, \mathcal{M}_\mathcal{C}, \mathcal{M}_{ap}, \mathcal{M}_I)$ is a CL-model. In particular, axioms 1 and 2 give the condition that $\mathcal{M}_I$ is a left unit of $\mathcal{M}_{ap}$. Axioms 3 to 6 give the condition $(c, d, d') \in \mathcal{M}_{ap} \Leftrightarrow (c, d', d) \in \mathcal{M}_\blacktriangleleft \Leftrightarrow (d, d', c) \in \mathcal{M}_\blacktriangleright$, which captures exactly the relationship between connectives $ap, \blacktriangleleft, \blacktriangleright$. $\square$

Finally, we connect the proof theory of CL and AX_{CL}.

Lemma 22. *Given arbitrary $P_1, P_2 \in \mathcal{P}$ and $K_1, K_2 \in \mathcal{K}$,*

$$P_1 \vdash_{CL} P_2 \quad \textit{iff} \quad \vdash_{AX_{CL}} |P_1| \Rightarrow |P_2|$$
$$K_1 \vdash_{CL} K_2 \quad \textit{iff} \quad \vdash_{AX_{CL}} |K_1| \Rightarrow |K_2|$$

Proof. For simplicity of notation we omit the explicit conversion $|P|$. The proof consists of two parts:

1. the rules of $\vdash_{CL}$ are derivable in $\vdash_{AX_{CL}}$;
2. the axioms in AX_{CL} are derivable in $\vdash_{CL}$.

For each part we give one case in detail, the other cases follow similarly. For the first part, we show that the following is derivable

$$\frac{\vdash_{AX_{CL}} K(P_1) \Rightarrow P_2}{\vdash_{AX_{CL}} K \Rightarrow (P_1 \triangleright P_2)}$$

First observe that

$$\vdash_{AX_{CL}} K \Rightarrow (P_1 \triangleright P_2) \text{ iff } \vdash_{AX_{CL}} \neg((P_1 \blacktriangleright \neg P_2) \wedge K).$$

Using AX_{CL}-axiom 5, we obtain :

$$\vdash_{AX_{CL}} K \wedge (P_1 \blacktriangleright \neg P_2) \Rightarrow \text{true} \blacktriangleright (\neg P_2 \wedge K(P_1))$$

From the assumption $\vdash_{AX_{CL}} K(P_1) \Rightarrow P_2$ we have

$$\vdash_{AX_{CL}} \text{true} \blacktriangleright (\neg P_2 \wedge K(P_1)) \Rightarrow \text{true} \blacktriangleright (\neg P_2 \wedge P_2)$$

Since $\vdash_{AX_{CL}}$ true $\blacktriangleright (\neg P_2 \wedge P_2) \Leftrightarrow$ false, we have proved

$$\vdash_{AX_{CL}} \neg((P_1 \blacktriangleright \neg P_2) \wedge K)$$

For the second part, we show that axiom 3 is derivable, by proving the following stronger version:

$$\vdash_{CL} q \wedge k(p) \Rightarrow k(p \wedge (k \blacktriangleleft q))$$

Note that, for propositional variable r, we have:

$$\vdash_{CL} k(p) \Leftrightarrow k(p \wedge (r \vee \neg r)) \Leftrightarrow k(p \wedge r) \vee k(p \wedge \neg r)$$

If we replace r by $\neg(k \blacktriangleleft q)$, then our stronger version of axiom 3 follows from proving that

$$\vdash_{CL} q \wedge k(p \wedge \neg(k \blacktriangleleft q)) \Rightarrow \text{false}$$

Since $\vdash_{CL} p \wedge \neg(k \blacktriangleleft q) \Rightarrow \neg(k \blacktriangleleft q)$ and $\vdash_{CL} \neg(k \blacktriangleleft q) \Leftrightarrow k \lhd \neg q$ by definition, we derive

$$\vdash_{CL} q \wedge k(p \wedge \neg(k \blacktriangleleft q)) \Rightarrow q \wedge k(k \lhd \neg q)$$

Finally, since $\vdash_{CL} k(k \lhd \neg q)) \Rightarrow \neg q$, we conclude that

$$\vdash_{CL} q \wedge k(k \lhd \neg q) \Rightarrow q \wedge \neg q \Rightarrow \text{false}$$

□

Theorem 23 (Soundness and Completeness)**.** *The proof theory of CL (Definition 5) is sound and complete with respect to the class of CL-models (Definition 2).*

Proof. Immediate from Theorem 19, using lemmas 21 and 22. □

Using this result, it is also possible to prove completeness for the restricted class of functional CL-models: that is, those CL-models where ap is a function. For each relational model, it is possible to construct a functional model which satisfies the same formulae. This proof is given in Zarfaty's forthcoming thesis. In [3], we also study CL with an additional zero formula 0, since it has interesting logical structure. It is possible to give additional axioms for 0, and provide an analogous completeness result.

3.2 Bunched Logic as ML

We show how BL can be expressed in ML, by analogy with CL. The ML-signature Σ_{BL} consists of one sort D with the modalities $\circ : (\mathrm{D}, \mathrm{D}) \to \mathrm{D}$, $0 : () \to \mathrm{D}$, $\bullet\!\!- : (\mathrm{D}, \mathrm{D}) \to \mathrm{D}$ and $-\!\!\bullet : (\mathrm{D}, \mathrm{D}) \to \mathrm{D}$. The axiom set AX_{BL} is:

1. $0 \circ p \Rightarrow p$
2. $p \Rightarrow 0 \circ p$
3. $(p \circ q) \circ r \Rightarrow p \circ (q \circ r)$
4. $p \circ (q \circ r) \Rightarrow (p \circ q) \circ r$
5. $q \wedge (r \circ p) \Rightarrow \text{true} \circ (p \wedge (r \bullet\!\!- q))$
6. $q \wedge (r \circ p) \Rightarrow (r \wedge (p -\!\!\bullet q)) \circ \text{true}$
7. $p \wedge (r \bullet\!\!- q) \Rightarrow \text{true} \bullet\!\!- (q \wedge (r \circ p))$
8. $r \wedge (p -\!\!\bullet q) \Rightarrow \text{true} -\!\!\bullet (q \wedge (r \circ p))$

The set AX_{BL} is a set of very simple Sahlqvist formulae, and hence we have an analogous completeness result to CL. We do not know how to prove completeness for the functional BL-models, because the construction of a functional model from a relational model does not preserve associativity.

4. Applications of Context Logic

We shall study three applications of CL: heaps to give an example where CL- and BL-reasoning are the same; sequences to give an example where CL- and BL-reasoning is different; trees to provide a more substantial example where the reasoning is different.

4.1 Heaps

CL for heaps is CL extended by specific modalities which can be interpreted in the CL-model $Heap$ (Example 3). The extra modalities specify the empty heap and heaps containing a specific cell.

Definition 24 (CL for $Heap$)**.** *CL for heaps, denoted $\mathcal{CL}_{Heap}$, is given by the ML-signature $\Sigma_{CL+Heap}$ consisting of Σ_{CL} extended by the modalities:*

$$0 : () \to D, \quad n \hookrightarrow m : () \to D$$

for every $n \in \mathbb{N}^+$ and $m \in \mathbb{N}$. The CL-model $Heap$ is a model of $\mathcal{CL}_{Heap}$ with the the additional modalities interpreted as:

$$\begin{array}{rcl} \mathcal{M}_0 & = & \{e\} \quad \textit{where } e \textit{ denotes the empty heap} \\ \mathcal{M}_{n \hookrightarrow m} & = & \{h \in \mathcal{D} \mid h(n) = m\} \end{array}$$

We have the following derived formulae:

- $P_1 \circ P_2 \triangleq (0 \rhd P_1)(P_2)$ specifying that a heap can be split into two disjoint parts, one satisfying P_1 and the other P_2;
- $n \mapsto m \triangleq (n \hookrightarrow m) \wedge \neg(\neg 0 \circ \neg 0)$ specifying that the given heap h contains just one cell with $h(n) = m$;
- $P_1 -\!\!\circ\, P_2 \triangleq (0 \rhd P_1) \lhd P_2$ specifying that, whenever a heap satisfying P_1 can be composed with the given heap, the result satisfies P_2.

To illustrate the difference between $\circ$ and $\wedge$, notice that the formula $(n \mapsto m) \circ (n \mapsto m)$ is not satisfied by any heap, whereas the formula $(n \mapsto m) \wedge (n \mapsto m)$ specifies a one-cell heap. Also, notice that logical equivalence of $\mathcal{CL}_{Heap}$ is heap equality. This strength of analysis is typical for this style of logical reasoning.

BL for heaps is Separation Logic [11]. It is defined similarly to $\mathcal{CL}_{Heap}$, by extending the signature Σ_{BL} to include modalities for specifying the existence of heap cells.

Definition 25 (BL for $Heap$)**.** *BL for heaps, denoted $\mathcal{BL}_{Heap}$, is given by the ML-signature*

$$\Sigma_{BL+Heap} = \Sigma_{BL} \cup \{n \hookrightarrow m : () \to \mathcal{D} \mid n \in \mathbb{N}^+, m \in \mathbb{N}\}.$$

The BL-model $Heap$ is a model of $\mathcal{BL}_{Heap}$, with the interpretation of the additional modalities defined in Definition 24.

