Adjunct Elimination in Context Logic

Thomas Dinsdale-Young

June 16, 2006

Supervisor: Philippa Gardner
With Cristiano Calcagno
Second Marker: Iain Hodkinson

Abstract

In recent years, interest has been growing in the field of spatial logics, which have a variety of useful applications, including modular reasoning about data update and specifying safety properties. Context Logic is a new spatial logic designed for reasoning about data on the level of data-structures. It extends first-order logic with connectives that permit reasoning about disjoint portions of data. A feature of this and related logics is the presence of adjunct connectives, which are important for defining weakest preconditions in Hoare reasoning and for specifying perfect firewalls, for instance. Recent results, first by Lozes, then Dawar, Gardner and Ghelli, have shown that adjuncts can be eliminated from related logics.

In this report, we consider adjunct elimination in Context Logic, particularly based on the model of ordered trees. We provide a counterexample to the elimination of one of the adjuncts and a proof of elimination of the other. We also discuss possible avenues for future investigation, such as modifications to the Logic that may permit the elimination of the adjunct that was not possible within the current definition for Context Logic for Trees.

Acknowledgements

I would particularly like to thank Philippa Gardner, my supervisor, for her help and support, as well as her dedication and enthusiasm for this project. Thanks are also due to Cristiano Calcagno for his invaluable help throughout the project. I would also like to thank Ian Hodkinson, my second marker, for his useful feedback on the project.

Contents

List of Figures

List of Tables

1 Introduction

Structured data is a ubiquitous concept in computer systems: hard disk storage systems, distributed databases and memory heaps are all examples of this. Writing programs that correctly update and manipulate such structures is difficult, and when we have a correct program we would also like to prove that it is indeed correct.

Hoare Logics are a standard tool for reasoning about data update. O'Hearn, Reynolds and Yang introduced Separation Logic[6], extending Hoare's approach to reasoning by extending first-order logic with a connective (and a corresponding adjunct) that expresses the notion that properties hold on disjoint portions of a low-level memory heap. This allows one to apply Hoare reasoning to address the particular section of the data that is being manipulated, while also expressing the notion that the remainder of the data is unchanged. Thus, problems which had eluded modular reasoning in the traditional Hoare approach, such as pointer arithmetic, could now be addressed in a modular fashion. Since the logic operates on low-level heaps, one cannot reason about complex data-structures, such as trees, without losing the abstraction the data-structure provides.

Context Logic[2] is a new logic developed by Calcagno, Gardner and Zarfaty to address this problem by reasoning on the same level as the data-structures that are to be operated upon. Like Separation Logic, it extends first order logic with spatial connectives and adjuncts, which permit reasoning about disjoint portions of a data-structure. This development was motivated by the wish to apply the modularity of Separation Logic to problems in reasoning about tree update (XML update). Initial attempts were to develop a Hoare Logic based on the Ambient Logic[3] of Cardelli and Gordon, a logic for reasoning locally about static trees (such as for reasoning about security properties of firewalls). It was found, however, that Ambient Logic was not sufficiently expressive for this purpose. Thus, Context Logic introduces the notion that a tree may be split at any point — one can then reason about the sub-tree that is to be modified separately from the context. Context Logic, therefore, is able to reason both about data-structures and contexts — essentially a data-structure from which some sub-structure has been removed.

A feature common to Separation Logic, Context Logic and Ambient Logic, which are all spatial logics, is the presence of the adjuncts[1] of the spatial connectives. These adjuncts provide important properties of their respective logics. For instance, they are fundamental to expressing the weakest preconditions in Hoare reasoning with Separation Logic and Context Logic. In Ambient Logic, they can be used for expressing perfect firewalls.

Fascinatingly, however, Lozes proved that the adjunct connectives add no expressive power in Separation Logic and static Ambient Logic (without quantifiers)[5]. That is, every formula that uses adjunct connectives is equivalent to (i.e. satisfied by exactly the same trees or heaps) some formula which does not make use of such connectives. This result is rather surprising since, in the logic with adjuncts, validity may be reduced to model checking, but without the adjuncts, validity is undecidable, yet model checking is decidable in PSPACE. This implies that, while a formula which makes use of adjuncts has

[1] The term 'adjoint' is also used in the literature.

an equivalent adjunct-free formula, determining exactly what that formula is is not decidable.

Dawar, Gardner and Ghelli later produced a new proof of adjunct elimination based on Ehrenfeucht-Fraïssé games[4]. This technique resulted a proof that is more modular in the logical connectives, and gave a better insight into the generality of adjunct elimination. Their result also demonstrated that the adjunct-free formula corresponding to some formula with adjuncts was contained within a finite set, parametrised by its use of connectives, a further curiosity given the undecidability result we have described.

Even more recent work by Calcagno, Gardner and Zarfaty has looked at parametric expressivity. This new type of result helps to explain the apparent contradiction introduced by the eliminability of adjuncts. The parametric inexpressivity of the logic without adjuncts means that a formula with adjuncts that has parameters (that is, we can replace specific components with arbitrary formulae) has no equivalent without adjuncts. Thus the equivalent formula for the adjunct formula with one set of parameters may have one structure, while the equivalent formula for that same adjunct formula when given different parameters may have a very different structure.

1.1 Overview

An interesting and natural question to ask is "Do these adjunct elimination results extend to Context Logic?". The tree model is of particular interest, since Context Logic's spatial splitting connective is more powerful than those for Ambient Logic, and consequently an adjunct that is conceptually different to those of Ambient Logic arises. This adjunct states that a context satisfies the property that whenever any tree satisfying a particular property is placed within the context hole the resulting tree satisfies a second specified property.

We shall begin by giving formal definitions for Context Logic, and the tree model we shall be working with. Next, we shall look at the general problem of adjunct elimination for Context Logic, and show that there are models for Context Logic which do not admit adjunct elimination. We shall then show an example of a logic and model for which adjunct elimination is possible, but for which the proof technique based on Ehrenfeucht-Fraïssé games cannot be applied in a naïve fashion.

Following on from this, we shall formalise Ehrenfeucht-Fraïssé games for Context Logic for Trees, which are adapted in a straight-forward manner from those given by Dawar *et al.* for Ambient Logic. Next, we shall look specifically at the question of adjunct elimination in Context Logic for Trees. We shall give a counterexample to the elimination of the new adjunct, but give a games-based proof that the other adjunct may be eliminated without reducing the expressivity of the logic. An evaluation section shall follow, in which we shall discuss how the properties we have shown may be adapted to other models for Context Logic and consider how it may be possible to adapt the logic in order to overcome the non-eliminability of the new adjunct.

1.2 Background

1.2.1 Separation Logic

As we have stated previously, the key to Separation Logic's success in reasoning about data update is its use of a separating connective or "spatial conjunction": $P * Q$. The meaning attached to this is that both P and Q hold but for distinct portions of the data.

The Hoare triple $\{P\}\ C\ \{Q\}$ essentially states that, if P holds initially, then Q will hold after the program C is executed. The separating connective means that a rule can be introduced that says if $\{P\}\ C\ \{Q\}$ then also $\{P * R\}\ C\ \{Q * R\}$. That is, since '$*$' ensures that R refers to a part of the heap that is entirely disjoint from P and Q, when R holds before the heap update performed by C it also holds after.

As well as '$*$', Separation Logic includes an adjunct connective '$-\!\!*$'. When $P -\!\!* Q$ is satisfied by some heap, it means that when we combine this with some heap that satisfies P the result satisfies Q. This is important in expressing weakest preconditions: the most general condition that can be allowed to hold before execution such that some arbitrary given condition holds after execution. For instance, the following holds:

$$\{(E \mapsto -) * ((E \mapsto F) -\!\!* Q)\}\ [E] := F\ \{Q\} \tag{1.1}$$

What this means is that, for property Q to hold after F has been stored at memory location E, the heap should map E to $-$ and contain a disjoint portion such that, when the heap mapping E to F is combined with it, the resulting heap satisfies Q.

The adjuncts in Context Logic are similarly important for specifying such weakest preconditions.

1.2.2 Adjunct Elimination

Lozes' method for proving adjunct elimination was to define a set of bisimulation equivalences between tree structures. These equivalences state that the trees cannot be distinguished in a certain number of test steps, a notion that is not dissimilar to the way in which games, parametrised by rank, may be used to distinguish between trees. He proves that the number of equivalence classes for a given number of test steps is finite. He then introduces characteristic adjunct-free formulae for trees, parametrised by the number of test steps, which are only satisfied by trees that are bisimilar (in the given number of test steps) to the defining tree. Finally, he establishes that there is an equivalent adjunct-free formula to any formula with adjuncts by taking the disjunction of the characteristic formulae of the a finite set of non-equivalent (with respect to a number of test steps derived from the adjunct formula) trees which satisfy the adjunct formula.

In the games-based proof of Dawar *et al.*, a different approach is used, establishing that there are finitely many inequivalent formulae of a given rank — such formulae correspond closely with the games played on that rank with regard to which trees they can distinguish. It is then shown that the adjunct moves in the games do not assist with discriminating between pairs of trees. The relationship between games and formulae means that this can be used to

infer that the inclusion of adjuncts does not increase the ability of formulae to distinguish between trees, and thus they do not increase the expressive power of the logic.

An additional result in these papers is that quantification ('$\exists$' and '$\forall$') prevents adjuncts from being eliminated in the logic. For this reason, we shall not consider logics with quantification here.

2 Context Logic

In this section, we present the definitions and theory for Context Logic, and, in particular, Context Logic for trees, which we shall use in subsequent sections.

2.1 Definitions

2.1.1 Basic Context Logic

We consider the general theory of Context Logic first, introducing the grammar, the notion of a model and the forcing semantics for a model (that is, the relations that state when an element of the model can be said to satisfy a formula of the logic).

Definition 2.1. The basic Context Logic consists of a set of *data formulae* and a set of *context formulae* which are defined by the grammars:

$$\begin{array}{lrll} \text{data formulae} & P ::= K(P) \mid K \lhd P & \text{structural formulae} & \\ & P \Rightarrow P \mid \mathit{false} & \text{additive formulae} & (2.1) \\ \text{context formulae} & K ::= I \mid P \rhd P & \text{structural formulae} & \\ & K \Rightarrow K \mid \mathit{False} & \text{additive formulae} & (2.2) \end{array}$$

We shall also use derived Boolean formulae, such as $P \wedge P$ or $\neg K$, which may be defined easily in terms of the given Boolean formulae. (e.g. $\neg K \triangleq K \Rightarrow \mathit{False}$, $P_1 \wedge P_2 \triangleq ((P_1 \Rightarrow (P_2 \Rightarrow \mathit{false})) \Rightarrow \mathit{false})$.) All of the standard classical Boolean formulae can be derived in this way.

Further, we define the following additional derived formulae:

- $\diamond P \triangleq \mathit{True}(P)$
- $P_1 \blacktriangleright P_2 \triangleq \neg(P_1 \rhd \neg P_2)$
- $K \blacktriangleleft P \triangleq \neg(K \lhd \neg P)$

Definition 2.2. A *model* $\mathcal{M}$ for Context Logic is a tuple $(\mathcal{D}, C, ap, \mathbf{I})$ such that

1. $\mathcal{D}$ and C are sets;
2. $ap : C \times \mathcal{D} \rightharpoonup \mathcal{D}$ is a partial function, called the *application function*;
3. $\mathbf{I} \subseteq C$ acts as a left identity to ap, that is
   - $\forall a \in \mathcal{D}, \exists i \in \mathbf{I}.ap(i, a)$ is defined, and
   - $\forall a \in \mathcal{D}, \forall i \in \mathbf{I}.$if $ap(i, a)$ is defined then $ap(i, a) = a$.

This general definition of a model has many useful and important instances, some of which we shall examine in much greater detail later. In the 'natural' examples which motivate this abstract definition, the set $\mathcal{D}$ is usually considered to be a set of data-structures, and the set C to be of contexts: effectively data-structures but containing a hole, into which a data-structure may be placed to produce another data-structure. It should come as no surprise, therefore, that these models are generated by recursive grammars.

We shall frequently use the symbols a, a', a_1, etc. (and sometimes b similarly) to range over $\mathcal{D}$, and the symbols c, c', c_1, etc. (and occasionally d similarly) to range over C. We shall also (later) abbreviate $ap(c, a)$ by $c(a)$, a notational convention which emphasises the relationship between the model and the logic.

Definition 2.3. Given a model, $\mathcal{M}$, we define two satisfaction relations: $\mathcal{M}, a \models_D P$ and $\mathcal{M}, c \models_K K$ inductively over the structure of data and context formulae respectively.

$$\mathcal{M}, a \models_D K(P) \text{ iff } \exists c \in C, a' \in \mathcal{D}.(ap(c, a') = a \text{ and } \mathcal{M}, c \models_K K \text{ and } \mathcal{M}, a' \models_D P) \tag{2.3}$$

$$\mathcal{M}, a \models_D K \lhd P \text{ iff } \forall c \in C.(\text{if } \mathcal{M}, c \models_K K \text{ and } ap(c, a) \text{ is defined then } \mathcal{M}, ap(c, a) \models_D P) \tag{2.4}$$

$$\mathcal{M}, a \models_D P_1 \Rightarrow P_2 \text{ iff } \mathcal{M}, a \models_D P_1 \text{ implies } \mathcal{M}, a \models_D P_2 \tag{2.5}$$

$$\mathcal{M}, a \not\models_D \mathit{false} \tag{2.6}$$

$$\mathcal{M}, c \models_K I \text{ iff } c \in \mathbf{I} \tag{2.7}$$

$$\mathcal{M}, c \models_K P_1 \rhd P_2 \text{ iff } \forall a \in \mathcal{D}.(\text{if } \mathcal{M}, a \models_D P_1 \text{ and } ap(c, a) \text{ is defined then } \mathcal{M}, ap(c, a) \models_D P_2) \tag{2.8}$$

$$\mathcal{M}, c \models_K K_1 \Rightarrow K_2 \text{ iff } \mathcal{M}, c \models_K K_1 \text{ implies } c \models_K K_2 \tag{2.9}$$

$$\mathcal{M}, c \not\models_K \mathit{False} \tag{2.10}$$

In the application case, we assume that $ap(c, a') = a$ means that $ap(c, a')$ is defined and equal to a.

These satisfaction relations give meaning to the logic. Expressed informally, the meaning of $K(P)$ is that a satisfying data-structure may be split into a context and another data-structure such that the context satisfies K and the data-structure satisfies P. The $K \lhd P$ and $P \rhd P$ connectives are adjuncts of application. What this means is that they express that what satisfies them gives a certain result when used in an application. Specifically, if a data-structure satisfies $K \lhd P$ then, whenever any context satisfying K is applied to it, the resulting data-structure (if defined) satisfies P. Similarly, if a context satisfies $P_1 \rhd P_2$ then, whenever it is applied to any data-structure that satisfies P_1, the resulting data-structure (if defined) satisfies P_2. The I connective is satisfied by something that, when applied to any data-structure, returns that data-structure (provided the application is defined).

We can also state the informal meanings of the derived connectives we have defined. $\diamond P$ specifies that somewhere property P holds — that a context may be split from the data-structure, leaving a data-structure that satisfies P. $K \blacktriangleleft P$ specifies that *there exists* some context satisfying K such that, when it is applied to the given data-structure, the result satisfies P. $P_1 \blacktriangleright P_2$ specifies that *there exists* some data-structure satisfying P_1 such that, when the given context is applied to it, the result satisfies P_2.

2.1.2 Context Logic with Zero

A useful extension to the basic logic is the addition of a structural data formula 0. Context Logic with Zero has an extended concept of model.

Definition 2.4. A *model*, $\mathcal{M}$, for Context Logic with Zero is a tuple $(\mathcal{D}, C, ap, \mathbf{I}, \mathbf{0})$ where

1. $(\mathcal{D}, C, ap, \mathbf{I})$ is a model for Context Logic;
2. $\mathbf{0} \subseteq \mathcal{D}$;
3. the projection $p : C \to \mathcal{D}$ defined by $p(c) = a \Leftrightarrow \exists o \in \mathbf{0}.ap(c, o) = a$ is a total surjective function;
4. $\forall c \in C, \forall o \in \mathbf{0}.p(c) = o \Rightarrow c \in \mathbf{I}$.

