Revisiting separation: Algorithms and complexity

DANIEL OLIVEIRA[*], *Departamento de Matemática, Instituto Superior Técnico, U Lisboa, Portugal.*

JOÃO RASGA[**], *Departamento de Matemática, Instituto Superior Técnico, U Lisboa, Portugal and Centro de Matemática, Aplicações Fundamentais e Investigação Operacional, U Lisboa, Portugal.*

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Abstract

Linear temporal logic (LTL) with Since and Until modalities is expressively equivalent, over the class of complete linear orders, to a fragment of first-order logic known as FOMLO (first-order monadic logic of order). It turns out that LTL, under some basic assumptions, is expressively complete if and only if it has the property, called separation, that every formula is equivalent to a Boolean combination of formulas that each refer only to the past, present or future. Herein we present simple algorithms and their implementations to perform separation of the LTL with Since and Until, over discrete and complete linear orders, and translation from FOMLO formulas into equivalent temporal logic formulas. We additionally show that the separation of a certain fragment of LTL results in at most a double exponential size growth.

1 Introduction

In 1957, Prior [9] introduced tense logic, by extending propositional logic with two operators P and F, with the intended semantics that $P\varphi$ is true if φ was true at some time in the past, and F meaning the same but in the future direction.

A few years later, Pnueli [8] proposed the use of this logic in the analysis of properties of computer programs, and although Prior's tense logic is strong enough to express many useful properties, there are still properties that are not expressible. Examples of properties of practical interest that cannot be expressed can be found in [4].

The question of how expressive these logics are naturally arose. A way to measure the expressiveness follows by considering a fragment of first-order logic which seems to appropriately describe everything that we ought to be able to say, and then check if all these properties can also be expressed in temporal logic. Or, more precisely, a temporal logic is *expressively complete* if for every first-order formula in this fragment, there is a temporal logic formula that has exactly the same models (and vice versa from temporal logic to the first-order fragment).

In 1968, Kamp [6] introduced two temporal operators Since and Until, with the intended meaning that $A\,\text{Until}\,B$ holds if in the future B will be true and until then A is always true, and similarly for Since but in the past direction. He also defined a fragment of first-order logic now known as FOMLO (first-order monadic logic of order) which appears appropriate in the sense described above, and showed that if we consider a Dedekind-complete linear model of time, then a temporal logic with just these two operators is expressively complete. A shorter proof of this result can be found in [10].

[*] E-mail: daniel.r.oliveira@ist.utl.pt

[**] E-mail: joao.rasga@tecnico.ulisboa.pt

Vol. 29, No. 3, © The Author(s) 2020. Published by Oxford University Press. All rights reserved.
For permissions, please e-mail: journals.permission@oup.com.
Advance Access published 18 February 2020 doi:10.1093/jigpal/jzz081

Still in [4], it is shown that a logic with only the temporal operator Until is expressively complete over the natural numbers, when considering satisfaction at the initial point of time. Or alternatively over complete and discrete linear orders when considering a fragment of FOMLO that restricts quantifiers such that they can only do bounded quantification and never into the past. This result also implies that, at the initial point, having the additional Since operator does not add expressive power. Although it does not add expressive power, it still gives us something, namely succinctness. In [7] it is shown that there are formulas in the language with both operators such that an equivalent formula using only Until is necessarily at least exponentially larger. Even so, nowadays, this language only with Until is quite popular and has found many practical applications, e.g. in the analysis of the correctness of computer programs and systems with concurrent processes. This language is now known as LTL (linear temporal logic).

Dov Gabbay showed that the logic with both Since and Until has the property, called separation, that for every formula there is an equivalent formula consisting of a Boolean combination of formulas that each talk only about the past, present or future. This is mentioned already in [4] and a proof of this over the integers can be found in [2]. A proof over Dedekind-complete linear time was later developed in [3]. This property not only leads to a proof of Kamp's theorem, but it turns out that, over any class of linear orders, a temporal logic has the separation property if and only if it is expressively complete, provided that the temporal logic can at least express Prior's P and F operators. This is shown in [3] and [5].

In this paper we provide simple algorithms that perform separation and translation. We show that they are sound and complete; i.e. given any formula of the temporal language with Since and Until, the separation algorithm constructs a separated formula equivalent to the first over complete and discrete linear time, and given any FOMLO formula, the translation algorithm constructs an equivalent temporal logic formula. In particular we show that these algorithms always terminate and that separation of a certain fragment of temporal logic results in at most doubly exponential size growth.

The translation algorithm is a simplified version of the algorithm that can be extracted from [2, Theorem 2.3]. The same translation algorithm can in fact be used to translate from FOMLO to any expressively complete temporal logic, when provided with a separation algorithm for the logic. We also give an algorithm to translate FOMLO to LTL, which only requires a small addition to the main translation algorithm.

The algorithm that can be extracted from Gabbay's proof of separation over integer time is of a different nature. It works by using a family of algorithms that can separate a restricted class of formulas, with the last one being able to perform separation on every formula. Each algorithm considers the whole formula, identifying the most 'unseparated' subformulas and substituing certain subformulas with atoms so that the formula as a whole is of a form that can be processed with the previous algorithm. Then it substitutes the previously introduced atoms back with the subformulas they had replaced, and continues recursively. We instead opt for a simple unified recursive algorithm to perform separation. We also simplify some of the basic steps in the separation process and provide detailed proofs after introducing some concepts and notation that make it easier to do so.

All of the algorithms we describe have been implemented in the Haskell programming language (https://github.com/drdo/logic-translation).

1.1 Outline

Section 2 is an introduction to temporal logic and FOMLO. We define their syntax and semantics and some notation we will be using.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

In Section 3 we concern ourselves with separation of temporal logic. It begins by proving some results about temporal logic, then in Section 3.1 the algorithm is fully defined, with Section 3.2 being devoted to proving its correctness. Section 3.3 shows a double exponential upper bound on the size blowup of separation for a certain fragment of temporal logic.

In Section 4 we turn to the problem of translating from FOMLO to temporal logic. We define the algorithm in Section 4.1 and prove its correctness in Section 4.2. Section 4.3 shows how to use the algorithm to perform the translation for other temporal logics.

2 Temporal logic and FOMLO

In this section we introduce the logics, some notation, and show how to translate from temporal logic to FOMLO. For the remainder of this text, unless explicitly mentioned, when we say 'temporal logic' or 'TL', we mean the temporal logic defined in this section.

2.1 Syntax

Fix a countable set Pred that will serve as both the propositional variables in temporal logic and monadic predicates in FOMLO.

Definition 2.1
The language of temporal logic $\mathcal{L}_{\text{TL}}$ is defined inductively by

- $\bot \in \mathcal{L}_{\text{TL}}$
- $\top \in \mathcal{L}_{\text{TL}}$
- $\neg A \in \mathcal{L}_{\text{TL}} \qquad (A \in \mathcal{L}_{\text{TL}})$
- $A \vee B \in \mathcal{L}_{\text{TL}} \qquad (A, B \in \mathcal{L}_{\text{TL}})$
- $A \wedge B \in \mathcal{L}_{\text{TL}} \qquad (A, B \in \mathcal{L}_{\text{TL}})$
- $p \in \mathcal{L}_{\text{TL}} \qquad (p \in \text{Pred})$
- $A\mathcal{S}B \in \mathcal{L}_{\text{TL}} \qquad (A, B \in \mathcal{L}_{\text{TL}})$
- $A\mathcal{U}B \in \mathcal{L}_{\text{TL}} \qquad (A, B \in \mathcal{L}_{\text{TL}})$

The connectives are ordered by binding strength as follows: $\neg\ \mathcal{S},\mathcal{U}\ >\ \vee,\wedge$.

Definition 2.2
The set $\mathcal{L}_{\text{LTL}}$ of LTL formulas is defined just like the one of TL, but omitting the $\mathcal{S}$ case.

Definition 2.3
Some additional temporal operators:

- $\bullet A := \bot\mathcal{S}A$ (Previous)
- $\blacklozenge A := \top\mathcal{S}A$ (Eventually in the past)
- $\blacksquare A := \neg\blacklozenge\neg A$ (Forever in the past)
- $\bigcirc A := \bot\mathcal{U}A$ (Next)
- $\Diamond A := \top\mathcal{U}A$ (Eventually)
- $\Box A := \neg\Diamond\neg A$ (Forever)

Definition 2.4
We say a formula A is *simple* if it has no outer Boolean structure, i.e. if it is of the form p, $B\mathcal{S}C$ or $B\mathcal{U}C$ (for some $p \in \text{Pred}$ and $B, C \in \mathcal{L}_{\text{TL}}$).

Definition 2.5
A formula is called *non-future* if it has no occurrences of $\mathcal{U}$ and *non-past* if it has no occurrences of $\mathcal{S}$.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

A *pure past* formula is then a Boolean combination of formulas of the form $A\mathcal{S}B$ where both A and B are non-future and similarly a formula is *pure future* if it is a Boolean combination of formulas of the form $A\mathcal{U}B$ with A and B non-past.

A formula is *pure present* if it is a Boolean combination of variables, $\bot$ and $\top$.

Definition 2.6
A formula is *separated* if it is a Boolean combination of pure formulas.

For FOMLO, we also need a set of variables, we call it Var and assume it is countable and infinite.

Definition 2.7
The set $\mathcal{L}_{\text{FOMLO}}$ of FOMLO formulas is defined inductively by

- $\bot \in \mathcal{L}_{\text{FOMLO}}$
- $\top \in \mathcal{L}_{\text{FOMLO}}$
- $\neg\alpha \in \mathcal{L}_{\text{FOMLO}}$ $\quad(\alpha \in \mathcal{L}_{\text{FOMLO}})$
- $\alpha \vee \beta \in \mathcal{L}_{\text{FOMLO}}$ $\quad(\alpha, \beta \in \mathcal{L}_{\text{FOMLO}})$
- $\alpha \wedge \beta \in \mathcal{L}_{\text{FOMLO}}$ $\quad(\alpha, \beta \in \mathcal{L}_{\text{FOMLO}})$
- $P(x) \in \mathcal{L}_{\text{FOMLO}}$ $\quad(P \in \text{Pred},\ x \in \text{Var})$
- $x = y \in \mathcal{L}_{\text{FOMLO}}$ $\quad(x, y \in \text{Var})$
- $x < y \in \mathcal{L}_{\text{FOMLO}}$ $\quad(x, y \in \text{Var})$
- $\exists_x\alpha \in \mathcal{L}_{\text{FOMLO}}$ $\quad(x \in \text{Var}, \alpha \in \mathcal{L}_{\text{FOMLO}})$
- $\forall_x\alpha \in \mathcal{L}_{\text{FOMLO}}$ $\quad(x \in \text{Var}, \alpha \in \mathcal{L}_{\text{FOMLO}})$

In both TL and FOMLO, we additionally define $\varphi \rightarrow \psi$ as an abbreviation for $\neg\varphi \vee \psi$.

We use lower case letters from the latin alphabet to refer to elements of Pred in the context of temporal logic, and upper case latin letters to refer to these same elements in the context of FOMLO. Additionally, each letter refers to the same predicate symbol in both the lower case and upper case versions, e.g. p and P refer to the same predicate symbol.

We write Subs(A) to denote the set of subformulas of A.

2.2 Semantics

We will consider interpretation structures over a signature with a binary predicate $<$ and countable unary predicates $P \in$ Pred, with the interpretation of $<$ a *complete* and *discrete* linear order. Where *complete* means that every non-empty bounded above subset has a supremum and every non-empty bounded below subset has an infimum, and *discrete* means that every non-maximal element has an immediate successor and every non-minimal element has an immediate predecessor. These conditions also imply that every infimum is in fact a minimum element of the subset, as if the infimum was not a minimum element, then its immediate successor would be a larger lower bound for the subset. A similar reasoning also allows us to conclude that every non-empty bounded above subset has a greatest element.

Definition 2.8
An interpretation structure I is a tuple $I = \left\langle D, <^I, \left\{P^I\right\}_{P \in \text{Pred}}\right\rangle$ where D is a non-empty set called the domain of I, $<^I$ is a complete and discrete linear order over D and each $P^I \subseteq D$.

We write Domain(I) to refer to this D.

In temporal logic, we use the notation $I, (t,s) \Vdash A$ to mean that all the points in the interval (t,s) satisfy A. That is, for all $r^{>t}_{<s}$: $I, r \Vdash A$. And similarly for $[t,s)$, $(t,s]$ and $[t,s]$, which mean $r^{\geq t}_{<s}$, $r^{>t}_{\leq s}$ and $r^{\geq t}_{\leq s}$, respectively. When we use $\nVdash$ with this interval notation, we mean that *no* point in the interval satisfies the formula; it does not mean that some point in the interval fails to satisfy.

This use of $<$ should in fact have been $<^I$. We will continue to write $<$ to refer to $<^I$ when there is no risk of confusion. We might also write $t \in I$ to mean $t \in$ Domain(I).

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Definition 2.9

Let I be an interpretation structure and $t \in I$. Then we define satisfaction of a temporal logic formula by the structure I at the point t by

- $I, t \nVdash_{TL} \bot$
- $I, t \Vdash_{TL} \top$
- $I, t \Vdash_{TL} p$ if and only if $t \in P^I$
- $I, t \Vdash_{TL} \neg A$ if and only if $I, t \nVdash_{TL} A$
- $I, t \Vdash_{TL} A \vee B$ if and only if $I, t \Vdash_{TL} A$ or $I, t \Vdash_{TL} B$
- $I, t \Vdash_{TL} A \wedge B$ if and only if $I, t \Vdash_{TL} A$ and $I, t \Vdash_{TL} B$
- $I, t \Vdash_{TL} A\mathcal{S}B$ if and only if there is an$s <^I t$ such that $\begin{array}{l} I, s \Vdash_{TL} B \\ I, r \Vdash_{TL} A \text{ for all } r \in (s,t) \end{array}$
- $I, t \Vdash_{TL} A\mathcal{U}B$ if and only if there is an$s >^I t$ such that $\begin{array}{l} I, s \Vdash_{TL} B \\ I, r \Vdash_{TL} A \text{ for all } r \in (t,s) \end{array}$

Naturally, in the last two cases, s and r are over the domain of I.

We call the s in the definition of $\mathcal{S}$ ($\mathcal{U}$) satisfaction a *witness* for $A\mathcal{S}B$ ($A\mathcal{U}B$) at t.

Satisfaction for FOMLO is defined in the usual way.

Definition 2.10

An assignment ρ into an interpretation structure I is a function $\rho : \text{Var} \to \text{Domain}(I)$.

We use the notation $[x \mapsto x_0, y \mapsto y_0]$ to denote an assignment ρ such that $\rho(x) = x_0$ and $\rho(y) = y_0$. And the notation $\rho[x \mapsto x_0, y \mapsto y_0]$ for an assignment identical to ρ except possibly at x and y, to which it assigns x_0 and y_0, respectively.

Definition 2.11

Let I be an interpretation structure and ρ an assignment into I, then

- $I, \rho \nVdash_{FOMLO} \bot$
- $I, \rho \Vdash_{FOMLO} \top$
- $I, \rho \Vdash_{FOMLO} P(x)$ if and only if $\rho(x) \in P^I$
- $I, \rho \Vdash_{FOMLO} (x = y)$ if and only if $\rho(x) = \rho(y)$
- $I, \rho \Vdash_{FOMLO} (x < y)$ if and only if $\rho(x) <^I \rho(y)$
- $I, \rho \Vdash_{FOMLO} \neg\alpha$ if and only if $I, \rho \nVdash_{FOMLO} \alpha$
- $I, \rho \Vdash_{FOMLO} \alpha \vee \beta$ if and only if $I, \rho \Vdash_{FOMLO} \alpha$ or $I, \rho \Vdash_{FOMLO} \beta$
- $I, \rho \Vdash_{FOMLO} \alpha \wedge \beta$ if and only if $I, \rho \Vdash_{FOMLO} \alpha$ and $I, \rho \Vdash_{FOMLO} \beta$
- $I, \rho \Vdash_{FOMLO} \exists_x \alpha$ iff there is an assignment ρ' identical to ρ except possibly at x such that $I, \rho' \Vdash_{FOMLO} \alpha$
- $I, \rho \Vdash_{FOMLO} \forall_x \alpha$ iff for all assignments ρ' identical to ρ except possibly at x: $I, \rho' \Vdash_{FOMLO} \alpha$

Definition 2.12

In both TL and FOMLO, we may simply write $\Vdash$ when it is clear which logic we mean. And in TL we may write $t \Vdash A$ instead of $I, t \Vdash A$ when it is clear from the context which interpretation we are considering.

We use $\equiv$ to denote semantic equivalence of formulas. That is, in the case of temporal logic, $A \equiv B$ means that for all interpretations I and points t of I: $I, t \Vdash A$ if and only if $I, t \Vdash B$. And in the case of FOMLO, $\varphi \equiv \psi$ means that for all interpretations I and assignments ρ: $I, \rho \Vdash \varphi$ if and only if $I, \rho \Vdash \psi$.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

3 Separation

In this section we define and prove the correctness of an algorithm that separates a temporal logic formula. But before we show the algorithm, we prove some crucial properties of temporal logic.

DEFINITION 3.1
The *dual* of a formula is a formula obtained by replacing every $\mathcal{S}$ with a $\mathcal{U}$, and vice versa, in the given formula. We write $\mathrm{Dual}(A)$ for the dual of A.

We now show that this Dual has some expected properties, which allows us to only worry about $\mathcal{U}$ inside $\mathcal{S}$ and get the cases where $\mathcal{S}$ is inside $\mathcal{U}$ 'for free'.

DEFINITION 3.2
Given an interpretation structure I, I^{op} is called the opposite structure. It is defined just like I except that the interpretation of $<$ is precisely the opposite order of the one of I. That is,

$$\mathrm{Domain}(I^{\mathrm{op}}) = \mathrm{Domain}(I)$$

$$P^{I^{\mathrm{op}}} = P^{I} \quad (\text{for every } P \in \mathrm{Pred})$$

$$<^{I^{\mathrm{op}}} = \left\{ \langle y, x \rangle \mid \langle x, y \rangle \in <^{I} \right\}$$

PROPOSITION 3.3
For every formula A: $\mathrm{Dual}(\mathrm{Dual}(A)) = A$.

PROOF. It's clear that swapping $\mathcal{S}$ and $\mathcal{U}$ twice results in the same formula. □

PROPOSITION 3.4
Let I be an interpretation structure, t a point of I and A a formula. Then

$$I, t \Vdash A \text{ if and only if } I^{\mathrm{op}}, t \Vdash \mathrm{Dual}(A).$$

PROOF. Straightforward induction on the structure of A. □

PROPOSITION 3.5
For all $A, B \in \mathcal{L}_{\mathrm{TL}}$: $A \equiv B$ if and only if $\mathrm{Dual}(A) \equiv \mathrm{Dual}(B)$.

PROOF. We first show that $A \equiv B$ implies $\mathrm{Dual}(A) \equiv \mathrm{Dual}(B)$:

$$\begin{aligned}
& I, t \Vdash \mathrm{Dual}(A) & \\
\text{iff} \quad & I^{\mathrm{op}}, t \Vdash \mathrm{Dual}(\mathrm{Dual}(A)) = A & \text{(Propositions 3.4 and 3.3)} \\
\text{iff} \quad & I^{\mathrm{op}}, t \Vdash B & \text{(by assumption)} \\
\text{iff} \quad & \left(I^{\mathrm{op}}\right)^{\mathrm{op}} = I, t \Vdash \mathrm{Dual}(B) & \text{(Proposition 3.4)}
\end{aligned}$$

For the other direction, using the result just above and Proposition 3.3,

$$\mathrm{Dual}(A) \equiv \mathrm{Dual}(B) \implies \mathrm{Dual}(\mathrm{Dual}(A)) \equiv \mathrm{Dual}(\mathrm{Dual}(B)) \iff A \equiv B.$$

□

Thus far we have seen that dual formulas really are dual in the expected sense.

We now prove two important distribution results, namely that $\mathcal{S}$ distributes over $\wedge$ on the right and over $\vee$ on the left. Naturally, by duality, $\mathcal{U}$ also distributes over $\wedge$ on the right and $\vee$ on the left.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

PROPOSITION 3.6
For all $A, B, C \in \mathcal{L}_{\mathrm{TL}}$: $(A \wedge B)\mathcal{S}C \equiv A\mathcal{S}C \wedge B\mathcal{S}C$.

PROOF.
($\Rightarrow$) Let $t \Vdash (A \wedge B)\mathcal{S}C$.
Then there is a $t_C < t$ such that $t_C \Vdash C$, $(t_C, t) \Vdash A$ and $(t_C, t) \Vdash B$.
This same t_C is also a witness for both $A\mathcal{S}C$ and $B\mathcal{S}C$ at t.
($\Leftarrow$) Let $t \Vdash A\mathcal{S}C \wedge B\mathcal{S}C$
Then there are $t_C, s_C < t$ that satisfy C and $(t_C, t) \Vdash A$ and $(s_C, t) \Vdash B$.
Let r be the greatest of t_C and s_C.
Then $r \Vdash C$. And $(r, t) \subseteq (t_C, t) \cap (s_C, t)$, since $t_C, s_C \leq r$, so $(r, t) \Vdash A \wedge B$.
This means that r is a witness for $(A \wedge B)\mathcal{S}C$ at t. □

COROLLARY 3.7 ($\mathcal{S}$ right-distributes over $\wedge$).
For all $A_1, \ldots, A_n, B \in \mathcal{L}_{\mathrm{TL}}$:

$$\left(\bigwedge_{i=1}^{n} A_i\right)\mathcal{S}B \equiv \bigwedge_{i=1}^{n} (A_i\mathcal{S}B)\,.$$

PROOF. Induction on n using Proposition 3.6. □

COROLLARY 3.8 ($\mathcal{U}$ right-distributes over $\wedge$).
For all $A_1, \ldots, A_n, B \in \mathcal{L}_{\mathrm{TL}}$:

$$\left(\bigwedge_{i=1}^{n} A_i\right)\mathcal{U}B \equiv \bigwedge_{i=1}^{n} (A_i\mathcal{U}B)\,.$$

PROOF. Duality, replace A and B with variables, use Proposition 3.5 with Corollary 3.7 and replace back A and B. □

PROPOSITION 3.9
For all $A, B, C \in \mathcal{L}_{\mathrm{TL}}$: $A\mathcal{S}(B \vee C) \equiv A\mathcal{S}B \vee A\mathcal{S}C$.