Again, we can derive the BL-formula $n \mapsto m$ denoting a one-cell heap. In SL, this formula is primitive. We choose $n \hookrightarrow m$ as primitive here, because of a comparison with propositional logic given in Section 5, where $n \hookrightarrow m$ is the natural atomic formula.

From the derived CL-formulae given previously, we know that $\mathcal{BL}_{Heap}$ is a sublogic of $\mathcal{CL}_{Heap}$. In fact, there is a collapse of the CL-structure for the heap case, in that $\mathcal{CL}_{Heap}$ and $\mathcal{BL}_{Heap}$ are equivalent logics on data (Proposition 26). This result is strong, in that it implies that the logics are parametrically as expressive as each other on data (Definition 31).

Proposition 26. *Let $\mathcal{P}_{\Sigma_{CL+Heap}, \mathcal{V}_\mathcal{P}}$ denote the restriction of set $\mathcal{P}_{\Sigma_{CL+Heap}}$ to those formulae with propositional variables in $\mathcal{V}_\mathcal{P}$, and similarly for BL. There exists a bijection $\widehat{\ } : \mathcal{P}_{\Sigma_{CL+Heap}, \mathcal{V}_\mathcal{P}} \to \mathcal{P}_{\Sigma_{BL+Heap}, \mathcal{V}_\mathcal{P}}$ which preserves the satisfaction relation: that is,*

$$\begin{array}{rcl} \sigma, d \vDash_{CL} P & \Leftrightarrow & \sigma, d \vDash_{BL} \widehat{P} \\ \sigma, c \vDash_{CL} K & \Leftrightarrow & \sigma, c(0) \vDash_{BL} \widehat{K} \end{array}$$

Proof. The translation is defined inductively on the structure of data and context formulae. We give the cases for the modalities:

$$\begin{array}{ll} \widehat{K(P)} \triangleq \widehat{K} \circ \widehat{P} & \widehat{I} \triangleq 0 \\ \widehat{K \blacktriangleleft P} \triangleq \widehat{K} -\!\!\bullet \widehat{P} & \widehat{P_1 \blacktriangleright P_2} \triangleq \widehat{P_1} -\!\!\bullet \widehat{P_2} \\ \widehat{0} \triangleq 0 & \widehat{n \hookrightarrow m} \triangleq n \hookrightarrow m \end{array}$$

The proof follows by induction. □

4.2 Sequences

CL for sequences generated by alphabet A is CL extended by specific modalities which can be interpreted in the CL-model Seq_A presented in Example 3. The additional modalities specify the empty sequence, sequences with just one element $a \in A$, and modalities for analysing sequence contexts.

Definition 27 (CL for Seq_A). *CL for Sequences generated by alphabet A, denoted $\mathcal{CL}_{Seq_A}$, is given by the ML-signature Σ_{CL+Seq_A} consisting of Σ_{CL} extended by the modalities:*

$$0 : () \to D, \quad a : () \to D, \quad \circ_r : (C, D) \to C \quad \circ_l : (D, C) \to C$$

for $a \in A$. The CL-model Seq_A is a model of $\mathcal{CL}_{Seq_A}$ with the interpretation of the additional modalities given by:

$$\begin{array}{l} \mathcal{M}_0 = \{0\} \quad \textit{where } 0 \textit{ denotes the empty sequence} \\ \mathcal{M}_a = \{a\} \quad \textit{for each } a \in A \\ \mathcal{M}_{\circ_r} = \{(c, s, c \cdot s) \mid c \in \mathcal{C}_A, s \in \mathcal{D}_A\} \\ \mathcal{M}_{\circ_l} = \{(s, c, s \cdot c) \mid s \in \mathcal{D}_A, c \in \mathcal{C}_A\} \end{array}$$

We will use the notation $K \circ P$ for $\circ_r(K, P)$ and $P \circ K$ for $\circ_l(P, K)$, overloading $\circ$ as the subscripts can be inferred.

We require the additional modalities $\circ_r$ and $\circ_l$ for analysing sequence contexts, since unlike the heap case these cannot be derived from application. We can derive a formula for sequence composition $P_1 \circ P_2 \triangleq (P_1 \circ I)(P_2)$ which specifies that a sequence can be split into two sequences, the left one satisfying P_1 and the right one P_2. This is logically equivalent to $(I \circ P_2)(P_1)$. We also derive the two corresponding right adjoints: the formula $P_1 \circ\!\!- P_2 \triangleq (P_1 \circ I) \triangleleft P_2$ specifies that, whenever a sequence satisfying property P_1 is joined to the left of the given sequence, then the result satisfies P_2; similarly for $P_1 -\!\!\circ P_2 \triangleq (I \circ P_1) \triangleleft P_2$. For example, the formula $a -\!\!\circ P$ specifies that joining a to the right of the sequence results in a sequence satisfying P. In contrast, notice that the formula $(a \triangleright P)(0)$ specifies that, whenever an a is put *somewhere* in the given sequence, then the result satisfies property P. Again, logical equivalence of $\mathcal{CL}_{Seq_A}$ is sequence equality.

Definition 28 (BL for Seq_A). *BL for sequences generated from alphabet A, denoted $\mathcal{BL}_{Seq_A}$, is given by the ML-signature*

$$\Sigma_{BL+Heap} = \Sigma_{BL} \cup \{a \mid a \in A\}.$$

The BL-model Seq_A is a model of $\mathcal{BL}_{Seq_A}$ with the interpretation of the additional modalities defined as given in Definition 27.

$\mathcal{BL}_{Seq_A}$ is a sublogic of $\mathcal{CL}_{Seq_A}$. Unlike the heap case, there is no collapse of the CL-reasoning, and the question of whether $\mathcal{CL}_{Seq_A}$ is more expressive than $\mathcal{BL}_{Seq_A}$ is subtle. Consider the CL-formula $(0 \triangleright b \circ c)(a)$. It is logically equivalent to the formula $a \circ b \circ c \vee b \circ a \circ c \vee b \circ c \circ a$. Now consider the CL-formula $(0 \triangleright \text{True}(b))(a)$. It is equivalent to true $\circ\, b \circ$ true $\circ\, a \circ$ true $\vee$ true $\circ\, a \circ$ true $\circ\, b \circ$ true, which has very different structure to the previous example. We shall see in section 5.2 that $\mathcal{CL}_{Seq_A}$ and $\mathcal{BL}_{Seq_A}$ are equality expressive in the sense that every CL-formula without propositional variables has an equivalent BL-formula. However, they are not *parametrically* as expressive, in the sense that the CL-formula $(0 \triangleright p)(a)$, for propositional data variable p, cannot be expressed in $\mathcal{BL}_{Seq_A}$. By contrast, $\mathcal{CL}_{Seq_A}$ without the $\blacktriangleright$ modality is parametrically as expressive as $\mathcal{BL}_{Seq_A}$.

4.3 Trees

CL for trees generated by alphabet A is CL extended by specific modalities which can be interpreted in the CL-model $Tree_A$ presented in Example 3. The additional modalities correspond to the empty tree, and modalities for analysing tree contexts.

Definition 29 (CL for $Tree_A$). *CL for trees, denoted $\mathcal{CL}_{Tree_A}$, is given by the ML-signature $\Sigma_{CL+Tree_A}$ consisting of Σ_{CL} extended by the additional modalities:*

$$0 : () \to D, \quad \mu : (C) \to C, \quad \circ : (D, C) \to C$$

where $\mu ::= a \mid a^{\perp}$ for $a \in A$. The CL-model $Tree_A$ is a model of $\mathcal{CL}_{Tree_A}$ with the interpretation of the additional modalities given by:

$$\begin{array}{lcl} \mathcal{M}_0 & = & \{0\} \textit{ where } 0 \textit{ denotes the empty tree} \\ \mathcal{M}_a & = & \{(c, a[c]) \mid c \in \mathcal{C}\} \\ \mathcal{M}_{a^{\perp}} & = & \{(c, a'[c]) \mid c \in \mathcal{C}, a' \in A - \{a\}\} \\ \mathcal{M}_{\circ} & = & \{(t, c, t \mid c) \mid t \in \mathcal{D}, c \in \mathcal{C}\} \end{array}$$

We write $\mu[K]$ for $\mu(K)$, and $P \circ K$ for $\circ(P, K)$.

Apart from the 0, the additional modalities describe ways of analysing tree contexts: either tree contexts consist of a root μ with a subcontext underneath, or they can be split at the top level into data and a context. The root μ can either be the node label $a \in A$, or $a^{\perp}$ denoting any label which is not a. The $a^{\perp}$ modalities are not given explicitly in [3], since they are derivable with existential quantification and label equality. They are important for our comparison with BL-reasoning, and also play a prominent role in logic-based query languages for XML (see XDUCE [7]). We only require one modality $\circ$ for splitting contexts, since our tree composition is commutative. Analogous to the heap case, we have the derived formulae $P_1 \circ P_2 \triangleq (P_1 \circ I)(P_2)$ and $P_1 -\!\!\circ P_2 \triangleq (P_1 \circ I) \triangleleft P_2$.