We also extend satisfaction for Context Logic with Zero, by adding the following rule to the inductive definition of the satisfaction relations, which otherwise remains the same as for Context Logic:

$$\mathcal{M}, d \models_D 0 \text{ iff } d \in \mathbf{0} \tag{2.11}$$

The addition of zero essentially allows contexts to be mapped surjectively onto data-structures. Quite naturally then, identity contexts map onto zeros.

With the added 0, we define another derived formula: $P_1 * P_2 \triangleq (0 \triangleright P_1)(P_2)$. Informally, this formula specifies that the some part of the data-structure satisfies P_1 and the rest satisfies P_2.

2.1.3 Context Logic for Trees

The model that we shall primarily concern ourselves with is that of ordered trees.

Definition 2.5. Given a signatures set of names Σ, ranged over by α, we define (ordered) trees and tree contexts by the grammar:

$$\text{trees} \qquad a ::= 0 \mid \alpha[a] \mid a \mid a \tag{2.12}$$

$$\text{contexts} \qquad c ::= \_ \mid \alpha[c] \mid a \mid c \mid c \mid a \tag{2.13}$$

We name the set of trees $\mathcal{D}$ and the set of contexts C.

For ordered trees, certain congruences hold. In particular, ' | ' is associative, and '0' is a left and right identity with respect to it. Expressed formally:

$$0 \mid a = a = a \mid 0 \tag{2.14}$$

$$0 \mid c = c = c \mid 0 \tag{2.15}$$

$$a_1 \mid (a_2 \mid a_3) = (a_1 \mid a_2) \mid a_3 \tag{2.16}$$

$$a_1 \mid (a_2 \mid c_1) = (a_1 \mid a_2) \mid c_1 \tag{2.17}$$

$$a_1 \mid (c_1 \mid a_2) = (a_1 \mid c_1) \mid a_2 \tag{2.18}$$

$$c_1 \mid (a_1 \mid a_2) = (c_1 \mid a_1) \mid a_2 \tag{2.19}$$

Thus, for instance, we say

$$(\alpha[0] \mid \beta[0]) \mid \gamma[0] = \alpha[0] \mid (\beta[0] \mid \gamma[0]) \tag{2.20}$$

$$0 \mid \alpha[\_ \mid 0 \mid 0] = \alpha[\_]. \tag{2.21}$$

We shall also use the following basic and well-known properties concerning the decomposition of trees:

1. (a) If $a_1 \mid a_2 = \kappa[a_3]$, then either $a_1 = \kappa[a_3]$ and $a_2 = 0$ or $a_1 = 0$ and $a_2 = \kappa[a_3]$.
   (b) If $c_1 \mid a_1 = \kappa[c_2]$, then $c_1 = \kappa[c_2]$ and $a_1 = 0$.
   (c) If $a_1 \mid c_1 = \kappa[c_2]$, then $c_1 = \kappa[c_2]$ and $a_1 = 0$.
2. (a) If $a_1 \mid a_2 = a_3 \mid a_4$ then either $a_1 = a_3 \mid a_1'$ and $a_4 = a_1' \mid a_2$ or $a_2 = a_2' \mid a_4$ and $a_3 = a_1 \mid a_2'$.
   (b) If $a_1 \mid c_1 = a_2 \mid c_2$ then either $a_1 = a_2 \mid a_1'$ and $c_2 = a_1' \mid c_1$ or $c_1 = b_1 \mid c_2$ and $a_2 = a_1 \mid b_1$.
   (c) If $c_1 \mid a_1 = c_2 \mid a_2$ then either $a_1 = a_1' \mid a_2$ and $c_2 = c_1 \mid a_1'$ or $c_1 = c_2 \mid b_1$ and $a_2 = b_1 \mid a_1$.
   (d) If $a_1 \mid c_1 = c_2 \mid a_2$ then $c_1 = c_1' \mid a_2$ and $c_2 = a_1 \mid c_1'$.
   (e) If $c_1 \mid a_1 = a_2 \mid c_2$ then $c_1 = a_2 \mid c_1'$ and $c_2 = c_1' \mid a_1$.

We define the application function $ap : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$ by:

$$ap(\_, a) = a \tag{2.22}$$
$$ap(\kappa[c], a) = \kappa[ap(c, a)] \tag{2.23}$$
$$ap(a_1 \mid c, a) = a_1 \mid ap(c, a) \tag{2.24}$$
$$ap(c \mid a_1, a) = ap(c, a) \mid a_1 \tag{2.25}$$

Notice that, by our definitions, $(\mathcal{C}, \mathcal{D}, \{\_\}, \{0\})$ is a model for Context Logic with zero.

When working with the tree model, we use an extended definition of Context Logic, incorporating connectives that are related to the structural connectives in the tree and context grammars.

Definition 2.6. The Context Logic for Trees consists of *data formulae* and *context formulae* constructed from the signature set Σ. They are defined as for Context Logic with Zero, except with the addition of the following additional tree-specific context formulae:

$$\text{specific context formulae} \qquad u[K] \;\Big|\; P \mid K \;\Big|\; K \mid P \qquad u \in \Sigma. \tag{2.26}$$

Definition 2.7. The satisfaction relations for the Context Logic for Trees are defined inductively using the same rules as Context Logic with Zero, taking $\mathbf{I} = \{\_\}$ and $\mathbf{0} = \{0\}$, with the following additions:

$$c \models_K u[K] \text{ iff } \exists c' \in \mathcal{C}. c = u[c'] \text{ and } c' \models_K K \tag{2.27}$$
$$c \models_K P \mid K \text{ iff } \exists a \in \mathcal{D}, c' \in \mathcal{C}. c = a \mid c' \text{ and } a \models_D P \text{ and } c' \models_K K \tag{2.28}$$
$$c \models_K K \mid P \text{ iff } \exists a \in \mathcal{D}, c' \in \mathcal{C}. c = c' \mid a \text{ and } a \models_D P \text{ and } c' \models_K K \tag{2.29}$$
$$\tag{2.30}$$

The reader might wonder why we did not also define data formulae corresponding to the grammar connectives for trees. The reason for this is that such connectives can be defined in a straightforward manner based on the fragment we have defined.

$$P_1 \mid P_2 \triangleq (P_1 \mid \_)(P_2) \tag{2.31}$$
$$u[P] \triangleq (u[P \mid \_])(0) \tag{2.32}$$

2.2 Additional Notes

It is worth noting that Context Logic is equipped with a sound and complete Hilbert-style proof theory. However, since we do not make use of it here, we do not present it. It has very recently been shown that Context Logic can presented as standard Modal Logic[1] with a set of Sahlqvist axioms. Because of this, the soundness and completeness of the proof theory follow from a general result for Modal Logic.

3 Adjunct Elimination

For Context Logic, we would say adjunct elimination holds if the expressive power of the logic is the same whether or not it contains adjunct connectives. Specifically, each formula which makes use of adjunct connectives is logically equivalent to some formula which does not make use of those connectives in the following sense:

$$P_1 \equiv P_2 \iff \{a \in \mathcal{D} : a \models P_1\} = \{a \in \mathcal{D} : a \models P_2\} \tag{3.1}$$
$$K_1 \equiv K_2 \iff \{c \in \mathcal{C} : c \models K_1\} = \{c \in \mathcal{C} : c \models K_2\} \tag{3.2}$$
$$\tag{3.3}$$

In this section, we shall present some simple results concerning adjunct elimination. In particular, we shall see examples of models for context logic that do not admit adjunct elimination, and therefore demonstrate explicitly that such a property is not inherent in the logic, and rather dependent on the particular model. Also, we present a simple logic in which adjunct elimination holds, yet does not submit trivially to proof of this fact through games.

3.1 Models that do not admit adjunct elimination

We now present some models for which adjunct elimination is not possible, to demonstrate, for instance, that adjunct elimination is not a general property of Context Logic (and similar logics).

Consider the sets

$$\begin{aligned} \mathcal{C} &= \{1, c\} \\ \mathcal{D} &= \{a, b, d, h\} \end{aligned}$$

with application defined as in table 1.

ap	1	c
a	a	a
b	b	a
d	d	a
h	h	b

Table 1: Application relation for example model.

We assert that, in context logic without any model-specific formulae, adjunct elimination is not possible for this model. In order to do this, we shall consider truth sets for formulae, defined as

$$[\![P]\!] \triangleq \{a \in \mathcal{D} \mid a \models P\} \tag{3.4}$$
$$[\![K]\!] \triangleq \{c \in \mathcal{C} \mid c \models K\}\,. \tag{3.5}$$

Based on the satisfaction rules of the logic, we can derive equations for these sets in terms of the logical connectives, which will allow us to consider which sets of contexts and data structures correspond to truth sets of formulae.

For context formulae:

$$[\![True]\!] = \{1, c\} \tag{3.6}$$
$$[\![K_1 \wedge K_2]\!] = [\![K_1]\!] \cap [\![K_2]\!] \tag{3.7}$$
$$[\![K_1 \vee K_2]\!] = [\![K_1]\!] \cup [\![K_2]\!] \tag{3.8}$$
$$[\![\neg K]\!] = C \setminus [\![K]\!] \tag{3.9}$$
$$[\![I]\!] = \{1\} \tag{3.10}$$
$$[\![P_1 \rhd P_2]\!] = \{k \in C \mid \forall p \in [\![P_1]\!].k(p) \in [\![P_2]\!]\} \tag{3.11}$$

For data formulae:

$$[\![true]\!] = \{a, b, d, h\} \tag{3.12}$$
$$[\![P_1 \wedge P_2]\!] = [\![P_1]\!] \cap [\![P_2]\!] \tag{3.13}$$
$$[\![P_1 \vee P_2]\!] = [\![P_1]\!] \cup [\![P_2]\!] \tag{3.14}$$
$$[\![\neg P]\!] = \mathcal{D} \setminus [\![P]\!] \tag{3.15}$$
$$[\![K(P)]\!] = \{k(p) \mid k \in [\![K]\!],\ p \in [\![P]\!]\} \tag{3.16}$$
$$[\![K \lhd P]\!] = \{p \in \mathcal{D} \mid \forall k \in [\![K]\!].k(p) \in [\![P]\!]\} \tag{3.17}$$

To consider the statements expressible in a subset of the logic that includes the Boolean operations, it is sufficient to consider partitions of the sets C and $\mathcal{D}$ such that each set in each partition is the truth set of some formula. It is clear, by (3.8) and (3.14), that any union of these is expressible as a disjunction of the formulae. Suppose we have such a partition, $\mathcal{P}$, then we can construct another partition, $\mathcal{P}[A]$ for formula A, such that:

1. each subset in the partition $\mathcal{P}$ can be expressed as a union of subsets in the partition $\mathcal{P}[A]$, and so any formula whose truth set was a union of subsets in $\mathcal{P}$ is also a union of subsets in $\mathcal{P}[A]$;

2. the formula A is expressible as a union of subsets in $\mathcal{P}[A]$;

3. each $S \in \mathcal{P}[A]$ is the truth set of a Boolean combination of the formula A and some formula whose truth set is in $\mathcal{P}$.

$\mathcal{P}[A]$ may be defined as

$$\mathcal{P}[A] \triangleq \{[\![A]\!] \cap S,\ [\![A]\!]' \cap S \mid S \in \mathcal{P}\} \setminus \{\emptyset\}\,. \tag{3.18}$$

The properties we require are easily checked, with the last following from (3.7), (3.9), (3.13) and (3.15).

Clearly, for contexts we can construct the partition $\{\{1\}, \{c\}\}$ in any subset of the logic which includes I. Since this partition is into singleton sets, we need not consider any further refinement.

We shall show that for data formulae without adjuncts, the finest partition we can construct is $\{\{a\}, \{b\}, \{d, h\}\}$. This may be constructed as follows:

$$\mathcal{P}_0 = \{\{a, b, d, h\}\} \tag{3.19}$$
$$\mathcal{P}_1 = (\mathcal{P}_0)[(\neg I)(true)] = \{\{a, b\}, \{d, h\}\} \tag{3.20}$$
$$\mathcal{P}_2 = (\mathcal{P}_1)[(\neg I)((\neg I)(true))] = \{\{a\}, \{b\}, \{d, h\}\} \tag{3.21}$$

If it is possible to refine this partition any further, we would have to be able to find some formula with truth set containing d but not h (or vice-versa). In fact, we only require to show that such a formula can be constructed by applying a connective to subformulae whose truth sets are themselves unions of sets in the partition. It is helpful here to use the equations we have specified that give the truth sets of formulae in terms of the truth sets of subformulae.

Clearly, applying Boolean connectives will not construct any new sets, and so we need only consider terms of the form $K(P)$. Notice that $(K_1 \vee K_2)(P)$ is equivalent to $K_1(P) \vee K_2(P)$ and $K(P_1 \vee P_2)$ is equivalent to $K(P_1) \vee K(P_2)$. Thus we only have to consider terms $K(P)$ where $[\![K]\!] \in \{\{1\}, \{c\}\}$, $[\![P]\!] \in P_2$. But these are just $\{a\}$, $\{d, h\}$ and $\{a, b\}$, as given in table 2, so we are done.

$[\![K(P)]\!]$	$\{1\}$	$\{c\}$
$\{a\}$	$\{a\}$	$\{a\}$
$\{b\}$	$\{b\}$	$\{a\}$
$\{d, h\}$	$\{d, h\}$	$\{a, b\}$

Table 2: Truth sets of terms of the form $K(P)$.

Suppose, as we may reasonably do, that D is some formula such that $[\![D]\!] = \{b\}$. The formula $(\neg I) \lhd D$ is verifiably only satisfied by h. Thus the adjunct $\lhd$ may not be eliminated.

We now present another example of a context logic model, which demonstrates that even symmetry in finite models is not sufficient for adjunct elimination to hold. This time $\mathcal{C} = \mathcal{D} = \{a, b, c, d, e\}$, and the application relation is given by table 3.

ap	a	b	c	d	e
a	a	b	c	d	e
b	b	b	b	b	b
c	c	b	b	b	b
d	d	b	b	b	b
e	e	b	b	b	c

Table 3: Application relation for symmetric example model.

Here, we can partition $\mathcal{D}$ as $\{\{a\}, \{b\}, \{c\}, \{d, e\}\}$ without adjuncts, but as $\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$ with. Also, the $\lhd$ adjunct cannot be eliminated. Clearly, $\{\{a\}, \{b, c, d, e\}\}$ is the finest partition of $\mathcal{C}$ without $\lhd$, but, even without $\rhd$, we can improve this to $\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$. (Consequently, we could eliminate $\lhd$ if we kept $\rhd$ in this example.)

Now, let us consider extending Context Logic with constants for each element of the sets $\mathcal{C}$ and $\mathcal{D}$, so we have the additional formulae in the logic with satisfaction (for $c, c' \in \mathcal{C}$, $d, d' \in \mathcal{D}$):

$$c' \models \mathbf{k}_c \iff c = c' \tag{3.22}$$

$$d' \models \mathbf{p}_d \iff d = d' \tag{3.23}$$

Obviously, for any finite model adjunct elimination holds with these additional constructs. Thus we shall consider infinite models. In particular, the model $C = \mathcal{D} = \{e, f, a_i, b_i \mid i \in \mathbb{N}\}$, with application defined as in table 4 does not admit adjunct elimination. Specifically, $\mathbf{k}_f \lhd \mathbf{p}_e$ is not expressible without adjuncts.

ap	e	f	a_j	b_j
e	e	f	a_j	b_j
f	f	f	f	e
a_i	a_i	f	f	f
b_i	b_i	e	f	f

Table 4: Application relation for infinite example model.

3.2 Adjunct elimination where games are not suitable

We now present a logic for which adjunct elimination can easily be shown to hold through the use of truth sets, but for which a naïve application of the games-based proof is not applicable.

