Flat fragments of CTL and CTL* : separating the expressive and distinguishing powers
Dams, D.R.

Published: 01/01/1998

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 12. Jan. 2019
Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers

by

Dennis Dams

98/05

ISSN 0926-4515

All rights reserved
editors: prof.dr. R.C. Backhouse
prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs
Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers

Dennis Dams*

Abstract

We study both the expressive and the distinguishing powers of flat temporal logics. These are fragments obtained by restricting the first argument of the Until operator to propositions. Both the linear and the branching-time cases are considered.

1 Introduction

Temporal logic lies at the basis of several specification formalisms that are widely used in practice. For a large part, this acceptance stems from the availability of software tools for automated verification, that allow to prove or disprove the satisfaction of a temporal property interpreted over a modelling of the system under consideration. Model checking is such an approach, that has proven successful in the debugging and verification of hardware circuitry and communication protocols for example. Being based on an exhaustive inspection of the state space of the model, the scalability of model checking is limited, which is referred to as the state explosion problem. One way to alleviate this problem is to "collapse" the model by identifying states that are indistinguishable through properties expressed in the temporal logic being used for specification. Obviously, the lower the distinguishing power of the logic, the better reductions can be achieved. On the other hand, the expressivity of the logic should not be compromised too much.

In this article we investigate the expressive and distinguishing powers of a number of variations on CTL (Computation Tree Logic, see [CES86]). The Until modality is used to express that some property $\varphi$ has to remain true until property $\psi$ occurs, where $\varphi$ and $\psi$ can be arbitrary temporal formulae again, expressing properties about sequences. We consider a restriction where $\varphi$ is limited to a proposition stating a property about single states only — the resulting logics are called flat. This fragment is of interest because it is indeed being used in the practice of specification. For example, timing diagrams, part of a visual specification formalism ([DJS95]), are automatically translated into temporal logic prior to model checking, and it can be shown that the resulting formulae are always flat.

The issue of expressivity is first studied for the case of the linear-time temporal logic LTL. We show in Section 2 that by flattening LTL, its expressive power decreases. In Section 3 we turn to variations of the branching-time logics CTL* and CTL. The result on linear-time expressivity is shown to carry over to the branching-time cases. Then, we investigate the distinguishing powers of flat versions of CTL* and CTL by linking them to "adequate" behavioural equivalences. These equivalences are then

*Dept. of Math. & Comp. Science, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven. E-mail: wdam@dwin.tue.nl. Web: http://www.win.tue.nl/win/cs/fm/Dennis.Dams/. Part of this research was carried out during the author's participation in the Special Year on Logics and Algorithms, sponsored by the Center for Discrete Mathematics & Theoretical Computer Science (DIMACS) at Rutgers University, NJ, during the summer of 1996.
compared to each other and to those induced by the non-flat versions of the logics. Section 4 concludes.

Comparative expressivity of CTL-like temporal logics is studied in, among others, [EH86, GK94, EW96]. Behavioural equivalences ([DN87]) induced by temporal logics are the subject of [HM80, BCG88, Stu89, Jos90, DNV90, BFG+91, GKP92, vBvES94, Dam96].

2 Flat Linear-time Temporal Logic: Expressivity

Throughout this article, we assume given a non empty set Prop of propositions.

2.0.1 Definition The logic LTL is the set of formulae \( \varphi \) defined inductively by the following grammar, where \( p \in \text{Prop} \).

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid X\varphi \mid U(\varphi, \varphi).
\]

LTL(U) is the fragment of LTL obtained by disallowing the use of the Next operator X. flatLTL(U) is obtained by restricting the first argument of the U operator to a local formula, i.e. a boolean combination of propositions. The abbreviations true, false, \( \lor \), \( \rightarrow \), etc. are defined as usual.

Given a formula \( \varphi = U(\varphi_1, \varphi_2) \), \( \varphi_2 \) is sometimes referred to as the eventuality of \( \varphi_1 \), and \( \varphi_1 \) as its initial invariant.

2.0.2 Definition For \( \varphi \in \text{LTL} \), Udepth(\( \varphi \)) is the maximal number of nested U operators in \( \varphi \). I.e. Udepth(\( p \)) = 0 for \( p \in \text{Prop} \), Udepth(\( \neg \varphi \)) = Udepth(\( X\varphi \)) = Udepth(\( \varphi_1 \land \varphi_2 \)) = max(Udepth(\( \varphi_1 \)), Udepth(\( \varphi_2 \))), and Udepth(\( U(\varphi_1, \varphi_2) \)) = 1 + max(Udepth(\( \varphi_1 \)), Udepth(\( \varphi_2 \))).

Let \( \Sigma \) be a set of states and \( \mathcal{L} : \Sigma \to \mathcal{P}(\text{Prop}) \) a labelling function indicating which propositions hold in each state. For a sequence \( X = s_0s_1 \ldots \) of states, \( X(k) \) denotes the state \( s_k \) and \( X(k, \ldots) \) denotes the sequence \( s_ks_{k+1}\ldots \). We sometimes identify states with one-element sequences. Note that, e.g., \( X(k, \ldots)(i) = X(k + i) \) and \( X(k, \ldots)(i, \ldots) = X(k + i, \ldots) \). If \( X \) is finite and \( Y \) is also a sequence of states, then \( XY \) denotes their concatenation. Parentheses are used to disambiguate expressions like \( XY(k) \). \( |X| \) denotes the length of \( X \), so e.g. \( (AA)(A) = A(0) \).

We interpret LTL formulae over (single) infinite sequences of states, as follows.

2.0.3 Definition Let \( A \) be an infinite sequence of states, \( p \in \text{Prop} \), and \( \varphi \in \text{LTL} \).

1. \( A \models p \) iff \( p \in \mathcal{L}(A(0)) \).
2. \( A \models \neg \varphi \) iff \( A \not\models \varphi \).
3. \( A \models \varphi_1 \land \varphi_2 \) iff \( A \models \varphi_1 \) and \( A \models \varphi_2 \).
4. \( A \models X\varphi \) iff \( A(1, \ldots) \models \varphi \).
5. \( A \models U(\varphi_1, \varphi_2) \) iff \( \exists i \geq 0 \ A(i, \ldots) \models \varphi_2 \) and \( \forall 0 \leq j < i \ A(j, \ldots) \models \varphi_1 \).

In this section we focus on LTL(U). While the Next operator provides the power to precisely count states in a sequence, the Until operator alone allows to count certain patterns. The following lemma states that every pattern counted requires an additional Until: it is not possible to distinguish numbers of patterns that exceed the Until depth. In this lemma, the patterns are specified by a finite sequence \( A \), and the counting formula is restricted to flatLTL(U).

\footnote{See [Lam83] for a plea against the Next operator from the point of view of specification of systems.}
2.0.4 Lemma. Let $A$ be a finite sequence of states and $X$ an infinite sequence of states. Let $\varphi \in \text{flat-LTL}(U)$. Then for every $h > Udepth(\varphi)$, and every $0 \leq k < |A|$, we have $(A^h X)(k, \ldots) \models \varphi \iff (A^{h+1} X)(k + |A|, \ldots) \models \varphi$.

Proof. We prove the following, equivalent, fact: for every $h > Udepth(\varphi)$, and every $0 \leq k < |A|$, we have $(A^h X)(k, \ldots) \models \varphi \iff (A^{h+1} X)(k + |A|, \ldots) \models \varphi$.

By induction on the structure of $\varphi$. Consider the base case, where $\varphi \in \text{Prop}$ and hence $Udepth(\varphi) = 0$. Let $h > 0$ and $0 \leq k < |A|$. Clearly, the valuation of $\varphi$ over the sequences $(A^h X)(k, \ldots)$ and $(A^{h+1} X)(k + |A|, \ldots)$ only depends on the (propositions labelling the) states $(A^h X)(k)$ and $(A^{h+1} X)(k + |A|)$ respectively. Because $h > 0$ and $0 \leq k < |A|$, we have $(A^h X)(k) = (A^{h+1} X)(k + |A|)$ — namely, both states are equal to $A(k)$ — and therefore $(A^h X)(k, \ldots) \models \varphi$ iff $(A^{h+1} X)(k + |A|, \ldots) \models \varphi$.

Next, we consider the induction step. We concentrate on the case that $\varphi = U(q, \varphi')$; the other cases are straightforward (recall that $\varphi \in \text{flat-LTL}(U)$ and hence cannot contain $X$’s). Let $h > Udepth(\varphi)$ (so $h \geq 2$ and $0 \leq k < |A|$). We first prove the $\Rightarrow$ direction of the biimplication. Assume that $(A^{h+1} X)(k + |A|, \ldots) \models \varphi$. By definition of satisfaction, this means that we can choose $l \geq 0$ such that $(A^h X)(k, \ldots)(l, \ldots) \models \varphi'$ and for every $0 \leq i < l$, $(A^h X)(k, \ldots)(i) \models q$. We consider the following 3 cases, distinguishing whether the eventual truth $\varphi'$ is fulfilled in some state of the first $A$, the second $A$, or later.

1. $k + l < |A|$. Then by the i.h., $(A^h X)(k + l + |A|, \ldots) \models q$, i.e. $(A^h X)(k + |A|, \ldots)(l, \ldots) \models q$. By $k + l < |A|$ and the fact that $h \geq 1$, it easily follows that $(A^h X)(k + |A| + i) = (A^{h+1} X)(k + i)$ for every $0 \leq i < l$. Therefore we also have $(A^h X)(k + |A|, \ldots)(l, \ldots) \models q$ for every $0 \leq i < l$. So $(A^{h+1} X)(k + |A|, \ldots) \models \varphi$.

2. $|AA| \leq k + l < |AAA|$. Then $(A^{h-1} X)(k + l - |AA|, \ldots) \models q'$ and therefore, by the i.h. (note that $h - 1 > Udepth(\varphi')$), $(A^{h-1} X)(k + l - |AA|, \ldots) \models q'$, hence $(A^{h-1} X)(k + |AA| + i)(\ldots) \models q'$. By $k + l - |AA| < |AAA|$ and the fact that $h \geq 2$, it easily follows that $(A^h X)(k + |AA| + i) = (A^{h+1} X)(k + i)$ for every $0 \leq i < l$. Therefore we have $(A^h X)(k + |AA|, \ldots)(i) \models q$ for those $i$. So $(A^{h+1} X)(k + |AA|, \ldots) \models \varphi$.

3. $|AAA| \leq k + l < |AAA|$. We can choose $l' \geq 0$ such that $(A^h X)(k, \ldots)(l', \ldots) \models q'$, namely $l' := l + |A|$. From the fact that for every $0 \leq i < l'$, $(A^h X)(k, \ldots)(i) \models q$ holds, it follows directly that for every $0 \leq i < l'$, $(A^h X)(k + |AA|, \ldots)(i) \models q$ holds. We conclude that $(A^{h+1} X)(k + |AA|, \ldots) \models \varphi$.

We proceed with the $\Leftarrow$ direction. Assume that $(A^{h+1} X)(k + |A|, \ldots) \models \varphi$. By definition of satisfaction, this means that we can choose $l \geq 0$ such that $(A^h X)(k + |A|, \ldots)(l, \ldots) \models q'$ and for every $0 \leq i < l$, $(A^h X)(k, \ldots)(i) \models q$. We consider the following 3 cases, distinguishing whether the eventuality is fulfilled in some state of the second $A$, the third $A$, or later.

1. $k + |A| + l < |AA|$. Then by the i.h., $(A^{h} X)(k + l, \ldots) \models q'$, i.e. $(A^{h} X)(k, \ldots)(l, \ldots) \models q'$. Because $k + l < |A|$ and $h \geq 1$, it easily follows that $(A^h X)(k + l + i) = (A^{h+1} X)(k + |AA| + i)$ for every $0 \leq i < l$. Therefore we also have $(A^h X)(k, \ldots)(i) \models q$ for every $0 \leq i < l$. So $(A^{h+1} X)(k, \ldots) \models \varphi$.