Definition 30 (BL for $Tree_A$). *BL for trees generated from alphabet A, denoted $\mathcal{BL}_{Tree_A}$, is given by the ML-signature $\Sigma_{BL+Tree_A}$ consisting of Σ_{BL} extended by the modalities:*

$$\mu : (D) \to D, \quad \widehat{\mu} : (D) \to D, \quad \diamond : (D) \to D \quad \widehat{\diamond} : (D) \to D$$

The BL-model $Tree_A$ is a model of $\mathcal{BL}_{Tree_A}$ with the interpretation of the additional modalities given by:

$$\begin{array}{lcl} \mathcal{M}_a & = & \{(t, a[t]) \mid t \in \mathcal{D}\} \\ \mathcal{M}_{a^{\perp}} & = & \{(t, a'[t]) \mid t \in \mathcal{D}, a' \in A - \{a\}\} \\ \mathcal{M}_{\widehat{\mu}} & = & \{(a'[t], t) \mid (t, a'[t]) \in \mathcal{M}_{\mu}\} \\ \mathcal{M}_{\diamond} & = & \{(t, c(t)) \mid t \in \mathcal{D}, c \in \mathcal{C}\} \\ \mathcal{M}_{\widehat{\diamond}} & = & \{(c(t), t) \mid (t, c(t)) \in \mathcal{M}_{\diamond}\} \end{array}$$

$\mathcal{BL}_{Tree_A}$ is a sublogic of $\mathcal{CL}_{Tree_A}$. The BL-formula $\mu[P]$ specifies a tree with top node described by μ. It is derivable in $\mathcal{CL}_{Tree_A}$ as $(\mu[P \circ I])(0)$. The modalities $a^{\perp}$ are essential to express deep properties of trees in BL. For example, let $\mathbf{1}$ denote all the trees with one node. In $\mathcal{CL}_{Tree_A}$, this can be expressed as $\neg(\neg I(\neg 0)) \wedge \neg 0$. It is expressed in $\mathcal{BL}_{Tree_A}$ as $a[0] \vee a^{\perp}[0]$, and is not expressible without the $a^{\perp}$ modality. The BL-formula $\widehat{\mu}[P]$ is the adjoint. It specifies that, whenever a top node is added to the given tree with label specified by μ, the resulting tree satisfies P. It corresponds to the CL-formula $\mu[I] \triangleleft P$. AL has the formula $\widehat{a}[P]$, with $\widehat{a^{\perp}}[P]$ derivable using existential quantification and label equality. The BL-formula $\diamond(P)$ denotes that there is a subtree satisfying P. Recall that this is expressible in CL as True(P). Finally, the BL-formula $\widehat{\diamond}(P)$ is the corresponding adjoint. It states that the given tree can be put in a context such that the result satisfies P, and is expressible in CL as True $\triangleleft P$. AL has the formulae $\diamond(P)$, but surprisingly not the corresponding adjoint. We will see that $\widehat{\diamond}$ plays an important role in an expressivity result linking BL- and CL-reasoning without $\blacktriangleright$ (Theorem 40).

Just as for sequences, the CL-reasoning does not collapse for trees, and the comparison between $\mathcal{CL}_{Tree_A}$ and $\mathcal{BL}_{Tree_A}$ is subtle. Consider the CL-formula $(0 \triangleright b[0])(a[\text{true}])$, which specifies that we can remove a subtree with root label a to obtain a tree with one node labelled b. It corresponds to the BL-formula $a[\text{true}] \circ b[0] \vee b[a[\text{true}]]$. Now consider the CL-formula $(0 \triangleright \diamond b[\text{true}])(a[\text{true}])$. It corresponds to the BL-formula $\diamond(b[\diamond a[\text{true}]]) \vee \diamond(\diamond(b[\text{true}]) \circ \diamond(a[\text{true}]))$. Notice that the structure of the CL-formulae is similar, but the structure of the derivable BL-formulae is quite different. In Section 5.3, we shall see that $\mathcal{CL}_{Tree_A}$ and $\mathcal{BL}_{Tree_A}$ are not parametrically as expressive, in the sense that CL-formula $(0 \triangleright p)(a[\text{true}])$ cannot be expressed in $\mathcal{BL}_{Tree_A}$. $\mathcal{CL}_{Tree_A}$ without the $\blacktriangleright$ modality is as expressive as $\mathcal{BL}_{Tree_A}$.

5. Parametric Expressivity

We present our expressivity results. We prove parametric inexpressivity results comparing BL for heaps with and without the structural connectives, using a direct proof method. We prove strong inexpressivity results for specific interpretations σ, corresponding to the extension of BL with list predicates. We also study parametric inexpressivity results comparing CL and BL for sequences and trees. We cannot prove the parametric results directly for these applications. Instead, we prove strong inexpressivity results, and hence by implication the parametric results. In addition, we study BL for bounded heaps, presenting logics which have the same strong expressivity for every interpretation σ, but different parametric expressivity.

First some notation. Given ML-signature $\Sigma = (\mathcal{S}, O, \rho)$ and AX-model $\mathcal{M}_\Sigma$, we let $\mathcal{L}_{\mathcal{M}_\Sigma,\mathcal{V}}$ denote the modal logic determined by Σ and $\mathcal{M}_\Sigma$ with propositional variables in $\mathcal{V}$. We use $\mathcal{L}_{\mathcal{M}_\Sigma,\mathcal{V}_S}$ to denote the restriction to propositional variables of sort S, and $\mathcal{L}_{\mathcal{M}_\Sigma,\{p\}}$ for the restriction to propositional variable p. We write $\mathcal{L}_{\mathcal{M}_\Sigma,V}(-\triangle)$ for the logic without modality $\triangle$. For example, $\mathcal{CL}_{Seq_A,\mathcal{V}_\text{D}}(-\{\blacktriangleright\})$ denotes $\mathcal{CL}_{Seq_A}$ without modality $\blacktriangleright$ and with the propositional variables restricted to sort D.

Definition 31 (Parametric Expressivity). *Let $\Sigma = (\mathcal{S}, O, \rho)$ and $\Sigma' = (\mathcal{S}', O', \rho')$ be two ML-signatures with sort $S \in \mathcal{S} \cap \mathcal{S}'$. Consider two models $\mathcal{M}, \mathcal{M}'$ over the respective signatures such that $\mathcal{M}_S = \mathcal{M}'_S$. Consider the logics $\mathcal{L}_{\mathcal{M},\mathcal{V}_S}$ and $\mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$.*

1. *Logic $\mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ is as expressive as $\mathcal{L}_{\mathcal{M},\mathcal{V}_S}$ with respect to sort S, written $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subseteq_S \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$, if and only if $\forall P \in \mathcal{L}_{\mathcal{M},\mathcal{V}_S}.\exists P' \in \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}.\,\forall\sigma.\forall m \in \mathcal{M}_S.$*
$$\sigma, \mathcal{M}, m \vDash_S P \Leftrightarrow \sigma, \mathcal{M}', m \vDash_S P' \ .$$
We often write $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subseteq \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ since the sort S is apparent.
2. *Given σ, logic $\mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ is as σ-expressive as $\mathcal{L}_{\mathcal{M},\mathcal{V}_S}$ with respect to sort S, written $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subseteq_{S,\sigma} \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$, if and only if $\forall P \in \mathcal{L}_{\mathcal{M},\mathcal{V}_S}.\exists P' \in \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}.\,\forall m \in \mathcal{M}_S.$*
$$\sigma, \mathcal{M}, m \vDash_S P \Leftrightarrow \sigma, \mathcal{M}', m \vDash_S P' \ .$$
Again, we write $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subseteq_\sigma \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ since S is apparent.

We write $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} = \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ when $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subseteq \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ and $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \supseteq \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$, and $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} =_\sigma \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ analogously.

In fact, we concentrate on inexpressivity results, asking when a logic is strictly more expressive than another. We say that $\mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ is *parametrically* more expressive than $\mathcal{L}_{\mathcal{M},\mathcal{V}_S}$ iff $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subsetneq \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$. We say that $\mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ is *strongly* more expressive than $\mathcal{L}_{\mathcal{M},\mathcal{V}_S}$ iff there exists a σ such that $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subsetneq_\sigma \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$.

To prove our strong inexpressivity results we rely heavily on the notion of bisimulation. Bisimulation has the property that, if two elements of the models are bisimilar, then they cannot be distinguished from each other in the logic, and hence can be directly used to prove strong inexpressivity.

Definition 32 (Bisimulation). *A symmetric binary relation $\sim = \bigcup_{S\in\mathcal{S}} \sim_S$ with $\sim_S \subseteq \mathcal{M}_S \times \mathcal{M}_S$ is a* bisimulation *for ML-model $\mathcal{M}_\Sigma$ if and only if, whenever $((m_1, \ldots, m_n), m) \in \mathcal{M}_\triangle$ and $m \sim m'$, then there exist $m'_i \in \mathcal{M}_{S_i}$ such that $m_i \sim_{S_i} m'_i$ for $i = 1 \ldots n$ and $((m'_1, \ldots, m'_n), m') \in \mathcal{M}_\triangle$. A bisimulation $\sim$ is* compatible *with interpretation σ if and only if, for all m, p, $m \in \sigma(p) \wedge m \sim m' \Rightarrow m' \in \sigma(p)$.*

Proposition 33. *Let $\sim$ be a bisimulation compatible with σ for ML-model $\mathcal{M}_\Sigma$. Then $m_1 \sim_S m_2$ implies $\sigma, \mathcal{M}, m_1 \vDash_S P$ iff $\sigma, \mathcal{M}, m_2 \vDash_S P$ for all $P \in \mathcal{P}_{\Sigma_S}$.*

To show that $\mathcal{L}_{\mathcal{M},\mathcal{V}_S} \subsetneq_\sigma \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$, our proof method consists of finding a formula $P \in \mathcal{L}_{\mathcal{M}',\mathcal{V}_S}$ of sort S and a bisimulation $\sim$ for $\mathcal{M}$ compatible with σ, such that P distinguishes two elements which are identified by $\sim$. Inexpressivity then follows from Proposition 33 that no formula in $\mathcal{P}_\Sigma$ can distinguish the two elements.