PROOF.
($\Rightarrow$) Let $t \Vdash A\mathcal{S}(B \vee C)$.
Then there is a $t_T < t$ such that $t_T \Vdash B \vee C$ and $(t_T, t) \Vdash A$.
If t_T satisfies B then $t \Vdash A\mathcal{S}B$, similarly if t_T satisfies C then $t \Vdash A\mathcal{S}C$.
($\Leftarrow$) Let $t \Vdash A\mathcal{S}B \vee A\mathcal{S}C$.
There's either a t_B such that $t_B \Vdash B$ and $(t_B, t) \Vdash A$, or a t_C such that $t_C \Vdash C$ and $(t_C, t) \Vdash A$.
Either way, both t_B and t_C satisfy $B \vee C$ as well and are witnesses for $A\mathcal{S}(B \vee C)$ at t. □

COROLLARY 3.10 ($\mathcal{S}$ left-distributes over $\vee$).
For all $A, B_1, \ldots, B_n \in \mathcal{L}_{\mathrm{TL}}$:

$$A\mathcal{S}\left(\bigvee_{i=1}^{n} B_i\right) \equiv \bigvee_{i=1}^{n} (A\mathcal{S}B_i).$$

PROOF. Induction on n using Proposition 3.9. □

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

COROLLARY 3.11 ($\mathcal{U}$ left-distributes over $\vee$).
For all $A, B_1, \ldots, B_n \in \mathcal{L}_{\mathrm{TL}}$:

$$A\mathcal{U}\left(\bigvee_{i=1}^{n} B_i\right) \equiv \bigvee_{i=1}^{n} (A\mathcal{U}B_i).$$

PROOF. Duality with Corollary 3.10. □

3.1 Algorithm

The algorithm works by recursively separating the immediate subformulas, which is all that is required for a Boolean operator. Instead of worrying about both $\mathcal{S}$ and $\mathcal{U}$, the $\mathcal{U}$ case is handled by duality and the bulk of the algorithm is on how to separate a $\mathcal{S}$.

When trying to separate a $\mathcal{S}$, the distribution results we proved earlier help in the following way: consider a formula $A\mathcal{S}B$ where A and B are already separated. A and B can be arbitrary Boolean combinations of simple pure formulas. If we convert the outer Boolean structure of A to conjunctive normal form and the one of B to disjunctive normal form, we can then use distribution to 'split' $A\mathcal{S}B$ into a Boolean combination of formulas of the form $C\mathcal{S}D$, where the outer Boolean structures are much simpler, namely C is a disjunctive clause of simple pure formulas and D is a conjunctive clause of simple pure formulas. We then only have to worry about 'pulling' the simple pure $\mathcal{U}$s from inside the $\mathcal{S}$ in this narrower case where the left side is a disjunctive clause and the right side is a conjunctive clause. We then eliminate one $\mathcal{U}$ from inside the $\mathcal{S}$, removing it from both sides at once. There are eight possibilities for each; it can occur only on the left, or only on the right, or in both, and it can appear negated or not. Having pulled out one of the $\mathcal{U}$s, we continue separating the result recursively.

Before we show the algorithm, we introduce the concept of path and paths of a formula. This is necessary later for the proof of correctness, but also allows us to define the algorithm a bit more conveniently, so we introduce it now.

DEFINITION 3.12
A *path* is a finite sequence over $\{\mathcal{S},\mathcal{U}\}$. We use $\langle\rangle$ to denote the empty path.

To avoid confusion, we call a path (in the usual sense of graph theory) on the syntax tree of a formula, a *syntax path*. And we call the path corresponding to a syntax path the sequence obtained by considering the labels on the syntax path and then deleting the labels outside of $\{\mathcal{S},\mathcal{U}\}$.

For example, take $A = p\mathcal{S}q \wedge \top\mathcal{U}(\bot\mathcal{S}r)$. It has the following syntax tree:

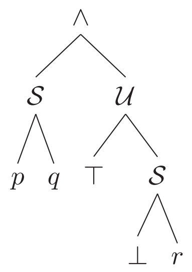

FIGURE 1 Syntax tree of A.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

In this example, the path corresponding to the syntax path that starts from the root and goes left twice is $\mathcal{S}$, and the one from the root to the right twice and then left is $\mathcal{US}$.

This leads to the definition of the paths of a formula.

Definition 3.13
Given a formula A, Paths(A) is the set of paths corresponding with the syntax paths from the root of the syntax tree of A to a leaf, except for the empty path.

The following definition, used in the algorithm, is why we needed to define Paths beforehand.

Definition 3.14
The *temporal depth* of a formula A is the maximum length of a path of A.

We can finally present the algorithm. The eight transformations ($\mathcal{T}_1$, $\mathcal{T}_2$, etc.) are defined afterwards.

When we write $(\neg ?)$ in the algorithm, we mean to allow the possibility of a negation occurring there.

Definition 3.15
We assume the following regarding normal forms and normal form conversions in the context of the Sep algorithm:

- Non-future formulas and simple pure future formulas are considered atoms.
- Each clause in a normal form has no repeated literals and no complementary literals.
- Normal form conversion is done in the conventional way, by first converting to negation normal form and then distributing $\vee$ over $\wedge$ or vice versa.

```
procedure Sep($X$)
    $^0$if $X$ is separated then return $X$
    $^1$else if $X = \neg A$ then return $\neg\mathrm{Sep}(A)$
    $^2$else if $X = A \vee B$ then return $\mathrm{Sep}(A) \vee \mathrm{Sep}(B)$
    $^3$else if $X = A \wedge B$ then return $\mathrm{Sep}(A) \wedge \mathrm{Sep}(B)$
    $^4$else if $X = A\mathcal{S}B$ with $A$ a separated disjunctive clause, and $B$ a separated conjunctive clause
        let $\bigvee_{i=1}^{m} (\neg ?)A_i := A$
        let $\bigwedge_{j=1}^{n} (\neg ?)B_j := B$
        let $\mathbf{C} := \{(\neg ?)A_i \mid A_i \text{ is a pure future formula}, i \in [1,m]\}$
        let $\mathbf{A} := \{(\neg ?)A_i \mid i \in [1,m]\} - \mathbf{C}$
        let $\mathbf{D} := \{(\neg ?)B_j \mid B_j \text{ is a pure future formula}, j \in [1,n]\}$
        let $\mathbf{B} := \{(\neg ?)B_j \mid j \in [1,n]\} - \mathbf{D}$                    $(X \equiv (\bigvee \mathbf{A} \vee \bigvee \mathbf{C})\mathcal{S}(\bigwedge \mathbf{B} \wedge \bigwedge \mathbf{D}))$
        let $(\neg ?)(F\mathcal{U}G) \in \mathbf{C} \cup \mathbf{D}$ with maximal temporal depth in $\mathbf{C} \cup \mathbf{D}$
        let $A' := \bigvee((\mathbf{A} \cup \mathbf{C}) - \{F\mathcal{U}G, \neg(F\mathcal{U}G)\})$
        let $B' := \bigwedge((\mathbf{B} \cup \mathbf{D}) - \{F\mathcal{U}G, \neg(F\mathcal{U}G)\})$
        $^{4.1}$if $F\mathcal{U}G \in \mathbf{C}$ and $F\mathcal{U}G, \neg(F\mathcal{U}G) \notin \mathbf{D}$                    $(X \equiv (A' \vee F\mathcal{U}G)\mathcal{S}B')$
            return $\mathrm{Sep}(\mathcal{T}_1(A', B', F, G))$
```

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

```
    4.2 else if $F\mathcal{U}G, \neg(F\mathcal{U}G) \notin \mathbf{C}$ and $F\mathcal{U}G \in \mathbf{D}$        $(X \equiv A'\mathcal{S}(B' \wedge F\mathcal{U}G))$
        return Sep($\mathcal{T}_2\,(A', B', F, G, )$)
    4.3 else if $F\mathcal{U}G \in \mathbf{C}$ and $F\mathcal{U}G \in \mathbf{D}$        $(X \equiv (A' \vee F\mathcal{U}G)\mathcal{S}(B' \wedge F\mathcal{U}G))$
        return Sep($\mathcal{T}_3\,(A', B', F, G)$)
    4.4 else if $\neg(F\mathcal{U}G) \in \mathbf{C}$ and $F\mathcal{U}G, \neg(F\mathcal{U}G) \notin \mathbf{D}$        $(X \equiv (A' \vee \neg(F\mathcal{U}G))\mathcal{S}B')$
        return Sep($\mathcal{T}_4\,(A', B', F, G)$)
    4.5 else if $F\mathcal{U}G, \neg(F\mathcal{U}G) \notin \mathbf{C}$ and $\neg(F\mathcal{U}G) \in \mathbf{D}$        $(X \equiv A'\mathcal{S}(B' \wedge \neg(F\mathcal{U}G)))$
        return Sep($\mathcal{T}_5\,(A', B', F, G, )$)
    4.6 else if $F\mathcal{U}G \in \mathbf{C}$ and $\neg(F\mathcal{U}G) \in \mathbf{D}$        $(X \equiv (A' \vee F\mathcal{U}G)\mathcal{S}(B' \wedge \neg(F\mathcal{U}G)))$
        return Sep($\mathcal{T}_6\,(A', B', F, G, )$)
    4.7 else if $\neg(F\mathcal{U}G) \in \mathbf{C}$ and $\neg(F\mathcal{U}G) \in \mathbf{D}$  $(X \equiv (A' \vee \neg(F\mathcal{U}G))\mathcal{S}(B' \wedge \neg(F\mathcal{U}G)))$
        return Sep($\mathcal{T}_7\,(A', B', F, G, )$)
    4.8 else if $\neg(F\mathcal{U}G) \in \mathbf{C}$ and $F\mathcal{U}G \in \mathbf{D}$        $(X \equiv (A' \vee \neg(F\mathcal{U}G))\mathcal{S}(B' \wedge F\mathcal{U}G))$
        return Sep($\mathcal{T}_8\,(A', B', F, G, )$)
5 else if $X = A\mathcal{S}B$
    let $\bigwedge_{i=1}^{m} A_i$ be a CNF of Sep($A$)
    let $\bigvee_{j=1}^{n} B_j$ be a DNF of Sep($B$)
    if $m = 0$ then set $m$ to 1 and $A_1$ to $\top$
    if $n = 0$ then set $n$ to 1 and $B_1$ to $\bot$
    return $\bigvee_{j=1}^{n} \bigwedge_{i=1}^{m}$ Sep($A_i\mathcal{S}B_j$)
6 else if $X = A\mathcal{U}B$ then return Dual(Sep(Dual($X$)))
```

In case 4 what we are doing is essentially picking one of the ‘hardest’ cases and choosing to eliminate that one first.

3.1.1 Transformations To make things easier to follow, we will be using diagrams such as the following:

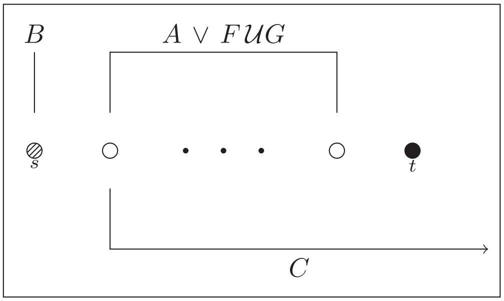

FIGURE 2 Example diagram.

where ⊘, ○ and ● represent points in the domain of the interpretation, with ● referring to a specific point, ⊘ to the existence of a point and ○ to a point that may or may not exist.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

The order of the points in the diagram, from left to right, reflects the intended relationship between them in the interpretation's order. And if a point is next to another, it means there is no other point between them. To represent an arbitrary number of points, we use • • •.

Figure 2 would mean that there is a point $s < t$, s satisfies B, (s,t) satisfies $A \vee F\mathcal{U}G$ and all points greater than s satisfy C.

PROPOSITION 3.16
Let $A, B \in \mathcal{L}_{\mathrm{TL}}$, t a point and $t \Vdash A\mathcal{S}B$.

Then there is a witness t_B for $A\mathcal{S}B$ at t such that $(t_B, t) \Vdash \neg B$, and we call such a t_B the *nearest witness*.

PROOF. Let $W = \{s < t \mid s \Vdash B \text{ and} (s,t) \Vdash A\}$ be the set of witnesses. W is non-empty because $t \Vdash A\mathcal{S}B$, and is bounded above by t. So let t_B be the greatest element of W.

If any $s' \in (t_B, t)$ satisfied B, then s' would be a witness for $A\mathcal{S}B$ at t greater than t_B, but this contradicts the fact that t_B is the greatest element of W, so $(t_B, t) \Vdash \neg B$. □

DEFINITION 3.17
Given a point t in a discrete order, $\mathrm{Suc}(t)$ is the immediate successor of t and $\mathrm{Pred}(t)$ is the immediate predecessor of t.

We are indeed overloading the use Pred for the immediate predecessor as well as the set of propositions, but we will do so as confusion is unlikely.

PROPOSITION 3.18
For all $A, B \in \mathcal{L}_{\mathrm{TL}}$: $A\mathcal{U}B \equiv \bigcirc(B \vee (A \wedge A\mathcal{U}B))$.

PROOF. $(\Rightarrow)$ Let $t \Vdash A\mathcal{U}B$ with t_B as witness. If $t_B = \mathrm{Suc}(t)$ then $\mathrm{Suc}(t) \Vdash B$ which is equivalent to $t \Vdash \bigcirc B$. Otherwise $\mathrm{Suc}(t) \in (t, t_B)$, and so $\mathrm{Suc}(t) \Vdash A$ and also $\mathrm{Suc}(t) \Vdash A\mathcal{U}B$ (with t_B as witness).

$(\Leftarrow)$ If $\mathrm{Suc}(t) \Vdash B$ then $\mathrm{Suc}(t)$ is a witness for $t \Vdash A\mathcal{U}B$. If $\mathrm{Suc}(t) \Vdash A$ and $\mathrm{Suc}(t) \Vdash A\mathcal{U}B$ with t_B as witness, then t_B is also a witness for $A\mathcal{U}B$ at t, as $(t, t_B) = [\mathrm{Suc}(t), t_B) \Vdash A$. □

PROPOSITION 3.19
Let $\mathcal{T}_1\,(A,B,F,G) := P \wedge P'\mathcal{S}B$, where

$$N := (\neg G \wedge \neg B)\mathcal{S}(\neg A \wedge \neg B)$$

$$P' := N \;\rightarrow\; G \vee F$$

$$P := N \;\rightarrow\; G \vee (F \wedge F\mathcal{U}G).$$

Then $(A \vee F\mathcal{U}G)\mathcal{S}B \equiv \mathcal{T}_1\,(A,B,F,G)$.

In this case, if t_0 satisfies $(A \vee F\mathcal{U}G)\mathcal{S}B$, then there is some $t_B < t_0$ with $t_B \Vdash B$ and $(t_B, t_0) \Vdash A \vee F\mathcal{U}G$. The difficulty here is finding a way to ensure all the points in (t_B, t_0) satisfy $F\mathcal{U}G$ if they don't happen to satisfy A, but without explicitly using $\mathcal{U}$.

For some point to satisfy $F\mathcal{U}G$, there needs to be an uninterrupted chain of Fs followed by a G. When a point $s \in (t_B, t_0)$ satisfies $N = (\neg G \wedge \neg B)\mathcal{S}(\neg A \wedge \neg B)$, this intuitively means that there is some point in (t_B, s) that did not satisfy A and so needs to satisfy $F\mathcal{U}G$, but there has been no witness so far since $\neg G$ has been true.

Our point s then needs to ensure that this happens by either being a witness itself or continuing the chain of Fs; this is what $P' = N \;\rightarrow\; G \vee F$ means. Finally at t_0, if there is still no witness, t_0

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

itself must either be a witness or ensure that one exists in the future; this is done with $P = N \rightarrow G \vee (F \wedge F\mathcal{U}G)$.

PROOF. ($\Rightarrow$) Let t_0 satisfy $(A \vee F\mathcal{U}G)\mathcal{S}B$ with a witness t_B. We show that $t_0 \Vdash P$ and $(t_B, t_0) \Vdash P'$, which implies that t_B is a witness for $P'\mathcal{S}B$ at t_0.

To do this, we first notice that $\vDash P \rightarrow P'$, and prove the stronger result $(t_B, t_0] \Vdash P$. So, for this purpose, let $s \in (t_B, t_0]$ with $s \Vdash N = (\neg G \wedge \neg B)\mathcal{S}(\neg A \wedge \neg B)$, and let $s_{\neg A}$ be a witness for this fact. $s_{\neg A}$ cannot be before t_B, as that would imply that $t_B \in (s_{\neg A}, s)$ and so would have to satisfy $\neg B$, which is a contradiction. And it cannot be equal to t_B, for the same reason. Hence, $s_{\neg A}$ occurs between t_B and s. This means that $s_{\neg A}$ must satisfy $F\mathcal{U}G$ as it does not satisfy A. A diagram illustrating the situation:

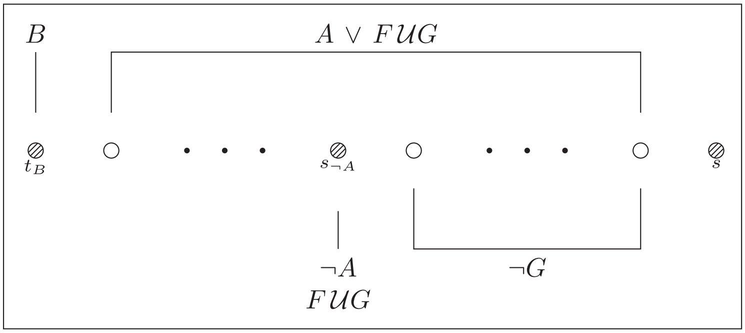

Figure 3 $s \Vdash N$ with $s_{\neg A}$ as witness.

Since $s_{\neg A}$ satisfies $F\mathcal{U}G$, there must be a witness $w > s_{\neg A}$ of this fact, and $w \geq s$ since no point between $s_{\neg A}$ and s satisfies G. If $w = s$ then $s \Vdash G$. If $w > s$ then s is between $s_{\neg A}$ and w and so satisfies F, and w is also a witness for $F\mathcal{U}G$ at s.

($\Leftarrow$) Let $t_0 \Vdash P \wedge P'\mathcal{S}B$ with t_B the nearest witness for $P'\mathcal{S}B$. If we show that $(t_B, t_0) \Vdash A \vee F\mathcal{U}G$ then t_B is also a witness for $(A \vee F\mathcal{U}G)\mathcal{S}B$ at t_0.

We now show that P is satisfied in the whole of $(t_B, t_0]$ by induction on the order $t_0 \rightarrow \mathrm{Pred}(t_0) \rightarrow \mathrm{Pred}^2(t_0) \rightarrow \cdots$. The base case is trivial; we already know $t_0 \Vdash P$. For the inductive step, let $\mathrm{Pred}(s) \in (t_B, t_0)$ and assume $\mathrm{Pred}(s) \Vdash N = (\neg G \wedge \neg B)\mathcal{S}(\neg A \wedge \neg B)$ with r as witness. Remember that we have $s \Vdash P$ by the induction hypothesis. Diagram:

If $\mathrm{Pred}(s) \Vdash G$ then also $\mathrm{Pred}(s) \Vdash G \vee (F \wedge F\mathcal{U}G)$ and we are done.

If $\mathrm{Pred}(s) \Vdash \neg G$, then we have $\mathrm{Pred}(s) \Vdash F$ via P', and remember that $\mathrm{Pred}(s) \in (t_B, t_0)$, which means that $\mathrm{Pred}(s) \Vdash \neg B$ since t_B is the nearest witness for $P'\mathcal{S}B$ at t_0. This gives us $\mathrm{Pred}(s) \Vdash \neg G \wedge \neg B$ and so r is also a witness for N at s, which gives us $s \Vdash G \vee (F \wedge F\mathcal{U}G)$ via P. Given this, by Proposition 3.17, $\mathrm{Pred}(s) \Vdash F\mathcal{U}G$ as well.

We have thus showed that $(t_B, t_0] \Vdash P$.

Now consider $s_{\neg A} \in (t_B, t_0)$ such that $s_{\neg A} \Vdash \neg A$. Again because t_B is the nearest witness, $s_{\neg A} \Vdash \neg A \wedge \neg B$ and so $s_{\neg A}$ is a witness for N at $\mathrm{Suc}(s_{\neg A}) \in (t_B, t_0]$. Therefore, $\mathrm{Suc}(s_{\neg A}) \Vdash G \vee (F \wedge F\mathcal{U}G)$ via P, and $s_{\neg A} \Vdash F\mathcal{U}G$ by Proposition 3.17. □

PROPOSITION 3.20
Let $\mathcal{T}_2(A, B, F, G) := H_< \vee H_= \vee H_>$, where

$$H_< := A\mathcal{S}(G \wedge A \wedge (A \wedge F)\mathcal{S}B)$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

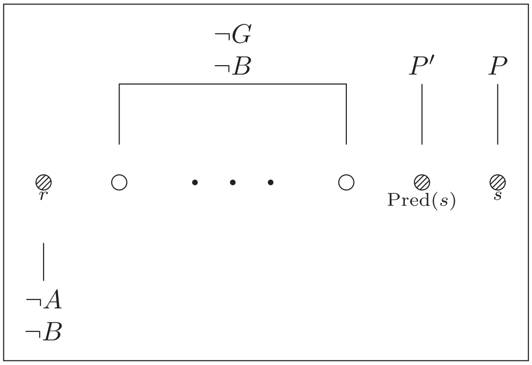

FIGURE 4 Pred$(s) \Vdash N$ with r as witness.

$$H_= := (A \wedge F)\mathcal{S}B \wedge G$$

$$H_> := (A \wedge F)\mathcal{S}B \wedge F \wedge F\mathcal{U}G$$

Then $A\mathcal{S}(B \wedge F\mathcal{U}G) \equiv \mathcal{T}_2(A,B,F,G)$.