2. $|AA| \leq k + |A| + l < |AAA|$. Then $(A^{h-1} X)(k + l - |AA|, \ldots) \models q'$ and therefore, by the i.h. (note that $h - 1 > Udepth(\varphi')$), $(A^{h-1} X)(k + l - |AA|, \ldots) \models q'$, hence $(A^{h-1} X)(k, \ldots) \models q'$. Because $k + l - |AA| < |AAA|$ and the fact that $h \geq 2$, it easily follows that $(A^h X)(k + l - |AA|) = (A^{h+1} X)(k + i)$ for every $0 \leq i < l$. Therefore we have $(A^h X)(k, \ldots)(i) \models q$ for those $i$. So $(A^{h+1} X)(k, \ldots) \models \varphi$.

3. $|AAA| \leq k + |A| + l$. We can choose $l' \geq 0$ such that $(A^h X)(k, \ldots)(l', \ldots) \models q'$, namely $l' := l + |A|$. From the fact that for every $0 \leq i < l'$, $(A^h X)(k + l + |AA|, \ldots)(i) \models q$ holds, it follows directly that for every $0 \leq i < l'$, $(A^h X)(k, \ldots)(i) \models q$ holds. Furthermore, for $0 \leq i < |A|$, every state $(A^h X)(k + i)$ is equal to the state $(A^{h+1} X)(k + |AA| + i)$ and as $q$ holds for each of the latter, it also holds for the former states. We conclude that $(A^{h+1} X)(k, \ldots) \models \varphi$. ⌜
2.1 Flattening decreases expressivity

Do we lose expressive power by flattening LTL(U)? In this section this is answered affirmatively by exposing an LTL(U) formula $\psi$ that has no flat equivalent. We start by arguing why there are no simpler such witness formulae — this gives an impression how much expressivity is lost.

The simplest candidate is the formula $U(U(p, q), r)$. However, this formula may easily be rewritten into a flat equivalent using the following property.

2.1.1 Property Let $\varphi_1, \varphi_2 \in \text{LTL}$. The formula $U(\varphi_1, \varphi_2)$ is semantically equivalent to $U(\text{true}, \varphi_2) \land \neg U(\neg \varphi_2, \neg \varphi_1 \land \neg \varphi_2)$.

By applying this property, the eventuality $r$ of the outermost Until ends up (in negated form) in the first argument. A natural next choice for a candidate witness is therefore $U(U(p, q), U(r, s))$. However, the following property gives another way to remove an Until that occurs as the first argument of another Until. Both properties can be proven by submitting them to an automatic tautology checker such as [Jan] or [Ste] for propositional linear temporal logic.

2.1.2 Property Let $\varphi_1, \varphi_2, \varphi_3 \in \text{LTL}$. The formula $U(U(\varphi_1, \varphi_2), \varphi_3)$ is semantically equivalent to $\varphi_3 \lor U(\varphi_1 \lor \varphi_2, \varphi_2 \land U(\varphi_2, \varphi_3)) \lor U(\varphi_1 \lor \varphi_2, U(\varphi_1 \land \neg \varphi_2, \varphi_3 \land U(\varphi_1, \varphi_2)))$.

This suggests that the propositions $r$ and one of $p$ and $q$ need to be replaced by Until formulae, leading to nesting depth 3 in both arguments. It turns out that this is as far as we need to go: we will now show that the LTL(U) formula

$$\psi = U(r \lor U(p, q \land U(q, r)), s \land U(s, t \land U(t, u)))$$

cannot be flattened. The following technique is used in order to prove this (cf. [EH86]). We construct two infinite sequences of models $Y_i$ and $Z_i$ such that (1) the LTL(U) formula $\psi$ is true in all $Y_i$ but false in all $Z_i$, and (2) for any flatLTL(U) formula $\bar{\psi}$, there is a large enough $k$ such that $\bar{\psi}$ cannot distinguish between $Y_i$ and $Z_i$ for $i \geq k$.

Let $p, q, r$ be propositions which are pairwise mutually exclusive (so no two of them can occur together in the label of a state), and similarly $s, t$ and $u$. Let $A$ be the sequence $ps, qu, rt$ of states, i.e. $A$ consists of 3 states the first of which has label $\{p, s\}$, the second $\{q, u\}$, and the third $\{r, t\}$. Likewise, let $B = ps, ps, rt, qu, C = qu, ps, rt, qu$ and $D$ the infinite sequence $qu, qu, \ldots$. For $i \in \mathbb{N}$, define $X_i = A^i BA^i C$, and inductively define the sequences $Y_i$ by $Y_0 = D$ and $Y_{i+1} = X_{i+1} Y_i$, and the sequences $Z_i$ by $Z_0 = Y_0$ and $Z_{i+1} = A^{i+1} CY_i$. See Figure 1.

![Figure 1: $Y_i$ and $Z_i$](image)

First, we state a property that will be used frequently.
2.1.3 \textbf{PROPERTY} Let $\varphi$ be a boolean combination of propositions and $X$ a sequence composed of the blocks $A$, $B$, $C$ and $D$. If $\varphi$ holds in every state of $A$, then $\varphi$ holds in every state of $X$.

\textbf{PROOF} It suffices to note that each state in $B$, $C$ and $D$ also occurs in $A$. \hfill $\Box$

2.1.4 \textbf{LEMMA} For every $i > 0$, $Y_i \models \psi$ while $Z_i \not\models \psi$.

\textbf{PROOF} Let $i \in \mathbb{N}$. It is easily seen that $\psi_2$ holds for any sequence that starts with prefix $B$; in particular it holds in $BA'CY_{i-1}$. Furthermore, $\psi_1$ holds in all sequences that start with prefix $A$, with prefix $A(1)$, $A(2)$ (i.e. the last two states of $A$), or with state $A(2)$. We conclude that $A'BA'CY_{i-1} \models \psi$, i.e. $Y_i \models \psi$. On the other hand, a sequence starting with $C$ does not satisfy $\psi_1$ neither $\psi_2$. Also, $\psi_2$ does not hold in any sequence that starts with prefix $A$, with prefix $A(1)$, $A(2)$, or with state $A(2)$. So $A'CY_i \not\models \psi$, i.e. $Z_i \not\models \psi$. \hfill $\Box$

Below, we will show that when we restrict ourselves to the flat fragment $\text{flatLTL}(U)$, no formula can distinguish between all $Y_i$ on the one hand and all $Z_i$ on the other. More precisely, we prove that a formula $\varphi \in \text{flatLTL}(U)$ evaluates the same over $Y_i$ and $Z_i$ for any $h$ which exceeds the Until depth of $\varphi$. The intuition is as follows. One difference between $Y_i$ and $Z_i$ is that the first non-$A$ block in $Y_i$ is $B$ while this is $C$ in $Z_i$. It is this difference that the distinguishing formula $\varphi$ brings out: the eventuality $\varphi_2$ differs between $B$ and $C$, while the initial invariance of $\varphi_1$ ensures that there may be only $A$-blocks before the first occurrence of $B$ or $C$. However, flatLTL(U) cannot specify such a complex initial invariant. The first argument of an Until operator can only assert a local property, and it follows from Property 2.1.3 that any local property that is invariant over a prefix consisting of one or more $A$-blocks indeed also holds in any state of $B$ and $C$ and hence cannot prevent such blocks from occurring among the $A$'s. The only possible way to specify in flatLTL(U) that the first non-$A$ block in $Y_i$ is $B$, is by counting down the (individual states of the) leading $A$-blocks until $B$ is reached. However, as was shown in Lemma 2.0.4, the length of the formula increases with the number of $A$-blocks to be counted. Therefore, for every formula $\varphi \in \text{flatLTL}(U)$, there is a value of $i$ which is large enough such that $\varphi$ cannot count the leading $A$-blocks.

Thus, the counterpart to Lemma 2.1.4 above is as follows.

2.1.5 \textbf{LEMMA} Let $\varphi \in \text{flatLTL}(U)$. For every $h > \text{Udepth}(\varphi)$, $Y_h \models \varphi \iff Z_h \models \varphi$.

\textbf{PROOF} Directly from Lemma 2.1.7 below, taking $h = i + 1$. \hfill $\Box$

The proof uses two sublemmata. The first states that a flatLTL(U) formula $\varphi$ cannot distinguish between a point along $Y_{i+1}$ and a corresponding point along $Y_i$, provided that $i > \text{Udepth}(\varphi)$.

2.1.6 \textbf{LEMMA} Let $\varphi \in \text{flatLTL}(U)$. Then for every $i > \text{Udepth}(\varphi)$, each of the following holds.

1. for every $0 \leq k < |A|$: $Y_{i+1}(k, \ldots) \models \varphi \iff Y_i(k, \ldots) \models \varphi$.

2. for every $|A| \leq k < |A^{i+1}BA|$: $Y_{i+1}(k, \ldots) \models \varphi \iff Y_i(k - |A|, \ldots) \models \varphi$.

3. for every $|A^{i+1}BA| \leq k < |A^{i+1}BA^{i+1}C|$: $Y_{i+1}(k, \ldots) \models \varphi \iff Y_i(k - |AA|, \ldots) \models \varphi$.

\textbf{PROOF} The proof can be found in the appendix. It is similar in spirit to the proof of Lemma 2.0.4. \hfill $\Box$

Finally, the following lemma relates (suffixes of) $Y_i$ to (suffixes of) $Z_i$.

2.1.7 \textbf{LEMMA} Let $1 \leq h \leq i + 1$ and $\varphi \in \text{flatLTL}(U)$. If $\text{Udepth}(\varphi) < h$, then for every $0 \leq k < |A|$, we have $(A^{h}BA^{i+1}CY_i)(k, \ldots) \models \varphi \iff (A^{h}CY_i)(k, \ldots) \models \varphi$.

\textbf{PROOF} See appendix. \hfill $\Box$
3 Branching-time Logics

Linear-time temporal logic is interpreted over sequences — a formula states a property about the temporal ordering of certain events (occurrences of propositions). Branching-time temporal logic, being interpreted over tree-like structures, in addition offers primitives to talk about choice points along sequences. We focus here on the family of Computation Tree Logics, that has deserved much attention in the computing science community. These logics come with quantifiers that can be used to express that a certain temporal property holds for some (or all) sequences that start from the current point in the tree. The first such logic introduced, CTL ([CES86]), bears in it a restriction on the number of temporal operators that may appear in the scope of a quantifier. This results in a model checking algorithm of low complexity. Lifting this restriction (CTL', see [EH86]) restores the expressivity to subsume LTL, but complicates the model checking problem. In this section we turn our attention to flat versions of these branching-time temporal logics.

First, we show that the expressivity results about LTL(U) vs. flatLTL(U) of the previous section imply similar results for their branching-time brothers. Second, we focus on the relative distinguishing powers of the latter logics. Whereas the expressivity of a logic is determined by the classes of models that can be characterised by (single) formulae, distinctiveness is measured by the ability of formulae to distinguish between two given models. The distinguishing power of a logic L, interpreted over models from M, is captured by the logical equivalence induced by L, \( \equiv_L \subseteq M \times M \), defined by \( s \equiv_L t \) iff \( \forall \varphi \in L \, s \models \varphi \iff t \models \varphi \). Distinguishing power is not to be confused with expressive power. Writing \( L_1 \leq L_2 \) to denote that \( L_2 \) is at least as expressive as \( L_1 \) (i.e. \( \forall \varphi_1 \in L_1 \exists \varphi_2 \in L_2 \, \varphi_1 \) is semantically equivalent to \( \varphi_2 \)), we have the following relation between expressivity and distinctivity:

**3.0.1 Lemma** \( L_1 \leq L_2 \Rightarrow \equiv_{L_1} \supseteq \equiv_{L_2} \)

**Proof** Assume that (1) \( L_1 \leq L_2 \) and \( s \equiv_{L_2} t \), i.e. (2) \( \forall \varphi_2 \in L_2 \, s \models \varphi_2 \iff t \models \varphi_2 \). We have to show that then \( s \equiv_{L_1} t \), i.e. \( \forall \varphi_1 \in L_1 \, s \models \varphi_1 \iff t \models \varphi_1 \). Let \( \varphi_1 \in L_1 \) and assume (3) \( s \models \varphi_1 \). By 1, we can choose \( \varphi_2 \in L_2 \) such that (4) \( \varphi_1 \) is semantically equivalent to \( \varphi_2 \). From 3 and 4 we have \( s \models \varphi_2 \), from which by 2, \( t \models \varphi_2 \). With 4 again, we get \( t \models \varphi_1 \). The other direction is symmetric.

