Adjunct Elimination in Context Logic for Trees

Cristiano Calcagno Thomas Dinsdale-Young Philippa Gardner

Department of Computing, Imperial College, London, 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.

Key words: Context Logic, Adjunct Elimination, Ehrenfeucht-Fraïssé Games

1 Introduction

Separation Logic [1–3] and Ambient Logic [4] are related theories for reasoning, respectively, about heap update and static trees. Inspired by this work, Calcagno, Gardner and Zarfaty invented Context Logic [5] for reasoning about structured data. In particular, they used Context Logic applied to trees to reason locally about tree update, following the reasoning style of Separation Logic for reasoning locally about heap update. Such local reasoning is not possible using Ambient Logic [6].

All 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 [7] and then Dawar, Gardner and Ghelli [8] showed that the

Email addresses: ccris@doc.ic.ac.uk (Cristiano Calcagno), td202@doc.ic.ac.uk (Thomas Dinsdale-Young), pg@doc.ic.ac.uk (Philippa Gardner).

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 [9,6]. We expected an analogous result for Context Logic applied to trees, but instead found a counterexample (first reported in Dinsdale-Young's Masters thesis [10]).

Context Logic was originally 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. In Section 2, we present our counterexample to adjunct elimination for single-holed Context Logic. The key point is that, whereas structural composition reasons about trees by splitting them into contexts and trees, contexts cannot be split. One possible solution is simply to extend Context Logic with context composition and its corresponding adjoints. We do not know if adjunct elimination holds for this extension. We do know that current proof techniques cannot be immediately adapted. Instead, we prove an adjunct-elimination result for *multi-holed* Context Logic applied to trees, which provides a more general approach for splitting contexts.

Our adjunct-elimination result uses a technique based on Ehrenfeucht-Fraïssé games, which was first used to prove adjunct elimination for Ambient Logic in [8]. For Context Logic, this technique naturally requires 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. The structural composition 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.

We first published this adjunct-elimination result in the conference APLAS 2007 [11], although it does not contain most of the proofs. This journal paper provides the proofs, gives a more detailed account of adjunct elimination in the single-holed case (Section 2), where one adjoint can be removed and the other cannot, and provides a fuller account of multi-holed Context Logic (Section 3). We believe multi-holed Context Logic introduced 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.

Acknowledgements Calcagno acknowledges support of an EPSRC Advanced Fellowship. Dinsdale-Young acknowledges support of an EPSRC DTA award. Gardner acknowledges support of a Microsoft Research Cambridge/ Royal Academy of Engineering Senior Research Fellowship.

2 Single-holed Context Logic for Trees

In order to motivate our use of multi-holed Context Logic, we shall first summarise single-holed Context Logic for trees (CL^{s}_{Tree}) [5] and the known facts concerning adjunct elimination.

2.1 The Tree Model

We begin by defining the tree model which consists of finite, ordered, unranked trees and tree contexts. Throughout the paper, the nodes of trees are labelled from an infinite set of atoms, the set of *node labels* Σ, ranged over by u, v, w. [1]

In the literature, a distinction is often drawn between structures with a single root node, which are called 'trees', and structures with any number of roots, called 'forests'. Results in Context Logic, including those presented here, do not generally rely on this distinction, and so we use the term 'trees' to refer to structures with any number of root nodes.

Definition 1 (Trees and Tree Contexts)**.** The set of trees $\mathcal{T}$, ranged over by a, b, and the set of (single-holed) tree contexts $\mathcal{C}^{\mathrm{s}}$, ranged over by c, d, are defined as

$$\begin{array}{lll} a, b ::= & \varepsilon \mid u[a] \mid a_1 \mid a_2 & (u \in \Sigma) \\ c, d ::= & \_ \mid u[c] \mid a \mid c \mid c \mid a & (u \in \Sigma) \end{array}$$

modulo structural equivalences given by the '|' operators being mutually associative and having identity ε (the empty tree). The notation u is used to abbreviate $u[\varepsilon]$.

Definition 2 (Context Application)**.** Context application is a function, $ap : \mathcal{C}^{\mathrm{s}} \times \mathcal{T} \to \mathcal{T}$, defined inductively over the structure of contexts by

$$\begin{aligned} ap(\_, b) &= b \\ ap(u[c], b) &= u[ap(c, b)] \\ ap(a \mid c, b) &= a \mid ap(c, b) \\ ap(c \mid a, b) &= ap(c, b) \mid a. \end{aligned}$$

The notation $c(a)$ is used to abbreviate $ap(c, a)$. Note that $\_$ is the left identity of ap.

[1] We assume that the elements of Σ are distinct from all other constants introduced in this paper.

2.2 Single-holed Context Logic

Context Logic [5] was introduced by Calcagno, Gardner and Zarfaty to reason about structured data (for example, trees), in contrast with Bunched Logic of O'Hearn and Pym [12] which reasons about unstructured resource (for example, heaps). Using Context Logic, it is possible to provide local Hoare reasoning about tree update, following O'Hearn, Reynolds and Yang's work on local Hoare reasoning about heap update [1–3]. The key observation in [5] was that local data update typically identifies the portion of data to be replaced, removes it, and inserts new data *in the same place*. Context Logic was therefore introduced to reason about both data and this place of insertion (contexts).

We now define single-holed Context Logic applied to trees, denoted CL^{s}_{Tree}. Our definition follows a similar pattern to the definitions of Separation Logic and Ambient Logic. It extends the propositional connectives of classical logic with general *structural* connectives for analysing the structure of single-holed contexts, and *specific* connectives for analysing the particular model under consideration (in this case, trees and tree contexts).

Definition 3 (Formulae of CL^{s}_{Tree})**.** Single-holed Context Logic for trees consists of the set of *tree formulae*, $\mathcal{P}^{\mathrm{s}}$, ranged over by P, P_1, P_2, and the set of *context formulae*, $\mathcal{K}^{\mathrm{s}}$, ranged over by K, K_1, K_2. These sets are defined by:

$$
\begin{array}{rlll}
P ::= & 0 & & \text{tree-specific formulae} \\
 & K(P) \mid K \lhd P & & \text{structural formulae} \\
 & \mathit{false} \mid P_1 \Rightarrow P_2 & & \text{Boolean formulae} \\
K ::= & u[K] \mid P \mid K \mid K \mid P & (u \in \Sigma) & \text{tree-specific formulae} \\
 & \mathbf{I} \mid P_1 \rhd P_2 & & \text{structural formulae} \\
 & \mathit{False} \mid K_1 \Rightarrow K_2 & & \text{Boolean formulae.}
\end{array}
$$

The structural formulae relate to the single-holed context application operation. The formula $\mathbf{I}$ specifies that a context is the context hole, $\_$. The application formula $K(P)$ specifies that a tree can be viewed as the application of some context satisfying K to some tree satisfying P. The connectives '$\lhd$' and '$\rhd$' are the right *adjoints* of application. The formula $K \lhd P$ is satisfied by a tree if, whenever any context satisfying K is applied to it, the result satisfies P. Similarly, the formula $P_1 \rhd P_2$ is satisfied by a context if, whenever it is applied to a tree satisfying P_1, the result satisfies P_2. The specific connectives are used to express additional tree-specific properties of trees and tree contexts. The formula 0 specifies that a tree is the empty tree, ε. The formula $u[K]$ specifies that a tree context has a single topmost node labelled u above a context that satisfies the formula K. The formula $K \mid P$ specifies that a context is the horizontal concatenation of a context satisfying K and a tree satisfying P; the formula $P \mid K$ has an analogous interpretation. In this paper, we deal with basic Context Logic without quantification over node labels.

The formal semantics of the formulae is given by the satisfaction relations described

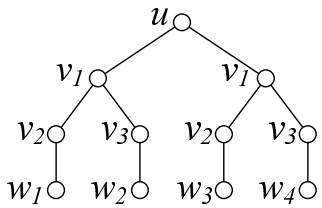

Fig. 1. The tree $a = u[v_1[v_2[w_1] \mid v_3[w_2]] \mid v_2[v_2[w_3] \mid v_3[w_4]]]$

below.

Definition 4 (Satisfaction Relations of CL^{s}_{Tree})**.** The *satisfaction relations*, $\models_{\mathcal{P}^{\mathrm{s}}} \subseteq \mathcal{T} \times \mathcal{P}^{\mathrm{s}}$ denoting the satisfaction of a tree formula by a tree, and $\models_{\mathcal{K}^{\mathrm{s}}} \subseteq \mathcal{C}^{\mathrm{s}} \times \mathcal{K}^{\mathrm{s}}$ denoting the satisfaction of context formula by a tree context, are defined inductively on the structure of the formulae by

$a \models_{\mathcal{P}^{\mathrm{s}}} 0$	iff	$a = \varepsilon$
$a \models_{\mathcal{P}^{\mathrm{s}}} K(P)$	iff	there exist $c \in \mathcal{C}^{\mathrm{s}}, b \in \mathcal{T}$ s.t. $a = c(b)$ and $c \models_{\mathcal{K}^{\mathrm{s}}} K$ and $b \models_{\mathcal{P}^{\mathrm{s}}} P$
$a \models_{\mathcal{P}^{\mathrm{s}}} K \lhd P$	iff	for all $c \in \mathcal{C}^{\mathrm{s}}, b \in \mathcal{T}$, $b = c(a)$ and $c \models_{\mathcal{K}^{\mathrm{s}}} K$ implies $b \models_{\mathcal{P}^{\mathrm{s}}} P$
$a \not\models_{\mathcal{P}^{\mathrm{s}}} \mathit{false}$		
$a \models_{\mathcal{P}^{\mathrm{s}}} P_1 \Rightarrow P_2$	iff	$a \models_{\mathcal{P}^{\mathrm{s}}} P_1$ implies $a \models_{\mathcal{P}^{\mathrm{s}}} P_2$
$c \models_{\mathcal{K}^{\mathrm{s}}} u[K]$	iff	there exists $d \in \mathcal{C}^{\mathrm{s}}$ s.t. $c = u[d]$ and $d \models_{\mathcal{K}^{\mathrm{s}}} K$
$c \models_{\mathcal{K}^{\mathrm{s}}} P \mid K$	iff	there exist $a \in \mathcal{T}, d \in \mathcal{C}^{\mathrm{s}}$ s.t. $c = a \mid d$ and $a \models_{\mathcal{P}^{\mathrm{s}}} P$ and $d \models_{\mathcal{K}^{\mathrm{s}}} K$
$c \models_{\mathcal{K}^{\mathrm{s}}} K \mid P$	iff	there exist $a \in \mathcal{T}, d \in \mathcal{C}^{\mathrm{s}}$ s.t. $c = d \mid a$ and $a \models_{\mathcal{P}^{\mathrm{s}}} P$ and $d \models_{\mathcal{K}^{\mathrm{s}}} K$
$c \models_{\mathcal{K}^{\mathrm{s}}} \mathbf{I}$	iff	$c = \_$
$c \models_{\mathcal{K}^{\mathrm{s}}} P_1 \rhd P_2$	iff	for all $a, b \in \mathcal{T}$, $a = c(b)$ and $b \models_{\mathcal{P}^{\mathrm{s}}} P_1$ implies $a \models_{\mathcal{P}^{\mathrm{s}}} P_2$
$c \not\models_{\mathcal{K}^{\mathrm{s}}} \mathit{False}$		
$c \models_{\mathcal{K}^{\mathrm{s}}} K_1 \Rightarrow K_2$	iff	$c \models_{\mathcal{K}^{\mathrm{s}}} K_1$ implies $c \models_{\mathcal{K}^{\mathrm{s}}} K_2$

We use the Boolean connectives '*false*', '*False*' and '$\Rightarrow$'. The other standard connectives, '*true*', '*True*', '$\neg$', '$\wedge$' and '$\vee$', are derivable. We also use the following derived formulae: $u[P] \triangleq u[\mathbf{I}](P)$, $P_1 \mid P_2 \triangleq (P_1 \mid \mathbf{I})(P_2)$, $K \blacktriangleleft P \triangleq \neg(K \lhd \neg P)$ and $P_1 \blacktriangleright P_2 \triangleq \neg(P_1 \rhd \neg P_2)$. The first derived formula has the expected meaning, that a forest consists of a root node labelled u above a forest satisfying P. The second is also intuitive, meaning that a forest is the horizontal concatenation of two forests with the left satisfying P_1 and the right P_2. The others are negation duals of the application right adjoints, which have an existential interpretation. For instance, $K \blacktriangleleft P$ is satisfied by a tree if there exists a context satisfying K which may be applied to it to produce a tree satisfying P.

We assume the following binding precedence among the connectives, with '$\neg$' binding tightest: '$\neg$', '$-(-)$', '$\mid$', '$\wedge$', '$\vee$', {'$\rhd$', '$\lhd$', '$\blacktriangleright$', '$\blacktriangleleft$'}, '$\Rightarrow$', with the braces denoting that there is no precedence between '$\rhd$', '$\lhd$', '$\blacktriangleright$' and '$\blacktriangleleft$'.

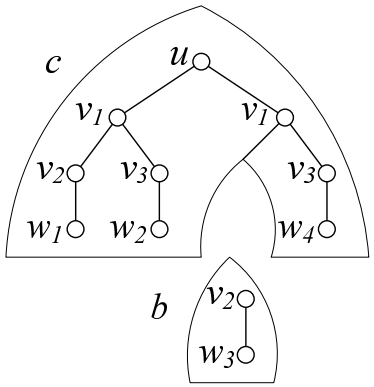

Fig. 2. Illustration of how the tree a may be split: $a = ap(c, b)$

Let us consider a few examples to illustrate the logic.

Example 5. Let $a = u[v_1[v_2[w_1]|v_3[w_2]]|v_1[v_2[w_3]|v_3[w_4]]]$, as illustrated in Figure 1. Then we have the following:

$$a \models_{\mathcal{P}^s} u[true \mid v_1[v_2[w_3[0]] \mid true] \mid true] \tag{1}$$
$$a \models_{\mathcal{P}^s} True(v_2[w_3[0]]) \tag{2}$$
$$a \not\models_{\mathcal{P}^s} u[True(v_2[w_5[0]])] \tag{3}$$
$$a \models_{\mathcal{P}^s} (w_6[0] \rhd True(v_2[w_1[0]] \mid v_3[w_6[0]]))(w_2[0]). \tag{4}$$

The first example expresses that the tree a has root node labelled u with a child labelled v_1 whose first child consists of a tree of the form $v_2[w_3]$. By examination, this can be seen to be true. The second example expresses that a has some subtree of the form $v_2[w_3]$. Figure 2 illustrates how a may be split into context c and tree b such that $c \models_{\mathcal{K}^s} True$ and $b \models_{\mathcal{P}^s} v_2[w_3[0]]$ as required for (2). In the third example, the formula is not satisfied by a since the tree cannot be split apart as required; in particular, there is no node labelled w_5. The fourth example illustrates how the adjoints are used to express hypothetical statements about trees. The formula is satisfied because, when a subtree satisfying $w_2[0]$ is replaced by a tree satisfying $w_6[0]$, the resulting tree satisfies $True(v_2[w_1[0]] \mid v_3[w_6[0]])$.

