Adjunct Elimination in Context Logic for Trees

Cristiano Calcagno, Thomas Dinsdale-Young, and Philippa Gardner

Department of Computing
Imperial College London
London, UK
{ccris,td202,pg}@doc.ic.ac.uk

Abstract. We study adjunct-elimination results for Context Logic applied to trees, following previous results by Lozes for Separation Logic and Ambient Logic. In fact, it is not possible to prove such elimination results for the original single-holed formulation of Context Logic. Instead, we prove our results for multi-holed Context Logic.

1 Introduction

Separation Logic [1,2,3] and Ambient Logic [4] are related theories for reasoning, respectively, about local heap update and static trees. Inspired by this work, Calcagno, Gardner and Zarfaty invented Context Logic [5] for reasoning about structured resource, extending the general theory of Bunched Logic [6] for reasoning about unstructured resource. In particular, we use Context Logic applied to trees to reason about tree update, following the local reasoning style of Separation Logic; such reasoning is not possible using Ambient Logic [7].

These logics extend the standard propositional connectives with a structural (separating) composition for reasoning about disjoint subdata and the corresponding structural adjoint(s) for expressing properties such as weakest pre-conditions and safety conditions. For Separation Logic and Ambient Logic, Lozes [8] and then Dawar, Gardner and Ghelli [9] showed that the structural adjoints provide no additional expressive power on closed formulae. This result is interesting, as the adjunct connectives introduce quantification over potentially infinite sets whereas the structural composition only requires quantification over finite substructures. Following this work, Calcagno, Gardner and Zarfaty proved adjunct elimination for Context Logic applied to sequences, and showed the correspondence with the $*$-free regular languages [10,7]. We expected an analogous result for Context Logic applied to trees, but instead found a counter-example (first reported in Dinsdale-Young's Masters thesis [11]). Instead, we prove an adjunct-elimination result for *multi-holed* Context Logic applied to trees.

The original Context Logic was introduced to establish local Hoare reasoning about tree update. For this application, it was enough to work with single-holed contexts, although we always understood that there were other forms of contexts requiring study. Our counter-example to the adjunct-elimination result for single-holed Context Logic motivates our exploration of these other context structures. In our original presentation, the data formula $K(P)$ expresses that

Z. Shao (Ed.): APLAS 2007, LNCS 4807, pp. 255–270, 2007.

the given tree is the result of applying a single-holed context satisfying context formula K to a tree satisfying data formula P. The adjunct context formula $P \triangleright Q$ expresses that, whenever a tree satisfying property P is put in the context hole, then the resulting tree satisfies Q. Consider the single-holed context formula $0 \triangleright (True(u[0]))$ expressing that, when the empty tree 0 is put in the context hole, then somewhere there is a subtree of the form $u[0]$ with top node labelled u and empty subtree. This formula cannot be expressed without the separating adjoint connective '$\triangleright$'. For example, consider contexts of the form $v^n[u[\_]]$, denoting a vertical line of n nodes labelled v, followed by one node labelled u and then the context hole. These contexts all satisfy the formula $0 \triangleright (True(u[0]))$, whereas the contexts $v^n[\_]$ do not. There is no adjoint-free context formula that can distinguish between these sets of contexts because, in our original presentation of Context Logic, trees can be split arbitrarily into contexts and trees, but contexts cannot be split. Our counter-example shows that such splitting is necessary for adjunct elimination to hold.

Context Logic can be extended with context composition, for analysing the splitting of contexts, and its adjoints. However, we currently do not know if adjunct elimination holds for this extension. We do know that current techniques for proving such results cannot be immediately adapted. Instead, we prove adjunct elimination for multi-holed Context Logic with context composition. Our proof, adapting the technique for proving adjunct elimination using model-checking games [9], naturally requires the extension to multi-holed contexts. To illustrate this, consider the tree $t = c_1(t_1)$ which denotes the application of context c_1 to tree t_1. An application move in a game will split t into $c_2(t_2)$, leading to a case analysis relating c_1 and t_1 with c_2 and t_2 involving multi-holed contexts. For example, when t_2 is a subtree of c_1, this case is simply expressed using a two-holed context $d(\_,\_)$ with $d(t_2,\_) = c_1$ and $d(\_,t_1) = c_2$. Using multi-holed Context Logic, we are thus able to provide an adjunct-elimination result which conforms with the analogous results for Separation Logic and Ambient Logic. In addition, we believe multi-holed Context Logic presented here will play an important role in our future development of Context Logic since, although analysing multi-holed contexts was not necessary for our preliminary work on tree update, they do seem to be fundamental for other applications such as reasoning about concurrent tree update.

2 Multi-holed Context Logic for Trees

We work with finite, ordered, unranked trees (strictly speaking, forests) and contexts, with nodes labelled from a node alphabet Σ ranged over by u, v, w. Our contexts are simply trees with some of the leaves — the context holes — uniquely labelled from a hole alphabet X, ranged over by x, y, z. We view trees as contexts without context holes.

Definition 1 (Multi-holed Tree Contexts). *We define the set of multi-holed tree contexts, $C_{\Sigma,X}$, ranged over by c, d, by the grammar*

$$c ::= \ \varepsilon \ \mid \ u[c] \ \mid \ c_1 \mid c_2 \ \mid \ x$$

with the restriction that each hole label, $x \in X$, occurs at most once *in the context c, and subject to the $\mid$ operator being associative and having identity ε. We denote the set of hole labels that occur in c by $fn(c)$. We use u as an abbreviation of $u[\varepsilon]$.*

Definition 2 (Context Application). *We define* context application *(or* context composition*) as a set of partial functions identified with the hole labels, $ap_x : C_{\Sigma,X} \times C_{\Sigma,X} \rightharpoonup C_{\Sigma,X}$.*

$$ap_x(c_1, c_2) = \begin{cases} c_1[c_2/x] & \text{if } x \in fn(c_1) \text{ and } fn(c_1) \cap fn(c_2) \subseteq \{x\} \\ \text{undefined} & \text{otherwise} \end{cases}$$

We abbreviate $ap_x(c_1, c_2)$ by $c_1 \circledast{x} c_2$.

This definition of multi-holed context, also studied in [12], seems to be the most appropriate for our reasoning style, since it allows contexts to be separated easily. An alternative formulation is to order the holes, rather than uniquely name them, but this approach does not sit so naturally with separating contexts.

Example 1. The context $c_1 = u[u[v] \mid u[u \mid v]] \mid v$ is a tree with no hole labels. It may be expressed as the application of a single-holed context to another tree, e.g. $c_1 = u[x \mid u[u \mid v]] \mid v \circledast{x} u[v]$. It may also be expressed as a two-holed context applied to two trees, e.g. $c_1 = (u[x \mid u[u \mid y]] \mid v \circledast{y} v) \circledast{x} u[v]$. Recall that the context holes are labelled uniquely by x and y, with the first application $u[x \mid u[u \mid y]] \mid v \circledast{y} v$ declaring that the argument v should be placed in the hole labelled y. Note that $u[x \mid u[u \mid x]] \mid v$ does not fit our definition of a context since the hole label x occurs more than once.