5.1 Heaps

We have the following results about heaps:

- $\mathcal{BL}_{Heap,\mathcal{V}_\text{D}} = \mathcal{CL}_{Heap,\mathcal{V}_\text{D}}$
- $\mathcal{BL}_{Heap,\emptyset}(-\{-\!\!\bullet\} + \{n\hookrightarrow -\}) = \mathcal{BL}_{Heap,\emptyset}$
- $\mathcal{BL}_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow -\}) \subsetneq \mathcal{BL}_{Heap,\{p\}}$

The first result is a parametric expressivity result, showing that the structure of CL collapses to BL in the heap case. This result follows from Proposition 26. The second result is a non-parametric expressivity result due to Lozes [9], which compares the expressivity of closed formulae. It states that the adjunct $-\!\!\bullet$ can be eliminated if we add formulae $n\hookrightarrow -$ to specify that address n is allocated: $n\hookrightarrow -$ specifies that there exists m such that $n\hookrightarrow m$. It is expressible using $-\!\!\bullet$ as $\neg((n\hookrightarrow 0) -\!\!\bullet \text{true})$; the current heap cannot be extended with cell n, so n must already be allocated. Lozes' result was initially quite a surprise, since the adjuncts are used in an essential way to describe the weakest preconditions for Hoare reasoning based on BL for heaps. We address this apparent contradiction here, by proving the third result.

The third result is a parametric inexpressivity result, which we prove by showing that formula $(n\mapsto 0) -\!\!\circ\, p$ cannot be expressed in BL for heaps without $-\!\!\bullet$. This formula is interesting because it expresses the weakest precondition of allocation. This inexpressivity result says nothing about whether the formula is expressible for a particular interpretation of p. By Lozes' results, we know that it is expressible for every interpretation of p as a closed formula. We also prove a strong inexpressivity result using the same formula and a natural interpretation σ_{list} which interprets p as $list(m)$: that is, the heap contains a 0-terminated linked list starting at m. This result is interesting because lists are one of the typical inductive predicates used in BL-reasoning.

Theorem 34 (Parametric Inexpressivity).
$$\mathcal{BL}_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow -\}) \subsetneq \mathcal{BL}_{Heap,\{p\}}$$

Proof. We define the notion of heap-reducing formulae, and prove that all the formulae in $\mathcal{BL}_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow -\})$ are heap-reducing, whereas formula $(n\mapsto 0) -\!\!\circ\, p$ is not. This captures our intuition is that, without $-\!\!\bullet$, the BL-modalities either leave the heap alone or make it smaller, whereas formula $(n\mapsto 0) -\!\!\circ\, p$ crucially tests p on an extension of the initial heap with cell n.

Given heap h, define a binary relation on interpretations by
$$\sigma \sim_h \sigma' \text{ iff } \forall h' \leq h.\, h' \in \sigma(p) \Leftrightarrow h' \in \sigma'(p)$$
where $h' \leq h$ means $dom(h') \subseteq dom(h)$ and $\forall n \in dom(h').\, h'(n) = h(n)$. We say that a formula P is heap-reducing if and only if, whenever $\sigma \sim_h \sigma'$, it follows that $\sigma, h \models P \Leftrightarrow \sigma', h \models P$.

Given formula P in $\mathcal{BL}_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow -\})$, we show that P is heap-reducing by induction on the structure of P. Case p is immediate from the definition of $\sigma \sim_h \sigma'$. The only other interesting case is $P_1 \circ P_2$. Suppose $\sigma \sim_h \sigma'$ and $\sigma, h \models P_1 \circ P_2$. There exists h_i for $i = 1, 2$ such that $h = h_1 \cdot h_2$ and $\sigma, h_i \models P_i$. Since $h_i \leq h$, we have $\sigma \sim_{h_i} \sigma'$ for $i = 1, 2$ and hence $\sigma', h_i \models P_i$ by the induction hypothesis. We conclude that $\sigma', h \models P$.

Let P' be $(n\mapsto 0) -\!\!\circ\, p$. We show that P' is not heap-reducing. Define $\sigma(p) = \{h \mid h : \mathbb{N}^+ \rightharpoonup_{fin} \mathbb{N}\}$ and $\sigma'(p) = \{e\}$ where e is the empty heap. Then $\sigma \sim_e \sigma'$ and $\sigma, e \models P'$, but $\sigma', e \not\models P'$. $\square$

We now prove our strong inexpressivity result, that $-\!\!\bullet$ cannot be eliminated with interpretation $\sigma_{list}(p) = list(m)$. With this interpretation, the formula $(n\mapsto 0) -\!\!\circ p$ is satisfied by a list segment starting at m and stopping with dangling pointer n. This cannot be expressed without $-\!\!\circ$, since only whole lists can be observed.

Theorem 35 (Strong Inexpressivity)**.**

$$\mathcal{B}L_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow-\}) \subsetneq_{\sigma_{list}} \mathcal{B}L_{Heap,\{p\}}$$

Proof. Consider the BL-formula $P \triangleq (n\mapsto 0) -\!\!\circ\, p$, and interpretation $\sigma_{list}(p) = list(m)$ for $m \neq n$. We show that there is no formula P' in $\mathcal{B}L_{Heap,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow-\})$ that is equivalent to P using σ_{list}. Expecting a contradiction, suppose that such a P' exists, and let $L \subseteq \mathbb{N}$ be the finite number of constants mentioned in P'. Consider the restriction of $\mathcal{B}L_{Heap}$ to constants in L, written $\mathcal{B}L_{Heap_L}$. We choose a bisimulation $\sim$ which identifies two heaps when they have the same domain, coincide on values in L, and are identical if one of them contains a list starting at m:

$$\begin{array}{lll} h_1 \sim h_2 & \text{iff} & dom(h_1) = dom(h_2) \text{ and} \\ & & \forall i,j \in L.\, h_1(i) = j \text{ iff } h_2(i) = j \text{ and} \\ & & \text{if } h_1 \models list(m) \text{ or } h_2 \models list(m) \text{ then } h_1 = h_2. \end{array}$$

Notice that $\sigma_{list}(p)$ is compatible with $\sim$, since if $h_1{\sim}h_2$ and $h_1 \models list(m)$ then $h_2 \models list(m)$. Assume for the moment that $\sim$ is indeed a bisimulation. With this assumption, we show that we do indeed obtain a contradiction. Take $m' \notin L \cup \{n, m\}$ and consider the heaps $h_1 = [m \mapsto m', m' \mapsto n]$ and $h_2 = [m \mapsto m', m' \mapsto m']$. Note that $h_1{\sim}h_2$ and $\sigma, h_1 \models (n\mapsto 0) -\!\!\circ\, p$ but $\sigma, h_2 \not\models (n\mapsto 0) -\!\!\circ\, p$. Assuming that $\sim$ is a bisimulation for $\mathcal{B}L_{Heap_L,\{p\}}(-\{-\!\!\bullet\}+\{n\hookrightarrow-\})$, Proposition 33 implies that $\sigma_{list}, h_1 \models_{BL} P'$ iff $\sigma_{list}, h_2 \models_{BL} P'$. We have therefore proved that there cannot be a BL-formula P' which is equivalent to P.

Finally, we must prove our assumption that $\sim$ is a bisimulation for the modalities $\{0, \circ\} \cup \{i\hookrightarrow j, i\hookrightarrow- \mid i,j \in L\}$. We only look at the $\circ$ modality; the other cases are trivial. Assume $h_1 = h_1' \cdot h_1''$ and $h_1{\sim}h_2$. We must show that $\exists h_2', h_2''$ such that $h_2 = h_2' \cdot h_2''$ and $h_1'{\sim}h_2'$ and $h_1''{\sim}h_2''$. Choose h_2', h_2'' as the unique splitting of h_2 such that $dom(h_2') = dom(h_1')$ and $dom(h_2'') = dom(h_1'')$. We show $h_1'{\sim}h_2'$; the case $h_1''{\sim}h_2''$ is identical by symmetry. The first and second conditions in the definition of $\sim$ are immediate. For the third condition, assume $h_1' \models list(m)$, which implies $h_1 \models list(m)$ since h_1' is a sublist of h_1. By definition of $h_1{\sim}h_2$, we have $h_1 = h_2$, hence $h_1' = h_2'$ which is the desired conclusion. □

Lozes also shows that, with an additional modality $size_r$ for determining the size of heaps, the $\circ$-modality can be removed. We show that $\circ$ is essential for parametric reasoning. The results are:

- $\mathcal{B}L_{Heap,\emptyset}(-\{\circ, -\!\!\bullet\} + \{size_r\}) = \mathcal{B}L_{Heap,\emptyset}(-\{-\!\!\bullet\})$
- $\mathcal{B}L_{Heap,\{p,q\}}(-\{\circ, -\!\!\bullet\}+\{size_r\}) \subsetneq \mathcal{B}L_{Heap,\{p,q\}}(-\{-\!\!\bullet\})$.