The logic that we shall consider is over models in the natural numbers, $\mathbb{N}$, and we shall denote it as $\mathcal{L}_{+,3}$

Definition 3.1. We define the set of formulae, $\mathcal{A}_{+,3}$ by the following grammar:

$$A, B ::= \mathbf{0} \mid A + B \mid A \wedge B \mid \neg A \mid \top \mid A \rhd B \mid \text{div3} \tag{3.24}$$

Definition 3.2. The satisfaction relation $N \models A$ is defined as follows:

$$n \models \mathbf{0} \iff n = 0 \tag{3.25}$$
$$n \models A + B \iff \exists a, b \in \mathbb{N} \text{ s.t. } a \models A,\ b \models B, \text{ and } n = a + b \tag{3.26}$$
$$n \models A \wedge B \iff n \models A \text{ and } n \models B \tag{3.27}$$
$$n \models \neg A \iff n \not\models A \tag{3.28}$$
$$n \models \top \tag{3.29}$$
$$n \models A \rhd B \iff \forall a, \text{ if } a \models A \text{ then } n + a \models B \tag{3.30}$$
$$n \models \text{div3} \iff n \in \{3m \mid m \in \mathbb{N}\} \tag{3.31}$$

We also define derived formulae $A \vee B \triangleq \neg(\neg A \wedge \neg B)$, $A \blacktriangleright B \triangleq \neg(A \rhd \neg B)$ and $\mathbf{1} \triangleq \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} + \neg\mathbf{0})$. The first is obviously satisfied if and only if A or B is satisfied (by de Morgan's laws). Satisfaction for the second is given by:

$$n \models A \blacktriangleright B \iff n \not\models A \rhd \neg B \tag{3.32}$$
$$\iff \text{it is not the case that } \forall a. \text{ if } a \models A \text{ then } n + a \not\models B \tag{3.33}$$
$$\iff \exists a \text{ s.t. it is not the case that if } a \models A \text{ then } n + a \not\models B \tag{3.34}$$
$$\iff \exists a \text{ s.t } a \models A \text{ and } n + a \models B \tag{3.35}$$

The third is easily checked to be satisfied only by 1.

We shall show adjunct elimination by considering the truth sets of formulae defined by

$$[\![A]\!] \triangleq \{n \in \mathbb{N} \mid n \models A\}. \tag{3.36}$$

This will be done in two steps: first, we shall show that all of the truth sets of formulae are of a particular form; we then shall show that all sets of that form are the truth sets of formulae without adjuncts.

To facilitate the proof, we define

$$t(l,n) = \{l + 3m \mid 0 \leq m \leq n\}, \tag{3.37}$$

and permit $n = \infty$. For $S, T \subseteq \mathbb{N}$, we define

$$S + T = \{s + t \mid s \in S,\ t \in T\}. \tag{3.38}$$

If $S = \bigcup_{i=0}^{k} S_i$ and $T = \bigcup_{j=0}^{l}$, we have

$$\begin{aligned}
a \in S + T &\iff a = s + t, \text{ some } s \in S,\ t \in T && (3.39)\\
&\iff a = s + t,\ s \in S_i,\ t \in T_j,\ 0 \leq i \leq k,\ 0 \leq, j \leq l && (3.40)\\
&\iff a \in S_i + T_j && (3.41)\\
&\iff a \in \bigcup_{i=0}^{k} \bigcup_{j=0}^{l} \left(S_i + T_j\right). && (3.42)
\end{aligned}$$

Also $m \in \mathbb{N} \mid l \leq m \leq u$ can obviously be written as a union of sets of the form $t(l', n')$. A further observation is that

$$\begin{aligned}
t(l,n) + t(l',n') &= \{(l + l') + 3(m + m') \mid 0 \leq m \leq n,\ 0 \leq m' \leq n'\} && (3.43)\\
&= \{(l + l') + 3m \mid 0 \leq m \leq (n + n')\} && (3.44)\\
&= t(l + l', n + n')). && (3.45)
\end{aligned}$$

Finally, supposing $l \leq l'$, if $l' = l + 3k$ for some k, then $t(l,n) \cap t(l',n') = t(l', \min(n - k, n'))$, otherwise $t(l,n) \cap t(l',n') = \emptyset$. We can extend this naturally to a finite intersection of such sets.

Proposition 3.1. *For every $A \in \mathcal{A}$, $[\![A]\!]$ can be written as a finite union of sets of the form $t(l,n)$.*

Proof. The proof is by induction on the structure of formulae. We use the alternative operators $A \vee B$ and $A \blacktriangleright B$ instead of $A \wedge B$ and $A \triangleright B$, since the latter are equivalent to $\neg(\neg A \vee \neg B)$ and $\neg(A \blacktriangleright \neg B)$ respectively, and since we consider negation.

Base cases:

$A = \mathbf{0}$:

$[\![\mathbf{0}]\!] = 0 = t(0,0)$.

$A = \top$:

$[\![\top]\!] = \mathbb{N} = t(0,\infty) \cup t(1,\infty) \cup t(2,\infty)$.

Inductive cases:

$A = B + C$:

Clearly

$$[\![B + C]\!] = [\![B]\!] + [\![C]\!]. \tag{3.46}$$

By the inductive hypothesis,

$$[\![B]\!] = \bigcup_{i=0}^{k} t(l_i, n_i) \tag{3.47}$$

$$[\![C]\!] = \bigcup_{j=0}^{k'} t(l'_i, n'_i). \tag{3.48}$$

Now

$$A = \bigcup_{i=0}^{k} \bigcup_{j=0}^{l} \left(t(l_i, n_i) + t(l'_j, n'_j) \right) \tag{3.49}$$

$$= \bigcup_{i=0}^{k} \bigcup_{j=0}^{l} \left(t(l_i + l'_j, n_i + n'_j) \right), \tag{3.50}$$

as required.

$A = B \vee C$:

$a \in [\![B \vee C]\!]$ if and only if $a \in [\![B]\!]$ or $a \in [\![C]\!]$. That is, if and only if $a \in [\![B]\!] \cup [\![C]\!]$, which, by the inductive hypothesis, is a finite union of sets of the form $t(l, n)$ as required.

$A = \neg B$:

$$[\![\neg B]\!] = [\![B]\!]' \tag{3.51}$$

$$= \bigcap_{i=0}^{k} t(l_i, n_i)'. \tag{3.52}$$

But

$$t(l_i, n_i)' = \mathbb{N} \setminus t(l_i, n_i) \tag{3.53}$$

$$= \{m \mid 0 \leq m < l_i\} \cup t(l_i + 1, n_i) \cup t(l_i + 2, n_i) \cup \{m \mid l_i + 3n_i < m\}, \tag{3.54}$$

which is a finite union of sets of the desired form. By applying distributivity of $\cap$ over $\cup$, we get that $[\![A]\!]$ is a finite union of finite intersections of such sets, which is, by a previous observation, just a finite union of sets of the required form.

$A = \top$:

$[\![\top]\!] = \mathbb{N} = t(0, \infty) \cup t(1, \infty) \cup t(2, \infty)$.

$A = B \blacktriangleright C$:

$$[\![B \blacktriangleright C]\!] = \{c - b \mid b \in [\![B]\!],\ c \in [\![C]\!],\ c \geq b\}. \tag{3.55}$$

Since $[\![B]\!] = \bigcup_{i=0}^{k} t(l_i, n_i)$ and $[\![C]\!] = \bigcup_{j=0}^{k'} t(l'_j, n'_j)$,

$$[\![A]\!] = \bigcup_{i=0}^{k} \bigcup_{j=0}^{k'} \left\{ c - b \mid b \in t(l_i, n_i),\ c \in t(l'_j, n'_j),\ b \leq c \right\}. \tag{3.56}$$

Each set in this union is of the required form, since $c - b = l_i - l'_j + 3(m - m')$.

$A = \mathrm{div3}$:

$[\![\mathrm{div3}]\!] = \{0 + 3m \mid 0 \leq m\} = t(0, \infty)$.

□

For the next proposition, we shall use the symbol $\mathbf{F}_n$, defined recursively by:

$\mathbf{F}_0 = 0$
$\mathbf{F}_n = \mathbf{1} + \mathbf{F}_{n-1}$ with $\mathbf{1}$ as defined previously.

It is easily seen that $m \models \mathbf{F}_n \iff m = n$.

Proposition 3.2. *Every finite union of sets of the form $t(l,n)$ is $[\![A]\!]$ for some formula A which does not use $\triangleright$.*

Proof. First, let us consider the set $t(l,n)$. It is clear that $t(l,n) = \{m \mid l \leq m\} \cap \{m \mid m \leq l + 3n\} \cap \{3m + r \mid m \in \mathbb{N}\}$ for some $r \in \{0,1,2\}$, such that $l \equiv r(\mod 3)$ (the choice of r for a given l is unique). Now consider the formula

$$T_{l,n} \triangleq (\mathbf{F}_l + \top) \wedge \neg(\mathbf{F}_{l+3n+1} + \top) \wedge (\mathrm{div}3 + \mathbf{F}_r) \tag{3.57}$$

Then

$$[\![T_{l,n}]\!] = \{l + m \mid m \in \mathbb{N}\} \cap (\mathbb{N} \setminus \{n + m \mid m \in \mathbb{N}\}) \cap \{m + r \mid m \in \{3k \mid k \in \mathbb{N}\}\} \tag{3.58}$$

$$= \{m \mid l \leq m\} \cap \{m \mid m \leq l + 3n\} \cap \{3m + r \mid m \in \mathbb{N}\} \tag{3.59}$$

$$= t(l,n) \tag{3.60}$$

as required. This works for finite n, but for $t(l,\infty)$ we have a similar formula

$$[\![T_{l,\infty}]\!] = \{l + m \mid m \in \mathbb{N}\} \cap \{m + r \mid m \in \{3k \mid k \in \mathbb{N}\}\} \tag{3.61}$$

$$= \{m \mid l \leq m\} \cap \{3m + r \mid m \in \mathbb{N}\} \tag{3.62}$$

$$= t(l,\infty) \tag{3.63}$$

Now for a finite union of such sets

$$t(l_1,n_1) \cup \ldots \cup t(l_k,n_k) = [\![T_{l_1,n_1} \vee \ldots \vee T_{l_k,n_k}]\!] \tag{3.64}$$

as required. □

The reason this logic doesn't yield simply to the game proof technique is that two pairs of numbers, each of which cannot be discriminated between by a formula of rank r, say, may be added to give a pair which may be discriminated between. For instance,

$$1 \models \neg\mathrm{div}3 \qquad 2 \models \neg\mathrm{div}3 \tag{3.65}$$

$$1 \models \neg\mathrm{div}3 \qquad 1 \models \neg\mathrm{div}3 \tag{3.66}$$

but

$$1 + 1 \models \neg\mathrm{div}3 \qquad 2 + 1 \not\models \neg\mathrm{div}3. \tag{3.67}$$

This means that we would not be able to prove the key theorem which would permit adjunct elimination:

Non-theorem 3.3. *For all ranks, r, and $n, n', m, m' \in \mathbb{N}$, if*

$$(n,n',r) \in DW \qquad \textit{and} \tag{3.68}$$

$$(m,m',r) \in DW \tag{3.69}$$

then

$$(n + m, n' + m', r) \in DW. \tag{3.70}$$

Although the notation used here has not yet been defined, it should become clear in later sections what is meant. In short, such a theorem means that adding the $\triangleright$ connective to formulae in finite sets (the logically distinct formulae of rank r) does not contribute to refining the partition of $\mathbb{N}$ that is generated. It says that, as long as there are two pairs of numbers, each of which are 'similar enough' (so they cannot be distinguished between by a certain set of formulae), point-wise addition of the pairs produces a pair which are just as similar. The above gives a counter-example to this, since we have two pairs that cannot be discriminated between using div3, but whose point-wise sum produces a pair which can be discriminated between with div3.

3.3 Evaluation

In subsection 3.1, we studied several simple models of context logic to demonstrate that elimination of the adjuncts in Context Logic is not a property of the logic itself, but depends very much on the model chosen. The examples given, however, fall far short of providing any general criteria for adjunct elimination properties to hold. Attempting to find some simple conditions which prevent adjunct elimination for finite models may be quite possible, particularly by considering the tricks used to define the models we have given.

Finding conditions that are exactly equivalent to adjunct elimination may also be possible, but would probably be more difficult. In fact, any such formulation would likely consist of a model condition that is just as difficult to demonstrate in any particular case as a direct proof of adjunct elimination. Nonetheless, finding such model conditions, both for models of basic Context Logic and also Context Logic with model-specific connectives, would present a different means to assess the question of adjunct elimination.

In subsection 3.2, we saw an example of a logic for which games cannot be applied naïvely, but for which adjunct elimination holds. This raises the question as to how useful the games-based proof technique might be.

The example we gave, however, is quite contrived, specifically in order to prevent direct application of the proof technique. Additionally, it is very likely that the proof can be adapted to cope with the problem. This is because the key theorem in adjunct elimination using games, non-theorem 3.3, is stronger than is strictly necessary in order to prove that the adjunct may be eliminated. A weaker result, and one that may well be true is the following conjecture:

Conjecture 3.1. For all ranks, r, and $n, n', m \in \mathbb{N}$, if

$$(n, n', r) \in DW \qquad \text{and} \tag{3.71}$$

$$(m, m', r) \in DW \tag{3.72}$$

then $\exists m' \in \mathbb{N}$ such that

$$(n + m, n' + m', r) \in DW. \tag{3.73}$$

This result should still be sufficient. Indeed, we shall see in section 5 that proving the stronger result is not possible in Context Logic for trees, and that we have to consider a weaker result instead.

It might be possible to consider a result concerning games that is equivalent to adjunct elimination. Explicitly determining this result would likely enhance

the usefulness of games in determining whether adjunct elimination holds: showing that such a result does not hold would then be sufficient to show that adjunct elimination also does not hold, even if it is difficult to construct a direct counterexample.

4 Games

In this section, we shall define Ehrenfeucht-Fraïssé games for Context Logic for Trees, as well as providing important results on soundness and completeness. Although we present several important results here, they are not novel and are merely reworkings of standard results, well known in the literature. Indeed, they are simple adaptations of those given for Ambient Logic in [4].

4.1 Preliminaries

We present definitions that will be required for Ehrenfeucht-Fraïssé games for Context Logic for Trees.

The following definition establishes the concept of a rank for a formula. The definition we will use is tailored particularly for Theorem 5.2.

Definition 4.1. For every formula, A, we define the *rank* of A,

$$|A| = r = (n, s, \mathcal{L}), \tag{4.1}$$

as a tuple with $n, s \in \mathbb{N}$, $\mathcal{L} \subseteq \Sigma$ such that

- n is the maximum nesting depth of all connectives, except for Boolean connectives and '$\triangleleft$' (that is, the connectives $\mathbf{0}$, $K(P)$, $\mathbf{I}$, $u[K]$, $K \mid P$, $P \mid K$ and $P \triangleright P$);
- s is the maximum nesting depth of the '$\triangleleft$' connective;
- and $\mathcal{L}$ is the finite set of labels used.

To help convey the notion expressed in this definition, we present some examples of ranks for formulae.

$$|\mathbf{0}| = (1, 0, \emptyset) \tag{4.2}$$
$$|(\mathit{false} \triangleright \mathit{false}) \wedge \neg \mathit{False})| = (1, 0, \emptyset) \tag{4.3}$$
$$|\iota_1[\mathit{True}](\mathbf{0})| = (2, 0, \{\iota_1\}) \tag{4.4}$$
$$|(\mathbf{0} \triangleright (\neg((\mathit{False} \triangleleft \mathit{false}) \triangleright (\mathbf{I} \triangleleft \neg \mathit{false}))) \triangleleft \mathit{false}))| = (3, 2, \emptyset) \tag{4.5}$$

For various purposes, it is useful to have a partial ordering relation on ranks.