PROOF. The satisfaction of $A\mathcal{S}(B \wedge F\mathcal{U}G)$ at t_0 is equivalent to the existence of a witness $t_B < t_0$ for $A\mathcal{S}(B \wedge F\mathcal{U}G)$ at t_0 and a witness $t_G > t_B$ for $F\mathcal{U}G$ at t_B.

Since we are working over linear orders, we can split this into a disjunction of three cases, depending on whether $t_G < t_0$, $t_G = t_0$ or $t_G > t_0$.

For the first case, we have the following diagram:

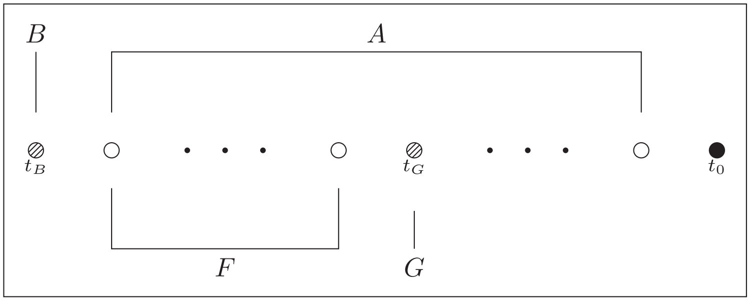

Figure 5 $t_0 \Vdash A\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G < t_0$.

Adjusting this diagram slightly by ‘splitting’ the A, we obtain

It is then clear that this diagram is equivalent to t_B being a witness for $(A \wedge F)\mathcal{S}B$ at t_G, and t_G a witness for $A\mathcal{S}(G \wedge A \wedge (A \wedge F)\mathcal{S}B) = H_<$ at t_0.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

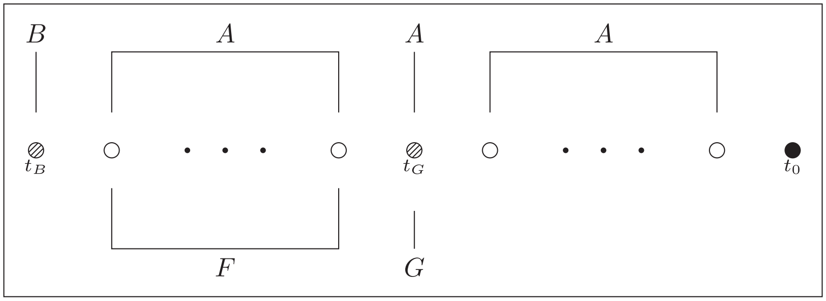

Figure 6 $t_0 \Vdash A\mathcal{S}(B \wedge F\,\mathcal{U}G)$ with $t_G < t_0$, simplified.

The second case ($t_G = t_0$) is equivalent to the diagram:

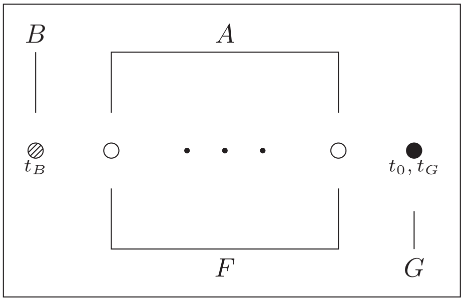

FIGURE 7 $t_0 \Vdash A\mathcal{S}(B \wedge F\,\mathcal{U}G)$ with $t_G = t_0$.

which is equivalent to $(A \wedge F)\mathcal{S}B \wedge G = H_=$ at t_0.
The third case ($t_G > t_0$):

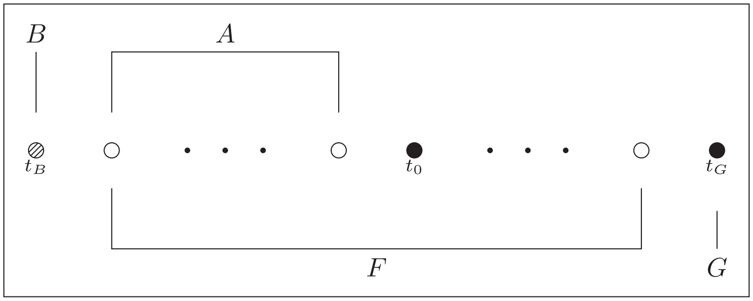

FIGURE 8 $t_0 \Vdash A\mathcal{S}(B \wedge F\,\mathcal{U}G)$ with $t_G > t_0$.

By splitting the F we get
And so Figure 9 is equivalent to $(A \wedge F)\mathcal{S}B \wedge F \wedge F\,\mathcal{U}G = H_>$ at t_0. □

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

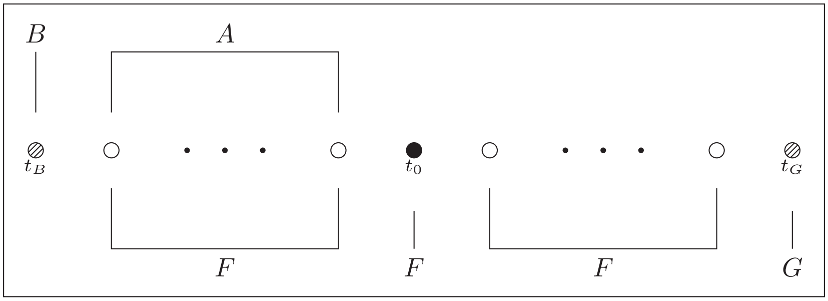

FIGURE 9 $t_0 \Vdash A\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G > t_0$, simplified.

PROPOSITION 3.21
Let $\mathcal{T}_3(A,B,F,G) := H_< \vee H_\geq$, where

$$N := \neg G\mathcal{S}\neg A$$

$$P' := N \rightarrow G \vee F$$

$$P := N \rightarrow G \vee (F \wedge F\mathcal{U}G)$$

$$H_< := P \wedge P'\mathcal{S}(G \wedge F\mathcal{S}B)$$

$$H_\geq := F\mathcal{S}B \wedge (G \vee (F \wedge F\mathcal{U}G))$$

Then $(A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G) \equiv \mathcal{T}_3(A,B,F,G)$.

PROOF. Let t_0 be a point. Then t_0 satisfies $(A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G)$ if and only if there is a $t_B < t_0$ such that $t_B \Vdash B \wedge F\mathcal{U}G$ and $(t_B, t_0) \Vdash A \vee F\mathcal{U}G$. The satisfaction of $F\mathcal{U}G$ at t_B is equivalent to the existence of a witness $t_G > t_B$. We can split this into two cases, $t_G \in (t_B, t_0)$ or $t_G \geq t_0$. We show that these cases are equivalent to $H_<$ and $H_\geq$, respectively.

The first case, where t_G occurs between t_B and t_0, is equivalent to the following diagram:

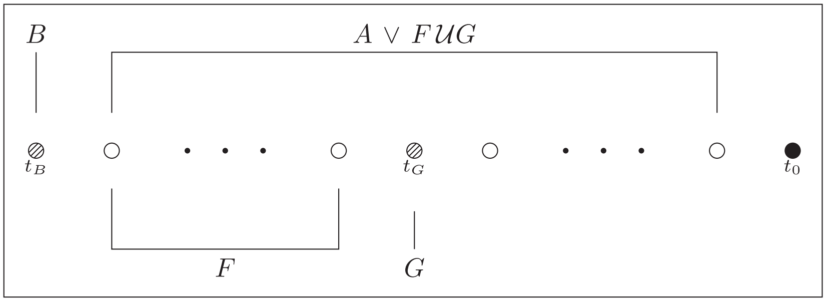

Figure 10 $t_0 \Vdash (A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G < t_0$.

Notice that in this case $F\mathcal{U}G$ is also satisfied in (t_B, t_G) with t_G as witness and so we can simplify the diagram to

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

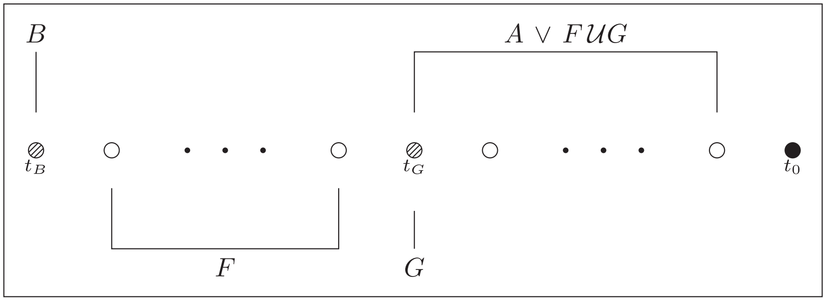

Figure 11 $t_0 \Vdash (A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G < t_0$, simplified.

We also have $t_G \Vdash G \wedge F\mathcal{S}B$; this last one with t_B as witness. Given that $\vDash P \rightarrow P'$, we will show that P is satisfied in $(t_G, t_0]$ to obtain the left to right implication, with t_G as a witness for $P'\mathcal{S}(G \wedge F\mathcal{S}B)$ at t_0.

Given a point $s \in (t_G, t_0]$, assume $s \Vdash N = \neg G\mathcal{S}\neg A$; we have to show that s satisfies either G or $F \wedge F\mathcal{U}G$. This means there is a witness $s_{\neg A} < s$, which cannot occur before t_G, as that would contradict the fact that $t_G \Vdash G$, and therefore occurs at t_G or between t_G and s, which means that $s_{\neg A} \Vdash F\mathcal{U}G$. In a (partial) diagram:

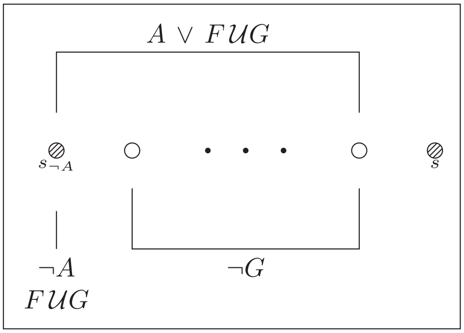

FIGURE 12 Consequences of $s \Vdash N$.

To satisfy $F\mathcal{U}G$ at $s_{\neg A}$ there must be a witness at s or after s, since no point between $s_{\neg A}$ and s can be a witness. If $s \Vdash G$ then we are done. If there is some witness after s, then s must satisfy F and the same witness provides $s \Vdash F\mathcal{U}G$.

Now, still in the first case, we worry about the other direction. Assume $t_0 \Vdash H_< = P \wedge P'\mathcal{S}(G \wedge F\mathcal{S}B)$, with t_G a witness for $P'\mathcal{S}(G \wedge F\mathcal{S}B)$ at t_0 and t_B a witness for $F\mathcal{S}B$ at t_G. Diagram:

We will show that $[t_G, t_0) \Vdash A \vee F\mathcal{U}G$, which gives us the situation in Diagram 11.

Similarly to the ($\Leftarrow$) case of $\mathcal{T}_1$, we first show $(t_G, t_0] \Vdash P$ by induction on the order $t_0 \rightarrow \text{Pred}(t_0) \rightarrow \text{Pred}^2(t_0) \rightarrow \cdots$.

The base case is again trivial since we have $t_0 \Vdash P$. So let $\text{Pred}(s) \in (t_G, t_0)$ and assume $\text{Pred}(s) \Vdash N = \neg G\mathcal{S}\neg A$ with r as witness. Diagram:

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

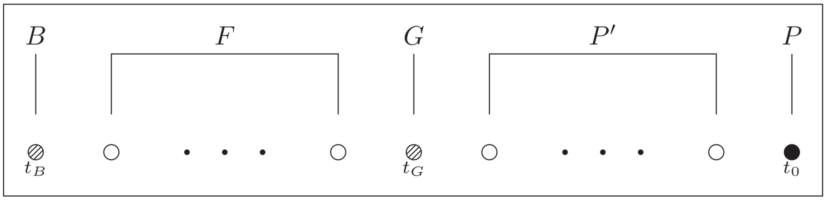

FIGURE 13 $t_0 \Vdash H_<$.

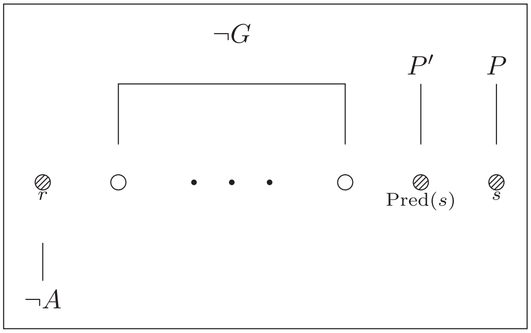

FIGURE 14 Pred$(s) \Vdash N$ with r as witness.

If Pred$(s) \Vdash G$ then Pred$(s) \Vdash G \vee (F \wedge F\mathcal{U}G)$.

If Pred$(s) \Vdash \neg G$ then Pred$(s) \Vdash F$ via P', and r is also a witness for N at s which gives us $s \Vdash G \vee (F \wedge F\mathcal{U}G)$ via P and then Pred$(s) \Vdash F\mathcal{U}G$ by Proposition 3.17.

Given this, any $s_{\neg A} \in [t_G, t_0)$ that satisfies $\neg A$ is a witness for N at Suc$(s_{\neg A}) \in (t_G, t_0]$, which gives us Suc$(s_{\neg A}) \Vdash G \vee (F \wedge F\mathcal{U}G)$ via P and $s_{\neg A} \Vdash F\mathcal{U}G$ by Proposition 3.17.

This concludes the first case.

For the second case, where $t_G \geq t_0$, we have the following diagrams for the cases where $t_G = t_0$ and $t_G > t_0$, respectively:

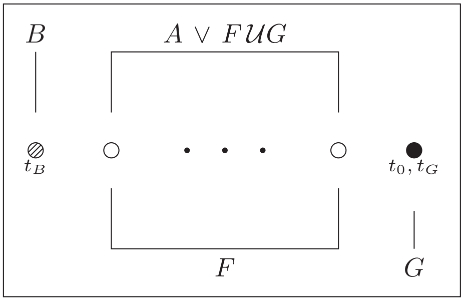

FIGURE 15 $t_0 \Vdash (A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G = t_0$.

In either case, t_G is a witness for $F\mathcal{U}G$ everywhere in (t_B, t_0), and so the diagrams without $A \vee F\mathcal{U}G$ in (t_B, t_0) are equivalent.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

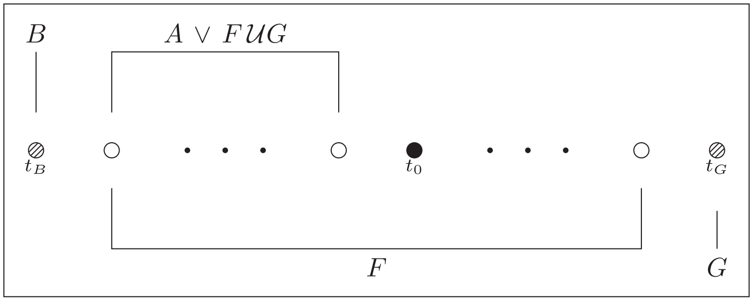

FIGURE 16 $t_0 \Vdash (A \vee F\mathcal{U}G)\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G > t_0$.

The first diagram is then equivalent to $t_0 \Vdash F\mathcal{S}B \wedge G$ and the second to $t_0 \Vdash F\mathcal{S}B \wedge F \wedge F\mathcal{U}G$, which combined by distribution give us $F\mathcal{S}B \wedge (G \vee (F \wedge F\mathcal{U}G)) = H_{\geq}$. □

PROPOSITION 3.22
For all $A \in \mathcal{L}_{\mathrm{TL}}$, $\neg\blacksquare\neg A \equiv \blacklozenge A$.

PROOF. Recall that $\blacksquare A$ is defined as $\neg\blacklozenge\neg A$, then $\neg\blacksquare\neg A = \neg\neg\blacklozenge\neg\neg A \equiv \blacklozenge A$. □

PROPOSITION 3.23
For all $A, B \in \mathcal{L}_{\mathrm{TL}}$: $\neg(A\mathcal{S}B) \equiv \neg B\mathcal{S}(\neg A \wedge \neg B) \vee \blacksquare\neg B$.

PROOF. ($\Rightarrow$) Let $t \Vdash \neg(A\mathcal{S}B)$ and consider the set $S = \{x < t \mid x \Vdash B\}$.

If S is empty then all points before t satisfy $\neg B$ and so $t \Vdash \blacksquare\neg B$.

Otherwise, let s be the greatest point in S. There must be a point in (s,t) that does not satisfy A, as otherwise we would have $t \Vdash A\mathcal{S}B$, call such point r. We are now in the situation described by the following diagram:

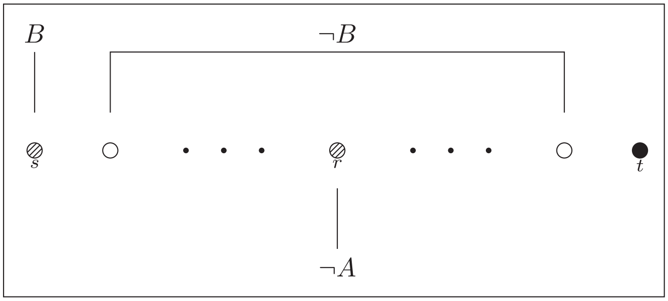

Figure 17 s is the greatest point before t that satisfies B.

which means that r is a witness for $\neg B\mathcal{S}(\neg A \wedge \neg B)$ at t.

($\Leftarrow$) If $t \Vdash \blacksquare\neg B$ then immediately $t \Vdash \neg(A\mathcal{S}B)$.

If $t \Vdash \neg B\mathcal{S}(\neg A \wedge \neg B)$ then let r be a witness. For every point s before t that satisfies B we must have $s < r$ since $[r,t) \Vdash \neg B$, but in that case there is at least one point in (s,t) that does not satisfy A, namely r. □

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

COROLLARY 3.24
For all $A, B \in \mathcal{L}_{\mathrm{TL}}$: $\neg(A \mathcal{U} B) \equiv \neg B \mathcal{U}(\neg A \wedge \neg B) \vee \Box \neg B$.

PROOF. Duality with Proposition 3.23. □

PROPOSITION 3.25
Let $\mathcal{T}_4(A, B, F, G) := \neg \mathcal{T}_2(\neg B, \neg A \wedge \neg B, F, G) \wedge \blacklozenge B$. Then $(A \vee \neg(F \mathcal{U} G)) \mathcal{S} B \equiv \mathcal{T}_4(A, B, F, G)$.

PROOF. We have $(A \vee \neg(F \mathcal{U} G)) \mathcal{S} B \equiv \neg\neg((A \vee \neg(F \mathcal{U} G)) \mathcal{S} B)$. By using Proposition 3.23 with the inner negation and the $\mathcal{S}$ we obtain

$$\neg\Big[\neg B \mathcal{S}(\neg(A \vee \neg(F \mathcal{U} G)) \wedge \neg B) \vee \blacksquare \neg B\Big].$$

Applying a De Morgan law to $\neg(A \vee \neg(F \mathcal{U} G))$ and then again on the outer negation and disjunction:

$$\neg(\neg B \mathcal{S}(\neg A \wedge F \mathcal{U} G \wedge \neg B)) \wedge \neg \blacksquare \neg B.$$

Rearranging the right side of the $\mathcal{S}$ and using Proposition 3.22 on $\neg \blacksquare \neg B$:

$$\neg(\neg B \mathcal{S}(\neg A \wedge \neg B \wedge F \mathcal{U} G)) \wedge \blacklozenge B.$$

The left conjunct is now in the form of $\mathcal{T}_2$ (Proposition 3.20), which gives us the final result

$$\neg \mathcal{T}_2(\neg B, \neg A \wedge \neg B, F, G) \wedge \blacklozenge B.$$

□

LEMMA 3.26
$A \mathcal{S}(B \wedge \Box G) \equiv (A \wedge G) \mathcal{S} B \wedge G \wedge \Box G$

PROOF. The left side of the equivalence is equivalent to the existence of the diagram:

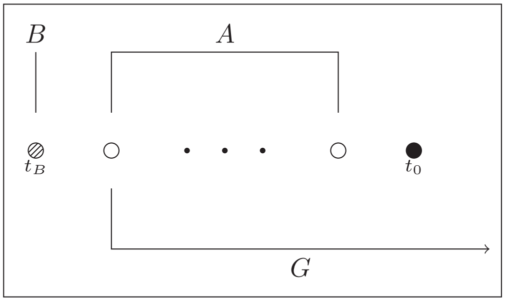

Figure 18 $t_0 \Vdash A \mathcal{S}(B \wedge \Box G)$ with t_B as witness.

By splitting the G we see that it's also equivalent to the right side. □

PROPOSITION 3.27
Let $\mathcal{T}_5(A, B, F, G) := A \mathcal{S}(\neg F \wedge \neg G \wedge A \wedge (A \wedge \neg G) \mathcal{S} B) \vee (A \wedge \neg G) \mathcal{S} B \wedge \neg G \wedge (\neg F \vee \neg(F \mathcal{U} G))$.
Then $A \mathcal{S}(B \wedge \neg(F \mathcal{U} G)) \equiv \mathcal{T}_5(A, B, F, G)$.

