Local Reasoning about Data Update

Cristiano Calcagno, Philippa Gardner and Uri Zarfaty

Imperial College London

References

Context Logic and Tree Update, POPL’05

Context Logic as Modal Logic: Completeness and Parametric Inexpressivity, submitted

Local Reasoning about Data Update, journal paper, submitted

Supported by EPSRC, Microsoft and Royal Academy of Engineering.
Reasoning about Data Update

Examples of data update

heap update, information on hard discs, XML update, term rewriting

Hoare Reasoning for Heap Update

Hoare’s original work based on First-order Logic

O’Hearn, Reynolds, Yang’s work on local reasoning using Separation Logic (SL), an application of Bunched Logic to heaps (O’Hearn, Pym).

Hoare Reasoning for Data Update

Hardly studied for other forms of data update

No unified account
Reasoning about Trees

Reasoning about Static Trees

Ambient Logic (AL) Cardelli, Gordon

Reasoning about web data Cardelli, Gardner, Ghelli

Similar reasoning to Separation Logic

Local Hoare Reasoning for Tree Update

Not possible using Ambient Logic

Possible using Context Logic (CL) Calcagno, Gardner, Zarfaty
Summary

- Context Logic
- Application to Trees
- Local Reasoning about Tree Update
- Inexpressivity Result for Ambient Logic
Reasoning about Trees

Ambient Logic

\[
P 
\rightarrow n[P]
\]

\[
P 
\rightarrow P \circ Q
\]

\[P \text{ and } Q \text{ are data formulae}\]

Context Logic for Trees

\[
K 
\rightarrow K(P)
\]

\[P \text{ is a data formula and } K \text{ a context formula}\]
**CL\(_0\)**-Formulae

### Data Formulae

\[ P ::= 0 \quad \text{zero formula} \]

\[ K(P) \quad \text{application} \]

\[ K \bowtie P \quad \text{data application adjoint} \]

\[ P \lor P \mid \neg P \mid \text{false} \quad \text{boolean additive formulae} \]

### Context Formulae

\[ K ::= I \quad \text{identity formula} \]

\[ P \rhd P \quad \text{context application adjoint} \]

\[ K \lor K \mid \neg K \mid \text{False} \quad \text{boolean additive formulae} \]
Reasoning about Trees

Application

Data Adjoint

Context Adjoint
A CL₀-model \( \mathcal{M} \) is a tuple \((\mathcal{D}, \mathcal{C}, \text{ap}, \mathbf{I}, 0)\) consisting of

1. data set \( \mathcal{D} \) and context set \( \mathcal{C} \)
2. application \( \text{ap} \subseteq (\mathcal{C} \times \mathcal{D}) \times \mathcal{D} \): we write \( \text{ap}(c, d_1) = d_2 \)
3. the left identity \( \mathbf{I} \subseteq \mathcal{C} \) to application:
   - \( \forall d \in \mathcal{D} \), \( \exists i \in \mathbf{I}, d' \in \mathcal{D} \). \( \text{ap}(i, d) = d' \);
   - \( \forall d, d' \in \mathcal{D}, \forall i \in \mathbf{I}. \) \( \text{ap}(i, d) = d' \) implies \( d = d' \);
4. the projection \( p : \mathcal{C} \rightarrow \mathcal{D} \) defined by

\[
p(c) = d \iff \exists o \in 0. \text{ap}(c, o) = d
\]

st. \( p \) is a total surjective function and \( \forall c, o. \) \( p(c) = o \Rightarrow c \in \mathbf{I} \).
**Example CL₀-Models**