The other direction of the implication does not hold. The following small example\(^2\) clarifies this. Consider the sets \( L_1 = \mathcal{P}(\mathbb{N}) \) and \( L_2 = \{ \mathbb{N} \setminus \{ k \} \mid k \in \mathbb{N} \} \) of propositions. As models over which the propositions are interpreted, take the natural numbers, defining for \( i \in \mathbb{N} \) and \( \varphi \in L_1 \), \( L_2 \): \( i \models \varphi \iff i \in \varphi \). Clearly \( L_2 \leq L_1 \). However, it is also easy to show that any two numbers that can be distinguished by \( L_1 \), can also be distinguished by \( L_2 \), implying that \( \equiv_{L_1} = \equiv_{L_2} \).

Another example, more related to the topic of this article, is the comparison between CTL\(^*\) and CTL. On the one hand, the star does increase the expressive power: in [EH86] it is shown that the CTL\(^*\) formula \( \forall F(p \land Xp) \) has no equivalent in CTL. On the other hand, as shown in [BCG88], the two logics are equally distinguishing: for both of them, the induced logical equivalence coincides with bisimulation\(^3\) ([Par81]). Below, these results will be extended for flat versions of the logics.

We start by defining the syntax and semantics of the various Computation Tree Logics.

\(^2\)Thanks to Ruurd Kuiper.

\(^3\)This correspondence is shown for finite Kripke structures in [BCG88] and can be shown to hold for image-finite Kripke structures as well — see [Dam96].
3.0.2 Definition The logic $\text{CTL}^*$ is the set of state formulae $\varphi$ defined inductively by the following grammar, where $p \in \text{Prop}$.

\[
\text{state formulae: } \quad \varphi := p \mid \neg \varphi \mid \varphi \land \varphi \mid \exists \psi \\
\text{path formulae: } \quad \psi := \varphi \mid \neg \psi \mid \psi \land \psi \mid X\psi \mid U(\psi, \psi)
\]

The abbreviations $\text{true}$, $\text{false}$, $\lor$, $\rightarrow$, $\forall$, etc. are defined as usual. In addition, we use the following abbreviations: $\text{F}\varphi$ stands for $U(\text{true}, \varphi)$, and $G\varphi$ for $\neg F\neg \varphi$. $\forall$ is used to denote the dual of $U$, so $V(q_1, q_2) \equiv \neg U(\neg q_1, \neg q_2)$. Finally, $W$, the Weak Until operator, is defined by $W(q_1, q_2) \equiv Gq_1 \lor U(q_1, q_2)$. Note that $\text{CTL}^*$ formulae without quantifiers are LTL formulae as well.

$\text{CTL}$ is the fragment of $\text{CTL}^*$ in which at most one occurrence of the temporal operators $X$ and $U$ may occur in the direct scope of any path quantifier; additional occurrences must be preceded by a new quantifier again. Formally, the defining clause, 2, of path formulae is replaced by

\[
\text{path formulae: } \quad \psi := \varphi \mid \neg \psi \mid X\psi \mid U(\psi, \psi)
\]

$\text{CTL}^*(U)$ and $\text{CTL}(U)$ are the fragments of $\text{CTL}^*$ and $\text{CTL}$ respectively obtained by disallowing the use of the Next operator $X$. $\text{flatCTL}^*(U)$ and $\text{flatCTL}(U)$ are obtained from these nextless versions by restricting the first argument of the $U$ operator to a local formula, i.e. a boolean combination of propositions.

A model for a $\text{CTL}^*$ formula is a Kripke structure $T = (\Sigma, \rightarrow, L)$ consisting of a set $\Sigma$ of states, a transition relation $\rightarrow$ that is assumed to be total (every state has a successor under $\rightarrow$), and a state-labelling function $L : \Sigma \rightarrow \mathcal{P}(\text{Prop})$. A path in $T$ is an infinite sequence $\pi = s_0s_1 \cdots$ of states such that for every $i \in \mathbb{N}, s_i \rightarrow s_{i+1}$ we say that $\pi$ starts in $s_0$. $\pi(i)$ denotes $s_i$. A subsequence of $\pi$, denoted $\pi(i, \ldots, j)$ with $i \in \mathbb{N}, j \in \mathbb{N} \cup \{\infty\}$, and $i \leq j$, is sometimes called a block (in particular, $\pi$ itself also is a block). A subsequence that starts in $s_0$ is called a prefix of $\pi$ while one that continues infinitely is called a suffix of $\pi$. A partitioning of a (sub)sequence $\delta$ is a (finite or infinite) sequence $\delta_0, \delta_1, \ldots$ of blocks whose concatenation is $\delta$. The length of $\delta$, denoted $\text{length}(\delta)$, is the number of states on it; note that the last state of $\delta$ is $\delta(\text{length}(\delta) - 1)$ if it exists. For $s \in \Sigma$, a $(T, s)$-path (or $s$-path when $T$ is clear from the context) is a path in $T$ that starts in $s$; similarly for prefixes. $\text{paths}(T, s)$ (or simply $\text{paths}(s)$) denotes the set of all $s$-paths while $\text{prefixes}(T, s)$ (or $\text{prefixes}(s)$) contains all their prefixes. The relation $\rightarrow$ is called image-finite iff for every $s \in \Sigma$, the set $\{s' \mid s \rightarrow s'\}$ has finite cardinality. $T$ is called finitely branching iff $\rightarrow$ is image-finite.

In the remainder of this section we fix a Kripke structure $T = (\Sigma, \rightarrow, L)$. The variables $s$ and $t$ range over $\Sigma$ unless stated otherwise.

3.0.3 Definition Let $p \in \text{Prop}$, $\varphi_1, \varphi_2$ be state formulae, $\psi$ a path formula, and $\pi$ a path in $T$.

Path formulae are interpreted along paths $((T, \pi) \models \psi$ or $\pi \models \psi$ for short) and state formulae in states $((T, s) \models \varphi$ or $s \models \varphi$ for short) as defined inductively by the following rules in conjunction with the rules of Definition 2.0.3.

1. $s \models p$ iff $p \in L(s)$.
2. $s \models \neg \varphi$ iff $s \not\models \varphi$.
3. $s \models \varphi_1 \land \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$.
4. $s \models \exists \psi$ iff there exists an $s$-path $\pi$ such that $\pi \models \psi$. 

5. $\pi \vDash \varphi$, where $\pi = s_0s_1 \cdots$ iff $s_0 \vDash \varphi$.

The next definition formalises a number of "compatibility relations" between temporal logics and "behavioural" equivalences on $T$.

3.0.4 Definition Let $\equiv \subseteq \Sigma \times \Sigma$ be an equivalence relation and $L$ a logic interpreted over $\Sigma$.

- $\equiv$ is fine for $L$ iff $\equiv \subseteq \equiv_L$.
- $\equiv$ is abstract for $L$ iff $\equiv \supseteq \equiv_L$.
- $\equiv$ is adequate for $L$ iff it is both fine and abstract for $\equiv_L$.

We introduce the following notation to represent transitions that "stutter" under some notion of equivalence.

3.0.5 Definition If $\equiv$ is an equivalence relation on $\Sigma$, then the transition relation $\rightarrow \subseteq \Sigma \times \Sigma$ is defined by $s \rightarrow t$ iff $s \rightarrow t \land s \equiv t$.

3.0.6 Definition For a sequence $\tilde{s}$ and an equivalence relation $\equiv$ over $\Sigma$, partit$_{\equiv}(\tilde{s})$ is the partitioning of $\tilde{s}$ into maximal blocks such that within each block, all states are $\equiv$-equivalent.

3.1 Expressivity

Given the results from Section 2.1, the difference in expressivity between $\text{CTL}^*(U)$ and $\text{CTL}(U)$ on the one hand, and their flat fragments on the other hand, is easily established.

3.1.1 Theorem $\text{CTL}^*(U)$ is strictly more expressive than $\text{flatCTL}^*(U)$, and $\text{CTL}(U)$ is strictly more expressive than $\text{flatCTL}(U)$.

Proof First, observe that for any state formula $\varphi$ in $\text{CTL}^*(U)$, and any linear model (i.e. infinite sequence) $X$, we have $X \models \varphi$ iff $X \models \text{delquant}(\varphi)$, where $\text{delquant}$ is a syntactic operation that removes all path quantifiers from $\varphi$. Now, consider the $\text{CTL}(U)$ formula $\psi' = \exists U(r \lor \exists U(p, q \land \exists U(q, r)), s \land \exists U(s, t \land \exists U(i, u)))$. Note that $\text{delquant}(\psi')$ is the formula $\psi$ from Section 2.1. So, from Lemma 2.1.4 it now follows that $\psi'$ distinguishes $Y_i$ from $Z_i$, for every $i$. Furthermore, using the same observation, it follows from Lemma 2.1.5 that there is no $\text{flatCTL}^*(U)$ formula which distinguishes all $Y_i$ from all $Z_i$. □

3.2 Distinctiveness

In this section we define and compare adequate behavioural equivalences for the logics $\text{flatCTL}^*(U)$ and $\text{flatCTL}(U)$. In order to position those results in a larger picture, we start by adapting some known results on the distinctiveness of $\text{CTL}^*(U)$ and $\text{CTL}(U)$ to our settings.

3.2.1 $\text{CTL}^*(U)$ and $\text{CTL}(U)$

The equivalences induced by branching-time logics coincide with various types of bisimulation. As we consider logics without next-state operator, we are interested in (variations on) stuttering equivalence [BCG88, DNV90].

3.2.1 Definition Let $\equiv$ be an equivalence relation on $\Sigma$. We say that $s \in \Sigma$ has infinite $\equiv$-stuttering, denoted $\text{infstat}_{\equiv}(s)$, iff there exists an $s$-path $\tilde{s}$ such that for all states $s' \text{ on } \tilde{s}$, $s' \equiv s$. 

8
3.2.2 DEFINITION Let \( \equiv \subseteq \Sigma \times \Sigma \) be a symmetric relation such that for every \( s, t \in \Sigma \), \( s \equiv t \) implies:

1. \( \mathcal{L}(s) = \mathcal{L}(t) \).
2. \( \text{infstut}_\equiv(s) \iff \text{infstut}_\equiv(t) \).
3. For every \( s \xrightarrow{} \ldots \xrightarrow{} s_{k-1} \xrightarrow{} s_k \) such that \( k \geq 0 \), there exists \( t \xrightarrow{} \ldots \xrightarrow{} t_{l-1} \xrightarrow{} t_l \) such that \( l \geq 0 \) and \( s_k \equiv t_l \).

Then \( \equiv \) is called a (divergence sensitive) stuttering equivalence (dss-equivalence). The largest stuttering equivalence is denoted \( \equiv_{\text{stut}} \).

