Context Logic

Imperial College London

Thomas Dinsdale-Young td202@doc.ic.ac.uk
Gareth Smith gds@doc.ic.ac.uk
Philippa Gardner pg@doc.ic.ac.uk
Mark Wheelhouse mjw03@doc.ic.ac.uk

Tree Structure

Tree $T ::= \varnothing \mid \mathbf{n}[T] \mid T|T$

Context $C ::= \_ \mid \mathbf{n}[C] \mid C|T \mid T|C$

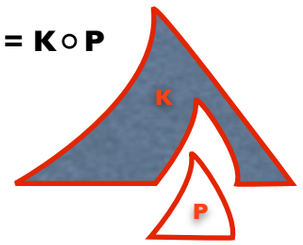
node identifiers **n** are unique
| is ordered with identity ∅

Context Logic for Trees

Classical Connectives + Structural Connectives
+ Tree-Specific Connectives

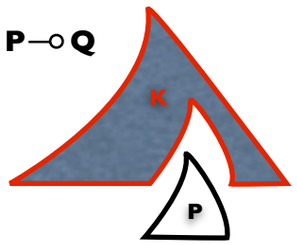
Structural Connectives - Context Application and Adjoints:

Q = K ∘ P



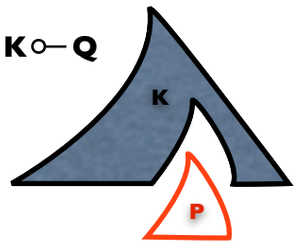
Tree **Q** can be split into context **K** applied to tree **P**

K = P —∘ Q



Whenever we put tree **P** into context **K** then we have tree **Q**

P = K ∘— Q



Whenever we put context **K** around tree **P** then we have tree **Q**

Local Hoare Reasoning

Hoare Triples:

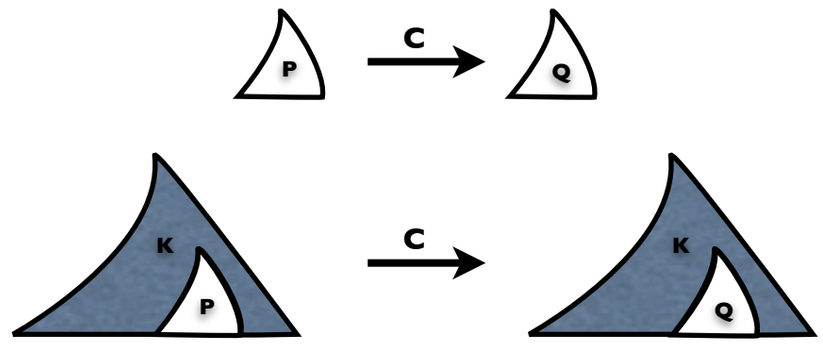
{P} **C** {Q}
pre post

If P holds before we run command **C** then the command will not fault and, if the command terminates, then Q will hold when **C** is finished.

Frame Rule:

$$\frac{\{P\}\, \mathtt{C}\, \{Q\}}{\{K \circ P\}\, \mathtt{C}\, \{K \circ Q\}}$$

where variables in C do not clash with those in K.



Specifying DOM – Core Level 1

DOM Data Structure:

trees	$t ::= s_{id}[f]_{fid} \mid \#text_{id}[s]$	*single root*
forests	$f ::= \varnothing_F \mid \langle t \rangle_F \mid f \otimes f$	*list of trees*
groves	$g ::= \varnothing_G \mid \langle t \rangle_G \mid g \oplus g$	*set of trees*
strings	$s ::= \varnothing_S \mid c \mid s \cdot s$	

node and forest identifiers id and fid are unique

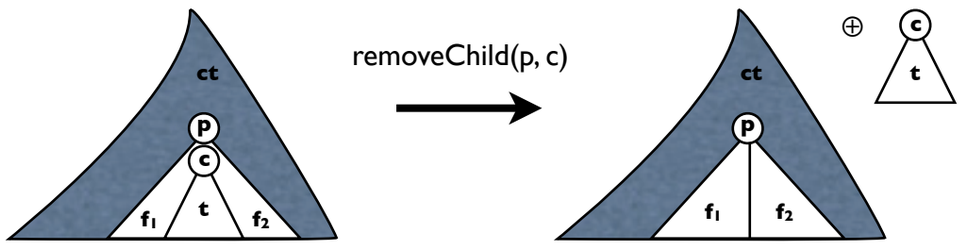
Part of DOM Core Level 1 covered:

Example DOM Command and Reasoning:

$$\{\langle ct \circ_T (\mathtt{name}_{parent}[f_1 \otimes \langle \mathtt{name'}_{child} \wedge t \rangle_F \otimes f_2]_{fid})\rangle_G\}$$

removeChild(parent, child)

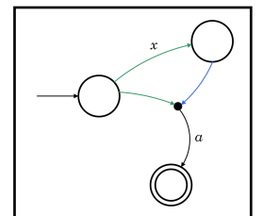
$$\{\langle ct \circ_T (\mathtt{name}_{parent}[f_1 \otimes f_2]_{fid})\rangle_G \oplus \langle \mathtt{name'}_{child} \wedge t \rangle_G\}$$



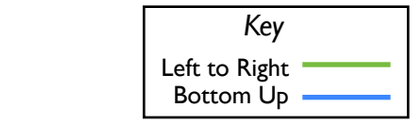
Decidability

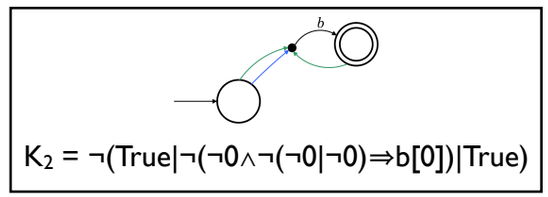
Related logics Ambient Logic and Separation Logic (without quantifiers) are decidable by the size argument. This doesn't work for Context Logic, because of application, but we can use automata:

Finite tree automata can implement (multi-holed) Context Logic, giving us a way to decide whether a tree satisfies a formula (model checking), or if there is any tree that satisfies a formula (satisfiability).

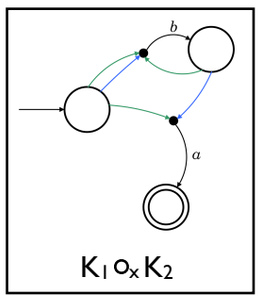


$K_1 = a[x]$





$K_2 = \neg(True|\neg(\neg 0 \wedge \neg(\neg 0|\neg 0) \Rightarrow b[0])|True)$



$K_1 \circ_x K_2$

Verification Tool

Our next step is to find an appropriate decidable fragment of Context Logic with quantification for automating program verification. For example one could verify that Javascript programs never violate schema invariants.