- \( \text{Mon}_D = (D, D, \cdot, \{e\}, \{e\}) \), with (partial) monoid \( \cdot \) and unit \( e \).
- \( \text{Term}_\Sigma = (T_\Sigma, C_\Sigma, \text{ap}, \{\_\}) \), with \( T_\Sigma \) the set of terms, \( C_\Sigma \) the contexts, \( \text{ap} \) context application and \( \_ \) the empty context.
- sequences, trees, multisets, heaps
- \( \text{Rel}_D = (D, \mathcal{P}(D \times D), \text{ap}, \{i\}) \), with \( \text{ap} \) relational application and \( i \) the identity relation, is a CL-model.
- \( \text{Step} = (\mathbb{N}, \{0, 1\}, +, \{0\}) \) is a CL-model.
- \( \mathcal{M}_1 + \mathcal{M}_2 = (D_1 \cup D_2, C_1 \cup C_2, \text{ap}_1 \cup \text{ap}_2, I_1 \cup I_2, 0_1 \cup 0_2) \)
  for CL₀-models \( \mathcal{M}_i = (D_i, C_i, \text{ap}_i, I_i, 0_i) \), \( i = 1, 2 \).
For $\mathcal{CL}_0$-model $\mathcal{M} = (\mathcal{D}, \mathcal{C}, \text{ap}, \mathbf{I}, \mathbf{0})$, the $\mathcal{CL}_0$-satisfaction relation $\models$ consists of two relations $\mathcal{M}, d \models P$ and $\mathcal{M}, c \models K$ given by:

1. $\mathcal{M}, d \models \mathbf{0}$ iff $d \in \mathbf{0}$
2. $\mathcal{M}, d \models K(P)$ iff $\exists c, d'. \text{ap}(c, d') = d \land \mathcal{M}, c \models K \land \mathcal{M}, d' \models P$
3. $\mathcal{M}, d \models K \triangleleft P$ iff $\forall c, d'. \mathcal{M}, c \models K \land \text{ap}(c, d) = d' \Rightarrow \mathcal{M}, d' \models P$
4. $\mathcal{M}, c \models \mathbf{I}$ iff $c \in \mathbf{I}$
5. $\mathcal{M}, c \models P_1 \triangleright P_2$ iff $\forall d, d'. \mathcal{M}, d \models P_1 \land \text{ap}(c, d) = d' \Rightarrow \mathcal{M}, d' \models P_2$

The boolean additive cases are standard.
**Derived CL-Data Formulae**

Standard derived formulae for the additive connectives.

- $\Diamond P \triangleq \text{True}(P)$ somewhere property $P$ holds;
  
  $\models P \Rightarrow \Diamond P$ and $\not\models \Diamond \Diamond P \Rightarrow \Diamond P$ (holds with context composition)

- $K \blacktriangleleft P_2 \triangleq \neg(K \bigtriangledown \neg P_2)$ there exists a context satisfying property $K$ such that, when the given data element is put in the hole, the resulting data satisfies $P_2$.

- $P_1 \blacktriangleright P_2 \triangleq \neg(P_1 \blacktriangledown \neg P_2)$ 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 $P_2$. 

1 $\triangleq \neg 0 \land \neg(\neg I)(\neg 0)$ size one

$P_1 \ast P_2 \triangleq (0 \triangleright P_1)(P_2)$ data can be split into subdata satisfying $P_2$ and a context satisfying $P_1$ when a zero is put in the hole.

$P_1 \not\triangleright P_3 \triangleq (0 \triangleright P_1) \triangleleft P_3$ whenever a context applied to a zero satisfies $P_1$, then the context applied to the given data satisfies $P_3$.

$P_2 \not\triangleright P_3 \triangleq \neg((\neg(P_2 \triangleright P_3)(0)))$ whenever data satisfying $P_2$ replaces empty subdata of the given data, then result satisfies $P_3$. 