Definition 4.2. We say $(n_1, s_1, \mathcal{L}_1) \leq (n_2, s_2, \mathcal{L}_2)$ iff $n_1 \leq n_2$, $s_1 \leq s_2$ and $\mathcal{L}_1 \subseteq \mathcal{L}_2$.

Definition 4.3. We say that a tree, a_1, is discriminated from a_2 by a tree-formula P iff $a_1 \models P$ and $a_2 \not\models P$ (or vice-versa).

Similarly, we say that a context, c_1, is discriminated from c_2 by context-formula K iff $c_1 \models K$ and $c_2 \not\models K$ (or vice-versa).

Definition 4.4. Let $\mathcal{T}$ be a set of trees. We say that a_1 and a_2 are $\mathcal{T}$-discriminated iff $a_1 \in \mathcal{T}$ and $a_2 \notin \mathcal{T}$.

Let $\mathcal{K}$ be a set of contexts. We say that c_1 and c_2 are $\mathcal{K}$-discriminated iff $c_1 \in \mathcal{K}$ and $c_2 \notin \mathcal{K}$.

Lemma 4.1. *For each finite rank r, there are finitely many inequivalent formulae of rank r.*

Proof. By induction on the rank r, and cases on the outermost operator of the formula.

If the outermost operator is $\mathbf{0}$, false, $\mathbf{I}$ or False, the result is immediate. If it is $u[K]$, $K \mid P$, $P \mid K$, $P \rhd P$, $K(P)$ or $K \lhd P$, the subformulae have strictly smaller rank, and therefore there are finitely many equivalence classes to choose from. Also, in the $u[K]$ case, there are finitely many choices of label, as it must be in $\mathcal{L}$. Thus, there are only finitely many inequivalent formulae with these operators at their outermost.

All other formulae must be Boolean combinations of finitely many formulae of rank r whose outermost operators are not Boolean. Upto equivalence, there are only finitely many such Boolean combinations. □

This lemma allows us to make the following definitions:

Definition 4.5. For each finite rank r, let $\mathscr{P}_r$ be the finite set of data-formulae such that every formula of rank r is equivalent to a formula in $\mathscr{P}_r$.

For each finite rank r, let $\mathscr{K}_r$ be the finite set of context-formulae such that every formula of rank r is equivalent to a formula in $\mathscr{K}_r$.

Definition 4.6. The characteristic formula of a tree a with respect to rank r denoted D^r_a is defined as

$$D^r_a = \bigwedge \{P \in \mathscr{P}_r : a \models P\}.$$

D^r_a has rank r by construction.

The characteristic formula of a context c with respect to rank r denoted D^r_c is defined as

$$D^r_c = \bigwedge \{K \in \mathscr{K}_r : c \models K\}.$$

D^r_c has rank r by construction.

Lemma 4.2. *For all trees, a_1, a_2, contexts, c_1, c_2, and ranks, r:*

$$(\forall P \in \mathscr{P}_r.\ a_1 \models P \Rightarrow a_2 \models P) \iff a_2 \models D^r_{a_1} \tag{4.6}$$
$$(\forall K \in \mathscr{K}_r.\ c_1 \models K \Rightarrow c_2 \models K) \iff c_2 \models D^r_{c_1} \tag{4.7}$$
$$(\exists P \in \mathscr{P}_r.\ a_1 \models P \wedge a_2 \not\models P) \iff a_2 \not\models D^r_{a_1} \tag{4.8}$$
$$(\exists K \in \mathscr{K}_r.\ c_1 \models K \wedge c_2 \not\models K) \iff c_2 \not\models D^r_{c_1} \tag{4.9}$$
$$(\forall P \in \mathscr{P}_r.\ a_1 \models P \Leftrightarrow a_2 \models P) \iff a_2 \models D^r_{a_1} \tag{4.10}$$
$$(\forall K \in \mathscr{K}_r.\ c_1 \models K \Leftrightarrow c_2 \models K) \iff c_2 \models D^r_{c_1} \tag{4.11}$$
$$a_2 \models D^r_{a_1} \iff a_1 \models D^r_{a_2} \tag{4.12}$$
$$c_2 \models D^r_{c_1} \iff c_1 \models D^r_{c_2} \tag{4.13}$$

Proof. For (4.6), by definition:

$$a_2 \models D^r_{a_1} \Leftrightarrow a_2 \models \Big(\bigwedge \{P : P \in \mathscr{P}_r . a_1 \models P\}\Big) \tag{4.14}$$
$$\Leftrightarrow \forall P \in \mathscr{P}_r . a_1 \models P \Rightarrow a_2 \models P. \tag{4.15}$$

Part (4.8) is immediate from (4.6).

For (4.10), the $\Rightarrow$ case is immediate from (4.6). For the $\Leftarrow$ case, assume $a_2 \models D^r_{a_1}$ and $a_1 \models P$. Hence, $a_2 \not\models \neg P$. By (4.6), $a_1 \not\models \neg P$, and so $a_1 \models P$.

For (4.12), assume that $a_2 \models D^r_{a_1}$. Then, by (4.10), $\forall P \in \mathscr{P}_r . a_2 \models P \implies a_1 \models P$. Thus, by (4.6), $a_1 \models D^r_{a_2}$. (The implication in the other direction is the same.)

Proofs of (4.7), (4.9), (4.11) and (4.13) are just the same as the above.

□

Lemma 4.3. *Let $\mathcal{T}$ be a set of trees such that, for any $\mathcal{T}$-discriminated pair (a_1, a_2), there exists a formula P_{a_1,a_2} of rank r that discriminates a_1 from a_2. Then $\mathcal{T}$ can be defined by a rank-r formula P.*

Let $\mathcal{K}$ be a set of trees such that, for any $\mathcal{K}$-discriminated pair (c_1, c_2), there exists a formula K_{c_1,c_2} of rank r that discriminates c_1 from c_2. Then $\mathbf{C}$ can be defined by a rank-r formula K.

Proof. Consider

$$\mathscr{P}_{\mathcal{T}} = \left\{ P \in \mathscr{P}_r : \exists a_2 \in \mathcal{T} . P \Leftrightarrow D^r_{a_2} \right\}.$$

We see $\mathscr{P}_{\mathcal{T}} \subseteq \mathscr{P}_r$ so it is finite. Hence, $Q = \bigvee \mathscr{P}_{\mathcal{T}}$ is a formula of rank r. We shall show that $a \models Q$ if and only if $a \in \mathcal{T}$.

Suppose $a \in \mathcal{T}$. By the definition of $\mathscr{P}_r$, there exists $P' \in \mathscr{P}_r$ which is equivalent to D^r_a. Hence, $a \models P'$ and $P' \in \mathscr{P}_{\mathcal{T}}$, and thus $a \models Q$.

Now suppose $a \models Q$. Then $a \models P'$ for some $P' \in \mathscr{P}_r$. P' is equivalent to $D^r_{a_2}$ for some $a_2 \in \mathcal{T}$. Hence, $a \models D^r_{a_2}$. If $a \notin \mathcal{T}$, then there exists a rank-r formula that discriminates between a_1 and a_2. But $a \models D^r_{a_2}$, so this is not possible. Hence, $a \in \mathcal{T}$.

The proof for contexts is essentially the same. □

4.2 Games

We now define the Ehrenfeucht-Fraïssé games that we shall use to prove adjunct elimination. The game is either played on a pair of trees or a pair of contexts, and also has an associated rank, r. Two players participate in the game: Spoiler and Duplicator.

The state of the game at any time is given by a triple consisting of a pair of trees or contexts and a rank. Thus game states are either (a, a', r) or (c, c', r) for some $a, a' \in \mathcal{D}, c, c' \in \mathbf{C}$ and finite rank r. The initial state is just such a triple.

At each round of the game, Spoiler selects a move to play, provided the rank permits it and that the conditions of the move can be met. For some moves, Spoiler can win immediately. For others, the game continues with a different pair of trees or contexts and a smaller rank. Still others require Duplicator to respond to Spoiler's move, and the game continues with a new state of smaller rank, which Spoiler may have some further choice in determining.

Definition 4.7. The moves which can be played from a given game state depend on whether trees or contexts are currently in play. At the start of each move, Spoiler is allowed to select either of the trees a, a' (or contexts c, c'). Within this definition, we shall call whichever Spoiler chooses b (or d) and the other b' (or d'). Additionally, the availability of moves is dependent on the rank. Specifically, for the rank $(n, s, \mathcal{L})$, the $K \lhd P$ move may only be played when $s > 0$ and the other moves may only be played when $n > 0$.

For the game $(a, a', \mathcal{L})$, the moves are:

0 move Spoiler chooses b so that $b = 0$ and $b' \neq 0$, and wins.

$K(P)$ move	Spoiler chooses b and a context $c_1 \in C$ and tree $a_1 \in \mathcal{D}$ such that $b = c_1(a_1)$. Duplicator then chooses a context $c_1' \in C$ and tree $a_1' \in \mathcal{D}$ such that $b' = c_1'(a_1')$. Spoiler then decides whether the game will continue with $(c_1, c_1', (n-1, s, \mathcal{L}))$ or $(a_1, a_1', (n-1, s, \mathcal{L}))$.
$K \triangleleft P$ move	Spoiler chooses b and a context $c_1 \in C$. Duplicator then chooses a context $c_1' \in C$. Spoiler then decides whether the game will continue with $(c_1, c_1', (n, s-1, \mathcal{L}))$ or $(c_1(b), c_1'(b'), (n, s-1, \mathcal{L}))$.

For the game $(c, c', \mathcal{L})$, the moves are:

I move	Spoiler chooses d so that $d = \_$ and $d' \neq \_$, and wins.
$K \mid P$ move	Spoiler chooses d and a context $c_1 \in C$ and tree $a_1 \in \mathcal{D}$ such that $d = c_1 \mid a_1$. Duplicator then chooses a context $c_1' \in C$ and tree $a_1' \in C$ such that $d' = c_1' \mid a_1'$. Spoiler then decides whether the game will continue with $(c_1, c_1', (n-1, s, \mathcal{L}))$ or $(a_1, a_1', (n-1, s, \mathcal{L}))$.
$P \mid K$ move	Spoiler chooses d and a context $c_1 \in C$ and tree $a_1 \in \mathcal{D}$ such that $d = a_1 \mid c_1$. Duplicator then chooses a context $c_1' \in C$ and tree $a_1' \in C$ such that $d' = a_1' \mid c_1'$. Spoiler then decides whether the game will continue with $(c_1, c_1', (n-1, s, \mathcal{L}))$ or $(a_1, a_1', (n-1, s, \mathcal{L}))$.
$u[K]$ move	Spoiler chooses d and a label $\kappa \in \mathcal{L}$ such that $d = \kappa[c_1]$ for some $c_1 \in C$. If $d' = \kappa[c_1']$ for some $c_1' \in C$ then the game continues with $(c_1, c_1', (n-1, s, \mathcal{L}))$. Otherwise, Spoiler wins.
$P \triangleright P$ move	Spoiler chooses d and a tree $a_1 \in \mathcal{D}$. Duplicator then chooses a tree $a_1' \in \mathcal{D}$. Spoiler then decides whether the game will continue with $(a_1, a_1', (n-1, s, \mathcal{L}))$ or $(d(a_1), d'(a_1'), (n-1, s, \mathcal{L}))$.

Within the definition we have given for games, only the means by which Spoiler can win are explicitly stated. It is quite possible, however, for a game to enter such a state that Spoiler is unable to make a move. In this case, we say that Duplicator wins the game.

The outcome of individual games is not necessarily important. For instance, either player might make a poor series of moves, allowing the other player to win, where a victory could have easily be obtained by the player. Instead, we talk about winning strategies. One player has a winning strategy if he can win the game regardless of how the other player plays.

Definition 4.8. We define $SW_{\mathcal{D}}$ and SW_C to be the sets of games of the forms (a, a', r) and (c, c', r) respectively for which Spoiler has a winning strategy. Similarly, we define $DW_{\mathcal{D}}$ and DW_C to be the sets of games of the forms (a, a', r) and (c, c', r) respectively for which Duplicator has a winning strategy.

For any given game, either Spoiler or Duplicator must have a winning strategy. Suppose, for instance, that Spoiler does not have a winning strategy for some game. Then there must be some strategy that Duplicator may use, such that no matter how Spoiler plays, he does not win. That is, Duplicator has a winning strategy.

The following lemma, downward closure, is simple (it holds easily by the definition of the games) yet useful.

Lemma 4.4. *If Duplicator has a winning strategy for the game* (a, a', r) *(or* (c, c', r)*) then for any* $r' \leq r$*, Duplicator has a winning strategy for the game* (a, a', r') *(or* (c, c', r')*).*

Proof. In the game (a, a', r'), Spoiler may only play moves he could have played in the game (a, a', r). Since those moves did not give him a winning strategy, he cannot have one in the game (a, a', r'), and so Duplicator has a winning strategy. □

By a simple corollary, if Spoiler has a winning strategy for (a, a', r') then he has a winning strategy for (a, a', r) where $r \geq r'$.

4.3 Game soundness

We now state and prove that the games we have defined are sound with respect to the logic. The notion of soundness, informally stated, is that: if some formula of rank r discriminates between a pair of trees or contexts, then Spoiler has a winning strategy for the game of rank r on those trees or contexts.

Lemma 4.5. *If there exists a data-formula* P *of rank* r *such that* $a \models P$ *and* $a' \not\models P$*, then Spoiler has a winning strategy for the game* (a, a', r)*.*

If there exists a context-formula K *of rank* r *such that* $c \models K$ *and* $c' \not\models K$*, then Spoiler has a winning strategy for the game* (c, c', r)*.*

Proof. The proof is by induction on the structure of the formula, P or K. We look at the cases for the outermost operator. In each case, Spoiler plays the corresponding move in the game in order to win inductively.

$P = \mathbf{0}$:

This case is trivial: $n \geq 1$, so Spoiler can play the $\mathbf{0}$ move and win the game.

$P = K_1(P_1)$:

Spoiler plays the $K(P)$ move, choosing a. Since $a \models P$, Spoiler can choose a context $c_1 \in \mathcal{C}$ and tree $a_1 \in \mathcal{D}$ such that $a = c_1(a_1)$, $c_1 \models K_1$ and $a_1 \models P_1$. Duplicator chooses $c_1' \in \mathcal{C}$, $a_1' \in \mathcal{D}$ such that $a' = c_1'(a_1')$. Since we know that $a' \not\models P$, we must have either $c_1' \not\models K_1$ or $a_1' \not\models P_1$. We also know that K_1 and P_1 have rank at most $(n-1, s, \mathcal{L})$, and so in the former case, Spoiler has a winning strategy for the game $(c_1, c_1', (n-1, s, \mathcal{L}))$ by induction and downward closure. Similarly, in the latter case, Spoiler will have a winning strategy for the game $(a_1, a_1', (n-1, s, \mathcal{L}))$. In future cases, we shall consider the use of downward closure implicit when using induction to show that Spoiler has a winning strategy.

$P = K_1 \lhd P_1$:

Spoiler plays the $K \lhd P$ move, choosing a'. Spoiler chooses a context $c_1' \models K_1$ such that $c_1'(a') \not\models P_1$. (Such a c_1' must exist, or else we would have $a' \models P$.) Duplicator chooses a context c_1. If $c_1 \not\models K_1$, then Spoiler can choose to continue with the game $(c_1, c_1', (n, s-1, \mathcal{L}))$, for which he has a winning strategy by induction. Otherwise, $c_1 \models K_1$ and so $c_1(a) \models P_1$. Thus, by induction, Spoiler has a winning strategy by choosing to continue with the game $(c_1(a), c_1'(a'), (n, s-1, \mathcal{L}))$.

$K = \mathbf{I}$:

This case is trivial: $n \geq 1$, so Spoiler can play the $\mathbf{I}$ move and win the game.