PROOF. We start with $A \mathcal{S}(B \wedge \neg(F \mathcal{U} G))$. By Corollary 3.24, we can split the $\neg(F \mathcal{U} G)$ to get

$$A \mathcal{S}(B \wedge (\neg G \mathcal{U}(\neg F \wedge \neg G) \vee \Box \neg G)).$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Performing distribution with the newly introduced disjunction, this is equivalent to the disjunction of the following two formulas

$$A\mathcal{S}(B\wedge\neg G\mathcal{U}(\neg F\wedge\neg G))$$

$$A\mathcal{S}(B\wedge\Box\neg G)$$

The first one is in the form of $\mathcal{T}_2$ (Proposition 3.20), and we use Lemma 3.26 on the second, obtaining

$$\mathcal{T}_2(A,B,\neg G,\neg F\wedge\neg G)$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge\Box\neg G$$

Expanding $\mathcal{T}_2$ in the first line:

$$A\mathcal{S}(\neg F\wedge\neg G\wedge A\wedge(A\wedge\neg G)\mathcal{S}B)$$

$$(A\wedge\neg G)\ \mathcal{S}B\wedge\ ((\neg F\wedge\neg G)\ \wedge\ (\neg G\wedge\neg G\mathcal{U}\ (\neg F\wedge\neg G)))$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge\Box\neg G$$

Distribution on the second formula with $\neg G$:

$$A\mathcal{S}(\neg F\wedge\neg G\wedge A\wedge(A\wedge\neg G)\mathcal{S}B)$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge(\neg F\ \vee\ \neg G\mathcal{U}(\neg F\wedge\neg G))$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge\Box\neg G$$

Distribution again, the second and third formulas have a common prefix

$$A\mathcal{S}\ (\neg F\wedge\neg G\wedge A\wedge\ (A\wedge\neg G)\ \mathcal{S}B)$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge(\neg F\ \vee\ \neg G\mathcal{U}(\neg F\wedge\neg G)\ \vee\ \Box\neg G)$$

Finally, using Corollary 3.24 again, we can get back the $\neg(F\mathcal{U}G)$

$$A\mathcal{S}\ (\neg F\wedge\neg G\wedge A\wedge\ (A\wedge\neg G)\ \mathcal{S}B)$$

$$(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge(\neg F\ \vee\ \neg(F\mathcal{U}G))$$

□

PROPOSITION 3.28

Let $\mathcal{T}_6(A,B,F,G) :=\ \ \mathcal{T}_1(A,(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge\neg F\wedge A,F,G)\ \vee\ \mathcal{T}_3(A,(A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge\neg F, F,G)\ \ \vee\ \ (A\wedge\neg G)\mathcal{S}B\wedge\neg G\wedge(\neg F\ \vee\ \neg(F\mathcal{U}G))$. Then $(A\ \vee\ F\mathcal{U}G)\mathcal{S}(B\wedge\neg(F\mathcal{U}G))\equiv\mathcal{T}_6(A,B, F,G)$.

PROOF. By Corollary 3.24, $(A\ \vee\ F\mathcal{U}G)\mathcal{S}(B\wedge\neg(F\mathcal{U}G))$ is equivalent to

$$\big(A\ \vee\ F\mathcal{U}G\big)\mathcal{S}\big(B\wedge(\neg G\mathcal{U}(\neg F\wedge\neg G)\ \vee\ \Box\neg G)\big).$$

Using distribution, this can split this into the disjunction of

$$(A\ \vee\ F\mathcal{U}G)\mathcal{S}(B\wedge\neg G\mathcal{U}(\neg F\wedge\neg G)) \tag{1}$$

$$(A\ \vee\ F\mathcal{U}G)\mathcal{S}(B\wedge\Box\neg G) \tag{2}$$

Looking at (1) first, we can split this into a disjunction of three cases depending on where the witness for $\neg G\mathcal{U}(\neg F\wedge\neg G)$ occurs: before t_0, at t_0 or after t_0. Call this witness t_T.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

The first case leads to the diagram:

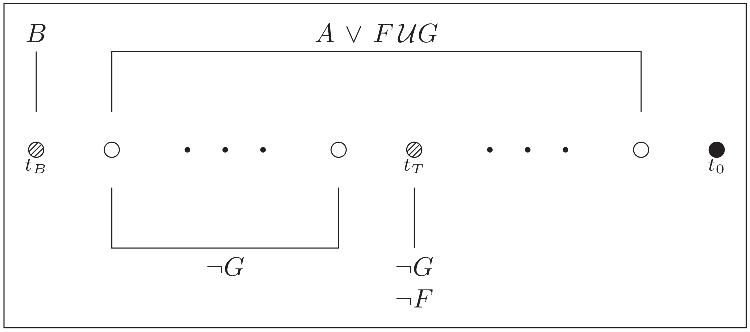

Figure 19 t_0 satisfies (1) with $t_T < t_0$.

Notice that $\neg G\mathcal{U}(\neg F \wedge \neg G)$ is true in (t_B, t_T), which implies $\neg(F\mathcal{U}G)$ by Corollary 3.24, and so this is equivalent to A being true in (t_B, t_T). Updated diagram:

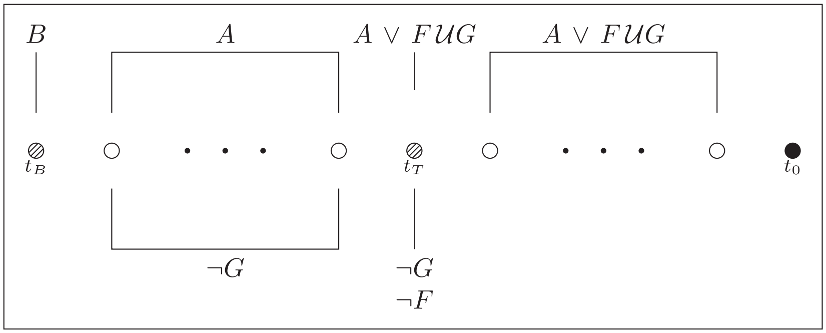

Figure 20 t_0 satisfies (1) with $t_T < t_0$, updated.

This last diagram is then equivalent to

$$(A \vee F\mathcal{U}G)\mathcal{S}((A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F \wedge (A \vee F\mathcal{U}G))\,. \tag{1.<}$$

Again, using distribution on $(A \vee F\mathcal{U}G)$, in the right side of the outer $\mathcal{S}$, we get a disjunction of

$$(A \vee F\mathcal{U}G)\mathcal{S}((A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F \wedge A) \tag{1.<.1}$$

$$(A \vee F\mathcal{U}G)\mathcal{S}((A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F \wedge F\mathcal{U}G) \tag{1.<.2}$$

By Proposition 3.19, (1.<.1) is equivalent to $\mathcal{T}_1(A, (A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F \wedge A, F, G)$. And by Proposition 3.21, (1.<.2) is equivalent to $\mathcal{T}_3(A, (A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F, F, G)$.

We have now obtained the first and second disjuncts of the definition of $\mathcal{T}_6$; we will see that the disjunction of the two remaining cases and (2) is equivalent to the third disjunct.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Continuing with the other cases of (1). The $t_T = t_0$ case leads to the following diagram:

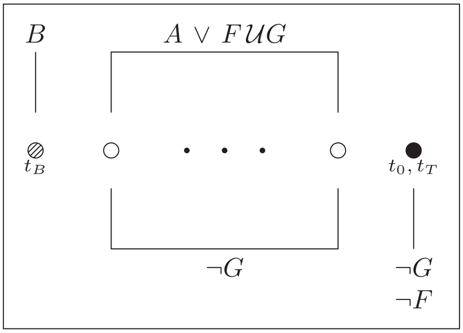

FIGURE 21 t_0 satisfies (1) with $t_T = t_0$.

For the same reason as previously, this is equivalent to having the stronger A instead of $A \vee F\mathcal{U}G$ in (t_B, t_0), and the diagram is equivalent to the formula

$$(A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg F. \tag{1.=}$$

We continue by looking at the final case of (1), where $t_T > t_0$, before coming back to (1.=). Diagram:

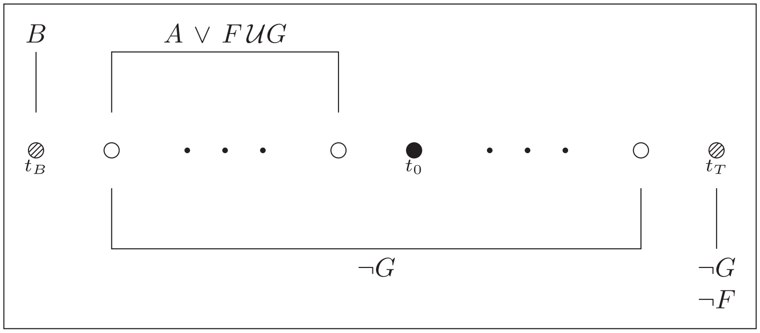

FIGURE 22 t_0 satisfies (1) with $t_T > t_0$.

Similarly, $A \vee F\mathcal{U}G$ can be replaced with A and we get the formula

$$(A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg G\mathcal{U}(\neg F \wedge \neg G)\,. \tag{1.>}$$

We still have the formulas (1.=) and (1.>) to deal with, but we explore (2) before we do so.

A diagram equivalent to (2) is as follows:

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

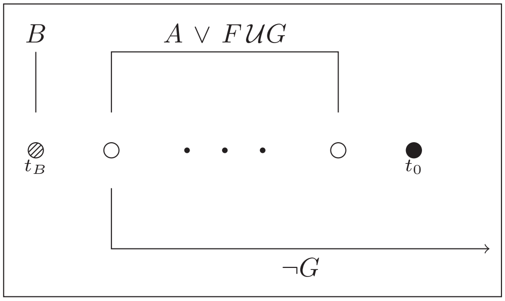

FIGURE 23 t_0 satisfies (2).

Again by Corollary 3.24, $\neg(F\mathcal{U}G)$ is satisfied in (t_B, t_0) and so $A \vee F\mathcal{U}G$ in (t_B, t_0) is equivalent to A in (t_B, t_0), which means (2) is equivalent to $A\mathcal{S}(B \wedge \Box\neg G)$. Using Lemma 3.26, we get

$$(A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \Box\neg G. \tag{2'}$$

We now combine (1.>) and (2′) by distribution (notice the common prefix) to get

$$(A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge (\neg G\mathcal{U}(\neg F \wedge \neg G) \vee \Box\neg G)\,, \tag{3}$$

which, by Corollary 3.24, is equivalent to

$$(A \wedge \neg G)\mathcal{S}B \wedge \neg G \wedge \neg(F\mathcal{U}G)\,. \tag{3'}$$

Combining (1.=) and (3′) by distribution we obtain the third disjunct. □

PROPOSITION 3.29
Let $\mathcal{T}_7(A, B, F, G) := \neg(\mathcal{T}_3(\neg B, \neg A, F, G)) \wedge \mathcal{T}_5(\top, B, F, G)$. Then $(A \vee \neg(F\mathcal{U}G))\mathcal{S}(B \wedge \neg(F\mathcal{U}G)) \equiv \mathcal{T}_7(A, B, F, G)$.

PROOF. For clarity, let $X = A \vee \neg(F\mathcal{U}G)$ and $Y = B \wedge \neg(F\mathcal{U}G)$ and perform these substitutions in the original formula.

This first part is quite similar to the one of Proposition 3.25. We have $X\mathcal{S}Y \equiv \neg\neg(X\mathcal{S}Y)$. Using Proposition 3.23 we get

$$\neg\Big[\neg Y\mathcal{S}(\neg X \wedge \neg Y) \vee \blacksquare\neg Y\Big].$$

Rewriting using a De Morgan law with the outer negation and disjunction, and then using Proposition 3.22 on the right conjunct:

$$\neg(\neg Y\mathcal{S}(\neg X \wedge \neg Y)) \wedge \blacklozenge Y.$$

Consider $\blacklozenge Y$ first. By substituting back Y we get

$$\blacklozenge(B \wedge \neg(F\mathcal{U}G)) = \top\mathcal{S}(B \wedge \neg(F\mathcal{U}G)) \equiv \mathcal{T}_5(\top, B, F, G)\,.$$

Now consider $\neg Y\mathcal{S}(\neg X \wedge \neg Y)$; by substituting back X and Y we get

$$\neg(B \wedge \neg(F\mathcal{U}G))\mathcal{S}(\neg(A \vee \neg(F\mathcal{U}G)) \wedge \neg(B \wedge \neg(F\mathcal{U}G)))\,.$$

Using De Morgan laws on $\neg(A \vee \neg(F\mathcal{U}G))$ and $\neg(B \wedge \neg(F\mathcal{U}G))$:

$$(\neg B \vee F\mathcal{U}G)\mathcal{S}(\neg A \wedge F\mathcal{U}G \wedge (\neg B \vee F\mathcal{U}G))\,.$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

And then using absorption on the right side of the $\mathcal{S}$:

$$(\neg B \vee F\mathcal{U}G)\mathcal{S}(\neg A \wedge F\mathcal{U}G) \equiv \mathcal{T}_3(\neg B, \neg A, F, G)\,.$$

The original formula is then equivalent to $\neg(\mathcal{T}_3(\neg B, \neg A, F, G)) \wedge \mathcal{T}_5(\top, B, F, G)$. □

PROPOSITION 3.30
Let $\mathcal{T}_8(A,B,F,G) := \mathcal{T}_4(A, (A \wedge F)\mathcal{S}B \wedge G \wedge A, F, G) \vee \mathcal{T}_7(A, (A \wedge F)\mathcal{S}B \wedge G, F, G) \vee (A \wedge F)\mathcal{S}B \wedge (G \vee (F \wedge F\mathcal{U}G))$. Then $(A \vee \neg(F\mathcal{U}G))\mathcal{S}(B \wedge F\mathcal{U}G) \equiv \mathcal{T}_8(A,B,F,G)$.

PROOF. This proof follows fairly similarly to the proof of Proposition 3.28.

We begin by considering the three cases depending on where the witness for $F\mathcal{U}G$ in the right side of $\mathcal{S}$ (t_G) occurs, which give rise to the following three diagrams:

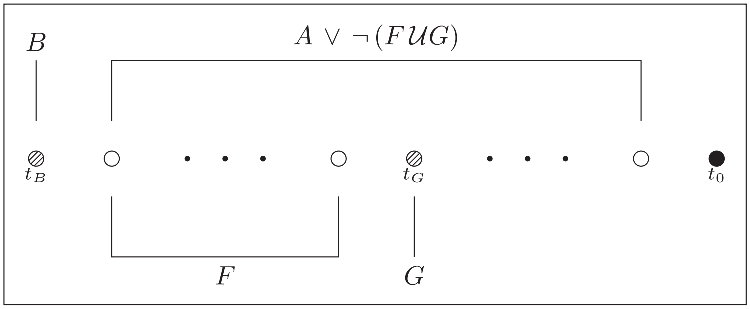

Figure 24 $t_0 \Vdash (A \vee \neg(F\mathcal{U}G))\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G < t_0$.

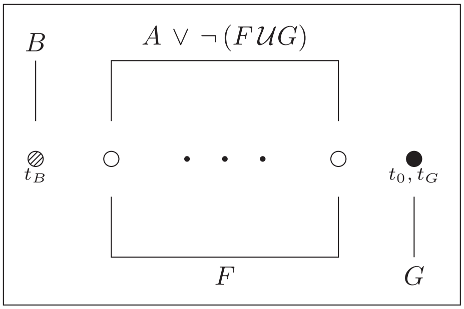

Figure 25 $t_0 \Vdash (A \vee \neg(F\mathcal{U}G))\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G = t_0$.

In all three diagrams, t_G is a witness for $F\mathcal{U}G$ in the region (t_B, t_G), which means that $(t_B, t_G) \Vdash A \vee \neg(F\mathcal{U}G)$ is equivalent to $(t_B, t_G) \Vdash A$. This then gives us the formulas, respectively:

$$(A \vee \neg(F\mathcal{U}G))\mathcal{S}((A \wedge F)\mathcal{S}B \wedge G \wedge (A \vee \neg(F\mathcal{U}G))) \tag{1}$$

$$(A \wedge F)\mathcal{S}B \wedge G \tag{2}$$

$$(A \wedge F)\mathcal{S}B \wedge F \wedge F\mathcal{U}G \tag{3}$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

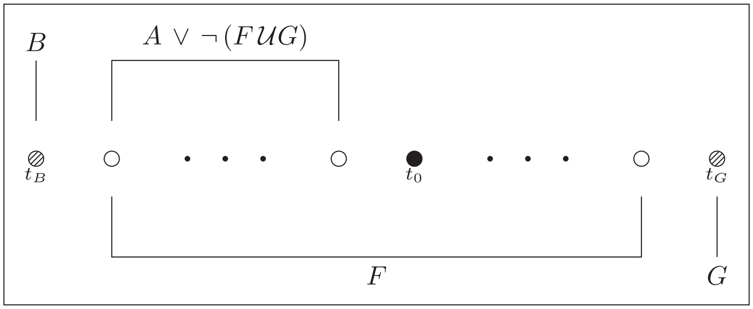

FIGURE 26 $t_0 \Vdash (A \vee \neg(F\mathcal{U}G))\mathcal{S}(B \wedge F\mathcal{U}G)$ with $t_G > t_0$.

We can further split (1) by the $(A \vee \neg(F\mathcal{U}G))$ on right side into

$$(A \vee \neg(F\mathcal{U}G))\mathcal{S}((A \wedge F)\mathcal{S}B \wedge G \wedge A) \tag{1.1}$$

$$(A \vee \neg(F\mathcal{U}G))\mathcal{S}((A \wedge F)\mathcal{S}B \wedge G \wedge \neg(F\mathcal{U}G)) \tag{1.2}$$

(1.1) and (1.2) are equivalent to $\mathcal{T}_4(A, (A \wedge F)\mathcal{S}B \wedge G \wedge A, F, G)$ and $\mathcal{T}_7(A, (A \wedge F)\mathcal{S}B \wedge G, F, G)$, by Propositions 3.25 and 41, respectively.

(2) and (3) combine into the third disjunct of the definition of $\mathcal{T}_8$. □

The algorithm is now completely defined, and the next section is devoted to showing that it is correct.

3.2 *Algorithm correctness*

We have now seen that the transformations produce equivalent formulas. But we still have to show that the algorithm constructs an equivalent separated formula. In particular we have to show that the algorithm does in fact terminate. We do so by induction on a well-founded partial order on formulas.

This order is obtained by combining several relations, each necessary for some cases of the algorithm.

We start by defining some functions on paths that measure how ‘unseparated’ they are.

DEFINITION 3.31
Given a path π, the *degree* of π, $\mathcal{D}(\pi)$, is the number of adjacent pairs in π with both $\mathcal{S}$ and $\mathcal{U}$. Or, perhaps more intuitively, it is the number of transitions from $\mathcal{S}$ to $\mathcal{U}$ and vice versa.

DEFINITION 3.32
Let π be a path. We define $\mathcal{L}_1(\pi)$ to be the length of the last *homogeneous segment*, except in the case where the segment is the whole path, in which case we take $\mathcal{L}_1(\pi)$ to be zero. That is,

$$\mathcal{L}_1(\pi) := \begin{cases} n+1 & \text{if } \pi = \lambda\mathcal{S}\mathcal{U}^{n+1} \text{ or } \pi = \lambda\mathcal{U}\mathcal{S}^{n+1} \\ 0 & \text{otherwise.} \end{cases}$$

For convenience, we combine these into one.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

DEFINITION 3.33
Let π be a path. Then,

$$\mathrm{Score}_1(\pi) := \langle \mathcal{D}(\pi), \mathcal{L}_1(\pi) \rangle .$$

So far these have been over paths; we now extend the definitions to formulas.

Whenever we have a Cartesian product of orders, the intended order on it is the usual lexicographic order. That is, if $\langle x, y \rangle, \langle x', y' \rangle \in X \times Y$ with $<_X$ an order over X and $<_Y$ an order over Y, $\langle x, y \rangle < \langle x', y' \rangle$ if and only if $x <_X x'$ or ($x = x'$ and $y <_Y y'$). If all such orders are well founded then so is the lexicographic order, and if they are total then so is the lexicographic order.

DEFINITION 3.34
Let $A \in \mathcal{L}_{\mathrm{TL}}$. Then,

$$\mathrm{Score}_1(A) := \begin{cases} \langle 0, 0 \rangle & \text{if } \mathrm{Paths}(A) = \emptyset \\ \max \mathrm{Score}_1[\mathrm{Paths}(A)] & \text{otherwise.} \end{cases}$$

To obtain something that is reduced by every transformation, we need an additional thing, which we call leader count, and whose definition requires the idea of contexted formula.

DEFINITION 3.35
A *contexted formula* is a pair $\langle \pi, A \rangle$ where π is a path and A is a formula.

DEFINITION 3.36
Let A be a formula. Then $\mathcal{C}(A)$ is a set of contexted formulas, consisting of the subformulas of A together with their context (the path corresponding to the syntax path that goes from the root of A to the root of the subformula's tree).

As an example of a contexted subformula, consider the following formula

$$A = (q\mathcal{U}(\top\mathcal{U}p) \vee p\mathcal{U}q)\mathcal{S}(p\mathcal{S}(r\mathcal{U}(\top\mathcal{U}p))) .$$

The formula A has the following syntax tree:

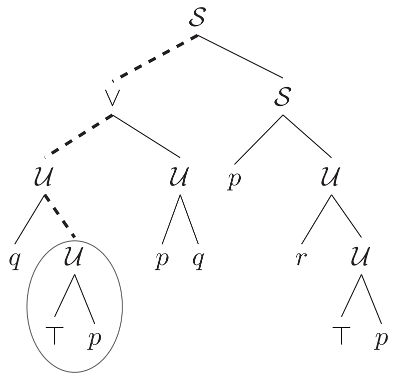

FIGURE 27 Syntax tree of A.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

The circled subformula is $\top\mathcal{U}p$. The syntax path that goes from the root of A to the root of this subformula is dashed, and is associated with the path $\mathcal{SU}$. This then represents the contexted subformula $\langle\mathcal{SU}, \top\mathcal{U}p\rangle$ of A.

For the purposes of our algorithm, it suffices to look at the subset of these contexted subformulas where the subformula is simple and pure, which leads to the following definition.

Definition 3.37
Let $A \in \mathcal{L}_{\mathrm{TL}}$.

$$\mathcal{SPC}(A) := \{\langle\pi, B\rangle \in \mathcal{C}(A) \mid B \text{ is simple and pure}\}\,.$$

We now extend the definition of Paths and Score_1 to contexted formulas in a natural way.

Definition 3.38
Let $\langle\pi, A\rangle$ be a contexted formula. Then,

$$\mathrm{Paths}(\langle\pi, A\rangle) := \{\pi\lambda \mid \lambda \in \mathrm{Paths}(A)\}$$

$$\mathrm{Score}_1(\langle\pi, A\rangle) := \begin{cases} \langle 0, 0\rangle & \text{if } \mathrm{Paths}(\langle\pi, A\rangle) = \emptyset \\ \max \mathrm{Score}_1[\mathrm{Paths}(\langle\pi, A\rangle)] & \text{otherwise.} \end{cases}$$

With this, we can define the concepts of leader and leader count.