The results also hold with modality $n\hookrightarrow-$. Thus, BL is as expressive as PL with atomic formulae 0, $size_r$ and $n\hookrightarrow-$, but not parametrically so. We believe this parametric inexpressivity result demonstrates what has been always known intuitively, but by example only, that the $\circ$-modality is essential for modular reasoning.

We give a direct proof of our parametric inexpressivity result for $\circ$ based on the trivial observation that, without $\circ$ and $-\!\!\bullet$, all the modalities leave the current heap unchanged. The modality $size_r$, for each $r \in \mathbb{N}^+$, is interpreted by $h \models size_r$ iff $|dom(h)| \leq r$. We do not require $size_0$ as it corresponds to the zero formula 0.

Theorem 36 (Parametric Inexpressivity)**.**

$$\mathcal{B}L_{Heap,\{p,q\}}(-\{\circ, -\!\!\bullet\} + \{size_r\}) \subsetneq \mathcal{B}L_{Heap,\{p,q\}}(-\{-\!\!\bullet\})$$

Proof. The proof is analogous to that of Thm. 34. A formula P is *heap-invariant* iff, when $h \in \sigma(p) \Leftrightarrow h \in \sigma'(p)$ for all p, then $\sigma, h \models P \Leftrightarrow \sigma', h \models P$. The formulae in $\mathcal{B}L_{Heap,\{p,q\}}(-\{\circ, -\!\!\bullet\}+\{size_r\})$ are heap-invariant, but $p \circ q$ is not. □

We also prove a second strong inexpressivity result, that $\circ$ cannot be eliminated with fixed interpretation σ_{list}, which interprets p and q as $list(m_1)$ and $list(m_2)$ respectively for $m_1 \neq m_2$. The $\circ$ modality is essential for specifying the property that the two lists are in the heap and their tails never meet.

Theorem 37 (Strong Inexpressivity)**.**

$$\mathcal{B}L_{Heap,\{p,q\}}(-\{\circ, -\!\!\bullet\}+\{size_r\}) \subsetneq_{\sigma_{list}} \mathcal{B}L_{Heap,\{p,q\}}(-\{-\!\!\bullet\})$$

Proof. The structure of the proof is analogous to Theorem 35. Consider the BL-formula $p \circ q$. As in Theorem 35, we restrict our attention to BL-formulae mentioning at most a finite set $L \subseteq \mathbb{N}$ of constants and the propositional variables p, q.

Consider the interpretation $h \in \sigma_{list}(p)$ iff h satisfies $list(m_1)$, and $h \in \sigma_{list}(q)$ iff h satisfies $list(m_2)$. Define relation $\sim$ by:

$$\begin{array}{lll} h_1 \sim h_2 & \text{iff} & |dom(h_1)| = |dom(h_2)| \text{ and} \\ & & \sigma, h_1 \models P \Leftrightarrow \sigma, h_2 \models P \text{ for} \\ & & \quad P \in \{p, q\} \cup \{i\hookrightarrow j \mid i,j \in L\} \end{array}$$

$\sim$ is compatible with σ and is a bisimulation for the modalities $\{0, size_r\} \cup \{i \hookrightarrow j \mid i,j \in L\}$. Take $h_1 = [m_1 \mapsto n', m_2 \mapsto n'', n' \mapsto 0, n'' \mapsto 0]$ and $h_2 = [m_1 \mapsto n', m_2 \mapsto n', n' \mapsto 0, n'' \mapsto 0]$ for distinct $n', n'' \notin L \cup \{m_1, m_2\}$. Then $h_1 \sim h_2$ and $\sigma, h_1 \models p \circ q$, but $\sigma, h_2 \not\models p \circ q$. □

From the previous examples, one might be tempted to conclude that whenever a parametric inexpressivity result holds, a corresponding strong inexpressivity result holds too. In fact, this is not the case as we demonstrate using bounded heaps. In bounded heaps, $-\!\!\bullet$ cannot be eliminated parametrically for reasons identical to the unbounded case. However, given a specific interpretation σ, any formula is equivalent to a disjunction of characteristic formulae without $-\!\!\bullet$.

Theorem 38. *Let $Heap_k$ denote the restriction of heaps, and correspondingly formulae, to locations $\leq k$. The following hold:*

1. $\mathcal{B}L_{Heap_k,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow-\}) \subsetneq \mathcal{B}L_{Heap_k,\{p\}}$
2. $\mathcal{B}L_{Heap_k,\{p\}}(-\{-\!\!\bullet\} + \{n\hookrightarrow-\}) =_\sigma \mathcal{B}L_{Heap_k,\{p\}}$ *for all* σ.

Proof. The proof of part 1 is identical to the proof of Theorem 34. For part 2, we in fact prove a stronger claim: for any set of heaps $H \subseteq Heap_k$, there exists a formula $P_H \in \mathcal{B}L_{Heap_k,\emptyset}(-\{0, \circ, -\!\!\bullet\})$ such that $h \models P_H \Leftrightarrow h \in H$. We show that the conclusion then follows from the claim. Given any σ and $P \in \mathcal{B}L_{Heap_k,\{p\}}$, let $H = \{h \mid \sigma, h \models P\}$. Then P is equivalent to P_H, by definition. To prove the claim, we first define, given heap h, the characteristic formula P_h as

$$\bigwedge_{n,m \leq k.h(n)=m} n\hookrightarrow m \wedge \bigwedge_{n,m \leq k.h(n)\neq m)} \neg(n\hookrightarrow m)$$

Clearly $h' \models P_h \Leftrightarrow h = h'$, and $P_h \in \mathcal{B}L_{Heap_k,\emptyset}(-\{0, \circ, -\!\!\bullet\})$. To conclude, we define $P_H = \bigvee_{h\in H} P_h$. □

5.2 Sequences

We have the following results for sequences for infinite alphabet A:

- $\mathcal{B}L_{Seq_A,\emptyset}(-\{-\!\!\bullet, \bullet\!\!-\}) = \mathcal{B}L_{Seq_A,\emptyset} = \mathcal{C}L_{Seq_A,\emptyset}$
- $\mathcal{B}L_{Seq_A,\mathcal{V}_D}(-\{-\!\!\bullet, \bullet\!\!-\}) = \mathcal{C}L_{Seq_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$
- $\mathcal{B}L_{Seq_A,\mathcal{V}_D} = \mathcal{C}L_{Seq_A,\mathcal{V}_D}(-\{\blacktriangleright\})$
- $\mathcal{B}L_{Seq_A,\{p\}} \subsetneq \mathcal{C}L_{Seq_A,\{p\}}$.

The first result is a standard expressivity result showing that BL and CL for sequences without propositional variables are equally expressive. The proof will appear in a forthcoming paper. The second result is a parametric expressivity result. It shows that, without adjuncts, BL for sequences is as expressive as CL for sequences.

The third result shows that full BL is parametrically as expressive as CL without the $\blacktriangleright$ modality. The fourth result illustrates that the importance of CL-reasoning lies in the $\blacktriangleright$ modality, by showing that CL for sequences is parametrically more expressive than BL for sequences. Unlike the heap case, we are unable to give a direct proof of the parametric result. We give a proof of strong inexpressivity, and hence prove parametric inexpressivity.

Our first parametric expressivity result for sequences shows that, without adjuncts, CL-application can be specified by BL-composition. The proof shows that any context formula can be expressed as the disjunction of formulae of the form $P_1 \circ I \circ P_2$, and the application $(P_1 \circ I \circ P_2)(P)$ corresponds to $P_1 \circ P \circ P_2$.

Theorem 39 (Parametric Expressivity)**.**

$$\mathcal{B}L_{Seq_A,\mathcal{V}_D}(-\{\mathbin{-\!\bullet}, \mathbin{\bullet\!-}\}) = \mathcal{C}L_{Seq_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$$

Proof. Note that propositional variables are restricted to sort D, so the context formulae of $\mathcal{C}L_{Seq_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$ are

$$K \quad ::= \quad I \mid K \vee K \mid \neg K \mid \text{False} \mid K \circ P \mid P \circ K.$$

We define a canonical subset

$$\underline{K} \quad ::= \quad \underline{K} \vee \underline{K} \mid P \circ I \circ P$$

and show that

(1) every CL-formula K is equivalent to a canonical formula $\underline{K}$;
(2) every application formula $\underline{K}(P)$ is equivalent to the substitution formula $\underline{K}[P/I]$.

The result follows. Given an arbitrary CL-formula, first replace the context subformulae by canonical formulae, then replace the application subformulae by the equivalent substitution formulae. The resulting formula is a BL-formula equivalent to the original CL-formula.