Lemma 1. *If $y = x$ or $y \notin fn(c_1)$, then $c_1 \circledast{x} (c_2 \circledast{y} c_3) = (c_1 \circledast{x} c_2) \circledast{y} c_3$, where defined.*

Lemma 2. *If $y \neq x$, $x, y \in fn(c_1)$, $y \notin fn(c_2)$, $x \notin fn(c_3)$, then*

$$(c_1 \circledast{x} c_2) \circledast{y} c_3 = (c_1 \circledast{y} c_3) \circledast{x} c_2.$$

We define multi-holed Context Logic for trees, denoted CL^m_{Tree}. For those who are familiar with Separation Logic and Ambient Logic, this follows the familiar pattern of extending the propositional connectives of classical logic with structural connectives for analysing the structure of multi-holed contexts, and specific connectives for analysing the particular data structure under consideration (in this case, trees).

Definition 3 (Formulae of CL^m_{Tree}). *Let Θ be an alphabet of hole variables ranged over by α, β, γ. Multi-holed Context Logic formulae, $K_1, K_2, \ldots$, are defined by the grammar:*

$$\begin{aligned} K ::= {} & 0 \mid u[K] \mid K_1 \mid K_2 \\ & \alpha \mid K_1 \circ_\alpha K_2 \mid K_1 \circ\!\!- _\alpha K_2 \mid K_1 -\!\!\circ_\alpha K_2 \mid \exists \alpha.\, K \\ & \mathit{False} \mid K_1 \Rightarrow K_2. \end{aligned}$$

We use the Boolean connectives '*False*' and '$\Rightarrow$'. The specific connectives '0', '$u[]$' and '$|$' express basic structural properties of our tree contexts: a tree context is empty, has top node labelled u, or is the concatenation of two contexts respectively. The structural connectives 'α', '$\circ_\alpha$', '$\circ\!\!-_\alpha$' and '$-\!\!\circ_\alpha$' describe fundamental properties of multi-holed contexts. The connective 'α' expresses that a context is a hole whose label is the value of the variable α. The '$\circ_\alpha$' specifies that a context is a composition of two contexts where the hole being filled is the value of α. The '$\circ\!\!-_\alpha$' and '$-\!\!\circ_\alpha$' are the adjoints of composition: $K_1 \circ\!\!-_\alpha K_2$ expresses that, whenever a context satisfying K_1 is α-composed *on the left* with the given context, the result satisfies K_2; while $K_1 -\!\!\circ_\alpha K_2$ expresses that, whenever a context satisfying K_1 is α-composed *on the right* with the given context, the result satisfies K_2. In addition, we have existential quantification over hole labels, which allows us to specify context composition without specific reference to the hole name.

Definition 4 (Satisfaction relation of CL^m_{Tree}). *An* environment *is a finite partial function $\sigma : \Theta \rightharpoonup X$ which assigns hole labels to variables. We denote the empty environment by $\emptyset$, and the extension of σ with a new domain element α with value y by $\sigma[\alpha \mapsto y]$. The satisfaction relation for CL^m_{Tree} is given with respect to an environment as follows, where $x = \sigma\alpha$:*

$$
\begin{array}{lcl}
c, \sigma \models 0 & \iff & c = \varepsilon \\
c, \sigma \models u[K] & \iff & \exists c'.\, c = u[c'] \wedge c', \sigma \models K \\
c, \sigma \models K_1 \mid K_2 & \iff & \exists c_1, c_2.\, c = c_1 \mid c_2 \wedge c_1, \sigma \models K_1 \wedge c_2, \sigma \models K_2 \\
c, \sigma \models \alpha & \iff & c = x \\
c, \sigma \models K_1 \circ_\alpha K_2 & \iff & \exists c_1, c_2.\, c = c_1 \circledast c_2 \wedge c_1, \sigma \models K_1 \wedge c_2, \sigma \models K_2 \\
c, \sigma \models K_1 \circ\!\!-_\alpha K_2 & \iff & \forall c_1, c_2.\, c_2 = c_1 \circledast c \wedge c_1, \sigma \models K_1 \implies c_2, \sigma \models K_2 \\
c, \sigma \models K_1 -\!\!\circ_\alpha K_2 & \iff & \forall c_1, c_2.\, c_2 = c \circledast c_1 \wedge c_1, \sigma \models K_1 \implies c_2, \sigma \models K_2 \\
c, \sigma \models \exists \alpha.\, K & \iff & \exists y.\, c, \sigma[\alpha \mapsto y] \models K \\
c, \sigma \not\models \mathit{False} & & \\
c, \sigma \models K_1 \Rightarrow K_2 & \iff & c, \sigma \models K_1 \implies c, \sigma \models K_2.
\end{array}
$$

We use two conventions for convenience. Firstly, we adopt Barendregt's convention and assume that bound variable names differ from free variable names, and furthermore differ from elements of the domain of any environment under consideration; if that is not the case, the bound variables may and are assumed to be renamed. Secondly, we only ever consider satisfaction of a formula when all of its free variables are assigned values by the environment. We also make use of standard derived connectives, where appropriate: $\mathit{True}, \neg, \wedge, \vee, \forall$. We assume the following binding order among the connectives: $\neg, |, \circ_\alpha, \wedge, \vee, \{\circ\!\!-_\alpha, -\!\!\circ_\alpha\}, \Rightarrow, \exists, \forall$, with no precedence between $\circ\!\!-_\alpha$ and $-\!\!\circ_\alpha$.

Example 2. We present a few example formulae

1. The formula $u[0]$ expresses that a tree consists of a single node labelled u.

Table 1. Ranks of Selected Formulae

Formula	Rank
$u[0] \mid (u[0] \mid u[0]) \vee \neg 0$	$(4, 0, \{u\})$
$\exists\alpha.\,(\neg u[v[0] \mid True]) \circ_\alpha \beta$	$(6, 0, \{u, v\})$
$u[v[\alpha] \multimap_\alpha (w[0] \circ\!\!- _\beta v[u[w[0]]])]$	$(5, 2, \{u, v, w\})$

2. The formula $\exists\alpha.\,(True \circ_\alpha u[0])$ expresses that a context contains tree $u[0]$.
3. The formula $(True \circ_\alpha \alpha)$ expresses that the value of α must be in the context.
4. The formula $\exists\alpha.\,(True \circ_\alpha \alpha) \wedge (0 \multimap_\alpha (\exists\beta.\, True \circ_\beta u[0]))$ expresses that the empty tree may be placed into some context hole such that the resulting tree has some leaf node labelled u.

As in the original Context Logic, we can derive the adjoints of the specific formulae: the adjoint of $u[-]$ is $\forall\alpha.\,(u[\alpha] \circ\!\!- _\alpha -)$; that of $- \mid K$ is $\forall\alpha.\,((\alpha \mid K) \circ\!\!- _\alpha -)$; and that of $K \mid -$ is $\forall\alpha.\,((K \mid \alpha) \circ\!\!- _\alpha -)$.

3 Games

We define Ehrenfeucht-Fraïssé style games for CL^m_{Tree}. The games are useful for our results because they are sound and complete with respect to the logic: two contexts can be distinguished by a logical formula if and only if Spoiler has a winning strategy for a corresponding game. Our presentation is similar to that of [9]. However, we use a more relaxed definition of rank, which simply distinguishes between the adjunct and non-adjunct moves. The proofs of the lemmata in this section will appear in the full version of this paper.

We first define the rank of a logical formula, a concept which is also used to parametrise games. Some examples are given in Table 1.