This definition of stuttering equivalence is different from those given in [BCG88] and [DNV90], while on the side of the models, we have lifted the restriction that the Kripke structures be finite. Yet, it can be shown (see [Dam96]) that the defined equivalences coincide. Furthermore, the fineness results of [BCG88] and [DNV90] carry over to the case of infinite structures:

3.2.3 LEMMA If \( s \equiv_{\text{stut}} t \), then \( \forall \varphi \in \text{CTL}'(U) \, s \models \varphi \iff t \models \varphi \).

The converse, abstractness, only holds for Kripke structures that satisfy a certain (strong) form of finite-branchingness.

3.2.4 DEFINITION Let \( \equiv \) be an equivalence relation on \( \Sigma \). We say that \( T \) is finitely branching under \( \equiv \)-stuttering iff the reflexive transitive closure \( \xrightarrow{}^* \) of the relation \( \xrightarrow{} \) is image-finite. For a logic \( L \), “finitely branching under \( L \)-stuttering” abbreviates “finitely branching under \( \equiv_{L} \)-stuttering”.

The following property is easily proven.

3.2.5 PROPERTY Let \( \equiv_1 \) and \( \equiv_2 \) be equivalence relations on \( \Sigma \) such that \( \equiv_1 \subseteq \equiv_2 \). If \( T \) is finitely branching under \( \equiv_2 \)-stuttering then \( T \) is finitely branching under \( \equiv_1 \)-stuttering.

3.2.6 LEMMA Assume that \( T \) is finitely branching and also finitely branching under \( \text{CTL}(U) \)-stuttering. If \( \forall \varphi \in \text{CTL}(U) \, s \models \varphi \iff t \models \varphi \), then \( s \equiv_{\text{stut}} t \).

PROOF See [Dam96].

Although the condition that \( T \) is finitely branching under \( \text{CTL}(U) \)-stuttering is the weakest condition that suffices to prove the above lemma, it may be impractical to check. Note that by Property 3.2.5, it follows that finite branchingness under \( \text{CTL}(U) \)-stuttering is implied by finite branchingness under \( \text{Prop-stuttering} \).

As Lemma 3.2.6 immediately implies that \( \equiv_{\text{stut}} \) is abstract for \( \text{CTL}^+(U) \), we can conclude by the following

3.2.7 COROLLARY Assume that \( T \) is finitely branching and also finitely branching under \( \text{CTL}(U) \)-stuttering. Then \( \equiv_{\text{stut}} \) is adequate for both \( \text{CTL}^+(U) \) and \( \text{CTL}(U) \).

\( ^*\text{Note that the "vice versa" is not needed because } \equiv \text{ is required to be symmetric.} \)
3.2.2 flatCTL* (U)

Moving on to the flat versions of CTL* (U) and CTL (U), we adapt the stuttering equivalence as follows.

3.2.8 DEFINITION For s, t ∈ Σ, s ≡^0 t ⇔ ℒ(s) = ℒ(t).

3.2.9 DEFINITION Let ≡ be a symmetric relation such that for every s, t ∈ Σ, s ≡ t implies:

1. ℒ(s) = ℒ(t).
2. infstut_m(s) iff infstut_m(t).
3. For every s ∈ prefixes(s) there exists t ∈ prefixes(t) such that:
   (a) For every s_0 →_0 → ... →_0 → s_{k-1} → s_k such that 0 ≤ k < length(s), there exists t_0 →_0 → ... →_0 → t_{l-1} → t_l such that 0 ≤ l < length(t) and s_k ≡ t_l.
   (b) For every s_0 →_0 → ... →_0 → s_{k-1} → s_k such that 0 ≤ k < length(s), there exists t_0 →_0 → ... →_0 → t_{l-1} → t_l such that 0 ≤ l < length(s) and s_k ≡ t_l.

Then ≡ is called a flat (stuttering) equivalence. The largest flat star equivalence is denoted ≡_{flat*}.

The form of point 3 in this definition may be slightly surprising. One may wonder whether it could not be as follows.

3'. For every s →_0 → ... →_0 → s_{k-1} → s_k such that k ≥ 0, there exists t →_0 → ... →_0 → t_{l-1} → t_l such that l ≥ 0 and s_k ≡ t_l.

This may be clarified by considering the equivalence induced by flatCTL* (U) in game-theoretic terms. Consider states s and t and suppose that they must satisfy the same flatCTL* (U) formulae. In particular, we consider formulae of the form \( \exists \psi \), where \( \psi \) is an arbitrary path formula, which may consist of a conjunction of (negations of) smaller path formulae. If t has to satisfy formulae of the same form, then Defender must have a winning strategy to the following two-phase game:

1. Phase 1: Attacker either chooses an s-path, say \( \vec{s} \), which should be matched by the choice by Defender of a t-path, say \( \vec{t} \), or Attacker chooses a t-path, say \( \vec{t} \), which should be matched by the choice by Defender of an s-path, say \( \vec{s} \).

   This phase reflects the choice that corresponds to the \( \exists \) quantifier in the formula.

2. Phase 2: Attacker chooses either \( \vec{s} \) or \( \vec{t} \) to proceed. Denoting the result of this choice by \( \vec{u} \), Attacker then chooses a state \( \vec{u}(k) \) on \( \vec{u} \) such that for every \( 0 ≤ i < k, \vec{u}(i) ≡^0 \vec{u}(0) \). Defender now has to proceed from the other path, call it \( \vec{v} \) (so, \( \vec{v} = \vec{s} \) if \( \vec{u} = \vec{s} \) and \( \vec{v} = \vec{t} \) if \( \vec{u} = \vec{t} \)). She should match the move of Attacker with the choice of a position \( \vec{u}(l) \) on \( \vec{v} \) such that for every \( 0 ≤ i < l, \vec{v}(i) ≡^0 \vec{v}(0) \), and also \( \vec{v}(l) ≡^0 \vec{u}(k) \) (\( k = l = 0 \) is possible). The game continues from \( \vec{u}(k) \) and \( \vec{v}(l) \).

This second phase reflects the statement of an arbitrary path property: The fact that Attacker chooses either \( \vec{s} \) or \( \vec{t} \) to proceed reflects the fact that this path property may occur in positive or negated form, while the fact that all states up to \( \vec{u}(k) \) have to be \( ≡^0 \)-equivalent carries in it the restriction of Until formulae to a propositional first argument.
This game-theoretic formulation explains the inadequacy of choosing point 3' instead of 3. The problem is that Defender has to choose \( \bar{t} \) in Phase 1, without knowing which \( k \) Attacker is going to choose. In Phase 2, point 3' only guarantees that Defender can match any move of Attacker in which the choice for \( k \) is made at the same moment at which \( \tilde{s} \) is chosen.

Fineness of \( \equiv_{\text{flat}} \) for flatCTL* (\( U \)) (and hence for flatCTL(U)) is proven by an inductive argument on the structure of the formulae. Because the inductive definition of these formulae involves path formulae, flat star equivalence is extended to paths so that the induction hypothesis can be strengthened with a part stating that any two equivalent paths satisfy the same flatCTL* (\( U \)) path formulae.

3.2.10 Definition \( \equiv_{\text{flat}} \) is extended to paths by defining  
\[ \tilde{s} \equiv_{\text{flat}} \text{ for } \tilde{t} \text{ iff} \]
1. For every \( k \geq 0 \) there exists \( l \geq 0 \) such that \( \tilde{s}(k) = \tilde{t}(l) \) and furthermore, letting 
\[ \text{partit}_{\equiv}(\tilde{s}[0, \ldots, k-1]) = \tilde{s}_1, \ldots, \tilde{s}_k \text{ and partit}_{\equiv}(\tilde{t}[0, \ldots, l-1]) = \tilde{t}_1, \ldots, \tilde{t}_l, K = L \]
for every \( 0 \leq i < k \), \( k \leq j < l \), \( \tilde{s}_i \equiv_{\text{flat}} \tilde{t}_j \), and vice versa. Consider the shortest such \( t \)-prefix, \( \tilde{t} \). Clearly, all states on \( \tilde{t} \), with the exception of its last, are \( \equiv_{\text{flat}} \)-equivalent. Define \( C_0 \) to be \( \tilde{t} \)'s last state excepted, while \( C_1 \) (the first state of block \( C_1 \) to be defined) is defined to be the last state of \( \tilde{t} \). This way, we can inductively define states \( C_l \) and blocks \( C_l \) for all \( l \geq 0 \) for which \( B_l \) exists. If some \( B_l \) is infinite, then point 2 in Definition 3.2.9 guarantees the existence of an appropriate \( C_l \). It is now easily seen that for the path \( \tilde{t} \) formed by \( C_0, C_1, \ldots \), we have \( \tilde{s} = \tilde{t} \).

Fineness now follows easily.

3.2.11 Lemma If \( s \equiv_{\text{flat}} t \), then for every \( \tilde{s} \in \text{paths}(s) \) there exists \( \tilde{t} \in \text{paths}(t) \) such that \( \tilde{s} \equiv_{\text{flat}} \tilde{t} \).

Proof. Let \( \text{partit}_{\equiv}(\tilde{s}) \) be \( B_0, B_1, \ldots \). For every \( i \geq 0 \) for which \( B_i \) exists, let \( b_i \) be the first state of \( B_i \). Let \( c_0 = t \). By point 3 in Definition 3.2.9, there exists a \( t \)-prefix \( \tilde{t} \) such that for every \( 0 \leq k \leq \text{length}(B_0) \) (note that by definition of \( B_0 \), all states on it are \( \equiv_{\text{flat}} \)-equivalent, and that \( s(\text{length}(B_0)) = b_1 \), if \( \text{length}(B_0) < \omega \), there exists \( 0 \leq l \leq \text{length}(\tilde{t}) \) such that for every \( 0 \leq j < l \), \( \tilde{t}(j) \equiv_{\text{flat}} t \) and \( s(k) \equiv_{\text{flat}} \tilde{t}(l) \), and vice versa. Consider the shortest such \( t \)-prefix, \( \tilde{t} \). Clearly, all states on \( \tilde{t} \), with the exception of its last, are \( \equiv_{\text{flat}} \)-equivalent. Define \( C_0 \) to be \( \tilde{t} \)'s last state excepted, while \( C_1 \) (the first state of block \( C_1 \) to be defined) is defined to be the last state of \( \tilde{t} \). This way, we can inductively define states \( C_l \) and blocks \( C_l \) for all \( l \geq 0 \) for which \( B_l \) exists. If some \( B_l \) is infinite, then point 2 in Definition 3.2.9 guarantees the existence of an appropriate \( C_l \). It is now easily seen that for the path \( \tilde{t} \) formed by \( C_0, C_1, \ldots \), we have \( \tilde{s} = \tilde{t} \).

3.2.12 Lemma If \( s \equiv_{\text{flat}} t \), then for all state formulae \( \varphi \in \text{flatCTL*}(U) \), \( s \models \varphi \iff t \models \varphi \).

Proof. We prove the following two points by induction on the structure of the formula.
1. If \( s \equiv_{\text{flat}} t \), then for all state formulae \( \varphi \in \text{flatCTL*}(U) \), \( s \models \varphi \iff t \models \varphi \).
2. For paths \( \tilde{s} \) and \( \tilde{t} \): if \( \tilde{s} = \tilde{t} \), then for all path formulae \( \varphi \) that occur in flatCTL* (\( U \)) formulae, \( \tilde{s} \models \varphi \iff \tilde{t} \models \varphi \).