$K = P_1 \rhd P_2$:

Spoiler plays the $P \rhd P$ move, choosing c'. Spoiler chooses a tree $a_1' \models P_1$

such that $c'(a'_1) \not\models P_2$. (Such a a'_1 must exist, or else we would have $c' \models K$.) Duplicator chooses a tree a_1. If $a_1 \not\models P_1$, then Spoiler can choose to continue with the game $(a_1, a'_1, (n-1, s, \mathcal{L}))$, for which he has a winning strategy by induction. Otherwise, $a_1 \models P_1$ and so $c(a_1) \models P_2$. Thus, by induction, Spoiler has a winning strategy by choosing to continue with the game $(c(a_1), c'(a'_1), (n-1, s, \mathcal{L}))$.

$K = P_1 \mid K_1$:
Spoiler plays $P \mid K$ move, choosing c. Spoiler can find $a_1 \in \mathcal{D}$, $c_1 \in C$ such that $c = a_1 \mid c_1$, $a_1 \models P_1$ and $c_1 \models K_1$. Duplicator chooses $a'_1 \in \mathcal{D}$, $c'_1 \in C$ such that $c' = a'_1 \mid c'_1$. Since $a' \not\models K$, either $a'_1 \not\models P_1$ or $c'_1 \not\models K_1$. In the former case, Spoiler has a winning strategy for $(a_1, a'_1, (n-1, s, \mathcal{L}))$ by induction, and so may choose to continue with that game to give a winning strategy. In the latter, Spoiler has a winning strategy for $(c_1, c'_1, (n-1, s, \mathcal{L}))$ by induction, and so will achieve a winning strategy by continuing with that game.

$K = K_1 \mid P_1$:
This case is symmetric to the $K = P_1 \mid K_1$ case.

$K = \kappa[K_1]$:
Spoiler plays the $u[K]$ move, choosing $c = \kappa[c_1]$. If $c' \neq \kappa[c'_1]$ then Spoiler wins. Otherwise, we know K_1 has a suitably smaller rank and so by induction Spoiler has a winning strategy when the game continues with $(c, c', (n-1, s, \mathcal{L}))$.

□

4.4 Game completeness

Here, we state and prove that the games we have defined are complete with respect to the logic. Informally, completeness means that: if Spoiler has a winning strategy for a game of rank r, the two trees or contexts are discriminated between by a formula of rank r.

Lemma 4.6. *If Spoiler has a winning strategy for the game (a, a', r) then there exists a formula, P, of rank at most r that discriminates a from a'.*

If Spoiler has a winning strategy for the game (c, c', r) then there exists a formula, K, of rank $\leq r$ that discriminates c from c'.

In the proof, where $r = (n, s, \mathcal{L})$ we shall use r^- to denote $(n, s-1, \mathcal{L})$ when we are discussing the $K \lhd P$ move and to denote $(n-1, s, \mathcal{L})$ otherwise.

Proof. We proceed by induction on the rank r and by cases on the first move Spoiler makes in his winning strategy for the game. At the start of each move, Spoiler may choose either a or a' (equivalently c or c'). We assume without loss of generality that he chooses a.

0 move:
Here, a and a' are distinguished by the formula **0**.

$K(P)$ move:
For his winning strategy, Spoiler chooses $c_1 \in C$, $a_1 \in \mathcal{D}$ such that $a = c_1(a_1)$. Now take P to be $D^{r^-}_{c_1}(D^{r^-}_{a_1})$, which has rank r. We know that $a \models P$. Suppose that $a' \models P$. Then there exist $c'_1 \in C$, $a'_1 \in \mathcal{D}$ such that $a' = c'_1(a'_1)$, $c'_1 \models D^{r^-}_{c_1}$ and $a'_1 \models D^{r^-}_{a_1}$. Thus c_1 and c'_1 cannot be discriminated by a formula of rank r^-, and similarly a_1 and a'_1 cannot be discriminated by a formula of rank r^- either. Hence, by the inductive hypothesis, Duplicator has a winning strategy for the games (c_1, c'_1, r^-) and (a_1, a'_1, r^-), which contradicts the assumption that Spoiler has a winning strategy for (a, a', r). Hence $a' \not\models P$ and we are done.

$K \lhd P$ move:
For his winning strategy, Spoiler chooses a and some new $c_1 \in C$. Now take P to be $D^{r^-}_{c_1} \blacktriangleleft D^{r^-}_{c_1(a)}$, which has rank $r^- = (n, s-1, \mathcal{L})$. Assume for a contradiction that $a' \models P$. Then there exists a c'_1 with $c'_1 \models D^{r^-}_{c_1}$ and $c'_1(a') \models D^{r^-}_{c_1(a)}$. Thus the pairs c_1, c'_1 and $c_1(a), c'_1(a')$ each cannot be discriminated by any formula of rank r^-. Hence, by the inductive hypothesis, Duplicator has a winning strategy for the games (c_1, c'_1, r^-) and $(c_1(a), c'_1(a'), r^-)$, contradicting the fact that Spoiler has a winning strategy. Hence $a' \not\models P$ and we are done.

I move:
In this case, selecting $K = \mathbf{I}$ discriminates between the contexts.

$P \rhd P$ move:
For his winning strategy, Spoiler chooses c and some new $a_1 \in \mathcal{D}$. Now take K to be $D^{r^-}_{a_1} \blacktriangleright D^{r^-}_{c(a_1)}$, which has rank $r^- = (n-1, s, \mathcal{L})$. Assume for a contradiction that $c' \models K$. Then there exists a a'_1 with $a'_1 \models D^{r^-}_{a_1}$ and $c'(a'_1) \models D^{r^-}_{c(a_1)}$. Thus the pairs a_1, a'_1 and $c(a_1), c'(a'_1)$ each cannot be discriminated by any formula of rank r^-. Hence, by the inductive hypothesis, Duplicator has a winning strategy for the games (a_1, a'_1, r^-) and $(c(a_1), c'(a'_1), r^-)$, contradicting the fact that Spoiler has a winning strategy. Hence $c' \not\models P$ and we are done.

$K \mid P$ move:
For his winning strategy, Spoiler chooses $c_1 \in C$, $a_1 \in \mathcal{D}$ such that $c = c_1 \mid a_1$. Now take K to be $D^{r^-}_{c_1} \mid D^{r^-}_{a_1}$, which has rank r. We know that $c \models K$. Suppose that $c' \models K$. Then there exist $c'_1 \in C$, $a'_1 \in \mathcal{D}$ such that $a' = c'_1 \mid a'_1$, $c'_1 \models D^{r^-}_{c_1}$ and $a'_1 \models D^{r^-}_{a_1}$. Thus c_1 and c'_1 cannot be discriminated by a formula of rank r^-, and similarly a_1 and a'_1 cannot be discriminated by a formula of rank r^- either. Hence, by the inductive hypothesis, Duplicator has a winning strategy for the games (c_1, c'_1, r^-) and (a_1, a'_1, r^-), which contradicts the assumption that Spoiler has a winning strategy for (a, a', r). Hence $a' \not\models P$ and we are done.

$P \mid K$ move:
This case is just symmetrical to the $K \mid P$ case.

$u[K]$ move:
Spoiler selects $\kappa \in \mathcal{L}$ such that $c = \kappa[c_1]$. If Spoiler wins because $c' \neq \kappa[c'_1]$ then we can choose $K = \kappa[True]$ and we are done. Otherwise, $c' = \kappa[c'_1]$ and Spoiler has a winning strategy for the game (c_1, c'_1, r^-). By induction, there is a formula K' such that $c_1 \models K'$ and $c'_1 \not\models K'$. Choose $K = \kappa[K']$, and we are done. □

4.5 Evaluation

In our definition of ranks, we separated out the $K \lhd P$ connective specifically. The reason for doing this is that we can tell exactly from the rank of a formula whether or not it contains the '$\lhd$' adjunct, which will be necessary in our reasoning about its elimination. This raises the question, "Why not separate the other connectives likewise?" That is what is done in the proof for Ambient Logic in [4], however, we choose not to distinguish the other connectives, since this permits more flexibility over choice of moves in games, which we shall put to use in proofs in section 5. We could, of course, have chosen to separate out the '$\rhd$' adjunct also, however, the presentation here is motivated by the proof of elimination of the '$\lhd$' adjunct and, since we shall see that '$\rhd$' cannot be eliminated in Context Logic for Trees, separating '$\rhd$' from the rest is not particularly useful.

Of course, it is certainly possible that the proof we shall present can be adapted to use a rank which distinguishes all connectives, which would imply that a formula without '$\lhd$' has an equivalent within a smaller set than the current proof asserts. This is not, however, likely to be of much significance, since determining exactly which formula in the finite set is equivalent is not decidable.

The reader may have observed that some of the definitions given above could have been expressed in terms of truth sets, which we have looked at previously in scrutinising adjunct elimination. However, since truth sets themselves do not have great use in the framework of games in which we are working, the use of more direct definitions avoids the addition of unnecessary obscurity.

As we have already mentioned, it has been shown very recently that Context Logic can be viewed as Modal Logic with Sahlqvist axioms. This means that soundness and completeness results for games can probably be derived from such general results for Modal Logic[1], instead of being derived from first principles, as here.

5 Adjunct Elimination in Context Logic for Trees

In this section, we investigate adjunct elimination for Context Logic for ordered trees. Specifically, we see what difficulties lie in naïvely applying the proof methodology used in [4], and demonstrate the existence of a counterexample to elimination of the '$\triangleright$' adjunct. We also provide a proof, using the games methodology, that the '$\triangleleft$' adjunct may be eliminated.

5.1 Overview

Given the game framework that we have now established for Context Logic for trees, we would now expect to proceed to attempt adjunct elimination by proving a conjecture, such as the following:

Conjecture 5.1. For all ranks $r, a, a' \in \mathcal{D}$ and $c, c' \in \mathcal{C}$, if

$$(a, a', r) \in DW_{\mathcal{D}} \tag{5.1}$$
$$(c, c', r) \in DW_{\mathcal{C}} \tag{5.2}$$

then

$$(c(a), c'(a'), r) \in DW_{\mathcal{D}} \tag{5.3}$$

This conjecture essentially expresses that playing either the $K \triangleleft P$ or $P \triangleright P$ move in a game does not help Spoiler to achieve a winning strategy. (In fact, not only this but also that any response that Duplicator makes in the move will give him a winning strategy, provided that it is 'sufficiently similar' to Spoiler's choice. As we have discussed in subsection 3.3, this is a stronger condition than would be necessary to prove adjunct elimination.) With such a result, we could then deduce the adjunct eliminability property using the results we have already proved that link games and formulae.

However, in attempting to prove this, we run into difficulties, particularly when we think about Spoiler playing the $K(P)$ move on the game $(c(a), c'(a'), r)$. In this move, Spoiler could, for instance, split $c(a)$ into c_1 and $a_1 = c_2(a)$ such that $c = c_1 \circ c_2$. This presents Duplicator with a significant obstacle, since the fact that Duplicator has a winning strategy for the game (c, c', r) is not enough to guarantee that c' can be split in a similar way.

This problem is, in fact, sufficient to put an end to the hope of proving the conjecture. We shall, guided by this, construct a counterexample which demonstrates that the '$\triangleright$' adjunct cannot be eliminated as would be possible if the conjecture held true.

5.2 Counterexample to elimination of the '$\triangleright$' adjunct

We shall now demonstrate that the '$\triangleright$' adjunct cannot be eliminated from Context Logic for Trees. In particular, we shall show a counterexample: a formula that does not have an equivalent formula that does not make use of '$\triangleright$'.

Proposition 5.1. *There is no formula without the $\triangleright$ connective which is equivalent to the formula $K_{\triangleright} = 0 \triangleright (True(\beta[0]))$.*

Proof. Suppose for a contradiction that K' is such an equivalent formula. Let n be the greatest nesting depth of the operator $u[K]$ in K'.

We recursively define some convenient context structures:

$$c_0 = \alpha[\_] \qquad c_{i+1} = \alpha[c_0] \tag{5.4}$$

$$d_0 = \beta[\_] \qquad d_{i+1} = \alpha[d_0] \tag{5.5}$$

Clearly $c_i \not\models \mathbf{0} \triangleright (True(\beta[0]))$, but $d_i \models \mathbf{0} \triangleright (True(c[0]))$ for all i.

We claim, however, that there is no formula K', not using '$\triangleright$' such that $c_i \not\models K'$, but $d_j \models K'$ for all i and $j \geq l$ for some l. To prove this claim, we proceed by induction on the nesting depth, n, of the $u[K]$ connective, supposing that some such K' exists.

For $n = 0$, the $u[K]$ connective is not used in K' at all, and so if $d_0 \models K'$, we have $c_0 \models K'$ also. (This is since no other connective can distinguish the labels α and β.)

For $n \geq 1$, we consider what the formula K' may be. K' is equivalent to some formula in disjunctive normal form (we manipulate the outer Boolean connectives, treating subformulae with non-Boolean outermost connectives as propositional atoms). We can therefore assume that K' is in this form. K' can only be a finite disjunction, and therefore it must have some component which is satisfied by an infinite number of the d_i contexts, but none of the c_i contexts. Thus, K' has a subterm of the form $\mathbf{I}$, $K \mid P$, $P \mid K$ or $u[K]$, which has nesting depth $k \leq n$ and is either satisfied by all d_j for all $j \geq l$ and by no c_i, or is satisfied by all c_i and no d_j for $j \geq l$.

Clearly, it could not be I. If it is $K \mid P$ or $P \mid K$ then $0 \models P$, $c_i \not\models K$ and $d_j \models K$ (or vice-versa). Since we have the same criteria on K, we can proceed this way until we obtain a subterm of the form $u[K'']$ which is satisfied by d_j and not c_i (or vice-versa) for all i and $j \geq l$.

Once we have a subterm of the form $u[K]$ which is satisfied by d_j and not c_i (or vice-versa) for any i and $j \geq l$, we see this must be of the form $\alpha[K]$ for some K which has nesting depth of $u[K]$ at most $n - 1$. Also, K is satisfied by d_j and not c_i (or vice-versa) for any i and $j \geq l - 1$. But by the induction hypothesis, such a K does not exist.

We have thus shown that there is no K' which is equivalent to $K_{\triangleright}$ and hence we have a counterexample to elimination of the '$\triangleright$' adjunct.

□

Informally, this counterexample exploits the fact that context connectives can, essentially, only look to a bounded depth within the context. The $K(P)$ connective permits looking at an arbitrary depth within a tree, but the only connectives that permit using tree connectives within a context formula are $P \triangleright P$, $K \mid P$ and $P \mid K$. The first of these does allow us to look effectively within a context at any depth, but the latter only permit us to look at arbitrary depth in branches of the context which are trees. The counterexample exploits this by hiding the difference between two trees in a context branch that is beyond the depth bound of the formula that is potentially equivalent to a formula with the '$\triangleright$' adjunct.

5.3 Elimination of the '◁' adjunct

Since we have demonstrated a counterexample to the elimination of '▷', we know that conjecture 5.1 is too strong to be provable. Instead, we state and prove a weaker theorem which allows us to prove elimination of the '◁' adjunct alone. This theorem essentially implies that, whenever Spoiler plays the $K \lhd P$ move, Duplicator can respond with the same context that Spoiler chose and have a winning strategy.

Theorem 5.2. *For all ranks,* $(n, s, \mathcal{L})$*, for all* $a, a' \in \mathcal{D}$*,* $c, c' \in \mathcal{C}$*, if*

$$(a, a', (n, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{i}$$
$$(c, c', (n, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{ii}$$

then for all contexts $d \in \mathcal{C}$

$$(d(a), d(a'), (n, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{A}$$
$$(d \mid a, d \mid a', (n, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{B}$$
$$(a \mid d, a' \mid d, (n, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{C}$$
$$(d \circ c, d \circ c'), (n, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{D}$$

For the purposes of this proof, we assume without loss of generality that in a game move Spoiler operates on the left hand context. We can make this assumption since the argument in the other case is identical upto renaming the variables, and we never assume that Spoiler will make this choice in moves other than that which we are directly considering (however, naturally, Spoiler *could* make this choice).

Proof. By induction on the rank r, and by cases on the possible moves that Spoiler may make in the games which we are to show that Duplicator will win. We denote the hypothesis that the proposition holds for lesser rank by $\mathbf{IH}_r$, to avoid confusion with inner inductions that we shall use.