Definition 5 (Rank). *The* rank *of a formula is a tuple $r = (n, s, \mathcal{L})$ where:*

- *n is the greatest nesting depth of the non-adjunct, non-Boolean connectives, i.e. $0, u[K], K_1 \mid K_2, \alpha, K_1 \circ_\alpha K_2, \exists\alpha.K$;*
- *s is the greatest nesting depth of the adjunct, non-Boolean connectives, i.e. $K_1 \circ\!\!- _\alpha K_2, K_1 \multimap_\alpha K_2$; and*
- *$\mathcal{L}$ is the subset of Σ consisting of the node labels that occur in the formula.*

Lemma 3. *For each rank r and finite set of variables $\mathcal{V} \subset \Theta$, there are finitely many non-equivalent formulae of rank r whose free variables are in $\mathcal{V}$.*

Lemma 4. *Let $\mathcal{T}$ be a set of context-environment pairs such that, for any $\mathcal{T}$-discriminated pair*[1] *$((c, \sigma), (c', \sigma'))$ there exists a formula $K_{(c,\sigma),(c',\sigma')}$ of rank r and free variables in finite set $\mathcal{V}$ such that $c, \sigma \models K_{(c,\sigma),(c',\sigma')}$ and $c', \sigma' \not\models K_{(c,\sigma),(c',\sigma')}$. Then $\mathcal{T}$ can be defined by a rank-r formula K with free variables in $\mathcal{V}$.*

[1] A $\mathcal{T}$-discriminated pair is a pair (a, b) with $a \in \mathcal{T}$ and $b \notin \mathcal{T}$, or $a \notin \mathcal{T}$ and $b \in \mathcal{T}$.

We now define the Ehrenfeucht-Fraïssé-style games that we shall use in our main result. A game state is a tuple, $((c,\sigma),(c',\sigma'),r)$, where c and c' are contexts, σ and σ' are environments with coincident domains, and $r = (n,s,\mathcal{L})$ is a rank. The game is played between two players, Spoiler and Duplicator. At each step, Spoiler selects a move to play, and the two players make choices according to the rules for that move. After a move is played out, either Spoiler will have won the game or the game will continue with a new state that has a reduced rank (either n or s will be reduced by one, depending on the move). If the rank reaches $(0,0,\mathcal{L})$, Duplicator wins.

Each move in the game $((c,\sigma),(c',\sigma'),(n,s,\mathcal{L}))$ begins by Spoiler selecting one of the pairs (c,σ) or (c',σ'). We shall call Spoiler's selection (d,ρ) and the other (d',ρ'). Spoiler may only play a particular move when the rank allows it. A move is also prohibited when Spoiler cannot make the choice stipulated by the move. The moves are defined as follows:

Moves playable when $n > 0$ (the *non-adjunct moves*):

EMP move. Spoiler's choice is such that $d = \varepsilon$ and $d' \neq \varepsilon$. Spoiler wins.

VAR move. Spoiler chooses $\alpha \in \Theta$ with $d = \rho\alpha$ and $d' \neq \rho'\alpha$. Spoiler wins.

LAB move. Spoiler chooses some $u \in \mathcal{L}$ and $d_1 \in \mathcal{C}$ such that $d = u[d_1]$. If $d' = u[d'_1]$ for some $d'_1 \in \mathcal{C}$, the game continues with $((d_1,\rho),(d'_1,\rho'),(n-1,s,\mathcal{L}))$. Otherwise, Spoiler wins.

PAR move. Spoiler chooses some $d_1, d_2 \in \mathcal{C}$ such that $d = d_1 \mid d_2$. Duplicator chooses some $d'_1, d'_2 \in \mathcal{C}$ such that $d' = d'_1 \mid d'_2$. Spoiler decides whether the game continues with $((d_1,\rho),(d'_1,\rho'),(n-1,s,\mathcal{L}))$ or $((d_2,\rho),(d'_2,\rho'),(n-1,s,\mathcal{L}))$.

CMP move. Spoiler chooses $x = \rho\alpha$ for some α, and $d_1, d_2 \in \mathcal{C}$ such that $d = d_1 \mathbin{⊙} d_2$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}$ such that $d' = d'_1 \mathbin{⊙} d'_2$ for $\acute{x} = \rho'\alpha$. Spoiler decides whether the game will continue with $((d_1,\rho),(d'_1,\rho'),(n-1,s,\mathcal{L}))$ or $((d_2,\rho),(d'_2,\rho'),(n-1,s,\mathcal{L}))$.

EXS move. Let $\alpha \in \Theta$ be some new hole variable (i.e. $\sigma\alpha$, and equivalently $\sigma'\alpha$, are undefined). Spoiler chooses some $x \in X$. Duplicator chooses an answering $\acute{x} \in X$. The game then continues with $((d,\rho[\alpha \mapsto x]),(d',\rho'[\alpha \mapsto \acute{x}]),(n-1,s,\mathcal{L}))$.

Moves playable when $s > 0$ (the *adjunct moves*):

LEF move. Spoiler chooses $x = \rho\alpha$ for some α, and $d_1, d_2 \in \mathcal{C}$ such that $d_2 = d_1 \mathbin{⊙} d$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}$ such that $d'_2 = d'_1 \mathbin{⊙} d'$ for $\acute{x} = \rho'\alpha$. Spoiler decides whether the game will continue with $((d_1,\rho),(d'_1,\rho'),(n,s-1,\mathcal{L}))$ or $((d_2,\rho),(d'_2,\rho'),(n,s-1,\mathcal{L}))$.

RIG move. Spoiler chooses $x = \rho\alpha$ for some α, and $d_1, d_2 \in \mathcal{C}$ such that $d_2 = d \mathbin{⊙} d_1$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}$ such that $d'_2 = d' \mathbin{⊙} d'_1$ for $\acute{x} = \rho'\alpha$. If Duplicator cannot make such a choice, Spoiler wins. Otherwise, Spoiler decides whether the game will continue with $((d_1,\rho),(d'_1,\rho'),(n,s-1,\mathcal{L}))$ or $((d_2,\rho),(d'_2,\rho'),(n,s-1,\mathcal{L}))$.

Of more interest than the outcome of an individual run of a game is the question of which player has a winning strategy for that game: Spoiler or Duplicator is capable of ensuring his or her victory regardless of how the other plays. If Spoiler has a winning strategy, we say $((c,\sigma),(c',\sigma'),r) \in SW$. Otherwise, we

say $((c,\sigma),(c',\sigma'),r) \in DW$. The following useful properties are direct consequences of the definitions.

Proposition 1 (Downward Closure). *If* $((c,\sigma),(c',\sigma'),(n,s,\mathcal{L})) \in DW$ *then* $((c,\sigma),(c',\sigma'),(n',s',\mathcal{L}')) \in DW$ *for any* $n' \leq n$, $s' \leq s$ *and* $\mathcal{L}' \subseteq \mathcal{L}$.

Proposition 2 (Downward Closure for Environments). *If* $((c,\sigma[\alpha \mapsto x]),(c',\sigma'[\alpha \mapsto \acute{x}]),r) \in DW$ *then* $((c,\sigma),(c',\sigma'),r) \in DW$.