2.3 Adjunct elimination

Dinsdale-Young studied adjunct elimination results for CL^{s}_{Tree} in his Masters' thesis [10]. He proved the following two results.

Theorem 6. *For every tree and context formula of CL^{s}_{Tree}, there is a logically equivalent formula which does not make use of the '$\lhd$'-connective.*

Theorem 7. *There is no adjunct-free formula of CL^{s}_{Tree} that is logically equivalent to the context formula $0 \rhd True\,(u[0])$.*

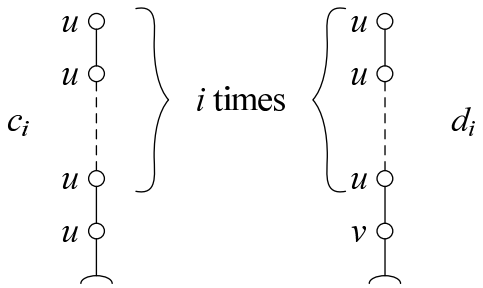

Fig. 3. Structure of the contexts c_i and d_i

The first result is a positive elimination result, that the '$\triangleleft$'-connective adds no expressive power to the logic. The proof is based on Ehrenfeucht-Fraïssé games, a technique we shall use later in this paper. It adapts the method introduced by Dawar et al. in [8] to prove adjunct elimination for Ambient Logic, whose adjuncts are essentially special cases of Context Logic's '$\triangleleft$'-connective. Indeed, in light of their paper and Lozes earlier work [7], this elimination result is not surprising.

The second result shows that elimination of the other right adjoint is not possible; that is, the '$\triangleright$'-connective does add some expressive power. Since no analogue of the '$\triangleright$'-connective exists in Ambient Logic, we have less expectation of a positive elimination result *a priori*. However, on an intuitive level we may find this result surprising. The adjunct connective, in essence, allows one to specify properties of a context in terms of the result of applying it to specified trees. The result of this application is a tree that incorporates the original context, and is, in a sense, bigger. Intuitively, we should not be able to specify any additional properties of the context in terms of this tree.

However, Theorem 7 shows that this is not the case. We present a short proof of this theorem.

Proof of Theorem 7. Let $K = 0 \triangleright \mathit{True}\,(u[0])$. Define contexts c_i and d_i (illustrated in Figure 3) by

$$\begin{aligned} c_0 &= u[\_] & \qquad c_{i+1} &= u[c_i] \\ d_0 &= v[\_] & \qquad d_{i+1} &= u[d_i]. \end{aligned}$$

Observe that, for all i, $c_i \models_{\mathcal{K}^{\mathrm{s}}} K$ whilst $d_i \not\models_{\mathcal{K}^{\mathrm{s}}} K$. We shall prove, by induction on the structure of formulae, that no adjunct-free formula shares this property. In particular, we shall prove that, for any given adjunct-free formula K', there is an n such that, for $j \geq n$, $c_j \models_{\mathcal{K}^{\mathrm{s}}} K'$ if and only if $d_j \models_{\mathcal{K}^{\mathrm{s}}} K'$.

In the base case, $K' = \mathbf{I}$ or $K' = \mathit{False}$. It is immediate that $n = 0$ is adequate for both cases, since $c_i \not\models_{\mathcal{K}^{\mathrm{s}}} K'$ and $d_i \not\models_{\mathcal{K}^{\mathrm{s}}} K'$ for all i.

In the inductive case, we consider the different cases for the structure of K'. Where $K' = w[K'']$, either $w = u$ or K' is not satisfied by c_i or d_i for any choice of $i > 0$ (and hence $n = 1$ works). Assuming the former, let n' be the value given by the inductive hypothesis such that, for $j \geq n'$, $c_j \models_{\mathcal{K}^{\mathrm{s}}} K''$ if and only if $d_j \models_{\mathcal{K}^{\mathrm{s}}} K''$.

Observe that $c_{i+1} \models_{\mathcal{K}^s} K'$ if and only if $c_i \models_{\mathcal{K}^s} K''$, and $d_{i+1} \models_{\mathcal{K}^s} K'$ if and only if $d_i \models_{\mathcal{K}^s} K''$. Therefore, selecting $n = n' + 1$ works.

In the case where $K' = P \mid K''$ or $K' = K'' \mid P$, c_i and d_i can only satisfy K' when they also satisfy K'', since any horizontal splitting of c_i or d_i into a context and tree will give the empty tree. Let n' be the value given by the inductive hypothesis for K''. Choosing $n = n'$ is sufficient for the result to hold. In the case where $K' = K_1 \vee K_2$, choosing n to be the maximum of n_1 and n_2 (where these values are given for K_1 and K_2 by the inductive hypothesis), is sufficient. Similarly, in the case where $K' = \neg K''$, choosing $n = n'$ suffices.

Thus, there is no adjunct-free formula equivalent to K. □

The key property of the logic that is exploited in Theorem 7 is that trees can be split arbitrarily while contexts cannot. Thus, inserting the empty tree into a context results in a tree which can be split in such a way that properties can be identified close to the context hole, regardless of how deep it was within the original context.

It should be noted that the counterexample does not show whether there is some *tree* formula that has no adjunct-free equivalent, which remains an open question. This problem is interesting, since contexts are in some sense a by-product of reasoning about trees. Yet it is difficult, since the counterexample from Theorem 7 precludes a direct inductive proof that '$\triangleright$' is eliminable from tree formulae, since it may not be eliminable from subformulae.

2.4 Single-holed Context Logic with Composition

A natural extension of single-holed Context Logic is to include connectives for reasoning about context composition: that is, a composition connective — enabling a context to be specified as the result of inserting one context into the context hole of another, much like the application connective for trees — plus the corresponding adjoints. This composition connective allows us to split contexts in a much more arbitrary fashion, without using '$\triangleright$'. As we might expect, this means that the counterexample to '$\triangleright$'-elimination for CL^{s}_{Tree} no longer applies. Indeed, adjunct elimination for this extended logic is an open problem.

Before we discuss the implications of this extension further, let us formalise it. We work with the same trees and contexts as before. The composition function, $cp : \mathcal{C}^{\mathrm{s}} \times \mathcal{C}^{\mathrm{s}} \to \mathcal{C}^{\mathrm{s}}$, is defined analogously to ap.

Definition 8 (Context Composition)**.** Context composition is a function, $cp : \mathcal{C}^{\mathrm{s}} \times$

$\mathcal{C}^{\mathrm{s}} \to \mathcal{C}^{\mathrm{s}}$, defined inductively on the structure of contexts by

$$\begin{aligned} cp(\_, c') &= c' \\ cp(u[c], c') &= u[cp(c, c')] \\ cp(a \mid c, c') &= a \mid cp(c, c') \\ cp(c \mid a, c') &= cp(c, c') \mid a. \end{aligned}$$

Note that $\_$ is the two-sided identity of cp.

We define single-holed Context Logic for trees with composition, denoted CL^{c}_{Tree}, by extending CL^{s}_{Tree} with a composition connective and its corresponding right adjoints.

Definition 9 (Formaule of CL^{c}_{Tree})**.** Single-holed Context Logic for trees with composition consists of the set of *tree formulae*, $\mathcal{P}^{\mathrm{c}}$, ranged over by P, P_1, P_2, and the set of *context formulae*, $\mathcal{K}^{\mathrm{c}}$, ranged over by K, K_1, K_2. These definitions are mutually recursive: the set $\mathcal{P}^{\mathrm{c}}$ is constructed as for CL^{s}_{Tree}; the set $\mathcal{K}^{\mathrm{c}}$ is defined by:

$$\begin{array}{lll} K ::= u[K] \mid P \mid K \mid K \mid P & (u \in \Sigma) & \text{tree-specific formulae} \\ \quad \mathbf{I} \mid P_1 \rhd P_2 & & \text{structural formulae} \\ \quad \mathit{False} \mid K_1 \Rightarrow K_2 & & \text{Boolean formulae} \\ \quad K_1 \circ K_2 \mid K_1 \circ\!\!- K_2 \mid K_1 -\!\!\circ K_2 & & \text{composition formulae.} \end{array}$$

Definition 10 (Satisfaction Relations of CL^{c}_{Tree})**.** The *satisfaction relations* for CL^{c}_{Tree}, $\models_{\mathcal{P}^{\mathrm{c}}} \subseteq \mathcal{T} \times \mathcal{P}^{\mathrm{c}}$ denoting the satisfaction of a tree formula by a tree, and $\models_{\mathcal{K}^{\mathrm{c}}} \subseteq \mathcal{C}^{\mathrm{s}} \times \mathcal{K}^{\mathrm{c}}$ denoting the satisfaction of a context formula by a tree context, are defined as for CL^{s}_{Tree}, but with the following additional inductive rules for the new connectives:

$$\begin{array}{lll} c \models_{\mathcal{K}^{\mathrm{c}}} K_1 \circ K_2 & \text{iff} & \text{there exist } c_1, c_2 \in \mathcal{C}^{\mathrm{s}} \text{ s.t.} \\ & & \quad c = cp(c_1, c_2) \text{ and } c_1 \models_{\mathcal{K}^{\mathrm{c}}} K_1 \text{ and } c_2 \models_{\mathcal{K}^{\mathrm{c}}} K_2 \\ c \models_{\mathcal{K}^{\mathrm{c}}} K_1 \circ\!\!- K_2 & \text{iff} & \text{for all } c_1, c_2 \in \mathcal{C}^{\mathrm{s}}, \\ & & \quad c_2 = cp(c_1, c) \text{ and } c_1 \models_{\mathcal{K}^{\mathrm{c}}} K_1 \text{ implies } c_2 \models_{\mathcal{K}^{\mathrm{c}}} K_2 \\ c \models_{\mathcal{K}^{\mathrm{c}}} K_1 -\!\!\circ K_2 & \text{iff} & \text{for all } c_1, c_2 \in \mathcal{C}^{\mathrm{s}}, \\ & & \quad c_2 = cp(c, c_1) \text{ and } c_1 \models_{\mathcal{K}^{\mathrm{c}}} K_1 \text{ implies } c_2 \models_{\mathcal{K}^{\mathrm{c}}} K_2. \end{array}$$

With context composition at our disposal, we can start to see how the problems we had earlier in showing adjunct elimination start to break down. For example, the formula $0 \rhd True(u[0])$ is equivalent to the adjunct-free formula

$$True \circ (u[\mathbf{I}] \vee (True(u[0]) \mid True) \vee (True \mid True(u[0]))).$$

Although our original counterexample no longer disproves adjunct elimination, as we have seen, we might expect that there are others which do. However, let us consider our earlier intuition that the '$\rhd$'-connective simply turns a context into a larger tree, and hence does not make it easier to discriminate between two contexts.

The trick of the counterexample was that we had a powerful tool for describing trees that we did not have for describing contexts, namely the arbitrary splitting into context and tree. This meant that the '$\triangleright$'-connective could be used indirectly to describe how a context could be split. Context composition allows us to split contexts directly, so the advantage of shifting from context reasoning to tree reasoning is apparently mitigated. Our original intuition, then, is more likely to be valid.

Despite this, there is an expressive subtlety: application allows us to reason about a subtree at any point in a tree, whereas context composition is limited to reasoning about subcontexts. To exemplify this, consider that we can express that the tree $u[w \mid u[v \mid u]]$ has subtree u in terms of application as

$$u[w \mid u[v \mid u]] = ap(u[w \mid u[v \mid \_]], u),$$

but there is no directly analogous way to express that u is a subtree of the context $u[w[\_] \mid u[v \mid u]]$. Such a direct analogue would require the use of two-holed contexts.

We can, however, express such properties indirectly. A subtree of a context has a lowest common ancestor with the context hole. The subcontext at this ancestor can be viewed as the horizontal concatenation of a context and a tree that contains the subtree of interest. Thus, the overall context is the composition of a context with the concatenation of a context and the application of a context to the tree of interest. In our example:

$$u[w[\_] \mid u[v \mid u]] = cp(u[\_], w[\_] \mid ap(u[v \mid \_], u)).$$

As long as we can express the potential properties of the two-holed context in terms of the properties of the component contexts, the '$\triangleright$'-connective no longer seems necessary for describing such splittings.

However, our attempts to prove adjunct elimination using games for single-holed Context Logic with composition are thwarted by this issue. The crux of the problem is that the complex splitting required to express the subtree of the context means that the inductive hypothesis is not strong enough to show that the required properties of the composition hold. Instead, we prove adjunct elimination for the multi-holed case.

3 Multi-holed Context Logic for Trees

We now present multi-holed Context Logic for trees, CL^{m}_{Tree}. Our definitions arise as extensions of those of CL^{c}_{Tree}. As before, we work with finite, ordered, unranked trees with nodes labelled from the set Σ. The difference is that contexts may now have multiple context holes, each labelled distinctly from an infinite set of atoms, the set of *hole labels* X, ranged over by $x, y, z, \ldots$. We assume that the sets X and Σ are disjoint. Since our contexts can have any number of holes, in CL^{m}_{Tree} we consider trees to be exactly the contexts with no context holes, and therefore do not explicitly distinguish them in our definitions.

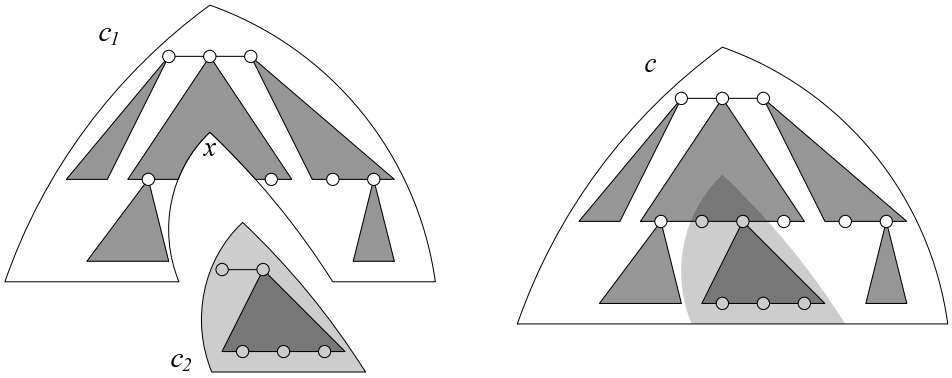

Fig. 4. An illustration of how contexts c_1 and c_2 combine to form $c = c_1 \circledast c_2$

Definition 11 (Multi-holed Tree Contexts)**.** The set of multi-holed tree contexts, denoted $\mathcal{C}^{\mathrm{m}}$ and ranged over by c, d, is defined by

$$c ::= \quad \varepsilon \;\mid\; x \;\mid\; u[c] \;\mid\; c_1 \mid c_2 \quad (x \in X, u \in \Sigma)$$

with the restriction that each hole label, $x \in X$, occurs *at most once* in the context c, and subject to the '|' operator being associative and having identity ε. The set of hole labels that occur in c, denoted $fn(c)$, is defined inductively by

$$\begin{aligned} fn(\varepsilon) &= \emptyset \\ fn(x) &= \{x\} \\ fn(u[c]) &= fn(c) \\ fn(c_1 \mid c_2) &= fn(c_1) \cup fn(c_2). \end{aligned}$$

The notation u is used as an abbreviation of $u[\varepsilon]$.