- Base: \( \varphi \in \text{Prop} \). \( s \equiv_{\text{flat}} t \) implies that \( \mathcal{L}(s) = \mathcal{L}(t) \). From this it follows that \( s \models p \iff t \models p \) for all \( p \in \text{Prop} \).
- Induction step:
  1. The cases that \( \varphi \) is a conjunction or negation of state or path formulae, or a state formula interpreted over a path, are straightforward.
  2. \( \varphi = U(p, \varphi') \). Assume that \( \tilde{s} \models \varphi \). By Definition 3.0.3, this means that we can choose \( k \geq 0 \) such that \( \tilde{s}(k) \models \varphi' \) and for every \( 0 \leq i < k \), \( \tilde{s}(i) \models p \). By definition of \( \tilde{s} = \tilde{t} \), there exists \( l \geq 0 \) such that \( \tilde{t}(l) = \tilde{s}(k) \) and for every \( 0 \leq j < l \), \( \tilde{t}(j) \models \varphi' \). Using the induction hypothesis, it follows that \( \tilde{t} \models \varphi \).
3. \( \varphi \equiv 3\varphi' \). Straightforward using Lemma 3.2.11.

For the other direction, abstractness, we again need to impose certain forms of finite branchingness.

3.2.13 LEMMA Assume that \( T \) is finitely branching and also finitely branching under Prop-stuttering.

If \( \forall \varphi \in \text{flat}\text{-CTL}'(U) \) \( s \models \varphi \iff t \models \varphi \), then \( s \equiv \text{flat}' \ t \).

PROOF. Assume that \( \forall \varphi \in \text{flat}\text{-CTL}'(U) \) \( s \models \varphi \iff t \models \varphi \). We have to show that \( s \equiv \text{flat}' \ t \). Because \( \equiv \text{flat}' \) is the largest flat star equivalence, we have to show that the pair \( (s, t) \) is an element of some flat star equivalence \( \equiv \subseteq \Sigma \times \Sigma \). We define this relation as follows: \( u \equiv v \) iff \( \forall \varphi \in \text{flat}\text{-CTL}'(U) \) \( u \models \varphi \iff v \models \varphi \). Clearly \( s \equiv t \). We show that \( \equiv \) is a flat star equivalence.

1. It is trivial that \( E(s) = E(t) \).

2. Suppose that \( \text{infstut} \text{-}(s) \), i.e. we can choose an \( s \)-path \( \tilde{s} \) such that for every \( i \geq 0, \tilde{s}(i) = s \). We have to show that also \( \text{infstut} \text{-}(t) \). Suppose that this is not the case. Then every \( t \)-path contains a state from the set \( T = \{ (t', t'') \mid t' \negrightarrow t'' \land t' \not\equiv t'' \} \). Because \( t \) is total (by assumption), \( T \) is nonempty. Because \( T \) is finitely branching under Prop-stuttering and also (plainly) finitely branching, this implies by Lemma 3.2.5 that \( T \) is finite, say \( T = \{ t_1, \ldots, t_n \} \). Because for every \( 1 \leq i \leq n, t_i \not\equiv t \) and \( t \equiv s \), we have \( t_i \not\equiv s \). Hence, by definition of \( \equiv \), we can choose formulae \( \varphi_i \in \text{flat}\text{-CTL}'(U) \) such that \( s \models \varphi_i \) and \( t_i \not\models \varphi_i \), for every \( 1 \leq i \leq n \). Because all states on \( s \) are \( \equiv \)-equivalent to \( s \), we have \( s \models \exists \varphi_1 \land \cdots \land \varphi_n \), but because every \( t \)-path contains some \( t_i \not\equiv \exists \varphi_1 \land \cdots \land \varphi_n \), implying that \( s \not\equiv t \), as \( \exists \varphi_1 \land \cdots \land \varphi_n \) is equivalent to a flat\text{-CTL}'(U) formula by Property 2.1.1. Contradiction.

3. Let \( \tilde{s} \) be an \( s \)-prefix. We have to show that there exists a \( t \)-prefix \( \tilde{t} \) such that:

(a) For every \( \tilde{s} \rightarrow a_0 \cdots \rightarrow \tilde{s}_{k-1} \rightarrow \tilde{s}_k \) such that \( 0 \leq k < \text{length}(\tilde{s}) \), there exists \( \tilde{t}_0 \rightarrow a_0 \cdots \rightarrow \tilde{t}_{i-1} \rightarrow \tilde{t}_i \) such that \( 0 \leq i < \text{length}(\tilde{t}) \) and \( \tilde{s}_k \equiv \tilde{t}_i \).

(b) Vice versa.

(*) Suppose that this is not the case. Consider the set \( T = \{ (t', t'') \mid t' \negrightarrow t'' \} \). Because \( t \) is total (by assumption), \( T \) is nonempty. Because \( T \) is finitely branching under Prop-stuttering and also (plainly) finitely branching, \( T \) is finite, say \( T = \{ (t_1', t_1''), \ldots, (t_n', t_n'') \} \). Let \( M \) be the largest number such that \( \tilde{s}(0) \rightarrow a_0 \cdots \rightarrow \tilde{s}(1) \rightarrow a_0 \cdots \rightarrow \tilde{s}(M-1) \rightarrow \tilde{s}(M) \). This implies that \( \tilde{s}(M-1) \not\equiv \tilde{s}(M) \). Therefore, we can choose \( p \in \text{Prop} \) such that \( \tilde{s}(M-1) \equiv p \) (and hence also \( \tilde{s}(i) \equiv p \) for every \( 0 \leq i \leq M-1 \)) and \( \exists \alpha \not\equiv p \). Furthermore, for every \( 0 \leq k \leq M \) and \( 0 \leq j \leq n \), choose formulae \( p_j \in \text{Prop} \) and \( \varphi_{k,j} \in \text{flat}\text{-CTL}'(U) \) as follows.

- \( p_j \) is true if \( t_j \equiv 0 \) \( t_j' \); otherwise, choose \( p_j \) such that \( t_j' \equiv p_j \) (and hence \( s \models p_j \)) and \( t_j' \not\equiv p_j \), which is possible by definition of \( \equiv \).

- \( \varphi_{k,j} \) is true if \( \exists \tilde{s}(k) \equiv \tilde{t}_j' \); otherwise, choose \( \varphi_{k,j} \) such that \( \exists \tilde{s}(k) \equiv \varphi_{k,j} \) and \( \tilde{t}_j' \not\equiv \varphi_{k,j} \), which is possible by definition of \( \equiv \).

For \( 0 \leq k \leq M \), define \( \psi_k = \bigcup \{ p_1 \land \cdots \land p_n \land \varphi_{k,1} \land \cdots \land \varphi_{k,n} \} \). Furthermore, for \( 0 \leq j \leq n \), define \( \xi_j = \bigcup \{ \varphi_{0,j} \land \cdots \land \varphi_{M-1,j} \land \neg p \} \). Define \( \varphi = \exists ((\land_{0 \leq j \leq M} \psi_k) \land (\land_{0 \leq j \leq M} \xi_j)) \).

Then \( s \models \varphi \), as can be seen as follows. Consider an \( s \)-path \( \tilde{s} \) that is an extension of \( \tilde{s} \) (such a path exists because \( \tilde{t} \) is total), so \( \exists \tilde{s}(k) = \tilde{s}(k) \) for every \( 0 \leq k \leq M \). First, we show that \( \exists \tilde{s}(k) \equiv \psi_k \) for every \( 0 \leq k \leq M \). Let \( 0 \leq k \leq M \). Then by definition of the \( \psi_k \), for every \( 0 \leq j \leq n \), \( \exists \tilde{s}(k) \models \varphi_{k,j} \) while for every \( 0 \leq i < k \), by definition of the \( p_j \) and by the fact that \( s \) is \( \equiv \)-equivalent to \( \exists \tilde{s}(i) \), we have \( \exists \tilde{s}(i) \models \psi_0 \land \cdots \land \psi_{M-1} \). Second, we show that \( \exists \tilde{s}(k) \equiv \xi_j \) for every \( 0 \leq j \leq n \). Let \( 0 \leq j \leq n \). By definition of \( p_j \), we have \( \exists \tilde{s}(M) \equiv \neg p_j \). Furthermore, for every \( 0 \leq k < M \), we have \( \exists \tilde{s}(k) \equiv \varphi_{k,j} \) by definition of the \( \varphi_{k,j} \) and \( \xi_j \). Our first observation is that by assumption (**), it must be
the case that for every $0 \leq k \leq M$, there exists $l \geq 0$ such that for every $0 \leq j < l$, $\tilde{t}(j) \equiv^0 t$ and $\tilde{s}(k) \equiv \tilde{t}(l)$. Namely, suppose it were not the case. Then we can choose $0 \leq k' \leq M$ such that for every $l \geq 0$, $l \geq 0$ such that $\tilde{t}(j) \not\equiv^0 t$ and $\tilde{s}(k') \equiv \tilde{t}(l)$. In the first case, $p_0 \land \cdots \land p_n$ does not hold for all $0 \leq j < l$ and in the second, $q_{k'} \land \cdots \land q_{k',n}$ does not hold in $\tilde{t}(l)$, hence $\phi_{k',1} \land \cdots \land \phi_{k',n}$ does not hold in $\tilde{t}(l)$. But that means that $\tilde{t} \not\equiv \psi$. Contradiction with assumption (**). So for every $0 \leq k \leq M$, there exists $l \geq 0$ such that for every $0 \leq j < l$, $\tilde{t}(j) \equiv^0 t$ and $\tilde{s}(k) \equiv \tilde{t}(l)$. By the definition of $M$, it is also the case for such a $k$ that for every $0 \leq i < k$, we have $\tilde{s}(i) \equiv^0 s$. So, condition 3a above holds. By assumption (*), it must then be the case that condition 3b does not hold, i.e. we can choose $l' \geq 0$ such that for every $0 \leq j < l'$, $\tilde{t}(j) \equiv^0 t$ and for every $k \geq 0$ such that $\tilde{s}(i) \equiv^0 s$ for every $0 \leq i < k$, we have $\tilde{s}(k) \not\equiv \tilde{t}(l')$. Let $N$ be the largest number such that $\tilde{t}(0) \rightarrow^{a^0} \tilde{t}(1) \rightarrow^{a^0} \cdots \rightarrow^{a^0} \tilde{t}(N-1) \rightarrow \tilde{t}(N)$. Clearly, we also have $l' \leq N$. We show that $\xi_{\ell}$ cannot hold. Namely, if it has to hold along $t$, then the eventuality $\neg \psi$ can only be fulfilled in $\tilde{t}(N)$ or beyond it — this follows from the definition of $p$, the fact that $s \equiv t$, and the definition of $N$. That means that the state formula $\phi_{0,l} \lor \cdots \lor \phi_{M-1,l}$ has to hold in $\tilde{t}(j)$ for every $0 \leq j < N$. However, it does not hold in $\tilde{t}(l')$. This follows from the definition of the $\phi_{k,j}$ and from the fact that by definition of $N$, for every $0 \leq j < N$, $\tilde{t}(j)$ is equal to some $t_k^i (0 \leq h \leq n)$.

We conclude that $\psi$ distinguishes between $s$ and $t$. By Property 2.1.1, it follows that there is also a formula $\psi' \in \text{flatCTL}^*(U)$ that distinguishes between $s$ and $t$, from which it follows that $s \not\equiv t$. Contradiction. So assumption (*) cannot be true. \hfill \Box

Note that the distinguishing formula that is constructed in this proof is not in $\text{flatCTL}(U)$: it contains several conjuncted occurrences of $U$ in the direct scope of a single $\exists$. This situation is to be contrasted with the distinctiveness results for the non-flat versions of the logics. In Lemma 3.2.6, $\text{CTL}(U)$-equivalence was a sufficient condition for $\equiv_{\text{flat}-\text{equivalence}}$. In the proof of that lemma (see [Dam96]), the constructed distinguishing formula did not need to draw on the additional expressive power of the star. The obvious question is whether we could have found such a "starless" formula in the proof above as well, by being more clever. In an attempt to answer this question, we tackle the problem from the other side: in the next subsection an adequate behavioural equivalence for $\text{flatCTL}(U)$ is defined that will be compared to $\equiv_{\text{flat}-}$.

Meanwhile, we have the following

3.2.14 Corollary Assume that $T$ is finitely branching and also finitely branching under Prop-stuttering. Then $\equiv_{\text{flat}-}$ is adequate for $\text{flatCTL}^*(U)$.