At each stage, Spoiler is trying to show that the two contexts are different, while Duplicator is trying to show that they are similar enough that Spoiler cannot identify a difference. The game moves correspond closely with the (non-Boolean) connectives of the logic. For instance, the RIG move corresponds to $-\!\circ_\alpha$ connective: it speaks of applying the given context to a new one and then reasoning about the result or the new context. If Spoiler wins on playing that move, it means that the two (current) trees are differentiated by the formula *True* $-\!\circ_\alpha$ *False* — one tree has a α-labelled hole (so the formula is not satisfied) while the other does not (so the formula is satisfied trivially).

The reason for this correspondence is that formulae, of rank r, which distinguish between two contexts, will correspond to winning strategies for Spoiler for the game of rank r on those two contexts. This is formalised in the soundness and completeness results which we state.

Lemma 5 (Game Soundness). *For* $c, c' \in \mathcal{C}$ *and domain-coincident environments* σ, σ', *if there is a formula* K *of rank* r *such that* $c,\sigma \models K$ *and* $c',\sigma' \not\models K$, *then* Spoiler *has a winning strategy for the game* $((c,\sigma),(c',\sigma'),r)$.

Lemma 6 (Game Completeness). *If* Spoiler *has a winning strategy for the game* $((c,\sigma),(c',\sigma'),r)$ *then there exists a formula,* K, *of rank at most* r *such that* $c,\sigma \models K$ *and* $c',\sigma' \not\models K$.

The following two lemmata are useful for checking structural properties. The first establishes a relationship between the hole labels in two contexts, which provides a convenient way of checking that composition is well defined. The second establishes a structural similarity through games. Both are proven by showing how Spoiler would have a winning strategy for the game in a certain number of moves (hence the bounds on n) if the desired property did not hold.

Lemma 7. *If* $((c,\sigma),(c',\sigma'),(n,s,\mathcal{L})) \in DW$ *with* $n \geq 2$, *then, for* $x = \sigma\alpha$, $\acute{x} = \sigma'\alpha$,

$$x \in fn(c) \iff \acute{x} \in fn(c')$$

Lemma 8. *Suppose that* $((c,\sigma),(c',\sigma'),(n,s,\mathcal{L})) \in DW$ *with* $n \geq 2$. *Then if* $c = \bar{c} \mid x$ *for* $x = \sigma\alpha$, $\bar{c} \in \mathcal{C}$ *then* $c' = \bar{c}' \mid \acute{x}$ *for* $\acute{x} = \sigma'\alpha$ *and some* $\bar{c}' \in \mathcal{C}$. *Similarly, if* $c = x \mid \bar{c}$ *then* $c' = \acute{x} \mid \bar{c}'$.

The next lemma essentially gives two sufficient conditions on Duplicator's response to the EXS move in order for it to give a winning strategy for her. The key part is that if Spoiler introduces a fresh hole label, Duplicator may respond by introducing *any* fresh hole label. The restriction on n is used to establish freshness for the second of the cases.

Lemma 9 (Interchangablity of Fresh Labels). *If* $((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW$ *with* $n \geq 3$*, then* $((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto \acute{x}]), (n-1, s, \mathcal{L})) \in DW$ *if either* $x = \sigma\beta$ *and* $\acute{x} = \sigma'\beta$*, or* $x \notin fn(c) \cup range(\sigma)$ *and* $\acute{x} \notin fn(c') \cup range(\sigma')$*.*

4 Adjunct Elimination

We now have the background required to prove adjunct elimination for CL^m_{Tree}. Proposition 3 is the key, most complicated result. It states that, with no adjunct moves, a winning strategy for the composition of contexts follows from winning strategies for its components. A consequence is that if Duplicator has a winning strategy with adjunct moves, then she has a winning strategy without adjunct moves, since adjunct moves simply perform context composition. The final theorem then translates this move elimination result into an adjunct elimination result for the formulae of the logic.

Proposition 3 (One-step move elimination). *For all ranks of the form* $r = (n, 0, \mathcal{L})$*, for all* $c_1, c'_1, c_2, c'_2 \in \mathcal{C}$*, for all domain-coincident environments* σ, σ'*, if*

$$((c_1, \sigma), (c'_1, \sigma'), (3n, 0, \mathcal{L})) \in DW \tag{1}$$

$$((c_2, \sigma), (c'_2, \sigma'), (3n, 0, \mathcal{L})) \in DW \tag{2}$$

then for all $\alpha \in dom(\sigma)$ *with* $x = \sigma\alpha$, $\acute{x} = \sigma'\alpha$*: if* $c = c_1 \circledast_x c_2$ *and* $c' = c'_1 \circledast_{\acute{x}} c'_2$ *are defined then*

$$((c, \sigma), (c', \sigma'), r) \in DW. \tag{3}$$

Proof. The proof is by induction on n and by cases on Spoiler's choice of move in the game of (3). The base case, $n = 0$, is trivial, since Spoiler can never win a game of such a rank. We assume as the inductive hypothesis that the proposition holds for lesser values of n. Assume without loss of generality that Spoiler selects (c, σ) for his move.

Throughout the proof, we consider strategies that Spoiler might adopt in the games of (1) and (2). Knowing that Duplicator has a winning strategy in these games, we are able to establish properties, usually concerning the structure of c'_1 and c'_2, based on her strategy, and, often using the inductive hypothesis, use these to construct a winning response for Duplicator to Spoiler's move on (3).

EMP move. In order for Spoiler to be able to play this move, it must be the case that $c = \varepsilon$ and $c' \neq \varepsilon$. Thus $c_1 = x$ and $c_2 = \varepsilon$. Hence $c'_1 = \acute{x}$ and $c'_2 = \varepsilon$, so $c' = \varepsilon$. Therefore, Spoiler cannot play this move after all.

LAB move. Suppose that Spoiler plays this move picking $u \in \mathcal{L}$ and $d \in \mathcal{C}$ with $c = u[d]$. Then there are three cases of the possible structure of c_1 and c_2: 1. $c_1 = u[d_1]$ and $d = d_1 \circledast_x c_2$; 2. $c_1 = u[d] \mid x$ and $c_2 = \varepsilon$; 3. $c_1 = x \mid u[d]$ and $c_2 = \varepsilon$.

In the first of these cases, Spoiler could play the LAB move on the game of (1), with label u and context d_1. Hence, by (1), $c'_1 = u[d'_1]$ with

$$((d_1, \sigma), (d'_1, \sigma'), (3n - 1, 0, \mathcal{L})) \in DW. \tag{4}$$

By downward closure and the inductive hypothesis, noting that $d'_1 \mathbin{⊛} c'_2$ is defined, since $fn(d'_1) = fn(c'_1)$ and $c'_1 \mathbin{⊛} c'_2$ is defined, it follows that

$$((d_1 \mathbin{⊛} c_2, \sigma), (d'_1 \mathbin{⊛} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{5}$$

By structural considerations, $c' = u[d']$ where $d' = d'_1 \mathbin{⊛} c'_2$. Thus Duplicator has a winning strategy when Spoiler plays this way.