Definition 12 (Substitution)**.** The substitution of hole label $x \in X$ by context $c_2 \in \mathcal{C}^{\mathrm{m}}$ in context $c_1 \in \mathcal{C}^{\mathrm{m}}$, denoted $c_1[c_2/x]$, is defined as follows:

$$\begin{aligned} \varepsilon[c_2/x] &= \varepsilon \\ x[c_2/x] &= c_2 \\ y[c_2/x] &= y \quad \text{where } y \neq x \\ (u[c_1])[c_2/x] &= u[(c_1[c_2/x])] \\ (c_1 \mid c_1')[c_2/x] &= (c_1[c_2/x]) \mid (c_1'[c_2/x]). \end{aligned}$$

Definition 13 (Context Composition)**.** *Context composition* is a set of partial functions indexed by hole labels, $cp_x : \mathcal{C}^{\mathrm{m}} \times \mathcal{C}^{\mathrm{m}} \rightharpoonup \mathcal{C}^{\mathrm{m}}$, defined by

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

The notation $c_1 \circledast c_2$ is used as an abbreviation of $cp_x(c_1, c_2)$. Figure 4 shows an example of context application. Note that x is the two-sided identity of '$\circledast$'.

This definition of multi-holed context seems to be the most appropriate for our reasoning style, since it allows contexts to be separated easily. A similar definition is used in [13] to study the expressivity of First-Order Logic on ranked trees. 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 14. 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. $c_1 = u[x \mid u[u \mid v]] \mid v \circledcirc{x} u[v]$. It may also be expressed as a two-holed context applied to two trees, e.g. $c_1 = (u[x \mid u[u \mid y]] \mid v \circledcirc{y} v) \circledcirc{x} u[v]$. Recall that the context holes are labelled uniquely by x and y, with the first application $u[x \mid u[u \mid y]] \mid v \circledcirc{y} v$ declaring that the argument v should be placed in the hole labelled y. This means 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.

The following two lemmata, which are readily checked, state useful properties of context composition.

Lemma 15. *For $x, y \in X$ and $c_1, c_2, c_3 \in \mathcal{C}^{\mathrm{m}}$, if $y = x$ or $y \notin fn(c_1)$, then $c_1 \circledcirc{x} (c_2 \circledcirc{y} c_3) = (c_1 \circledcirc{x} c_2) \circledcirc{y} c_3$, where defined.*

Lemma 16. *For $x, y \in X$ and $c_1, c_2, c_3 \in \mathcal{C}^{\mathrm{m}}$, if $y \neq x$, $x, y \in fn(c_1)$, $y \notin fn(c_2)$, $x \notin fn(c_3)$, then*

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

We define multi-holed Context Logic for trees, CL^{m}_{Tree}. As for the single-holed case, we extend the propositional connectives of classical logic with structural connectives for analysing multi-holed contexts, and specific connectives for analysing tree contexts.

The formulae of CL^{m}_{Tree} use variables for hole labels. The variable names are taken from an infinite set of atoms, the set of *hole variables* Θ, ranged over by α, β, γ.

Definition 17 (Formulae of CL^{m}_{Tree})**.** Multi-holed Context Logic for trees consists of the set of CL^{m}_{Tree} *formulae*, $\mathcal{K}^{\mathrm{m}}$, ranged over by K, K_1, K_2. This set is defined by:

$$\begin{array}{lll}
K ::= 0 \mid u[K] \mid K_1 \mid K_2 & (u \in \Sigma) & \text{tree-specific formulae} \\
\quad \alpha \mid K_1 \circ_\alpha K_2 \mid K_1 \circ\!\!-_\alpha K_2 & (\alpha \in \Theta) & \text{structural formulae} \\
\quad K_1 -\!\!\circ_\alpha K_2 \mid \exists \alpha.\, K & & \\
\quad \mathit{False} \mid K_1 \Rightarrow K_2 & & \text{Boolean formulae.}
\end{array}$$

As in CL^{s}_{Tree}, we use the Boolean connectives '*False*' and '$\Rightarrow$'. 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 connective '$\circ_\alpha$' specifies that a context is a composition of two contexts where the hole being filled is the value of α. The connectives '$\circ\!\!-_\alpha$'

and '$-\!\circ_\alpha$' are the right 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. Finally, the tree-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.

Definition 18 (Environment)**.** An *environment* is a finite partial function $\sigma : \Theta \rightharpoonup_{\mathit{fin}} X$ which assigns hole labels to hole variables. The set of environments is ranged over by σ, ρ.

The empty environment is denoted by $\emptyset$, and the extension of σ with a new domain element α with value y is denoted by $\sigma[\alpha \mapsto y]$. The domain of definition of σ is defined as

$$\mathit{dom}(\sigma) = \{\alpha \in \Theta : \sigma(\alpha) \text{ is defined}\},$$

and we call σ and σ' *domain-coincident* if $\mathit{dom}(\sigma) = \mathit{dom}(\sigma')$. The range of σ is defined as

$$\mathit{range}(\sigma) = \{x \in X : \text{there exists } \alpha \in \Theta \text{ s.t. } x = \sigma(\alpha)\}.$$

We denote by $\sigma[y/x]$ the environment with

$$\sigma[y/x](\alpha) = \begin{cases} y & \text{if } \sigma(\alpha) = x \\ \sigma(\alpha) & \text{otherwise} \end{cases}$$

for all $\alpha \in \Theta$.

Whereas $CL^{\mathrm{s}}_{\mathit{Tree}}$ did not include variables in the logic, $CL^{\mathrm{m}}_{\mathit{Tree}}$ does. Consequently, the satisfaction judgement for $CL^{\mathrm{m}}_{\mathit{Tree}}$ takes into account the valuation of the free variables of a formula given by an environment.

Definition 19 (Satisfaction relation of $CL^{\mathrm{m}}_{\mathit{Tree}}$)**.** The *satisfaction relation*, $\models \subseteq (\mathcal{C}^{\mathrm{m}} \times (\Theta \rightharpoonup_{\mathit{fin}} X)) \times \mathcal{K}^{\mathrm{m}}$ denoting the satisfaction of a formula by a context with respect to an environment, is defined inductively on the structure of the formulae by

$$\begin{array}{lll} c, \sigma \models 0 & \text{iff} & c = \varepsilon \\ c, \sigma \models u[K] & \text{iff} & \text{there exists } c' \in \mathcal{C}^{\mathrm{m}} \text{ s.t. } c = u[c'] \text{ and } c', \sigma \models K \\ c, \sigma \models K_1 \mid K_2 & \text{iff} & \text{there exist } c_1, c_2 \in \mathcal{C}^{\mathrm{m}} \text{ s.t.} \\ & & \qquad c = c_1 \mid c_2 \text{ and } c_1, \sigma \models K_1 \text{ and } c_2, \sigma \models K_2 \end{array}$$

$$
\begin{array}{lll}
c, \sigma \models \alpha & \text{iff} & c = x \\
c, \sigma \models K_1 \circ_\alpha K_2 & \text{iff} & \text{there exist } c_1, c_2 \in \mathcal{C}^{\mathrm{m}} \text{ s.t.} \\
 & & \qquad c = c_1 \circledast c_2 \text{ and } c_1, \sigma \models K_1 \text{ and } c_2, \sigma \models K_2 \\
c, \sigma \models K_1 \circ\!\!- _\alpha K_2 & \text{iff} & \text{for all } c_1, c_2 \in \mathcal{C}^{\mathrm{m}}, \\
 & & \qquad c_2 = c_1 \circledast c \text{ and } c_1, \sigma \models K_1 \text{ implies } c_2, \sigma \models K_2 \\
c, \sigma \models K_1 -\!\!\circ_\alpha K_2 & \text{iff} & \text{for all } c_1, c_2 \in \mathcal{C}^{\mathrm{m}}, \\
 & & \qquad c_2 = c \circledast c_1 \text{ and } c_1, \sigma \models K_1 \text{ implies } c_2, \sigma \models K_2 \\
c, \sigma \models \exists \alpha.\, K & \text{iff} & \text{there exists } y \in X \text{ s.t. } c, \sigma[\alpha \mapsto y] \models K \\
c, \sigma \not\models \mathit{False} & & \\
c, \sigma \models K_1 \Rightarrow K_2 & \text{iff} & c, \sigma \models K_1 \text{ implies } c, \sigma \models K_2.
\end{array}
$$

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: 'True', '$\neg$', '$\wedge$', '$\vee$', '$\forall\alpha$'. As in $CL^{\mathrm{s}}_{\mathit{Tree}}$, we define negation duals for the adjunct connectives: $K_1 \bullet\!\!- _\alpha K_2 \triangleq \neg(K_1 \circ\!\!- _\alpha \neg K_2)$ and $K_1 -\!\!\bullet_\alpha K_2 \triangleq \neg(K_1 -\!\!\circ_\alpha \neg K_2)$.

We assume the following binding order among the connectives, with '$\neg$' binding the tightest: '$\neg$', '$|$', '$\circ_\alpha$', '$\wedge$', '$\vee$', {'$\circ\!\!- _\alpha$', '$-\!\!\circ_\alpha$', '$\bullet\!\!- _\alpha$', '$-\!\!\bullet_\alpha$'}, '$\Rightarrow$', '$\exists\alpha$', '$\forall\alpha$', with the braces denoting that there is no precedence between '$\circ\!\!- _\alpha$', '$-\!\!\circ_\alpha$', '$\bullet\!\!- _\alpha$' and '$-\!\!\bullet_\alpha$'.

Example 20. We present a few example formulae:

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

(2) The formula $\exists\alpha.\,(\mathit{True} \circ_\alpha u[0])$ expresses that a context contains tree $u[\varepsilon]$.

(3) The formula $(\mathit{True} \circ_\alpha \alpha)$ expresses that a hole labelled with the value of α must be in the context.

(4) The formulae $\exists\alpha.\,(0 -\!\!\circ_\alpha u[0]) \circ_\alpha \mathit{True}$ and $u[\mathit{True}] \vee (\mathit{True} \,|\, u[0]) \vee (u[0] \,|\, \mathit{True})$ both express that a context has a root node labelled u and either children or siblings on the left or siblings on the right.

(5) The formula $\exists\alpha.\,(\neg(u[\mathit{True}] -\!\!\circ_\alpha \neg K)) \circ_\alpha 0$ expresses that it is possible to add some subcontext with a single root node labelled u at some point in the context (not in a context hole) to produce a context satisfying K.

(6) The formula $\exists\alpha.\,(\mathit{True} \circ_\alpha \alpha) \wedge (0 -\!\!\circ_\alpha (\exists\beta.\,\mathit{True} \circ_\beta u[0]))$ expresses that the empty tree may be placed into some context hole such that the resulting tree has some leaf node labelled u.

The logic $CL^{\mathrm{m}}_{\mathit{Tree}}$ has several useful properties. Firstly, $CL^{\mathrm{s}}_{\mathit{Tree}}$ can be embedded in it as described below.

Fix some hole label $x \in X$. The sets $\mathcal{T}$ and $\mathcal{C}^{\mathrm{s}}$ are embedded into $\mathcal{C}^{\mathrm{m}}$ by the functions $[\![-]\!]_{\mathcal{T}} : \mathcal{T} \to \mathcal{C}^{\mathrm{m}}$ and $[\![-]\!]_{\mathcal{C}^{\mathrm{s}}} : \mathcal{C}^{\mathrm{s}} \to \mathcal{C}^{\mathrm{m}}$ respectively, defined inductively as follows:

$$\begin{aligned}
[\![\varepsilon]\!]_{\mathcal{T}} &= \varepsilon \\
[\![u[a]]\!]_{\mathcal{T}} &= u[[\![a]\!]_{\mathcal{T}}] \\
[\![a_1 \mid a_2]\!]_{\mathcal{T}} &= [\![a_1]\!]_{\mathcal{T}} \mid [\![a_2]\!]_{\mathcal{T}} \\
[\![_]\!]_{\mathcal{C}^{\mathrm{s}}} &= x \\
[\![u[c]]\!]_{\mathcal{C}^{\mathrm{s}}} &= u[[\![c]\!]_{\mathcal{C}^{\mathrm{s}}}] \\
[\![a \mid c]\!]_{\mathcal{C}^{\mathrm{s}}} &= [\![a]\!]_{\mathcal{C}^{\mathrm{s}}} \mid [\![c]\!]_{\mathcal{C}^{\mathrm{s}}} \\
[\![c \mid a]\!]_{\mathcal{C}^{\mathrm{s}}} &= [\![c]\!]_{\mathcal{C}^{\mathrm{s}}} \mid [\![a]\!]_{\mathcal{C}^{\mathrm{s}}}.
\end{aligned}$$

Fix some hole variable $\alpha \in \Theta$. The sets $\mathcal{P}^{\mathrm{s}}$ and $\mathcal{K}^{\mathrm{s}}$ are embedded into $\mathcal{K}^{\mathrm{m}}$ by the functions $[\![-]\!]_{\mathcal{P}^{\mathrm{s}}} : \mathcal{P}^{\mathrm{s}} \to \mathcal{K}^{\mathrm{m}}$ and $[\![-]\!]_{\mathcal{K}^{\mathrm{s}}} : \mathcal{K}^{\mathrm{s}} \to \mathcal{K}^{\mathrm{m}}$ respectively, defined inductively as follows:

$$\begin{aligned}
[\![0]\!]_{\mathcal{P}^{\mathrm{s}}} &= 0 \\
[\![K(P)]\!]_{\mathcal{P}^{\mathrm{s}}} &= [\![K]\!]_{\mathcal{K}^{\mathrm{s}}} \circ_{\alpha} [\![P]\!]_{\mathcal{P}^{\mathrm{s}}} \\
[\![K \lhd P]\!]_{\mathcal{P}^{\mathrm{s}}} &= [\![K]\!]_{\mathcal{K}^{\mathrm{s}}} \circ\!\!- _{\alpha} [\![P]\!]_{\mathcal{P}^{\mathrm{s}}} \\
[\![\mathit{false}]\!]_{\mathcal{P}^{\mathrm{s}}} &= \mathit{false} \\
[\![P_1 \Rightarrow P_2]\!]_{\mathcal{P}^{\mathrm{s}}} &= [\![P_1]\!]_{\mathcal{P}^{\mathrm{s}}} \Rightarrow [\![P_2]\!]_{\mathcal{P}^{\mathrm{s}}} \\
[\![u[K]]\!]_{\mathcal{K}^{\mathrm{s}}} &= u[[\![K]\!]_{\mathcal{K}^{\mathrm{s}}}] \\
[\![P \mid K]\!]_{\mathcal{K}^{\mathrm{s}}} &= [\![P]\!]_{\mathcal{P}^{\mathrm{s}}} \mid [\![K]\!]_{\mathcal{K}^{\mathrm{s}}} \\
[\![K \mid P]\!]_{\mathcal{K}^{\mathrm{s}}} &= [\![K]\!]_{\mathcal{K}^{\mathrm{s}}} \mid [\![P]\!]_{\mathcal{P}^{\mathrm{s}}} \\
[\![\mathbf{I}]\!]_{\mathcal{K}^{\mathrm{s}}} &= \alpha \\
[\![P_1 \rhd P_2]\!]_{\mathcal{K}^{\mathrm{s}}} &= [\![P_1]\!]_{\mathcal{P}^{\mathrm{s}}} \multimap_{\alpha} [\![P_2]\!]_{\mathcal{P}^{\mathrm{s}}} \\
[\![\mathit{False}]\!]_{\mathcal{K}^{\mathrm{s}}} &= \mathit{False} \\
[\![K_1 \Rightarrow K_2]\!]_{\mathcal{K}^{\mathrm{s}}} &= [\![K_1]\!]_{\mathcal{P}^{\mathrm{s}}} \Rightarrow [\![K_2]\!]_{\mathcal{P}^{\mathrm{s}}}.
\end{aligned}$$

These embeddings preserve satisfaction in the following sense.