We treat each part of the theorem separately, although they are interdependent inductively.

(A): $(d(a), d(a'), (n, s, \mathcal{L})) \in DW_{\mathcal{D}}$. (Data-structure moves apply.)

0 move:

If Spoiler can play this move, $d(a) = 0$ and $d(a') \neq 0$. Therefore $d = \_$ and $a = 0$. Further, $a' \neq 0$. This implies that Spoiler could play the **0** move and win the game $(a, a', (n, s, \mathcal{L}))$. This contracts (i), which states that Duplicator wins that game. Thus Spoiler must not be able to play the **0** move.

$K \lhd P$ move:

On playing this move, Spoiler chooses some context, call it $c_1 \in \mathcal{C}$. We shall show that if Duplicator responds by choosing $c_1' = c_1$ then Duplicator wins.

Applying downward closure to (i), we get

$$(a, a', (n, s - 1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.6}$$

Applying $\mathbf{IH}_r$ to this, we know that for all choices of context $\bar{c} \in \mathcal{C}$,

$$(\bar{c}(a), \bar{c}(a'), (n, s - 1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.7}$$

In particular, we may choose $\bar{c} = c_1 \circ d$, so

$$(c_1(d(a)), c_1(d(a')), (n, s-1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.8}$$

By game completeness, we also have

$$(c_1, c_1, (n, s-1, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.9}$$

Since, assuming Duplicator chooses $c_1' = c_1$, the game continues as either $(c_1(d(a)), c_1(d(a')), (n, s-1, \mathcal{L}))$ or $(c_1, c_1, (n, s-1, \mathcal{L}))$, and we have demonstrated that Duplicator has a winning strategy in both of these cases, we conclude that Duplicator wins in the case that Spoiler plays the $\lhd$ move.

$K(P)$ move:
Recall, the proposition we wish to show in this case is that, for all $d \in \mathcal{C}$, Duplicator has a winning strategy for the game $(d(a), d(a'), (n, s, \mathcal{L}))$ if Spoiler starts it by playing the $K(P)$ move. We shall prove this by induction on the structure of the context, d. To distinguish the induction hypothesis from $\mathbf{IH}_r$, we shall call it $\mathbf{IH}_d$, which denotes that the proposition holds for smaller d.

Base case: $d = \_$. Then we have $d(a) = a$ and $d(a') = a'$ and the result is immediate from (i).

Inductive case (1): $d = \kappa[d_1]$ for some $\kappa \in \Sigma$, $d_1 \in \mathcal{C}$.
For his move, Spoiler splits $d(a)$ into $c_1 \in \mathcal{C}, a_1 \in \mathcal{D}$. We consider the possibilities for c_1 and a_1, namely

$$c_1 = \_ \qquad a_1 = d(a) \tag{5.10}$$
$$c_1 = \kappa[c_2] \qquad a_1 = a_2 \tag{5.11}$$
$$c_1 = d(a) \mid \_ \qquad a_1 = 0 \tag{5.12}$$
$$c_1 = \_ \mid d(a) \qquad a_1 = 0. \tag{5.13}$$

Case (5.10) is trivial — Spoiler is effectively wasting his move. Specifically, if Duplicator responds with $c_1' = \_$ and $a_1' = d(a')$, game completeness gives Duplicator a winning strategy if Spoiler decides to continue with the game $(c_1, c_1', (n-1, s, \mathcal{L}))$. Also, $\mathbf{IH}_r$ part (A) (with the preconditions met by downward closure) gives Duplicator a winning strategy if Spoiler decides to continue with the game $(d(a), d(a'), (n-1, s, \mathcal{L}))$.

Case (5.11): Since $d(a) = (\kappa[d_1])(a) = \kappa[d_1(a)]$ and $\kappa[c_2(a_2)] = \kappa[d_1(a)]$, we know $c_2(a_2) = d_1(a)$. Thus, in the game $(d_1(a), d_1(a'), (n, s, \mathcal{L}))$, Spoiler may play the $K(P)$ move, splitting $d_1(a)$ as c_2 and a_2. $\mathbf{IH}_d$ specifies that Duplicator has an answer that gives a winning strategy in that case. In particular we can conclude that $d_1(a') = c_2'(a_2')$ such that

$$(c_2, c_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{5.14}$$
$$(a_2, a_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.15}$$

Since the game in (5.14) is of a lower rank, we can use $\mathbf{IH}_r$ part (D). This gives us that

$$(\kappa[\_] \circ c_2, \kappa[\_] \circ c_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.16}$$

Letting $a_1' = a_2'$ and $c_1' = \kappa[c_2']$, we can see that Duplicator has a winning strategy with this response, as (5.15) and (5.16) are the same as

$$(c_1, c_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{5.17}$$
$$(a_1, a_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.18}$$

Cases (5.12) and (5.13): In each of these cases, we shall show that the response $c_1' = d(a') \mid \_, a_1' = 0$ (or, symmetrically, $c_1' = \_ \mid d(a'), a_1' = 0$) is a winning one for Duplicator. Indeed, if Spoiler chooses to continue the game on the data-structure in the split, we know immediately by game completeness that

$$(a_1, a_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.19}$$

We apply downward closure on (i) to get the precondition necessary to make use of $\mathbf{IH}_r$ part (A):

$$(d(a), d(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.20}$$

With this, we can make use of $\mathbf{IH}_r$ parts (B) and (C) to obtain

$$(d(a) \mid \_, d(a') \mid \_, (n-1, s, \mathcal{L})) \in DW_C \text{ and} \tag{5.21}$$

$$(\_ \mid d(a), \_ \mid d(a'), (n-1, s, \mathcal{L})) \in DW_C \tag{5.22}$$

as necessary. These are each the remaining possibility for the game to continue (Spoiler choosing to continue with the context) and we have now demonstrated that Duplicator has a winning strategy, so Duplicator wins in this event.

Inductive case (2): $d = a_1 \mid d_1$, with $a_1 \in \mathcal{D}$ and $d_1 \in C$. In particular, we assume that $a_1 = \kappa[a_2]$ for some $\kappa \in \Sigma, a_2 \in \mathcal{D}$. This is a reasonable assumption, as associativity of the '$\mid$' operator permits us to take the left-most branch of the context and be left with a smaller context. Here, $\mathbf{IH}_d$ tells us that Duplicator has a winning strategy if Spoiler begins the game $(d_1(a), d_1(a'), (n, s, \mathcal{L}))$ with the $K(P)$ move.

We shall consider the ways in which Spoiler may choose to spilt $d(a) = \kappa[a_2] \mid d_1(a)$ into c_1 and a_3, with $d(a) = c_1(a_3)$. These are:

$$c_1 = c_2 \mid d_1(a) \qquad c_2(a_3) = a_1 \tag{5.23}$$

$$c_1 = a_1 \mid c_2 \qquad c_2(a_3) = d_1(a) \tag{5.24}$$

$$c_1 = \_ \mid a_4 \qquad a_3 = a_1 \mid a_5 \qquad d_1(a) = a_5 \mid a_4 \tag{5.25}$$

In (5.23) we shall show that, if Duplicator responds with $c_1' = c_2 \mid d_1(a'), a_3$, he has a winning strategy. By downward closure on (i) and $\mathbf{IH}_r$ part (A) we know that

$$(d_1(a), d_1(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.26}$$

Using this, we can apply $\mathbf{IH}_r$ part (B) to deduce that

$$(c_2 \mid d_1(a), c_2 \mid d_1(a'), (n-1, s, \mathcal{L})) \in DW_C. \tag{5.27}$$

Completeness of games gives the other required result, that is

$$(a_3, a_3, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.28}$$

proving that this gives a winning strategy for Duplicator.

In (5.24), we notice that $\mathbf{IH}_d$ provides Duplicator with a c_2' and a_3' such that

$$c_2'(a_3') = d_1(a)' \tag{5.29}$$

$$(c_2, c_2', (n-1, s, \mathcal{L})) \in DW_C \tag{5.30}$$

$$(a_3, a_3', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.31}$$

We can apply $\mathbf{IH}_r$ part (D) to (5.30), to deduce

$$((a_1 \mid \_) \circ c_2, (a_1 \mid \_) \circ c_2', (n-1, s, \mathcal{L})) \in DW_C. \tag{5.32}$$

Together, (5.31) and (5.32) show that the response of $a_1 \mid c_2', a_3'$ gives Duplicator a winning strategy.

In (5.25), we notice that $\mathbf{IH}_d$ provides Duplicator with a c_1' and a_5' such that

$$c_1'(a_5') = d_1(a)' \tag{5.33}$$
$$(\_ \mid a_4, c_1', (n-1, s, \mathcal{L})) \in DW_C \tag{5.34}$$
$$(a_5, a_5', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.35}$$

This is not immediately sufficient, since we are not assured that $c_1'(a_1 \mid a_5') = a_1 \mid d_1(a')$. We shall see that, for a large enough rank, we can ensure this condition. For smaller ranks, we shall take a more direct approach to finding Duplicator a winning strategy.

To prove the result, we shall look at cases of the possible value of n. In order to be playing the $K(P)$ move we know that $n > 0$. Since Spoiler cannot win immediately with the $K(P)$ move, we can see that any split is a winning strategy for Duplicator if $n = 1$.

In the case where $n = 2$, we need to show that Duplicator has a response such that the data-structures cannot be discriminated between with the $\mathbf{0}$ move and the contexts cannot be discriminated between with either of the $\mathbf{I}$ or $u[K]$ moves. We shall show that either the response $(\_ \mid d_1(a'), a_1)$ or $(\_, a_1 \mid d_1(a'))$ is a winning one for Duplicator. That is, either

$$(\_ \mid a_4, \_ \mid d_1(a'), (n-1, s, \mathcal{L})) \in DW_C \quad \text{and} \tag{5.36}$$
$$(a_1 \mid a_5, a_1, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}, \quad \text{or} \tag{5.37}$$
$$(\_ \mid a_4, \_, (n-1, s, \mathcal{L})) \in DW_C \quad \text{and} \tag{5.38}$$
$$(a_1 \mid a_5, a_1 \mid d_1(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.39}$$

We already know that Spoiler's choice of data-structure $(a_1 \mid a_5)$ is not 0, and a_1 and $a_1 \mid d_1(a')$ are also not 0, so the $\mathbf{0}$ move cannot be used to discriminate.

If Spoiler picks as his context $\_$, the second response pair, where Duplicator replies with $\_$, is a winning one for Duplicator.

If Spoiler does not pick the context $\_$, then the first pair will be a winning response for Duplicator, since neither the $\mathbf{I}$ move nor the $u[K]$ move can be applied to let Spoiler win.

In the case where $n \geq 3$, we can consider how a game on (5.34) would unfold. In particular, Spoiler could play the $K \mid P$ move, so we know that $c_1' = c_2' \mid a_4'$ for some $c_2' \in \mathcal{C}$, $a_4' \in \mathcal{D}$ with

$$(\_, c_2', (n-2, s, \mathcal{L})) \in DW_C \tag{5.40}$$
$$(a_4, a_4', (n-2, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.41}$$

Since $n \geq 3$, we know by (5.40) that $c_2' = \_$, so $c_1' = \_ \mid a_4'$. This shows that there is a splitting of $d(a')$ into $c_1' = \_ \mid a_4'$ and $a_3' = a_1 \mid a_5'$. Further, we wish to show

$$(c_1, c_1', (n-1, s, \mathcal{L})) \in DW_C \tag{5.42}$$
$$(a_3, a_3', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.43}$$

Note that we have (5.42) already, since this is just (5.34).

We can apply $\mathbf{IH}_r$ part (A) to (5.35) to give

$$((a_1 \mid \_)(a_5), (a_1 \mid \_)(a_5'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}, \tag{5.44}$$

which is just (5.43) as required.

Inductive case (3): $d = d_1 \mid a_1$, with $a_1 \in \mathcal{D}$ and $d_1 \in \mathcal{C}$. This case is symmetric to case (2), hence the reasoning is identical and so omitted here.

Cases (1), (2) and (3) and the base case cover all possible inductive constructions for the context d, and we have proven each of them. Therefore, Duplicator has a winning strategy whenever Spoiler starts with the $K(P)$ move.

$P \mid P$ move:

Again, we reason by induction on the structure of the context d, calling the inductive hypothesis '$\mathbf{IH}_d$'.

Base case: $d = \_$. As before, this case is trivial, as the result is exactly the assumption (i).

Inductive case (1): $d = \kappa[d_1]$. This case is also trivial, as Spoiler must split $d(a)$ into 0 and $d(a)$ (in some order). Therefore, responding with 0 and $d(a')$ (in the same order) gives Duplicator a winning strategy by downward closure on (i) and $\mathbf{IH}_r$ part (A).

Inductive case (2): $d = a_1 \mid d_1$, for some $a_1 \in \mathcal{D}$, $d_1 \in \mathcal{C}$. Consider the possibilities for Spoiler to split $d(a) = a_1 \mid d_1(a)$ into a_2 and a_3, which are

$$a_1 = a_2 \mid a_4 \qquad\qquad a_3 = a_4 \mid d_1(a) \tag{5.45}$$

$$a_2 = a_1 \mid a_4 \qquad\qquad d_1(a) = a_4 \mid a_3. \tag{5.46}$$

In (5.45), we show that Duplicator has a winning strategy by responding with $a_2' = a_2$ and $a_3' = a_4 \mid d_1(a')$. Indeed, by game completeness we know that

$$(a_2, a_2, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.47}$$

We apply downward closure to (i) and use $\mathbf{IH}_r$ part (A) to get

$$((a_4 \mid d_1)(a), (a_4 \mid d_1)(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.48}$$

The above assert that the choices for Duplicator give a winning strategy, whichever way Spoiler chooses to continue the game.

In (5.46), we use that by $\mathbf{IH}_d$, $d_1(a') = a_4' \mid a_3'$ such that

$$(a_4, a_4', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.49}$$

$$(a_3, a_3', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.50}$$

Using (5.49) with $\mathbf{IH}_r$ part (A), we get

$$((a_1 \mid \_)(a_4), (a_1 \mid \_)(a_4'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.51}$$

Together, (5.51) and (5.50) show that, if Duplicator responds with $a_2' = a_1 \mid a_4'$ and a_3', there is a winning strategy.

Inductive case (3): $d = d_1 \mid a_1$, for some $a_1 \in \mathcal{D}$, $d_1 \in \mathcal{C}$. Once again, this case is symmetric to case (2), so the proof is omitted.

The above inductive reasoning proves by induction that Duplicator has a winning strategy when Spoiler plays the $P \mid P$ move.

Further, we have now looked at all the game moves that apply to case (A).

(B): $(d \mid a, d \mid a', (n, s, \mathcal{L})) \in DW_C$. (Context moves apply.)

I move:

If Spoiler were able to play the **I** move, $d \mid a = \_$. Therefore, $d = \_$ and $a = 0$. By the assumption (i), we conclude that $a' = 0$ also. Thus $d \mid a' = \_$ and Spoiler would not be able to play this move after all.

$P \triangleright P$ move:

For this move, Spoiler picks some $a_1 \in \mathcal{D}$ to play with. We shall demonstrate that if Duplicator chooses $a'_1 = a_1$ then he has a winning strategy.

If Spoiler decided to continue the game on a_1 and a'_1, game completeness automatically gives Duplicator a winning strategy, that is

$$(a_1, a'_1, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.52}$$

Spoiler may also continue the game on $(d \mid a)(a_1) = d(a_1) \mid a$ and $(d \mid a')(a_1) = d(a_1) \mid a'$. In this event, applying downward closure to (i) and $\mathbf{IH}_r$ part (A) we get

$$((d(a_1) \mid \_)(a), (d(a_1) \mid \_)(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.53}$$

Thus, Duplicator has a winning strategy in this case also.