In the second of the cases, $c'_2 = \varepsilon$ by (2). Further, Spoiler could play the PAR move on (1) so we have $c'_1 = d'_1 \mid d'_2$ with

$$((u[d], \sigma), (d'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{6}$$
$$((x, \sigma), (d'_2, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{7}$$

Since $3n - 1 \geq 1$, by (7) we know $d'_2 = \acute{x}$. Spoiler could play the LAB move on the former, using u as the label, so that we must have $d'_1 = u[d']$ with

$$((d, \sigma), (d', \sigma'), (3n-2, 0, \mathcal{L})) \in DW. \tag{8}$$

We now have $c' = (u[d'] \mid \acute{x}) \mathbin{⊛} \varepsilon = u[d']$. Hence, Duplicator can respond and the game continues as $((d, \sigma), (d', \sigma'), (n-1, 0, \mathcal{L}))$ and, by downward closure on (8), Duplicator has a winning strategy. The third case is essentially the same as this.

In each of the three cases, Duplicator has a winning strategy, so she has a winning strategy if Spoiler plays the LAB move.

PAR move. In this move, Spoiler splits $c = d_1 \mid d_2$ in one of three ways:

1. Spoiler splits in c_1 to the left of the x. That is, $c_1 = d_1 \mid d_3$, $d_2 = d_3 \mathbin{⊛} c_2$.
2. Spoiler splits in c_1 to the right of the x. This case is essentially the same as the first, so we shall not consider it.
3. Spoiler splits in c_2. In order for this case to be applicable, the x must occur at the top level of c_1, so $c_1 = \bar{d}_3 \mid x \mid \bar{d}_4$, $d_1 = \bar{d}_3 \mid d_5$ and $d_2 = d_6 \mid \bar{d}_4$ with

$$c_1 \mathbin{⊛} c_2 = d_1 \mid d_2 = (d_3 \mathbin{⊛} d_5) \mid (d_4 \mathbin{⊛} d_6)$$
$$d_3 = \bar{d}_3 \mid x \qquad d_4 = x \mid \bar{d}_4$$
$$c_1 = d_3 \mathbin{⊛} d_4 = (\bar{d}_3 \mid x) \mathbin{⊛} (x \mid \bar{d}_4) \qquad c_2 = d_5 \mid d_6.$$

In the first case, $c_1 \mathbin{⊛} c_2 = (d_1 \mid d_3) \mathbin{⊛} c_2 = d_1 \mid (d_3 \mathbin{⊛} c_2)$. As Spoiler could play the PAR move in the game in (1), we know that $c'_1 = d'_1 \mid d'_3$ such that

$$((d_1, \sigma), (d'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{9}$$
$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{10}$$

Note that $fn(d'_3) \subseteq fn(c'_1)$ and $\acute{x} \in fn(d'_3)$ by Lemma 7 (since $x \in fn(d_3)$), so $d'_2 = d'_3 \mathbin{⊛} c'_2$ is defined. By downward closure on (10) and (2) and by the inductive hypothesis,

$$((d_3 \mathbin{⊛} c_2, \sigma), (d'_3 \mathbin{⊛} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{11}$$

Observe that $c' = c'_1 \mathbin{⊛} c'_2 = (d'_1 \mid d'_3) \mathbin{⊛} c'_2 = d'_1 \mid (d'_3 \mathbin{⊛} c'_2) = d'_1 \mid d'_2$. Thus responding with d'_1 and d'_2 gives Duplicator a winning strategy in this case, by downward closure on (9) and by (11).

In the third case, Spoiler could play the CMP move on the game in (1), so $c'_1 = d'_3 \mathbin{⊛} d'_4$ with

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{12}$$
$$((d_4, \sigma), (d'_4, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{13}$$

Also, Spoiler could play the PAR move on the game in (2), so $c'_2 = d'_5 \mid d'_6$ with

$$((d_5, \sigma), (d'_5, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{14}$$
$$((d_6, \sigma), (d'_6, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{15}$$

Since $c'_1 = d'_3 \mathbin{⊛} d'_4$ and $c'_2 = d'_5 \mid d'_6$, it follows that that $\acute{x} \in fn(d'_3) \subseteq fn(c'_1)$, $\acute{x} \in fn(d'_4) \subseteq fn(c'_1)$, $fn(d'_5) \subseteq fn(c'_2)$ and $fn(d'_6) \subseteq fn(c'_2)$. Hence $d'_1 = d'_3 \mathbin{⊛} d'_5$ and $d'_2 = d'_4 \mathbin{⊛} d'_6$ are well defined. By downward closure and the inductive hypothesis on (12) and (14), and on (13) and (15), we get

$$((d_3 \mathbin{⊛} d_5, \sigma), (d'_3 \mathbin{⊛} d'_5, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{16}$$
$$((d_4 \mathbin{⊛} d_6, \sigma), (d'_4 \mathbin{⊛} d'_6, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{17}$$

It remains to show that $c' = d'_1 \mid d'_2$. For this to be the case, it is sufficient that $d'_3 = \bar{d'_3} \mid \acute{x}$ and $d'_4 = \acute{x} \mid \bar{d'_4}$, which both hold by applying Lemma 8 to (12) and (13). Thus, by structural considerations, $c' = c'_1 \mathbin{⊛} c'_2 = (d'_3 \mathbin{⊛} d'_4) \mathbin{⊛} (d'_5 \mid d'_6) = ((\bar{d'_3} \mid \acute{x}) \mathbin{⊛} (\acute{x} \mid \bar{d'_4})) \mathbin{⊛} (d'_5 \mid d'_6) = \bar{d'_3} \mid d'_5 \mid d'_6 \mid \bar{d'_4} = (d'_3 \mathbin{⊛} d'_5) \mid (d'_4 \mathbin{⊛} d'_6) = d'_1 \mid d'_2$. Hence, by (16) and (17), Duplicator has a winning strategy if she responds by splitting c' as $d'_1 \mid d'_2$.

Thus, Duplicator has a winning strategy whenever Spoiler plays the PAR move.

CMP move. In this move, Spoiler chooses $y = \sigma\beta$ (let $\acute{y} = \sigma'\beta$), and splits $c_1 \mathbin{⊛} c_2$ as $d_1 \mathbin{⊚} d_2$. Note that Spoiler cannot play the CMP move as the final move of a winning strategy, so we may therefore assume that $n \geq 2$. (If $n = 1$, Duplicator would have a winning strategy by splitting $c' = \acute{y} \mathbin{⊚} c'$, for instance.)

There are four cases for how Spoiler can make the splitting $c = d_1 \mathbin{⊚} d_2$. We shall consider each in turn.

Case 1: Spoiler splits inside c_2, as

$$c_1 \mathbin{⊛} c_2 = c_1 \mathbin{⊛} (d_3 \mathbin{⊚} d_2) = (c_1 \mathbin{⊛} d_3) \mathbin{⊚} d_2 = d_1 \mathbin{⊚} d_2$$
$$c_2 = d_3 \mathbin{⊚} d_2 \qquad d_1 = c_1 \mathbin{⊛} d_3.$$

Spoiler would be able to play the CMP move on the game in (2), so Duplicator must be able to split c'_2 as $d'_3 \mathbin{⊚} d'_2$ such that

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{18}$$
$$((d_2, \sigma), (d'_2, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{19}$$

Note that $fn(d'_3) \subseteq fn(c'_2) \cup \{\acute{y}\}$. Also, by Lemma 7, $\acute{y} \notin fn(c'_1)$ since $y \notin fn(c_1)$. Hence $d'_1 = c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_3$ is well defined. By downward closure on (1) and (18) and by the inductive hypothesis,

$$((c_1 \mathbin{\textcircled{\scriptsize x}} d_3, \sigma), (c'_2 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_3, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{20}$$

By Lemma 1, since $\acute{y} \notin fn(c'_1)$, $c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_3 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2) = (c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_3) \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2 = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$. Hence, by (20) and by downward closure on (19), Duplicator has a winning strategy if she splits c' as $d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$.