Proposition 21. *For all $a \in \mathcal{T}$, $P \in \mathcal{P}^{\mathrm{s}}$, $c \in \mathcal{C}^{\mathrm{s}}$ and $K \in \mathcal{P}^{\mathrm{s}}$*

$$\begin{aligned}
a \models_{\mathcal{P}^{\mathrm{s}}} P \ \textit{iff} \ \ [\![a]\!]_{\mathcal{T}}, [\alpha \mapsto x] \models [\![P]\!]_{\mathcal{P}^{\mathrm{s}}} \\
c \models_{\mathcal{K}^{\mathrm{s}}} K \ \textit{iff} \ \ [\![c]\!]_{\mathcal{C}^{\mathrm{s}}}, [\alpha \mapsto x] \models [\![K]\!]_{\mathcal{K}^{\mathrm{s}}}.
\end{aligned}$$

This embedding is readily extended to CL^{c}_{Tree}.

Another useful property is given by the use of quantified variables over hole labels in the logic: it is possible to express compounded splittings of a context in a way that is agnostic to the actual labels used to identify the holes. This can result in greater modularity than if we did not have quantification — a subformula can say that a property holds for some choice of hole label without explicitly stating which, and so maintain its sense despite changes in the rest of the formulae which may introduce other hole names. For example, the formula $K_1 \wedge \exists\alpha.\mathit{True} \circ_{\alpha} K_2$ expresses that K_1 holds for the context and that K_2 holds for some subcontext. The choice of which hole label α should be bound to may depend on K_1, but with quantification it is not necessary to re-write the rest of the formula to accommodate a change in K_1.

A further consequence of quantification is that sentences of CL^{m}_{Tree} (that is, formulae with no free variables) describe the shape of contexts completely independently of the hole labels that appear in them. For instance, the formula $\neg\exists\alpha.\ \mathit{True} \circ_\alpha \alpha$ expresses that a context is a tree (that is, it has no holes); the formula $\exists\alpha.\ (\mathit{True} \circ_\alpha \alpha) \wedge \forall\beta.\ \mathit{True} \circ_\beta \beta \Rightarrow \beta \circ_\alpha \mathit{True}$ expresses that a context has exactly one context hole.

A third property of CL^{m}_{Tree} that is significant to this paper is that the games methodology used to prove adjunct elimination for Ambient Logic in [8] can be successfully adapted to prove adjunct elimination for CL^{m}_{Tree}.

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 '$\forall\alpha.\ ((K \mid \alpha) \circ\!\!-_\alpha -)$'.

The following lemma establishes formally the natural intuition that the labels used for holes are not important; that is, we can substitute them and maintain the satisfaction relation. We will make use of this fact in our later results.

Lemma 22 (Hole Substitution Property)**.** *For any tree context $c \in \mathcal{C}^{\mathrm{m}}$, environment $\sigma : \Theta \rightharpoonup_{fin} X$, formula $K \in \mathcal{K}^{\mathrm{m}}$, and hole labels $y, x \in X$ such that $x \notin fn(c) \cup range(\sigma)$,*

$$c, \sigma \models K \quad \textit{iff} \quad c[x/y], \sigma[x/y] \models K.$$

4 Games

We define Ehrenfeucht-Fraïssé-style games for CL^{m}_{Tree}. These games 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 [8], except we use a more relaxed definition of rank, which simply distinguishes between the adjunct and non-adjunct moves.

4.1 Ranks

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. Informally, the rank of a formula is a tuple[2] $r = (n, s, \mathcal{L}) \in \mathbb{N} \times \mathbb{N} \times \mathscr{F}(\Sigma)$ where:

- n is the greatest nesting depth of the non-adjunct, non-Boolean connectives, i.e. '0', '$u[-]$', '$\mid$', 'α', '$\circ_\alpha$', '$\exists\alpha$';

[2] We use the notation $\mathscr{F}(\Sigma)$ to denote the *finite power set* of Σ; that is, the set of all finite subsets of Σ.

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

Table 1
Ranks of Selected Formulae

- s is the greatest nesting depth of the adjunct, non-Boolean connectives, i.e. '$\circ\!\!-_\alpha$', '$\multimap_\alpha$'; and
- $\mathcal{L}$ is the finite subset of Σ consisting of the node labels that occur in the formula.

Definition 23 (Rank)**.** The *rank* of a formula $K \in \mathcal{K}^{\mathrm{m}}$ is $rank(K)$, where the function $rank : \mathcal{K}^{\mathrm{m}} \to \mathbb{N} \times \mathbb{N} \times \mathscr{F}(\Sigma)$ is defined inductively over the structure of CL^{m}_{Tree} formulae by

$$
\begin{aligned}
rank(0) &= (1, 0, \emptyset) \\
rank(u[K]) &= (n + 1, s, \mathcal{L} \cup \{u\}) \\
&\quad \text{where } (n, s, \mathcal{L}) = rank(K) \\
rank(K_1 \mid K_2) &= (\max(n_1, n_2) + 1, \max(s_1, s_2), \mathcal{L}_1 \cup \mathcal{L}_2) \\
&\quad \text{where } (n_1, s_2, \mathcal{L}_1) = rank(K_1) \text{ and } (n_2, s_2, \mathcal{L}_2) = rank(K_2) \\
rank(\alpha) &= (1, 0, 0) \\
rank(K_1 \circ_\alpha K_2) &= (\max(n_1, n_2) + 1, \max(s_1, s_2), \mathcal{L}_1 \cup \mathcal{L}_2) \\
&\quad \text{where } (n_1, s_2, \mathcal{L}_1) = rank(K_1) \text{ and } (n_2, s_2, \mathcal{L}_2) = rank(K_2) \\
rank(K_1 \circ\!\!-_\alpha K_2) &= (\max(n_1, n_2), \max(s_1, s_2) + 1, \mathcal{L}_1 \cup \mathcal{L}_2) \\
&\quad \text{where } (n_1, s_2, \mathcal{L}_1) = rank(K_1) \text{ and } (n_2, s_2, \mathcal{L}_2) = rank(K_2) \\
rank(K_1 \multimap_\alpha K_2) &= (\max(n_1, n_2), \max(s_1, s_2) + 1, \mathcal{L}_1 \cup \mathcal{L}_2) \\
&\quad \text{where } (n_1, s_2, \mathcal{L}_1) = rank(K_1) \text{ and } (n_2, s_2, \mathcal{L}_2) = rank(K_2) \\
rank(\exists\alpha.\, K) &= (n + 1, s, \mathcal{L}) \\
&\quad \text{where } (n, s, \mathcal{L}) = rank(K) \\
rank(False) &= (0, 0, \emptyset) \\
rank(K_1 \Rightarrow K_2) &= (\max(n_1, n_2), \max(s_1, s_2), \mathcal{L}_1 \cup \mathcal{L}_2) \\
&\quad \text{where } (n_1, s_2, \mathcal{L}_1) = rank(K_1) \text{ and } (n_2, s_2, \mathcal{L}_2) = rank(K_2).
\end{aligned}
$$

Lemma 24. *For each rank $r \in \mathbb{N}\times\mathbb{N}\times\mathscr{F}(\Sigma)$ and finite set of variables $\mathcal{V} \subset \Theta$, there are finitely many non-equivalent CL^{m}_{Tree} formulae of rank r whose free variables are in $\mathcal{V}$.*

Proof. By induction on the rank, r.

In the base case $r = (0, 0, \mathcal{L})$. Any formula of rank r must consist only of Boolean connectives, and therefore be equivalent to either *True* or *False*.

In the inductive case, any formula of rank $r = (n, s, \mathcal{L})$ either

- consists of one of the connectives '0', 'α' (for each $\alpha \in \mathcal{V}$), '$u[-]$' (for any $u \in \mathcal{L}$), '$\mid$', '$\circ_\alpha$' (for any $\alpha \in \mathcal{V}$), '$\circ\!\!-_\alpha$' (for any $\alpha \in \mathcal{V}$), '$-\!\!\circ_\alpha$' (for any $\alpha \in \mathcal{V}$), or '$\exists\beta$' (for some $\beta \notin \mathcal{V}$) applied to operands with lower rank, or
- consists of a Boolean combination of formulae of rank at most r whose outermost connectives are non-Boolean.

In the first case, by induction there are only finitely many choices for the operands, since they have lower rank. Note that in the case of '$\exists\beta$' the operand may have free variables in $\mathcal{V} \cup \{\beta\}$ — the inductive hypothesis still covers this. We do, however, only have to consider one choice of β since any other would be equivalent to that. Since there are only finitely many choices of connective (up to alpha equivalence) and finitely many choices of inequivalent operands, there are only finitely many inequivalent formulae that can be constructed in this way.

In the second case, note that there are only finitely many inequivalent Boolean combinations of a finite number of formulae. Since there are only finitely many inequivalent formulae of rank at most r whose outermost connectives are non-Boolean, it follows that only finitely many inequivalent formulae can be constructed in this way. □

Definition 25. For each rank $r \in \mathbb{N} \times \mathbb{N} \times \mathscr{F}(\Sigma)$ and finite set of free variables $\mathcal{V} \subset \Theta$, choose $\mathcal{K}_{r,\mathcal{V}} \subset \mathcal{K}^{\mathrm{m}}$ to be a finite set of CL^{m}_{Tree} formulae of rank r with free variables in $\mathcal{V}$, such that every formula in $\mathcal{K}^{\mathrm{m}}$ of rank r with free variables in $\mathcal{V}$ is equivalent to some formula in $\mathcal{K}_{r,\mathcal{V}}$.

For context c, the defining formula of c with respect to rank r and environment σ is

$$D^r_{c,\sigma} = \bigwedge \{K \in \mathcal{K}_{r,\mathcal{V}} : c, \sigma \models K\},$$

where $\mathcal{V} = dom(\sigma)$. By construction, $D^r_{c,\sigma}$ has rank r and $c, \sigma \models D^r_{c,\sigma}$.

Lemma 26. *For contexts $c, c' \in \mathcal{C}^{\mathrm{m}}$, environments $\sigma, \sigma' : \Theta \rightharpoonup_{fin} X$ with $dom(\sigma) = dom(\sigma') = \mathcal{V}$, and rank $r \in \mathbb{N} \times \mathbb{N} \times \mathscr{F}(\Sigma)$,*

$$(\textit{for all } K \in \mathcal{K}_{r,\mathcal{V}},\ c, \sigma \models K \textit{ implies } c', \sigma' \models K) \quad \textit{iff} \quad c', \sigma' \models D^r_{c,\sigma} \tag{5}$$

$$(\textit{there exists } K \in \mathcal{K}_{r,\mathcal{V}} \textit{ s.t. } c, \sigma \models K \textit{ and } c', \sigma' \not\models K) \quad \textit{iff} \quad c', \sigma' \not\models D^r_{c,\sigma} \tag{6}$$

$$(\textit{for all } K \in \mathcal{K}_{r,\mathcal{V}},\ c, \sigma \models K \textit{ iff } c', \sigma' \models K) \quad \textit{iff} \quad c', \sigma' \models D^r_{c,\sigma} \tag{7}$$

$$c', \sigma' \models D^r_{c,\sigma} \quad \textit{iff} \quad c, \sigma \models D^r_{c',\sigma'} \tag{8}$$

Proof. For (5), by definition:

$$\begin{aligned} c', \sigma' \models D^r_{c,\sigma} \ \text{ iff } \ & c', \sigma' \models \left(\bigwedge \{K \in \mathcal{K}_{r,\mathcal{V}} : c, \sigma \models K\}\right) \\ \text{iff } \ & \text{for all } K \in \mathcal{K}_{r,\mathcal{V}},\ c, \sigma \models K \text{ implies } c', \sigma' \models K. \end{aligned}$$

Assertion (6) is immediate from (5).

For (7), the implication from left to right is immediate from (5). To show the implication from right to left, suppose that $c', \sigma' \models D^r_{c,\sigma}$ and fix some $K \in \mathcal{K}_{r,\mathcal{V}}$. If $c, \sigma \models K$ then $c', \sigma' \models K$ by (5). Conversely, if $c', \sigma' \models K$ then $c', \sigma' \not\models \neg K$ and so $c, \sigma \not\models \neg K$ by (5) and $c, \sigma \models K$.

Assertion (8) follows immediately from (7). □

Definition 27. Let $\mathcal{S}$ be a set. A pair (a, b) is said to be $\mathcal{S}$-*discriminated* if either $a \in \mathcal{S}$ and $b \notin \mathcal{S}$, or $a \notin \mathcal{S}$ and $b \in \mathcal{S}$.

Lemma 28. *Let $\mathcal{S} \subseteq \mathcal{C}^{\mathrm{m}} \times (\Theta \rightharpoonup_{fin} X)$ be a set of context-environment pairs, $r \in \mathbb{N} \times \mathbb{N} \times \mathscr{F}(\Sigma)$ be a rank, and $\mathcal{V} \subset \Theta$ be a finite set of hole variables. Suppose that for any $\mathcal{S}$-discriminated pair $((c, \sigma), (c', \sigma'))$ there exists a formula $K_{(c,\sigma),(c',\sigma')}$ of rank r and with free variables in $\mathcal{V}$ such that $c, \sigma \models K_{(c,\sigma),(c',\sigma')}$ and $c', \sigma' \not\models K_{(c,\sigma),(c',\sigma')}$. Then $\mathcal{S}$ can be defined by a rank-r formula K with free variables in $\mathcal{V}$.*

Proof. Consider $\mathcal{K}_{\mathcal{S}} = \left\{K \in \mathcal{K}_{r,\mathcal{V}} : \exists (c, \sigma) \in \mathcal{S}.\, K \text{ is equivalent to } D^r_{c,\sigma}\right\}$. By construction, $\mathcal{K}_{\mathcal{S}}$ is finite, hence, $K = \bigvee \mathcal{K}_{\mathcal{S}}$ is a formula of rank r and free variables in $\mathcal{V}$. We shall show that $d, \rho \models K$ if and only if $(d, \rho) \in \mathcal{S}$.

Suppose that $(d, \rho) \in \mathcal{S}$. By definition, there exists $K' \in \mathcal{K}_{r,\mathcal{V}}$ which is equivalent to $D^r_{d,\rho}$. Hence, $d, \rho \models K'$ and $K' \in \mathcal{K}_{\mathcal{S}}$, and so $d, \rho \models K$.