$u[K]$ move:

In order for Spoiler to play this move, it must be the case that $d \mid a = \kappa[d_1]$ for some $\kappa \in \mathcal{L}$, $d_1 \in C$. This implies that $a = 0$ and $d = \kappa[d_1]$. If $a = 0$, then the assumption (i), by considering the $\mathbf{0}$ move, gives that $a' = 0$ also.

$K \mid P$ move:

Consider the ways that Spoiler can split $d \mid a$ into $c_1 \in C$ and $a_1 \in \mathcal{D}$, which are:

$$d = c_1 \mid a_2 \qquad a_1 = a_2 \mid a \tag{5.54}$$

$$c_1 = d \mid a_2 \qquad a = a_2 \mid a_1 \tag{5.55}$$

In the case of (5.54), we shall show that the response $c'_1 = c_1, a'_1 = a_2 \mid a'$ gives Duplicator a winning strategy. By game completeness

$$(c_1, c'_1, (n-1, s, \mathcal{L})) \in DW_C. \tag{5.56}$$

By downward closure on (i) and applying $\mathbf{IH}_r$ part (A) we also have

$$((a_2 \mid \_)(a), (a-2 \mid \_)(a'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.57}$$

The deductions (5.56) and (5.57) establish that whichever way Spoiler chooses to continue this game, Duplicator has a winning strategy.

In the case of (5.55), we deduce from (i), by applying the $P \mid P$ move, that $a' = a'_2 \mid a'_1$ such that

$$(a_2, a'_2, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.58}$$

$$(a_1, a'_1, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.59}$$

We can then apply $\mathbf{IH}_r$ part (B) with (5.58) to give that

$$(d \mid a_2, d \mid a'_2, (n-1, s, \mathcal{L})) \in DW_C. \tag{5.60}$$

The deduction (5.59) and (5.60) establish that, if Duplicator responds with $c_1 = d \mid a_2'$ and a_1', then whichever way Spoiler chooses to continue this game, Duplicator has a winning strategy.

$P \mid K$ move:
Spoiler must split $d \mid a$ into a_1 and $d_1 = d_2 \mid a$ such that $d = a_1 \mid d_1$. If Duplicator responds with $a_1' = a_1$ and $d_1' = d_2 \mid a'$, then by downward closure on (i) and $\mathbf{IH}_r$ part (B) we have

$$(d_2 \mid a, d_2 \mid a', (n-1, s, \mathcal{L})) \in DW_C. \tag{5.61}$$

Game completeness also gives

$$(a_1, a_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.62}$$

which proves that this response gives a winning strategy for Duplicator.

We have now looked at all the game moves that apply to case (B).

(C): $(a \mid d, a' \mid d, (n, s, \mathcal{L})) \in DW_C$. (Context moves apply.) This is symmetric to case (B), so the proof is omitted here.

(D): $(d \circ c, d \circ c', (n, s, \mathcal{L})) \in DW_C$. (Context moves apply.)

I move:
If Spoiler were able to play this move, that would mean that $d \circ c = \_$, so $d = \_$ and $c = \_$. By (ii), applying the **I** move, we conclude that $c' = \_$ also, and Spoiler would not have been able to play this move after all.

$P \rhd P$ move:
For this move, Spoiler picks some $a_1 \in \mathcal{D}$ to play with. By applying the $P \rhd P$ move on (ii) we deduce that there is an $a_1' \in \mathcal{D}$ such that

$$(a_1, a_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.63}$$

$$(c(a_1), c'(a_1'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.64}$$

If we then apply $\mathbf{IH}_r$ part (A) with (5.64) we get

$$(d(c(a_1)), d(c'(a_1')), (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.65}$$

Between them, (5.63) and (5.65) prove that the response of a_1' and $(d \circ c')(a_1')$ gives Duplicator a winning strategy.

$u[K]$ move:
If Spoiler is to play this move, then there are two cases for the context d (since $d(c) = \kappa[c_1]$ for some $\kappa \in \mathcal{L}$, $c_1 \in C$): either $d = \_$ or $d = \kappa[d_1]$ for some $d_1 \in C$.

If $d = \_$ then the result follows immediately from (ii).

If $d = \kappa[d_1]$ then Spoiler does not win in this move and, by applying downward closure to (ii) and applying $\mathbf{IH}_r$ part (D) we conclude that

$$(d_1 \circ c, d_1 \circ c', (n-1, s, \mathcal{L})) \in DW_C. \tag{5.66}$$

This is enough to conclude that Duplicator has a winning strategy.

$K \mid P$ move:
We shall once again make use of induction on the structure of the context d.

Base case: $d = \_$. The result is immediate.

Inductive case (1): $d = \kappa[d_1]$ for some $\kappa \in \Sigma$, $d_1 \in \mathcal{C}$. In this case, Spoiler may only choose the context $\kappa[d_1] \circ c$ and data-structure 0. By applying downward closure to (ii) and using $\mathbf{IH}_r$, we get

$$(d \circ c, d \circ c', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.67}$$

Also, plainly

$$(0, 0, (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.68}$$

(5.67) and (5.68) give that Duplicator has a winning strategy with the response $d \circ c'$, 0.

Inductive case (2): $d = d_1 \mid a_1$ for some $d_1 \in \mathcal{C}$, $a_1 \in \mathcal{D}$. Consider the ways Spoiler may spit $d \circ c = (d_1 \circ c) \mid a_1$ into c_1 and a_2, which are

$$c_1 = (d_1 \circ c) \mid a_3 \qquad a_1 = a_3 \mid a_2 \tag{5.69}$$

$$d_1 \circ c = c_1 \mid a_3 \qquad a_2 = a_3 \mid a_1 \tag{5.70}$$

In the case of (5.69), we will show that the response $c_1' = (d_1 \circ c') \mid a_3$, $a_2' = a_2$ gives Duplicator a winning strategy. By applying downward closure to (ii) and using $\mathbf{IH}_r$ we get that

$$((d_1 \mid a_3) \circ c, (d_1 \mid a_3) \circ c', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.71}$$

Further, by game completeness

$$(a_2, a_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.72}$$

Together, (5.71) and (5.72) prove that the response does indeed give a winning strategy for Duplicator.

In the case of (5.70), we know by $\mathbf{IH}_d$ that $d_1 \circ c' = c_1' \mid a_3'$ for $c_1' \in \mathcal{C}$, $a_3' \in \mathcal{D}$ such that

$$(c_1, c_1', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{5.73}$$

$$(a_3, a_3', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.74}$$

We can then apply $\mathbf{IH}_r$ part (A) with (5.74) to deduce

$$((\_ \mid a_1)(a_3), (\_ \mid a_1)(a_3'), (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.75}$$

Together, (5.73) and (5.75) prove that the response $c_1', a_2 = a_3' \mid a_1$ gives Duplicator a winning strategy.

Inductive case (3): $d = a_1 \mid d_1$ for some $d_1 \in \mathcal{C}$, $a_1 \in \mathcal{D}$. In this case, the only way that Spoiler can split $d \circ c = a_1 \mid (d_1 \circ c)$ is into $c_1 = a_1 \mid c_2$ and a_2 such that $d_1 \circ c = c_2 \mid a_2$.

By $\mathbf{IH}_d$ we know that $d_1 \circ c' = c_2' \mid a_2'$ such that

$$(c_2, c_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}} \tag{5.76}$$

$$(a_2, a_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.77}$$

We now apply $\mathbf{IH}_r$ part (D) with (5.76) and deduce that

$$((a_1 \mid \_) \circ c_2, (a_1 \mid \_) \circ c_2', (n-1, s, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{5.78}$$

Together, (5.77) and (5.78) give that, if he responds with $c'_1 = a_1 \mid c'_2$ and a'_2, Duplicator has a winning strategy in this case.

$P \mid K$ move:
This case is just symmetric to the $K \mid P$ case, and so the reasoning is omitted.

□

The next corollary states that we can eliminate adjunct moves and Spoiler will still have a winning strategy.

Corollary. *If $r = (n, s, \mathcal{L})$ and $r' = (n, s+s', \mathcal{L})$ for some $s' \in \mathbb{N}$, then, for all $a, a' \in \mathcal{D}$, $c, c' \in C$:*

$$(a, a', r) \in DW_{\mathcal{D}} \Longrightarrow (a, a', r') \in DW_{\mathcal{D}} \tag{5.79}$$
$$(c, c', r) \in DW_C \Longrightarrow (c, c', r') \in DW_C \tag{5.80}$$
$$(a, a', r') \in SW_{\mathcal{D}} \Longrightarrow (a, a', r) \in SW_{\mathcal{D}} \tag{5.81}$$
$$(c, c', r') \in SW_C \Longrightarrow (c, c', r) \in SW_C \tag{5.82}$$

Proof. We prove (5.79) and (5.80) by induction on the rank r' and cases on the first move that Spoiler plays in the game (a, a', r') or (c, c', r'). Parts (5.81) and (5.82) follow immediately.

If Spoiler's move allows him to win, then he could have played precisely the same move on (a, a', r) (or (c, c', r)). Thus Spoiler cannot win immediately in the new game.

If Spoiler plays the $K \lhd P$ move, then we have by induction that

$$(a, a', (n, s + s' - 1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.83}$$

Spoiler selects some context c_1. By theorem 5.2, we know that

$$(c_1(a), c_1(a'), (n, s + s' - 1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.84}$$

By game completeness, we know that

$$(c_1, c_1, (n, s + s' - 1, \mathcal{L})) \in DW_C. \tag{5.85}$$

Therefore, if Duplicator responds with the same context, c_1, he has a winning strategy however the game continues.

In any other case, the game continues with some other pair of trees or contexts and a smaller rank. Spoiler would be able to play that same move on the original game, and so Duplicator has a winning strategy in the proceeding game with rank $(n - 1, s, \mathcal{L})$. By induction, Duplicator has a winning strategy for that same game with rank $(n - 1, s + s', \mathcal{L})$ and so has a winning strategy overall. □

We are now in a position to show the final corollary, that the '$\lhd$' adjunct can be eliminated from the logic.

Corollary. *If $r = (n, 0, \mathcal{L})$ and $r' = (n, s, \mathcal{L})$, then any formula of rank r' is equivalent to a formula of rank r.*

Proof. Suppose P is a formula of rank r', and let $\mathcal{T} = \{a : a \models P\}$. By game soundness, if $a \in \mathcal{T}$ and $a' \notin \mathcal{T}$ then $(a, a', r') \in SW_{\mathcal{D}}$, and so, by the above corollary, $(a, a', r) \in SW_{\mathcal{D}}$. Hence, by game completeness, there is a formula $P_{a,a'}$ of rank r that discriminates between a and a'. We then apply lemma 4.3 do see that there is a formula P' of rank r which defines the set $\mathcal{T}$. Thus P' is equivalent to P.

The argument in the case of contexts is just the same. □

5.4 Comments on the Proof

It is probably not clear, without examination of the proof, why theorem 5.2 includes parts (B), (C) and (D), since these are not used in the corollaries proving adjunct eliminability.

It is clear from examination of the proof of theorem 5.2 that the most complicated case necessary of consideration is that when, in the game in part (A), Spoiler plays the $K(P)$ move. This is since Spoiler may split $d(a)$ in a highly arbitrary manner, giving rise to numerous cases for which we must prove

Note that we have also made use of a derived $P \mid P$ move in the proof. The reason for including this is to simplify finding a response for Duplicator in the case of (5.55). This is at the expense of introducing an additional case in the proof of part (i). Instead of doing this, we could have made use of the method used in (5.25).

5.5 Simplifications to the Proof

We now present a lemma, which permits a simplification to the proof of theorem 5.2 by effectively rendering consideration of the adjunct move $K \lhd P$ redundant. We did not use this lemma in the proof that we gave, as the $K \lhd P$ case is relatively short and provides a more direct proof. Variations on the lemma, however, should be even more useful when more than one move type is to be eliminated, for instance, if we were attempting to eliminate both the adjuncts of application. We shall later consider the question of adjunct elimination in Context Logic extended with composition and its adjuncts. In any proof of this, being able to ignore cases of all four adjunct connectives would certainly permit a more compact presentation.

Lemma 5.3. *If for ranks* $(n, 0, \mathcal{L})$ *and all* $a, a' \in \mathcal{D}$

$$(a, a', (n, 0, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.86}$$

implies that for all $d \in C$

$$(d(a), d(a'), (n, 0, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.87}$$

then for all ranks $(n, s, \mathcal{L})$ *and all* $a, a' \in \mathcal{D}$

$$(a, a', (n, 0, \mathcal{L})) \in DW_{\mathcal{D}} \tag{5.88}$$

implies that for all $d \in C$

$$(d(a), d(a'), (n, s, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.89}$$

Proof. By induction on s.

If $s = 0$ the result is immediate.

If $s > 0$ then consider how the game $(d(a), d(a'), (n, s, \mathcal{L}))$ would be played, with Spoiler attempting to win. When Spoiler plays a move that is not $K \lhd P$, we shall have Duplicator's strategy be to make the same response as he would for the game $(d(a), d(a'), (n, s-1, \mathcal{L}))$. The game continues this way, and if Spoiler never plays the $K \lhd P$ move, Duplicator will obviously win, since Spoiler may make no other move than those covered in the game we know that Duplicator has a winning strategy for. If Spoiler eventually decides to play the $K \lhd P$ move, the game will be of the form

$$(a_1, a_1', (n', s, \mathcal{L})), \tag{5.90}$$

with $n' \leq n$, and for Duplicator to have a winning strategy, it is sufficient to show that, for any $d_1 \in \mathcal{C}$ (which Spoiler may choose),

$$(d_1, d_1, (n', s-1, \mathcal{L})) \in DW_{\mathcal{C}} \tag{5.91}$$

$$(d_1(a_1), d_1(a_1'), (n', s-1, \mathcal{L})) \in DW_{\mathcal{D}}. \tag{5.92}$$

Now (5.91) holds by game completeness. (5.92) holds by induction, after we apply downward closure on (5.88). Thus by responding with the same context, d_1, Duplicator has a winning strategy.

□

Variations on this lemma can also be used to simplify other games-based adjunct elimination results, and henceforth we shall generally take advantage of the simplification such lemmas afford, namely ignoring adjunct cases in proofs as appropriate.

6 Adjunct Elimination in Context Logic for Trees: Evaluation

We shall now take a critical view of the work we have presented on adjunct elimination in Context Logic for trees, and see what work remains to be done, partially as a consequence of our results.

6.1 General Notes

The counterexample presented in the previous section described a particular context formula that is not expressible without the '$\triangleright$' adjunct. We have not, however, demonstrated the existence of any data formula that is not expressible without '$\triangleright$', and so the question remains as to whether such a formula exists.

One might consider this to be unlikely, since the issue we exploited in our construction was that certain differences between contexts cannot be examined using the non-adjunct context formulae. When considering data-structures, however, the $K(P)$ connective would seem to permit any given substructure to be examined in order to notice differences. On the other hand, one might suspect that such a formula may well exist, on account of the subtleties involved in finding differences between structures given the adjuncts. Such subtleties are a problem we shall also see later in discussions considering whether Context Logic with context composition may admit adjunct elimination.

Proving that no such counterexample exists is likely to be difficult. It would require a sophisticated adaption to the games methodology, or some other manipulation on the logic, in order to render the proof technique we have used here suitable for solving this problem.

It does, however, remain an interesting question, as proving the result one way or the other will give us greater insight into the expressivity of the logic without adjuncts.

6.2 Applicability to Other Models

Since the definition of Context Logic is by no means specific to the tree model, a significant question is, "In which models is adjunct elimination possible?". Of particular interest are other forms of data-structure, such as unordered trees and directed acyclic graphs. Context Logic's ability to reason about data on the level of data-structures is a key motivating feature for its use, and determining adjunct elimination results for these other models will also be interesting.

6.2.1 Sequences

One data-structure of interest is sequences over some alphabet, Σ. A grammar for sequences and sequence-contexts might be:

$$\text{sequences} \qquad d ::= 0 \mid a \in \Sigma \mid d \cdot d \tag{6.1}$$

$$\text{contexts} \qquad s ::= \_ \mid d \cdot c \mid c \cdot d \tag{6.2}$$

We take the '$\cdot$' operator to be associative but (obviously) not commutative.