Definition 3.39
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and $\langle\pi, B\rangle \in \mathcal{SPC}(A)$.

Then we say $\langle\pi, B\rangle$ is a *leader* of A if it has maximal Score_1 in $\mathcal{SPC}(A)$ and call $\mathrm{Leaders}(A)$ the set of leaders of A.

Notice that both $\mathcal{SPC}$ and Leaders can be empty, e.g. the formula $\top \wedge \bot$ has no simple pure subformulas and as such $\mathcal{SPC}(\top \wedge \bot) = \emptyset$ and consequently $\mathrm{Leaders}(\top \wedge \bot) = \emptyset$.

Definition 3.40
Given a formula A, $\mathcal{N}(A)$ is the number of leaders of A with distinct formulas, i.e.

$$\mathcal{N}(A) := \#\{B \mid \langle\pi, B\rangle \in \mathrm{Leaders}(A)\}\,.$$

Coming back to the previous example, here we have the syntax tree of A with the pure simple formulas circled:

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

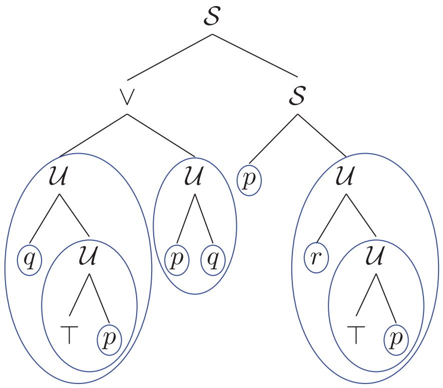

FIGURE 28 Syntax tree of A with $\mathcal{SPC}$ circled.

Now consider only the ones with maximal Score$_1$; these are the leaders of A, circled in following tree diagram:

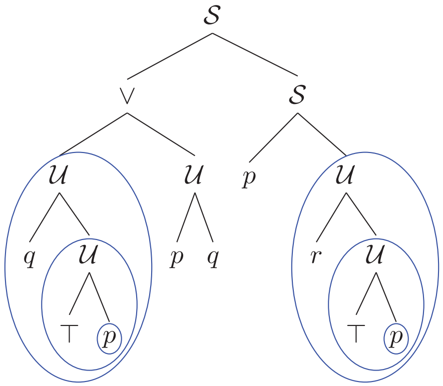

FIGURE 29 Syntax tree of A with leaders circled.

We have $\mathcal{N}(A) = 4$ and not 6, as the subformulas $\top\mathcal{U}p$ and p are repeated in the leaders, even though they appear with different contexts.

So far we have defined Score$_1$ and $\mathcal{N}$ on formulas, and indeed these together are enough to obtain something that is reduced by every transformation when used in the conditions of the algorithm. The idea is that when a transformation 'pulls out' one of the $F\mathcal{U}G$ from inside the $\mathcal{S}$, it either reduces $\mathcal{N}$ or it pulled out the last leader and Score$_1$ decreases.

We now define another pair of functions, quite similar to $\mathcal{L}_1$ and Score$_1$, which are necessary for case 6 of the algorithm.

DEFINITION 3.41
Let π be a path. Then $\mathcal{L}_0(\pi)$ is the length of the first homogeneous segment, with the exception of a

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

fully homogeneous path. Score_0 is analogous to Score_1.

$$\mathcal{L}_0(\pi) := \begin{cases} n+1 & \text{if } \pi = \mathcal{S}^{n+1}\mathcal{U}\lambda \text{ or } \pi = \mathcal{U}^{n+1}\mathcal{S}\lambda \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Score}_0(\pi) := \langle \mathcal{D}(\pi), \mathcal{L}_0(\pi) \rangle$$

This leads to a definition of Score_0 on formulas analogous to the definition of Score_1.

Definition 3.42
Let A be a formula, then

$$\text{Score}_0(A) := \begin{cases} \langle 0, 0 \rangle & \text{if Paths}(A) = \emptyset \\ \max \text{Score}_0[\text{Paths}(A)] & \text{otherwise.} \end{cases}$$

We now define three functions measuring how complex a formula is. These are useful when we do the normal form convertions and $\mathcal{S}$ distribution.

Definition 3.43
Let $A \in \mathcal{L}_{\text{TL}}$. Then

- $\mathcal{H}(A)$ is the number of vertices under a negation in the syntax tree of A.
- $\mathcal{J}(A)$ is the maximum number of transitions from $\wedge$ to $\vee$ and vice versa. Analogous to $\mathcal{D}$.
- $\mathcal{V}(A)$ is the number of vertices in A's syntax tree.

And finally something to handle the case where we convert an $\mathcal{U}$ to its Dual.

Definition 3.44
Let $A \in \mathcal{L}_{\text{TL}}$.

$$\mathcal{P}(A) := \begin{cases} 0 & \text{if } A \text{ has no paths} \\ 1 & \text{if all paths of } A \text{ start with } \mathcal{S} \\ 2 & \text{if all paths of } A \text{ start with } \mathcal{U} \\ 3 & \text{otherwise} \end{cases}$$

For convenience we bundle the previously defined functions in a function called *weight*.

Definition 3.45
Given a formula A:

$$\mathcal{W}(A) := \langle \text{Score}_1(A), \mathcal{N}(A), \text{Score}_0(A), \mathcal{H}(A), \mathcal{J}(A), \mathcal{V}(A), \mathcal{P}(A) \rangle .$$

The final relation is then defined by $\mathcal{W}$.

Definition 3.46
The binary relation $< \subseteq \mathcal{L}_{\text{TL}} \times \mathcal{L}_{\text{TL}}$ is defined as $A < B$ if and only if $\mathcal{W}(A) < \mathcal{W}(B)$.

This $<$ is well founded by the result below.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

PROPOSITION 3.47
Let R be a well-founded binary relation on A, S a binary relation on X and $f : X \to A$ a function such that

$$xSy \implies f(x)Rf(y).$$

Then S is well founded.

PROOF. Let $Y \subseteq X$ be non-empty and let $f(m) \in f[Y] \subseteq A$ be R-minimal in $f[Y]$. Then for any $x \in Y$, xSm implies $f(x)Rf(m)$, which is a contradiction. Hence, m is S-minimal in Y. □

The relation $<$ is in fact also a partial order but we shall not show this.

We now prove some properties related to Score_1 and Leaders.

PROPOSITION 3.48
For any paths π,λ,μ and formulas A,B we have

(i) $\mathcal{D}(\pi\mu) \leq \mathcal{D}(\pi\lambda\mu)$
(ii) $\mathrm{Score}_1(\pi\mu) \leq \mathrm{Score}_1(\pi\lambda\mu)$
(iii) $\mathrm{Score}_1(\mathcal{S}^n\pi) \leq \mathrm{Score}_1(\mathcal{S}\pi)$
(iii) $\mathrm{Score}_1(\mathcal{U}^n\pi) \leq \mathrm{Score}_1(\mathcal{U}\pi)$
(iv) $\mathrm{Score}_1(\langle\pi\mu, A\rangle) \leq \mathrm{Score}_1(\langle\pi\lambda\mu, A\rangle)$
(v) $\mathrm{Score}_1(\langle\mathcal{S}^n\pi, A\rangle) \leq \mathrm{Score}_1(\langle\mathcal{S}\pi, A\rangle)$
(vi) If B is a subformula of A then $\mathrm{Score}_1(\langle\pi, B\rangle) \leq \mathrm{Score}_1(\langle\pi, A\rangle)$

PROOF.

(i) We first consider the case where $\lambda = \mathcal{S}$.
Inserting a single $\mathcal{S}$ at the start of a path either increases the number of transitions, if the initial symbol is $\mathcal{U}$, or does not change it, if the initial symbol is $\mathcal{S}$ or if the path is empty.
Similarly, inserting an $\mathcal{S}$ at the end also cannot decrease the number of transitions.
If a $\mathcal{S}$ is inserted in between two symbols, the number of transitions increases if both symbols are $\mathcal{U}$, and remains the same in the other three cases.
The case where $\lambda = \mathcal{U}$ is very similar to the case above.
The general case can then be obtained by induction on the length of λ.

(ii) By (ii) we have $\mathcal{D}(\pi\mu) \leq \mathcal{D}(\pi\lambda\mu)$.
Assume that $\mathcal{D}(\pi\mu) = \mathcal{D}(\pi\lambda\mu)$, $\mathcal{L}_1(\pi\mu) > \mathcal{L}_1(\pi\lambda\mu)$ and $\lambda \neq \langle\rangle$; we show that this is a contradiction.
By assumption, $\mathcal{L}_1(\pi\mu) > 0$ as 0 is a minimum. So let us assume that $\pi\mu = \tau\mathcal{U}^{n+1}$, with $\mathcal{U}^{n+1}$ the final homogeneous segment. The proof for the other case being obtained by swapping $\mathcal{S}$ with $\mathcal{U}$.
If λ is inserted before the final homogeneous segment, then it cannot reduce the length of such segment; hence, we can assume that λ was inserted inside the final segment.
If $\lambda = \mathcal{U}^{m+1}$ (for some m), then the insertion inside the final segment only increases its length; this implies that λ must have at least one $\mathcal{S}$. But this means that $\mathcal{D}$ has increased, a contradiction.

(iii) We show only the $\mathcal{S}$ case.
If $n = 0$ we can use (ii) directly. Assume then that $n > 0$.
There are no transitions in $\mathcal{S}^{n-1}$, or between $\mathcal{S}^{n-1}$ and $\mathcal{S}$, so deleting $\mathcal{S}^{n-1}$ does not affect degree.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

If the final homogeneous segment is the whole of $\mathcal{S}^n\pi$, then $\mathcal{L}_1(\mathcal{S}^n\pi) = \mathcal{L}_1(\mathcal{S}\pi) = 0$. Otherwise, the deletion of the $\mathcal{S}^{n-1}$ prefix does not affect the final homogeneous segment, and so $\mathcal{L}_1(\mathcal{S}^n\pi) = \mathcal{L}_1(\mathcal{S}\pi)$ again.

(iv)

$$\begin{aligned} \mathrm{Score}_1(\langle \pi\mu, A\rangle) &= \max\{\mathrm{Score}_1(\pi\mu\tau) \mid \tau \in \mathrm{Paths}(A)\} && \text{(by definition of Score}_1\text{)} \\ &\leq \max\{\mathrm{Score}_1(\pi\lambda\mu\tau) \mid \tau \in \mathrm{Paths}(A)\} && \text{(item (ii))} \\ &= \mathrm{Score}_1(\langle \pi\lambda\mu, A\rangle) && \text{(by definition of Score}_1\text{)} \end{aligned}$$

(v)

$$\begin{aligned} \mathrm{Score}_1(\left\langle \mathcal{S}^n\pi, A\right\rangle) &= \max\left\{\mathrm{Score}_1(\mathcal{S}^n\pi\lambda) \mid \lambda \in \mathrm{Paths}(A)\right\} && \text{(by definition of Score}_1\text{)} \\ &\leq \max\{\mathrm{Score}_1(\mathcal{S}\pi\lambda) \mid \lambda \in \mathrm{Paths}(A)\} && \text{(item (iii))} \\ &= \mathrm{Score}_1(\langle \mathcal{S}\pi, A\rangle) && \text{(by definition of Score}_1\text{)} \end{aligned}$$

(vi) Given any path $\lambda \in \mathrm{Paths}(\langle \pi, B\rangle)$, we have $\lambda = \pi\lambda'$ for some $\lambda' \in \mathrm{Paths}(B)$.
Since B is a subformula of A, there is a syntax path from the root of A to the root of B, call the corresponding path μ. Then $\mu\lambda' \in \mathrm{Paths}(A)$, which means $\pi\mu\lambda' \in \mathrm{Paths}(\langle\pi, A\rangle)$.
By item (ii), $\mathrm{Score}_1(\pi\lambda') \leq \mathrm{Score}_1(\pi\mu\lambda')$. □

Item (iii) of the previous proposition is the reason why we defined $\mathcal{L}_1$ to be zero in case the path has degree zero. This is technically not required for our proof of correctness, but it makes things easier. In the case of $\mathcal{L}_0$, we do so purely for symmetry with $\mathcal{L}_1$.

PROPOSITION 3.49
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and $x \in \mathcal{C}(A)$. Then $\mathrm{Paths}(x) \subseteq \mathrm{Paths}(A)$.

PROOF. Let $x = \langle \pi, B\rangle$ and let $\lambda \in \mathrm{Paths}(x)$. We then have $\lambda = \pi\lambda'$ where λ' is a path of B.

The syntax tree of B is a subtree of the one of A, and there's a syntax path from the root of A to the root of B, corresponding to π. And another syntax path from the root of B to a leaf, corresponding to λ'.

The concatenation of these syntax paths is a syntax path of A and corresponds to $\pi\lambda' = \lambda$. □

PROPOSITION 3.50
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and $x \in \mathcal{C}(A)$. Then $\mathrm{Score}_1(x) \leq \mathrm{Score}_1(A)$.

PROOF. If x has no paths then it has minimal Score_1 and we are done. Otherwise, by Proposition 3.49, any path of x is also one of A, in particular a maximal Score_1 one, hence $\mathrm{Score}_1(x) \leq \mathrm{Score}_1(A)$. □

The next proposition shows that the Score_1 of a formula is precisely the Score_1 of its leaders.

PROPOSITION 3.51
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and $x \in \mathcal{SPC}(A)$. Then,

$$\mathrm{Score}_1(x) = \mathrm{Score}_1(A) \iff x \in \mathrm{Leaders}(A).$$

PROOF. ($\Rightarrow$) From Proposition 3.50 we see that the maximum Score_1 of any contexted subformula is $\mathrm{Score}_1(A)$. The contexted subformula x achieves this maximum and so is by definition a leader.

($\Leftarrow$) If $\mathrm{Score}_1(A) = \langle 0, 0\rangle$, this is the minimum so $\mathrm{Score}_1(x)$ can't be any lower.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

From now on we assume that $\text{Score}_1(A) > \langle 0, 0\rangle$ and let π be a maximum Score_1 path of A.

Since we are assuming that x is a leader, it is enough to find some $y \in \mathcal{SPC}(A)$ such that $\text{Score}_1(\pi) \leq \text{Score}_1(y)$, since we also have $\text{Score}_1(y) \leq \text{Score}_1(x)$ by definition of leader.

Now consider a syntax path corresponding to π. We can go up the syntax tree from the leaf until we reach a $\mathcal{S}$ or an $\mathcal{U}$. Call B the simple subformula of A obtained in this way and λ its context. Notice that we have $\pi = \lambda\mathcal{S}$ or $\pi = \lambda\mathcal{U}$, depending on which temporal operator is B's outer operator. For convenience, assume that we are in the $\mathcal{S}$ case, i.e. $B = C\mathcal{S}D$ (for some formulas C and D) and $\pi = \lambda\mathcal{S}$. The $\mathcal{U}$ case is very similar.

If B is pure, then $\langle\lambda, B\rangle \in \mathcal{SPC}(A)$, and there is some μ such that $\mathcal{S}\mu$ is a path of B; hence, $\lambda\mathcal{S}\mu$ is a path of $\langle\lambda, B\rangle$. By Proposition 3.48, $\text{Score}_1(\lambda\mathcal{S}) \leq \text{Score}_1(\lambda\mathcal{S}\mu)$.

If B is not pure, then there must be an $\mathcal{U}$ in either C or D, and so there is some path μ that contains an $\mathcal{U}$ such that $\lambda\mathcal{S}\mu$ is a path of $\langle\lambda, B\rangle$, and thus also a path of A by Proposition 3.49. This path $\lambda\mathcal{S}\mu$ has at least one additional transition from $\mathcal{S}$ to $\mathcal{U}$ and so its degree is greater than the one of π, which contradicts the assumption that π is a maximum Score_1 path of A. □

Proposition 3.52
Let $A \in \mathcal{L}_{\text{TL}}$ and $B \in \text{Subs}(A)$. Then $\mathcal{W}(B) \leq \mathcal{W}(A)$.

Proof. By Proposition 3.50, $\text{Score}_1(B) = \text{Score}_1(\langle\langle\rangle, B\rangle) \leq \text{Score}_1(A)$. Assume $\text{Score}_1(B) = \text{Score}_1(A)$.

For any leader $\langle\pi, C\rangle$ of B there must be a leader $\langle\lambda\pi, C\rangle$ of A since every contexted subformula of B is also contained in A (possibly with an additional prefix of context). This gives us that $\mathcal{N}(B) \leq \mathcal{N}(A)$. Assume $\mathcal{N}(B) = \mathcal{N}(A)$.

For every maximum degree path π of B there is a path $\lambda\pi$ of A and since it is of maximum degree (which is the same for both A and B by assumption), we cannot get a degree increase from the addition of the prefix λ. The prefix can then only increase $\mathcal{L}_0$. Assume $\text{Score}_0(B) \leq \text{Score}_0(A)$.

A subformula clearly cannot have more vertices under negation or more transitions from $\wedge$ to $\vee$ (or vice versa); hence, $\mathcal{H}(B) \leq \mathcal{H}(A)$ and $\mathcal{J}(B) \leq \mathcal{J}(A)$.

But a subformula definitely has decreased vertex count. □

Next we show that transformations do not increase Score_1, proving first some results to make this easier.

Proposition 3.53
For every transformation, let $X(A, B, F, G)$ be the starting formula and X' the resulting formula.

Then X' is constructed using only A,B,F,G,$F\mathcal{U}G$,$\top$,$\neg$,$\vee$,$\wedge$ and $\mathcal{S}$.

Moreover, $F\mathcal{U}G$ is never inside a $\mathcal{S}$ in this construction.

Proof. Every explicit use of $\mathcal{U}$ in any transformation is in the form $F\mathcal{U}G$ and never inside any other temporal operator. This is true even in the transformations that use other transformations, as that is always done with $F := F$ and $G := G$. □

Definition 3.54
Let $A \in \mathcal{L}_{\text{TL}}$ and $S \subseteq \mathcal{C}(A)$. We say S is *complete* (for A) if $\bigcup \text{Paths}[S] = \text{Paths}(A)$.

Proposition 3.55
Let $A \in \mathcal{L}_{\text{TL}}$ and let $S \subseteq \mathcal{C}(A)$ be non-empty and complete. Then $\text{Score}_1(A) = \max \text{Score}_1[S]$.

Proof. The maximum Score_1 of S is the maximum Score_1 of a path of an element of S, which by definition of complete considers exactly the paths of A. □

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

This notion of completeness is too strong for our purposes; we can define a weaker notion of completeness that is still sufficient by allowing deletion of contexted formulas when they also appear with a stronger context and the same formula.

Definition 3.56
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and $S \subseteq \mathcal{C}(A)$. We say S is *quasi-complete* (for A) if there is an $S' \subseteq \mathcal{C}(A)$ such that

- $S \subseteq S'$
- S' is complete
- For all $\langle \pi, C\rangle \in S'$ there is a path λ obtained from π by performing a finite number of insertions and $\langle \lambda, C\rangle \in S$

We call S' a *completion* of S.

Proposition 3.57
Let $A \in \mathcal{L}_{\mathrm{TL}}$ and let $S \subseteq \mathcal{C}(A)$ be non-empty and quasi-complete. Then $\mathrm{Score}_1(A) = \max \mathrm{Score}_1[S]$.

Proof. Let S' be a completion of S. We prove $\max \mathrm{Score}_1[S] = \max \mathrm{Score}_1[S']$ and then use Proposition 3.55.

We have $S \subseteq S'$; therefore, $\mathrm{Score}_1[S] \subseteq \mathrm{Score}_1[S']$.

For the other direction, let $\langle \pi, C\rangle \in S'$. Then, since S' is a completion of S, there is a $\langle \lambda, C\rangle \in S$ and λ is obtained by inserting a finite number of elements into π, which means that $\mathrm{Score}_1(\langle \pi, C\rangle) \leq \mathrm{Score}_1(\langle \lambda, C\rangle)$ by Proposition 3.48. □

Proposition 3.58
For every transformation, let $X(A, B, F, G)$ be the starting formula and X' the resulting formula. Then,

- $S = \{\langle \mathcal{S}, A\rangle, \langle \mathcal{S}, B\rangle, \langle \mathcal{S}, F\mathcal{U}G\rangle\}$ is a complete set for X.
- There are $n_A, n_B, n_F, n_G, n_\top \in \mathbb{N}$ such that some non-empty subset of

$$S' = \left\{\left\langle \mathcal{S}^{n_A}, A\right\rangle, \left\langle \mathcal{S}^{n_B}, B\right\rangle, \left\langle \mathcal{S}^{n_F}, F\right\rangle, \left\langle \mathcal{S}^{n_G}, G\right\rangle, \langle\langle\rangle, F\mathcal{U}G\rangle, \left\langle \mathcal{S}^{n_\top}, \top\right\rangle\right\}$$

  is a quasi-complete set for X'.

Proof. It's easy to see that S is complete for X.

For the second claim, given Proposition 3.53, a path of X' is a path of $F\mathcal{U}G$, A, B, F, G or $\top$, possibly prefixed by some number of $\mathcal{S}$s in the case of the last five. □

Proposition 3.59
For every transformation, let $X(A, B, F, G)$ be the starting formula and X' the resulting formula. Then $\mathrm{Score}_1(X') \leq \mathrm{Score}_1(X)$.

Proof. Let S and S' be quasi-complete sets for X and X',respectively, as given by Proposition 3.58. Using Proposition 3.48 we get

- $\mathrm{Score}_1(\langle \mathcal{S}^{n_A}, A\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}, A\rangle)$
- $\mathrm{Score}_1(\langle \mathcal{S}^{n_B}, B\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}, B\rangle)$
- $\mathrm{Score}_1(\langle \mathcal{S}^{n_F}, F\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}, F\mathcal{U}G\rangle)$
- $\mathrm{Score}_1(\langle \mathcal{S}^{n_G}, G\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}, F\mathcal{U}G\rangle)$
- $\mathrm{Score}_1(\langle\langle\rangle, F\mathcal{U}G\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}, F\mathcal{U}G\rangle)$
- $\mathrm{Score}_1(\langle \mathcal{S}^{n_\top}, \top\rangle) = \langle 0, 0\rangle$ which is the minimum element in this order,

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

which means $\max \mathrm{Score}_1[S'] \leq \max \mathrm{Score}_1[S]$ and thus $\mathrm{Score}_1(X') \leq \mathrm{Score}_1(X)$ by Propositions 3.57 and 3.55. □

The next lemma is useful to handle case 4 of the algorithm in Theorem 3.63. It intuitively says that transformations do not introduce new simple non-past formulas and do not increase the Score_1 of any of them.