Now suppose that $d, \rho \models K$. Then $d, \rho \models D^r_{c,\sigma}$ for some $(c, \sigma) \in \mathcal{S}$. If $(d, \rho) \notin \mathcal{S}$, then there exists a rank-r formula with free variables in $\mathcal{V}$ that discriminates between (d, ρ) and (c, σ), which is not possible by Lemma 26. Hence, $(d, \rho) \in \mathcal{S}$. □

4.2 Games

We adapt the Ehrenfeucht-Fraïssé games to CL^{m}_{Tree}. 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.[3] 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 Spoiler is unable to play a move, for instance, 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

[3] For convenience, we sometimes use pronouns to refer to the players, adopting male personal pronouns for Spoiler and female personal pronouns for Duplicator. The reader is asked to forgive the implied personification of these abstract entities.

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

PAR move. Spoiler chooses some $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ such that $d = d_1 \mid d_2$. Duplicator chooses some $d'_1, d'_2 \in \mathcal{C}^{\mathrm{m}}$ 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 $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ such that $d = d_1 \mathbin{⊛} d_2$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}^{\mathrm{m}}$ 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 hole label $x \in X$. 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 $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ such that $d_2 = d_1 \mathbin{⊛} d$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}^{\mathrm{m}}$ 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 $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ such that $d_2 = d \mathbin{⊛} d_1$. Duplicator then chooses $d'_1, d'_2 \in \mathcal{C}^{\mathrm{m}}$ 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: either 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 29 (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 30 (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 of a game, 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 the '$\multimap$'-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 this move then the two (current) trees are differentiated by the formula $\mathit{True} \multimap_\alpha \mathit{False}$: one tree has an α-labelled hole (so the formula is not satisfied) while the other does not (so the formula is satisfied trivially).

This correspondence between game moves and logical connectives is formalised in the soundness and completeness results below. The results establish that the existence of a formula of rank r that discriminates between two contexts is equivalent to the existence of a winning strategy for Spoiler for the game of rank r on those two contexts.

Lemma 31 (Game Soundness)**.** *For* $c, c' \in \mathcal{C}^{\mathrm{m}}$ *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* $((c, \sigma), (c', \sigma'), r)$*.*

Proof. The proof is by induction on the structure of the formula K. We look at the cases for the outermost operator. In each case, we show inductively that Spoiler can play the corresponding game move and thereby obtain a winning strategy.

$K = 0$**.** In this case, $c = \varepsilon$ and $c' \neq \varepsilon$. Hence, Spoiler may play the EMP move and win the game.

$K = \alpha$**.** In this case, $c = \sigma(\alpha)$ and $c' \neq \sigma'(\alpha)$. Hence, Spoiler may play the VAR move, choosing c and α, and win the game.

$K = u[K_1]$**.** In this case, $c = u[c_1]$ for some $c_1, \sigma \models K_1$. Suppose that Spoiler plays the LAB move choosing c and label u. Since $c', \sigma' \not\models u[K_1]$, either $c' \neq u[c_1']$ for any c_1', in which case Spoiler wins immediately, or $c' = u[c_1']$ with $c_1', \sigma' \not\models K_1$, in which case Spoiler will win since, by the inductive hypothesis, he has a winning strategy for the game $((c_1, \sigma), (c_1', \sigma'), (n-1, s, \mathcal{L}))$.

$K = K_1 \mid K_2$**.** In this case, $c = c_1 \mid c_2$ with $c_1, \sigma \models K_1$ and $c_2, \sigma \models K_2$. Suppose that Spoiler plays the PAR move, splitting c into c_1 and c_2. Duplicator responds by splitting $c' = c_1' \mid c_2'$. Since $c', \sigma' \not\models K_1 \mid K_2$, either $c_1', \sigma' \not\models K_1$ or $c_2', \sigma' \not\models K_2$. If the former, by the inductive hypothesis, $((c_1, \sigma), (c_1', \sigma'), (n-1, s, \mathcal{L})) \in SW$. If the latter, $((c_2, \sigma), (c_2', \sigma'), (n-1, s, \mathcal{L})) \in SW$. In either case, Spoiler has a winning strategy.

For the following cases, let $x = \sigma(\alpha)$, $\acute{x} = \sigma'(\alpha)$.

$K = K_1 \circ_\alpha K_2$**.** Here, $c = c_1 \mathbin{⊛} c_2$ for some $c_1, \sigma \models K_1$, $c_2, \sigma \models K_2$. Suppose that Spoiler plays the CMP move, choosing to split c as $c_1 \mathbin{⊛} c_2$. Then Duplicator responds with $c' = c'_1 \mathbin{⊛} c'_2$. Since $c', \sigma' \not\models K_1 \circ_\alpha K_2$, either $c'_1, \sigma' \not\models K_1$ or $c'_2, \sigma' \not\models K_2$. If the former, by the inductive hypothesis, $((c_1, \sigma), (c'_1, \sigma'), (n-1, s, \mathcal{L})) \in SW$. If the latter, $((c_2, \sigma), (c'_2, \sigma'), (n-1, s, \mathcal{L})) \in SW$. In either case, Spoiler has a winning strategy.

$K = K_1 \circ\!\!- _\alpha K_2$**.** In this case, since $c' \not\models K$, there are c'_1, c'_2 with $c'_2 = c'_1 \mathbin{⊛} c'$, $c'_1 \models K_1$ and $c'_2 \not\models K_2$. Suppose that Spoiler plays the LEF move, choosing to work with the pair (c', σ'), the contexts c'_1, c'_2 and the hole label $\acute{x}$. Duplicator then responds with some c_1, c_2 with $c_2 = c_1 \mathbin{⊛} c$. If $c_1 \not\models K_1$ then, by the inductive hypothesis, Spoiler has a winning strategy for the game $((c_1, \sigma), (c'_1, \sigma'), (n, s-1, \mathcal{L}))$. Otherwise, $c \models K$ implies that $c_2 \models K_2$, and so, by the inductive hypothesis, Spoiler has a winning strategy for the game $((c_1, \sigma), (c'_2, \sigma'), (n, s-1, \mathcal{L}))$.

$K = K_1 \mathrel{-\!\!\circ}_\alpha K_2$**.** In this case, since $c' \not\models K$, there are c'_1, c'_2 with $c'_2 = c' \mathbin{⊛} c'_1$, $c'_1 \models K_1$ and $c'_2 \not\models K_2$. Suppose that Spoiler plays the RIG move, choosing to work with (c', σ'), c'_1, c'_2 and $\acute{x}$. If $x \notin fn(c)$ then Spoiler wins immediately. Otherwise, Duplicator responds with some c_1, c_2 with $c_2 = c \mathbin{⊛} c_1$. If $c_1 \not\models K_1$ then, by the inductive hypothesis, Spoiler has a winning strategy for the game $((c_1, \sigma), (c'_1, \sigma'), (n, s-1, \mathcal{L}))$. Otherwise, $c \models K$ implies that $c_2 \models K_2$, and so, by the inductive hypothesis, Spoiler has a winning strategy for the game $((c_2, \sigma), (c'_2, \sigma'), (n, s-1, \mathcal{L}))$.

$K = \exists\alpha. K_1$**.** In this case, $c, \sigma[\alpha \mapsto x] \models K_1$ for some x and fresh α. Suppose that Spoiler plays the EXS move, choosing to instantiate α as x on σ. Duplicator responds by instantiating α as $\acute{x}$. Since $c', \sigma' \not\models \exists\alpha. K_1$, $c', \sigma'[\alpha \mapsto \acute{x}] \not\models K_1$. Thus, by induction, $((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto \acute{x}]), (n-1, s, \mathcal{L})) \in SW$, and so Spoiler has a winning strategy. □

Lemma 32 (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* $c', \sigma' \not\models K$*.*

Proof. The proof is by induction on the rank $r = (n, s, \mathcal{L})$, and by cases on the first move that Spoiler makes in his winning strategy for the game. At the start of the first move, Spoiler may choose either (c, σ) or (c', σ'); we assume that he chooses the former, without loss of generality. [4]

EMP move. Here, $c, \sigma \models 0$ and $c', \sigma' \not\models 0$.

VAR move. Suppose that Spoiler plays this move with $x = \sigma(\alpha)$. Then $c, \sigma \models \alpha$ and $c', \sigma' \not\models \alpha$.

For the following cases, let $r^- = (n-1, s, \mathcal{L})$.

[4] If he chooses (c', σ') instead, the proof gives K with $c', \sigma' \models K$, and $c, \sigma \not\models K$. Then $\neg K$ satisfies the required properties.

LAB move. Suppose that Spoiler plays this move using label $u \in \mathcal{L}$, and that $c = u[c_1]$. If Spoiler wins on this move, then $c' \neq u[c'_1]$ for any c'_1, and thus $c', \sigma' \not\models u[True]$ but $c, \sigma \models u[True]$. Otherwise, $c' = u[c'_1]$ for some c'_1, and Spoiler has a winning strategy for the game $((c_1, \sigma), (c'_1, \sigma'), r^-)$, and so by induction, there is a K_1 with $c_1, \sigma \models K_1$ and $c'_1, \sigma' \not\models K_1$. Therefore $c, \sigma \models u[K_1]$ and $c', \sigma' \not\models u[K_1]$.

PAR move. Suppose that Spoiler, for his winning strategy, plays this move, splitting $c = c_1 \mid c_2$. Then let $K = D^{r^-}_{c_1,\sigma} \mid D^{r^-}_{c_2,\sigma}$. We know that $c, \sigma \models K$. Suppose that $c', \sigma' \models K$ also. Then $c' = c'_1 \mid c'_2$ with $c'_1, \sigma' \models D^{r^-}_{c_1,\sigma}$ and $c'_2, \sigma' \models D^{r^-}_{c_2,\sigma}$ for some c'_1, c'_2. Thus there is no formula of rank r^-, free variables in $\mathcal{V} = dom(\sigma)$, that discriminates between (c_1, σ) and (c'_1, σ') or between (c_2, σ) and (c'_2, σ'). By the inductive hypothesis, this implies that Duplicator has a winning strategy for the games $((c_1, \sigma), (c'_1, \sigma'), r^-)$ and $((c_2, \sigma), (c'_2, \sigma'), r^-)$, which contradicts the fact that this move is part of Spoiler's winning strategy. Therefore $c', \sigma' \not\models K$ and $c, \sigma \models K$.

CMP move. Suppose that Spoiler plays this move, splitting $c = c_1 \mathbin{⊚}_x c_2$ with $x = \sigma(\alpha)$. Then let $K = D^{r^-}_{c_1,\sigma} \circ_\alpha D^{r^-}_{c_2,\sigma}$. We know that $c, \sigma \models K$. Suppose that $c', \sigma' \models K$ also. Then $c' = c'_1 \mathbin{⊚}_{\acute{x}} c'_2$ with $c'_1, \sigma' \models D^{r^-}_{c_1,\sigma}$ and $c'_2, \sigma' \models D^{r^-}_{c_2,\sigma}$, for some c'_1, c'_2 and $\acute{x} = \sigma'(\alpha)$. Thus there is no formula of rank r^-, free variables in $\mathcal{V} = dom(\sigma)$, that discriminates between (c_1, σ) and (c'_1, σ') or between (c_2, σ) and (c'_2, σ'). By the inductive hypothesis, this implies that Duplicator has a winning strategy for the games $((c_1, \sigma), (c'_1, \sigma'), r^-)$ and $((c_2, \sigma), (c'_2, \sigma'), r^-)$, which contradicts the fact that this move is part of Spoiler's winning strategy. Therefore $c', \sigma' \not\models K$ and $c, \sigma \models K$.

EXS move. Suppose that Spoiler plays this move, extending σ by $[\alpha \mapsto x]$. Then let $K = \exists\alpha.\, D^{r^-}_{c,\sigma[\alpha \mapsto x]}$, which has rank r, and free variables in $\mathcal{V} = dom(\sigma)$. We know that $c, \sigma \models K$. Suppose that $c', \sigma' \models K$ also. Then, for some $\acute{x}$, $c', \sigma'[\alpha \mapsto \acute{x}] \models D^{r^-}_{c,\sigma[\alpha \mapsto x]}$. Thus, there is no formula of rank r^- with free variables in $\mathcal{V} \cup \{\alpha\}$ that discriminates between $(c, \sigma[\alpha \mapsto x])$ and $(c', \sigma'[\alpha \mapsto \acute{x}])$. Hence, by induction, Duplicator has a winning strategy for the game $((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto \acute{x}]), r^-)$, which contradicts the fact that this move is part of Spoiler's winning strategy. Therefore $c', \sigma' \not\models K$ and $c, \sigma \models K$.

For the following cases, let $r^- = (n, s-1, \mathcal{L})$.

LEF move. Suppose that Spoiler plays this move, choosing $c_2 = c_1 \mathbin{⊚}_x c$. Then let $K = D^{r^-}_{c_1,\sigma} \bullet\!\!-_\alpha D^{r^-}_{c_2,\sigma}$. We know that $c, \sigma \models K$. Suppose that $c', \sigma' \models K$ also. Then $c'_2 = c'_1 \mathbin{⊚}_{\acute{x}} c'$ with $c'_1, \sigma' \models D^{r^-}_{c_1,\sigma}$ and $c'_2, \sigma' \models D^{r^-}_{c_2,\sigma}$, for some c'_1, c'_2 and $\acute{x} = \sigma'(\alpha)$. Thus there is no formula of rank r^-, free variables in $\mathcal{V} = dom(\sigma)$, that discriminates between (c_1, σ) and (c'_1, σ') or between (c_2, σ) and (c'_2, σ'). By the inductive hypothesis, this implies that Duplicator has a winning strategy for the games $((c_1, \sigma), (c'_1, \sigma'), r^-)$ and $((c_2, \sigma), (c'_2, \sigma'), r^-)$, which contradicts the fact that this move is part of Spoiler's winning strategy. Therefore $c', \sigma' \not\models K$ and $c, \sigma \models K$.