Proof theory and Completeness

Hilbert-style proof theory using $\triangleright$, $\triangleleft$

Modal-logic presentation using $\Rightarrow$, $\Leftarrow$ and specific CL$_0$-axioms

The CL$_0$-axioms are well-behaved (very simple Salqvist formulae)

Salqvist’s theorem implies completeness
Summary

- Context Logic
- Application to Trees
- Local Reasoning about Tree Update
- Inexpressivity Result for Ambient Logic
Application to Trees

Tree Model

trees $t ::= 0$ empty tree

$n[t]$ tree with top node $n \in N$

$t \circ t$ horizontal composition

contexts $c ::= _- | n[c] | c \circ t | t \circ c, \quad n \in N$

Equality states that $\circ$ is associative and commutative with unit 0.

We choose the node labels to be unique.
Specific $\text{CL}_0$-formulae for trees

- **data formulae**: $P ::= \ldots | n[P] | P \circ P$, $n \in N$
- **context formulae**: $K ::= \ldots | n[K] | K \circ P | P \circ K$, $n \in N$

**Satisfaction Relation**

- $\text{Tree}_N, c \models n[K]$ iff $\exists c'. c = n[c'] \land \text{Tree}_N, c' \models K$
- $\text{Tree}_N, c \models K \circ P$ iff $\exists c', d. c = c' \circ d \land \text{Tree}_N, c' \models K \land \text{Tree}_N, d \models P$

**Adjoints**

$\hat{n}[P] \triangleq n[I] \triangleleft P$ and $P_1 \rightarrow P_2 \triangleq (P_1 \circ I) \triangleleft P_2$. 
Derived Tree Formulae

- $n[0]$, the tree $n[0]; n[\text{true}]$, a tree with root node labelled $n$
- $\Diamond n[\text{true}]$, a tree containing a node $n$
- $n[\text{true}] \circ n[\text{true}], n[\text{true}] \ast n[\text{true}]$, unsatisfied as $n$ unique
- $m[\text{true}] \circ n[\text{true}]$, two trees with top nodes $m$ and $n$
- $m[\text{true}] \ast n[\text{true}]$, either two trees with top nodes $m$ and $n$, or one tree with top node $m$ and a subtree with top node $n$
- $(0 \triangleright P)(n[\text{true}])$ and $P \ast n[\text{true}]$, a tree containing $n$ that satisfies $P$ if the subtree at $n$ is replaced by $0$.
- $(m[\text{true}] \triangleright P)(n[\text{true}])$, a tree containing $n$ that satisfies $P$ whenever the subtree at $n$ is replaced by a tree with top node $m$
Summary

- Context Logic
- Application to Trees
- Local Reasoning about Tree Update
- Inexpressivity Result for Ambient Logic
Local Reasoning about Data Update

Update commands tend to operate in a local way, by accessing a small part of the data called the footprint O’Hearn, Reynolds, Yang

Local Hoare reasoning reflects this locality intuition:

small axioms specify the behaviour of commands on their footprints;

the frame rule automatically infers that the rest of the data (the context) remains unchanged.

CL-reasoning is ideally suited to this style of reasoning.

Here we focus on tree update. For our tree commands to be local, the node values must be unique.
**Tree Update**

Node variables $n, m, \ldots$

Tree variables $x, y, \ldots$

Stores map variables to values, denoted by $s$

Specific CL$_0$-formulae

- **data formulae**
  
  $P ::= \ldots | x$  
  $\ldots | \exists n. P | \exists x. P$  
  $x$ tree variable
  
  quantification

- **context formulae**
  
  $K ::= \ldots | n[K] | K \circ P$  
  $n$ node variable
  
  quantification

Satisfaction relation $\text{Tree}_N, s, t \models P$ and $\text{Tree}_N, s, c \models K$
Commands for Tree Update

\[ C ::= \quad n := n' \mid x := x' \quad \text{variable assignment} \]

\[ C_{up}(n) \quad \text{update at location} \ n \]

\[ C ; C \quad \text{sequencing} \]

\[ C_{up}(n) ::= \quad [n]_T := 0 \quad [n]_{SF} := 0 \quad \text{dispose} \]

\[ [n]_T *:= x \quad [n]_{SF} *:= x \quad \text{append} \]

\[ x := [n]_T \quad x := [n]_{SF} \quad \text{lookup} \]

\[ n' ::= \text{new} \ [n]_T \quad n' ::= \text{new} \ [n]_{SF} \quad \text{new} \]

\[ \text{free}(C) = \text{set of variables in} \ C; \ \text{mod}(C) \text{ given by the red variables.} \]