3.2.3 flatCTL(U)

In order to arrive at an adequate behavioural equivalence for $\text{flatCTL}(U)$, let us consider the game for $\text{flatCTL}(U)$. Consider states $s$ and $t$ and suppose that $s$ satisfies a $\text{flatCTL}(U)$ formula. Again, we concentrate on formulae of the form $\exists y \psi$. Then, $\psi$ is either $U(\varphi_1, \varphi_2)$ or $\neg U(\varphi_1, \varphi_2)$, where the $\varphi_i$ are state formulae again. If $t$ has to satisfy formulae of the same form, then Defender has a winning strategy to the following game:

1. Alternative 1: Attacker chooses an s-path, say $\tilde{s}$, together with a state $\tilde{s}(k)$ on $\tilde{s}$ such that for every $0 \leq i < k$, $\tilde{s}(i) \equiv^0 \tilde{s}(0)$. This should be matched by the choice by Defender of a t-path, say $\tilde{t}$, and a state $\tilde{t}(l)$ on $\tilde{t}$ such that for every $0 \leq j < l$, $\tilde{t}(j) \equiv^0 \tilde{t}(0)$, and also $\tilde{t}(l) \equiv^0 \tilde{s}(k)$. The game continues from $\tilde{s}(k)$ and $\tilde{t}(l)$.

This alternative corresponds to $\psi$ being of the form $U(\varphi_1, \varphi_2)$.

2. Alternative 2:
(a) **Phase 1**: Attacker chooses an s-path, say \( \bar{s} \), which should be matched by the choice by Defender of a t-path, say \( \bar{t} \).

This phase reflects the choice that corresponds to the \( \exists \) quantifier in the formula.

(b) **Phase 2**: Attacker chooses a state \( \bar{t}(l) \) on \( \bar{t} \) such that for every \( 0 \leq j < l \), \( \bar{t}(j) \equiv^0 \bar{t}(0) \).

Defender now has to proceed from \( \bar{s} \). She should match the move of Attacker with the choice of a position \( \bar{s}(k) \) on \( \bar{s} \) such that for every \( 0 \leq i < k \), \( \bar{s}(i) \equiv^0 \bar{s}(0) \). The game continues from \( \bar{s}(k) \) and \( \bar{t}(l) \).

This phase reflects the statement of the path formula \( \neg U(\varphi_1, \varphi_2) \). The fact that Attacker has to choose \( i \) to proceed reflects the fact that this path property occurs in negated form.

Thus, we propose the following definition of \( \equiv_{\text{flat}} \).

### 3.2.15 Definition

Let \( \equiv \subseteq \Sigma \times \Sigma \) be a symmetric relation such that for every \( s, t \in \Sigma \), \( s \equiv t \) implies:

1. \( \mathcal{L}(s) = \mathcal{L}(t) \).
2. (a) For every \( s \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} s_k \) such that \( k \geq 0 \), there exists \( t \xrightarrow{a_0} t_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{l-1}} t_l \) such that \( l \geq 0 \) and \( s_k \equiv t_l \).

(b) For every \( \bar{s} \in \text{paths}(s) \) there exists \( \bar{t} \in \text{paths}(t) \) such that: for every \( l \geq 0 \) such that \( \bar{t}(0) \xrightarrow{a_0} \bar{t}(1) \xrightarrow{a_1} \cdots \xrightarrow{a_{l-1}} \bar{t}(l-1) \xrightarrow{a_l} \bar{t}(l) \) there exists \( k \geq 0 \) such that \( \bar{s}(0) \xrightarrow{a_0} \bar{s}(1) \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} \bar{s}(k-1) \xrightarrow{a_k} \bar{s}(k) \) and \( \bar{s}(k) \equiv \bar{t}(l) \).

Then \( \equiv \) is called a flat (stuttering) equivalence. The largest flat equivalence is denoted \( \equiv_{\text{flat}} \).

The following lemma shows that this equivalence is fine enough to guarantee that equivalent states satisfy the same flat\(\text{CTL}(U)\) formulae. Note that we do not need to extend the definition of \( \equiv_{\text{flat}} \) to paths, as the inductive argument does not consider path formulae. Proofs from this subsection, being much alike those in the previous, have been moved into the appendix.

### 3.2.16 Lemma

If \( s \equiv_{\text{flat}} t \), then \( \forall \varphi \in \text{flat\(\text{CTL}(U)\)} \ s \models \varphi \iff t \models \varphi \).

Reversely, flat\(\text{CTL}(U)\) can distinguish any two states that are not \( \equiv_{\text{flat}} \)-equivalent. This follows from the following abstractness result.

### 3.2.17 Lemma

Assume that \( T \) is finitely branching and also finitely branching under Prop-stuttering. If \( \forall \varphi \in \text{flat\(\text{CTL}(U)\)} \ s \models \varphi \iff t \models \varphi \), then \( s \equiv_{\text{flat}} t \).

### 3.2.18 Corollary

Assume that \( T \) is finitely branching and also finitely branching under Prop-stuttering. Then \( \equiv_{\text{flat}} \) is adequate for flat\(\text{CTL}(U)\).

### 3.2.4 Separating the distinguishing powers of flat\(\text{CTL}^*(U)\) and flat\(\text{CTL}(U)\)

Towards the end of Section 3.2.2 we raised the question whether two states that are not \( \equiv_{\text{flat}} \)-equivalent can be distinguishing by a flat formula without drawing on the power of the star. Here, we will show that this is not the case: we present a finite Kripke structure in which states \( s \) and \( t \) are distinguishable by a flat\(\text{CTL}^*(U)\) formula (and hence, by Lemma 3.2.12, not \( \equiv_{\text{flat}} \)-equivalent), but \( s \equiv_{\text{flat}} t \) (and hence, by Lemma 3.2.16, not distinguishable by any flat\(\text{CTL}(U)\) formula).
The formula in flatCTL*(U) that distinguishes $s$ from $t$ is $\varphi = 3(-U(\neg q, \neg 3U(p, r) \land \neg q) \land U(true, q))$. In order to see this, we first translate $\varphi$ into an equivalent CTL(U) formula, as follows.

$$\exists(-U(\neg q, \neg 3U(p, r) \land \neg q) \land U(true, q))$$

$$\equiv$$

$$\neg \forall(U(\neg q, \neg 3U(p, r) \land \neg q) \lor G\neg q)$$

$$\equiv$$

$$\neg \forall(W(\neg q, \neg 3U(p, r) \land \neg q)$$

$$\equiv$$

$$\forall(W(\neg q, \exists U(p, r), \neg q)$$

$$\equiv$$

$$\exists U(\exists U(p, r), q)$$

So, $\varphi$ expresses the property that there exists a path along which eventually $q$ holds while in all states before that, there exists the possibility to reach an $r$-state via $p$-states only. It can easily be checked that there exists such a path starting from $s$, namely $s, s_3, s_4, s_5, \ldots$. On the other hand, the only path from $t$ that eventually hits a $q$-state is $t, t_1, t_2, t_3, \ldots$, but from $t_1$ there is no possibility to reach $r$ anymore. So, $s \models \varphi$ while $t \not\models \varphi$, and hence $s \not\equiv_{flat} t$.

Next, we show that $s \equiv_{flat} t$ by giving an equivalence relation $\equiv$ on states that satisfies Definition 3.2.15. $\equiv$ consists of the following pairs: \{(s, t), (s_1, t_1), (s_2, t_2), (s_3, t), (s_4, t_2), (s_5, t_3), (s_6, t_5)\}. The conditions of Definition 3.2.15 are obviously satisfied for the pairs (s_1, t_1), (s_2, t_2), (s_4, t_2), (s_5, t_3), and (s_6, t_5), as the corresponding subtrees are isomorphic for each pair. Next, note that $s$ and $s_3$ are bisimilar. We show that the pair ($s, t$) satisfies the conditions of Definition 3.2.15, then this follows for ($s_3, t$) as well. As for ($s, t$), it is clear that for every prefix or path that can be taken starting from $t$, there exists a corresponding (in the sense of conditions 2a and 2b of Definition 3.2.15) prefix or path starting from $s$. Conversely, starting from $s$, the only non-obvious cases are the prefixes and paths going via $s, s_3, s_4$. If we take any such prefix that ends in $s_4$, then the prefix $t, t_1, t_2$ matches it in the sense of condition 2a (note that $s_3 \equiv 0 t_1$). If we take the (infinite) path $s, s_3, s_4, s_5, \ldots$, then the matching path in the sense of condition 2b is the path that keeps cycling in $t$ forever.

Thus, unlike the cases CTL*/CTL and CTL*(U)/CTL(U), where in both cases the starred and unstarred versions had the same distinguishing powers, we have now identified a fragment for which

![Figure 2: $s \not\equiv_{flat} t$ but $s \equiv_{flat} t$.](image-url)
these distinguishing powers are different. Other interesting comparisons are between flatCTL*(U) and CTL*(U), and between flatCTL(U) and CTL(U). In the next subsection, we consider the first of these.

3.2.5 Separating the distinguishing powers of flatCTL*(U) and CTL*(U): a surprise

It is not possible to separate $\equiv_{\text{stat}}$ from $\equiv_{\text{flat}}$: the following lemma says that they indeed coincide.

3.2.19 LEMMA $\equiv_{\text{stat}} = \equiv_{\text{flat}}$.

PROOF Because $\equiv_{\text{stat}}$ is adequate for CTL*(U), $\equiv_{\text{flat}}$ is adequate for flatCTL*(U), and flatCTL*(U) $\subseteq$ CTL*(U), we clearly have $\equiv_{\text{stat}} \subseteq \equiv_{\text{flat}}$. In order to prove $\equiv_{\text{stat}} \supseteq \equiv_{\text{flat}}$, we have to show that $\equiv_{\text{flat}}$ satisfies the conditions in Definition 3.2.2. Points 1 and 2 are easy. As to point 3, assume that $s \rightarrow a s \rightarrow s_1 \rightarrow a s_1 \rightarrow \ldots \rightarrow a s_{k-1} \rightarrow s_k$ such that $k \geq 0$. By point 3 in Definition 3.2.9 of $\equiv_{\text{flat}}$, it is easy to see that there exists $l \rightarrow a^i \rightarrow t_1 \rightarrow a^j \rightarrow \ldots \rightarrow a^i t_{l-1} \rightarrow t_l$ such that $l \geq 0$ and $s_k \equiv_{\text{flat}} t_l$. Next, we show that any two states $t_j$ and $t_j'$ with $0 \leq j \leq j' < l$ are $\equiv_{\text{stat}}$-equivalent. Let $0 \leq j \leq j' < l$. By point 3 in Definition 3.2.9 of $\equiv_{\text{flat}}$, we can choose $0 \leq i \leq l < k$ such that $t_j \equiv_{\text{stat}} s_i$ and $t_j' \equiv_{\text{stat}} s_{i'}$. By definition, $s_i \equiv_{\text{flat}} s_{i'}$. So $t_j \equiv_{\text{flat}} t_j'$. □

4 Conclusions

We have investigated the effects of flattening on the expressivity and distinctiveness of temporal logics. In Section 2 we have demonstrated an LTL(U) formula that cannot be expressed by a flat equivalent. Furthermore, we have argued that this witness formula is the shortest in terms of the number of Until operators, thereby quantifying the loss of expressivity that is caused by flattening. We expect that the restriction to the flat fragment will have no repercussions on the use of temporal logic as a specification formalism: a single Until operator nested in the initial invariant of another can still be rewritten into flat form, and more will hardly ever be needed in practice.