RIG move. Suppose that Spoiler plays this move, choosing $c_2 = c \mathbin{⊚}_x c_1$. Then let $K = D^{r^-}_{c_1,\sigma} \mathbin{-\!\!\bullet}_\alpha D^{r^-}_{c_2,\sigma}$. We know that $c, \sigma \models K$. Suppose that $c', \sigma' \models K$ also. Then $c'_2 = c' \mathbin{⊚}_{\acute{x}} c'_1$ with $c'_1, \sigma' \models D^{r^-}_{c_1,\sigma}$ and $c'_2, \sigma' \models D^{r^-}_{c_2,\sigma}$, for some c'_1, c'_2 and

$\acute{x} = \sigma'(\alpha)$. Thus there is no formula of rank r^-, free variables in $\mathcal{V} = dom(\sigma)$, that discriminates between (c_1, σ) and (c_1', σ') or between (c_2, σ) and (c_2', σ'). By the inductive hypothesis, this implies that Duplicator has a winning strategy for the games $((c_1, \sigma), (c_1', \sigma'), r^-)$ and $((c_2, \sigma), (c_2', \sigma'), r^-)$, which contradicts the fact that this move is part of Spoiler's winning strategy. Therefore $c', \sigma' \not\models K$ and $c, \sigma \models K$. □

The following two lemmata are useful for checking structural properties in our adjunct-elimination results. 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 33. *If* $((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW$ *with* $n \geq 2$*, then, for* $x = \sigma(\alpha)$*,* $\acute{x} = \sigma'(\alpha)$*,*

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

Proof. Suppose that $x \in fn(c)$. We know that Spoiler would be able to play the CMP move and split $c = c \circledast x$. Since $((c, \sigma), (c', \sigma'), r) \in DW$, there exist c_1', c_2' with $c' = c_1' \circledast \acute{x} c_2'$ and

$$((x, \sigma), (c_2', \sigma'), (n-1, s, \mathcal{L})) \in DW. \tag{9}$$

Since Spoiler could then play the VAR move with $x = \sigma(\alpha)$ on (9), it must be that $c_2' = \sigma'(\alpha) = \acute{x}$. Therefore, $\acute{x} \in fn(c')$. The argument in the reverse direction is the same. □

Lemma 34. *Suppose that* $((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW$ *with* $n \geq 2$*. If* $c = c_1 \mid x$ *for* $x = \sigma(\alpha)$ *and* $c_1 \in \mathcal{C}^{\mathrm{m}}$*, then* $c' = c_1' \mid \acute{x}$ *for* $\acute{x} = \sigma'(\alpha)$ *and some* $c_1' \in \mathcal{C}^{\mathrm{m}}$*. Similarly, if* $c = x \mid c_1$ *then* $c' = \acute{x} \mid c_1'$*.*

Proof. Suppose that $c = c_1 \mid x$. We know that Spoiler could play the PAR move and split $c = c_1 \mid x$. Since $((c, \sigma), (c', \sigma'), r) \in DW$, we know that $c' = c_1' \mid c_2'$ so that

$$((x, \sigma), (c_2', \sigma'), (n-1, s, \mathcal{L})) \in DW. \tag{10}$$

Since Spoiler could then play the VAR move with $x = \sigma(\alpha)$ on (10), it must be that $c_2' = \sigma'(\alpha) = \acute{x}$. Therefore, $c' = c_1' \mid \acute{x}$, as required. The proof for the other case is analogous. □

Lemma 35 (Hole Substitution Property for Games)**.** *Suppose that*

$$((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW \tag{11}$$

and that $x \notin fn(c) \cup range(\sigma)$*. Then*

$$((c[x/y], \sigma[x/y]), (c', \sigma'), (n, s, \mathcal{L})) \in DW. \tag{12}$$

Proof. By game soundness, for every formula K of rank $r = (n, s, \mathcal{L})$, $c, \sigma \models K$ if and only if $c', \sigma' \models K$. By Lemma 22, for every formula K of rank r, $c[x/y], \sigma[x/y] \models K$ if and only if $c', \sigma' \models K$. Hence, by game completeness, (12) holds. □

The next lemma allows us to consider a particular response to the EXS move that we know can give a winning strategy for Duplicator, even though there may be other responses. The 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.

Lemma 36 (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

(1) $x = \sigma(\beta)$ *and* $\acute{x} = \sigma'(\beta)$ *for some hole variable* $\beta \in \Theta$*, or*

(2) $x \notin fn(c) \cup range(\sigma)$ *and* $\acute{x} \notin fn(c') \cup range(\sigma')$*.*

Proof. In the first case, suppose that Spoiler were to choose to play the EXS move on the original game. We know then that, for $x = \sigma(\beta)$, there exists a y such that

$$((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto y]), (n-1, s, \mathcal{L})) \in DW.$$

By playing CMP move, choosing to split $c = x \circledast c$ (using α), we get that for some $\bar{c}'$

$$((x, \sigma[\alpha \mapsto x]), (\bar{c}', \sigma'[\alpha \mapsto y]), (n-2, s, \mathcal{L})) \in DW.$$

Now, Spoiler could play the VAR move (using β) and win unless $\bar{c}' = \sigma'[\alpha \mapsto y](\beta)$ (since $x = \sigma[\alpha \mapsto x](\beta)$) and $\bar{c}' = \sigma'[\alpha \mapsto y](\alpha) = y$ (since $x = \sigma[\alpha \mapsto x](\alpha)$). Hence, for $\acute{x} = \sigma'(\beta) = y$,

$$((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto \acute{x}]), (n-1, s, \mathcal{L})) \in DW.$$

In the second case, we know that Spoiler would be able to play the EXS move on the original game and introduce x. As in the previous case, we know that there exists a y such that

$$((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto y]), (n-1, s, \mathcal{L})) \in DW.$$

By Lemma 33, since we know that $x \notin fn(c)$ we can conclude that $y \notin fn(c')$, and so $c'[\acute{x}/y] = c'$. If we suppose that $y = \sigma'(\beta)$ for some β then, since we know that $x \neq \sigma(\beta)$, we can see that Spoiler would have a winning strategy by playing the CMP move to split $c' = y \circledast c'$ (using α) followed by the CMP move (using β). Therefore, we have that $y \notin range(\sigma')$, and so $(\sigma'[\alpha \mapsto y])[\acute{x}/y] = \sigma'[\alpha \mapsto \acute{x}]$. Hence, by Lemma 35,

$$((c, \sigma[\alpha \mapsto x]), (c', \sigma'[\alpha \mapsto \acute{x}]), (n-1, s, \mathcal{L})) \in DW,$$

as required. □

5 Adjunct Elimination

We now have the background required to prove adjunct elimination for CL^{m}_{Tree}. Proposition 37 is the key result. It states that, with no adjunct moves, a winning strategy for Duplicator for the game on the composition of contexts follows from Duplicator's winning strategies for its components. A consequence is that, if Duplicator has a winning strategy without adjunct moves, then she has a winning strategy with 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 37 (One-step move elimination)**.** *For all ranks of the form* $r = (n, 0, \mathcal{L})$*, for all* $c_1, c_1', c_2, c_2' \in \mathcal{C}^{\mathrm{m}}$*, for all domain-coincident environments* σ, σ'*, if*

$$((c_1, \sigma), (c_1', \sigma'), (3n, 0, \mathcal{L})) \in DW \tag{13}$$
$$((c_2, \sigma), (c_2', \sigma'), (3n, 0, \mathcal{L})) \in DW \tag{14}$$

then, for all $\alpha \in dom(\sigma)$ *with* $x = \sigma(\alpha)$*,* $\acute{x} = \sigma'(\alpha)$*, if* $c = c_1 \mathbin{⊘} c_2$ *and* $c' = c_1' \mathbin{\hat{⊘}} c_2'$ *are defined then*

$$((c, \sigma), (c', \sigma'), (n, 0, \mathcal{L})) \in DW. \tag{15}$$

Proof. The proof is by induction on n and by cases on Spoiler's choice of move in the game of (15). The base case, $n = 0$, is trivial, since Spoiler can never win a game of such a rank. In the inductive case, where $n > 0$, we assume as the inductive hypothesis that the proposition holds for all 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 (13) and (14). Knowing that Duplicator has a winning strategy in these games, we can establish properties, usually concerning the structure of c_1' and c_2', and construct a winning response for Duplicator to Spoiler's move on (15) by way of the inductive hypothesis.

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.

VAR move. In order for Spoiler to be able to play this move, it must be the case that $c = y = \sigma(\beta)$ and $c' \neq \acute{y} = \sigma'(\beta)$ for some β. There are three cases of the possible structure of c_1 and c_2:

(1) $c_1 = x$ and $c_2 = y$;

(2) $c_1 = y \mid x$ and $c_2 = \varepsilon$;

(3) $c_1 = x \mid y$ and $c_2 = \varepsilon$.

In the first case, $c'_1 = \acute{x}$ and $c'_2 = \acute{y}$, for otherwise Spoiler could play the VAR move to win the games in (13) and (14) respectively. Hence $c' = \acute{y}$, and Spoiler would not have been able to play the VAR move after all.

In the second case, we have from (13) that $c'_1 = \bar{c}'_1 \mid \hat{c}'_1$ with

$$((y, \sigma), (\bar{c}'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{16}$$
$$((x, \sigma), (\hat{c}'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{17}$$

Hence $\bar{c}'_1 = \acute{y}$ and $\hat{c}'_1 = \acute{x}$, or else Spoiler would have a winning strategy for these games by playing the VAR move. Also, by (14), $c'_2 = \varepsilon$, or else Spoiler would have a winning strategy for that game by playing the EMP move. We have $c' = c'_1 \mathbin{⊛} c'_2 = (\acute{y} \mid \acute{x}) \mathbin{⊛} \varepsilon = \acute{y}$, and so Spoiler would not have been able to play the VAR move after all.

The third case is analogous to the second.

Since we know that in all cases Spoiler could not have played the VAR move (and thereby have a winning strategy in the game (15)), we know that Spoiler does not have a winning strategy by playing the VAR move.

LAB move. Suppose that Spoiler plays this move picking $u \in \mathcal{L}$ and $d \in \mathcal{C}^{\mathrm{m}}$ 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 $d = d_1 \mathbin{⊛} c_2$;

(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 (13), with label u and context d_1. Hence, by (13), $c'_1 = u[d'_1]$ with

$$((d_1, \sigma), (d'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{18}$$

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

$$((d_1 \mathbin{⊛} c_2, \sigma), (d'_1 \mathbin{⊛} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{19}$$

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 (14). Further, Spoiler could play the PAR move on (13) so we have $c'_1 = d'_1 \mid d'_2$ with

$$((u[d], \sigma), (d'_1, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{20}$$
$$((x, \sigma), (d'_2, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{21}$$

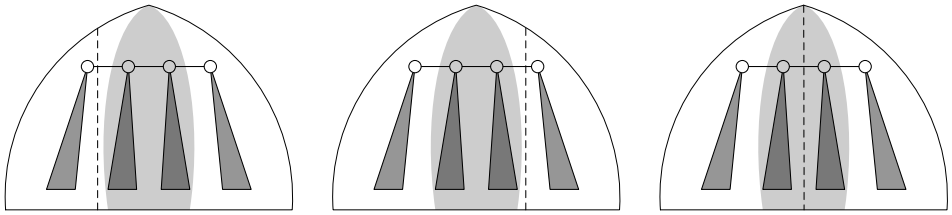

Fig. 5. In left-to-right order, the three cases for splitting $c = d_1 \mid d_2$

Since $3n - 1 \geq 1$, by (21) we know $d_2' = \acute{x}$, since otherwise Spoiler could play the VAR move. Spoiler could play the LAB move on the game in (20), using u as the label, so that we must have $d_1' = u[d']$ with

$$((d, \sigma), (d', \sigma'), (3n - 2, 0, \mathcal{L})) \in DW. \tag{22}$$

We now have $c' = (u[d'] \mid \acute{x}) \mathbin{\textcircled{\scriptsize $\acute{x}$}} \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 (22), Duplicator has a winning strategy.

The third case is analogous to the second.

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, $c_1 = d_1 \mid d_3$, $d_2 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2$.

(2) Spoiler splits in c_1 to the right of the x: that is, $c_1 = d_3 \mid d_2$, $d_1 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2$. 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 $c_1 = \bar{d}_3 \mid x \mid \bar{d}_4$, $c_2 = d_5 \mid d_6$, $d_1 = \bar{d}_3 \mid d_5$ and $d_2 = d_6 \mid \bar{d}_4$.

These three cases are illustrated by Figure 5. The shaded area indicates the c_2 subtree and the dashed line indicates the splitting point. Note that the third case does not apply to every possible choice of c_1 and c_2, but our example shows a choice for which it does.

In the first case,

$$\begin{aligned} c_1 \mathbin{\textcircled{\scriptsize x}} c_2 &= (d_1 \mid d_3) \mathbin{\textcircled{\scriptsize x}} c_2 \\ &= d_1 \mid (d_3 \mathbin{\textcircled{\scriptsize x}} c_2). \end{aligned}$$

As Spoiler could play the PAR move in the game in (13), we know that $c_1' = d_1' \mid d_3'$ such that

$$((d_1, \sigma), (d_1', \sigma'), (3n - 1, 0, \mathcal{L})) \in DW \tag{23}$$
$$((d_3, \sigma), (d_3', \sigma'), (3n - 1, 0, \mathcal{L})) \in DW. \tag{24}$$

Note that $fn(d'_3) \subseteq fn(c'_1)$ and $\acute{x} \in fn(d'_3)$ by Lemma 33 (since $x \in fn(d_3)$), so $d'_2 = d'_3 \mathbin{⊛} c'_2$ is defined. By downward closure on (24) and (14) and by the inductive hypothesis,

$$((d_3 \mathbin{⊛} c_2, \sigma), (d'_3 \mathbin{⊛} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{25}$$

Observe that

$$\begin{aligned} c' &= c'_1 \mathbin{⊛} c'_2 \\ &= (d'_1 \mid d'_3) \mathbin{⊛} c'_2 \\ &= d'_1 \mid (d'_3 \mathbin{⊛} c'_2) \\ &= d'_1 \mid d'_2. \end{aligned}$$

Thus responding with d'_1 and d'_2 gives Duplicator a winning strategy in this case, by downward closure on (23) and by (25).

In the third case, we know that

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

Spoiler could play the CMP move on the game in (13), so $c'_1 = d'_3 \mathbin{⊛} d'_4$ with

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{26}$$
$$((d_4, \sigma), (d'_4, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{27}$$

Also, Spoiler could play the PAR move on the game in (14), so $c'_2 = d'_5 \mid d'_6$ with

$$((d_5, \sigma), (d'_5, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{28}$$
$$((d_6, \sigma), (d'_6, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{29}$$

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 (26) and (28), and on (27) and (29), we get

$$((d_3 \mathbin{⊛} d_5, \sigma), (d'_3 \mathbin{⊛} d'_5, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{30}$$
$$((d_4 \mathbin{⊛} d_6, \sigma), (d'_4 \mathbin{⊛} d'_6, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{31}$$

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 34 to (26) and (27).

Thus, by structural considerations,

$$\begin{aligned} c' &= c'_1 \mathbin{\textcircled{\scriptsize \acute{x}}} c'_2 \\ &= (d'_3 \mathbin{\textcircled{\scriptsize \acute{x}}} d'_4) \mathbin{\textcircled{\scriptsize \acute{x}}} (d'_5 \mid d'_6) \\ &= ((\bar{d}'_3 \mid \acute{x}) \mathbin{\textcircled{\scriptsize \acute{x}}} (\acute{x} \mid \bar{d}'_4)) \mathbin{\textcircled{\scriptsize \acute{x}}} (d'_5 \mid d'_6) \\ &= \bar{d}'_3 \mid d'_5 \mid d'_6 \mid \bar{d}'_4 \\ &= (d'_3 \mathbin{\textcircled{\scriptsize \acute{x}}} d'_5) \mid (d'_4 \mathbin{\textcircled{\scriptsize \acute{x}}} d'_6) \\ &= d'_1 \mid d'_2. \end{aligned}$$

Hence, by (30) and (31), 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 $c_1 \mathbin{\textcircled{\scriptsize x}} c_2$ as $d_1 \mathbin{\textcircled{\scriptsize y}} d_2$. 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 $c' = \acute{y} \mathbin{\textcircled{\scriptsize \acute{y}}} c'$, for instance.)