LEMMA 3.60
For every transformation, let $X(A,B,F,G)$ be the starting formula and X' the resulting formula.

If $\langle \pi, C\rangle$ is a simple and non-past contexted subformula of X' then there is a λ such that $\langle \lambda, C\rangle \in \mathcal{SPC}(X)$ and $\mathrm{Score}_1(\langle \pi, C\rangle) \leq \mathrm{Score}_1(\langle \lambda, C\rangle)$.

PROOF. By Proposition 3.53, there are no explicit uses of variables in the construction of X' and the only explicit use of $\mathcal{U}$ is in $F\,\mathcal{U}G$; therefore, C must be a subformula of A,B,F,G or $F\,\mathcal{U}G$ with some context μ. This means that C appears in X' with context $\mathcal{S}^n\mu$ (for some n).

Looking at X, we can see that A, B and $F\,\mathcal{U}G$ appear in a Boolean combination immediately inside the outer $\mathcal{S}$; therefore, if C was taken from one of these it appears in X with a context $\mathcal{S}\mu$. If C comes from F or G, then it appears in X with context $\mathcal{SU}\mu$.

By Proposition 3.48, $\mathrm{Score}_1(\langle \mathcal{S}^n\mu, C\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}\mu, C\rangle) \leq \mathrm{Score}_1(\langle \mathcal{SU}\mu, C\rangle)$, so in either case we have the desired result. □

Finally, before we get into the main theorem, we prove two lemmas related to normal form conversion that are helpful in case 5.

LEMMA 3.61
Let X be a formula in negative normal form (NNF) with

- $\bigwedge_{i=1}^{n} Y_i := \mathrm{CNF}(X)$
- $\bigvee_{j=1}^{m} Z_j := DNF(X)$

Then $\mathcal{J}(Y_i) \leq \mathcal{J}(X)$ and $\mathcal{J}(Z_j) \leq \mathcal{J}(X)$, for all relevant i and j.

PROOF. We will show this only for the conjunctive normal form (CNF) case, the disjunctive normal form (DNF) case being identical with conjunction and disjunction swapped.

For this purpose let us then define a simplified version of the CNF algorithm for NNF formulas and ignoring any clause simplication rules described in Definition 3.15, noting that such rules would only remove literals from the clauses produced by this algorithm and not affect our argument.

```
procedure $CNF'(X)$
    $^0$if $X$ is a literal then return$X$
    $^1$else if $X = A \wedge B$
        let $\bigwedge_{i=1}^{n} A_i := \mathrm{CNF}'(A)$
        let $\bigwedge_{j=1}^{m} B_j := \mathrm{CNF}'(B)$
        return$\left(\bigwedge_{i=1}^{n} A_i\right) \wedge \left(\bigwedge_{j=1}^{m} B_j\right)$
    $^2$else if $X = A \vee B$
```

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

```
let $\bigwedge_{i=1}^{n} A_i$ := CNF$'(A)$
let $\bigwedge_{j=1}^{m} B_j$ := CNF$'(B)$
return $\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} (A_i \vee B_j)$
```

The proof proceeds by structural induction on the formula.

Given our assumptions, everything that is not a disjunction or conjunction is considered in case 0, which is trivial.

In case 1 we can use the induction hypothesis to get $\mathcal{J}(A_i) \leq \mathcal{J}(A) \leq \mathcal{J}(A \vee B)$, with B_j being analogous on the B side.

For case 2, notice that $\mathcal{J}(A_i \vee B_j)$ is given by considering the maximal alternation $\vee\wedge$-paths of each literal in the clause with the additional outer $\vee$. But each such literal already appears in $A \vee B$ under $\vee$. □

LEMMA 3.62
Let X be a formula in NNF where each literal appears only once. Then,

(i) If $C \wedge D \in \mathrm{Subs}(X)$ with the literal $c \in \mathrm{Subs}(C)$ and the literal $d \in \mathrm{Subs}(D)$, then c and d never appear together in the same clause in $\mathrm{CNF}(X)$.

(ii) If $C \vee D \in \mathrm{Subs}(X)$ with the literal $c \in \mathrm{Subs}(C)$ and the literal $d \in \mathrm{Subs}(D)$, then c and d never appear together in the same clause in $DNF(X)$.

As in Lemma 3.61, we shall show only the CNF case and proceed by structural induction in the same way.

Case 0 is trivial.

For case 1. Notice that the clauses of the final result are precisely those of $\mathrm{CNF}(A)$ and $\mathrm{CNF}(B)$. If c comes from A and d from B, then they are not put together in the same clause. If both c and d come from the same side (either A and B), then the induction hypothesis gives us the desired result.

In case 2, given that the outer operator of X is a disjunction, $C \wedge D$ must be a subformula of either A or B. For the sake of argument, assume that $C \wedge D$ is a subformula of A.

Consider a clause $A_i \vee B_j$ of the final result. The induction hypothesis gives us that the literals in A_i do not violate the condition by themselves, and given the fact that literals appear only once in X, they cannot occur in B_j since the literals of B_j do not occur in A (and thus do not occur in A_i).

The next theorem shows that Sep is correct. Item (ii) is useful when handling case 6 of the algorithm.

THEOREM 3.63
For every $X \in \mathcal{L}_{\mathrm{TL}}$ we have the following:

(i) Any Sep calls that $\mathrm{Sep}(X)$ reduces to are with a smaller argument (with respect to the order defined in Definition 3.46). That is, $\mathrm{Sep}(X)$ terminates.

(ii) Let A be a simple and non-past subformula of $\mathrm{Sep}(X)$. If X has no $\mathcal{S}$ inside an $\mathcal{U}$ then A is a subformula of X.

(iii) $X \equiv \mathrm{Sep}(X)$

(iv) $\mathrm{Sep}(X)$ is separated.

PROOF. We proceed by induction on $<$.

Notice that Sep exhaustively matches on formulas:

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

- Case 0 covers $\bot$, $\top$ and p;
- Case 1 covers $\neg$;
- Case 2 covers $\vee$;
- Case 3 covers $\wedge$;
- Cases 4,5 cover $\mathcal{S}$;
- Case 6 covers $\mathcal{U}$,

which means that every X fits into some case. We now look at each case.

Case 0: Trivial.

Cases 1,2,3:

(i) For the binary Boolean operators, we have to show that both the left and right arguments have this property, but the proof for each is identical to the other. So we will do this only once which should cover all the cases. Let A then be the subformula in consideration.
Any path of A is also a path of X; therefore, $\mathrm{Score}_1(A) \leq \mathrm{Score}_1(X)$ and $\mathrm{Score}_0(A) \leq \mathrm{Score}_0(X)$.
Assume $\mathrm{Score}_1(A) = \mathrm{Score}_1(X)$. Then any leader of A is also one of X; hence, $\mathcal{N}(A) \leq \mathcal{N}(X)$.
It's also clear that a subformula of a Boolean operator cannot have increased $\mathcal{H}$ or $\mathcal{J}$.
But $\mathcal{V}$ is reduced.

(ii) Assume X has no $\mathcal{S}$ inside an $\mathcal{U}$. Let C be a simple and non-past subformula of $\mathrm{Sep}(X)$. For convenience, assume $C \in \mathrm{Subs}(\mathrm{Sep}(A))$, the B case being very similar. Since X has no $\mathcal{S}$ inside an $\mathcal{U}$, neither does A, and so $C \in \mathrm{Subs}(A) \subseteq \mathrm{Subs}(X)$ by the induction hypothesis.

(iii) Substitution by equivalent formula.

(iv) The induction hypothesis tells us that A and B are separated, and a Boolean combination of separated formulas is also separated.

Case 4: In all the subcases, let X' refer to the result of applying the corresponding transformation (e.g. in case 4.1, $X' = \mathcal{T}_1\big(A', B', F, G\big)$).

(i) The set $\{\langle \mathcal{S}, E\rangle \mid E \in \mathbf{A} \cup \mathbf{B} \cup \mathbf{C} \cup \mathbf{D} \cup \{F\mathcal{U}G\}\}$ is complete for X. All the E that come from $\mathbf{A}$ and $\mathbf{B}$ are non-future; therefore, in this case $\langle \mathcal{S}, E\rangle$ has degree 0 and none can be leaders. All the elements of $\mathbf{C} \cup \mathbf{D} \cup \{F\mathcal{U}G\}$ are pure future and therefore their associated contexted formulas all have degree 1. But $F\mathcal{U}G$ has maximal temporal depth among them, which means that $\langle \mathcal{S}, F\mathcal{U}G\rangle$ has a maximal final homogeneous segment and maximal $\mathcal{L}_1$, and is therefore a leader. This also implies that the degree of X is 1, by Proposition 3.51.
By Proposition 3.59, $\mathrm{Score}_1(X') \leq \mathrm{Score}_1(X)$. So let us assume $\mathrm{Score}_1(X') = \mathrm{Score}_1(X)$.
Since X has degree 1 and the outer connective of X is an $\mathcal{S}$, there is no $\mathcal{S}$ inside an $\mathcal{U}$ in X as this would imply that X has a degree at least 2. Given Proposition 3.53, any $\mathcal{U}$ in X' must already appear in X; therefore, X' also has no $\mathcal{S}$ inside an $\mathcal{U}$. Given this and the fact that X' has degree 1, X' has a leader and it must be of the form $\big\langle \mathcal{S}^{m+1}\mathcal{U}^{n}, H\big\rangle$, with H non-past.
By Lemma 3.60 there is some $\langle \lambda, H\rangle \in \mathcal{SPC}(X)$ with $\mathrm{Score}_1(\big\langle \mathcal{S}^{m+1}\mathcal{U}^{n}, H\big\rangle) \leq \mathrm{Score}_1(\langle \lambda, H\rangle)$. We have

$$\begin{aligned}
& \mathrm{Score}_1(X') \\
&= \mathrm{Score}_1\left(\left\langle \mathcal{S}^{m+1}\mathcal{U}^{n}, H\right\rangle\right) && \text{(Proposition 3.51)} \\
&\leq \mathrm{Score}_1(\langle \lambda, H\rangle) \\
&\leq \mathrm{Score}_1(X) && \text{(Proposition 3.50)} \\
&= \mathrm{Score}_1(X') && \text{(assumption)}
\end{aligned}$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Therefore, $\mathrm{Score}_1(\langle\lambda,H\rangle) = \mathrm{Score}_1(X)$, $\mathrm{Score}_1(\langle\lambda,H\rangle)$ is a leader of X and $\mathcal{N}(X') \leq \mathcal{N}(X)$. The future formula $F\mathcal{U}G$ cannot appear in X inside one of the non-future literals (i.e. the ones in **A** and **B**). And $F\mathcal{U}G$ cannot appear inside one of the future literals (i.e. the ones in **C** and **D**) since otherwise such literal would have greater temporal depth, which contradicts the fact that $F\mathcal{U}G$ has maximal temporal depth. This together with Proposition 3.53 means that the only possible occurrences of $F\mathcal{U}G$ in X' are the explicit ones, which always have an empty context. Hence, $F\mathcal{U}G$ cannot be leader of X' as that would imply that X' has degree 0. This shows that X' has lost a leader.

(ii) Any simple and non-past subformula H of X' appears there with some context. Lemma 3.60 tells us that H also appears in X.

(iii) The transformations construct equivalent formulas, i.e. $X \equiv X'$, then the induction hypothesis tells us that $X' \equiv \mathrm{Sep}(X')$. By definition of the algorithm we have $\mathrm{Sep}(X) = \mathrm{Sep}(X')$.

(iv) Direct use of the induction hypothesis.

Case 5:

(i) For the Sep calls $\mathrm{Sep}(A)$ and $\mathrm{Sep}(B)$ we can use Proposition 3.52 directly.

Let us now consider the case where A and B are already separated. In this case we have $\mathrm{Sep}(A) = A$ and $\mathrm{Sep}(B) = B$.

Normal form conversion is only over outer Boolean structure of A and B; therefore, any path of A_i is a path of A and any path of B_j is a path of B. Given this, all the paths of $A_i\mathcal{S}B_j$ are paths of $A\mathcal{S}B$, $\mathrm{Score}_1(A_i\mathcal{S}B_j) \leq \mathrm{Score}_1(A\mathcal{S}B)$ and $\mathrm{Score}_0(A_i\mathcal{S}B_j) \leq \mathrm{Score}_0(A\mathcal{S}B)$.

If we now assume that $\mathrm{Score}_1(A_i\mathcal{S}B_j) = \mathrm{Score}_1(A\mathcal{S}B)$, then neither $A_i\mathcal{S}B_j$ nor $A\mathcal{S}B$ are pure as otherwise we would be in case 0. Hence, any leader of $A_i\mathcal{S}B_j$ comes from A_i or B_j and is also a leader of $A\mathcal{S}B$, and we have $\mathcal{N}(A_i\mathcal{S}B_j) \leq \mathcal{N}(A\mathcal{S}B)$.

For what follows, keep in mind Definition 3.15.

If an atom is under a negation in any A_i then it was already under a negation in A, and each atom that appears in A_i already appeared in A, so we get that $\mathcal{H}(A_i) \leq \mathcal{H}(A)$. A similar argument shows that $\mathcal{H}(B_j) \leq \mathcal{H}(B)$. Hence, $\mathcal{H}(A_i\mathcal{S}B_j) \leq \mathcal{H}(A\mathcal{S}B)$.

If a normal form conversion moved a negation inwards then we have at least one fewer vertex under negation in $A_i\mathcal{S}B_j$ and we are done. So let us assume that both A and B were already in NNF.

Using Lemma 3.61 we get $\mathcal{J}(A_i\mathcal{S}B_j) \leq \mathcal{J}(A\mathcal{S}B)$ as the $\mathcal{S}$ has no impact on $\mathcal{J}$.

If A or B have repeated literals, then we obtain $\mathcal{V}(A_i\mathcal{S}B_j) < \mathcal{V}(A\mathcal{S}B)$ by noticing that a clause has no repeated literals and has the minimum number of vertices of any formula containing such literals, every literal of A_i appeared in A and every literal of B_j appeared in B. Assume then that neither A nor B have repeated literals.

It must be the case that there is a conjunction in the outer Boolean structure of A or a disjunction in the outer Boolean structure of B, otherwise A and B are separated conjunctive and disjunctive clauses, respectively, and we would be in case 4.

For the sake of argument, assume that there is a conjunction in A, with the B case being analogous.

We can now use Lemma 3.62 to show that no clause A_i can contain every literal of A, which shows that $\mathcal{V}(A_i) < \mathcal{V}(A)$. This implies $\mathcal{V}(A_i\mathcal{S}B_j) < \mathcal{V}(A\mathcal{S}B)$.

Now for the case where either A or B is not separated.

We first prove that $\langle \mathrm{Score}_1, \mathcal{N}, \mathrm{Score}_0\rangle$ is smaller for $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ than for $A\mathcal{S}B$.

By the induction hypothesis, both $\mathrm{Sep}(A)$ and $\mathrm{Sep}(B)$ are separated, which implies that they have degree 0. Hence, $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ has degree at most 1.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Notice that $A\mathcal{S}B$ cannot have degree 0 since otherwise we would be in case 0. If $A\mathcal{S}B$ has degree greater than 1 or if $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ has degree 0, we have reduced the degree. Assume then that both $A\mathcal{S}B$ and $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ have degree 1.

Now we show that $\mathrm{Score}_1(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) \leq \mathrm{Score}_1(A\mathcal{S}B)$.

Let $\langle \mathcal{S}\pi, C\rangle \in \mathrm{Leaders}(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B))$ with $\langle \pi, C\rangle \in \mathcal{SPC}(\mathrm{Sep}(A))$ or $\langle \pi, C\rangle \in \mathcal{SPC}(\mathrm{Sep}(B))$. Notice that every simple pure contexted subformula of $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$, and hence every leader must be of this form. Since both $\mathrm{Sep}(A)$ and $\mathrm{Sep}(B)$ have degree 0, so does $\langle \pi, C\rangle$. This means that if π has any $\mathcal{S}$ then C must be non-future, but that would be mean that $\langle \mathcal{S}\pi, C\rangle$ has degree 0, which is a contradiction since it is the leader of a degree 1 formula. Hence, $\pi = \mathcal{U}^n$ for some n.

Given this, we can walk up the syntax tree of $\mathrm{Sep}(A)$ or $\mathrm{Sep}(B)$ from the root of C until we consume all the n $\mathcal{U}$s, giving us a superformula D of C. Notice that D is simple by definition and must be non-past because otherwise would imply that either $\mathrm{Sep}(A)$ or $\mathrm{Sep}(B)$ is not separated.

Since we are assuming that $A\mathcal{S}B$ has degree 1, it cannot have a $\mathcal{S}$ inside an $\mathcal{U}$. Therefore, we can use (ii) of the induction hypothesis to conclude that D is also a subformula of A or B, appears there with some context λ and so appears in $A\mathcal{S}B$ with a context $\mathcal{S}\lambda$. Remember that $\langle \pi, C\rangle \in \mathcal{SPC}(D)$, which means that $\langle \mathcal{S}\lambda\pi, C\rangle \in \mathcal{SPC}(A\mathcal{S}B)$.

We now have

$$\mathrm{Score}_1(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) = \mathrm{Score}_1(\langle \mathcal{S}\pi, C\rangle) \leq \mathrm{Score}_1(\langle \mathcal{S}\lambda\pi, C\rangle) \leq \mathrm{Score}_1(A\mathcal{S}B).$$

Assume $\mathrm{Score}_1(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) = \mathrm{Score}_1(A\mathcal{S}B)$. Then $\mathrm{Score}_1(\langle \mathcal{S}\lambda\pi, C\rangle) = \mathrm{Score}_1(A\mathcal{S}B)$ and so $\langle \mathcal{S}\lambda\pi, C\rangle$ is a leader of $A\mathcal{S}B$, and we get $\mathcal{N}(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) \leq \mathcal{N}(A\mathcal{S}B)$ as well since we have just shown that for every possible leader of $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$, which must have the form $\langle \mathcal{S}\pi, C\rangle$, there is a matching leader of $A\mathcal{S}B$ of the form $\langle \mathcal{S}\lambda\pi, C\rangle$.

We finally show that if neither Score_1 nor $\mathcal{N}$ decrease, then Score_0 must do so.

A maximum degree path of $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ must be of the form $\mathcal{S}\mathcal{U}^{m+1}$, since $\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)$ has degree 1, $\mathrm{Sep}(A)$ and $\mathrm{Sep}(B)$ are separated and the outer connective is $\mathcal{S}$.

Hence, $\mathrm{Score}_0(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) = \langle 1, 1\rangle$.

We know that either A or B must not be separated. Assume then that A is not separated, with the other case being analogous.

Then A has some path with degree 1, and it must start with $\mathcal{S}$ as otherwise $A\mathcal{S}B$ would have degree at least 2. This implies that $A\mathcal{S}B$ has a degree 1 path starting with $\mathcal{S}\mathcal{S}$, and so the $\mathcal{L}_0$ of this path is at least 2, and the Score_0 of $A\mathcal{S}B$ is at least $\langle 1, 2\rangle$.

We still have to show that each $A_i\mathcal{S}B_j$ is smaller than $A\mathcal{S}B$.

If $A_i\mathcal{S}B_j$ is separated then we have reduced $\mathcal{D}$ since $A\mathcal{S}B$ is not separated.

If $A_i\mathcal{S}B_j$ is not separated then we can use the same argument we did at the very beginning of the proof of item (i) of case 5 to show that $\mathrm{Score}_1(A_i\mathcal{S}B_j) \leq \mathrm{Score}_1(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B))$, $\mathcal{N}(A_i\mathcal{S}B_j) \leq \mathcal{N}(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B))$ and $\mathrm{Score}_0(A_i\mathcal{S}B_j) \leq \mathrm{Score}_0(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B))$. We already know that $\mathrm{Score}_0(\mathrm{Sep}(A)\mathcal{S}\,\mathrm{Sep}(B)) < \mathrm{Score}_0(A\mathcal{S}B)$ so we are done.

(ii) First of all notice that if X has no $\mathcal{S}$ inside an $\mathcal{U}$, then neither do A, B and $A_i\mathcal{S}B_j$ (for all relevant i and j).

Let C be a simple and non-past subformula of $\mathrm{Sep}(A\mathcal{S}B)$. Then $C \in \mathrm{Subs}(\mathrm{Sep}(A_i\mathcal{S}B_j))$ for some i and j since $\mathrm{Sep}(A\mathcal{S}B)$ is a Boolean combination of formulas of this form. By the induction hypothesis, C is then a subformula of $A_i\mathcal{S}B_j$. We additionally know that C is a subformula of either A_i or B_j since C is non-past.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Normal form conversion is only over Boolean structure and does not 'make up' simple formulas; hence, C is a subformula of either $\mathrm{Sep}(A)$ or $\mathrm{Sep}(B)$.
Again by the induction hypothesis, we get that C is a subformula of either A or B and so is a subformula of $A\mathcal{S}B$.

(iii) We have $\mathrm{Sep}(A) \equiv A$ and $\mathrm{Sep}(B) \equiv B$ by the induction hypothesis. Normal form conversions produce equivalent formulas. Finally we use Corollaries 3.10 and 3.7 to perform distribution and use the induction hypothesis again for each $A_i\mathcal{S}B_j$.

(iv) By the induction hypothesis, all the $\mathrm{Sep}(A_i\mathcal{S}B_j)$ are separated, and a Boolean combination of separated formulas is also separated.

Case 6:

(i) The definitions of Score_1, $\mathcal{N}$, Score_0, $\mathcal{H}$, $\mathcal{J}$ and $\mathcal{V}$ are symmetric with respect to $\mathcal{S}$ and $\mathcal{U}$, so the Dual has identical values for these. But $\mathcal{P}$ is reduced.