These results about expressivity carry over to the branching time logics CTL*(U) and CTL(U) and their flat versions, as is shown in Section 3. Figure 3 summarises the results. Relation 1 was established in [EH86] while 4 and 7 can be proven in a similar fashion. The vertical relations 2 and 3 are obvious: without Next operator one cannot count individual states. In this article, we established 5 and 6.

\[
\begin{array}{c c}
\text{CTL*} & \overset{(1)}{\geq} & \text{CTL} \\
\overset{(2)}{\{\forall} & \overset{(3)}{\{\forall} \\
\text{CTL*(U)} & \overset{(4)}{\geq} & \text{CTL(U)} \\
\overset{(5)}{\{\forall} & \overset{(6)}{\{\forall} \\
\text{flatCTL*(U)} & \overset{(7)}{\geq} & \text{flatCTL(U)}
\end{array}
\]

Figure 3: Expressive powers

Most of Section 3 is devoted to an investigation of adequate behavioural equivalences for flatCTL*(U) and flatCTL(U), and their comparison. It turns out that the pattern that applies for the comparisons CTL* vs. CTL and CTL*(U) vs. CTL(U), namely that the star affects the expressive but not the distinguishing powers, breaks down in the case of the flat versions: we have shown an example of two states that are distinguished by a flatCTL*(U) formula, but that are equivalent under the equivalence induced by flatCTL(U). Indeed, it turns out that flatCTL*(U) has the same distinguishing power as
Figure 4 puts these results into perspective. The relations 1' and 4' are based on results from [BCG88], [DNV90] and [Dam96]. The vertical relations 2' and 3' are obvious again. The somewhat surprising 5' and 7' were proven in Section 3 of this article, and 6' follows from these and 4'. The adequate behavioural equivalences are shown alongside. Note that the logics are interpreted over possibly infinite Kripke structures with certain restrictions on the branching degree.

\[
\begin{align*}
\equiv_{\text{bis}} & \quad \text{CTL}^* (U) \quad (1') \quad \equiv_{\text{bis}} \\
\equiv_{\text{stat}} & \quad \text{CTL}^*(U) \quad (2') \quad \equiv_{\text{stat}} \\
\equiv_{\text{flat}} & \quad \text{flatCTL}^*(U) \quad (7') \quad \equiv_{\text{flat}}
\end{align*}
\]

Figure 4: Distinguishing powers

References


18
A Proofs of Section 2

PROOF OF LEMMA 2.1.6. By induction on the structure of \( \varphi \). The base case is easy. As to the induction step, we concentrate on the case that \( \varphi = U(\varphi', \varphi'' \rangle \); the other cases are straightforward (recall that \( \varphi \in \text{flatLTL}(U) \) and hence cannot contain \( \times \)'s). Let \( i > U\text{depth}(\varphi) \) (so \( i \geq 2 \)). We consider separately each of the three cases of the lemma.

1. Let \( 0 \leq k < |A| \).

\( \Rightarrow \) direction. Assume that \( Y_{i+1}(k, \ldots ) \models \varphi \). By definition of satisfaction, this means that we can choose \( l \geq 0 \) such that \( Y_{i+1}(k, \ldots )(l, \ldots ) = \varphi' \) and that for every \( 0 \leq j < l \), \( Y_{i+1}(k, \ldots )(j) \models \varphi' \). We consider the following cases, distinguishing in which part of \( Y_{i+1} \) the eventuality is fulfilled.

(a) \( k + l < |A| \). Then by point 1 of the i.h., we have \( Y_{i}(k, \ldots )(l, \ldots ) \models \varphi' \). Because \( k + l < |A| \) and \( i \geq 1 \), it easily follows that \( Y_{i}(k, \ldots )(j) = Y_{i+1}(k, \ldots )(j) \) for every \( 0 \leq j < l \). Therefore we also have \( Y_{i}(k, \ldots )(j) \models \varphi' \) for every \( 0 \leq j < l \). So \( Y_{i+1}(k, \ldots ) \models \varphi' \).

(b) \( |A| \leq k + l < |A| \). Then by point 2 of the i.h., we have \( Y_{i}(k+l-|A|, \ldots ) \models \varphi' \). So \( \varphi' \) holds in some state of the first A-block of \( Y_{i} \). However, this does not necessarily mean that along \( Y_{i}(k, \ldots ) \) the eventuality \( \varphi' \) is fulfilled: namely, it may be the case that \( k+l-|A| < k \), i.e. the state of \( Y_{i} \) where \( \varphi' \) holds comes before \( Y_{i}(k) \). Therefore, we “shift \( \varphi' \) forward” by one A-block using Lemma 2.0.4. By that lemma, (note that \( i - 1 > U\text{depth}(\varphi') \)), we have that also \( Y_{i}(k+l, \ldots ) \models \varphi' \). Because \( k + l < |A| \) and \( i \geq 2 \), it easily follows that \( Y_{i}(k, \ldots , k+l-1) = Y_{i+1}(k, \ldots , k+l-1) \). Therefore we also have \( Y_{i}(k, \ldots )(j) \models \varphi' \) for every \( 0 \leq j < l \). So \( Y_{i+1}(k, \ldots ) \models \varphi' \).

(c) \( |A| \leq k+l < |A| \). Then by point 2 of the i.h., we have \( Y_{i}(k+l-|A|, \ldots ) \models \varphi' \). In contrast to the previous case, we do not have to shift \( \varphi' \) forward as \( k+l-|A| \) is guaranteed to be greater than \( k \) in this case. As for the intermediate states, we have by Property 2.1.3 that \( Y_{i}(k, \ldots )(j) \models \varphi' \) for every \( 0 \leq j < l \). So \( Y_{i+1}(k, \ldots ) \models \varphi' \).

\( \Leftarrow \) direction.

(a) \( k+l < |A| \). Use point 1 of the i.h.

(b) \( |A| \leq k+l < |A| \). Use point 2 of the i.h. and Lemma 2.0.4 to shift \( \varphi' \) backward from the third to the second A-block of \( Y_{i+1} \).

(c) \( |AA| \leq k+l \). By choosing \( \ell := |X_{i+1}|+l \), we have \( Y_{i+1}(k, \ldots ) U\ell = \varphi' \). By Property 2.1.3 it follows that \( Y_{i+1}(k, \ldots )(j) \models \varphi' \) for every \( 0 \leq j < \ell \). (In the previous two cases we could not use this "direct" argument because Property 2.1.3 could not be applied there.)

2. Let \( |A| \leq k < |A| \).

\( \Rightarrow \) direction. Assume that \( Y_{i+1}(k, \ldots ) \models \varphi \). \( l \) is defined in a similar way as in the previous case.

(a) \( |A| \leq k+l < |A| \). Use point 2 of the i.h.

(b) \( |A| \leq k+l < |A| \). Use point 3 of the i.h. and Property 2.1.3.

(c) \( |A| \leq k+l < |A| \). Use point 1 of the i.h. for \( i - 1 \) (to shift \( \varphi' \) forward to the first A-block of \( Y_{i-1} \)) and Property 2.1.3.
(d) \( |A^{i+1} B A^{i+1} C A| \leq k + l \). If \( k + l - |X_{i+1}| < k \), i.e. the state of \( Y_i \) where \( \varphi' \) holds comes before \( Y_i(k) \), then use point 2 of the i.h. for \( i-1 \) (to shift \( \varphi' \) forward to \( Y_{i-1} \)) and \( \varphi' \) and Property 2.1.3. Otherwise, it is "direct", only using Property 2.1.3.

\[ \Rightarrow \text{direction. Straightforward by now.} \]

3. \( |A^{i+1} B A| \leq k < |A^{i+1} B A^{i+1} C| \). Similar to the previous case. \( \square \)

**Proof of Lemma 2.1.7** By induction on the structure of \( \varphi \). The base case is easy. As to the induction step, we concentrate on the case that \( \varphi = U(q', \varphi') \); the other cases are straightforward (recall that \( \varphi \in \text{flatLTL}(U) \) and hence cannot contain \( X \)'s). Let \( 1 \leq h \leq i + 1 \) with \( h > Udepth(\varphi) \) (so \( h \geq 2 \)). Let \( 0 \leq k < |A| \).

\( \Rightarrow \text{direction. Assume that } (A^i B A^{i+1} C Y_i)(k, \ldots)(j, \ldots) \models \varphi \) and for every \( 0 \leq j < l \), \( (A^i B A^{i+1} CY_i)(k, \ldots)(j) \models \varphi \). We distinguish the following cases.

1. \( k + l < |A| \). Then by the i.h., \( (A^i C Y_i)(k, \ldots)(j, \ldots) \models \varphi \). Because \( k + l < |A| \) and \( h \geq 1 \), \( (A^i C Y_i)(k, \ldots)(j) = (A^i B A^{i+1} CY_i)(k, \ldots)(j) \) for every \( 0 \leq j < l \). Therefore, \( (A^i C Y_i)(k, \ldots)(j) \models \varphi \) for every such \( j \). So \( (A^i C Y_i)(k, \ldots) \models \varphi \).

2. \( |A| \leq k + l < |A| \). Then \( (A^i B A^{i+1} CY_i)(k - |A|, \ldots)(l, \ldots) \models \varphi \) and therefore, by the i.h. (note that \( h - 1 > Udepth(\varphi') \) and \( k + l - |A| < |A| \)), \( (A^{i-1} B A^{i+1} CY_i)(k - |A|, \ldots)(l, \ldots) \models \varphi' \) and thus \( (A^i C Y_i)(k, \ldots)(l, \ldots) \models \varphi \). Similar as in the previous case we have \( (A^i C Y_i)(k, \ldots)(j) \models \varphi \) for every \( 0 \leq j < l \). So \( (A^i C Y_i)(k, \ldots) \models \varphi \).

3. \( |AA| \leq k + l < |A^{i+1} B A| \). In this case we have by Lemma 2.1.6, point 2, that the corresponding position of \( Y_i \) also satisfies \( \varphi' \): \( Y_i((i - h) \cdot |A| + k + l - |A|) \models \varphi' \). So, we can choose \( t' \geq 0 \) such that \( (A^i C Y_i)(k, \ldots)(l', \ldots) \models \varphi' \), namely \( t' = |A^{i+1} C| + (i - h) \cdot |A| + l - |A| \). Using Property 2.1.3, we also have that \( (A^i C Y_i)(k, \ldots)(j) \models \varphi \) for every \( 0 \leq j < t' \). So \( (A^i C Y_i)(k, \ldots) \models \varphi \).

4. \( |A^{i+1} B A| \leq k + l < |A^{i+1} B A^{i+1} C| \). In this case we have by Lemma 2.1.6, point 3, that the corresponding position of \( Y_i \) also satisfies \( \varphi' \): \( Y_i((i - h) \cdot |A| + k + l - |AA|) \models \varphi' \). So, we can choose \( t' \geq 0 \) such that \( (A^i C Y_i)(k, \ldots)(l', \ldots) \models \varphi' \), namely \( t' = |A^{i+1} C| + (i - h) \cdot |A| + l - |AA| \). Using Property 2.1.3, we also have that \( (A^i C Y_i)(k, \ldots)(j) \models \varphi \) for every \( 0 \leq j < t' \). So \( (A^i C Y_i)(k, \ldots) \models \varphi \).

5. \( |A^{i+1} B A^{i+1} C| \leq k + l \). "Direct", only using property Property 2.1.3.