There are four cases for how Spoiler can make the splitting $c = d_1 \mathbin{\textcircled{\scriptsize y}} d_2$. Using Figure 4 as an example instance of $c = c_1 \mathbin{\textcircled{\scriptsize x}} c_2$, the cases are illustrated in Figures 6, 7, 8 and 9. In the diagrams, the darker subtree denotes the c_2 part of $c = c_1 \mathbin{\textcircled{\scriptsize x}} c_2$ and the dashed outline denotes the d_2 part of $c = d_1 \mathbin{\textcircled{\scriptsize y}} d_2$. The cases are:

(1) Spoiler splits c within c_2 (Figure 6), so we get $c_2 = d_3 \mathbin{\textcircled{\scriptsize y}} d_2$ and $d_1 = c_1 \mathbin{\textcircled{\scriptsize x}} d_3$.

(2) Spoiler splits c outside c_2, including all of c_2 (Figure 7), so we get $c_1 = d_1 \mathbin{\textcircled{\scriptsize y}} d_3$ and $d_2 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2$.

(3) Spoiler splits c so that d_2 consists of part of c_1 and part (but not all) of c_2 (Figure 8). Here, the part of c_1 must be a subtree adjacent to the x hole, and the part of c_2 must be subtree at the root of c_2 and on the appropriate side.

(4) Spoiler splits c so that d_2 is a subtree that is completely disjoint from the hole (Figure 9). Here, $c_1 = d_3 \mathbin{\textcircled{\scriptsize y}} d_2$ and $d_1 = d_3 \mathbin{\textcircled{\scriptsize x}} c_2$, providing $x \neq y$. We shall also consider the case when $x = y$.

We consider each of these cases individually.

Case 1: Spoiler splits inside c_2, as

$$\begin{aligned} c_1 \mathbin{\textcircled{\scriptsize x}} c_2 &= c_1 \mathbin{\textcircled{\scriptsize x}} (d_3 \mathbin{\textcircled{\scriptsize y}} d_2) \\ &= (c_1 \mathbin{\textcircled{\scriptsize x}} d_3) \mathbin{\textcircled{\scriptsize y}} d_2 \\ &= d_1 \mathbin{\textcircled{\scriptsize y}} d_2 \\ c_2 &= d_3 \mathbin{\textcircled{\scriptsize y}} d_2 \\ d_1 &= c_1 \mathbin{\textcircled{\scriptsize x}} d_3. \end{aligned}$$

Note that $y \notin fn(c_1)$, since otherwise this type of splitting is not applicable.

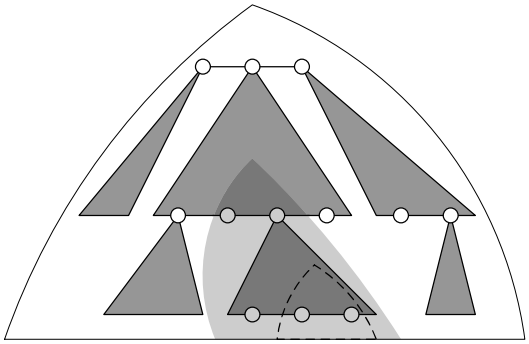

Fig. 6. Splitting type 1

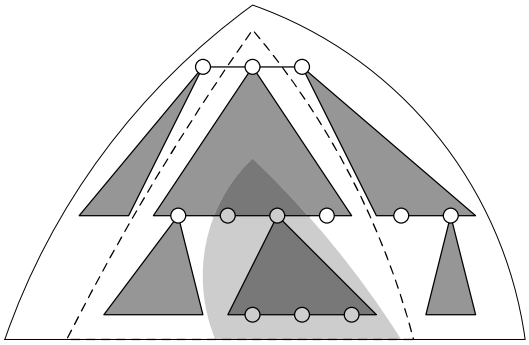

Fig. 7. Splitting type 2

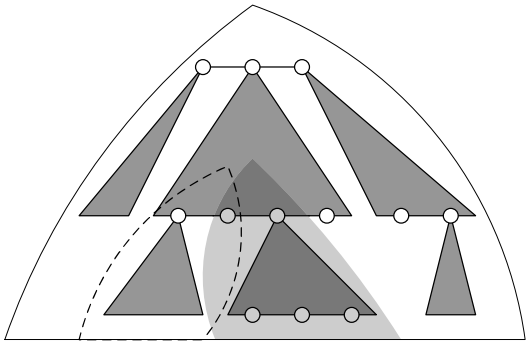

Fig. 8. Splitting type 3

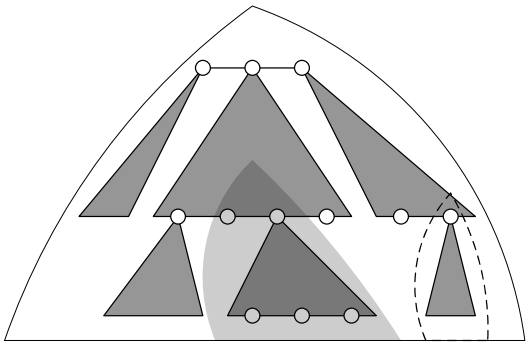

Fig. 9. Splitting type 4

Spoiler would be able to play the CMP move on the game in (14), so Duplicator must be able to split c_2' as $d_3' \mathbin{⊛} d_2'$ such that

$$((d_3, \sigma), (d_3', \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{32}$$
$$((d_2, \sigma), (d_2', \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{33}$$

Note that $fn(d_3') \subseteq fn(c_2') \cup \{\acute{y}\}$. Also, by Lemma 33, $\acute{y} \notin fn(c_1')$ since $y \notin fn(c_1)$. Hence $d_1' = c_1' \mathbin{⊛} d_3'$ is well-defined. By downward closure on (13) and (32) and by the inductive hypothesis,

$$((c_1 \mathbin{⊛} d_3, \sigma), (c_1' \mathbin{⊛} d_3', \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{34}$$

By Lemma 15, since $\acute{y} \notin fn(c_1')$,

$$\begin{aligned} c_1' \mathbin{⊛} c_2' &= c_1' \mathbin{⊛} (d_3' \mathbin{⊛} d_2') \\ &= (c_1' \mathbin{⊛} d_3') \mathbin{⊛} d_2' \\ &= d_1' \mathbin{⊛} d_2'. \end{aligned}$$

Hence, by (34) and by downward closure on (33), Duplicator has a winning strategy if she splits c' as $d_1' \mathbin{⊛} d_2'$.

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

$$\begin{aligned} c_1 \mathbin{⊛} c_2 &= (d_1 \mathbin{⊛} d_3) \mathbin{⊛} c_2 \\ &= d_1 \mathbin{⊛} (d_3 \mathbin{⊛} c_2) \\ &= d_1 \mathbin{⊛} d_2 \\ c_1 &= d_1 \mathbin{⊛} d_3 \\ d_2 &= d_3 \mathbin{⊛} c_2. \end{aligned}$$

Spoiler would be able to play the CMP move on the game in (13), so Duplicator must be able to split c_1' as $d_1' \mathbin{⊛} d_3'$ such that

$$((d_1, \sigma), (d_1', \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{35}$$
$$((d_3, \sigma), (d_3', \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{36}$$

Note that $fn(d_3') \subseteq fn(c_1')$ and that, by Lemma 33, $\acute{x} \in fn(d_3')$ since $x \in fn(d_3)$. Thus $d_2' = d_3' \mathbin{⊛} c_2'$ is well-defined. By downward closure on (36) and (14) and by the inductive hypothesis,

$$((d_3 \mathbin{⊛} c_2, \sigma), (d_3' \mathbin{⊛} c_2', \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{37}$$

By Lemma 15, since either $\acute{x} = \acute{y}$ or $\acute{x} \notin fn(d_1')$ (since $\acute{x} \in fn(d_3')$ and $c_1' = d_1' \mathbin{⊛} d_3'$),

$$\begin{aligned} c_1' \mathbin{⊛} c_2' &= (d_1' \mathbin{⊛} d_3') \mathbin{⊛} c_2' \\ &= d_1' \mathbin{⊛} (d_3' \mathbin{⊛} c_2') \\ &= d_1' \mathbin{⊛} d_2'. \end{aligned}$$

Hence, by downward closure on (35) and by (37), Duplicator has a winning strategy if she splits c' as $d_1' \mathbin{⊛} d_2'$.

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

$$c_1 = d_3 \textcircled{x} d_4 \qquad\qquad c_2 = d_5 \textcircled{y} d_6$$

where

$$d_1 = d_3 \textcircled{x} d_5 \qquad\qquad d_2 = d_4 \textcircled{x} d_6$$

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 $d_5 = \bar{d}_5 \mid y$. If the former, for instance, we have

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

Spoiler could play the CMP move on (13), so $c'_1 = d'_3 \textcircled{\acute{x}} d'_4$ such that

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{38}$$
$$((d_4, \sigma), (d'_4, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{39}$$

Similarly, from (14), we have that $c'_2 = d'_5 \textcircled{\acute{y}} d'_6$ such that

$$((d_5, \sigma), (d'_5, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{40}$$
$$((d_6, \sigma), (d'_6, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{41}$$

Note that $\acute{x} \in fn(d'_3) \subseteq fn(c'_1)$ and $fn(d'_5) \subseteq fn(c'_2) \cup \{\acute{y}\}$. Furthermore, either $y = x$ and $\acute{y} = \acute{x}$ (since otherwise Spoiler could win (13) by playing CMP followed by VAR), or $\acute{y} \notin fn(d'_3)$, by Lemma 33 since $y \notin fn(d_3)$. Thus $d'_1 = d'_3 \textcircled{\acute{x}} d'_5$ is well-defined. Similarly, $\acute{x} \in fn(d'_4) \subseteq fn(c'_1)$ and $fn(d'_6) \subseteq fn(c'_2)$, so $d'_2 = d'_4 \textcircled{\acute{x}} d'_6$ is well-defined. Hence, by downward closure on (38), (40), (39) and (41), and by the inductive hypothesis, we have

$$((d_3 \textcircled{x} d_5, \sigma), (d'_3 \textcircled{\acute{x}} d'_5, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{42}$$
$$((d_4 \textcircled{x} d_6, \sigma), (d'_4 \textcircled{\acute{x}} d'_6, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{43}$$

It remains to show that $c'_1 \textcircled{\acute{x}} c'_2 = d'_1 \textcircled{\acute{y}} d'_2$. By Lemma 15,

$$\begin{aligned} c'_1 \textcircled{\acute{x}} c'_2 &= (d'_3 \textcircled{\acute{x}} d'_4) \textcircled{\acute{x}} (d'_5 \textcircled{\acute{y}} d'_6) \\ &= d'_3 \textcircled{\acute{x}} (d'_4 \textcircled{\acute{x}} (d'_5 \textcircled{\acute{y}} d'_6)). \end{aligned}$$

Now suppose that $d_4 = \bar{d}_4 \mid x$ and $d_5 = y \mid \bar{d}_5$. By Lemma 34, we must have that $d'_4 = \bar{d}'_4 \mid \acute{x}$ and $d'_5 = \acute{y} \mid \bar{d}'_5$. Thus,

$$\begin{aligned} d'_4 \textcircled{\acute{x}} (d'_5 \textcircled{\acute{y}} d'_6) &= \bar{d}'_4 \mid d'_6 \mid \bar{d}'_5 \\ &= d'_5 \textcircled{\acute{y}} (d'_4 \textcircled{\acute{x}} d'_6). \end{aligned}$$

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 15 (recalling that either $\acute{y} = \acute{x}$ or $\acute{y} \notin fn(d'_3)$),

$$\begin{aligned} c'_1 \mathbin{⊛}_{\acute{x}} c'_2 &= d'_3 \mathbin{⊛}_{\acute{x}} (d'_5 \mathbin{⊛}_{\acute{y}} (d'_4 \mathbin{⊛}_{\acute{x}} d'_6)) \\ &= (d'_3 \mathbin{⊛}_{\acute{x}} d'_5) \mathbin{⊛}_{\acute{y}} (d'_4 \mathbin{⊛}_{\acute{x}} d'_6) \\ &= d'_1 \mathbin{⊛}_{\acute{y}} d'_2, \end{aligned}$$

as required. We can see that Duplicator could respond to Spoiler's move by splitting c' as $d'_1 \mathbin{⊛}_{\acute{y}} d'_2$ and that, by (42) and (43), 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) $y \neq x$:

$$\begin{aligned} c_1 \mathbin{⊛}_x c_2 &= (d_3 \mathbin{⊛}_y d_2) \mathbin{⊛}_x c_2 \\ &= (d_3 \mathbin{⊛}_x c_2) \mathbin{⊛}_y d_2 \\ &= d_1 \mathbin{⊛}_y d_2 \\ c_1 &= d_3 \mathbin{⊛}_y d_2 \\ d_1 &= d_3 \mathbin{⊛}_x c_2 \end{aligned}$$

Since $y \neq x$, we know that $\acute{y} \neq \acute{x}$, for otherwise Spoiler would have a winning strategy for (13) by playing the CMP move followed by the VAR move. Spoiler would be able to play the CMP move on the game in (13), so we know that $c'_1 = d'_3 \mathbin{⊛}_{\acute{y}} d'_2$ for some d'_3, d'_2 such that

$$((d_3, \sigma), (d'_3, \sigma'), (3n-1, 0, \mathcal{L})) \in DW \tag{44}$$
$$((d_2, \sigma), (d'_2, \sigma'), (3n-1, 0, \mathcal{L})) \in DW. \tag{45}$$

Note that $fn(d'_3) \subseteq fn(c'_1) \cup \{\acute{y}\}$. Also, by Lemma 33, $\acute{x} \in fn(d'_3)$ and $\acute{y} \notin fn(c'_2)$. Thus $d'_1 = d'_3 \mathbin{⊛}_{\acute{x}} c'_2$ is well-defined. By downward closure on (44) and (14), and by the inductive hypothesis,

$$((d_3 \mathbin{⊛}_x c_2, \sigma), (d'_3 \mathbin{⊛}_{\acute{x}} c'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{46}$$

By Lemma 16, since $\acute{x} \in fn(d'_3)$ and $\acute{y} \notin fn(c'_2)$, $(d'_3 \mathbin{⊛}_{\acute{y}} d'_2) \mathbin{⊛}_{\acute{x}} c'_2 = (d'_3 \mathbin{⊛}_{\acute{x}} c'_2) \mathbin{⊛}_{\acute{y}} d'_2$. Hence, by (46) and downward closure on (45), we know that Duplicator has a winning strategy by splitting c' as $d'_1 \mathbin{⊛}_{\acute{y}} d'_2$.