Case 2: Spoiler splits outside c_2, including all of c_2 itself:

$$c_1 \mathbin{\textcircled{\scriptsize x}} c_2 = (d_1 \mathbin{\textcircled{\scriptsize y}} d_3) \mathbin{\textcircled{\scriptsize x}} c_2 = d_1 \mathbin{\textcircled{\scriptsize y}} (d_3 \mathbin{\textcircled{\scriptsize x}} c_2) = d_1 \mathbin{\textcircled{\scriptsize y}} d_2$$
$$c_1 = d_1 \mathbin{\textcircled{\scriptsize y}} d_3 \qquad d_2 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2.$$

Spoiler would be able to play the CMP move on the game in (1), so Duplicator must be able to split c'_1 as $d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_3$ such that

$$((d_1, \sigma), (d'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{21}$$
$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{22}$$

Note that $fn(d'_3) \subseteq fn(c'_1)$ and that, by Lemma 7, $\acute{x} \in fn(d'_3)$ since $x \in fn(d_3)$. Thus $d'_2 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2$ is well defined. By downward closure on (22) and (1) and by the inductive hypothesis,

$$((d_3 \mathbin{\textcircled{\scriptsize x}} c_2, \sigma), (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{23}$$

By Lemma 1, since $\acute{x} \notin fn(d'_1)$ (since $\acute{x} \in fn(d'_3)$ and $c'_1 = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_3$), $c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = (d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_3) \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2) = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$. Hence, by downward closure on (21) and by (23), Duplicator has a winning strategy if she splits c' as $d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$.

Case 3: Spoiler splits part of c_1 and part of c_2:

$$c_1 = d_3 \mathbin{\textcircled{\scriptsize x}} d_4 \qquad c_2 = d_5 \mathbin{\textcircled{\scriptsize y}} d_6 \qquad d_1 = d_3 \mathbin{\textcircled{\scriptsize x}} d_5 \qquad d_2 = d_4 \mathbin{\textcircled{\scriptsize x}} d_6$$

with either: $d_4 = \bar{d}_4 \mid x$ and $d_5 = y \mid \bar{d}_5$; or $d_4 = x \mid \bar{d}_4$ and $d_5 = \bar{d}_5 \mid y$. In the former, for instance, we have

$$\begin{aligned} c_1 \mathbin{\textcircled{\scriptsize x}} c_2 &= (d_3 \mathbin{\textcircled{\scriptsize x}} d_4) \mathbin{\textcircled{\scriptsize x}} (d_5 \mathbin{\textcircled{\scriptsize y}} d_6) = (d_3 \mathbin{\textcircled{\scriptsize x}} (\bar{d}_4 \mid x)) \mathbin{\textcircled{\scriptsize x}} ((y \mid \bar{d}_5) \mathbin{\textcircled{\scriptsize y}} d_6) \\ &= d_3 \mathbin{\textcircled{\scriptsize x}} (\bar{d}_4 \mid d_6 \mid \bar{d}_5) = (d_3 \mathbin{\textcircled{\scriptsize x}} (y \mid \bar{d}_5)) \mathbin{\textcircled{\scriptsize y}} ((\bar{d}_4 \mid x) \mathbin{\textcircled{\scriptsize x}} d_6) \\ &= (d_3 \mathbin{\textcircled{\scriptsize x}} d_5) \mathbin{\textcircled{\scriptsize y}} (d_4 \mathbin{\textcircled{\scriptsize x}} d_6) = d_1 \mathbin{\textcircled{\scriptsize y}} d_2. \end{aligned}$$

Spoiler could play the CMP move on (1), so $c'_1 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_4$ such that

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{24}$$
$$((d_4, \sigma), (d'_4, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{25}$$

Similarly, from (2), we have that $c'_2 = d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_6$ such that

$$((d_5, \sigma), (d'_5, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{26}$$
$$((d_6, \sigma), (d'_6, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{27}$$

Note that $\acute{x} \in fn(d'_3) \subseteq fn(c'_1)$ and $fn(d'_5) \subseteq fn(c'_2) \cup \{\acute{y}\}$. Furthermore, by Lemma 7, $\acute{y} \notin fn(d'_3)$, since $y \notin fn(d_3)$. Thus $d'_1 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_5$ is well defined. Similarly, $\acute{x} \in fn(d'_4) \subseteq fn(c'_1)$ and $fn(d'_6) \subseteq fn(c'_2)$, so $d'_2 = d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_6$ is well defined. Hence, by downward closure on (24), (26), (25) and (27), and by the inductive hypothesis, we have

$$((d_3 \mathbin{\textcircled{\scriptsize x}} d_5, \sigma), (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_5, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{28}$$

$$((d_4 \mathbin{\textcircled{\scriptsize x}} d_6, \sigma), (d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_6, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{29}$$

It remains to show that $c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$. By Lemma 1, $c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_4) \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_6) = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_6))$. Now suppose that $d_4 = \bar{d}_4 \mid x$ and $d_5 = y \mid \bar{d}_5$. By Lemma 8, we must have that $d'_4 = \bar{d}'_4 \mid \acute{x}$ and $d'_5 = \acute{y} \mid \bar{d}'_5$. Thus, $d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_6) = \bar{d}'_4 \mid \bar{d}'_6 \mid \bar{d}'_5 = d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} (d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_6)$. In the alternative case (where $d_4 = x \mid \bar{d}_4$ and $d_5 = \bar{d}_5 \mid y$) the analogous result can be deduced. Hence, and by Lemma 1 (recalling that $\acute{y} \notin fn(d'_3)$), $c'_1 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} (d'_5 \mathbin{\textcircled{\scriptsize $\acute{y}$}} (d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_6)) = (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_5) \mathbin{\textcircled{\scriptsize $\acute{y}$}} (d'_4 \mathbin{\textcircled{\scriptsize $\acute{x}$}} d'_6) = d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$, as required. We can see that Duplicator could respond to Spoiler's move by splitting c' as $d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$ and that, by (28) and (29), this gives her a winning strategy.

Case 4: Spoiler splits part of c_1 disjoint from c_2. There are two subcases on Spoiler's choice of y that we shall consider separately: (a) $y \neq x$ and (b) $y = x$.

(a) $y \neq x$:

$$c_1 \mathbin{\textcircled{\scriptsize x}} c_2 = (d_3 \mathbin{\textcircled{\scriptsize y}} d_2) \mathbin{\textcircled{\scriptsize x}} c_2 = (d_3 \mathbin{\textcircled{\scriptsize x}} c_2) \mathbin{\textcircled{\scriptsize y}} d_2 = d_1 \mathbin{\textcircled{\scriptsize y}} d_2$$
$$c_1 = d_3 \mathbin{\textcircled{\scriptsize y}} d_2 \qquad d_1 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2$$

Spoiler would be able to play the CMP move on the game in (1), so we know that $c'_1 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$ for some d'_3, d'_2 such that

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{30}$$

$$((d_2, \sigma), (d'_2, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{31}$$

Note that $fn(d'_3) \subseteq fn(c'_1) \cup \{\acute{y}\}$. Also, by Lemma 7, $\acute{x} \in fn(d'_3)$ and $\acute{y} \notin fn(c'_2)$. Thus $d'_1 = d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2$ is well defined. By downward closure on (30) and (2), and by the inductive hypothesis,

$$((d_3 \mathbin{\textcircled{\scriptsize x}} c_2, \sigma), (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{32}$$

By Lemma 2, since $\acute{x} \in fn(d'_3)$ and $\acute{y} \notin fn(c'_2)$, $(d'_3 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2) \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2 = (d'_3 \mathbin{\textcircled{\scriptsize $\acute{x}$}} c'_2) \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$. Hence, by (32) and downward closure on (31), we know that Duplicator has a winning strategy by splitting c' as $d'_1 \mathbin{\textcircled{\scriptsize $\acute{y}$}} d'_2$.