\( \Leftarrow \text{direction. The cases } k + l < |A| \text{ and } |AA| \leq k + l < |A^{i+1} B A| \text{ are similar to the corresponding cases for the } \Rightarrow \text{direction. The case } |AA| \leq k + l \text{ is "direct", only using Property 2.1.3.} \ \square \)

**B Proofs of Section 3**

**Proof of Lemma 3.2.16.** By induction on the structure of the formula.

- **Base:** \( \varphi \in \text{Prop.} \) \( s \models_{\text{the}} t \) implies that \( L(s) = L(t) \). From this it follows that \( s \models t \) iff \( t \models p \) for all \( p \in \text{Prop.} \).
- **Induction step:**
  1. The cases that \( \varphi \) is a negation or conjunction are straightforward.
  2. \( \varphi = \exists U(p, \varphi') \). Assume that \( s \models \varphi \). By Definition 3.0.3, this means that we can choose an \( s \)-path \( \bar{s} \) and \( n \geq 0 \) such that \( \bar{s}(n) \models \varphi' \) and for every \( 0 \leq i < n \), \( \bar{s}(i) \models p \). From \( s \models_{\text{the}} t \) and clause 2a in Definition 3.2.15, we can prove that there exists an \( s \)-path \( \bar{t} \) and \( m \geq 0 \) such that \( \bar{s}(m) \models_{\text{the}} \bar{t}(m) \) and for every \( 0 \leq j < m \), there exists \( 0 \leq i < n \) such that \( \bar{t}(j) \equiv_{\text{the}} \bar{s}(i) \). By the induction hypothesis, it follows from \( \bar{s}(n) \models \varphi' \) and \( \bar{s}(i) \models_{\text{the}} \bar{t}(m) \) that \( \bar{t}(m) \models \varphi' \). Because for every \( 0 \leq j < m \), \( \bar{t}(j) \) is \( \equiv_{\text{the}} \)-equivalent to some \( \bar{s}(i) \) with \( 0 \leq i < n \), and for every such \( i \) we have \( \bar{s}(i) \models p \), we also have \( \bar{t}(j) \models p \) for every \( 0 \leq j < m \). Hence, \( t \models \exists U(p, \varphi') \).
3. \( \varphi = \exists \neg U(p, \varphi) \). Assume that \( s \models \varphi \). By Definition 3.0.3, this means that we can choose an \( s \)-path \( \tilde{s} \) such that \( \tilde{s} \not\models U(p, \varphi) \). We have to show that there exists a \( t \)-path \( \tilde{t} \) such that \( \tilde{t} \not\models U(p, \varphi) \).

Suppose \( \neg \varphi \). This is not the case; we will derive a contradiction. Because \( s \equiv_{\text{flat}} t \), by clause 2b in Definition 3.2.15, we can choose a \( t \)-path \( \tilde{t} \) such that: for every \( l \geq 0 \) such that \( \tilde{t}(0) \rightarrow e_0 \rightarrow \tilde{t}(1) \rightarrow e_1 \rightarrow \cdots \rightarrow e_{l-1} \rightarrow \tilde{t}(l) \) there exists \( k \geq 0 \) such that \( \tilde{s}(0) \rightarrow e_0 \rightarrow \tilde{s}(1) \rightarrow e_1 \rightarrow \cdots \rightarrow e_{k-1} \rightarrow \tilde{s}(k) \) and \( \tilde{s}(k) \equiv \tilde{t}(l) \). By our assumption, \( \tilde{t} \models U(p, \varphi) \), i.e., we can choose \( m \geq 0 \) such that \( \tilde{t}(m) \models \varphi \) and for all \( 0 \leq j < m \), \( \tilde{t}(j) \models \varphi \). From clause 2b in Definition 3.2.15, we can prove that we can choose \( k' \geq 0 \) such that \( \tilde{s}(k') \equiv_{\text{flat}} \tilde{t}(m) \) and for every \( 0 \leq i < k' \), there exists \( 0 \leq j < m \) such that \( \tilde{s}(i) \equiv \tilde{t}(j) \). By the induction hypothesis, it follows from \( \tilde{t}(m) \models \varphi \) and \( \tilde{s}(k') \equiv_{\text{flat}} \tilde{t}(m) \) that \( \tilde{s}(k') \models \varphi \). Because for every \( 0 \leq i < k', \tilde{s}(i) \equiv_{\text{flat}} \tilde{t}(j) \), we have that \( \varphi \).

Hence, \( \tilde{s} \equiv_{\text{flat}} U(p, \varphi) \). Contradiction.

Theorem 3.2.17. Assume that \( \forall \varphi \in \text{flatCtL}(U) \) \( s \equiv_{\text{flat}} t \). We have to show that there exists a \( s \)-path \( \tilde{s} \) such that \( \tilde{s} \not\models \varphi \). We have to show that \( \exists \neg \varphi \). Assume that \( \varphi \). By Definition 3.0.3, this means that we can choose an \( s \)-path \( \tilde{s} \) such that \( \tilde{s} \not\models U(p, \varphi) \). We have to show that there exists a \( t \)-path \( \tilde{t} \) such that \( \tilde{t} \not\models U(p, \varphi) \).

Suppose \( \neg \varphi \). This is not the case; we will derive a contradiction. Because \( s \equiv_{\text{flat}} t \), by clause 2b in Definition 3.2.15, we can choose a \( t \)-path \( \tilde{t} \) such that: for every \( l \geq 0 \) such that \( \tilde{t}(0) \rightarrow e_0 \rightarrow \tilde{t}(1) \rightarrow e_1 \rightarrow \cdots \rightarrow e_{l-1} \rightarrow \tilde{t}(l) \) there exists \( k \geq 0 \) such that \( \tilde{s}(0) \rightarrow e_0 \rightarrow \tilde{s}(1) \rightarrow e_1 \rightarrow \cdots \rightarrow e_{k-1} \rightarrow \tilde{s}(k) \) and \( \tilde{s}(k) \equiv \tilde{t}(l) \). By our assumption, \( \tilde{t} \models U(p, \varphi) \), i.e., we can choose \( m \geq 0 \) such that \( \tilde{t}(m) \models \varphi \) and for all \( 0 \leq j < m \), \( \tilde{t}(j) \models \varphi \). From clause 2b in Definition 3.2.15, we can prove that we can choose \( k' \geq 0 \) such that \( \tilde{s}(k') \equiv_{\text{flat}} \tilde{t}(m) \) and for every \( 0 \leq i < k' \), there exists \( 0 \leq j < m \) such that \( \tilde{s}(i) \equiv \tilde{t}(j) \). By the induction hypothesis, it follows from \( \tilde{t}(m) \models \varphi \) and \( \tilde{s}(k') \equiv_{\text{flat}} \tilde{t}(m) \) that \( \tilde{s}(k') \models \varphi \). Because for every \( 0 \leq i < k', \tilde{s}(i) \equiv_{\text{flat}} \tilde{t}(j) \), we have that \( \varphi \).

Hence, \( \tilde{s} \equiv_{\text{flat}} U(p, \varphi) \). Contradiction.
Define $\varphi = \exists t \subseteq U(p_1 \land \cdots \land p_n, \varphi_1 \lor \cdots \lor \varphi_n)$. Then, by the definition of the $p_i$ and $\varphi_i$, we have for every $t \in \text{paths}(U): \bar{t} \models U(p_1 \land \cdots \land p_n, \varphi_1 \lor \cdots \lor \varphi_n)$, and thus $t \not\models \varphi$. Next, we show that $s \models \varphi$, by showing that $\bar{s} \models \neg U(p_1 \land \cdots \land p_n, \varphi_1 \lor \cdots \lor \varphi_n)$. Namely, suppose that, conversely, $\bar{s} \models U(p_1 \land \cdots \land p_n, \varphi_1 \lor \cdots \lor \varphi_n)$, i.e. we can choose $k \geq 0$ and $1 \leq i \leq n$ such that $\bar{s}(k) \models \varphi_i$ while for every $0 \leq i' < k$, $\bar{s}(i') \not\models p_1 \land \cdots \land p_n$. But the fact that $\bar{s}(k) \models \varphi_i$ means, by definition of the $\varphi_i$, that it cannot be the case that $s(0) \rightarrow s(1) \rightarrow \cdots \rightarrow s(k - 1) \rightarrow s(k)$. Hence, by the definition of the $p_i$, there must be $0 \leq i \leq k - 2$ such that $\bar{s}(i + 1) \not\models \varphi_i$. This contradicts the fact that for every $0 \leq i' < k$, $\bar{s}(i') \models p_1 \land \cdots \land p_n$. Therefore, we conclude that $\bar{s} \models \neg U(p_1 \land \cdots \land p_n, \varphi_1 \lor \cdots \lor \varphi_n)$, i.e., $s \models \varphi$. But then $s \not\models t$, as $\varphi \in \text{flatCTL}(U)$. Contradiction. \qed
<table>
<thead>
<tr>
<th>Computing Science Reports</th>
<th>Department of Mathematics and Computing Science</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In this series appeared:</strong></td>
<td><strong>Eindhoven University of Technology</strong></td>
</tr>
<tr>
<td>96/01 M. Voorhoeve and T. Basten</td>
<td>Process Algebra with Autonomous Actions, p. 12.</td>
</tr>
<tr>
<td>96/02 P. de Bra and A. Aerts</td>
<td>Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12.</td>
</tr>
<tr>
<td>96/04 S. Mauw</td>
<td>Example specifications in phi-SDL.</td>
</tr>
<tr>
<td>96/06 W.M.P. van der Aalst and T. Basten</td>
<td>Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.</td>
</tr>
<tr>
<td>96/07 M. Voorhoeve</td>
<td>Structural Petri Net Equivalence, p. 16.</td>
</tr>
<tr>
<td>96/11 T. Laan</td>
<td>AUTOMATH and Pure Type Systems, p. 30.</td>
</tr>
<tr>
<td>96/12 F. Kamareddine and T. Laan</td>
<td>A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.</td>
</tr>
<tr>
<td>96/13 T. Borghuis</td>
<td>Priorlean Tense Logics in Modal Pure Type Systems, p. 61.</td>
</tr>
<tr>
<td>96/15 M.A. Reniers and J.J. Vereijken</td>
<td>Completeness in Discrete-Time Process Algebra, p. 139.</td>
</tr>
<tr>
<td>96/17 E. Boiten and P. Hoogendijk</td>
<td>Nested collections and polymorphism, p. 11.</td>
</tr>
<tr>
<td>96/18 P.D.V. van der Stok</td>
<td>Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.</td>
</tr>
<tr>
<td>96/19 M.A. Reniers</td>
<td>Static Semantics of Message Sequence Charts, p. 71.</td>
</tr>
<tr>
<td>96/20 L. Feijs</td>
<td>Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.</td>
</tr>
<tr>
<td>96/23 W.M.P. van der Aalst</td>
<td>Structural Characterizations of sound workflow nets, p. 22.</td>
</tr>
<tr>
<td>96/24 M. Voorhoeve and W. van der Aalst</td>
<td>Conservative Adaption of Workflow, p.22</td>
</tr>
<tr>
<td>96/25 M. Vaccari and R.C. Backhouse</td>
<td>Deriving a systolic regular language recognizer, p. 28.</td>
</tr>
<tr>
<td>97/06 M. Franissen</td>
<td>Tools for the Construction of Correct Programs: an Overview, p. 33.</td>
</tr>
<tr>
<td>97/07 J.C.M. Baeten and J.A. Bergstra</td>
<td>Bounded Stacks, Bars and Queues, p. 15.</td>
</tr>
</tbody>
</table>
97/08  P. Hoogendijk and R.C. Backhouse  When do datatypes commute? p. 35.
97/13  W.M.P. van der Aalst  Exploring the Process Dimension of Workflow Management, p. 56.
97/14  J.F. Groote, F. Monin and J. Springintveld  A computer checked algebraic verification of a distributed summation algorithm, p. 28.
97/15  M. Franssen  2P: A Pure Type System for First Order Logic with Automated Theorem Proving, p. 35.
97/16  W.M.P. van der Aalst  On the verification of inter-organizational workflows, p. 23.
97/17  M. Vaccari and R.C. Backhouse  Calculating a Round-Robin Scheduler, p. 23.
98/02  M. Voorhoeve  State / Event Net Equivalence.
98/03  J.C.M. Baeten and J.A. Bergstra  Deadlock Behaviour in Split and ST Bisimulation Semantics, p. 15.