All the commands are local.
Move Example

\[ \text{move}(n, n') \triangleq x := [n]_T ; \]
\[ [n]_T := 0 ; \]
\[ [n']_{SF} *:= x \]

Three cases
Local Hoare Reasoning

Hoare triples \( \{P\} \vdash \{Q\} \) partial, fault-avoiding interpretation

Frame Rule

\[
\frac{\{P\} \vdash \{Q\}}{\{K(P)\} \vdash \{K(Q)\}} \mod(C) \cap \text{free}(K) = \emptyset
\]

Plus consequence, auxiliary variable elimination and sequencing.

Soundness The rules are sound if the commands are local.
Sample Small Axioms

\[
\begin{align*}
\{ n[\text{true}] \} & \quad [n]_T := 0 \quad \{0\} \\
\{ n[y] \} & \quad n' := \text{new} [n]_T \quad \{ n[y] \circ n'[0] \}
\end{align*}
\]
Weakest Preconditions

\[
\begin{align*}
\{(0 \triangleright P)(n[\text{true}])\} & \quad [n]_T := 0 \quad \{P\} \\
\exists y. \forall n'. ((n[y] \circ n'[0]) \triangleright P)(n[y]) & \quad n' := \text{new } [n]_T \quad \{P\}
\end{align*}
\]

where \( y \notin \text{free}(P) \)
Derivations

\[
\begin{align*}
\{n[\text{true}]\} & \quad [n]_T := 0 \{0\} & \text{FRAME} \\
\{(0 \triangleright P)(n[\text{true}])\} & \quad [n]_T := 0 \{(0 \triangleright P)(0)\} & \text{CONS} \\
\{(0 \triangleright P)(n[\text{true}])\} & \quad [n]_T := 0 \{P\} & \text{CONS} \\
\end{align*}
\]

\[
\begin{align*}
\{n[y]\} & \quad n' := \text{new} [n]_T \{n[y] \circ n'[0]\} & \text{FRAME} \\
\{K(n[y])\} & \quad n' := \text{new} [n]_T \{K(n[y] \circ n'[0])\} & \text{CONS/VARS} \\
\exists y.K(n[y]) & \quad n' := \text{new} [n]_T \{P\} \\
K & = (\forall n'. (n[y] \circ n'[0]) \triangleright P) \text{ and } y \notin \text{free}(P)
\end{align*}
\]

Uses structural modus ponens \((P \triangleright P')(P) \vdash P' \land \text{True}(P)\).
Reasoning about Move

Safety property for \text{move}(n, n')

\[
\{(0 \triangleright \text{True}(n'[\text{true}]))(n[\text{true}])\}
\]

\[
x := [n]_T
\]

\[
\{(0 \triangleright \text{True}(n'[\text{true}]))(n[\text{true}])\}
\]

\[
[n]_T := 0
\]

\[
\{\text{True}(n'[\text{true}])\}
\]

\[
[n']_{SF} * := x
\]

\[
\{\text{true}\}
\]
Reasoning about Move

Specification of move\( (n, n') \)

\[
\{ (0 \triangleright \text{True}(n'[x]))(n[y]) \} \\
\text{move}(n, n') \\
\{ \text{True}(n'[x \circ n[y]]) \}
\]
Other Examples of Update

$CL_0$-reasoning about heap update is exactly analogous to
SL-reasoning about heap update.

CL-reasoning about term rewriting possible, not possible using SL.

CL-reasoning about tree update, heap update and term rewriting is
strikingly similar.