(b) $y = x$: For some $z \notin fn(c_1) \cup fn(c_2) \cup range(\sigma)$,

$$c = ((d_3 \mathbin{\textcircled{\scriptsize x}} d_2) \mathbin{\textcircled{\scriptsize z}} x) \mathbin{\textcircled{\scriptsize x}} c_2 = (d_3 \mathbin{\textcircled{\scriptsize x}} d_2) \mathbin{\textcircled{\scriptsize z}} c_2 = (d_3 \mathbin{\textcircled{\scriptsize z}} c_2) \mathbin{\textcircled{\scriptsize x}} d_2 = d_1 \mathbin{\textcircled{\scriptsize x}} d_2$$
$$c_1 = \bar{c}_1 \mathbin{\textcircled{\scriptsize z}} x \qquad \bar{c}_1 = d_3 \mathbin{\textcircled{\scriptsize x}} d_2 \qquad d_1 = d_3 \mathbin{\textcircled{\scriptsize z}} c_2.$$

By Lemma 9, for some $\acute{z} \notin fn(c'_1) \cup fn(c'_2) \cup range(\sigma')$,

$$((c_1, \sigma[\gamma \mapsto z]), (c'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-1, 0, \mathcal{L})) \in DW \tag{33}$$

$$((c_2, \sigma[\gamma \mapsto z]), (c'_2, \sigma'[\gamma \mapsto \acute{z}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{34}$$

Spoiler could play the CMP move on the game in (33), splitting c_1 as $\bar{c}_1 \textcircled{z} x$, so $c'_1 = \bar{c}'_1 \textcircled{\acute{z}} \hat{c}'_1$ such that

$$((\bar{c}_1, \sigma[\gamma \mapsto z]), (\bar{c}'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-2, 0, \mathcal{L})) \in DW \tag{35}$$

$$((x, \sigma[\gamma \mapsto z]), (\hat{c}'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-2, 0, \mathcal{L})) \in DW. \tag{36}$$

Since $3n - 2 \geq 1$, (36) implies that $\hat{c}'_1 = \acute{x}$. Spoiler could then play the CMP move on the game in (35), splitting $\bar{c}_1$ as $d_3 \textcircled{x} d_2$, so $\bar{c}'_1 = d'_3 \textcircled{\acute{x}} d'_2$ such that

$$((d_3, \sigma[\gamma \mapsto z]), (d'_3, \sigma'[\gamma \mapsto \acute{z}]), (3n-3, 0, \mathcal{L})) \in DW \tag{37}$$

$$((d_2, \sigma[\gamma \mapsto z]), (d'_2, \sigma'[\gamma \mapsto \acute{z}]), (3n-3, 0, \mathcal{L})) \in DW. \tag{38}$$

By construction and by Lemma 7 (recalling that $n \geq 2$), $\{\acute{x}, \acute{z}\} \subseteq fn(d'_3) \subseteq (fn(c') \setminus fn(c'_2)) \cup \{\acute{x}, \acute{z}\}$. Further, by Lemma 7 and by definition, neither $\acute{x}$ nor $\acute{z}$ occurs in c'_2. Hence $d'_1 = d'_3 \textcircled{\acute{z}} c'_2$ is well defined. Now we may apply the inductive hypothesis, using (37) and downward closure on (34), to obtain

$$((d_3 \textcircled{z} c_2, \sigma[\gamma \mapsto z]), (d'_3 \textcircled{\acute{z}} c'_2, \sigma[\gamma \mapsto \acute{z}]), (n-1, 0, \mathcal{L})) \in DW. \tag{39}$$

By (environment) downward closure on (39) and (38), we have

$$((d_1, \sigma), (d'_1, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{40}$$

$$((d_2, \sigma), (d'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{41}$$

Note that, by construction and by Lemma 7, $\acute{x}, \acute{z} \notin fn(d'_2)$ and $\acute{x} \notin fn(c'_2)$. Thus, by structural considerations and Lemma 2, $c' = ((d'_3 \textcircled{\acute{x}} d'_2) \textcircled{\acute{z}} \acute{x}) \textcircled{\acute{x}} c'_2 = (d'_3 \textcircled{\acute{x}} d'_2) \textcircled{\acute{z}} c'_2 = (d'_3 \textcircled{\acute{z}} c'_2) \textcircled{\acute{x}} d'_2 = d'_1 \textcircled{\acute{x}} d'_2$. Hence Duplicator could respond by splitting c' as $d'_1 \textcircled{\acute{x}} d'_2$ and by (40) and (41) that gives her a winning strategy.

We have considered all of the possible cases for how Spoiler could play CMP move, and shown that Duplicator has a winning response in each. Therefore, Duplicator has a winning strategy if Spoiler plays the CMP move.

EXS move. In playing this move, Spoiler chooses to instantiate β as y, say. If $n = 1$, any choice gives Duplicator a winning strategy, so assume $n \geq 2$. We consider four mutually exclusive cases for Spoiler's choice: 1. $y \in range(\sigma)$; 2. $y \in fn(c_1)$ but $y \notin range(\sigma)$; 3. $y \in fn(c_2)$ but $y \notin range(\sigma)$; and 4. y is fresh ($y \notin fn(c_1) \cup fn(c_2) \cup range(\sigma)$).

In case 1, $y = \sigma\alpha$ for some α, and Duplicator can respond with $\acute{y} = \sigma'\alpha$. By the first case of Lemma 9, we know

$$((c_1, \sigma[\beta \mapsto y]), (c'_1, \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW \tag{42}$$

$$((c_2, \sigma[\beta \mapsto y]), (c'_2, \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW \tag{43}$$

and so, by downward closure and the inductive hypothesis,

$$((c, \sigma[\beta \mapsto y]), (c', \sigma'[\beta \mapsto \acute{y}]), (n-1, 0, \mathcal{L})) \in DW. \tag{44}$$

Hence choosing $\acute{y}$ gives Duplicator a winning strategy in this case.

In case 2, note that Spoiler could play the EXS move on the game in (1). Let $\acute{y}$ be Duplicator's response for her winning strategy:

$$((c_1, \sigma[\beta \mapsto y]), (c_1', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{45}$$

Since $y \notin range(\sigma)$ and $3n - 2 \geq 2$, $\acute{y} \notin range(\sigma')$.[2] Also, since $y \in fn(c_1)$ and $3n - 2 \geq 2$, $\acute{y} \in fn(c_1')$ by Lemma 7. Thus, $y \notin fn(c_2) \cup range(\sigma)$ and $\acute{y} \notin fn(c_2') \cup range(\sigma')$, and hence, by the second case of Lemma 9,

$$((c_2, \sigma[\beta \mapsto y]), (c_2', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{46}$$

So by downward closure and the inductive hypothesis we have

$$((c, \sigma[\beta \mapsto y]), (c, \sigma'[\beta \mapsto \acute{y}]), (n-1, 0, \mathcal{L})) \in DW. \tag{47}$$

Hence choosing $\acute{y}$ gives Duplicator a winning strategy in this case.