(b) $y = x$: The reason we consider this case separately is that the construction for the previous case would give d_3 with two holes labelled x. To avoid this, we rename the x hole of c_1 as the z hole of $\bar{c}_1$. For some $z \notin fn(c_1) \cup fn(c_2) \cup range(\sigma)$,

$$\begin{aligned} c &= ((d_3 \mathbin{⊛}_x d_2) \mathbin{⊛}_z x) \mathbin{⊛}_x c_2 \\ &= (d_3 \mathbin{⊛}_x d_2) \mathbin{⊛}_z c_2 \\ &= (d_3 \mathbin{⊛}_z c_2) \mathbin{⊛}_x d_2 \\ &= d_1 \mathbin{⊛}_x d_2 \\ c_1 &= \bar{c}_1 \mathbin{⊛}_z x \\ \bar{c}_1 &= d_3 \mathbin{⊛}_x d_2 \\ d_1 &= d_3 \mathbin{⊛}_z c_2. \end{aligned}$$

Since $y = x$, we know that $\acute{y} = \acute{x}$, for otherwise Spoiler would have a winning strategy for (13) by playing the CMP move followed by the VAR move. By Lemma 36, for some $\acute{z} \notin fn(c'_1) \cup fn(c'_2) \cup range(\sigma')$,

$$((c_1, \sigma[\gamma \mapsto z]), (c'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-1, 0, \mathcal{L})) \in DW \tag{47}$$
$$((c_2, \sigma[\gamma \mapsto z]), (c'_2, \sigma'[\gamma \mapsto \acute{z}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{48}$$

Spoiler could play the CMP move on the game in (47), splitting c_1 as $\bar{c}_1 \circledz x$, so $c'_1 = \bar{c}'_1 \circledz \hat{c}'_1$ such that

$$((\bar{c}_1, \sigma[\gamma \mapsto z]), (\bar{c}'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-2, 0, \mathcal{L})) \in DW \tag{49}$$
$$((x, \sigma[\gamma \mapsto z]), (\hat{c}'_1, \sigma'[\gamma \mapsto \acute{z}]), (3n-2, 0, \mathcal{L})) \in DW. \tag{50}$$

Since $3n-2 \geq 1$, (50) implies that $\hat{c}'_1 = \acute{x}$. Spoiler could then play the CMP move on the game in (49), splitting $\bar{c}_1$ as $d_3 \circledx d_2$, so $\bar{c}'_1 = d'_3 \circledx d'_2$ such that

$$((d_3, \sigma[\gamma \mapsto z]), (d'_3, \sigma'[\gamma \mapsto \acute{z}]), (3n-3, 0, \mathcal{L})) \in DW \tag{51}$$
$$((d_2, \sigma[\gamma \mapsto z]), (d'_2, \sigma'[\gamma \mapsto \acute{z}]), (3n-3, 0, \mathcal{L})) \in DW. \tag{52}$$

By construction and by Lemma 33 (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 33 and by definition, neither $\acute{x}$ nor $\acute{z}$ occurs in c'_2. Hence $d'_1 = d'_3 \circledz c'_2$ is well-defined. Now we may apply the inductive hypothesis, using (51) and downward closure on (48), to obtain

$$((d_3 \circledz c_2, \sigma[\gamma \mapsto z]), (d'_3 \circledz c'_2, \sigma'[\gamma \mapsto \acute{z}]), (n-1, 0, \mathcal{L})) \in DW. \tag{53}$$

By Proposition 30 and downward closure on (53) and (52), we have

$$((d_1, \sigma), (d'_1, \sigma'), (n-1, 0, \mathcal{L})) \in DW \tag{54}$$
$$((d_2, \sigma), (d'_2, \sigma'), (n-1, 0, \mathcal{L})) \in DW. \tag{55}$$

Note that, by construction and by Lemma 33, $\acute{x}, \acute{z} \notin fn(d'_2)$ and $\acute{x} \notin fn(c'_2)$. Thus, by Lemmata 15 and 16,

$$\begin{aligned} c' &= (\bar{c}'_1 \circledz \acute{x}) \circledx c'_2 \\ &= ((d'_3 \circledx d'_2) \circledz \acute{x}) \circledx c'_2 \\ &= (d'_3 \circledx d'_2) \circledz (\acute{x} \circledx c'_2) \\ &= (d'_3 \circledx d'_2) \circledz c'_2 \\ &= (d'_3 \circledz c'_2) \circledx d'_2 \\ &= d'_1 \circledx d'_2. \end{aligned}$$

Hence Duplicator could respond by splitting c' as $d'_1 \circledx d'_2$ and by (54) and (55) 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 (that is, $y \notin fn(c_1) \cup fn(c_2) \cup range(\sigma)$).

In case 1, $y = \sigma(\alpha)$ for some α, and Duplicator can respond with $\acute{y} = \sigma'(\alpha)$. By the first case of Lemma 36, we know

$$((c_1, \sigma[\beta \mapsto y]), (c_1', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW \tag{56}$$
$$((c_2, \sigma[\beta \mapsto y]), (c_2', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW \tag{57}$$

and so, by downward closure and the inductive hypothesis,

$$((c, \sigma[\beta \mapsto y]), (c', \sigma'[\beta \mapsto \acute{y}]), (n-1, 0, \mathcal{L})) \in DW. \tag{58}$$

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 (13). Let $\acute{y}$ be Duplicator's response for her winning strategy:

$$((c_1, \sigma[\beta \mapsto y]), (c_1', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{59}$$

Since $y \notin range(\sigma)$ and $3n - 2 \geq 2$, $\acute{y} \notin range(\sigma')$.[5] Also, since $y \in fn(c_1)$ and $3n - 2 \geq 2$, $\acute{y} \in fn(c_1')$ by Lemma 33. 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 36,

$$((c_2, \sigma[\beta \mapsto y]), (c_2', \sigma'[\beta \mapsto \acute{y}]), (3n-1, 0, \mathcal{L})) \in DW. \tag{60}$$

So by downward closure and the inductive hypothesis we have

$$((c, \sigma[\beta \mapsto y]), (c, \sigma'[\beta \mapsto \acute{y}]), (n-1, 0, \mathcal{L})) \in DW. \tag{61}$$

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 (14). 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 (15) and concluded that Duplicator has a winning strategy in each case, we have shown that (15) holds. □

[5] To see this, suppose that Spoiler were to play the CMP move in (59) and split $c_1 = y \circledcirc{y} c_1$. Then he would have a winning strategy, since there is some γ with $\acute{y} = \sigma'(\gamma)$ but $y \neq \sigma(\gamma)$.

Corollary 38 (Multi-step Move Elimination)**.** *For all ranks $r = (n, s, \mathcal{L})$, for all $c, c' \in \mathcal{C}^{\mathrm{m}}$ and for all domain-coincident environments σ, σ', if*

$$((c, \sigma), (c', \sigma'), (3^s(n+1), 0, \mathcal{L})) \in DW \tag{62}$$

then

$$((c, \sigma), (c', \sigma'), (n, s, \mathcal{L})) \in DW. \tag{63}$$

Proof. By induction on values of s.

If $s = 0$ then the conclusion follows by downward closure.

For $s > 0$, let us consider how an arbitrary instance of the game in (63) would proceed. Until Spoiler first plays an adjunct move, Duplicator may respond in the game in (63) just as she would for the game in (62), preventing Spoiler from winning up to that point. Spoiler first plays an adjunct move for, say, the $(k+1)^{\text{th}}$ move (so $k \leq n$). At this stage, the game state is

$$((c_1, \sigma_1), (c_1', \sigma_1'), (n-k, s, \mathcal{L})) \tag{64}$$

and we know

$$((c_1, \sigma_1), (c_1', \sigma_1'), (3^s(n+1) - k, 0, \mathcal{L})) \in DW. \tag{65}$$

Spoiler now plays either the LEF or the RIG move on (64); let us consider each case.

LEF move. Spoiler chooses one of c_1, c_1' (assume without loss of generality that he picks c_1), $x = \sigma_1\alpha$ (let $\acute{x} = \sigma_1'\alpha$) and $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ with $d_2 = d_1 \circledast{x} c_1$. By downward closure on (65),

$$((c_1, \sigma_1), (c_1', \sigma_1'), (3 \cdot 3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW. \tag{66}$$

Note that $\sigma_1\beta = \sigma_1\gamma \iff \sigma_1'\beta = \sigma_1'\gamma$. This follows from (65) by considering that, if $x = \sigma_1\beta = \sigma_1\gamma$, Spoiler could play the CMP move choosing $x = \sigma_1\beta$ split $c_1 = x \circledast{x} c_1$. Duplicator's response for her winning strategy must split $c_1' = \acute{x} \circledast{\acute{x}} c_1'$ with $\acute{x} = \sigma_1'\beta$ and $\acute{x} = \sigma_1'\gamma$, or else Spoiler would be able to win by playing the VAR move.

Now let d_1' be d_1 with the hole labels renamed as follows: for each β, $\sigma_1\beta$ is renamed to $\sigma_1'\beta$; and the remaining hole labels (which are distinct from $fn(c_1) \cup range(\sigma_1)$) renamed to be fresh with respect to $fn(c_1') \cup range(\sigma_1')$. From this construction, $d_1, \sigma_1 \models K \iff d_1', \sigma_1' \models K$ for all K, since each variable may be instantiated according to the renaming of hole labels. Thus, by game completeness,

$$((d_1, \sigma_1), (d_1', \sigma_1'), (3 \cdot 3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW, \text{ and} \tag{67}$$

$$((d_1, \sigma_1), (d_1', \sigma_1'), (n-k, s-1, \mathcal{L})) \in DW. \tag{68}$$

Notice that $d_2' = d_1' \circledast{\acute{x}} c_1'$ is defined by construction. By Proposition 37, from (66) and (67) we get

$$((d_1 \circledast{x} c_1, \sigma_1), (d_1' \circledast{\acute{x}} c_1', \sigma_1'), (3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW. \tag{69}$$

Hence, by the inductive hypothesis,

$$((d_1 \circledast c_1, \sigma_1), (d_1' \circledast c_1', \sigma_1'), (n-k, s-1, \mathcal{L})) \in DW. \tag{70}$$

From (68) and (70), we see that Duplicator has a winning strategy in this case, by playing d_1' and d_2'.

RIG move. Again, Spoiler chooses, without loss of generality, c_1, $x = \sigma_1\alpha$ (let $\acute{x} = \sigma_1'\alpha$) and $d_1, d_2 \in \mathcal{C}^{\mathrm{m}}$ with $d_2 = c_1 \circledast d_1$. As before,

$$((c_1, \sigma_1), (c_1', \sigma_1'), (3 \cdot 3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW. \tag{71}$$

Let d_1' be the relabelling of d_1, as previously. By construction and by Lemma 33 (using (65), since $3^s(n+1) - k \geq 3$), $d_2' = c_1' \circledast d_1'$ is defined. Also, by game completeness,

$$((d_1, \sigma_1), (d_1', \sigma_1'), (3 \cdot 3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW, \text{ and} \tag{72}$$
$$((d_1, \sigma_1), (d_1', \sigma_1'), (n-k, s-1, \mathcal{L})) \in DW. \tag{73}$$

Thus, by Proposition 37, we get

$$((c_1 \circledast d_1, \sigma_1), (c_1' \circledast d_1', \sigma_1'), (3^{s-1}(n-k+1), 0, \mathcal{L})) \in DW. \tag{74}$$

Hence, by the inductive hypothesis,

$$((c_1 \circledast d_1, \sigma_1), (c_1' \circledast d_1', \sigma_1'), (n-k, s-1, \mathcal{L})) \in DW. \tag{75}$$

So, by (73) and (75), Duplicator has a winning strategy in this case also, by playing d_1' and d_2'. □

These game results are now translated to results in the logic in the following theorem.

Theorem 39 (Adjunct Elimination)**.** *For any CL^{m}_{Tree} formula of rank $r = (n, s, \mathcal{L})$, there exists an equivalent formula of rank $r' = (3^s(n+1), 0, \mathcal{L})$.*

Proof. Suppose that K is a formula of rank r and having free variables in $\mathcal{V}$.[6] Let $\mathcal{S} = \{(d, \rho) : d, \rho \models K\}$. By game soundness, if $(c, \sigma) \in \mathcal{S}$ and $(c', \sigma') \notin \mathcal{S}$ then $((c, \sigma), (c', \sigma'), r) \in SW$. By Corollary 38, this means $((c, \sigma), (c', \sigma'), r') \in SW$. Hence, by game completeness, there is a formula $K_{(c,\sigma),(c',\sigma')}$ of rank r' and free variables in $\mathcal{V}$, which discriminates (c, σ) from (c', σ').

Therefore, by Lemma 28, there is a formula K' of rank r', free variables in $\mathcal{V}$, which defines $\mathcal{S}$. Hence, K' is equivalent to K. □

[6] We assume that all environments in this proof have domain $\mathcal{V}$.

6 Conclusions

We have introduced multi-holed Context Logic for trees (CL^{m}_{Tree}) and proved an adjunct-elimination result for this logic. 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 [10], 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 [13] 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 [14] 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 [7] 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 maximum nesting depth of adjoint connectives of the original formula. The total number of connectives might be larger still, although by Lemma 24 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 [6], which compares logics on *open* formulae containing propositional variables. Despite our expressivity results on *closed* formulae in this paper,

[7] We thank an anonymous referee of our conference paper [11] for this question.

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 [6] 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] S. S. Ishtiaq, P. W. O'Hearn, BI as an assertion language for mutable data structures, in: POPL 2001, ACM Press, New York, 2001, pp. 14–26.

[2] J. C. Reynolds, Separation Logic: a logic for shared mutable data structures, in: LICS 2002, IEEE Computer Society, Los Alamitos, 2002, pp. 55–74.

[3] H. Yang, P. W. O'Hearn, A semantic basis for local reasoning, in: M. Nielsen, U. Engberg (Eds.), ETAPS 2002 and FOSSACS 2002, Vol. 2303 of LNCS, Springer, Heidelberg, 2002, pp. 402–416.

[4] L. Cardelli, A. D. Gordon, Anytime, anywhere: modal logics for mobile ambients, in: POPL 2000, ACM Press, New York, 2000, pp. 365–377.

[5] C. Calcagno, P. Gardner, U. Zarfaty, Context Logic and tree update, in: POPL 2005, ACM Press, New York, 2005, pp. 271–282.

[6] C. Calcagno, P. Gardner, U. Zarfaty, Context logic as modal logic: completeness and parametric inexpressivity, in: POPL 2007, ACM Press, New York, 2007, pp. 123–134.

[7] E. Lozes, Adjuncts elimination in the static Ambient Logic, in: F. Corradini, U. Nestmann (Eds.), EXPRESS 2003, Vol. 96 of ENTCS, Elsevier, Amsterdam, 2003, pp. 51–72.

[8] A. Dawar, P. Gardner, G. Ghelli, Adjunct elimination through games in static Ambient Logic, in: K. Lodaya, M. Mahajan (Eds.), FSTTCS 2004, Vol. 3328 of LNCS, Springer, Heidelberg, 2004, pp. 211–223.

[9] C. Calcagno, P. Gardner, U. Zarfaty, Separation Logic, Ambient Logic and Context Logic: parametric inexpressivity results, unpublished (2006).

[10] T. Dinsdale-Young, Adjunct elimination in Context Logic, Master's thesis, Imperial College London (2006).

[11] C. Calcagno, T. Dinsdale-Young, P. Gardner, Adjunct elimination in Context Logic for trees, in: APLAS 2007, LNCS, Springer, Heidelberg, 2007, pp. 255–270.

[12] P. O'Hearn, D. Pym, Logic of bunched implications, Bulletin of Symbolic Logic 5 (2) (1999) 215–244.

[13] U. Heuter, First-order properties of trees, star-free expressions, and aperiodicity, Informatique théorique et applications 25 (2) (1991) 125–145.

[14] M. Bojańczyk, Forest expressions, in: J. Duparc, T. A. Henzinger (Eds.), CSL 2007, Vol. 4646 of LNCS, Springer, Heidelberg, 2007, pp. 146–160.