To show (1), we define a translation tr from context formulae to the canonical formulae by:

$$\begin{array}{rcl}
tr(I) & \triangleq & 0 \circ I \circ 0 \\
tr(K_1 \vee K_2) & \triangleq & tr(K_1) \vee tr(K_2) \\
tr(\neg K) & \triangleq & \mathit{Not}(tr(K)) \\
tr(\text{False}) & \triangleq & \text{false} \circ I \circ \text{false} \\
tr(K \circ P) & \triangleq & \mathit{Add}_l(tr(K), P) \\
tr(P \circ K) & \triangleq & \mathit{Add}_r(P, tr(K))
\end{array}$$

where $\mathit{Not}(\underline{K})$, $\mathit{Add}_r(P, \underline{K})$ and $\mathit{Add}_l(\underline{K}, P)$ are defined below by induction on the structure of the canonical formulae.

Before defining *Not*, we define a function And on canonical formulae such that $\mathit{And}(\underline{K}_1, \underline{K}_2)$ is equivalent to $\underline{K}_1 \wedge \underline{K}_2$:

$$\begin{array}{rcl}
\mathit{And}(\underline{K}_1 \vee \underline{K}_2, \underline{K}_3) & \triangleq & \mathit{And}(\underline{K}_1, \underline{K}_3) \vee \mathit{And}(\underline{K}_2, \underline{K}_3) \\
\mathit{And}(P_1 \circ I \circ P_2, P_3 \circ I \circ P_4) & \triangleq & (P_1 \wedge P_3) \circ I \circ (P_2 \wedge P_4)
\end{array}$$

We now define function *Not*:

$$\begin{array}{rcl}
\mathit{Not}(\underline{K}_1 \vee \underline{K}_2) & \triangleq & \mathit{And}(\mathit{Not}(\underline{K}_1), \mathit{Not}(\underline{K}_2)) \\
\mathit{Not}(P_1 \circ I \circ P_2) & \triangleq & (\neg P_1) \circ I \circ \text{true} \vee \text{true} \circ I \circ (\neg P_2)
\end{array}$$

$\mathit{Add}_r(P, \underline{K})$ and $\mathit{Add}_l(\underline{K}, P)$ are defined similarly to And.

(2) is proved by induction on $\underline{K}$. For case $\underline{K}_1 \vee \underline{K}_2$, note that $(\underline{K}_1 \vee \underline{K}_2)(P)$ is equivalent to $\underline{K}_1(P) \vee \underline{K}_2(P)$ and, by the induction hypothesis, it is equivalent to $\underline{K}_1[P/I] \vee \underline{K}_2[P/I]$. Case $P_1 \circ I \circ P_2$ is immediate since $(P_1 \circ I \circ P_2)(P)$ is equivalent to $P_1 \circ P \circ P_2$. ☐

As an example, consider the CL-formula $(\neg I)(P)$ satisfied by any sequence having a strictly smaller subsequence satisfying P. We have $tr(\neg I) = \mathit{Not}(tr(I)) = \mathit{Not}(0 \circ I \circ 0) = (\neg 0) \circ I \circ \text{true} \vee \text{true} \circ I \circ (\neg 0)$. Therefore $(\neg I)(P)$ is equivalent to $(\neg 0) \circ P \circ \text{true} \vee \text{true} \circ P \circ (\neg 0)$, which is satisfied by any sequence containing a subsequence satisfying P composed with a nonempty sequence on at least one side.

Our next result shows that CL for sequences minus $\blacktriangleleft$ is parametrically as expressive as BL for sequences.

Theorem 40 (Parametric Expressivity)**.**

$$\mathcal{B}L_{Seq_A,\mathcal{V}_D} = \mathcal{C}L_{Seq_A,\mathcal{V}_D}(-\{\blacktriangleright\})$$

Proof. As in the proof of Theorem 39, we define a canonical subset:

$$\underline{K} \quad ::= \quad \underline{K} \vee \underline{K} \mid P \circ I \circ P$$

The only difference compared with Theorem 39 is that now data formulae may contain the adjoint formulae $K \blacktriangleleft P$. Given an arbitrary CL-formula, first replace the context subformulae by canonical formulae, as in (1) of Theorem 39, then replace the application subformulae by the equivalent substitution formulae as in (2). We must show how to eliminate the adjoint formulae $\underline{K} \blacktriangleleft P$, by induction on the structure of $\underline{K}$. When $\underline{K}$ is $P_1 \circ I \circ P_2$, then $(P_1 \circ I \circ P_2) \blacktriangleleft P$ is equivalent to $P_1 \mathbin{\bullet\!-} (P_2 \mathbin{-\!\bullet} P)$. When $\underline{K}$ is $\underline{K}_1 \vee \underline{K}_2$, then $(\underline{K}_1 \vee \underline{K}_2) \blacktriangleleft P$ is equivalent to $(\underline{K}_1 \blacktriangleleft P) \vee (\underline{K}_2 \blacktriangleleft P)$. ☐

Finally, we show that CL for sequences is parametrically more expressive than BL for sequences. This additional expressivity for CL must lie in the use of the $\blacktriangleright$ modality. Intuitively, BL can only add elements to either side of a given sequence, whilst $\blacktriangleright$ can add elements wherever the hole happens to be. We initially searched for a direct proof of this result, trying to identify a property analogous to the heap-reducing formulae of theorem 34 which captured this difference between adding elements to the side or the middle of sequences. BL-formulae can however affect the middle of the sequence, by using $\circ$ and $\mathbin{-\!\bullet}$ to remove the whole sequence and adding any desired sequence. We do not know if such a direct result is possible. Here, we prove our parametric expressivity result via a strong inexpressivity result using bisimulation.

Theorem 41 (Parametric Inexpressivity)**.** *Let A be an infinite alphabet. Then $\mathcal{B}L_{Seq_A,\{p\}} \subsetneq \mathcal{C}L_{Seq_A,\{p\}}$.*

Proof. We consider BL and CL for sequences containing formulae with at most one propositional variable p. Consider CL-formula $P \triangleq (0 \triangleright p)(a)$ for some $a \in A$. Expecting a contradiction, assume that P is equivalent to data formula P', and let $A' \subseteq A$ be the finite set of letters occurring in P'. Formula P says that p holds after removing an element a somewhere from the current sequence. By contrast, BL can only observe subsequences obtained by removing letters from either side, not from the middle. With BL, we can only compare adjacent pairs of elements. With CL, we can compare arbitrary pairs of elements. We must find an interpretation function σ, and a bisimulation relation $\sim$ which is compatible with σ and captures this intuitive difference in expressivity.

We choose an interpretation σ which states that the interpretation of p is the set of all sequences with equal elements. To express this formally, we first introduce some notation. Let α, β denote sequences, let α_i denote the i-th element of sequence α, and let $|\alpha|$ denote the length of the sequence. We define σ by

$$\sigma(p) \triangleq \{\alpha \mid \forall i \in 1..|\alpha| - 1.\ \alpha_i = \alpha_{i+1}\}$$

In addition, we define a bisimulation relation which observes elements in the set A' and equality of adjacent elements:

$$\begin{array}{lll}
\alpha \sim \beta & \text{iff} & \exists n.\ |\alpha| = |\beta| = n\ \wedge \\
 & & \forall i \in 1 \ldots n-1.\ \alpha_i = \alpha_{i+1} \Leftrightarrow \beta_i = \beta_{i+1}\ \wedge \\
 & & \forall i \in 1 \ldots n, a' \in A'.\ \alpha_i = a' \Leftrightarrow \beta_i = a'
\end{array}$$

Clearly $\sim$ is compatible with σ. Assume for the moment that it is indeed a BL-bisimulation. With this assumption, we prove the inexpressivity result we seek. Consider two sequences $\alpha_1 = a' \cdot a \cdot a'$ and $\alpha_2 = a' \cdot a \cdot a''$, where $a' \neq a''$ are not in A'. Observe that $\alpha_1 \sim \alpha_2$, as adjacent letters are distinct in both sequences, $\sigma, \alpha_1 \models (0 \rhd p)(a)$ but $\sigma, \alpha_2 \not\models (0 \rhd p)(a)$. Assuming $\sim$ is a bisimulation for $\mathcal{B}L_{Seq_{A'},\{p\}}$, Proposition 33 implies that $\sigma, \alpha_1 \models_{BL} P'$ iff $\sigma, \alpha_2 \models_{BL} P'$ for all BL-formulae P'. We have therefore proved that there cannot be a BL-formula P' which is equivalent to P.