It is clear that sequences are merely a special case of trees: specifically, the flat trees. Knowing this, we can expect to apply the elimination result for the '$\triangleleft$' adjunct almost directly to Context Logic for sequences.

We might also expect that the '$\triangleright$' adjunct can be eliminated in the sequence case, especially since our counterexample and reasoning for the tree case is obviously no longer applicable.

6.2.2 Unordered Trees

Another interesting model for Context Logic is unordered trees: when we take the '$\mid$' operator to be commutative. This is the model that has been used in the proofs regarding adjunct elimination in Ambient Logic.

Clearly, since we do not consider the '$\mid$' connective in the construction or proof of the counterexample to elimination of '$\triangleright$', the counterexample will hold just as well in the case of unordered trees. Within the proof of elimination of '$\triangleleft$', we did make assumptions about how trees and contexts can be split — assumptions which do not hold in the unordered case. However, we expect that the cases requiring consideration to adapt the proof to unordered trees will be sufficiently similar to those for the ordered case that such an adaptation will not be difficult.

6.3 Context Logic with Context Composition

6.3.1 Motivation

We have seen that $0 \triangleright (True(\beta[0]))$ is a counterexample to the elimination of the '$\triangleright$' adjunct in Context Logic for trees, however, it is apparent that this counterexample does have an equivalent adjunct-free formula if the logic is extended with a context composition connective. This is since the counterexample relies on being unable to discriminate two contexts if the difference is at a sufficient depth. With context composition, it is possible to break the context up at an arbitrary depth.

With the composition connective, $K \circ K$, it seems natural to introduce the corresponding adjunct connectives, also: $K \circ\!\!- K$ and $K -\!\!\circ K$. Indeed, extending Context Logic with context composition has been investigated previously and found not to increase the expressive power of data formulae.

We note that one of the composition adjuncts was implicitly eliminated as a consequence of the proof of elimination of $K \triangleleft P$ previously. Thus we expect that a theorem that eliminates $K \triangleright P$ will also eliminate the remaining adjunct of $K \circ K$.

6.3.2 Issues with proof

It is not currently known whether a general adjunct elimination result holds for this logic with composition, but we shall highlight some of the problems encountered in an attempt to prove and refute such a claim. Much of the subtlety of the problem is generated by the cases in which a composed tree $c(a)$ may be split into c_1 and a_1 with $c(a) = c_1(a_1)$. There are four essentially different types of splitting, which are illustrated by figures 1, 2, 3, 4 and 5.

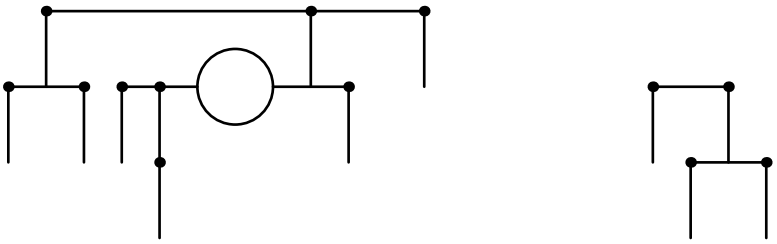

Figure 1: Context c and data a

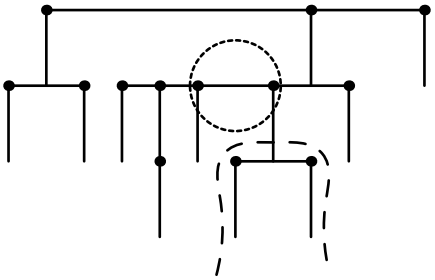

Figure 2: Splitting type (1)

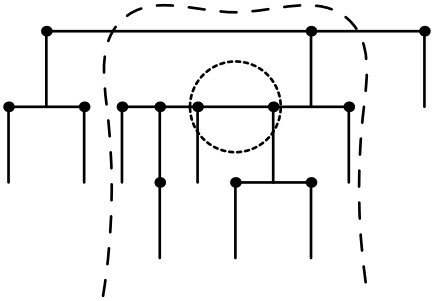

Figure 3: Splitting type (2)

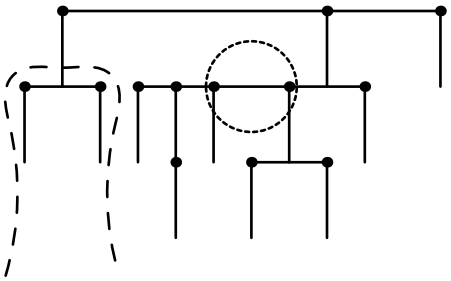

Figure 4: Splitting type (3)

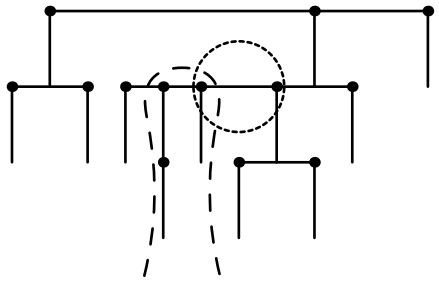

Figure 5: Splitting type (4)

For the purposes of this section, we shall express a game rank as a tuple $(r, s, \mathcal{L})$ where r is the number of non-adjunct moves ($\mathbf{0}$, $K(P)$, $\mathbf{I}$, $K \circ K$, $u[K]$ and $K \mid K$) that may be played, s is the number of adjunct moves that may be played, and $\mathcal{L}$ is the finite set of labels that may be used. Additionally, we may abbreviate the rank $(r, 0, \mathcal{L})$ as simply r, where this is unambiguous.

The following conjecture, which we may wish to prove, would permit elimination of all of the adjuncts in the logic:

Conjecture 6.1. For all $r \in \mathbb{N}$, finite $\mathcal{L} \subseteq \Sigma$. If, for $c, c', d, d' \in \mathcal{C}$, $a, a' \in \mathcal{D}$,

$$(c, c', (r, 0, \mathcal{L})) \in DW_{\mathcal{C}} \tag{6.3}$$
$$(d, d', (r, 0, \mathcal{L})) \in DW_{\mathcal{C}} \tag{6.4}$$
$$(a, a', (r, 0, \mathcal{L})) \in DW_{\mathcal{D}} \tag{6.5}$$

then

$$(c(a), c'(a'), (r, 0, \mathcal{L})) \in DW_{\mathcal{D}} \tag{6.6}$$
$$(c \circ d, c' \circ d', (r, 0, \mathcal{L})) \in DW_{\mathcal{C}}. \tag{6.7}$$

We shall look at how an attempt to prove this becomes problematic. In particular, we shall suppose the proof will be by induction on the rank and by cases on Spoiler's choice of move in the games in (6.6) and (6.7), and examine the $K(P)$ move case in the game (6.6).

To play the $K(P)$ move, Spoiler splits $c(a)$ into a context, c_1, and data-structure, a_1, such that $c(a) = c_1(a_1)$. For practical purposes, there are four different ways in which this can be done, as illustrated in figures (1, 2, 3, 4, 5). We shall consider each of these cases individually. In each case, we wish to show that Duplicator has a corresponding splitting $c'(a') = c'_1(a'_1)$ such that $(c_1, c'_1, r-1) \in DW_{\mathcal{C}}$ and $(a_1, a'_1, r-1) \in DW_{\mathcal{D}}$.

Splitting type (1): In this case, $a = c_2(a_1)$ and $c_1 = c \circ c_2$. By (6.5), we know that $a' = c'_2(a'_1)$ such that $(c_2, c'_2, kr-1) \in DW_{\mathcal{C}}$ and $(a_1, a'_1, kr-1) \in DW_{\mathcal{D}}$. (This is by playing the $K(P)$ move on that game.)

By downward closure, $(c, c', r-1) \in DW_{\mathcal{C}}$. Thus, by induction, (6.7) gives $(c \circ c_2, c' \circ c'_2, r-1) \in DW_{\mathcal{C}}$. Thus we conclude that the response $c'_1 = c' \circ c'_2$, a'_1 would be a winning one for Duplicator.

Splitting type (2): In this case, $c = c_1 \circ c_2$ and $a_1 = c_2(a)$. By (6.3), we know that $c' = c'_1 \circ c'_2$ such that $(c_1, c'_1, r-1) \in DW_{\mathcal{C}}$ and $(c_2, c'_2, r-1) \in DW_{\mathcal{C}}$. (This is by playing the $K \circ K$ move on that game.)

By downward closure, $(a, a', r-1) \in DW_{\mathcal{D}}$. Thus, by induction, (6.6) gives $(c_2(a), c_2'(a'), r-1) \in DW_C$. Thus we conclude that the response c_1', $a_1' = c_2'(a')$ is a winning one for Duplicator.

Splitting Type (4): In this case, $c = c_2 \circ (a_2 \mid \_)$, $a = (\_ \mid a_4)(a_3)$, $c_1 = c_2 \circ (\_ \mid a_4)$ and $a_1 = (a_2 \mid \_)(a_3)$. By (6.3) $c' = c_2' \circ c_3'$ with $(c_2, c_2', r-1) \in DW_C$ and $((a_2 \mid \_), c_3', r-1) \in DW_C$. By (6.5) $a' = c_4'(a_3')$ with $((\_ \mid a_4), c_4', r-1) \in DW_C$ and $(a_3, a_3', r-1) \in DW_{\mathcal{D}}$.

We expect Duplicator's response to be $c_1' = c_2' \circ c_3'$, $a_1' = c_4'(a_3')$, however, in order to do so we would need to demonstrate that $c'(a') = c_1'(a_1')$. In particular, we can do this in the case that $r \geq 3$. Specifically, we can demonstrate that $c_3' = a_2' \mid \_$ and $c_4' = \_ \mid a_4'$. Applying the $P \mid K$ move to the game $((a_2 \mid \_), c_3', r-1)$, $c_3' = a_2' \mid c_5'$ where $(\_, c_5', r-2) \in DW_C$. It follows then that $c_5' = \_$, so $c_3' = a_2' \mid \_$.

Likewise, playing the $K \mid P$ move on the game $((\_ \mid a_4), c_4', r-1)$ we get that $c_4' = c_6' \mid a_4'$ with $(\_, c_6', r-1) \in DW_C$. Thus $c_6' = \_$ and so $c_4' = \_ \mid a_4'$.

Assuming that the result can be shown for the cases where $r < 3$, we can apply induction to deduce that $(c_2 \circ c_3, c_2' \circ c_3', r-1) \in DW_C$ and $(c_4(a_3), c_4'(a_3'), r-1) \in DW_{\mathcal{D}}$. We omit this check, as it is likely similar to a case in the proof of $\lhd$ elimination previously.

Splitting type (3): In this case, $c = c_2 \circ (c_3(a_1) \mid c_4)$ and $c_1 = c_2 \circ (c_3 \mid c_4(a))$. There is also a distinct but symmetric case where $c = c_2 \circ (c_4 \mid c_3(a_1))$, however, the proof in this case is the same upto symmetry, so we shall only consider the first case.

This is the case where particularly problematic issues arise, specifically, concerning establishing the existence of a c_1' and a_1' with the properties that

$$(a_2 \circ (c_3 \mid c_4(a)), c_1', r-1) \in DW_C \tag{6.8}$$

$$(a_1, a_1', r-1) \in DW_{\mathcal{D}} \tag{6.9}$$

The most obvious method is to try to construct an answering $c_1' = c_2' \circ (c_3' \mid c_4'(a'))$ and a_1' with $c' = c_2' \circ (c_3'(a_1') \mid c_4')$. This is done by splitting up c' in a way to match c, then applying the inductive hypothesis to put the parts back together.

This issue with this is that, in order to apply the inductive hypothesis as we have stated it to achieve (6.8), we require the parts we shall be composing to be indistinguishable from their counterparts in a game of rank $r-1$. However, there is no single game move which can produce these components (c_2',c_3', c_4' and a_1') to match.

To attack the problem from another angle, if we can produce the components which correspond to Spoiler's counterparts in some fixed number of moves. If we modify the assumptions (6.5), (6.3) and (6.4) to use a rank greater than r, given by some function $f(r)$. By downward closure of games, we may assume that f is monotone increasing. (Suppose for $r > r'$ that $f(r) < f(r')$, then the assumptions for r' imply the assumptions for r (by downward closure), which in turn imply the results for r (by the theorem, if true), which further implies the results for r' (by downward closure). Thus we could have taken $f(r') = f(r)$ instead.)

If we are to prove it in this manner, we'd have to be able to assert that $f(r-1) < f(r)$, since we require to establish that $(c_2, c_2', f(r-1)) \in DW_C$ and we would have to do so by playing a move on the game $(c, c', f(r)) \in DW_C$ (and applying downward closure as appropriate). However, we note that we would have to apply the inductive hypothesis at least twice in order to recombine the

components. Thus, we also require $f(f(r-1)) < f(r)$. Since f is monotone, this implies $f(r-1) < r$. Since f is a function on the natural numbers, we are forced to conclude that $f(r) = r$ for all r.

A further angle is to consider the addition of a "super" move to the games, which is capable of decomposing c_1' into the appropriate components in a single move. This would enable the particular case to be resolved, however, since we suppose that the inductive hypothesis may be applied more than once within that proof, it would be necessary to include the super move as a possible case in the proof. This in turn implies that Duplicator would have to be able to answer a complicated splitting of $c(a)$, with many possible cases. It is likely that yet more super moves would be needed for Duplicator to construct this response, and so on. Therefore, a super move does not appear to offer a solution, but merely obfuscate the essential and remaining issue.

There does, of course, remain the possibility that the conjecture is false. This does not necessarily imply that adjunct elimination fails, although a counterexample to the proof would likely give considerable insight into how a counterexample to adjunct elimination may be constructed.

6.3.3 Multi-holed Contexts

If a counterexample arises from the particular splitting considered here, it is likely that it can be circumvented by introducing contexts with more than one hole. Two-holed contexts could allow the splitting of a_1' from c' to produce a two-holed context into which a' could be inserted to produce a c_1' that is indistinguishable from c_1 in $r-1$ game moves. Due to constraints on time and resources, consideration of multi-holed contexts is beyond the scope of this project.

7 Conclusions and Future Work

The purpose of this project has been to investigate to what extent adjuncts may be eliminated in Context Logic. A key goal was to show (or refute) that adjuncts do not add expressive power to Context Logic for Trees.

We have discussed previous and contemporary related work, and given the key background definitions for Context Logic and Ehrenfeucht-Fraïssé games. We explored various considerations relating to the elimination of adjuncts in general cases, particularly providing specific counterexamples in order build our intuition for adjunct elimination.

We then concentrated our attentions on adjunct elimination in Context Logic for Trees. We provided an explicit counterexample to elimination of the '$\triangleright$' adjunct and a proof of elimination of the '$\triangleleft$' adjunct, satisfying the key goal of the project. We then evaluated these results, looking, for instance, at how we might extend context logic in order that '$\triangleright$' would become eliminable.

While we have answered some interesting and significant questions in this project, we have raised new questions, and indicated what other work remains to be done in this are. For instance:

- Can adjuncts be eliminated in Context Logic for Trees when
  - we add context composition?
  - we use multi-holed contexts?
  - we concern ourselves only with elimination within data formulae?
- Can elimination results be applied to other types of data-structure, or generalised to an abstract data-structure?
- Parametric inexpressivity results for the elimination of adjuncts would complement the work in this project.
- Is it possible to define model conditions that are satisfied by exactly those models which admit adjunct elimination?

Continued work on these and related problems will enhance our understanding of Context Logic, and help to build a sound theoretical framework underpinning the use of the logic in Computer Science.

References

[1] P. Blackburn, M. de Rijke, Y. Venema. *Modal Logic.* Cambridge University Press, 2001.

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

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

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

[5] E. Lozes. Elimination of spatial connectives in static spatial logics. In *TCS 330(3)*, 2005.

[6] J. Reynolds. Separation logic: a logic for shared mutable data structures. Invited Paper, LICS'02, 2002. 29