(ii) Since we are here and not in case 0, $A\mathcal{U}B$ is not separated, and so there is an $\mathcal{S}$ inside the $\mathcal{U}$.

(iii)

$$\begin{aligned}
&\mathrm{Dual}(X) \equiv \mathrm{Sep}(\mathrm{Dual}(X)) && \text{(induction hypothesis)}\\
\Longleftrightarrow\ &\mathrm{Dual}(\mathrm{Dual}(X)) \equiv \mathrm{Dual}(\mathrm{Sep}(\mathrm{Dual}(X))) && \text{(proposition 3.5)}\\
\Longleftrightarrow\ &X \equiv \mathrm{Dual}(\mathrm{Sep}(\mathrm{Dual}(X))) && (X = \mathrm{Dual}(\mathrm{Dual}(X)), \text{by by proposition 15})
\end{aligned}$$

(iv) $\mathrm{Sep}(\mathrm{Dual}(X))$ is separated, by the induction hypothesis. Separation is equivalent to having degree 0, and Dual preserves degree. □

3.3 *Complexity*

In this section we show that separation of a certain fragment of temporal logic causes at most a double exponential blow-up in the size of the formula. The fragment is defined as follows:

Definition 3.64
Given a formula $A \in \mathcal{L}_{\mathrm{TL}}$, $A \in \mathcal{L}_{\mathrm{RTL}}$ if and only if the following conditions are met:

(i) There is no $\mathcal{S}$ or $\mathcal{U}$ inside a negation.

(ii) $\mathcal{S}$ and $\mathcal{U}$ do not appear on the left side of any $\mathcal{S}$ or $\mathcal{U}$.

For our current purposes we need to adjust the separation algorithm very slightly. So let us define a new separation algorithm.

Definition 3.65
The algorithm $\mathrm{Sep}_{\mathrm{RTL}}$ is defined just like Sep except that we replace every appearance of Sep with $\mathrm{Sep}_{\mathrm{RTL}}$ and we redefine the result of case 3 (conjunction) by replacing $\mathrm{Sep}(A) \wedge \mathrm{Sep}(B)$ with the following:

$$\textbf{let } \bigvee_{i=1}^{m} A_i := \mathrm{Sep}_{\mathrm{RTL}}(A)$$

$$\textbf{let } \bigvee_{j=1}^{n} B_j := \mathrm{Sep}_{\mathrm{RTL}}(B)$$

$$\textbf{return } \bigvee_{i=1}^{m}\left(\bigvee_{j=1}^{n}\left(A_i \wedge B_j\right)\right)$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Where the outer disjunctions described here are obtained by considering only the outer disjunctions, i.e. disjunctions that do not occur inside any other operator other than another disjunction.

This can always be done since if there are no outer disjunctions we can just take m and/or n to be 1.

Notice that this change to case 3 does not affect our ability to prove properties of Sep_{RTL} using induction over the same order as before, as the recursive calls have not changed.

PROPOSITION 3.66
For any formula $X \in \mathcal{L}_{\text{TL}}$ we have the following:

(i) $\text{Sep}_{\text{RTL}}(X)$ terminates
(ii) $\text{Sep}_{\text{RTL}}(X) \equiv X$
(iii) $\text{Sep}_{\text{RTL}}(X)$ is separated

PROOF. Very similar to the proof of Theorem 3.63. With minor changes when considering case 3.□

DEFINITION 3.67
Given a formula $A \in \text{TL}$, $\text{t}(A)$ is the number of temporal operators ($\mathcal{S}$ and $\mathcal{U}$) that occur in A.

PROPOSITION 3.68
Let $A \in \mathcal{L}_{\text{RTL}}$. Then $\text{Sep}_{\text{RTL}}(A)$ is a disjunction of formulas of the form $B \in \mathcal{L}_{\text{RTL}}$ such that $t(B) \leq t(A)$.

PROOF. This will proceed by induction on the same order on formulas as we used in the correctness proof of Sep. Again we treat each case separately.

Case 0: Trivial.

Case 1: Since A is inside a negation, it cannnot have any occurrences of $\mathcal{S}$ or $\mathcal{U}$ and so is already separated. This means that $\text{Sep}(A) = A$ and $\text{Sep}(X) = X$.

Case 2: By the induction hypothesis $\text{Sep}(A)$ is a disjunction of formulas of the form $C \in \mathcal{L}_{\text{RTL}}$ such that $\text{t}(C) \leq \text{t}(A)$ and $\text{Sep}(B)$ is a disjunction of formulas of the form $D \in \mathcal{L}_{\text{RTL}}$ such that $\text{t}(D) \leq \text{t}(B)$. Since we have $\text{t}(A \vee B) = \text{t}(A) + \text{t}(B)$, we then also get $\text{t}(C) \leq \text{t}(A \vee B)$ and $\text{t}(D) \leq \text{t}(A \vee B)$.

Case 3: By the induction hypothesis both $\text{Sep}(A)$ and $\text{Sep}(B)$ are disjunctions of formulas with the desired property. Now notice that a subformula never has a greater number of temporal operators and so we know that $\text{t}(A_i) \leq \text{t}(A)$ and $\text{t}(B_j) \leq \text{t}(B)$. Now we have $\text{t}(A_i \wedge B_j) = \text{t}(A_i) + \text{t}(B_j) \leq \text{t}(A) + \text{t}(B) = \text{t}(A \wedge B)$.

Case 4: Remember that $X \in \mathcal{L}_{\text{RTL}}$. This means that there are no $\mathcal{U}$s inside A and so cases 4.1, 4.3, 4.4, 4.6, 4.7 and 4.8 are impossible. Case 4.5 is also impossible since it would mean that there is a negation inside a $\mathcal{S}$. This leaves only case 4.2.

Given again that there is no $\mathcal{U}$ on the left side of the $\mathcal{S}$, $\mathbf{C} = \emptyset$ and so $A' = A$.

Expanding the definition of $\mathcal{T}_2$ as used in the algorithm we get

$$H_< \vee H_= \vee H_>,$$

where

$$H_< := A\mathcal{S}\big(G \wedge A \wedge (A \wedge F)\mathcal{S}B'\big)$$

$$H_= := (A \wedge F)\mathcal{S}B' \wedge G$$

$$H_> := (A \wedge F)\mathcal{S}B' \wedge F \wedge F\mathcal{U}G$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

There is no explicit use of negation in any of these so all three satisfy condition (i) of Definition 3.64 as A,B (and thus B'),F and G already did so. In the resulting formulas there is no explicit nesting of temporal operators to the left and neither B' nor G appear on the left of a temporal operator. So to show that $H_<$, $H_=$ and $H_>$ satisfies condition (ii) we only have to worry about A and F. But we know that A and F appeared in X on the left of a $\mathcal{S}$ and $\mathcal{U}$, respectively. This means that neither A nor F contain any occurrences of $\mathcal{S}$ or $\mathcal{U}$. We have now shown that $H_<$, $H_=$ and $H_>$ are all in $\mathcal{L}_{\mathrm{RTL}}$.

Now we have $\mathrm{Sep}(H_< \vee H_= \vee H_>) = \mathrm{Sep}(H_<) \vee \mathrm{Sep}(H_=) \vee \mathrm{Sep}(H_>)$ by case 2 of the algorithm and

$$\mathrm{t}(H_<) = 2 + t(B') + t(G)$$

$$\mathrm{t}(H_=) = 1 + t(B') + t(G)$$

$$\mathrm{t}(H_>) = 2 + t(G) + t(B')$$

Given that $\mathrm{t}(X) = \mathrm{t}(A\mathcal{S}(B' \wedge F\mathcal{U}G)) = 2 + t(B') + t(G)$, by the induction hypothesis each of those three separations results in a disjunction of formulas with the desired property.

Case 5: Since $A\mathcal{S}B \in \mathcal{L}_{\mathrm{RTL}}$, A is pure present, which means $\mathrm{Sep}(A) = A$, $m = 1$ and A_1 is pure present as well.

In each B_j, a normal form atom may appear at most once. But every such atom already appears in $\mathrm{Sep}(B)$. Hence, $\mathrm{t}(B_j) \leq \mathrm{t}(Sep(B))$.

By the induction hypothesis, $\mathrm{Sep}(B)$ is a disjunction of $\mathcal{L}_{\mathrm{RTL}}$ formulas. Remember that normal form conversion does not mess with temporal operators and does not introduce new negations or move existing negations outwards; it can at most push negations inwards. This means that each $B_j \in \mathcal{L}_{\mathrm{RTL}}$.

Given that A_1 is pure present and $B_j \in \mathcal{L}_{\mathrm{RTL}}$, we get that $A_1\mathcal{S}B_j \in \mathcal{L}_{\mathrm{RTL}}$ and $\mathrm{Sep}(A_1\mathcal{S}B_j) \in \mathcal{L}_{\mathrm{RTL}}$, this last one by the induction hypothesis. Then,

$$\mathrm{t}(\mathrm{Sep}(A_1\mathcal{S}B_j)) \leq \mathrm{t}(A_1\mathcal{S}B_j) = 1 + \mathrm{t}(B_j) \leq 1 + \mathrm{t}(\mathrm{Sep}(B)) \leq 1 + \mathrm{t}(B) = \mathrm{t}(A\mathcal{S}B).$$

Case 6: Dual preserves $\mathcal{L}_{\mathrm{RTL}}$ membership and t. The induction hypothesis provides the rest. □

DEFINITION 3.69 (Normal forms).
We now define three normal forms as follows:

A formula is in $\mathcal{F}_0$ if it is of the forms $A\mathcal{S}\left(B \wedge \bigwedge_{i=1}^{n} C_i\right)$ or $A\mathcal{U}\left(B \wedge \bigwedge_{i=1}^{n} C_i\right)$, with A a pure present formula in conjunctive normal form, B a conjunction of literals and each C_i in $\mathcal{F}_0$ or $C_i = \top$.

A formula is in $\mathcal{F}_1$ if it is of the form $A \wedge \bigwedge_{i=1}^{n} X_i$, with A a pure present formula in conjunctive normal form, and each X_i in $\mathcal{F}_0$.

A formula is in $\mathcal{F}_2$ if it is a disjunction of $\mathcal{F}_1$ formulas.

PROPOSITION 3.70 (Normal form conversion).
Every RTL formula is equivalent to a formula in $\mathcal{F}_2$. Additionally, each $\mathcal{F}_1$ subformula in the $\mathcal{F}_2$ disjunction has no more temporal operators than the original formula.

PROOF. We will describe an algorithm to perform this conversion.

1. Bring negations inwards until each of their scopes contains only atoms.
2. Pull disjunctions outwards from the right side of every $\mathcal{S}$ and $\mathcal{U}$, using distribution.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

3. Rewrite the right side of every $\mathcal{S}$ and $\mathcal{U}$ into the desired form.
4. Convert the left side of every $\mathcal{S}$ and $\mathcal{U}$ into conjunctive normal form. □

PROPOSITION 3.71 (Non-equivalent $\mathcal{F}_1$ formulas bound).
Consider a finite set of variables V with $\#V = m$. There are $\mathcal{O}(3^{5\cdot 3^m \cdot k^2})$ non-equivalent separated $\mathcal{F}_1$ formulas with variables from V and k temporal operators.

PROOF. There are no more than 3^m disjunctive clauses with m variables and as such this bounds the choices for each B.

Similarly, there are no more than 2^{3^m} choices for an A.

An $\mathcal{F}_0$ formula can be thought of as tree with a vertex for each temporal operator, A and B, with exactly one A and one B for each temporal operator. Given this, an $\mathcal{F}_1$ can be seen as a tree rooted at the outer conjunction with subtrees given by the extra A and the $\mathcal{F}_0$ trees just described.

If we first ignore the A and B vertices, there are no more than $2^{(k+1)k}$ such trees, as there are $k+1$ vertices and thus $(k+1)k$ possible edges, giving us no more than $2^{(k+1)k}$ such graphs.

Putting it all together we get $2^{(k+1)k} \cdot \left(2^{3^m}\right)^{k+1} \cdot (3^m)^k$, as there are $k+1$ As and k Bs.

Now notice that $2^{(k+1)k} \cdot \left(2^{3^m}\right)^{k+1} \cdot (3^m)^k = 2^{k^2} \cdot 2^k \cdot 2^{3^m \cdot k} \cdot 2^{3^m} \cdot 3^{mk}$ and that each of these five is $\mathcal{O}(3^{3^m \cdot k^2})$ and we obtain the final result. □

PROPOSITION 3.72 ($\mathcal{F}_1$ size bound).
An $\mathcal{F}_1$ formula with at most m variables and k temporal operators has $\mathcal{O}(m \cdot k \cdot 2^m)$ vertex count.

PROOF. Let's consider the A type first. A conjunctive normal form formula with at most m variables has at most $m \cdot 2^m$ variables and so has $\mathcal{O}(m \cdot 2^m)$ vertex count; there are at most $k+1$ such formulas and so we get $\mathcal{O}(k \cdot m \cdot 2^m)$.

For the B type we have $\mathcal{O}(m \cdot k)$ using similar reasoning.

Adding them together we obtain $\mathcal{O}(m \cdot k \cdot 2^m)$. □

PROPOSITION 3.73 (Size bound).
Given any formula $X \in \mathcal{L}_{\mathrm{RTL}}$ with m variables and k temporal operators, the size of $\mathrm{Sep}_{\mathrm{RTL}}(X)$ is $\mathcal{O}(3^{8\cdot 3^m \cdot k^2})$.

PROOF. Let $Y = \mathrm{Sep}_{\mathrm{RTL}}(X)$. By Proposition 3.68, Y is a disjunction of RTL formulas each with k or fewer temporal operators (and involving a subset of the variables of X).

By Proposition 3.70, we can convert each such formula into $\mathcal{F}_2$, which gives us that Y is equivalent to a disjunction of $\mathcal{F}_1$ formulas with k or fewer temporal operators and involving a subset of the variables of X.

Given Propositions 3.71 and 3.72 we then have $\mathcal{O}(3^{5\cdot 3^m \cdot k^2})$ such formulas, each of $\mathcal{O}(m \cdot k \cdot 2^m)$ size; multiplying and doing some manipulation we obtain the final result. □

COROLLARY 3.74 (Size bound).
Given any formula $X \in \mathcal{L}_{\mathrm{RTL}}$ with size n, the size of $\mathrm{Sep}_{\mathrm{RTL}}(X)$ is $\mathcal{O}(3^{8\cdot 3^n \cdot n^2})$.

PROOF. Consider Proposition 3.73. The size of X is at least as large as both m and k. □

4 Translation

In this section we define an algorithm that given a FOMLO formula with at most one free variable, produces an equivalent temporal formula. This algorithm makes crucial use of the separation algorithm defined previously.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Intuitively, the main difficulty in performing this conversion is that in FOMLO we can always refer to any variable, no matter how many more levels of quantification there have been since the quantifier that introduced a variable, while temporal logic has a sequential nature, only allowing us to relate the 'current point' with the next. This is overcome by converting the binary predicates in FOMLO to unary ones, 'Before', 'Now' and 'After', that carry the same information, essentially converting FOMLO into truly monadic first-order logic without equality.

The ideas described in this chapter are heavily based on work found in [2].

4.1 Algorithm

Before we present the translation algorithm itself, we need some auxiliary algorithms and definitions.

The idea of the following definition is that in a pulled out formula there are no occurrences of $P(x)$ inside irrelevant quantifiers. In the translation algorithm, we convert a formula into an equivalent pulled out formula before beginning the translation process.

DEFINITION 4.1
We say that a FOMLO formula φ is *pulled out* if for every subformula of φ of the form $P(x)$, if $P(x)$ occurs inside the scope of a quantifier then the deepest quantifier inside whose scope $P(x)$ occurs binds x.

We are not going to explicitly write down an algorithm to convert formulas into pulled out formulas. This can however be done by first recursively pulling out the immediate subformulas and then, in the case of the quantifiers, performing normal form conversion (DNF for the existential quantifier and CNF for the universal quantifier) followed by bringing out of the quantifier the literals where the quantifier's bound variable does not appear free.

PROPOSITION 4.2
Given a FOMLO formula φ, $\text{Pullout}(\varphi)$ is an equivalent pulled out formula.

The following definition makes clear how we are getting rid of the binary predicates.

DEFINITION 4.3 (Extend).
Given a variable t and a formula φ, $\text{Extend}(t, \varphi)$ is a formula over a signature extended with three new unary predicate symbols Before, Now and After, obtained by applying the following substitutions to φ whenever they occur in a context where t is free:

- $(t = t) \mapsto \top$
- $(t < t) \mapsto \bot$
- $(x < t) \mapsto \text{Before}(x)$
- $(x = t) \mapsto \text{Now}(x)$
- $(t = x) \mapsto \text{Now}(x)$
- $(t < x) \mapsto \text{After}(x)$

These predicates are ordinary predicates; all we have to do is add three new symbols to Pred and then to remove these extra monadic predicates:

DEFINITION 4.4 (Unextend).
Given a separated temporal formula A over an extended signature, $\text{Unextend}(A)$ is the formula over the unextended signature obtained from A by performing the following replacements:

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

- In every pure past subformula, Before, Now and After are replaced with $\top$, $\bot$ and $\bot$, respectively.
- In every pure present subformula, Before, Now and After are replaced with $\bot$, $\top$ and $\bot$, respectively.
- In every pure future subformula, Before, Now and After are replaced with $\bot$, $\bot$ and $\top$, respectively.

We can finally define the translation algorithm.

```
procedure Translate($\varphi$)
    return Translate$'$(Pullout($\varphi$))

procedure Translate'($\varphi$)
    let $t$ be the free variable of $\varphi$ (or an arbitrary variable if $\varphi$ is a sentence)
    if $\varphi = \bot$ then return$\bot$
    else if $\varphi = \top$ then return$\top$
    else if $\varphi = P(t)$ then return$p$
    else if $\varphi = (t = t)$ then return$\top$
    else if $\varphi = (t < t)$ then return$\bot$
    else if $\varphi = \neg\alpha$ then return$\neg$Translate$'(\alpha)$
    else if $\varphi = \alpha \vee \beta$ then returnTranslate$'(\alpha) \vee$ Translate$'(\beta)$
    else if $\varphi = \alpha \wedge \beta$ then returnTranslate$'(\alpha) \wedge$ Translate$'(\beta)$
    else if $\varphi = \exists_s \alpha$
        let $A =$ Translate$'$(Extend$(t, \alpha)$)
        return Unextend(Sep($\blacklozenge A \vee A \vee \Diamond A$))
    else if $\varphi = \forall_s \alpha$
        let $A =$ Translate$'$(Extend$(t, \alpha)$)
        return Unextend(Sep($\blacksquare A \wedge A \wedge \Box A$))
```

4.2 Algorithm correctness

We begin by making precise the sense in which the extra predicates preserve the meaning of a formula.

DEFINITION 4.5
Let I be an interpretation structure and t_0 a point of I.

Then $\llparenthesis I \rrparenthesis_{t_0}$ is a structure over the extended signature, identical to I over the regular signature, and where

$$\text{Before}^{\llparenthesis I \rrparenthesis_{t_0}} = \left\{ x \mid x <^I t_0 \right\}$$

$$\text{Now}^{\llparenthesis I \rrparenthesis_{t_0}} = \{t_0\}$$

$$\text{After}^{\llparenthesis I \rrparenthesis_{t_0}} = \left\{ x \mid t_0 <^I x \right\}$$

PROPOSITION 4.6
Let I be an interpretation structure, ρ an assignment and φ a formula. Then,

$$I, \rho \Vdash \varphi \;\text{ if and only if } \llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash \text{Extend}(t, \varphi).$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

PROOF. Induction on the structure of φ.

For the cases where t does not appear free in φ, we have $\mathrm{Extend}(t,\varphi) = \varphi$, and $\llparenthesis I \rrparenthesis_{\rho(t)}$ coindices with I over the unextended signature, so the equivalence is immediate.

We now consider the cases where t appears free, in every case assume that $x \neq t$:

If $\varphi = P(t)$: $I, \rho \Vdash P(t)$ is by definition equivalent to $\rho(t) \in P^I = P^{\llparenthesis I \rrparenthesis_{\rho(t)}}$, which in turn is equivalent to $\llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash P(t) = \mathrm{Extend}(t, P(t))$.

If $\varphi = (t = t)$: Then φ is equivalent to $\top = \mathrm{Extend}(t, t = t)$.

If $\varphi = (t < t)$: Then φ is equivalent to $\bot = \mathrm{Extend}(t, t < t)$.

If $\varphi = (x < t)$: $I, \rho \Vdash (x < t)$ iff $\rho(x) <^I \rho(t)$ iff $\rho(x) \in \mathrm{Before}^{\llparenthesis I \rrparenthesis_{\rho(t)}}$ iff $\llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash \mathrm{Before}(x) = \mathrm{Extend}(t, x < t)$.

If $\varphi = (x = t)$ or $\varphi = (t = x)$ (we show only the first): $I, \rho \Vdash (x = t)$ iff $\rho(x) = \rho(t)$ iff $\rho(x) \in \mathrm{Now}^{\llparenthesis I \rrparenthesis_{\rho(t)}}$ iff $\llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash \mathrm{Now}(x) = \mathrm{Extend}(t, x = t)$.

If $\varphi = (t < x)$: iff $I, \rho \Vdash (t < x)$ iff $\rho(t) <^I \rho(x)$ iff $\rho(x) \in \mathrm{After}^{\llparenthesis I \rrparenthesis_{\rho(t)}}$ iff $\llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash \mathrm{After}(x) = \mathrm{Extend}(t, t < x)$.

The Boolean operator cases are almost immediate by the induction hypothesis.