Case 3 is essentially the same as case 2, except that Duplicator's choice, $\acute{y}$ is derived from her winning response for the game in (2). Case 4 admits the same proof as case 2 (or indeed case 3). Having examined each case, we see that Duplicator has a winning response to Spoiler playing the EXS move.

Since we have now examined each possible move Spoiler could make in the game of (3) and concluded that Duplicator has a winning strategy in each case, we have shown that (3) holds. □

Corollary 1 (Multi-step Move Elimination). *For all ranks $r = (n, s, \mathcal{L})$, for all $c, c' \in \mathcal{C}$ and for all domain-coincident environments σ, σ', if*

$$((c, \sigma), (c', \sigma'), (3^s(n+1), 0, \mathcal{L})) \in DW \tag{48}$$

then

$$((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW. \tag{49}$$

Proof (Sketch[3]). The proof is by induction on the number of adjunct moves, s. We suppose that Spoiler is trying to find a winning strategy for the game in (49) and see that the moves he makes in that game can be replicated on the game in (48) until he first plays one of the adjunct moves. When he plays his first adjunct move, he introduces a new context to either apply around one of the contexts in the current state, or to apply the current context to.

We find a response for Duplicator by renaming the holes of Spoiler's choice so that the application is defined for her side of the game and so that she has a winning strategy if Spoiler chooses to continue with these newly introduced contexts. Proposition 3 shows that Duplicator has a winning strategy for the composed pair with an adjunct-free rank. Now, we can use the inductive hypothesis to deduce that Duplicator has a winning strategy for the game with $s-1$ adjunct moves, as required. □

[2] To see this, suppose that Spoiler plays the CMP move and splits $c_1 = y \circledcirc{y} c_1$ (having played the EXS move as described). Duplicator could not have a winning strategy since there is some γ with $\acute{y} = \sigma'\gamma$ but $y \neq \sigma\gamma$.

[3] The full proof will appear in the full version of this paper.

These game results are now translated to results in the logic in the following theorem. The proof is not difficult (it depends on Lemma 4), and will appear in the full version of this paper.

Theorem 1 (Adjunct Elimination). *If* $r = (n, s, \mathcal{L})$ *and* $r' = (3^s(n+1), 0, \mathcal{L})$ *then, for any formula of rank* r*, there exists an equivalent formula of rank* r'*.*

5 Conclusions

We have introduced multi-holed Context Logic for trees (CL^m_{Tree}) and proved adjunct elimination. Our initial motivation was simply to understand if Lozes' results for Separation Logic and Ambient Logic extended to the original formulation of Context Logic. When we observed that this was not the case, this work turned from being a routine adaptation of previous results into a fundamental investigation of a natural version of Context Logic in which the adjoints could be eliminated.

Many open problems remain. We studied multi-holed Context Logic initially because we were unable to prove adjunct elimination for single-holed Context Logic with composition. We believe the result also holds for the single-holed case, but have not been able to prove it with current techniques. A further question, which would imply this result, is whether, in the absence of adjoints, multi-holed and and single-holed Context Logic with composition have equally expressive satisfaction relations on closed formulae for analysing trees (contexts without holes). This result appears to be difficult to prove.

Such results about expressivity on closed formulae form an important part of our investigation into the true nature of Context Logic for trees, not only because they provide a test on what is a natural formulation of Context Logic but also because they allow us to link our analysis of structured data (in this case trees) with traditional results about regular languages. For example, Heuter [12] has shown that a regular expression language, similar to multi-holed Context logic applied to *ranked* trees and without structural adjoints, is as expressive as First-order Logic (FOL) on ranked trees. Recently, Bojańczyk [13] has proved that a language equivalent to single-holed Context Logic for unranked trees, with composition but no adjoints, corresponds to FOL on forests. These results make use of the rich theory of formal languages, such as automata theory, which we hope to apply to CL^m_{Tree} to obtain a complete understanding of its place in the study of forest-regular languages.

An intriguing question (for which we thank one of the anonymous referees) is to what extent the adjoints permit properties of trees to be expressed succinctly. The results in this paper give an upper bound: given a formula with adjoints, a corresponding adjunct-free formula has maximum nesting depth of non-Boolean connectives that is exponential in the number of adjoint connectives of the original formula. The total number of connectives might still be large, although by Lemma 3 we know it is bounded. By refining our methods and studying examples, we expect to find closer bounds. It is not clear whether this will lead to tight bounds on how much more succinct formulae with adjoints can be.

Finally, we should mention Calcagno, Gardner and Zarfaty's recent work on *parametric* expressivity [7], which compares logics on *open* formulae containing propositional variables. Despite our expressivity results on *closed* formulae in this paper, stating that the adjoints can be eliminated, we intuitively know that adjunct connectives are important for expressing weakest preconditions for local Hoare reasoning using Separation Logic and Context Logic, and for expressing security properties in Ambient Logic. This intuition is formally captured in [7] where it is shown that the adjoints cannot be eliminated on open formulae. For our style of logical reasoning, both types of expressivity result seem to be important: the expressivity on open formulae captures our intuition that the structural connectives are important for modular reasoning; and the expressivity on closed formulae allows us to compare our reasoning about structured data with the literature on regular languages.

References

1. Ishtiaq, S.S., O'Hearn, P.W.: BI as an assertion language for mutable data structures. In: POPL (2001)
2. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: LICS (2002)
3. Yang, H., O'Hearn, P.W.: A semantic basis for local reasoning. In: Nielsen, M., Engberg, U. (eds.) ETAPS 2002 and FOSSACS 2002. LNCS, vol. 2303, Springer, Heidelberg (2002)
4. Cardelli, L., Gordon, A.D.: Anytime, anywhere: modal logics for mobile ambients. In: POPL (2000)
5. Calcagno, C., Gardner, P., Zarfaty, U.: Context logic and tree update. In: POPL (2005)
6. O'Hearn, P., Pym, D.: Logic of bunched implications. Bulletin of Symbolic Logic 5(2), 215–244 (1999)
7. Calcagno, C., Gardner, P., Zarfaty, U.: Context logic as modal logic: completeness and parametric inexpressivity. In: POPL (2007)
8. Lozes, E.: Adjunct elimination in the static Ambient Logic. In: EXPRESS (2003)
9. Dawar, A., Gardner, P., Ghelli, G.: Adjunct elimination through games in static ambient logic. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, Springer, Heidelberg (2004)
10. Calcagno, C., Gardner, P., Zarfaty, U.: Separation logic, ambient logic and context logic: parametric inexpressivity results (Unpublished, 2006)
11. Dinsdale-Young, T.: Adjunct elimination in context logic. Master's thesis, Imperial College London (2006)
12. Heuter, U.: First-order properties of trees, star-free expressions, and aperiodicity. Informatique théorique et applications 25(2), 125–145 (1991)
13. Bojańczyk, M.: Forest expressions. In: CSL (to appear, 2007)