Finally, we must show that $\sim$ is indeed a bisimulation for all the modalities of $\mathcal{B}L_{Seq_{A'},\{p\}}$. We only look at the $\circ$ and $-\!\!\bullet$ modalities; the other cases are analogous or trivial. For the $\circ$ modality, assume $\alpha \cdot \beta = \gamma$ and $\gamma \sim \gamma'$. We must show that $\exists \alpha', \beta'$ such that $\alpha' \cdot \beta' = \gamma'$ and $\alpha \sim \alpha'$ and $\beta \sim \beta'$. Choose α', β' as the unique splitting of γ' such that $|\alpha'| = |\alpha|$ and $|\beta'| = |\beta|$. Clearly $\alpha \sim \alpha'$ and $\beta \sim \beta'$, since adjacent elements in α' and β' are also adjacent in γ'. For the $-\!\!\bullet$ modality, assume $\alpha \cdot \beta = \gamma$ and $\alpha \sim \alpha'$. We need to show that $\exists \beta', \gamma'$ such that $\alpha' \cdot \beta' = \gamma'$ and $\beta \sim \beta'$ and $\gamma \sim \gamma'$. Since α and α' might end in different letters, we must construct a β' such that its first element relates to the last element of α' in the same way as the first element of β relates to the last element of α. Let $f : A \to A$ be a bijection such that $f(\alpha_i) = \alpha'_i$. Such an f exists since $\alpha \sim \alpha'$. We define β' as the unique sequence such that $|\beta'| = |\beta|$, $\beta'_i = f(\beta_i)$ and $\gamma' \triangleq \alpha' \cdot \beta'$. It is easy to see that $\beta \sim \beta'$ and $\gamma \sim \gamma'$. □

5.3 Trees

We have the following results for trees for infinite alphabet A:

- $\mathcal{B}L_{Tree_A,\mathcal{V}_D}(-\{-\!\!\bullet, \widehat{\mu}, \widehat{\diamond}\}) = \mathcal{C}L_{Tree_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$
- $\mathcal{B}L_{Tree_A,\mathcal{V}_D} = \mathcal{C}L_{Tree_A,\mathcal{V}_D}(-\{\blacktriangleright\})$
- $\mathcal{B}L_{Tree_A,\{p\}} \subsetneq \mathcal{C}L_{Tree_A,\{p\}}$.

The first result states that CL and BL for trees without adjoints have the same parametric expressive power. The second result states that adding the $\blacktriangleleft$ modality to CL gives the same expressivity as BL. This formalises our intuition that the $\blacktriangleleft$ modality is a compact way to express the adjoints of AL (plus $\widehat{\diamond}$). The third result is a parametric inexpressivity result, showing that CL for trees is parametrically more expressive than BL for trees. This result illustrates that $\blacktriangleright$ is the key modality for giving CL its additional expressive power. In [3], we observed that it was important for expressing the weakest preconditions of update commands. Our inexpressivity result formalises this intuition, which we had previously motivated by example.

Our first parametric expressivity result is that BL and CL for trees without adjoints are equally expressive. The proof shows that context formulae can be reduced to a canonical form, allowing context application to be eliminated by a form of syntactic substitution.

Theorem 42 (Parametric Expressivity)**.**

$$\mathcal{B}L_{Tree_A,\mathcal{V}_D}(-\{-\!\!\bullet, \widehat{\mu}, \widehat{\diamond}\}) = \mathcal{C}L_{Tree_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$$

Proof. Note that propositional variables are restricted to sort D, so the context formulae of $\mathcal{C}L_{Tree_A,\mathcal{V}_D}(-\{\blacktriangleright, \blacktriangleleft\})$ are:

$$K \quad ::= \quad I \mid K \vee K \mid \neg K \mid \text{False} \mid \mu[K] \mid P \circ K$$

We define a canonical subset, similar to that given in Theorem 39:

$$\begin{array}{rcl} \underline{K} & ::= & \underline{K} \vee \underline{K} \mid \text{True} \mid \text{False} \mid P \circ I \mid P \circ \eta[\underline{K}] \\ \eta & ::= & \{a_1, \ldots, a_n\} \mid \{a_1, \ldots, a_n\}^{\perp} \end{array}$$

The composition formulae analyse whether the hole is at the top level or under a node label. The formulae $\{a_1, \ldots, a_n\}[\underline{K}]$ and $\{a_1, \ldots, a_n\}^{\perp}[\underline{K}]$ are syntactic sugar for $\bigvee_i a_i[\underline{K}]$ and $\bigwedge_i a_i^{\perp}[\underline{K}]$ respectively. Just as for Theorem 39, it is enough to show the following results:

(1) every CL-formula K is equivalent to a canonical formula $\underline{K}$;
(2) every application $\underline{K}(P)$ is equivalent to the substitution formula $\underline{K}[P/I, \diamond P/\text{True}]$.

To prove (1), we define a translation tr from context formulae to the canonical subset by:

$$\begin{array}{rcl} tr(I) & \triangleq & 0 \circ I \\ tr(K_1 \vee K_2) & \triangleq & tr(K_1) \vee tr(K_2) \\ tr(\neg K) & \triangleq & \mathit{Not}(tr(K)) \\ tr(\text{False}) & \triangleq & \text{False} \\ tr(\mu[K]) & \triangleq & 0 \circ \mu[tr(K)] \\ tr(P \circ K) & \triangleq & \mathit{Add}(P, tr(K)) \end{array}$$

where $\mathit{Not}(\underline{K})$ and $\mathit{Add}(P, \underline{K})$ are defined by induction on the structure of the canonical formulae.

Before defining *Not*, we define a function And on canonical formulae such that $\mathit{And}(\underline{K}_1, \underline{K}_2)$ is equivalent to $\underline{K}_1 \wedge \underline{K}_2$:

$$\begin{array}{rcl} \mathit{And}(\underline{K}_1 \vee \underline{K}_2, \underline{K}_3) & \triangleq & \mathit{And}(\underline{K}_1, \underline{K}_3) \vee \mathit{And}(\underline{K}_2, \underline{K}_3) \\ \mathit{And}(\text{True}, \underline{K}) & \triangleq & \underline{K} \\ \mathit{And}(\text{False}, \underline{K}) & \triangleq & \text{False} \\ \mathit{And}(P_1 \circ I, P_2 \circ I) & \triangleq & (P_1 \wedge P_2) \circ I \\ \mathit{And}(P_1 \circ I, P_2 \circ \eta[\underline{K}]) & \triangleq & \text{False} \\ \mathit{And}(P_1 \circ \eta_1[\underline{K}_1], P_2 \circ \eta_2[\underline{K}_2]) & \triangleq & \\ & & (P_1 \wedge P_2) \circ (\eta_1 \wedge \eta_2)[\mathit{And}(\underline{K}_1, \underline{K}_2)] \end{array}$$

where, for sets S_i, we define $S_1 \wedge S_2 = S_1 \cap S_2$, $S_1 \wedge S_2^{\perp} = S_1 - S_2$, and $S_1^{\perp} \wedge S_2^{\perp} = (S_1 \cup S_2)^{\perp}$. We now define the function *Not* such that $\mathit{Not}(\underline{K})$ is equivalent to $\neg(\underline{K})$:

$$\begin{array}{rcl} \mathit{Not}(\underline{K}_1 \vee \underline{K}_2) & \triangleq & \mathit{And}(\mathit{Not}(\underline{K}_1), \mathit{Not}(\underline{K}_2)) \\ \mathit{Not}(\text{True}) & \triangleq & \text{False} \\ \mathit{Not}(\text{False}) & \triangleq & \text{True} \\ \mathit{Not}(P \circ I) & \triangleq & (\neg P) \circ I \vee \text{true} \circ \emptyset^{\perp}[\text{True}] \\ \mathit{Not}(P \circ \eta[\underline{K}]) & \triangleq & \text{true} \circ I \vee \text{true} \circ \eta^{\perp}[\text{True}] \vee \\ & & (\neg P) \circ \eta[\text{True}] \vee \text{true} \circ \eta[\mathit{Not}(\underline{K})] \end{array}$$

We finally define Add such that $\mathit{Add}(P, \underline{K})$ is equivalent to $P \circ \underline{K}$:

$$\begin{array}{rcl} \mathit{Add}(P, \underline{K}_1 \vee \underline{K}_2) & \triangleq & \mathit{Add}(P, \underline{K}_1) \vee \mathit{Add}(P, \underline{K}_2) \\ \mathit{Add}(P, \text{True}) & \triangleq & (P \circ \text{true}) \circ I \vee (P \circ \text{true}) \circ \emptyset^{\perp}[\text{True}] \\ \mathit{Add}(P, \text{False}) & \triangleq & \text{False} \\ \mathit{Add}(P_1, P_2 \circ I) & \triangleq & (P_1 \circ P_2) \circ I \\ \mathit{Add}(P_1, P_2 \circ \eta[\underline{K}]) & \triangleq & (P_1 \circ P_2) \circ \eta[\underline{K}] \end{array}$$

To prove (2), we proceed by induction on $\underline{K}$. For case $\underline{K}_1 \vee \underline{K}_2$, note that $(\underline{K}_1 \vee \underline{K}_2)(P)$ is equivalent to $\underline{K}_1(P) \vee \underline{K}_2(P)$ and hence, by induction, is equivalent to $\underline{K}_1[P/I, \diamond P/\text{True}] \vee \underline{K}_2[P/I, \diamond P/\text{True}]$. For case True, observe that $\text{True}(P)$ is equivalent to $\diamond P$. Case False is immediate. Case $P_1 \circ I$ is also immediate since $(P_1 \circ I)(P_2)$ is equivalent to $P_1 \circ P_2$. For case $P_1 \circ \eta[\underline{K}]$, first observe that $(P_1 \circ \eta[\underline{K}])(P_2)$ is equivalent to $P_1 \circ \eta[\underline{K}(P_2)]$ which, by the induction hypothesis, is equivalent to $(P_1 \circ \eta[\underline{K}[P_2/I, \diamond P/\text{True}]])$. □

Consider CL formula $(\neg I)(P)$, satisfied by any tree having a strictly smaller subtree satisfying P. We have $tr(\neg I) = \mathit{Not}(tr(I)) = \mathit{Not}(0 \circ I) = (\neg 0) \circ I \vee \text{true} \circ \emptyset^{\perp}[\text{True}]$. Therefore $(\neg I)(P)$ is equivalent to $(\neg 0) \circ P \vee \text{true} \circ \emptyset^{\perp}[\diamond P]$, satisfied by any tree with either a subtree satisfying P at the top level composed with a non-empty tree, or a subtree satisfying P under a tree node. Our second parametric expressivity result for trees shows that CL for trees minus $\blacktriangleleft$ corresponds to BL.