If $\varphi = \exists_x \alpha$:

$$\begin{array}{ll} & I, \rho \Vdash \exists_x \alpha \\ \text{iff} & \text{There is some } x_0 \text{ s.t.:} I, \rho[x \mapsto x_0] \Vdash \alpha \\ \text{iff} & \text{There is some } x_0 \text{ s.t.:} \llparenthesis I \rrparenthesis_{\rho(t)}, \rho[x \mapsto x_0] \Vdash \mathrm{Extend}(t, \alpha) \qquad \text{(induction hypothesis)} \\ \text{iff} & \llparenthesis I \rrparenthesis_{\rho(t)}, \rho \Vdash \exists_x \mathrm{Extend}(t, \alpha) = \mathrm{Extend}(t, \exists_x \alpha) \end{array}$$

The universal quantifier case is very similar to the existential one. □

The following two propositions are necessary for cases 8 and 9 of the translation algorithm.

PROPOSITION 4.7
For every interpretation I, point t_0 of I, and formula $A \in \mathcal{L}_{\mathrm{TL}}$, the following are equivalent:

(i) There is some $x_0 \in I$ such that $I, x_0 \Vdash A$
(ii) $I, t_0 \Vdash \blacklozenge A \lor A \lor \lozenge A$

PROOF. Since we are working over linear orders, $x_0 < t_0$ or $x_0 = t_0$ or $x_0 > t_0$. These cases are equivalent, respectively, to $I, t_0 \Vdash \blacklozenge A$, $I, t_0 \Vdash A$ and $I, t_0 \Vdash \lozenge A$. □

PROPOSITION 4.8
For every interpretation I, point t_0 of I, and formula $A \in \mathcal{L}_{\mathrm{TL}}$, the following are equivalent:

(i) For all $x_0 \in I$: $I, x_0 \Vdash A$
(ii) $I, t_0 \Vdash \blacksquare A \land A \land \square A$

PROOF. Similar to Proposition 4.7. □

Now we move on to showing the correctness of Unextend (Definition 4.4). The following four propositions just make precise the fact that pure formulas are indeed semantically pure. That is, a pure past formula only talks about the past, and so on.

PROPOSITION 4.9
Let A be a pure present formula, I and J be interpretation structures and t_0 a point of both such that

$$t_0 \in P^I \iff t_0 \in P^J \qquad \text{(for all } P \in \mathrm{Pred})$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Then,

$$I, t_0 \Vdash A \text{ if and only if } J, t_0 \Vdash A.$$

Proof. Induction on the structure of pure present formulas. □

In the next proposition, the orders do not have to match after t_0, but these stronger conditions are sufficient for our purposes. A similar thing is true for the two following propositions.

Proposition 4.10 (The future is irrelevant for a non-future formula).
Let A be a non-future formula, I and J be interpretation structures and t_0 a point of both, where

$$\text{Domain}(I) = \text{Domain}(J)$$

$$<^I = <^J$$

$$\text{For all} s_0 \leq t_0 \text{: } s_0 \in P^I \iff s_0 \in P^J \quad (\text{for all } P \in \text{Pred})$$

Then,

$$\text{For all } s_0 \leq t_0 \text{: } I, s_0 \Vdash A \text{ if and only if } J, s_0 \Vdash A.$$

Proof. By induction on the structure of the non-future formula A:

If $A = \bot$ or $A = \top$: Trivial.

If $A = p$: $I, s_0 \Vdash p$ iff $s_0 \in P^I$ iff $s_0 \in P^J$ iff $J, s_0 \Vdash p$.

If A is a Boolean operator: straightforward use of the induction hypothesis.

If $A = B\mathcal{S}C$: $I, s_0 \Vdash B\mathcal{S}C$ if and only if there is some $s_1 \in \text{Domain}(I)$ such that $s_1 <^I s_0, I, s_1 \Vdash C$ and all for all $r_0 \in \text{Domain}(I)$, if $s_1 <^I r_0 <^I s_0$ then $I, r_0 \Vdash B$.

The domains and interpretation of the order of I and J are identical, so we can replace Domain(I) with Domain(J) and $<^I$ with $<^J$ in the previous statement to obtain something equivalent.

Since A is non-future, so are B and C, and the points s_1 and all r_0 are before s_0, and so also before t_0. Therefore, we can use the induction hypothesis to replace $I, s_1 \Vdash C$ with $J, s_1 \Vdash C$ and $I, r_0 \Vdash B$ with $J, r_0 \Vdash B$. After doing this, we indeed obtain the definition of $J, s_0 \Vdash A$. □

Proposition 4.11 (Only the past is relevant for a pure past formula).
Let A be a pure past formula, I and J be interpretation structures and t_0 a point of both, where

$$\text{Domain}(I) = \text{Domain}(J)$$

$$<^I = <^J$$

$$\text{For all} s_0 < t_0 \text{: } s_0 \in P^I \iff s_0 \in P^J \quad (\text{for all } P \in \text{Pred})$$

Then,

$$\text{For all } s_0 < t_0 \text{: } I, s_0 \Vdash A \text{ if and only if } J, s_0 \Vdash A.$$

Proof. The proof is by induction on the structure of A. It is quite similar to the proof of Proposition 4.10, except in the $\mathcal{S}$ case the B and C are not necessarily pure past.

Assume then that we are in the same situation as the $\mathcal{S}$ case of the proof of Proposition 4.10, with $I, s_0 \Vdash B\mathcal{S}C$. We know that all predicates match between the structures at all points strictly before t_0 and that $s_0 < t_0$; therefore, all predicates match at and before s_0, and we can use Proposition 4.10 itself (with $t_0 := s_0$) instead of the induction hypothesis. □

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

We are in fact only going to make use of Proposition 4.10 in the proof of Proposition 4.11, so we omit the dual of Proposition 4.10 and show only the dual of Proposition 4.11.

PROPOSITION 4.12 (Only the future is relevant for a pure future formula).
Let A be a pure future formula, I and J be interpretation structures and t_0 a point of both, where

$$\text{Domain}(I) = \text{Domain}(J)$$

$$<^I = <^J$$

$$\text{For all} s_0 > t_0\text{: } s_0 \in P^I \iff s_0 \in P^J \qquad (\text{for all } P \in \text{Pred}).$$

Then,

$$\text{For all } s_0 > t_0\text{: } I, s_0 \Vdash A \text{ if and only if } J, s_0 \Vdash A.$$

PROOF. We have

- $\text{Dual}(A)$ is pure past
- $\text{Domain}(I^{\text{op}}) = \text{Domain}(I) = \text{Domain}(J) = \text{Domain}(J^{\text{op}})$
- $<^{I^{\text{op}}} = >^I = >^J = <^{J^{\text{op}}}$
- For alls_0 $<^{I^{\text{op}}} t_0$: $s_0 >^I t_0$ and so $s_0 \in P^{I^{\text{op}}} = P^I \iff s_0 \in P^J = P^{J^{\text{op}}}$ (for all $P \in \text{Pred}$)

We can therefore use Propositions 3.4 and 4.11 and conclude $I, s_0 \Vdash A$ iff $I^{\text{op}}, s_0 \Vdash \text{Dual}(A)$ iff $J^{\text{op}}, s_0 \Vdash \text{Dual}(A)$ iff $J, s_0 \Vdash A$. □

PROPOSITION 4.13
For every interpretation I, point t_0 of I, and separated temporal formula A over an extended signature:

$$\llparenthesis I \rrparenthesis_{t_0}, t_0 \Vdash A \text{ if and only if } I, t_0 \Vdash \text{Unextend}(A).$$

PROOF. We show this for a simple pure formula. The full result can be obtained by induction over the Boolean structure. The proof is by applying Propositions 4.11, 4.9 and 4.12. We show only the pure past case.

Consider the interpretation structure (over the extended signature) J that is identical to $\llparenthesis I \rrparenthesis_{t_0}$ except for the interpretations of Before, Now and After, which are

$$\text{Before}^J = \text{Domain}(J);$$

$$\text{Now}^J = \emptyset;$$

$$\text{After}^J = \emptyset.$$

At all points before t_0, these coincide with the interpretations given by $\llparenthesis I \rrparenthesis_{t_0}$; therefore, Proposition 4.11 gives us $\llparenthesis I \rrparenthesis_{t_0}, t_0 \Vdash A$ is equivalent to $J, t_0 \Vdash A$.

But, in J, Before is always true, while Now and After are always false. So they can be replaced by $\top$, $\bot$ and $\bot$, respectively, which is precisely what Unextend does. Hence, $J, t_0 \Vdash A$ if and only if $J, t_0 \Vdash \text{Unextend}(A)$.

Remember that $\text{Unextend}(A)$ is over the unextended signature, and $\llparenthesis I \rrparenthesis_{t_0}$ and I are identical except for their interpretation of the extended predicates, the same being true of $\llparenthesis I \rrparenthesis_{t_0}$ and J. Hence, I and J are also identical over the unextended signature, and so $J, t_0 \Vdash \text{Unextend}(A)$ iff $I, t_0 \Vdash \text{Unextend}(A)$. □

Now follows the well-founded relation we will be using in the proof of correctness of the translation algorithm.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

DEFINITION 4.14
We define a partial order $<$ on FOMLO formulas as the transitive closure of the union of the following partial orders:

(i) α is smaller than β if it has less quantifier depth.
(ii) α is smaller than β if it is a proper subformula of β.

PROPOSITION 4.15
The order defined in Definition 4.14 is well founded.

PROOF. First notice that (i) and (ii) are both well founded.

Given that a subformula never has increased quantifier depth, it can be shown that the union of (i) and (ii) is well founded. It's also known that the transitive closure of a well-founded relation is well founded. □

We can finally prove the correctness of translation.

PROPOSITION 4.16
For every pulled out FOMLO formula $\varphi(t)$ with at most one free variable, we have the following:

(i) Any Translate$'$ calls that Translate$'(\varphi)$ reduces to are with a smaller argument (i.e. Translate$'(\varphi)$ terminates).
(ii) For all interpretations I and points t_0 of I:

$$I, [t \mapsto t_0] \Vdash_{\mathrm{FOMLO}} \varphi \text{ if and only if } I, t_0 \Vdash_{\mathrm{TL}} \mathrm{Translate}'(\varphi).$$

PROOF. We proceed by induction on the well-founded partial order defined in Definition 4.14.

Notice that Translate$'$ exhaustively matches on formulas, so let us examine each case.

Cases 0,1,2,3,4: Trivial.

Cases 5,6,7:

(i) All calls are with proper subformulas.
(ii) Straightforward use of the induction hypothesis.

Case 8:

(i) Any occurrence of $P(t)$ (for every $P \in \mathrm{Pred}$) in φ must be inside the scope of the $\exists_x$; therefore, since φ is pulled out, this $P(t)$ must occur inside a quantifier that binds t, and so this t does not occur free. Moreover, Extend removes all free occurrences of t in a binary predicate. Therefore, Extend(t, α) has no free occurrences of t. It then has at most one free variable (s) and the quantifier depth is one fewer.
(ii)

$$\begin{array}{lll}
 & I, [t \mapsto t_0] \Vdash \exists_s \alpha(t, s) & \\
\text{iff} & \text{there is some } s_0 \text{ s.t.:} I, [t \mapsto t_0, s \mapsto s_0] \Vdash \alpha(t, s) & \\
\text{iff} & \text{there is some } s_0 \text{ s.t.:} \llparenthesis I \rrparenthesis_{t_0}, [t \mapsto t_0, s \mapsto s_0] \Vdash \mathrm{Extend}(t, \alpha)(s) & \text{(Proposition 92)} \\
\text{iff} & \text{there is some } s_0 \text{ s.t.:} \llparenthesis I \rrparenthesis_{t_0}, s_0 \Vdash_{\mathrm{TL}} A & \text{(induction hypothesis)} \\
\text{iff} & \llparenthesis I \rrparenthesis_{t_0}, t_0 \Vdash_{\mathrm{TL}} \blacklozenge A \vee A \vee \Diamond A & \text{(Proposition 93)} \\
\text{iff} & \llparenthesis I \rrparenthesis_{t_0}, t_0 \Vdash_{\mathrm{TL}} \mathrm{Sep}(\blacklozenge A \vee A \vee \Diamond A) & \text{(Theorem 75)} \\
\text{iff} & I, t_0 \Vdash_{\mathrm{TL}} \mathrm{Unextend}(\mathrm{Sep}(\blacklozenge A \vee A \vee \Diamond A)) & \text{(Proposition 99)}
\end{array}$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

Case 9:

(i) Similar to case 8.

(ii) Similar to case 8, using Proposition 4.8 instead of Proposition 4.7. □

THEOREM 4.17

For every $\varphi(t) \in \mathcal{L}_{\text{FOMLO}}$ with at most one free variable, interpretation structure I and point t_0 of I:

$$I, [t \mapsto t_0] \Vdash_{\text{FOMLO}} \varphi \text{ if and only if } I, t_0 \Vdash_{\text{TL}} \text{Translate}(\varphi).$$

PROOF.

$$\begin{array}{llr} & I, [t \mapsto t_0] \Vdash_{\text{FOMLO}} \varphi & \\ \text{iff} & I, [t \mapsto t_0] \Vdash_{\text{FOMLO}} \text{Pullout}(\varphi) & \text{(Proposition 88)} \\ \text{iff} & I, t_0 \Vdash_{\text{TL}} \text{Translate}'(\text{Pullout}(\varphi)) = \text{Translate}(\varphi) & \text{(Proposition 102)} \end{array}$$

□

4.3 Extensions

4.3.1 LTL The translation algorithm can also be adapted to translate FOMLO into LTL. But then we require that the order have a minimum element, and the equivalence only holds at that minimum element. This is quite natural as LTL cannot talk about the past since it has no past operators.

This can be done by performing the translation as before and then simply replacing any past formulas with $\bot$.

```
procedure ToLTL($X$)
(ii)  0 if $X$ is non-past then return $X$
      1 else if $X = \neg A$ then return $\neg$ToLTL($A$)
      2 else if $X = A \vee B$ then return ToLTL($A$) $\vee$ ToLTL($B$)
      3 else if $X = A \wedge B$ then return ToLTL($A$) $\wedge$ ToLTL($B$)
      4 else if $X = A\mathcal{S}B$ then return $\bot$
```

```
procedure TranslateLTL($\varphi$)
    return ToLTL(Translate($\varphi$))
```

PROPOSITION 4.18

For every FOMLO formula φ, Translate(φ) is separated.

PROOF. We have to show that Translate′ always produces a separated formula. This can be shown by induction on the same order used to show the correctness of Translate′, defined in Definition 4.14.

Cases 0, 1, 2, 3 and 4 are immediate. Cases 5, 6 and 7 are a straightforward use of the induction hypothesis. For cases 8 and 9, the result is obtained by calling Unextend, which preserves separation, on the output of Sep, which produces a separated formula by Theorem 3.63. □

PROPOSITION 4.19

Let $X \in \mathcal{L}_{\text{TL}}$ be separated and I an interpretation structure with t_0 its minimum point. Then,

$$I, t_0 \Vdash_{\text{TL}} X \text{ if and only if } I, t_0 \Vdash_{\text{LTL}} \text{ToLTL}(X).$$

PROOF. First notice that ToLTL matches exhaustively on X. $\bot$, $\top$, p and $A\mathcal{U}B$ are included in case 0, this last one because $A\mathcal{U}B$ is separated (by assumption) and so is non-past.

By induction on the Boolean structure, we can see that ToLTL terminates.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

And again by induction on the Boolean structure we can show that X and ToLTL(X) agree at t_0, noticing that as t_0 is the minimum point, $A\mathcal{S}B$ can never be true there. □

PROPOSITION 4.20
For every $\varphi(t) \in \mathcal{L}_{\mathrm{FOMLO}}$ with at most one free variable and all interpretations I with a minimum point t_0:

$$I, [t \mapsto t_0] \Vdash_{\mathrm{FOMLO}} \varphi \text{ if and only if } I, t_0 \Vdash_{\mathrm{LTL}} \mathrm{TranslateLTL}(\varphi).$$

PROOF.

$$\begin{array}{lll} & I, [t \mapsto t_0] \Vdash_{\mathrm{FOMLO}} \varphi & \\ \text{iff} & I, t_0 \Vdash_{\mathrm{TL}} \mathrm{Translate}(\varphi) & \text{(Theorem 103)} \\ \text{iff} & I, t_0 \Vdash_{\mathrm{LTL}} \mathrm{ToLTL}(\mathrm{Translate}(\varphi)) = \mathrm{TranslateLTL}(\varphi) & \text{(Proposition 105)} \end{array}$$

□

4.3.2 Expressively complete temporal logics We have described a translation algorithm from FOMLO to the particular temporal logic we have been calling TL. But this translation algorithm in fact only requires that the temporal logic in question be separable and able to express ⧫ and ◊. It makes no direct use of $\mathcal{S}$ and $\mathcal{U}$.

We actually do not even need separation in the sense described in this text, called *syntactic separation*. A weaker notion of separation, the one described by Propositions 4.11, 4.9 and 4.12, is all that is required. Additionally, the translation algorithm also makes no use of the fact that the linear orders are complete or discrete. This means that the very same algorithm can be used to translate from FOMLO to other temporal logics, over any class of linear orders, provided they meet these conditions.

We now consider general temporal logics. We say that such a logic can express ⧫ if for every formula A in that logic there is a (computable) formula B such that $⧫A \equiv B$, and similarly for ◊.

DEFINITION 4.21
We say that a formula A, in any temporal logic, is *predicatively purely past* if it meets the conditions of Proposition 4.11. That is, a formula A is predicatively purely past if for all interpretation structures I,J and point t_0 of both that meet the three conditions of Proposition 4.11, the conclusion of Proposition 4.11 is true for A.

We define *predicatively purely present* and *predicatively purely future* similarly, using Propositions 4.9 and 4.12, respectively.

DEFINITION 4.22
A temporal logic formula is *predicatively separated* if it is a Boolean combination of predicatively pure formulas.

PROPOSITION 4.23
Let L be a temporal logic able to express ⧫ and ◊ and Sep an algorithm that predicatively separates L over a class of linear orders $\mathcal{C}$. Let I be an interpretation structure whose interpretation of $<$ is in $\mathcal{C}$ (not necessarily complete or discrete). Then, for every $\varphi \in \mathcal{L}_{\mathrm{FOMLO}}$ and point t_0 of I:

$$I, [t \mapsto t_0] \Vdash_{\mathrm{FOMLO}} \varphi \text{ if and only if } I, t_0 \Vdash_{L} \mathrm{Translate}(\varphi).$$

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

PROOF. The proof is the same as the proof of Theorem 103, noticing that the conditions on the logic imposed here are all that is used, and no other properties of the specific TL we have been using are required. □

5 Conclusions

We have written fairly simple algorithms to perform separation of TL and translation from FOMLO to TL over complete and discrete time. Since the actual translation algorithm is quite agnostic about the temporal logic used, it can also be used to translate to any expressively complete logic with computable separation. It is also possible to eliminate the discreteness requirement by extracting an algorithm from the proof of separation for complete time found in [3], perhaps even write a simple algorithm to perform this separation.

We have not talked about the complexity of these algorithms so far, but it is in fact known that the translation algorithm must have non-elementary complexity. As explained in [5] and [1], it is known that there is a non-elementary succinctness gap between FOMLO and TL, i.e. there are FOMLO formulas whose smallest equivalent TL formula is non-elementarily larger. This means that a translation algorithm must also have non-elementary complexity. Another way to see this is that the decidability of FOMLO even over the naturals is known to be non-elementary (see [12]), while the decidability of TL is in PSPACE (see [11]). This implies that a translation from FOMLO to TL must be non-elementary as well, as otherwise we would be able to decide FOMLO with an elementary algorithm by translating to TL and back.

Although the complexity of translation is non-elementary, in empirical testing using large amounts of randomly generated formulas of the kind of size, number of quantifiers and quantifier depth at the upper limit of that would be written by a human in practice, the speed of execution seems good enough for use in practice.

As far as we know, the worst-case complexity of an algorithm that performs either syntactic or semantic separation for any LTL is still not known. It is suggested throughout the literature (and sometimes outright claimed) that Gabbay's algorithm for separation has non-elementary worst-case complexity, but we have been unable to find a proof of this. If this is indeed true, it would be interesting not only to find a family of formulas which results in this behaviour for these specific algorithms, but also to show that algorithms of this type based on these transformations must necessarily have non-elementary complexity in the worst case.

Acknowledgements

We would like to express our gratitude to Cristina Sernadas for the encouragement to submit this work. The second author also acknowledges the support by National Funding from FCT - Fundação para a Ciência e a Tecnologia, under the project UIDB/04561/2020 granted to CMAFcIO (Centro de Matemática, Aplicações Fundamentais e Investigação Operacional) of Universidade de Lisboa. We greatly acknowledge the excellent working environment provided by the Department of Mathematics of Instituto Superior Técnico, Universidade de Lisboa. The authors are deeply grateful to the anonymous referees for many valuable comments and suggestions.

References

[1] K. Etessami and T. Wilke. An until hierarchy and other applications of an Ehrenfeucht–Fraïssé game for temporal logic. *Information and Computation*, **160**, 88–108, 2000.

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021

[2] D. Gabbay. The declarative past and imperative future. In *Temporal Logic in Specification: Altrincham, UK, April 8–10, 1987 Proceedings*, B. Banieqbal, H. Barringer and A. Pnueli, eds, pp. 409–448. Springer, 1989.

[3] D. M. Gabbay, I. Hodkinson and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, vol. 1. Oxford University Press, 1994.

[4] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '80*, pp. 163–173. ACM, New York, USA, 1980.

[5] I. M. Hodkinson and M. Reynolds. Separation—past, present, and future. In *We Will Show Them!*, vol. 2, pp. 117–142. College Publications, 2005.

[6] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD Thesis, University of California, 1968.

[7] N. Markey. Temporal logic with past is exponentially more succinct. *EATCS Bulletin*, **79**, 122–128, 2003.

[8] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science, SFCS '77*, pp. 46–57. IEEE Computer Society, 1977.

[9] A. N. Prior. *Time and Modality*. John Locke Lecture. Clarendon Press, 2003.

[10] A. Rabinovich. A proof of Kamp's theorem. *Logical Methods in Computer Science*, **10**, 2014.

[11] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, **32**, 733–749, 1985.

[12] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD Thesis. Massachusetts Institute of Technology, 1974.

Received 3 December 2018

Downloaded from https://academic.oup.com/jigpal/article/29/3/251/5739889 by guest on 27 June 2021