Challenge unified Hoare reasoning about data update
Summary

- Context Logic
- Application to Trees
- Local Reasoning about Tree Update
- Inexpressivity Result for Ambient Logic
Expressivity for AL and CL

Assume no quantification

Result AL is as expressive as CL for trees minus $\triangleright$

Requires $\Diamond P \; n_{\perp}[P] \; n_{\perp}[P]$  

Result AL is as expressive as AL minus structural adjoints

Lozes, then Dawar, Gardner, Ghelli

Result SL is as expressive as SL minus structural adjoints Lozes

Conjecture CL is as expressive as CL minus $\triangleright$

Requires context composition, probably requires multi-holed contexts

CL-reasoning is still essential for reasoning about tree update.
Result

AL is not as expressive as $CL_0$ for trees using parametric expressivity

Intuition

CL-formula $(0 ▶ m_1[m_2[0]])(n[true])$

AL-formula
$m_1[m_2[n[true]]] \vee (m_1[m_2[0] \circ n[true]]) \vee (m_1[m_2[0]] \circ n[true])$

CL-formula $(0 ▶ \Diamond m_2[true])(n[true])$

AL-formula $\Diamond m_2[true] \land \Diamond n[\neg \Diamond m_2[true]]$

The CL-reasoning is more uniform than the AL-reasoning.
**Parametric inexpressivity**

Result

AL is not as expressive as CL\(_0\) for trees using parametric expressivity

Proof For simplicity, we assume that the node labels are not unique.

Consider formula \((0 \triangleright p)(n(\text{true}))\), where \(p\) is a propositional variable. This formula describes a function from sets of trees to sets of trees. We prove that it is not expressible in AL.

Let \(p\) denote the set of trees whose node labels are equal.

Consider \(m[m[0] \circ n[0]]\) and \(m[m'[0] \circ n[0]]\) for arb. \(m \neq m'\).

These trees cannot be distinguished by an AL-formula using \(p\).

We work with finite set \(N' \subseteq N\), with \(n \in N'\) and \(m, m' \not\in N'\).
Result from Modal Logic If $\sim$ is an AL-bisimulation, then $t \sim t'$ implies $t, t'$ are logically indistinguishable in AL.

Definition Define the symmetric relation $\sim$ by $t \sim t'$ iff

- $t$ has equal nodes iff $t'$ has equal nodes
- $t = n[t_1]$ implies there exists $n', t'_1$ st. $t' = n'[t'_1]$ and $t_1 \sim t'_1$ and if $n \in N'$ then $n = n'$, and vice versa
- $t = t_1 \circ t_2$ implies there exists $t'_1, t'_2$ st. $t' = t'_1 \circ t'_2$ and $t_1 \sim t'_1$ and $t_2 \sim t'_2$, and vice versa.

$m[m[0] \circ n[0]] \sim m[m'[0] \circ n[0]]$ for $m, m' \not\in N'$.

Result $\sim$ is a AL-bisimulation
Other Parametric Inexpressivity Results

Heaps

SL is parametrically as expressive as $\mathcal{CL}_0$ for heaps.

SL is as expressive as first-order logic plus atomic heap formulae $\mathcal{L}$. It is not parametrically as expressive. Direct proof

Bisimulation proof technique still interesting $p \ast q$ not parametrically expressible in first-order logic using $p = list(3)$ and $q = list(4)$.

Sequences $\mathcal{CL}_0$ for sequences is as expressive as $\mathcal{BL}$ for sequences

It describes the $\ast$-free regular languages

It is not parametrically as expressive as $\mathcal{BL}$ for sequences
**Conclusions**

Context Logic is a fundamental logic for reasoning about data.

Reasoning about data update requires reasoning about contexts.

Parametric inexpressivity results are intriguing.

**Future**

Combination of tree update with queries *Gardner, Zarfaty, MFPS’06*

Small-axiom approach prob. requires multi-holed contexts and wiring.

Integration of high-level and low-level reasoning.

Unified Hoare reasoning about data update.

Other applications of Context Logic.