Theorem 43 (Parametric Expressivity)**.**

$$\mathcal{B}L_{Tree_A,\mathcal{V}_D} = \mathcal{C}L_{Tree_A,\mathcal{V}_D}(-\{\blacktriangleright\})$$

Proof. The proof extends the proof of Theorem 42, just as the proof of Theorem 39 extends that of Theorem 40. We use the same canonical forms as in Theorem 42, and just need to show that $\underline{K} \blacktriangleleft P$ can be eliminated by induction on $\underline{K}$. For case $\underline{K}_1 \vee \underline{K}_2$, we have $(\underline{K}_1 \vee \underline{K}_2) \blacktriangleleft P$ equivalent to $(\underline{K}_1 \blacktriangleleft P) \vee (\underline{K}_2 \blacktriangleleft P)$. For case True, we have True $\blacktriangleleft P$ equivalent to $\widehat{\diamond} P$. Case False is immediate, since False $\blacktriangleleft P$ is equivalent to false. Case $P \circ I$ is also immediate, since $(P_1 \circ I) \blacktriangleleft P$ is equivalent to $P_1 \multimap\!\bullet\, P$. For case $P \circ \eta[\underline{K}]$, we have $(P_1 \circ \eta[\underline{K}]) \blacktriangleleft P$ equivalent to $\underline{K} \blacktriangleleft (\widehat{\eta}(P_1 \multimap\!\bullet\, P))$ where, if $S = \{a_1, \ldots, a_n\}$, then $\widehat{S}(P)$ is $\widehat{a_1}(P) \vee \cdots \vee \widehat{a_n}(P)$ and $\widehat{S^\perp}(P)$ is $\widehat{a_1^\perp}(P) \wedge \cdots \wedge \widehat{a_n^\perp}(P)$. The result follows by the induction hypothesis. □

Finally, we show that CL for trees is parametrically more expressive than BL. We are unable to give a direct proof. We instead show a strong inexpressivity result based on interpretation σ_{eq}, which interprets p as the property that all the labels in the tree are equal. We show that formula $(0 \triangleright p)(a[\text{true}])$, corresponding to the weakest precondition of deleting a subtree with root label a, is not expressible in BL. Intuitively, the result holds since CL can remove a subtree at an arbitrary position, while BL can only split trees at the top level using $\circ$ or under a fixed number of edges using μ.

Theorem 44 (Parametric Inexpressivity)**.**

$$\mathcal{B}L_{Tree_A,\{p\}} \subsetneq \mathcal{C}L_{Tree_A,\{p\}}$$

Proof. The structure of the proof is identical to that of Theorem 41. Consider the CL formula $P \triangleq (0 \triangleright p)(a[\text{true}])$, which says that we can remove a subtree with root a from the current tree and the result satisfies p. Intuitively, BL can remove subtrees from the top level of the current tree using $\circ$, but cannot remove them from arbitrary positions. As in Theorem 41 for sequences, we restrict our attention to BL-formulae with a single propositional variable p and node labels from a finite set $A' \subseteq A$. We define the interpretation as $t \in \sigma(p)$ iff all the labels in tree t are equal.

Let $\sim$ be the unique relation such that:

$$\begin{array}{l}
t \sim t' \quad \text{iff} \quad t \in \sigma(p) \Leftrightarrow t' \in \sigma(p) \text{ and} \\
\quad \text{if } t \equiv a_1[t_1] \text{ then } \exists a_1', t_1' \text{ such that} \\
\qquad t' \equiv a_1'[t_1'] \text{ and } t_1 \sim t_1' \text{ and} \\
\qquad a_1 \in A' \text{ or } a_1' \in A' \text{ implies } a_1 = a_1' \text{ and} \\
\quad \text{if } t \equiv t_1 | t_2 \text{ then } \exists t_1', t_2' \text{ such that} \\
\qquad t' \equiv t_1' | t_2' \text{ and } t_1 \sim t_1' \text{ and } t_2 \sim t_2'
\end{array}$$

It is easy to see that $\sim$ is compatible with σ, and we will show that it is a BL-bisimulation. Let b, c be distinct labels not in $A' \cup \{a\}$, and define $t_1 \triangleq b[a[0]|b[0]]$ and $t_2 \triangleq b[a[0]|c[0]]$. We have $t_1 \sim t_2$ and $\sigma, t_1 \models P$ but $\sigma, t_2 \not\models P$, hence the result.

We must show that $\sim$ is a bisimulation for all the modalities of $\mathcal{B}L_{Tree_{A'},\{p\}}$. The modalities $0, \circ, \mu$ are immediate from the definition of $\sim$. Modalities $\multimap\!\bullet, \widehat{\mu}, \widehat{\diamond}$ follow from the fact that $\sim$ is a congruence: that is, if $t \sim t'$ then $c(t) \sim c(t')$ for all tree contexts c. For the $\diamond$ modality, we need to show that, if $c(t_1) \equiv t$ and $t \sim t'$, then there exist c', t_1' such that $t' \equiv c'(t_1')$ and $t_1 \sim t_1'$. The proof is straightforward, by induction on the size of c. □

6. Concluding Remarks

We have shown how to present CL and BL as ML, interpreting the structural connectives as modalities satisfying a set of well-behaved ML-axioms. We have given two applications of the general theory of ML: we have proved completeness results for CL and BL using a general theorem about ML due to Sahlqvist, and inexpressivity results using the standard ML-bisimulation technique.

Our parametric inexpressivity results for arbitrary formulae contrast with Lozes' expressivity results on closed formulae. We prove that SL is parametrically more expressive than PL for heaps, whereas Lozes shows that these logics have the same expressivity on closed formulae. We also prove that CL for trees is parametrically more expressive than AL, whereas Lozes shows that they have the same expressivity on closed formulae. Our definition of parametric expressivity corresponds to that studied in the ML-literature, and corroborates our intuition that the structural connectives of CL and BL are essential for our local Hoare reasoning. Lozes' style of expressivity result is not typically explored in the ML-literature. It is interesting for our application of structured data: for example, we have shown that CL for sequences corresponds to the $*$-free regular languages on closed formulae. The structural connectives of CL and BL give rise to several examples of logics which are parametrically more expressive, but which have the same expressivity on closed formulae. We currently do not know of other examples of ML where this is the case, except for simple examples such as S4 and S5 where all closed formulae correspond to either true or false.

We are only at the beginning of studying expressivity results for CL: for example, two natural extensions involve higher-order quantification and first-order quantification. The parametric expressivity results in this paper are based on formulae with propositional variables. Our results say nothing about the expressivity of higher-order SL over higher-order logic. Also, our results compare CL and BL without first-order quantification, whereas their applications to analysing trees and heaps usually involve quantification over node labels and heap addresses. Dawar, Gardner and Ghelli (with thanks to Yang) have shown that adjunct-elimination in AL with quantification is not possible [6], strengthening a previous result by Lozes [9]. Despite this inexpressivity result, it still makes sense to ask for parametric inexpressivity results about particular formulae, to pin down exactly why full SL seems to be more appropriate for modular reasoning about programs than first-order logic.

References

[1] J. Berdine, C. Calcagno, and P.W. O'Hearn. Smallfoot: Modular automatic assertion checking with separation logic. In *Proceedings of FMCO'05*, volume 4111 of *LNCS*, 2006.

[2] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.

[3] C. Calcagno, P. Gardner, and U. Zarfaty. Context logic and tree update. In *POPL*, 2005.

[4] L. Cardelli and G. Ghelli. TQL: A query language for semistructured data based on the ambient logic. To appear in MSCS.

[5] L. Cardelli and A. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *POPL*, 2000.

[6] A. Dawar, P. Gardner, and G. Ghelli. Adjunct elimination using Enrenfeuch's games. In *FSTTCS*, 2004.

[7] H.Hosoya and B. Pierce. Xduce: A typed xml processing language. *ACM Transactions on Internet Technology*, 3:117–148, 2003.

[8] S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *POPL*, 2001.

[9] Etienne Lozes. Elimination of spatial connectives in static spatial logics. In TCS 330(3), 2005.

[10] D. Pym, P. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 315(1), 2004.

[11] J.C. Reynolds. Separation logic: a logic for shared mutable data structures. Invited Paper, LICS'02, 2002.

[12] H. Yang. *Local Reasoning for Stateful Programs*. Ph.D. thesis, University of Illinois, Urbana-Champaign, Illinois, USA, 2001.

[13] H. Yang and P. O'Hearn. A semantic basis for local reasoning. FOSSACS, 2002.